GP AND GP-FEA MULTISCALE MODELING: MODEL SIZE EFFECTS AND APPLICATIONS

BY

ROSS J. STEWART

A THESIS
SUBMITTED TO THE FACULTY OF

ALFRED UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE

IN

MECHANICAL ENGINEERING

ALFRED, NEW YORK

APRIL, 2015

GP AND GP-FEA MULTISCALE MODELING: MODEL SIZE EFFECTS AND APPLICATIONS

BY

ROSS J. STEWART

B.S. ALFRED UNIVERSITY (2010)

SIGNATURE OF AUTHOR_____

APPROVED BY_____

JINGHONG FAN, ADVISOR

_____

S. K. SUNDARAM, ADVISORY COMMITTEE

_____

YIQUAN WU, ADVISORY COMMITTEE

_____

CHAIR, ORAL THESIS DEFENSE

ACCEPTED BY_____

DOREEN D. EDWARDS, DEAN
KAZUO INAMORI SCHOOL OF ENGINEERING

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Jinghong Fan, for his guidance, advice and creating an environment for me to learn and grow into a professional. Thanks for his patience, diligence and persistent lifestyle, which has lead me to push myself forward and to raise my scientific and professional level.

I want to thank my past and present colleagues and friends: Christopher Powell, Long He, Rongpeng Xu, Steve Gruspier, Taolong Xu, and Nicholas Fuller for their assistance, support and stimulating discussion.

I would also like to thank my colleagues and superiors at Corning Inc. for their patience, support, and source of alternative perspectives which has not only enriched my own education but allowed me to step out of academia and into corporate research.

Finally, I want express my thanks to my parents Mr. Michael Stewart and Mrs. Lynn Stewart, and my sister Mrs. Gwen Gagnon for their unwavering patience, love and support.

TABLE OF CONTENTS

Page

Acknowledgments iii

Table of Contents iv

List of Tables ix

List of Figures x

Abstract xiv

I. INTRODUCTION 1

A. Motivation 1

B. Classification of Multiscale methods 3

1. Hierarchical 4

2. Concurrent 5

C. Atomistic-based multiscale analysis, Class-I 7

D. Inadequacies and Existing Needs 8

II. METHODOLOGY OF THE ATOMISTIC-BASED MULTISCALE ANALYSIS 10

A. The Generalized Particle (GP) Method 10

1. Material property scale independence 12

2. Scale interfaces 14

3. Surface corrections 17

4. Scale duality 20

a. Decomposition 20

b. Dislocation propagation beyond scale boundary 23

c. Automatic Decomposition 25

B. Linking GP with FEA 29

1. Advantages of the GP-FEA Methodology 29

2. Model Structure and Design of Coupling subsystems 30

3. The “Bottom-Up” and “Top-Down” iteration bridging scheme 31

4. Numerical Algorithms 34

C. Shortcomings 36

D. Summary 36

III. ACCURACY VERIFICATIONS OF ATOMISTICALLY-BASED MULTISCALE ANALYSIS WITH CONTINUUM SOLUTIONS BY THE GP-FEA METHODS 37

A. Introduction 37
B. Model Development 42
1. GP Hole Model development 42
2. GP--FEA interface design 43
3. FEA Mesh generation 45
C. Verification of the GP and GP-FEA multiscale methods with Elasticity Solutions for a plate with a central hole under tensile loading 46
1. The scheme for the comparison of atomistically- based simulation with a continuum solution 46
2. Result comparison for pure GP and GP-FEA model 48
D. Summary and Conclusions 50
IV. ACCURACY VALIDATION WITH LEFM SOLUTION AT CRACK-TIPS: CONCEPT OF CRITICAL MODEL SIZE 53
A. Introduction 53
B. GP-FEA Model Design 61
C. Comparison between LEFM solutions and GP-FEA simulation results for the crack-tip displacement 63
1. LEFM singularity solution and two-term solutions of crack-tip displacement field 63
2. Comparison between atomistically-based multiscale simulation with both LEFM singularity and two-term solutions 66
3. Model size effects on numerical results of displacement distribution near crack-tip in comparison with LEFM two-term solutions 69
D. Summary and discussions 72
V. APPLICATION IN CRACK PROPAGATION 75
A. Introduction 75
B. Simulation Model and Methods 77
C. Crack Propagation Results via GP-FEA 78
1. Fracture Energy 79
2. Crack-Tip Phase Transformation 83
D. Conclusion 86
VI. APPLICATIONS IN IMPACT 87
A. Introduction 87
B. GP Model for Wave Propagation 89
C. MD Results for Wave Propagation 91
D. Scale-2 Results for Wave Propagation 92
E. Auto-Duality Results for Wave Propagation 95

F. Summary and Recommendations 97
VII. PARTICLE-BASED MULTISCALE ANALYSIS PROGRAM (PMAP) STRUCTURE 99
A. Introduction 99
B. Functionality, constitution and flow charts of three basic processes of PMAP 101
1. Initialization Process 102
2. Equilibration Process 105
3. Loading Process 107
C. Constructions and Executions 109
1. Post-processing 110
D. Summary 110
VIII. CONCLUSIONS AND RECOMMENDATIONS 111
A. Conclusions 112
B. Recommendations for Future Work 115
REFERENCES 118
APPENDICES 130
A. USUAL MOLECULAR DYNAMIC FEATURES 131
1. Reading EAM Tables 131
2. Damped Shifted Coulomb Potential (DSC) 132
a. Implementation 133
3. Simulation Revival 134
4. Periodic Boundary Conditions (PBC) using Verlet Neighbor Lists 135
5. Barostat (NPT) 138
a. Berendsen Barostat 138
b. Anisotropic barostat 139
B. GP FEATURES 142
1. Automatic Duality Domains (ADD) with stress and energy (internal duality) 142
a. Preparation 143
i. Special List 143
ii. NLC Linking 143
iii. NLC Breakage 144
b. Detector and Process 145
i. Detector 145
ii. Process 146

2. Link GP with FEA 146
a. FEA-GP connexion 147
3. Using local/scale BoxSizes for Verlet neighbor Lists to save memory in large models 152
C. MODEL DEVELOPMENT 153
1. GP model development program 153
a. Auto-Scale Interface Creation 158
2. Nano-structure generation 159
3. External Decomposition (blind) versions: 2W 3, 5, 6one 162
a. decomp2W.f90 162
b. decomp3.f90 163
c. decomp5.f90 163
d. decomp6one.f90 165
4. Delete duplicate positions within a cutoff: delMDdup & delMD3dup 177
5. Considerations when designing very large micron scale models 177
6. FE Mesh input file for the GP-FEA simulation. 178
D. THE PARTICLE-BASED MULTISCALE ANALYSIS PROGRAM (PMAP) 181
1. PMAP Structure: Subroutines and functions 181
a. Compiling and running 186
2. PMAP Input file 186
a. Examples 192
E. DATA PROCESSING 197
1. Make XYZ files from CONFIG, MD, MD3 files: mkxyz.exe 202
2. Make Model.MD file from MD3 or Revive.MD file: mkMD.sh 203
3. Extract a specific scale or local domain from MD3 files: getScale.sh getADD.sh 205
4. Domain utilities 205
a. domainMove and domainScale 206
b. domainExtract 206
c. inDomain 207
d. domainCoordTrans 209
e. domainFind-rec 209
5. Plot stress contour from FEA: meshplot.exe 209
6. Plot FEA output and plotfiles: plot.sh 211
7. Plot local ADDomains: plotLS.sh 212

8. Movie Generation 213
F. ANALYSIS 217
1. Void detection: matterInvert.exe 217
2. Structure analysis: CNeval.exe 219
a. Common Neighbour Analysis (CNA) 222
i. Method 222
(a) CNA Values 222
(b) Characterization 224
ii. Code Specifics 224
b. Coordination Number (CN) 226
c. Coordination Vector (CV) 229
d. Near-Neighbor Grouping (NNg) 230

LIST OF TABLES

Page

Table I. List of Subroutines and Functions in the PMAP 182

Table II. Available Directives for PMAP 187

Table III. Model-1. Input File for the Crystal Generation Code 193

Table IV. Input-1. Scale 1 Model in Tensile Load, ... 193

Table V. Model-2. Two Coupled Scale Model Input File, ... 194

Table VI. Input-2. Scale 1 Model in Tensile Load with Surface Images. 194

Table VII. Model-3. Two Coupled Scale Model Input File, ... 195

Table VIII. List of Utility Programs to Augment the Capability of PMAP; 197

Table IX. Element Types Used in CNA Visualization 225

Table X. Sample “xyz” File 227

Table XI. CN Summary Output on Stderr 227

Table XII. CN Density Across the Model 228

Table XIII. Coordination Vector Across the Model 229

LIST OF FIGURES

Page

Figure 1. Example of DC interface with atoms.[31] 6

Figure 2. A 1D example model, exemplifying the scale interface and imaginary domains. The scale ratio equals three. 11

Figure 3. Stress-strain curve for pure scale models 13

Figure 4. Example two-scale model, S2 on top and S1 (atoms) on the bottom. 15

Figure 5. Stress strain curves of pure S1, S2 and the two-scale model..... 16

Figure 6. Configuration energy for the pure S1 and S2 models with free surfaces in the X direction. 17

Figure 7. Example of surface images, *i*, and *j*, linking to real particles 19

Figure 8. The difference of configuration energy from the 3D PBC condition for the free surface models and those with surface images, ... 20

Figure 9. A generic atomic cell (1, 2...9) which links to a generic particle M..... 22

Figure 10. Overview of deep-notch models for validation of dislocation passing scale interface 24

Figure 11. A comparison of dislocation patters between fully atomistic and the GP method before and after passing through the scale boundary..... 25

Figure 12. ADD Model with central hole. 27

Figure 13. Model configuration colored for Coordination Number (CN) 28

Figure 14. Loading stress strain curve for the example model, ... 29

Figure 15. Schematic of the four main domains of a simple GP-FEA model. 31

Figure 16. Schematic of the three-scale GP model for a thin plate with a central hole .. 43

Figure 17. GP-FEA interface design. 44

Figure 18. Continuous WF domain design and its relation with high-scale particles..... 45

Figure 19. Discontinuous WF domain design and its relation with high-scale particles 49

Figure 20. 2% strain displacement comparison 49

Figure 21. 2.5% strain displacement comparison 50

Figure 22. 3% strain displacement comparison 50

Figure 23. A survey for the model size in different modeling work. 56

Figure 24. Schematic of a micrometer-size GP-FEA model..... 62

Figure 25. Distribution of data acquisition domains 66

Figure 26. Y-Displacement comparison between results of GP-FEA 67

Figure 27. X-Displacement comparison between results of GP-FEA 68

Figure 28. Radial displacement comparison between results of GP-FEA 68

Figure 29. Effects of model size, L_y, on the displacement component u_y..... 70

Figure 30. Model size effects on the radial displacement component, $u_{r,}$..... 70

Figure 31. Model size effect on boundary stress component, ... 71

Figure 32. Size effect on the stress intensity factor K_I at 1% strain, ... 79

Figure 33. Energy G_{IC} distribution during cracking the local domains. 80

Figure 34. Critical energy release rate, G_{IC} trend with model size 81

Figure 35. T-S response for domain 6. 82

Figure 36. T-S response for domain 7. 82

Figure 37. Example of crack-tip phase configuration 84

Figure 38. FCC evolution during crack extension. 85

Figure 39. Mid-phase evolution from BCC to FCC. 86

Figure 40. Impact model configuration, 90

Figure 41. Kinetic energy and pressure evolution for the pure atomistic model 91

Figure 42. Max principal stress evolution of auto-duality domains 1-7. 92

Figure 43. Kinetic and Potential energy evolution for the pure Scale-2 model 93

Figure 44. Max principal stress evolution of auto-duality domains 1-7. 94

Figure 45. Lumping and Decomposition events for each AD Domain 95

Figure 46. Kinetic energy and Pressure evolution for the auto-duality model 96

Figure 47. Max principal stress evolution of auto-duality domains 1-7. 97

Figure 48. Main flow process of the PMAP. 102

Figure 49. Initialization process flow, 104

Figure 50. Equilibration process flow. 106

Figure 51. General flow of the GP load process 108

Figure 52. General flow of the GP-FEA load process. 109

Figure 53. The Ewald sum 132

Figure 54. Every charge, q_j has a corresponding charge $-q_j$ 133

Figure 55. Particle *i* with cut-off radius, ... 137

Figure 56. Two layer DC interface. 147

Figure 57. Particle displacement interpolation..... 150

Figure 58. Current loading flow..... 151

Figure 59. FEA-GP V29f10 Flowchart revision 3. 151

Figure 60. Distance ratios are the same for the atom 'a' as for the particles. 164

Figure 61. Subroutine tree of PMAP,..... 182

Figure 62. FEA-GP Cu model at 13% strain and 384 ps, ... 211

Figure 63. The relationship between real matter and inverted mater models. 217

Figure 64. Examples of the two types of pair relationships, ... 223

Figure 65. Examples of common atom configurations, ... 223

Figure 66. $\overline{CV}$ concept for atom *i* with a vacancy. 230

Figure 67. Five step process of grouping particles by a single-link method..... 231

Figure 68. Two types of clusters, (a) a dense group or (b) a Void group. 232

ABSTRACT

Multiscale analysis is the study that bridges the gap between theory and experiment via numerical simulation and works to couple material behavior across disparate length and time scales. There is a growing need for techniques that operate within this regime as progressing technology requires ever more accurate and fundamental understanding of their materials. Multiscale analysis seeks to offer atomistically-based informed solutions to the leading technological problems in materials science. In this thesis the Generalized Particle (GP) method is validated with elasticity solutions and coupled to a Finite Element Mesh capable of efficiently modelling in the micro-scale with atomistic detail at local regions such as crack tips. This coupling method (GP-FEA) is used to investigate model size effects on local atomistic phenomena. Size effects were found for both elastic and inelastic (crack propagation) pre-cracked crystalline Iron samples. For these examples it was seen that models 500nm and larger were consistent with Linear Elastic Fracture Mechanics predictions for the displacement filed around a crack tip for a given load, however models smaller than 500nm underestimated the amount of deformation and had smaller zones of crack-tip phase transformation causing a lower toughness. These results show that the model size used in simulation and modelling must be large enough for the interesting atomistic phenomena to be accurate. This work sets the stage for further model size research based on atomistic analyses in hopes of proving a useful model size guideline for future work to be more accurate.

In addition to this research the multiscale program and numerous tools and utilities developed to acquire this data are explained and examples given. Those scientists interested in particle based dynamic simulation methods and analyses are encouraged to peruse the latter chapters and appendices for detailed information regarding the specific algorithms used and the structure of the programs. This additional rich information is appended to encourage further work and development in the field of multiscale analysis.

INTRODUCTION

In the early part of the 21st century there has been success in the semiconductor and information technology industry which has caused a significant expectation for nanotechnological breakthroughs. The progress in these areas is still not quite mature as they currently rely on scientific advances. This leaves open the study of predictive methods to hasten the expected nanotechnological breakthroughs. Of particular interest is computational modeling which can take advantage of supercomputers to solve complex problems is a wide variety of subjects.

Material properties are inherently of hierarchical multiscale nature, with behavior at the atomistic scale dictating the behavior at higher meso-scale domains and higher to macro-scale domains.[1] It becomes important to capture this behavior of different scales to accurately model the material properties. The big challenge comes from uniting the phenomena at low scales with material behavior at high scales; creating a demand for new and unique scale bridging methods to solve engineering problems from hardening materials used in agriculture to implementing new alloys for use in aviation. Modeling with a fundamental foundation will help old theories become more accurate and robust and assist in the formation of new theories.

Science traditionally functions based on two stages of understanding, theory and experiment. Computational simulation bridges these two stages for a smoother more continuous understanding by expanding upon the simplified assumptions made by theory allowing more complicated and more realistic simulation conditions. Simulation has the added benefit of unrealistic conditions, or circumstances that are unattainable in laboratory conditions. This allows for extrapolation and predictive power which aides in a more intrinsically connected picture of fundamental phenomena. Multiscale analysis has a place in Science in general as it is an intermediate methodology that not only connects physical scales but also disciplines, such as material science, solid mechanics, physics of condensed matter, chemistry, biology, et cetera.

A. Motivation

Since material behavior originates at the atomistic scale it makes sense to investigate the atomistic phenomena such as defects like voids, dislocations and grain boundaries, in an attempt to elucidate how they cause large scale material behavior. This is essentially the motivation and perspective that multiscale analysis functions upon.

Crystal defects and stable structures of non-crystalline materials are widely studied in material science because of its importance in technology.[2-4] Crystal defects include point defects (i.e. vacancies), linear defects (dislocations), and planar defects (twin and grain boundaries).

To investigate atomic level defects in materials the thermodynamic concept of minimum free energy can be used, such that for a defect to be stable, the defect state must have the lowest free energy for the given boundary conditions. This is akin to the principle of least virtual work often used in continuum mechanics. Methods used to analyze such defects include the M-T model proposed by Mott & Littleton,[5] MD and ab initio. The last two methods suffer from size limitations but the M-T model does not, mainly due to the small spherical region I that surrounds the defect of interest. This region I is the only region that undergoes any kind of atomistic dynamics, the surrounding medium in region II is the continuum region that is held fixed with the boundary conditions. This boundary constraint allows the method to be very fast. The M-T method is included in such well known simulation programs as GULP.[6] Although this method can be used to investigate the general properties of defects it lacks adequate interaction with service loading conditions, the model is over-simplified and suffers from small model sizes which struggle to simulate defect nucleation in realistic processes.

Dislocation nucleation and evolution are fundamental mechanisms of inelastic deformation such as plasticity. Understanding these physical mechanisms is a major scientific task, and multiscale analysis of inelasticity for nucleation and scale transfer of dislocations may provide the key to understanding it. Recently, discrete dislocation analysis confirms experimental results that the flow strength of polycrystals depends on grain size.[7] This kind of analysis enables the interpretation of plastic mechanisms, as well as size and surface effects, for freestanding FCC thin films.[8] Based on continuum dislocation theory, the analysis can also indicate typical material behavior, such as the Bauschinger effect.[9,10] The length scale effect of plasticity is incorporated through the

explicit coupling of the density of geometrically necessary dislocations with the crystallographic slip rule.[11] Dislocations are line defects originating in atomic motion; they can be simulated by atomistic analysis. A review on atomistic modeling of fatigue is provided by Horstemeyer et al.[12]

The investigation of material failure is important for industrial technology and fracture mechanics has been used widely in failure analysis since the 1950s. However, as technology pushes the limits of material properties, it becomes ever important to understand and control the microscopic and even nano-properties of materials. Specifically, how cracks nucleate to cause materials to fail is based on the atomic structure and atomic scale properties of the material. As a consequence, the classic parameters used in fracture mechanics such as the fracture toughness K, fracture energy J, and crack opening displacement COD require more fundamental definitions distinct from their empirical origins.[13] Just as classical thermodynamics can be described by statistics which formed the very successful new study of statistical thermodynamics, coupled the nano- and micro- scales to the macro-scale bulk thermodynamics processes and properties. For this pursuit of coupling small scale behavior and properties with macro scale behavior and properties there is extensive ongoing research using Molecular Dynamics (MD) simulations to investigate material failure.[2-4,6,13-15]

B. Classification of Multiscale methods

There are many ways to use multiscale analysis and many different models and disciplines to use it in. This wide variety of applications causes people to classify different types of multiscale analysis by their methodology and by the scale at which the analysis is performed. For the methodological classification, multiscale analysis can be split in twain to describe methods that couple different spatial scales together concurrently (parallel), transferring information simultaneously both ways from one scale to the other[16,17] and those that model low scale behavior and send the resulting property data to a separate higher scale model in a one-way transfer.[18] These methods are called, respectively, the concurrent and hierarchical (sequential) multiscale methods.[19] The other way to classify multiscale analysis based on the scale of application is also with two different categories. The first is class I which describes multiscale models primarily used

for low scales from Quantum mechanics and the atomic scale up to about the nano-scale. Class II focuses on the larger scales from the micron scale up to the macroscopic and tends to be very empirical. With these two types of classification any multiscale analysis could be given two descriptions, one describing the methodology used and the other at what scale the analysis is addressing. The following subsections will focus mainly on the hierarchical and concurrent classification primarily of Class I being atomistically-based methods.

The main reason why there is such a distinction between Class I and Class II is due to the experimental limitations and the modeling limitations. Experiments can directly validate Class II models however, experiments generally have scale limits on the micron scale and it becomes difficult to validate Class I models directly. From the modeling perspective, the methods used today have a difficult time linking the behaviors of materials on the nanoscale to the large scale bulk properties.[19] This gap in capability prevents atomistic-based models from predicting large scale material behavior thereby failing to link the two classes together. If micron sized models can be developed, it would open a door to solving important problems in engineering based on a fundamental atomistic foundation.

1. Hierarchical

The hierarchical approach uses a one-way transfer from low-scale properties into key variables and functions for the higher scale.[18,19] This approach is very general and can be used in most disciplines. Coarse graining is a type of hierarchical multiscale and is the act of integrating out redundant degrees of freedom.[20] Lyubartsev et al. modeled higher scales in this way by leaving out "uninteresting" degrees of freedom. From first-principles simulations they obtain a set of atomistic pair-wise effective interaction potentials to be used as a force field with no need for empirical data. These force fields are then used in classical all-atom simulations to scale up the system size by 2–3 orders of magnitude. They then use the MD results to develop a set of effective potentials for a chosen coarse-grain level suitable for large-scale mesoscopic or soft-matter simulations otherwise unobtainable by atomic resolution simulations.[21]

Hierarchical multiscale is used extensively in micro-biological simulations, for example Ortoleva et al. proposed a force-field based methodology that "takes advantage

of the structural hierarchy natural to macromolecular assemblies in defining the system as a collection of mutually interacting subsystems with internal dynamics, which simultaneously preserves the all-atom description." where the main driving force is the free energy dynamics. Although their approach is computationally intensive, the drawback is outweighed by the large Langevin timestep.[22]

A popular method used to coarse grain atomistic level biological simulations is the use of the MARTINI force field and bead method.[23] Their rational and motivation comes from the idea that interesting phenomena in biological systems occur at long time scales that cannot be accessed by molecular dynamics, such as the dynamics of large proteins and material self-assembly. Coarse graining allows simulations to run at 2-3 orders of magnitude larger time scales.

A well-known Class II continuum model that predicts the brittle to ductile transition based on a competition between the loss of atomic cohesion and the emission of crack tip dislocations, this model is known as the Rice-Thomson (RT) model.[24] The difficulty with this model is acquiring the critical energy release of the slipping events. Later in the 1990s a method that uses Frenkel's sinusoidal distribution[25,26] for the shear stress along the slip plane ahead of the crack tip was proposed by Peierls[27] and Nabarro (PN). The PN model allowed Rice to calculate the slip energy via the J- integral. An advantage if the PN model is that dislocations need not exist initially, but traces its evolution from initiation to termination. Even with this advancement the PN model has a shortcoming based on its transformation of the atomic behavior to the continuum representation and thus the model does not give the best results for the calculated dislocation slip energy.[28] it so happens that in this case the Class II RT model can be enhanced with a Class I atomistic based model that can find the constitutive equation required by the RT model.[29,30] Cleri et al. carried out MD simulations and obtained a more reliable constitutive equation for the RT model to determine the brittle to ductile transition. This example shows how well atomistic-based hierarchical multiscale analysis can be used to acquire accurate results by enhancing long used continuum level models.

2. Concurrent

The concurrent approach to multiscale analysis simultaneously operates on a small scale that is coupled in space to a higher scale. Applications where this is

advantageous have the similar characteristic where there is a local small domain of significant interest that is surrounded by a less interesting domain. For example, the grain boundary created by micron-sized grains, turbulence with small scale vortices surrounded by large scale bulk fluid motion and elasto-plastic crack tip propagation surrounded by a finite continuum. All of these examples have interesting behavior at small scales in specific local areas and are surrounded by less interesting material behavior, such behavior that could be homogenized into continuum constitutive laws.

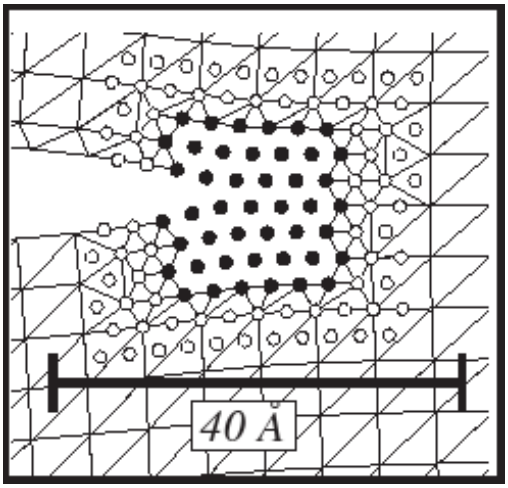


Figure 1. Example of DC interface with atoms.[31]

Some existing concurrent multiscale methods directly connect (DC) atoms in the atomistic domain with finite element (FE) nodes in the continuum domain, see Fig. 1, or through some form of interpolation.[32] Methods such as the Quasicontinuum (QC)[33,34] and CADD.[35,36] These methods require that the element size at the scale interface be equal to the lattice constant to perform well. This constraint requires many FE elements at the interface when the model size is extended from nano- to submicron- and micrometer-scales. Thus, a new principle and method for variables to bridge scales should be developed to avoid this deficiency.

Most existing concurrent multiscale methods do not truly show seamless transitions of key variables such as deformation, motion and force at the scale interface.[36] The transition problem is caused by the incompatibility between materials on both sides of the scale interface; the atomistic scale has non-local behavior while the continuum has local behavior. Here non-local behavior refers to the inter-atomic potential that couples atomic

motion that acts not just on the nearest atomic neighbor, but also on atoms that are far from each other. In contrast, local-behavior refers to models that depend only on the deformation gradient at a given location in space to provide force or stress to that same location in space. The incompatibility produces non-physical phenomena such as ghost forces and artificial wave reflection at the interface. A new type of scale-bridging method should be developed to eliminate sources of incompatibility, thus realizing a seamless transition. The GP method is taken as a likely candidate to accomplish this bridging.

C. Atomistic-based multiscale analysis, Class-I

Due to the desire to have fundamental atomistic behavior describe large scale phenomena, atomistic-based multiscale analysis is required. There has been certain success in using such Class I techniques to elucidate important mechanisms that have been used to drive theoretical models and increase the accuracy of existing ones. Class-I and concurrent multiscale methods will be the main focus of this thesis.

The QC method has been applied by Miller and Rodney for investigating the dislocation nucleation during indentation.[37] Since QC is a concurrent method it was able to increase the model size much larger than an MD simulation could have, as a result is was able to obtain not only detailed information but also more accurate. They did two types of studies, one model in 2D and another in 3D. They found that for the 2D model dislocations nucleated spontaneously in a dipole along a slip plane, involving only about 10 atoms on either side. Other than these atomic motions there was hardly any other deformation. What is very interesting is what they found in their 3D model. Instead of a dislocation dipole they discovered a small ring or loop of dislocations form spontaneously, the size of which depended only on the size of the indenter. This multiscale analysis reveals that dislocation nucleation is an inherently 3D phenomenon involving the collective motion of a disk of atoms within two adjacent slip planes. This shows that the PN model used in Rice's method is not as accurate, but with the help from atomistic-based multiscale analysis these long established theoretical models can be further improved.

In an attempt to study the effects of dislocations on a larger scale, multiscale analysis of plasticity was developed. A three dimensional (3D) simulation model of

discrete dislocation dynamics (DD) was developed recently[38] to understand the relationship between dislocation glide and macro-scale plastic slip behavior in single crystal BCC: Tantalum. They concluded that the incorporation of fundamental atomistic information is critical to develop a physics-based, predictive meso-scale DD model. The latter has been used to investigate individual dislocation glide behavior and macro-scale plastic slip behavior in single crystal BCC metals. When dislocation multiplication occurs, they move to the interior of the model and pass scale interfaces such that the accuracy of dislocations smoothly passing the scale interface is essential. Unfortunately, the problem remains unsolved. With remeshing of the continuum to increase the atomistic domain, dislocations passing the scale interface may be avoided; the QC multiscale method uses this technique.[39]

A promising technique for bridging Class-I and Class-II multiscale models through the use of atomistically-based multiscale is with a hybrid approach, using Class-I concurrent multiscale methods to derive critical key variables for use hierarchically in Class-II methods. The most common usage of this practice is in fracture analysis. Fan and Yuen used MD to derive a traction-displacement relation to use as cohesive zone parameters for larger scale FEA models.[40] This approach is advantageous since it avoids the time-scale problem of large concurrent multiscale to simulate the entire propagation process.

D. Inadequacies and Existing Needs

Although there has been great progress in the use of MD to investigate crack propagation and the origins of failure there are two main shortcomings:

First, most work concentrates on crack propagation rather than crack nucleation. Although studying defect and crack nucleation is important the connection with failure is a difficult topic. These defects are both theoretically and practically important since they are the key properties for understanding the underlying mechanisms of failure. For brittle materials these defects can be the key property to denote material failure.

Secondly, much work has imposed special treatments to ensure that the crack propagates along a desired path. Such treatments included the cutting of atomic bonds along the designed propagation path,[41] designing the desired path to be “weak”,[42,43] using

a screening method to prevent atomistic interactions across a Grain Boundary (GB).[44] These treatments are convenient when using the cohesive zone model (CZM) to investigate and model the crack propagation behavior; however their effectiveness must be further validated because the crack propagation essentially depends on how it was nucleated.

From the modeling perspective, the methods used today have a difficult time linking the behaviors of materials on the nanoscale to the large scale bulk properties.[19] This is mainly due to the complicated question: what is the minimum size of material that can be considered a continuum? The answer to this question is highly material dependent. For nano and micro grained structures the Hall-Petch relationship is valid when the grain size is larger than 20 nm[45] making it possible to describe continuum behavior based on the mechanism of dislocation pile-up at grain boundaries. Another reason for ambiguous answers to this question is due to the strong surface effects at the nanoscale. These effects become more significant when the area/volume ratio increases. The area/volume ratio for a sphere of radius r is $3/r$, thus for a 1 nm radius the ratio is 3 whereas for a 1 micron radius the ratio is 0.003 that is 1000 times smaller and surface effects are generally neglected at scales above the micron. This indicates that the surface energy is very important to consider on the nanoscale and must be included if an accurate material model is to be used at these small scales. There is work that exists that incorporates a surface elasticity theory into continuum mechanics for use at nanometer scales.[46] In the elastic range continuum models are incredibly reliable and can be applied to discrete materials having a size of a few nanometers.[47] However these continuum models break down when plasticity is involved at these small scales, such that the minimum required model size for plasticity remains unsolved.

This gap in capability prevents atomistic-based models from predicting large scale material behavior thereby failing to link the two classes together. If micron sized models can be developed it would open a door to solving important problems in engineering based on a fundamental atomistic foundation.

METHODOLOGY OF THE ATOMISTIC-BASED MULTISCALE ANALYSIS

Within this chapter the Generalized Particle method will be explained and examples to show that material properties do not change with particle scale under the given small-strain assumptions. How the concurrent scale coupling is accomplished via imaginary particle domains; the need for surface corrections in higher-scales will also be discussed preceding a description of a method for compensating for it by using imaginary particles. The ability to switch from higher-scales to lower scales will be explained and a method described. Certain shortcomings of the GP method will be discussed that outline the suitability and applicability of the GP method. Lastly a capability or option for the GP method is coupling with Finite Element domains and will be used in a proof of the GP method's elastic capabilities in the Chapter III.

A. The Generalized Particle (GP) Method

The GP method has some very important geometric features that make it ideal for use in multiscale inelasticity; its main feature is Scale Duality for particle domains. The method assumes a material model consisting of generalized particle domains of different scales beginning with n=1 as the atomistic scale and higher values of *n* corresponding to the continuum scale. These higher scales maintain the material structure of the atomic scale which is more closely related to real material structure, consisting of discretized atoms or in this case, particles. These higher domains can be represented easily as the atoms they constitute, by a mathematically proven process called inverse mapping,[48] thus enabling the dual nature particle domains.

These high scales can effectively reduce the degree of freedom dramatically. Generally, the number of atoms ℓ_n that one generalized particle represents at the n^{th} scale domain for BCC, FCC and other cubic crystal structures can be calculated by

$$l_{(n)} = k^{3(n-1)} \tag{1}$$

Likewise, the particle mass, $m_{(n)}$, the lattice constant a_n and the position vector $\bar{R}_{IJn}$ between particle *I* and *J* in the particle domain can be expressed, respectively, by atom

mass m_0, crystalline atomic lattice constant a_0 and the position vector $\bar{r}_{ij}$ between atom *i* and *j* in equations (2)-(4), where atoms *i* and *j* correspond to particles *I, J*.

$$m_n = k^{3(n-1)} m_0, \;\; a_n = k^{n-1} a_0, \;\; R_{ijn} = k^{n-1} r_{ij} \qquad (2-4)$$

where the ratio of adjacent scales, k , is defined as a ratio of the lattice constants, as a_{n+1} of $(n+1)^{th}$ domain over that of a_n of the n^{th} domain (n=1…m-1) or a_{n+1}/a_n. To see how effective the GP method is at reducing the degrees of freedom, assume a scale ratio of $k=3$. In this case, equation (1) shows that for scale domain n=3 the generalized particle is lumped from $\ell_3 = 729$ atoms; for n=4, $\ell_4 = 19283$; for n=5 and 6, $\ell_5 = 531441$ and $\ell_6 = 1.43489 \times 10^7$. For an aluminum cube with volume of 1 μm^3, the total atomic number is approximately 60.2136×10^9 (a_0= 4.05 Å). If all atoms are lumped into particles, the whole system can be represented by 113.3×10^3 particles of 5th scale or 4196 particles of 6th scale. Thus, one can have sufficient degrees of freedom to put atoms in the critical areas around dislocations, interfaces and grain boundaries while keeping the model size large enough.

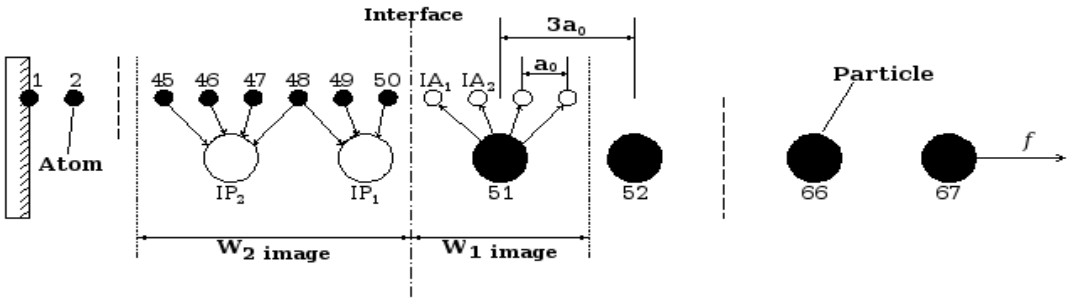


Figure 2. A 1D example model, exemplifying the scale interface and imaginary domains. The scale ratio equals three.

Natural interaction forces passing through a scale interface must make both sides of the interface the same scale, whether they are particles or atoms. Therefore atoms only interact with atoms and particles only interact with same-scale particles at its natural scale interface. The GP model satisfies this requirement by introducing two imaginary domains, $W_{n+1image}$ and W_{nimage}, respectively, to the left and the right of the n^{th} scale interface. (See Figure 2) The imaginary particles' positions are determined by the average position of the atoms in its Neighbor Link Cell (NLC), these are the atoms that would be

lumped together to form that imaginary particle. In this case these atoms define the position of the particle. This particles' position is then used as part of the interaction force with the real particles. The imaginary atoms also have an NLC, their NLC consists of the nearest real particles, and the imaginary atom's position is determined by these particles. The imaginary atoms then are used to impose a force upon the real atoms. It is through this geometric averaging process that forces and displacements are transferred across the scale boundary. Each imaginary particles' NLC are defined when the model is created, and they keep this NLC throughout the simulation.

Simulation dynamics are not performed on the imaginary particles, they only influence their real particles; as a result the imaginary domain does not produce surface effects since they are within, and defined by the domain of the other scale. Particles are used for the Higher scale behavior since they can be treated in the same manner as atoms, and can use the same atomic potentials -- once properly scaled down and using the Cauchy-Born rule.[48] Treating the higher scales as atoms or particles, gives the domain the advantage of non-local behavior. Other multiscale methods use a finite element continuum that consists of local behavioral interactions, which is not well suited to plasticity.

1. Material property scale independence

To illustrate the fact that high scale domains may be used in place of atoms in areas of uniform deformation, three models were developed with the same size of X=72.580 Å, Y=145.16 Å and Z=72.580 Å. However each had a difference scale; one model was made entirely of atoms, another purely of scale-2 particles and the third of S3 particles. Each of the models were a single Copper crystal thus each model had 64000, 8000 and 1000 particles, respectively, for a scale ratio of k=2. These models were equilibrated in an NPT ensemble before an application of load at a temperature of 300 K and pressure of 1 atm; the Morse potential was used. All models were simulated with 3D PBCs. Tensile load was applied along the Y direction.

When loading uniaxially in 3D PBCs there remains the question of the treatment of the transverse dimensions. Typically 3D PBCs fix the system dimensions thus maintaining a constant volume. Other times the system dimensions are scaled in order to change the system pressure, this is the technique used to barostat systems. However when

loading along one direction many keep their transverse system dimensions constant which causes a constraint on the system's Poisson ratio fixing it to be null. Others may scale the transverse dimensions to preserve the system volume, this may alleviate the system triaxial stress caused by the former treatment but this also fixes the system's Poisson ratio, this time to be $1-\sqrt{0.5}=0.2928$. This may be a more realistic approximation to the material behavior however there may still be transverse residual stresses. A third approach is to relieve the transverse stresses in a similar way to barostatting the system, only this time the transverse stresses are used rather than the full system pressure. This third technique and applications for anisotropic barostatting is described in detail in Appendix A.5.b. Pedone et al. called the first technique that makes v=0 a Constrained Simulation (CS) and called the barostatting technique an Unconstrained Simulation (UCS)[49] both of these loading techniques were used in this study and they both yielded the same conclusion: that high scale particle domains can replace the atomic domain. Figure 3 shows the stress strain curves for the CS condition it can be seen that each scale is coincident until about 12% strain at which point the S1 model began to fail, indicated by the drop in stress. These results show that whether the loading conditions had or had no tri-axial stress, replacing atoms with higher scale particles is supported before material failure. So limiting material failure phenomena to the atomistic scale should preserve the correct material response.

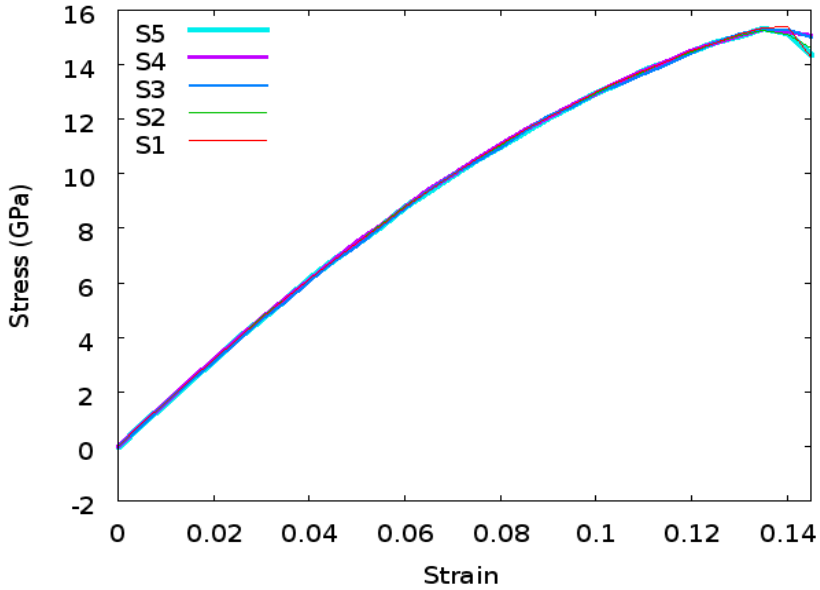


Figure 3. Stress-strain curve for pure scale models

2. Scale interfaces

Since individual GP scales have been shown to be sufficiently accurate to take the place of atomic domains, it is natural to consider how well these scales can be coupled together to reduce the DOF in areas that don't require atomic resolution.

When designing a coupled two scale model one should start first by considering the placement of the real atom domain and the higher scale real particle domains in relation to it. The atomistic domain should be placed in areas with large deformation gradients, such as surfaces, grain boundaries, and defects. Since real particles and atoms must interact with the same scale, i.e. an S2 particle can only interact with another S2 particle; imaginary domains are used to interact with the real particles. They are imaginary particles due to the fact that their positions are determined by the adjacent scale deformations not from dynamic motion calculations as described at the beginning of this section.

In this example, the model, see Fig. 4, is simulated in 3D PBCs and equilibrated in the NPT ensemble at 300K and 1atm, the model dimensions are X and Z=43.548 Ang, Y=72.58 Ang. The atomic real domain is on the bottom (as if the Y direction points up) and the S2 real particles are on the top. So for the atoms to pass information to the scale-2 domain, there must be an imaginary S2 domain overlapping the real S1 atoms at the interface (Y=0.0 and Y=-36.29). Thus the real atoms will determine the imaginary S2 particle positions and the real particles will interact with imaginary particles.

Another imaginary domain must be used to pass scale-2 information down to the atomic real domain. To do this an imaginary S1 domain is used overlapping the real particles on the upper side of Y=0.0 and underside of Y=36.29. Their positions are determined by the real particles and interact with the atoms. Thus we have complete scale coupling based on deformation transfer. The reason for the imaginary domains at the very top and bottom is due to periodicity in the Y direction, it could be imagined as an infinity layered model with alternating scale domains, this feature and loading condition make the model simpler to make, simulate, and compare.

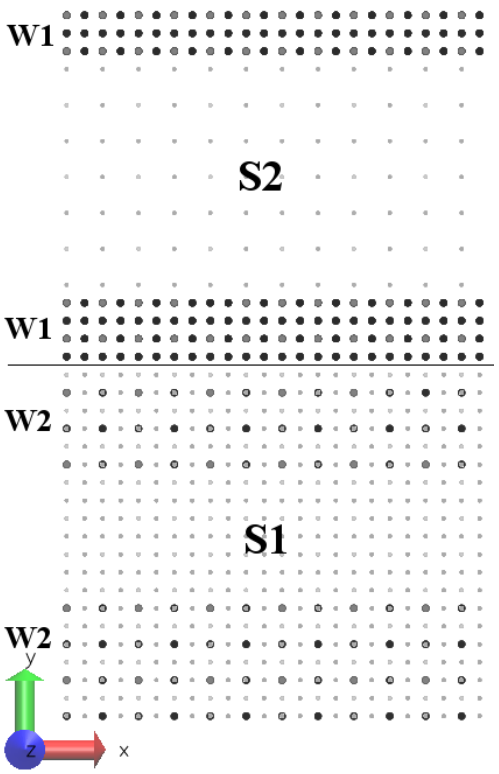


Figure 4. Example two-scale model, S2 on top and S1 (atoms) on the bottom. The dark points are the imaginary particles (W2) and atoms (W1) that couple the two scales. It is periodic in all direction so the top and bottom also couple across the boundary.

The size of the imaginary domains should span the entire interface (from X,Y=-21.774 to X,Y=21.774). But the depth of the imaginary domains (deviation from Y=0.0 and Y=±36.29) should be equivalent to the scale's interatomic cutoff radius. So the imaginary S1 domains (overlapping the real particles above Y=0.0 and below Y=36.29) should have a depth of 6.5Ang for the Morse Potential so they should go from Y=0.0 to Y=6.5 and Y=29.79 to Y=36.29. The imaginary S2 domains that overlap the real atoms below Y=0.0 and above Y=-36.29 should go from Y=-13 to Y=0.0 and Y=-36.29 to Y=-23.29. These domains are twice as deep as the atomic imaginary domain because they interact with scale-2. And scale 2 must scale up the cutoff radius used, so that when the inverse mapping method is used to determine particle forces and accelerations, the correct number and range of particles (inverse mapped into atoms) are used so that the same interatomic potential can be effective on the particle scale. For this reason the

imaginary depth should be equal to the cutoff radius so that the real particles 'feel' as though they are in bulk material. If the imaginary domain is not deep enough, then the real particles would 'feel' like they are close to a surface and their behavior would be different than bulk behavior.

One may notice in Fig. 4 that the particle layers of the imaginary domains are not consistent; i.e. those above the interface have four layers while those below have only three layers. This is an artifact of the measurement used during model development. This perceived problem may easily be fixed but this case is included here to illustrate a point. This model is a single crystal of FCC copper with a lattice constant of 3.629 Ang. When using a cutoff radius of $r=r_c=6.5$ Ang, four layers will be included if counting the layer at r=0.0 (r in [0.0, 6.5]) but only three will be included if not counting the first layer at r=0.0 (r in (0.0, 6.5]). Thus we see that three layers of imaginary particles is sufficient, however, due to the nature of imaginary particles, a fourth layer causes no penalty. So a first look at the model might seem as though there is a problem, but upon careful inspection it is nothing to worry about.

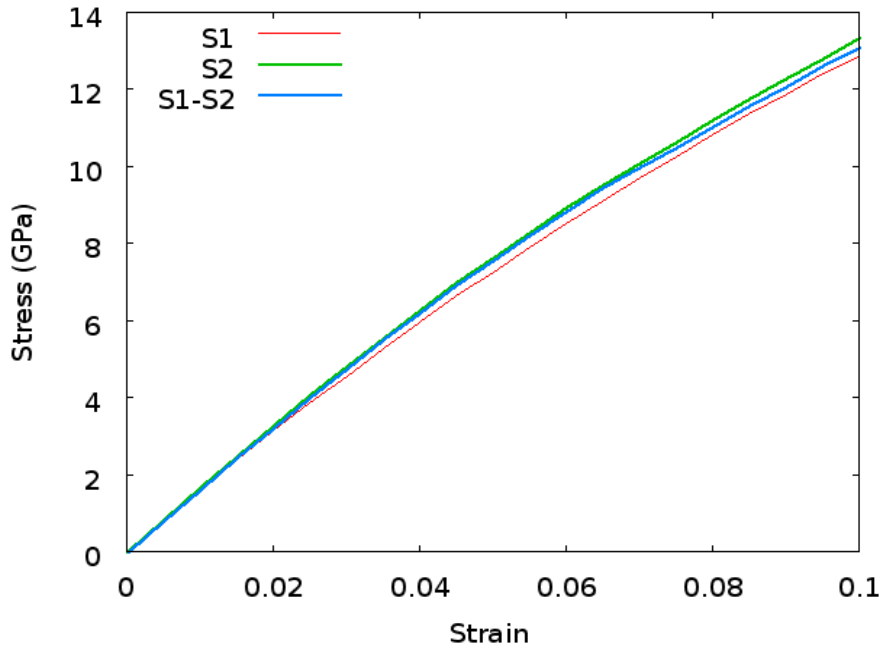


Figure 5. Stress strain curves of pure S1, S2 and the two-scale model.

This two-scale model was loaded in tension along the Y direction in CS and compared with pure S1 and S2 models. The comparison can be seen in Fig. 5 to be very similar to both pure S1 and S2 indicating that the use of the scale coupling scheme is satisfactory

3. Surface corrections

It is not always feasible to use periodic boundary conditions and not always physical. In many cases free surfaces, or at least non-periodicity, is desired. Since a surface is a discontinuity between a solid/liquid and a vacuum/gas it requires atomic resolution to be accurate. However in many cases this accuracy may not be needed. In this subsection will be discussed the cause of surface effects in higher scales and a way to minimize or circumvent them.

The same models as used in section II.A.1 are used to demonstrate the effects of having a free surface. The periodicity in the X direction was removed and the models were equilibrated at 300K and 1atm then loaded in tension after 15 ps. Figure 6 shows the configuration energy for the atomistic S1 model and the scale-2 model. It is seen that the S2 model has significantly higher energy than the S1 model.

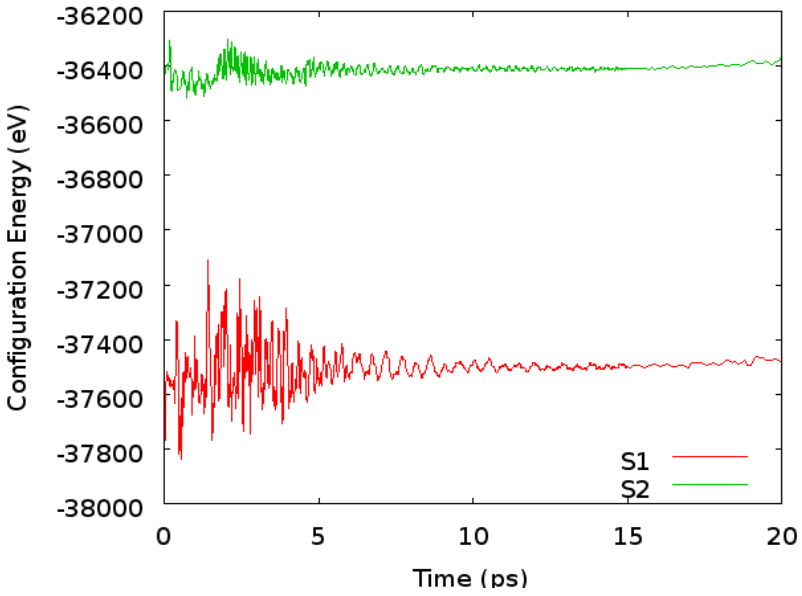


Figure 6. Configuration energy for the pure S1 and S2 models with free surfaces in the X direction.

The reason why S2 has larger energy is due to two reasons, the surface energy and the fact that it is S2. Since it is S2, the cutoff radius is *k* times as large as in S1, meaning that the surface effects will be *k* times as deep, however the model volume is the same as in S1, this causes the surface to volume ratio to be larger in S2 than S1. Essentially, higher scales have larger surface effects than smaller scales because their surface effects go deeper into the material. Imagine an atom at a distance of *r* from a free surface, if *r* is less than the atomic cutoff radius, r_c, then this atom is influenced by the free surface. Now an S2 particle that is *kr* from a free surface will have about the same configuration as that atom, and will be influenced in the same way due to inverse mapping, yet the particle is twice as deep in the bulk.

When high scale general particles are exposed to a vacuum, the material discontinuity causes a greater surface effect than an atomic surface. This surface effect can affect the surrounding deformation and cause trouble in various parts of the model. This very noticeable change in configuration energy caused by surfaces could be large enough to nucleate dislocations from the surface when the atomistic model would not. This difference could cause incompatible failure phenomena, thus some technique should be used to minimize this effect.

An alternative to using periodic boundary conditions in an effort to reduce or eliminate the surface effect of high scales is to use a way that would reduce the force imbalance at the surface. One way to do this is to place imaginary particles at the surface so that the real particles near the surface just *see* other particles, making them *feel* as though they are part of a bulk material. The trick comes from the question of how to determine the positions of these imaginary surface particles? They could be rigidly linked to real particles on the inside. Another approach would be to restrain the displacement of the particles at the surface from moving perpendicular to the surface. This would roughly maintain their perpendicular strain to that of the inner material; for a uniform strain, there is no wide spread effect from the surface force imbalance.

In this instance the former definition will be used around the edges of the model. These edges are exposed to a vacuum and not in periodic conditions so they will be affected by the surface effects. The surface image layer, to be most effective, should have

a depth equal to the inter-particle cutoff radius, for the same reasons as for scale interfaces. For instance if the interatomic cutoff radius is 6.5 Å and the scale ratio k=2, for S2 it would be 13.0 Å. Surface images are created before the simulation is run. The user must specify domains within-which real particles will be converted to imaginary particles and be rigidly linked to a group of the closest real particles of the same scale, see Fig. 7. So in this case, domains should be specified all along the edges of the S2 domain having a depth of 13.0 Å.

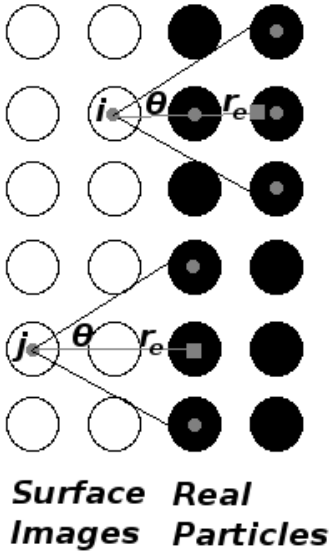


Figure 7. Example of surface images, *i*, and *j*, linking to real particles within a cutoff and angle θ. Their fixed distance from the average, r_e.

Surface images are still required even if the model will be connected to an FEA mesh, because the GP model must equilibrate alone first, before connecting with FEA. The Surface images will later become the WG domain during FEA connection, which will be discussed later.

When the models have surface images and are equilibrated again their configuration energy is the same as if they were equilibrated with 3D periodic boundaries. Their energy difference is shown in Fig. 8 to be zero. This is a significant improvement to the free surface case. This shows that the use of surface images effectively eliminates the free surface effects. This is useful for models with geometries that are unsuitable for periodic conditions, or that are of a nature inherently unperiodic,

such as a thin plate with an edge crack; where the use of periodic conditions would cause the model to behave as if it had a center crack that recurred at every model width.

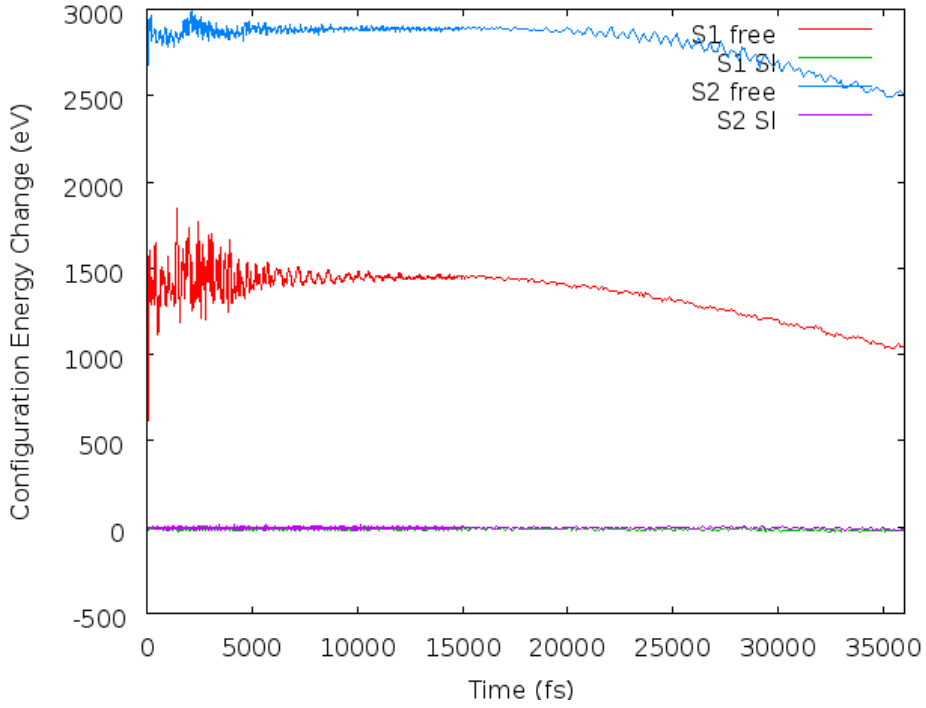


Figure 8. The difference of configuration energy from the 3D PBC condition for the free surface models and those with surface images, showing that the surface images recover the same configuration energy as the 3D PBC condition.

4. Scale duality

Scale duality is a GP concept that allows higher scale domains to be mapped to lower scales and vice versa. This concept allows the use of inter-atomic potentials to be used to describe high scale particle dynamics.

a. Decomposition

This concept can also be used to decompose a higher scale domain into a lower scale domain for better resolution. This is particularly important when studying the migration of dislocations or the path of a crack.

The basic approach is to use scale duality to decompose particles into their corresponding atoms for a smooth transition. This is a more thorough and accurate way for defects (e.g. dislocations, twins and point defects) to smoothly pass scale interfaces.

After decomposing the particle region into atoms along the dislocation path, dislocations can be naturally transferred.

The critical issue is how to decompose the deformed particle into atoms to keep the deformation pattern. This will involve some treatment[50] to transfer the reference atoms to the currently deformed ones.

Two strategies can be chosen for particle decomposition. One could be considered a Lagrange type decomposition method; that is to decompose a generic particle M (M=1, 2… M_P) into atoms from which the particle was lumped. Here, M_P is the total number of particles in the selected particle domain. The other is a quasi-Euler type of decomposition method that needs only a regional distribution of the decomposed atoms with no need to distinguish which atoms belong to which particle. Here, the term "quasi" is used to denote that the particles can move in the space before decomposition and they are fixed in the space during the decomposition process. Specifically, the second strategy focuses on the decomposed atoms in the deformed unit cells which directly connect to the generic particle M (see figure 9 below). Summarizing the distribution of the decomposed atoms in these cells for each particle M (M=1, 2… M_P) distributions of decomposed atoms in all regions of the selected particle domain will be determined. In this work we will take the second strategy for simplicity.

For ease of visualization we discuss a decomposition of second-scale particles into atoms with the scale ratio k=2. Since the decomposition into atoms should be completed before dislocations reach this region, all particles and their corresponding atoms in the given region are deformed but the variation in deformation gradient between different neighbor cells is mild. Figure 9(a) shows two cubic unit cells located in one quadrant which directly links to a generic particle M before deformation. These cells can be FCC, BCC and HCP. In Figure 9(a), the BCC structure is shown where the small cube with labels 1, 2… 9 is the atomistic unit cell and the large cube with bold numbers within parenthesis, (1), (2)…(9), is the unit cell of the second scale.

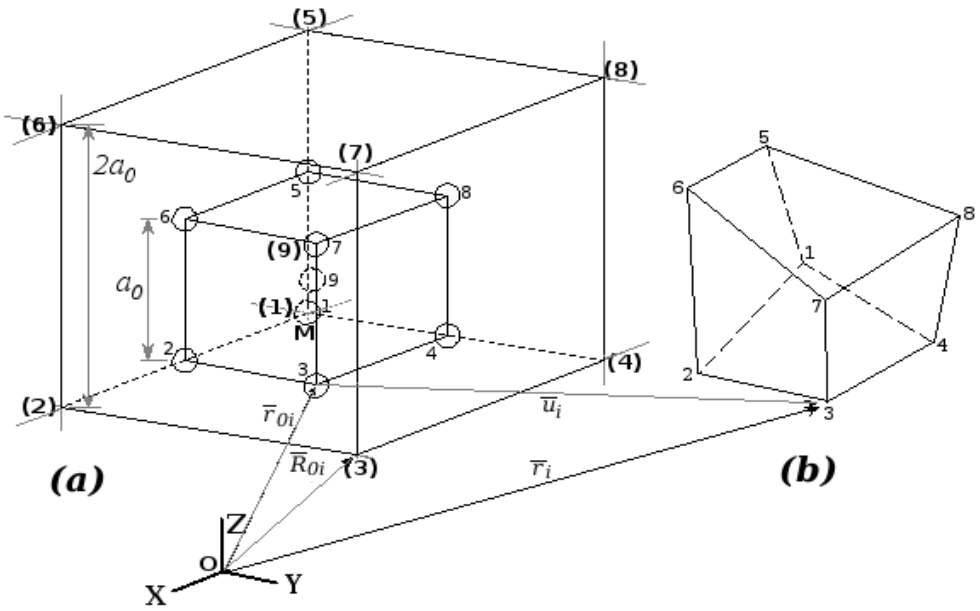


Figure 9. A generic atomic cell (1, 2...9) which links to a generic particle M and is located in M's first quadrant and inside of a second-scale particle cell ((**1**), (**2**)…(**9**)). (a) Configuration of the atomistic cell before deformation which is inside of the particle cell, (b) Configuration of the atomistic unit cell after deformation at an arbitrary loading step.

All positions of the atoms in the small unit cell before deformation can be uniquely determined by its crystal structure; the same is true for atoms in the other 7 atomistic unit cells inside the other 7 octants linked to particle M (M=1, 2…M_P). If M is not inside the bulk but on the surface it will involve only 4 unit cells inside 4 octants. Now, it is more clear if after deformation the positions of atoms in the small unit cell can be determined then the configuration of those in the 8 (or 4) octants and, in turn, all decomposed atoms inside the selected second scale can be determined. The task therefore is how to use the simulation-obtained displacements of particles around the generic particle M to determine the positions of atoms in the deformed generic unit cell as shown in Fig. 9(b).

Two approximate methods are used for this purpose which are introduced in detail in Section 3.2 and 3.3 of Ref. 50:

Decomposition Method 1: Using shape functions to determine atom positions based on particle displacements at unit cell vertexes. This method is simple with a certain accuracy which is based on the interpolation function used in the 8-node hexahedral element.

Decomposition Method 2: Calculating deformation gradient matrix $\underline{F}$ based on particle displacements at unit cell vertexes for determining atom positions. This method uses the Cauchy-Born rule to determine the deformation gradient. $\underline{F}$ In turn, the position vector $\bar{r}_i$ of the generic atom *i* after deformation can be expressed as a function of $\bar{r}_{0i}$; its position vector before deformation as follows:[11]

$$\bar{r}_i = \underline{F}\bar{r}_{0i} \quad i = 1,2.....n \tag{5}$$

This concept shows that the high-scale particle can fully represent the position, velocity and force of the atoms, which lump together to form the particle, to save DOF. High accuracy can be obtained in highly inhomogeneous deformation fields by the decomposition of particles into atoms. However, this method has constraints because in the mathematical proof,[48] the Cauchy-Born rule is used. This rule requires the deformation gradient tensor, $\underline{F}$, to be sufficiently smooth over the region that the particle represents the atoms. If this condition is not satisfied, one needs to decompose the particles into atoms to maintain accuracy.[50]

The difference between the QC's adaptive meshing method and the GP's decomposition method is: QC changes the FE mesh into atoms and GP changes particles into atoms. This difference is essential since GP particles are in the same material structure as the atomistic structure and have the same non-local constitutive behavior as atoms. However, in the QC method, "it is necessary to recalculate the ghost forces after each re-meshing step since the atoms/nodes experiencing ghost forces are changing,..".[36]

b. Dislocation propagation beyond scale boundary

In Ref. 48 and Ref. 50, 3-scale GP models were used for scale ratio k=2 and k=3, respectively, to compare with fully atomistic analysis for dislocation initiation based on the CN method (see Section III.A.1.). The result shows a high accuracy of the GP method in prediction of applied strain value for dislocation initiation and the dislocation evolution pattern. Furthermore, two special calculations are conducted to show the GP model has

energetic preference to dislocation nucleation and multiplication.[50] This is done by computing and then comparing the energy of crystal before and after dislocation nucleation. Interested readers are referred to Section 7 of Ref. 50.

For validation that dislocations can pass scale boundaries in a more general case in which surface effects are minimized, models with deep symmetric notches for both fully atomistic and 3-scale GP methods were designed as shown, respectively, in Figure 6(a) and (b). The fully atomistic model has 154206 atoms and the GP model has 24136 atom/particles of which 1290 are imaginary located around the interfaces. This simulation relaxed for 100ps.

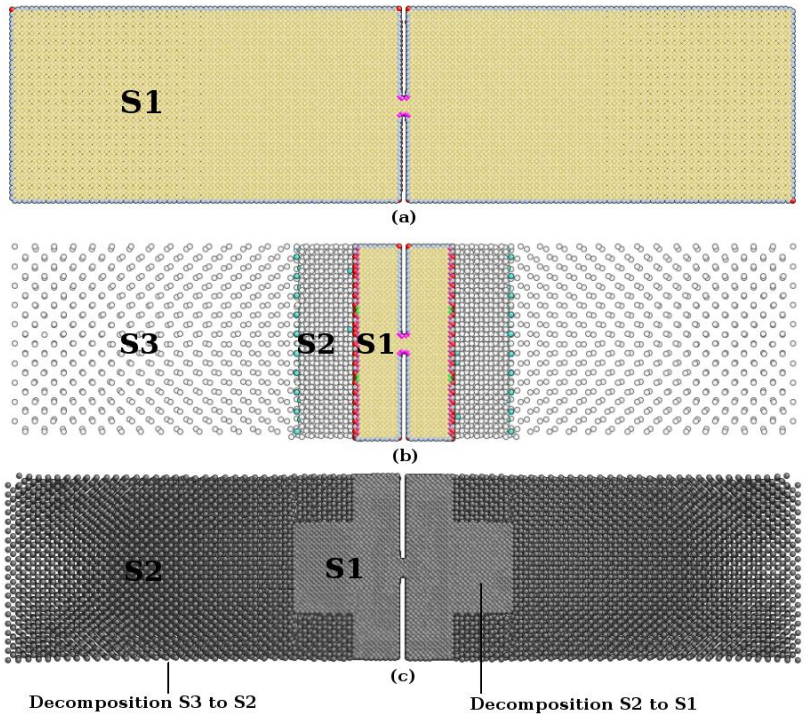


Figure 10. Overview of deep-notch models for validation of dislocation passing scale interface near the center of the nanowire, (a) fully atomistic model, (b) 3-scale GP model and (c) decomposition of higher scale into lower scale, especially decomposition of S2 scale into atoms along dislocation propagation path.

A partial second scale (S2) region along the dislocation path near the S1/S2 interface was decomposed at a strain of 12.83% (see Figure 9(c)). At the same strain, the

S3 regions were also decomposed into S2 regions, which make the validation simple. For simplicity, decomposition method 1 was used.

After the decomposition, the loading continued from strain 12.83% to about 19% and analyzed using the CN method. The results shown in Fig. 11(c) and (d) were compared to the CN of the fully atomistic simulation in Fig. 11(a) and (b). This figure depicts the CN of the atoms in the central symmetric plane of the model (i.e., near y=0). Those in yellow are a perfect FCC crystalline structure inside of the body; the blue, CN=13 and the white, CN=14 are atoms indicating dislocations. At 16.85%, dislocations had not yet passed the scale interface, however at 18.47% dislocation patterns show propagation across the scale interface, denoted by the vertical bold line. It is seen that, after comparison between the GP and MD results, the patterns are very similar; indicating that the decomposition method proposed in this work is effective for dislocation passing through the scale boundary. Some differences during the passing process, as shown in the inset chart on the right of Fig. 11 at strain of 17.65%, is reasonable due to the statistic distribution of a large amount of dislocation cores simultaneously passing through the interface.

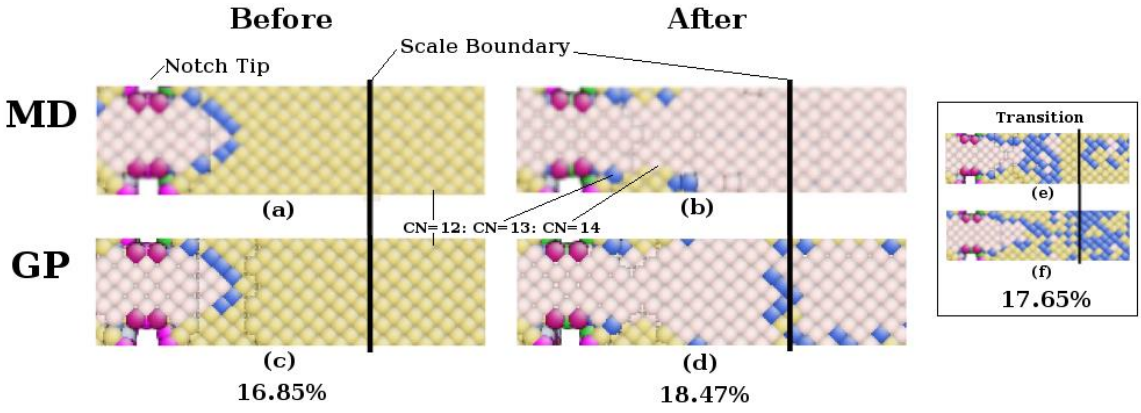


Figure 11. A comparison of dislocation patters between fully atomistic and the GP method before and after passing through the scale boundary.

c. Automatic Decomposition

In practice, a clever way to decompose a domain is to have an imaginary atomistic domain overlapping the real particle domain. In this way the imaginary atoms will follow the real particle deformation via their NLC. At a given point in time their

roles may reverse; the imaginary atoms will become real and the real particles will become imaginary, thus decomposition is realized! This method is simple to implement as an automatic decomposition technique and most closely resembles Decomposition Method 1 described above. Since the newly made real atoms will have a position based on interpolation of the formerly real particles they will need some time to relax to a more realistic structure, however because they followed the deformation of the particles, the time required is significantly short.

The domains chosen to be 'auto-duality' domains (ADD) require some kind of criterion that determines whether they should decompose or lump if already decomposed. In this work the temporal average of the maximum principal stress over the last ten timesteps of the ADD is used. The max principal stress for a given ADD is a function of the domain's virial stress.

$$\sigma_1 = \frac{1}{2}\left(\sigma_x + \sigma_y\right) + \sqrt{\left(\frac{1}{2}\left(\sigma_x - \sigma_y\right)\right)^2 + \tau_{xy}^2} \qquad (6)$$

This applies only for 2D models or sufficiently thin atomistic based models. A dimensionally invariant alternative would be to use the average potential energy of the domain. When using criteria such as these, appropriate values to use for controlling the decomposition etc. can be difficult to make. Usually a calibration model is required to determine a suitable value for the decomposition criterion.

An example copper plate model was made with a small hole in the center from which dislocations will nucleate when it is sufficiently loaded. In order to sustain a hole in an atomistic model the hole must large enough that the atoms cannot “see” or “feel” each other, so the diameter must be larger than the interatomic cutoff radius. In this case the hole diameter is made to be 10 Ang. To have the boundary conditions far enough from the hole the majority of the model is modeled with high scale S2 particles with atoms around the surface of the central hole going a depth of 10 Ang into the model. The entire height of the model along the Y direction is 145 A and 87 Ang in width; the plate thickness in Z is 43.5 Ang. The W1 imaginary domain initially extended into the S2 domain by a depth of the cutoff radius, but was extended in the X direction to be used as six separate auto-Duality Domains; three on either side of the hole. See Fig. 12.

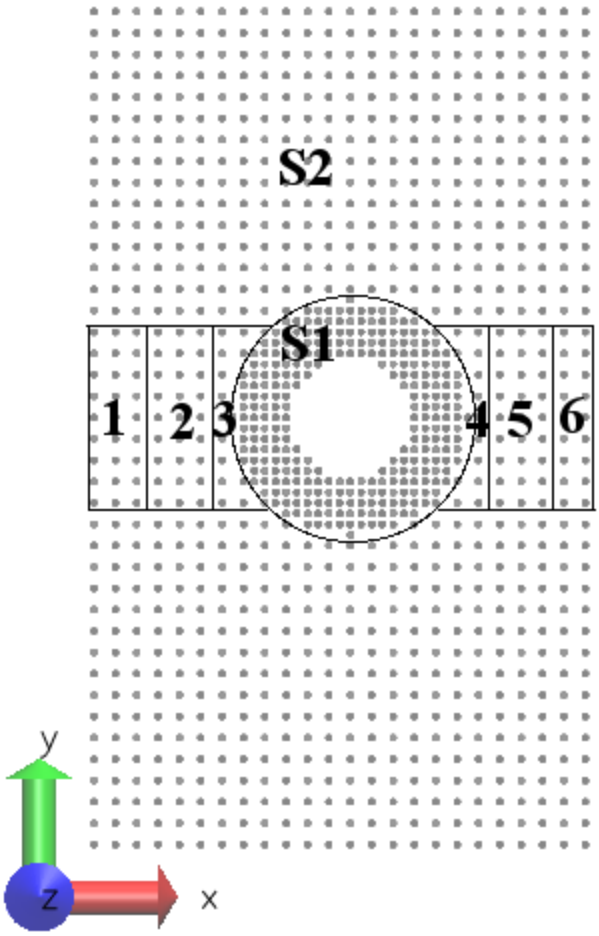
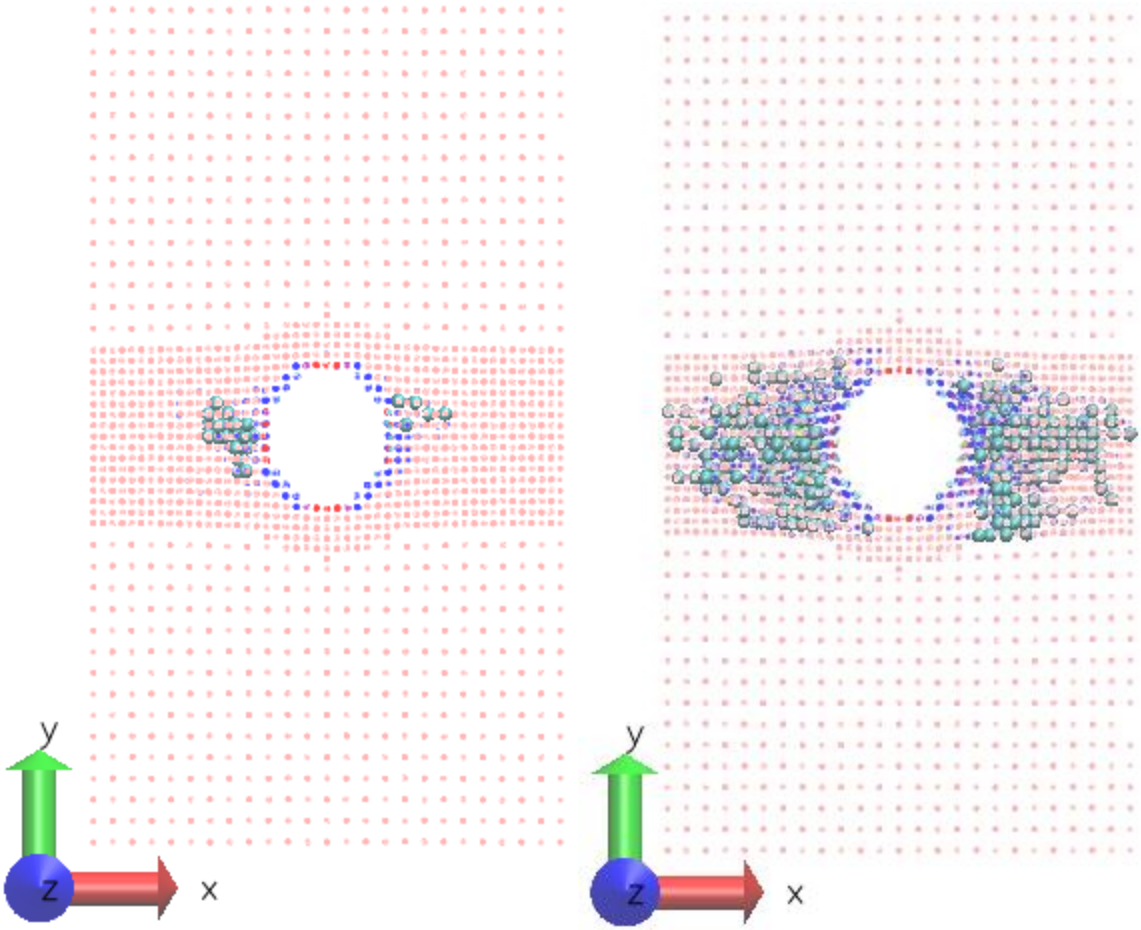


Figure 12. ADD Model with central hole.

The domains identified in Fig. 12 as 1-6 are domains that will be monitored for their maximum principal stress during the calibration simulation. When dislocations form in the S1 domain, the values in the neighboring domains will be recorded and used as decomposition values, so that the auto-duality simulation will decompose these domains at that value to allow further dislocation development to propagate horizontally into the newly made S1 domains. In this case the decomposition stress was set to 15.0 GPa. This value is low enough to ensure that dislocations are not present at the time of decomposition.



(a) (b)

Figure 13. Model configuration colored for Coordination Number (CN) of each atom and particle. The large light blue spheres represent dislocation cores with CN=13 at (a) 9.5% strain and (b) 10.5% strain.

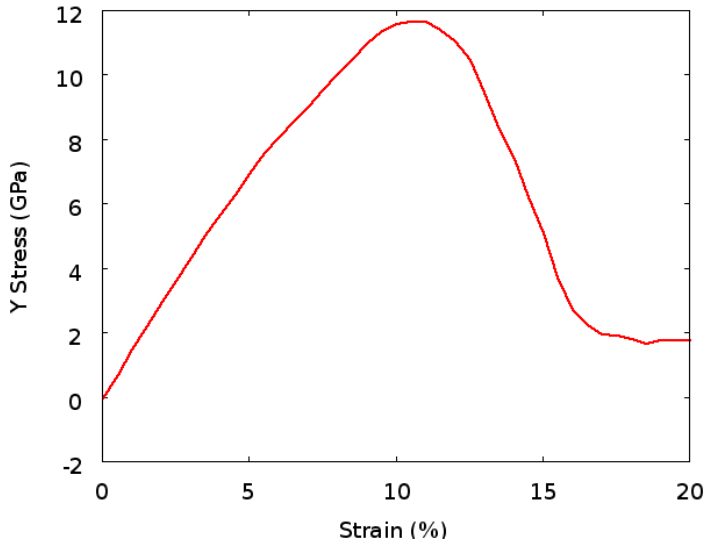


Figure 14. Loading stress strain curve for the example model, the two configurations of figure 12 are before and at the stress peak, respectively.

This process feature allows the GP model to capture the failure process at the atomistic level while maintaining higher-scale resolution far from the region of local failure phenomena.

B. Linking GP with FEA

In this section, we will show how the model size, which has been already enlarged by the GP method, can be further extended to the micrometer scale or even larger by the proposed GP-FEA multiscale method. This extension makes it possible to investigate the model size effects and satisfy the accuracy validation requirement by comparing the atomistically-based multiscale analysis with the well-known solutions of continuum theory.

1. Advantages of the GP-FEA Methodology

The GP method can save a large degree of DOF and computational resources by using generalized particles for domains with smooth deformation gradient. However, if the particle domain is too large the computational saving may not be sufficient due to time consumption of its non-local calculations. It appears natural for the GP-FEA method to take the advantage of FEA for connecting the high-scale particle domain to a FE mesh

by which large scale continuum level phenomena can be more accurately modeled. This new method links the continuum via finite element (FE) nodes with high-scale particles but not directly with atoms, as is the case for most existing multiscale analyses such as the quasicontinuum (QC) method. Two advantages for the change of connection location from the atomistic domain to the high-scale particle domain can be observed. First, since a high scale particle domain, n, has very large generalized lattice constant, a_n as shown in eqn. (3), the shortcoming of the DC method listed in Section I.B.2. can be avoided. Specifically, if the particle domain has a scale ratio k=3 and scale n=3, then following eqn. (3) we have $a_3=3^2a_0=9a_0$, it indicates that this replacement in the GP-FEA method can save about 2 orders of elements in the interface domain compared with the DC method. Second, as a consequence of moving the interface from the atomic boundary to the high-scale boundary, the distance from the external and the interface boundary to the atomistic domain is much larger. In turn, any variation or oscillation of the BC will have less disturbance on the atomic motion in the domain of interest, which increases the atomistic accuracy, avoiding instability and preventing the effects of artificial phenomenon such as the ghost forces from influencing the atomic domain.

In addition, for the atomistically-based multiscale analysis it becomes difficult to accurately connect the thermally active atoms to the quasi-static nodes of an FE mesh, thus zero temperature MD simulation is used.[35] However, inside the GP domain there is no FE nodes existed thus there are no constraints from FE nodes for the description of thermal activity in these low scales. This advantage is important since the atomistic domain is the most important domain which needs to have a high accuracy.

2. Model Structure and Design of Coupling subsystems

The GP-FEA model embeds an inner multiscale particle system within a surrounding FE domain continuum. This will reduce DOF of the system while not disturbing any important phenomena in the atomistic domain as it is "far away" from the interface between the GP and the FEA nodes and "more far away" from the external boundary. The method for how to link the high-scale particles to the FE nodes through iteration processes is schematically shown in Fig. 15. The structure of the GP-FEA model mainly consists of four regions with the ascending order from inside to the outside of the model: (1) An inner multiscale GP region. (2) An interface WF domain whose main role

is to determine the FE node inner boundary position according to the particle relaxation position after the last iteration. (3) an interface WG domain whose main role is to determine the generalized particle boundary position according to the FE node positions after the last iteration. (4) The outside FEA domain in which standard FEM is used. Two separate but coupling processes occurs alternatively in the two sub-systems, each involves two domains. The sub-system WF-FEA, short for domains of WF and FEA, control the FE node displacement based on WF and remote boundary node positions. The WG-GP, short for domains of WG and GP control the motion of the generalized particles. The calculation of the second sub-system is necessary as a part of the global loading iteration process. The WG and WF domains are defined before the loading but after the GP equilibration to prevent any residual stresses.

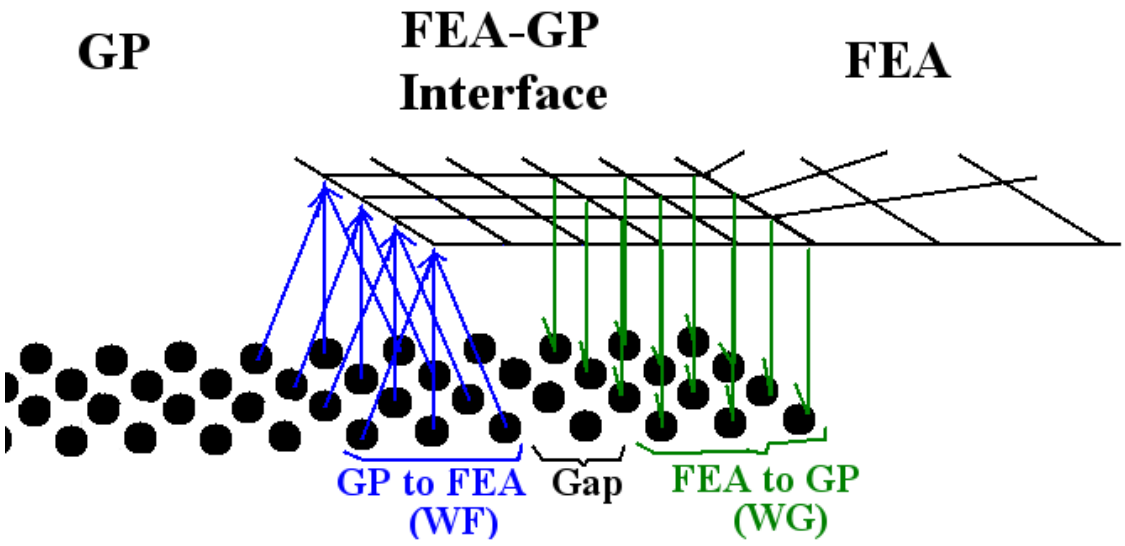


Figure 15. Schematic of the four main domains of a simple GP-FEA model. From the inner region to the outside in an ascending order: The GP domain, the WF and WG domains and the FEA domain.

3. The "Bottom-Up" and "Top-Down" iteration bridging scheme

The "bottom-up" transformation from particles to FE nodes, denoted by the up blue arrows, is through the particle-node overlapped WF domain and the "top-down" transformation from FE nodes to particles, denoted by the down green arrows, is through the overlapped node-particle WG domain. The function of WF and WG are similar to the

role of $W_{(n+1)image}$ and $W_{(n)image}$, (or W2 and W1 domains if n=1) in the GP model. The WF domain transfers data "bottom-up" by averaging particle's position $\overline{r_i}(i = 1, \dots l_n)$ to determine the position of the corresponding FEA node I in the WF domain by

$$\overline{R_I} = \sum_{i=1}^{l_n} \frac{\overline{r_i}}{l_n} \tag{7}$$

where l_n is the number of the particles in the NLC of the FE node I. Likewise, the WG domain transfers the data "top-down" by a certain decomposition of the FEA nodes' position to determine the particle position within the FE elements in the WG domain. The former and the latter processes are important, respectively, to control the deformation of the FEA domain and the motion of the GP domain during the iterative process. In fact, the deformation of the FEA domain is controlled by both the nodes at the remote boundary and the FE nodes in the WF domain; the motion of the GP domain is controlled by both the inside body of the GPs and the particles in the WG domain. To make the two separate but coupling processes of the system functional, it merits note that before any iteration of a load step all FEA nodes in the WF domain are fixed for the WF-FEA process and all particles in the WG domain are fixed for the WG-GP process. In turn, for the first process these nodes serve to be an inner boundary to control the deformation of the FEA domain under remote external loading, thus the GP displacement controls the upper scale FEA node motion in the WF domain to carry on the "bottom-up" transition.

On the other hand, the above controlled deformation process of the FEA domain will change the position of the FEA nodes which overlap the WG domain. In turn, these displaced FEA node coordinates will be used for determining the new coordinates of particles in the WG domain. The principle for assigning the new position of particles with the new coordinates of the FE nodes is based on the FEA isoparametric formulation.[51] The latter is simply to use the FE shape function matrix [N (ξ, η}] to determine the coordinates {x (ξ, η)} of any particle inside of the element with the coordinate matrix {X} of the FE element nodes. Here ξ, η are the natural (or non-dimensional) coordinates of the particle and will not change during the FE node displacements. Since after the iteration the node position {X} is known, shape functions are given then the new particle position can be obtained by the matrix product of [N] and {X}. Then the position of these

particles will be held fixed as the external boundary of the GP domain to start the new WG-GP sub-system process for its relaxations of particles and atoms. Thus, the FE node controls the particle motion in the WG domain to make the "top-down" transition realized. Note, shape functions can be linear or bi-linear, using the bi-linear functions is more accurate but also more complicated, the unique solution for bi-linear functions developed by Hua[52] to determine the new coordinates of these particles from the element node coordinates should be used.

Specifically, after the equilibration of the GP model and the FE mesh design (see Section 3.5.3) is completed all initial positions, x(j) and y(j), of GP particle j (j=1...n_j) and initial position X_L^J, Y_L^J of element node J (J=1...N_J, L=1..4) in the WG domain are determined. Where, n_j and N_J denotes the total number, respectively, of GP particles and FE elements in that domain, L denotes the node number of a given element which varies from 1 to 4 for the quadrilateral element. Using the inverse transformation method developed by Hua, one can find the natural coordinates ξ (x, y), η (x, y) for each generic particle, j, based on the node coordinates of the element that particle j belongs to. With the deformation process of the WF-FEA subsystem, the node coordinates X_L^J, Y_L^J change, the corresponding position, x and y of the particle j also changes. The new position x and y of the particle in each iteration is important to formulate the WG-GP external boundary for particle relaxation. It can be determined by shape functions [N] as (Remark: the superscript J of the element ID is dropped for simplicity)

$$\begin{Bmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{Bmatrix} = [N]\{X\}, \quad \{X\} = \{X_1, Y_1, X_2, Y_2, X_3, Y_3, X_4, Y_4\}^T \tag{8, 9}$$

$$[N] = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \tag{10}$$

Where

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta), N_2 = \frac{1}{4}(1+\xi)(1-\eta),$$

$$N_3 = \tfrac{1}{4}(1+\xi)(1+\eta),\ N_4 = \tfrac{1}{4}(1-\xi)(1+\eta) \tag{11}$$

The key here in determining the particle position in the WG domain, based on the position (or) displacement of the FE nodes, is to make the natural coordinates (ξ, η) constant so the distribution pattern of particles inside the element is fixed as required naturally for a given material domain.

It may be interesting to look at the difference of the WF domain from the WG domain in treating the relationship of FE node with the particles. For the former, this relation is quantified by eqn. (7). In practice, after the initial GP equilibration the first step is to find what particles is inside the NLC of that FE node I and determine their position vectors, $\bar{r}_i(i = 1, \ldots l_n)$. These particles' identification number (ID) will be constant during the whole deformation process (i. e., the particle constituents of the NLC for the FE node I is fixed). Each of these individual particles move during iteration, the spatio-temporal average of their position determines the new position of the generic FE node is through the equation (7). It is truly a lumping process but not a one-to-one fixed relation between FE node and particle. This is natural since the determination of the FE node positions in the WF domain is controlled by a relaxation process of the WG-GP subsystem. Thus, the particles should move to make the system, and particularly the particles surrounding those FE nodes, equilibrated under the fixed WG BC. This is a dynamic process so one needs to integrate Newton's equation of motion for the system by, say, the Velocity Verlet method.[19] However, for the WG the situation is completely different, the particles change its role from "master" in the WF domain to "slave" in the WG. Their positions are controlled by the element node positions which are determined by the motion of the WF-FEA subsystem. Obviously, the efficient way to make this control is to fix the relation of these FE node positions with the internal particle through the fixed natural coordinates ξ, η determined after the equilibration. In a sense particle position is solely determined by the FE node displacements through an interpolation function, the WG method for the top-down transition may be considered as a decomposition of the FE node displacements to a generic particle.

4. Numerical Algorithms

The solution for the deformation and failure of the system under a given load P is nonlinear; it uses incremental loading schemes for numerical convergence. For each

increment of loading, there are many iteration steps k (k=1... k_m) for all of which the remote external boundary of the FE domain is fixed. What changes between the iterations is the position of the inner boundary only, which is defined by the FE nodes in the WF domain. The iteration process is to make the deformation consistent between two successive iterations, such that the last two iterations are iterations are identical under a given error tolerance. This requires the WG-GP sub-system also has the identical BC between the two iterations. This, in turn, will guarantee the identical particle position in the WF domain and thus cause FE nodes to be identical since they are determined by the particle positions surrounding them.

To mathematically express this identical position requirement for a single incremental load step, suppose the symbol k (k=1... k_m) denotes the k^{th} iteration of the WF-FEA subsystem in that step, I (I=1,.. N_I) is the ID number of a generic node and N_I – the total number of the FE nodes in the WF domain. Symbol $\bar{u}^I$ denotes displacement vector for the FE node I, $\bar{u}^I_k$ and $\bar{u}^I_{k-1}$ denote that displacement vector is obtained, respectively, from the kth and (k-1)th iteration. The displacement difference, d^I, between the two iterations for FE node I is defined as the so-called L_2 norm as

$$d^I = \|\bar{u}^I{}_k - \bar{u}^I{}_{k-1}\| \tag{12}$$

Likewise, the average, u_{kavg}, of the FE node displacement norm in the WF domain is defined by the average of the norm for all FE nodes in that domain

$$u_{kavg} = \frac{1}{N_I}\sum_{I=1}^{N_I}\|\bar{u}^I{}_k\| \tag{13}$$

Thus, the iteration error between the two iterations is similarly defined as the L_2-norm of the difference between the displacement vectors of all FE nodes in the WF domain, normalized first by the total number, N_I, of particles and then divided by its average displacement u_{kavg}.[53] This can be written as

$$\varepsilon_{err} = \frac{\sqrt{\dfrac{\sum_{I=1}^{N_I}(d_I)^2}{N_I}}}{u_{kavg}} \tag{14}$$

The iteration process for the incremental load step at hand will not stop until this error is

less than the prescribed tolerance ε_0 , i.e,

$$\varepsilon_{err} \leq \varepsilon_0 \tag{15}$$

or reaches the maximum iteration number k_m which is much smaller than the prescribed number of time steps, t_m, for the relaxation of the subsystem WG-GP. This is natural since the WG-GP calculation to determine FE node positions for the WF-FEA subsystem BC is mainly a relaxation process to make the system equilibrated.

C. Shortcomings

Most of the shortcomings of the GP method stem from violations of the underlying assumption: that for sufficiently small deformation gradients, the deformation can be approximated by sampling more distant locations; i.e. using a coarse mesh or larger particles with the same density; this is the Cauchy-Born rule.

D. Summary

The current incarnation of the GP method is seen to have a natural interface between scale domains where physical variables such as displacement and force may smoothly pass from one scale to another. Atomic phenomena such as dislocations may be passed into higher scale domains via the scale-duality concept[50] by decomposing particles into their constituent atoms. In addition, it has been mathematically proven[19,48] that all calculations in the particle domain can be conducted at the atomistic domain scale using the same potential, parameters and numerical algorithm as is used for the model's atomistic scale. Thus the GP method is essentially an extension of MD and can be easily delivered to applications by modifying existing MD codes. Readers interested in the details of the unique features of GP are referred to the literature.[18,19,48,50]

ACCURACY VERIFICATIONS OF ATOMISTICALLY-BASED MULTISCALE ANALYSIS WITH CONTINUUM SOLUTIONS BY THE GP-FEA METHODS

Most accuracy verifications of atomistically-based multiscale simulations adopt the comparison with the result of fully atomistic simulation due to the difficulty of experiments in getting high-resolution data at angstrom and nanometer scales. While accuracy verification with the well-known continuum field solutions is important for engineering applications, it is seldom to appear due to unknown model size effects and the limitation of many multiscale methods in developing a minimum model size necessary to make the comparison meaningful. Among these issues, the key is to develop new multiscale methods which can produce sufficiently large models. The key issue is addressed in this work within the framework of the GP and GP-FEA method where GP is short for the Generalized Particle Dynamics Method.[48] The GP-FEA is a further development of the GP which can make the model size as large as needed. This new method links the continuum via finite element (FE) nodes with high-scale particles but not directly with atoms, as is the case in the quasicontinuum (QC) method, etc., to avoid artificial effects such as ghost forces. In addition, its model structure and design, the ``bottom-up'' and ``top-down'' bridging scale scheme and the iterative process show convenience for applications. This method is exemplified by simulations for a central hole in a two-dimensional specimen under tensile load. The agreement between the continuum solution and simulation demonstrates that the GP-FEA method is a powerful tool for investigating model size effect and comparing atomistically-based simulation with continuum theory to validate the accuracy of multiscale analyses.

A. Introduction

It is by now widely recognized that materials are inherently of multiscale, hierarchical character.[48,54] Material behavior should not be considered as monolithic properties that manifest only at phenomenological levels, as historically taught. Rather, important properties and material responses can arise at a myriad of length scales and the

phenomenological behavior of the continuum follows from their atomic and microscopic structures. The significance of multiscale analysis naturally follows this understanding and it brings the hope that new concepts and methods can be developed based on low scale structures, behaviors and physics. In fact, in the past 20 years one sees a wide scope of interests and intensive research activities in developing multiscale methods. To name a few, quasicontinuum (QC),[55] CADD[56-58] are examples for the direct coupling (DC) methods between atoms and FE nodes of continuum in the concurrent multiscale analysis. Specifically, QC uses the Cauchy-Born rule to transfer atomistic energy to the strain energy density of FE analysis and CADD, short for coupled atomistic discrete dislocation dynamics, combines atomic scale analysis with discrete dislocation analysis in a continuum to perform multiscale modeling. Here, approaches that relate atoms and finite element nodes in a one to one manner, or through a form of interpolation, will be referred to as DC methods. According to this definition, most of the existing multiscale methods belong to the DC methods. On the other hand GP[18,19,48] and ESCM[32] belong to the non-DC methods category. The embedded statistical coupling method (ESCM) employs statistical averaging over selected time intervals and volume in atomistic subdomains at the MD/FE interface to determine nodal displacement for the continuum FE domain. The other non-DC method, called the generalized particle dynamics method (GP), is proposed to use constant material neighbor link cells (NLC) at the interface region to mutually transfer information from the bottom-scale up or from the top-scale down to quantitatively link variables at different scales. Readers interested in the details of classification, historic development, and applications of multiscale analysis are referred to "Multiscale Analysis of Deformation and Failure of Materials".[19]

Looking back on the developments of multiscale analysis in the past two decades, two basic issues for extending engineering applications of concurrent multiscale simulations can be addressed. The first one is how to quantify the accuracy of atomistically-based multiscale simulation and the second one is how to enlarge the model dimension to the minimum size necessary to make the model realistic.

The first issue is obvious since experimental accuracy verification of the multiscale analysis at atomistic/nano scale is difficult even when using high-resolution experimental methods. The most popular approach so far is to compare the results of

multiscale analysis with a fully atomistic simulation method such as molecular dynamics (MD). Examples can be found for dislocation at notches[48] and dislocations passing through scale boundaries.[50] Among those efforts, there are two notable verification studies. Curtin and Miller[36] used a one-dimensional (1D) spring model to compare the scale transition regions of the various methods such as QC/CLS,[59] QC-GFC[60]/FEAt[61] and CADD with the fully atomistic model. Their results confirm that besides the QC-GFC and FEAt methods most multiscale models do not truly meet the requirement for a seamless transition at the interface between atomic and FE domains. This discontinuity is caused by the intrinsic incompatibility of constitutive behavior between material models being coupled together at the two sides of the scale interface which causes non-physical phenomena at the interface, including the so-called ghost forces.[60]

Specifically, the behavior of the continuum on one side of the scale interface is local but that of atoms on the other side is non-local in nature. Here, non-local constitutive behavior indicates that the force at any atom depends not only on those atoms closest to it, but also on the atoms nearby, which are not direct neighbors with the atom at hand but within its neighborhood through interatomic forces; local behavior indicates that the force (stress) of a material point depends only on the deformation gradient (strain) at the same point of the continuum. Due to this local behavior, nodes in the continuum region cannot feel interactions from other nodes nearby as their atomic counterparts in the atomistic region. Based on the mechanism of this incompatibility, a dead ghost force correction method of QC, (i.e., QC-GFC) was developed[60] to use the ghost force as a dead force to recalculate the result. This correction method allows this approach to exhibit a seamless transition at the scale interface, as mentioned in the previous paragraph. It is therefore necessary to recalculate the ghost forces, in some cases, say, after each re-meshing, since the ghost forces may change.

The other verification performed is the benchmark computation of 14 models carried out by Miller and Tadmor.[53] The fully atomistic simulation is the benchmark against which one compares the multiscale models. Here, the corresponding atomistic solution is considered as the exact solution. The test used a common framework to examine the accuracy and efficiency of these methods using a test problem of single crystal aluminum containing a dipole of Lomer dislocations. While this test is significant,

the quantitative criterion for accuracy comparison used the global error which covers displacements of all atoms in the simulation system. The shortcomings of the method for accuracy verification are twofold: First, it is difficult to judge how accurately this method describes the local behavior of the material, say, at the interface which makes it difficult to guide improvement of the interface design. Second, it can also be hard to make a judgment for how accurately it describes the continuum behavior for engineering applications. In fact, the benchmark test indicated itself that slight error in global energy estimation can lead to profound effects on the resulting dislocation motion and, in turn, the continuum behavior.

The second issue related to a model size requirement needs some explanation even though it is a common problem that appears frequently during model design. This issue can be addressed from the following four aspects. Firstly, accuracy verification for low scales is important to find the deformation mechanisms, such as crack nucleation in fatigue, dislocation patterns in fatigue and creep, the inherent inhomogeneity of plastic deformation, the statistical nature of brittle failure and the effects of size, geometry and stress state on yield properties.[62] To make this finding meaningful, however, one must link these low-scale dynamics to material behavior and be characterized in the continuum scale for applications. This requires a relatively large continuum model size. Otherwise the approach of predicting material behavior by implicitly averaging atomistic/microscopic dynamics may not be valid. Secondly, if the model size is small the boundary conditions (BC) may likely affect the local fields of forces and displacements which are near the regions of interest. In turn, it will change the behavior of highly important domains (e.g. interfaces, crack-tips and flaws). In most practical cases, BC cannot be perfectly maintained and will have oscillatory and random perturbations. Various BCs can be accepted if they are sufficiently far away from the interested regions following the concept of Saint Venant's principle.[63,51] Otherwise, the obtained low-scale phenomena observed can be qualitatively different which can cause the instability of atomic motions, microstructural evolutions and unexpected material failure. Thirdly, some mathematical solutions for the continuum require the medium to be infinitely or sufficiently large to obtain results close to the analytical solution such as the crack-tip solution of linear elastic fracture mechanics (LEFM). In this case, model size

must not be small for a reasonable result. The fourth aspect is that for microsystems and nanotechnology, the model size should be equal/larger than micrometers or at least being sub-micrometers so the problem of micro- or nano-sensors/activators can be more accurately simulated. For nanotechnology, this is true for some designs since nanotubes, nanofibers, etc. need to be assembled and embedded in a matrix which has a certain size requirement. Thus, investigating the model size effects and choosing a minimum model size necessary for the accuracy requirement is essential.

This work is our first effort to address the issue of model size effects on accuracy. It is obvious that the premise for investigating this issue is to have new methodology for developing sufficiently large size models such that the effects can be investigated systematically by varying the model size. Thus, this paper focuses on the introduction of the new methodology within the framework of the GP and GP-FEA method, where the GP-FEA is a further development of the GP method. This new method can make the model size as large as needed. The use of this new methodology to investigate the model size effects will be exemplified in future works. To address the first issue, accuracy verification is conducted by a comparison of the GP-FEA simulation result with a classical solution of continuum theory. This classical problem is a two-dimensional elasticity solution of a specimen with a central hole under tensile load.[64] This solution from elasticity theory is accurate but using it to verify the accuracy of atomistically-based simulation faces new challenges. Firstly, the explicit expression needed to be modified so the direct comparison can be available. Secondly, the data process for the atomistic scale needs to be done correctly which should involve a group of atoms near a continuum point not a single atom in that position. The third challenge concerns how to estimate the error by the comparison. All these issues will be addressed through the example of a 2D hole specimen.

The Chapter is organized as follows: Section B will introduce the proposed GP-FEA method in detail. Section C will extend the elasticity solution to an explicit displacement expression around the hole of a 2-D plate specimen under tension and then make a comparison with the multiscale simulation. The paper is ended with summary and conclusions in Section D.

B. Model Development

The model of the GP-FEA simulation includes two inter-connected parts: one is the GP model consisting of multiscale particle domains and the other is continuum domain made of an FEA mesh. The first of three operations is to develop the GP model. It is of primary importance, as it includes the atomistic domain where the interesting phenomena are to be observed. The second is to clearly define the interface between the GP and FEA regions. The third is to develop the FE mesh starting from the designed interface, covering the remaining part of the GP model and extending outward to a large model size terminated at the external boundary. These three steps will be discussed in detail using the plate with a hole as an example in the following subsections.

1. GP Hole Model development

The GP model usually consists of several generalized particle domains with different scales. In the example of an iron plate with a hole shown in Fig. 16, the GP model is designed to have three scale domains. Scale-1 (S1) with scale $n=1$ is the atomistic scale with a hole in the center with diameter of 4 nm. Scale-2 (S2, $n=2$), the second particle domain, surrounds the atomistic domain whose outer boundary size is about 20.2 nm in width and 42.2 nm in height. Scale-3 (S3, $n=3$) - the third particle domain, surrounds the S2 domain space whose size is 36.3 nm in width and 80 nm in height with the scale ratio k equal to 2. Following eqn. (3), the generalized lattice constant for S2 and S3 are, respectively, $a_2=2a_0$, $a_3=4a_0$ and the corresponding number of atoms that a particle represents for S2 and S3 are $l_2=8$, $l_3=64$. The thickness of this GP model is about 4.5 nm. The translational DOF number of the GP model is 292,488 (3x97,496). If the model is fully filled by atoms, the DOF will be 3,396,384 (3x1,132,128), i.e., 11.6 times larger than the GP's. The bridging of bottom-up and top-down between the successive scales is accomplished by imaginary domains $W_{(n+1)image}$ and $W_{(n)image}$ that overlap, respectively, the real particle domain (n) and $(n+1)$, as introduced in Section A and discussed in the first paragraph of Section B. More details of the specifics about their design, function and purpose can be found in the literature.[19,48,50]

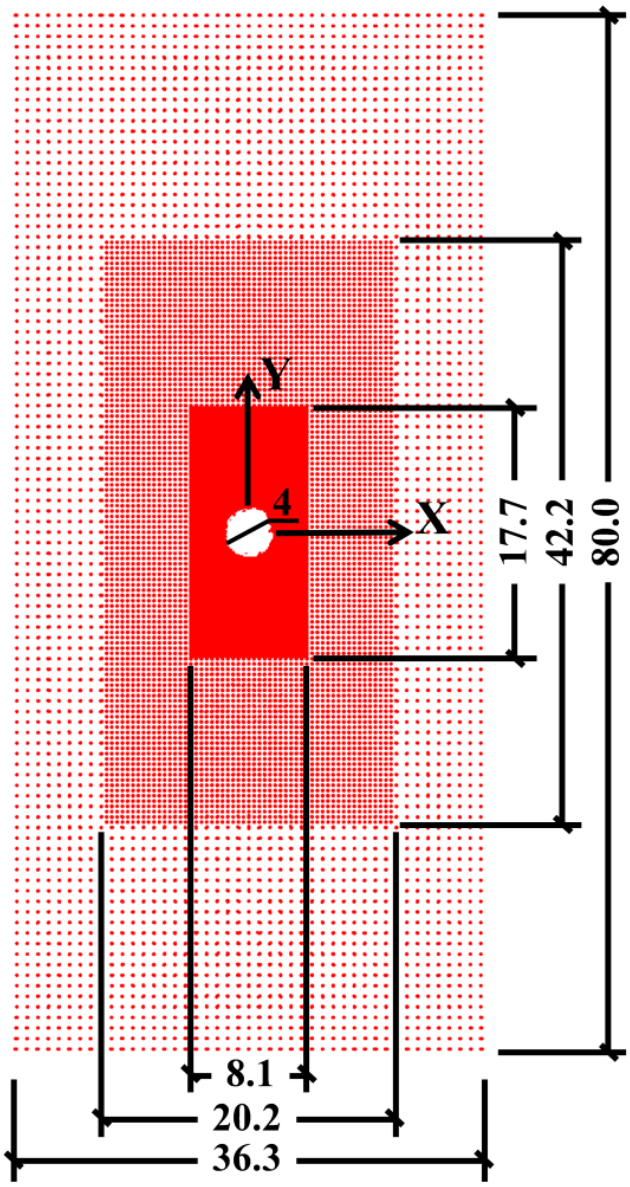


Figure 16. Schematic of the three-scale GP model for a thin plate with a central hole of 4 nm diameter. The loading direction [-110] is along the Y-axis and the X-axis, transverse to the loading direction, is along [110].

2. GP--FEA interface design

An exact interface design must first be drafted before developing a mesh for a GP model. The general design for the interface can be seen in Fig. 17. As mentioned in Section II.B.3. the function of WF and WG is to transfer data “bottom-up” and “top-down” to determine the FEA inner node positions and the outer GP particle positions, respectively, at the interface. Figure 17 shows the details of the ideal interface design.

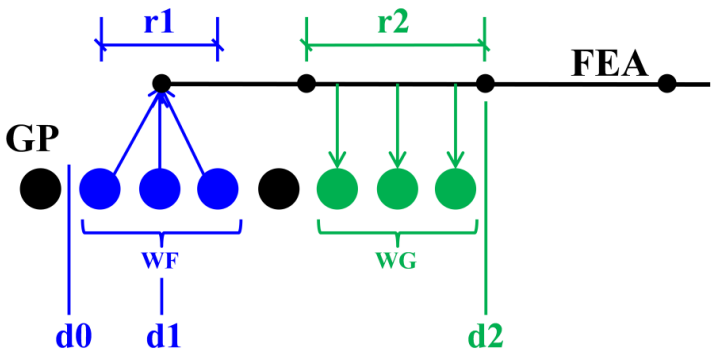


Figure 17. GP-FEA interface design. Particles in WF average their positions to determine the position of the FE node which are within r1 in the WF domain. As a result, it produces a displacement to the FE node at position d1. The WG particle positions, within r2 in the WG domain, are determined by interpolation with in the FE element.

The depth of the WG domain, r2, from the edge of the GP model, d2, as well as the WF domain width, r1, should be equal to the inter-particle cutoff radius as it follows from the length scaling rule of the GP method shown in eqn. (1). Since Mendelev's Embedded Atom Potential (EAM) is used whose inter-atomic cutoff radius is 0.53 nm,[65] by eqn. (1) the inter-particle cutoff radius for S3 when $k=2$ is calculated to be $a_3 = 2.12$ nm ($=2(3-1) \times 0.53$ nm). The nodes of the inner boundary of the FEA mesh should align with the center of the WF domain which is shown in the line d1 of Fig. 17 and is determined by the particles' position average in the WF domain. The depth of the interface, found by subtracting (d2-d0), is important for the scale bridging, and should be larger than two times the inter-particle cutoff radius to allow for a "gap" between WF and WG. This gap prevents undesirable feedback between WF and WG thus hastening the interface convergence, the recommended gap size is the inter-particle cutoff radius. The finite element size in the interface should be uniform about the size of the inter-particle cutoff radius and gradually increase in size farther from the GP domains. Figure 17 shows the interface for the 1D case. This profile is extended all along the edge of the GP model, in a sense tangentially, as illustrated in Fig. 18.

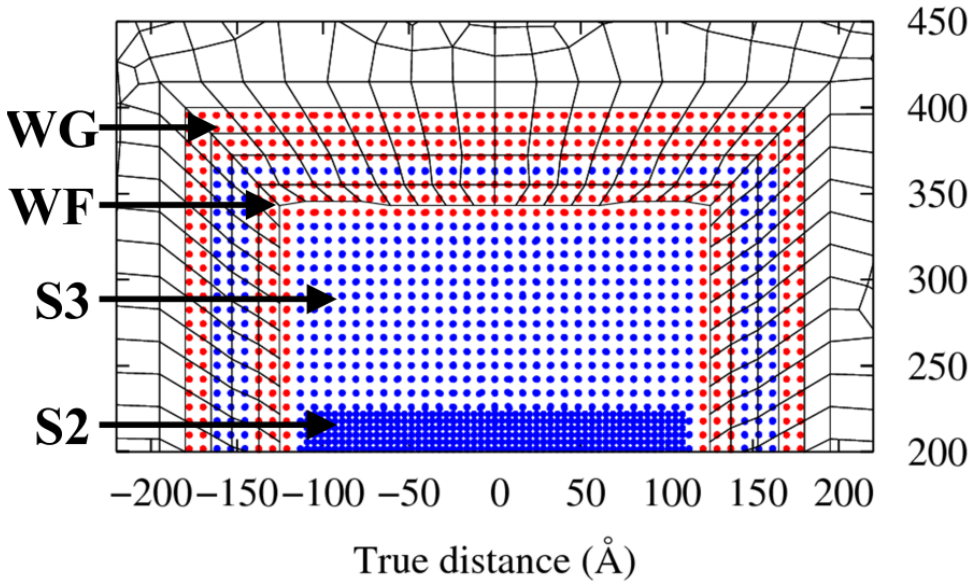


Figure 18. Continuous WF domain design and its relation with high-scale particles and the WG domain for the GP-FEA model interface.

3. FEA Mesh generation

Needless to say, code development is essential for wide applications of multiscale analysis. The GP-FEA method includes two source codes, Multi-Input57.f90 and FEA19.f90, which respectively, directly controls the GP simulation, including the WG-GP subsystem, and couples the calculation of the WF-FEA subsystem. Its corresponding input file is inp57f19.sh. Here, the numbers denote the code version for the GP and FEA trial. The structure and its related function of these codes are complicated and will be introduced elsewhere. Here, we will introduce the constitution of the input file FEAxx.inp. This file will be a part of the main input file but it is specifically designed for the FEA calculation parameters. In other words, when the FEA routine runs it will read this GP-FEA mesh input file. To develop this mesh input file, any FE program, such as ABAQUS, Ansys, etc. may be used to generate the FEA mesh based on the model size, the WF internal boundary and the interface dimensions. The process for the generation and its required format is described in Appendix C.6.

It is important that the GP model be equilibrated before it's connected to the FEA mesh to prevent residual stresses from manifesting at the interface. In the simulation, the NPT ensemble at 300 K was usually used for the equilibration if the system operates in

room temperature. After this separate equilibration the FEA mesh is connected to the GP model for the loading procedure. Here, the FEA mesh is added to the GP hole model such that the entire model size is about 200 nm wide and 500 nm high, nearly an order larger than the GP model. After the geometry has been sketched with the coordinate origin being the same for the GP model and FE mesh, it is best to partition the part around the interface so that it can be seeded to have a uniform element size and shape for both WF and WG domains. Since the element size at the interface is designed to have an edge length equal to the inter-particle cutoff radius, it will create two FE elements overlapping the GP model along the normal direction of the inner FE boundary. It is important to carefully seed the edges of the FE model for nodes by keeping a constant size element at the interface for both WF and WG-FEs then gradually increase the element size away from the interface.

C. Verification of the GP and GP-FEA multiscale methods with Elasticity Solutions for a plate with a central hole under tensile loading

1. The scheme for the comparison of atomistically- based simulation with a continuum solution

While using the classical solution of a plate with a central hole under tensile loading to verify the accuracy of multiscale simulation is a good idea, the difficulty is that the continuum field is very different from the particle field. Besides the difference in their constitutive relations as mentioned in Section A, the difference of the continuum from the non-continuum particle type structure makes the data comparison difficult. This makes the solution expression quite different in terms of the variables and the format. For instance, the elasticity solution for a central hole with radius, *a*, under the remote tensile boundary stress, *S*, is given by the following stress component form in the polar coordinates of radius, *r*, and the angle, θ, measured from the vertical loading axis, *y*:

$$\sigma_r = \frac{S}{2}\left(1 - \frac{a^2}{r^2}\right) + \frac{S}{2}\left(1 + \frac{3a^4}{r^4} - \frac{4a^2}{r^2}\right)\cos 2\theta \tag{16}$$

$$\sigma_\theta = \frac{S}{2}\left(1 + \frac{a^2}{r^2}\right) + \frac{S}{2}\left(1 + \frac{3a^4}{r^4}\right)\cos 2\theta \tag{17}$$

$$\tau_{r\theta} = -\frac{S}{2}\left(1 - \frac{3a^4}{r^4} + \frac{2a^2}{r^2}\right)\sin 2\theta \tag{18}$$

This solution was obtained by Prof. G. Kirsch in 1898. While it is derived for an infinitely large plate it has been well confirmed many times by strain measurement and by the photoelastic method for plates of a finite size. In 1907, Timoshenko proved that "If the width of the plate is not less than four diameters of the hole the error of the solution in calculating $(\sigma_\theta)_{max}$ does not exceed 6 percent."[64] The width of the model with the central hole in this work is 200 nm and the hole diameter is 4 nm, the ratio of the width over the diameter is about fifty, thus the accuracy is reasonable for this comparison.

On the other hand, the stress in the atomistic field is the so-called virial stress defined as

$$\underline{\sigma}^{\alpha} = \frac{1}{\Omega^{\alpha}}\left[-m^{\alpha}\bar{v}^{\alpha}\bar{v}^{\alpha} + \frac{1}{2}\sum_{\beta(\neq\alpha)}\left(\bar{f}^{\alpha\beta}\right)\left(\bar{r}^{\alpha\beta}\right)\right] \tag{19}$$

Here, the symbol "-" under a letter denotes a second order tensor and above a letter denotes a vector. In the first term of the right part in eqn. 19 there are two vectors of velocity, $\bar{v}$; in the second term, the two vectors are force vector $\bar{f}$ and the position vector $\bar{r}$. The superscripts α and β denote the atoms and Ω -- the volume of the atom at hand. Eqn. 19 shows the stress at atom α depends not only on those atoms closest to it, but also on the atoms, β, within its spherical neighborhood of cutoff radius through interatomic forces. This definition shows its nonlocal behavior. The different definition between the stress defined by a continuum and the atomistic field makes direct comparison difficult. On the other hand, the definition of displacement is clear and unified for both continuum and the atomistic field if a certain volume average can be conducted for the atomistic field. Unfortunately, to the authors' knowledge there is no explicit expression in the public domain for the displacement field around a central hole of a tensile specimen. These expressions are based on the solution of eqn. 16 to 18. The final results are given as follows:

$$u(r,\theta) = \frac{S}{2Er^3}\left\{(1-\nu)r^4 + (1+\nu)r^2a^2 + \left[\left(r^4 - a^4\right) + 4a^2r^2\right]\cos 2\theta\right\} \tag{20}$$

$$v(r,\theta) = -\frac{S}{2Er^3}\left\{\left[(1+\nu)r^4 + 2a^2r^2(1-\nu) + \left(1+\nu a^4\right)\sin\theta\right]\right\} \tag{21}$$

These expressions will be used to validate the simulation accuracy of the proposed GP-FEA multiscale analysis.

To make the GP calculation result of any generic point, *s*, on the plate face comparable with this continuum solution, the average displacement of atoms within a cylinder centered at that point with a small radius and extending through the entire thickness should be used. Here, we use 0.3615 nm for the cylinder radius which includes about 150 atoms on average for S1 and 20 particles for S2 per cylinder. All the data is placed in a file whose strain level is indicated. From that file one extracts the average displacement values from each of the cylinders specified. The obtained average displacements of the atoms/particles will then be compared with the corresponding data calculated from the analytical solution.

2. Result comparison for pure GP and GP-FEA model

The GP-FEA model developed in Section B.1. was loaded along the Y direction with periodic BC in the thickness direction. A barostat was used during loading to reduce the transverse stress average to zero to better match the conditions for the analytical plane stress solutions. Figures 20 to 22 show the angular displacement u_θ and radial displacement u_r by both the GP and the GP-FEA method along a circle with radius of 8 nm. The angle θ sweeps from the 0 degree along the Y-axis to 90 degrees along the X-axis, i.e., varies in the first quadrant. All the data are recorded at the strain of 2% -- 3% which is in the elastic range for the single iron crystal.

The corresponding result by using the analytical expressions 20 and 21 are also given with the key title of "Analytical". In the analytical solution, the following elastic constants are used: *E=224* GPa and *v=0.3*. However, it is found that any change of these constants has but a minor change on the results, which is consistent with elasticity theory. For the GP-FEA simulations, results for two WF design schemes shown, respectively, in Figure 18 and 19 are also given. From the comparison, it is seen that all the atomistically-based multiscale simulations listed in Figures 20 to 22 have sufficient accuracy in comparison with the analytical solution. It should be emphasized that agreement is along a circle with radius of 8 nm which crosses both S1 and S2 regions. The consistency of the simulation in both of the two scales with the analytical solution indicates, implicitly, that the transition between the scale boundary and the interface between the FEA and GP domain is sufficiently smooth, while a detailed direct verification at these boundaries should be done in the next step. The obtained result so far makes one confident to use the

GP and GP-FEA methods for middle size and large size models, respectively, for the sub-microsystem and microsystem. In addition, it is seen that the accuracy of the non-continuous (or separated) WF design shown in Fig. 19 is a little better than the continuous design of Fig. 18. The non-continuous one are distinguished in Figs. 20 to 3-7 with the symbol "GP-FEA model b" from the triangle of "GP-FEA model a". This may offer some clues for further improving the design of the WF and WG domain by developing design software to optimize.

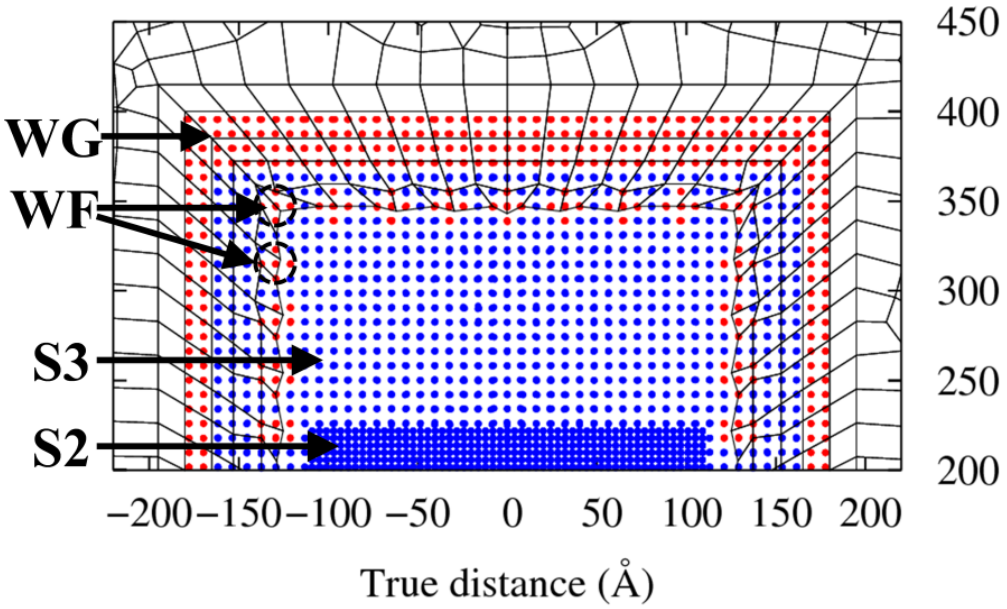


Figure 19. Discontinuous WF domain design and its relation with high-scale particles and the WG domain for the GP-FEA model.

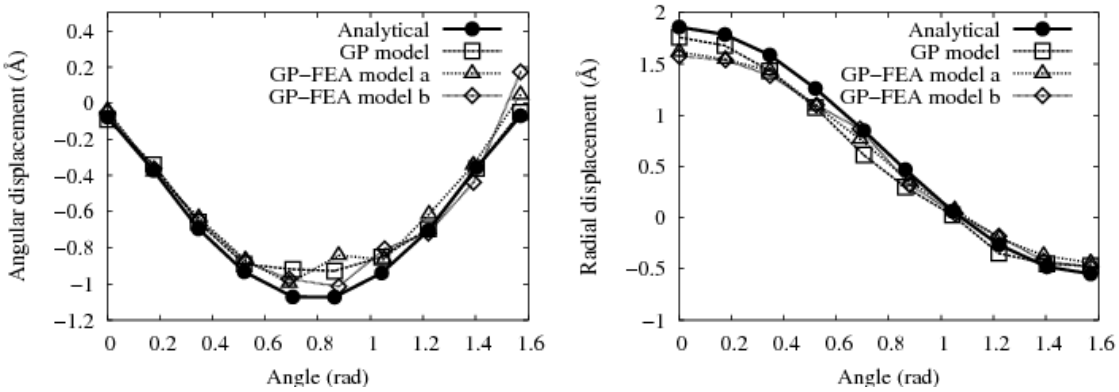


Figure 20. 2% strain displacement comparison of simulation-obtained and formulae-calculated angular displacement u_θ and radial displacement u_r along a circle with radius of 8 nm in the first quadrant for a single iron crystal.

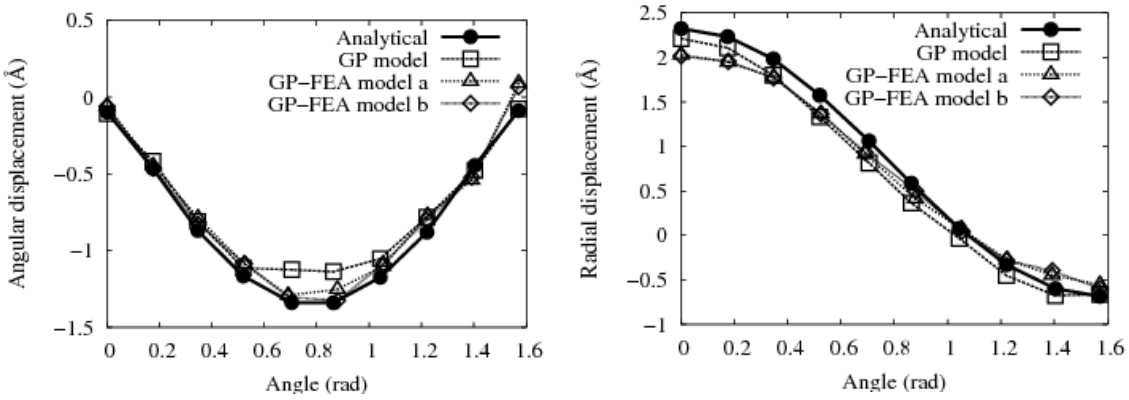


Figure 21. 2.5% strain displacement comparison of simulation-obtained and formulae-calculated angular displacement u_θ and radial displacement u_r along a circle with radius of 8 nm in the first quadrant for a single iron crystal.

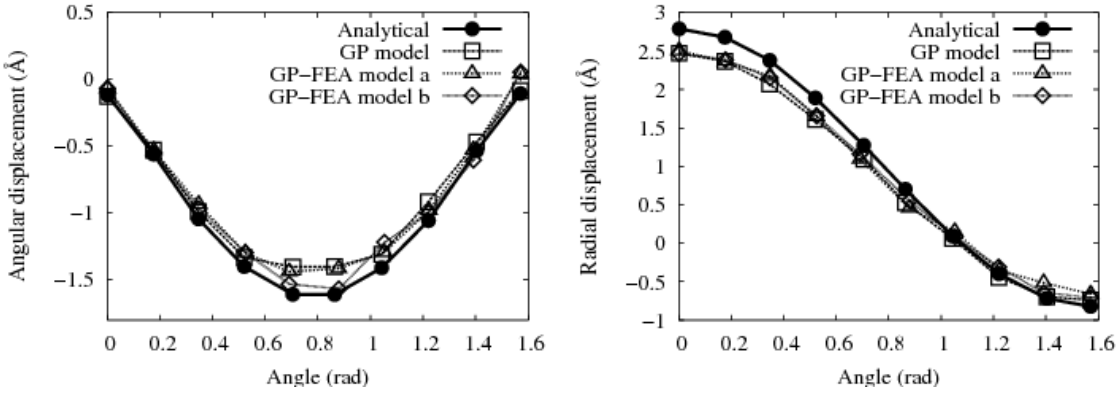


Figure 22. 3% strain displacement comparison of simulation-obtained and formulae-calculated angular displacement u_θ and radial displacement u_r along a circle with radius of 8 nm in the first quadrant for a single iron crystal.

D. Summary and Conclusions

Two basic issues for extending applications of concurrent multiscale simulations are discussed in this work. They include how to quantify the accuracy of atomistically-based multiscale simulation and how to enlarge the model size to the minimum necessary to guarantee the accuracy. These solutions are discussed with the GP and GP-FEA methods. The GP-FEA method, proposed in this work, is a new multiscale method which can make the model size as large as needed in the microsystem. In addition, it links the continuum via FEA nodes with high-scale particles, not directly with atoms, as is the case with other methods such as the QC method, to avoid artificial effects. Apart from the

conventional verification method with the full atomic solution (e.g. MD), a classic elastic stress solution of a two-dimensional specimen with a central hole under tensile load is extended in this work to its displacement distribution. This provides an effective tool for accuracy verification of the GP and GP-FEA multiscale methods by comparison of their simulation data with the analytical solution. The result of the comparison is encouraging. Main conclusions from this work include:

- The GP-FEA model embeds an inner multiscale particle system within a surrounding continuum FE domain. It moves the atom-FEA interface of the DC method far away to the particle-FEA interface. This greatly reduces DOF of the system while not disturbing any important phenomena in the focused atomistic domains, thus the artificial forces and deformation caused by, say, ghost forces can be avoided
- It should be noted that the elastic analytical solution is obtained under the condition that the width of the specimen be larger than four times the hole diameter to keep the error of the solution, e.g., $(\sigma_\theta)_{max}$ from exceeding 6 per cent. Our design satisfies this requirement. However, model size effects are problem-dependent. It may relate to material property, environmental conditions, the variables involved, the answer evoked, etc. Thus, it is hard to get a general answer analytically. In many cases one should carry on numerical simulations for models with different size to find the minimum necessary for accuracy.
- The satisfactory agreement between the displacement data obtained by the proposed GP-FEA methods with the classical analytical solution establishes a foundation to use these multiscale methods to investigate model size effects. In the parallel works we will demonstrate that the GP-FEA method is a powerful tool to identify the size effects and some new concepts such as critical model size, below which will cause serious error and above which may not be necessary under a given error tolerance.
- Practice has taught us that the extension of multiscale analysis to more applications requires essential development of computer code. During this work GP and FEA coupling source codes, two input files related to the modeling geometry and simulation control of the GP and the FEA have been developed.

While codes for model generation have also been developed, models still require some manual work to design transition domains such as WF, WG, W_n and W_{n+1}. This shortcoming will be overcome such that these domains can be formed automatically after the boundary line and its width and depth are given, etc.

ACCURACY VALIDATION WITH LEFM SOLUTION AT CRACK-TIPS: CONCEPT OF CRITICAL MODEL SIZE

Recently, ongoing multiscale research uses atomistic-based modeling for the simulation of crack-tip behavior to enhance the fidelity of predictive models. While there is a lack of verification analyses that the atomistically derived laws and information accurately reflects the originating atomistic results, some data by this approach show discrepancy from experiments. Naturally, this kind of simulation requires a relatively large model size for implicitly averaging atomistic/microscopic dynamics into continuum and avoids artificial boundary effects on the fields that are of interest. The requirement is not usually satisfied since most existing multiscale models are limited from developing sufficiently large model size. This issue is addressed in the present work. Firstly, a particle-FEA coupling multiscale model, called the GP-FEA method, is introduced which can make the model size as large as needed. Eight models with characteristic dimensions ranging from 120 nm to 5μm are developed for investigating model size effects on displacement fields of a Mode-I edge crack-tip. The accuracy of each individual model is then determined by a comparison with the singularity solution and two-term solution of Rice[24] and Leevers and Radon[66] in linear elastic fracture mechanics (LEFM). Results show that there exists a critical model size, L_{CR}, below which unrealistic crack-tip behavior may manifest and above which may not be necessary under a given error tolerance. This result may offer a guideline for the model size design and signifies the necessity for developing new multiscale methods which can produce large-size models, prescribed by the L_{CR}, to improve the simulation accuracy.

A. Introduction

Recently, much ongoing research uses molecular dynamics (MD) and atomistically-based multiscale simulations for crack analysis following the fact that crack nucleation and propagation originates from the atomistic and microscopic scales. The work of Gumbsch[67] may represent an early effort in developing a fracture theory based

on atomistic analysis. He and his colleagues first developed the FEAt (Finite Element-Atomistic) coupling scheme[61] between atoms and continuum. FEAt is the earliest version of the direct coupling (DC) multiscale method which links atoms with finite element nodes at the scale interface and has been proved to have smooth variable transition.[36] Using this method Gumbsch[67] carried on an atomistic study of brittle fracture to find explicit failure criteria from atomistic modeling. His report shows brittle cracks under general mixed mode loading follow an energy criterion (*G*-criterion) rather than an opening-stress criterion (K_I-criterion). His report on crack-tip blunting and failure modes of brittle fracture described below takes notice: "Depending on the shape of the blunted crack tip, the observed failure modes differ significantly and can drastically disagree with what one would anticipate from a continuum mechanical analysis". It also motivates a study on his model size design. To ensure that the results are not affected by the size of the model, scaling tests were carried out in that work. Specifically, the author used the lattice-trapping range ΔK ($=K_{Ic}^{+}-K_{Ic}^{-}$) as a criterion to choose the model size, where K_{Ic} is the critical stress intensity factor of mode I, superscript "+"and "−" denote a special loading and unloading status for a stationary mode I crack. Finally, the total system size of $70a_0 \times 70a_0$ and the inner atomic domain size of $12a_0 \times 12a_0$ were taken, where a_0 is the atomic lattice constant. This choice was determined based on the following results of the scaling test: First, varying the total model size from about $20a_0 \times 20a_0$ to $160a_0 \times 160a_0$ showed a decrease in the lattice-trapping range from $\Delta K = 0.04K_G$ at the lowest system size to $\Delta K = 0.02K_G$, here subscript *G* denotes Griffith. Second, if system sizes change from about $50a_0 \times 50a_0$ up, no further change of ΔK could be detected. Third, increasing the size of the atomistic region at a total size of $70a_0 \times 70a_0$ brought K_{Ic}^{+} from 1.03 K_G down to 1.02 K_G, but left the lattice trapping range essentially unchanged. While using ΔK is convenient, it may not be sufficient for investigating model size effects on crack-tip behavior. It is seen from the report ΔK is not sensitive to the total model size, thus the model size effects cannot be fully measured by the ΔK criterion. For instance, the larger models showed a more dramatic failure behavior than the small models but this reported phenomenon cannot be explained by the ΔK values.

Recently, one sees widely influenced efforts in the derivation of the traction-separation (T-S) laws for fracture analysis using cohesive surface elements (CSE) with atomistically-based modeling. Among these, Yamakov et al.[68] developed a cohesive surface model using MD simulations of intergranular fracture in an FCC metal, which displays both brittle and ductile fracture mechanisms. These authors combine estimates of crack opening displacement with normal stress to construct a qualitative model of T-S laws for both mechanisms. This approach was later combined by Yamakov et al.[69] with ESCM, short for the embedded statistical coupling method by Saether et al.,[32] to couple an embedded atomistic domain within a continuum domain. The resulting coupled system was used to derive a cohesive zone model (CZM) for interface debonding via moving averages of stress and opening displacement. Spearot et al.[70] proposed an internal state variable (ISV) framework that uses interface separation constitutive laws motivated by MD simulations of materials with EAM potentials by Foiles et al.[71] These authors considered both normal and tangential displacement loading, and developed a nonlinear elastic separation potential that included path-history dependent effects with active and passive ISVs.

Others have done work guided by the same concept and technical route in developing key features of the T-S laws with CSE analysis. Choi and Kim[72] constructed a cohesive model for single crystal gold. Krull and Yuan[73] designed an exponential T-S law using parameters derived from MD simulations of crack tip blunting and void initiation. Dandekar and Shin[74] developed an MD based cohesive zone law for describing Al–SiC interface mechanics of a composite system. In an attempt to increase the fidelity of an atomistically derived cohesive law, Zhou et al.[42,75] used the methodology of Yamakov et al[68] to develop a model for an 'ideal' BCC metal subject to mixed-mode loading conditions. To keep the crack propagation along the designed propagation path, they used a special potential to make the designed crack propagation path a "weak" one with lower values of cohesive energy, elastic moduli, and work of adhesion.

While these efforts are significant, some fundamental questions may be asked, for instance, whether the model size is sufficiently large to avoid any artificial effects from boundary conditions (BC) on the analysis. The latter effects can exacerbate ghost forces which make the distributions of strain and stress fields near the crack-tip distorted and

cause the estimate of crack opening displacement with normal stress on the CSE to have lower accuracy. While it is seldom that concrete answers to these apparent questions are found in the public domain, a known assessment is given by Fan and Yuen.[40] They conducted a cohesive FEM simulation for an epoxy/Cu interfacial fracture with a tensile double cantilever beam (TDCB) specimen. Directly using the T-S law obtained by them with atomistically-based multiscale analysis, this FEM crack simulation showed poor agreement with the experiment by Fan et al.[76] The predicted simulation force for a given displacement of TDCB at the interface between epoxy/Cu was two times higher than the experimental data. To see whether this low accuracy has any relationship with the model size an uncompleted survey for the model size used in different atomistically-based simulations are given in Fig. 23. It is seen that most existing atomistically-based models have sizes L_X and L_Y around 100 nm×100 nm, and the smallest one of 3.53 nm×3.53 nm has poor accuracy by Fan and Yuen.[40]

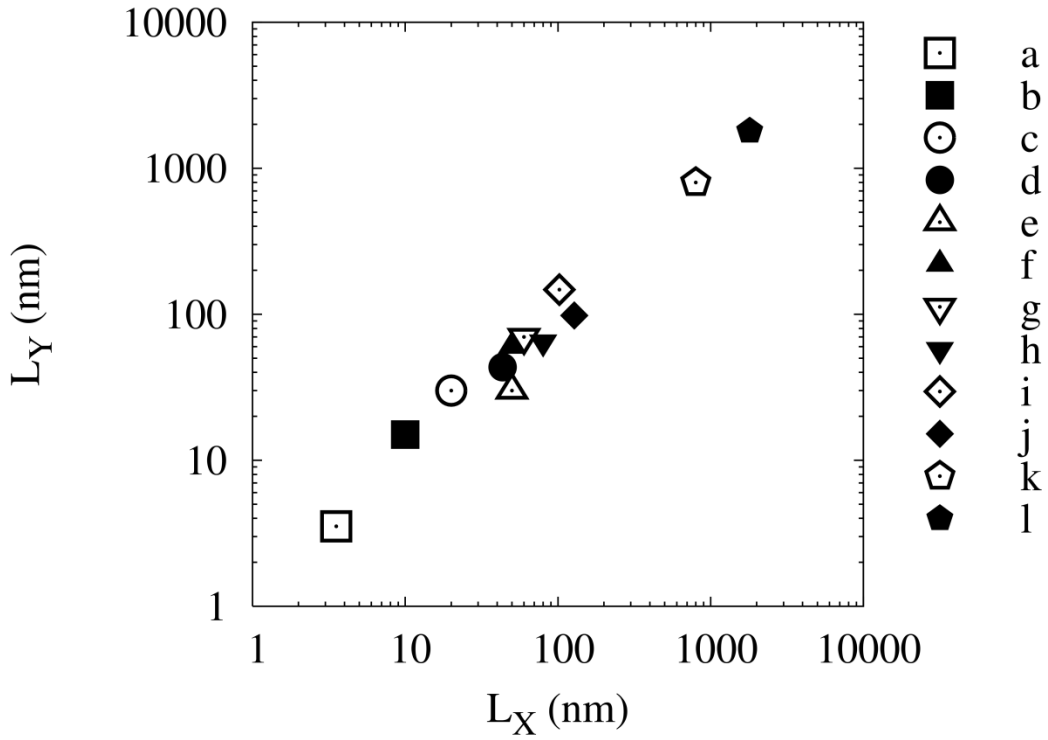


Figure 23. A survey for the model size in different modeling work.

a. Fan and Yuen, 2010[40] b, c, f and g. Dandekar and Shin, 2011[74]

d. Spearot et al., 2004[70] e. Krull and Yuan, 2011[73] f. Dandekar & Shin, 2011[74,77]

h. Zhou et al.2008[75] & 2009[42]; Lloyd et al. 2011[43] i. Tsai et al., 2010[41]

j. Yamakov et al., 2006[68] k. Yamakov et al., 2014[78] l. Yamakov et al., 2008[69]

Now is a good place to mention other progress in using atomistically-based modeling. Lloyd et al.[43] implemented one of the cohesive models developed by Zhou et al.[42] into UMAT of the software ABAQUS.[79] Then the FEA's performance was verified by examining the geometry and loading rate which was originally used to derive the T-S law by MD. Their analyses show close agreement between the FEM and the MD results for both stress and crack opening displacement profiles at the cohesive interface. This is significant since one can use FEA to reproduce the result by atomistic analysis to save a lot of computational time. However, this agreement indicates only that the MD-obtained T-S law is consistent with the cohesive FE analysis with the UMAT based on that law. It is difficult to be certain that the T-S law and the data, such as the K_{Ic}=1.154 MPa√m for BCC obtained by their work accurately reflects the material property.

From the above descriptions, it is seen that the accuracy of atomistically-based multiscale simulations may intrinsically relate to the model size and that there is a lack of accuracy verification methods to confirm it. Thus, the accuracy validation in terms of model size becomes essential and will be taken as the main objective of the paper. To reach the goal three aspects need to be addressed.

Firstly, one must find new accuracy validation methods for understanding the crack-tip behavior

In general, accuracy validation of physical concepts and methods by experiment is a traditional approach. However, the comparison is difficult for the atomistically-based simulation due to the resolution limitation of the test equipment in the angstrom and nanometer scales. So far, the most popular approach in concurrent multiscale analysis is to compare the results with the fully atomistic simulation such as MD. Among those efforts, two notable verifications should be noted. Curtin and Miller[36] used a one-dimensional (1D) spring model to compare the bridging behavior at scale transition regions of various models. The other was the benchmark computation test of 14 models carried out by Miller and Tadmor.[53] The fully atomistic simulation is the benchmark

against which one compares the multiscale models. Here, the corresponding atomistic solution is considered as the exact solution. Other examples include analysis of dislocations at notches[48] and dislocations passing through scale boundaries.[50] While this kind of verification is effective in many aspects related to atomistic analysis and its bridging to high scales, one of the shortcomings is the requirement for huge computational spatial capacity and time if the model size is reasonably large. In fact, accuracy verification in low scales is important to find the deformation mechanisms such as crack nucleation in fatigue, dislocation patterns in fatigue and creep, the inherent inhomogeneity of plastic deformation, etc. However, these mechanisms must link to material behavior and be characterized in the continuum scale for applications. This requires a relatively large continuum model size. Otherwise the approach of predicting material behavior by implicitly averaging atomistic/microscopic dynamics may not be valid.

Due to this limitation of computation power, it can be hard to make a judgment for how accurately the multiscale simulation describes the continuum behavior for engineering applications. One way to mitigate the problem is to augment the current validation methods with new approaches. Recently, the authors use a classical two-dimensional (2D) elastic solution of a specimen with a central cylindrical hole under tensile load to make a comparison with displacement data obtained by atomistically-based multiscale analysis to validate model accuracy. This work, to be published elsewhere, saves a lot of degree of freedom (DOF) and yields satisfactory results to check the applicability for using the multiscale analysis for continuum analysis. In this paper, we will continue this technical route to add the richness of continuum mechanics to the validation. Specifically, we will take the analytical solutions of linear elastic fracture mechanics (LEFM) for the displacement at the crack tip to validate the obtained displacement data by the atomistically-based multiscale simulation.

Secondly, one must find ways based on physics and mechanics to find the model size effects.

In Fig. 23, it is seen that the model size varies within a large range from 3.53nm×3.53 nm[40] to 1800 nm×1800 nm[69]. Among these, the work given by Dandekar

and Shin is notable.[74,77] They used four model sizes, namely 10×15×4, 20×30×8, 50×30×8 and 60×70×16 nm^3 and obtained the corresponding Young's modulus *E*. According to the above order, their *E* values are, respectively, 224.42, 188.66, 180.63 and 177.93 GPa. It merits note that the largest value of Young's modulus corresponds to the smallest model and that value is beyond the Hashin–Shtrikman bounds of 155~213 GPa. In addition, they found with a larger system the simulated Young's modulus tends toward the experimental observation and between the system sizes of 60×70×16 nm^3 and 50×60×16 nm^3 the difference in the Young's modulus is only ~1.5%. Therefore a system size of 50×60×16 nm^3 offers a good balance in terms of computation time and the required accuracy is selected to conduct the fracture studies. This study is good for investigating the Al-S_iC interface mechanics. However, most other works do not explain how their model sizes are determined. One reason may be due to a lack of knowledge and method based on physics and mechanics to guide the investigation of model size effects. There may be another practical reason which may relate to the third need described below.

Thirdly, one must find a way to enlarge the model dimension for investigating model size effects

The other reason for the large model size requirement comes from the boundary effects on the atomistic stress and strain fields which are in the area of interest. Actually, if the boundary is too close it will change the distribution of the variables making the simulation result less accurate. In some cases, the conditions for deriving the analytical solution cannot be fully satisfied. In this case, a minimum model size is searched for within a given error tolerance. For all these purposes, one needs to find a multiscale method which can develop large model sizes to carry out this analysis for size effects.

In this work, a solution for this challenge is discussed within the framework of GP and GP-FEA methods where GP is short for the Generalized Particle Dynamics Method proposed by Fan.[48] The GP-FEA, proposed by the authors, is a new method which can make the model size as large as needed. In addition, it links the continuum via FEA nodes with high-order particles but not directly with atoms, as is the case with methods such as the QC, short for quasicontinuum by Tadmor[55] and FEAt methods, to avoid artificial effects such as ghost forces.

Accuracy verification with the well-known continuum field solutions has great potential for extending applications of atomistically-based multiscale analysis because the great treasure of continuum mechanics and continuum physics opens new avenues to verify its accuracy. In this section, accuracy verification for crack-tip analysis of a stationary crack in an iron specimen is conducted by a comparison of the GP-FEA simulation result with a classical solution of LEFM. This analytical solution from elasticity theory near the crack-tip is accurate to some extent, and was contributed to by Westergaard[80], Irwin[81], Williams[82], Rice[24], Leevers and Radon[66].

Note that if the comparison is satisfactory the significance is far-reaching and the validity of the multiscale analysis may be extended to other problems where no exact analytical solutions exist. Alternatively, if the comparison fails then its validity for the prediction of more complicated cases is questionable. In other words, if the atomistically-based simulation cannot predict the elastic deformation fields by LEFM, it seems unlikely that the same method can accurately predict the T-S law which involves non-linear phenomena such as crack propagation, crack-tip blunting and elastic-plastic deformation. To deeply understand this judgment, one must realize that the accuracy control factors of atomistically-based multiscale modeling are much different from continuum mechanics. For the former, the accuracy mainly depends on the multiscale methodology and the potential used. If the methods, especially the scale-bridging method, is correct and the potential used, such as the EAM potential, is accurate then its solution should be accurate for wider cases where both methods and potential have not changed and remain applicable.

This feature is quite different from continuum mechanics where if material constitutive laws, strain-displacement relations and governing equations change then the accuracy will change and should be verified for every single case. This fundamental difference highlights the advantage of extending the verified atomistically-based multiscale analysis to wider applications. Based on this view point, the focus of this paper is on the comparison of deformation fields near the elastic crack-tip of a stationary brittle iron specimen by the proposed GP-FEA method with the LEFM analytical solution to determine under what conditions the accuracy of the method can be confirmed. In the

next paper, this accuracy-confirmed method will be applied to crack-tip behavior during crack propagation until failure.

This Chapter is organized as follows. Section B gives a foundation to introduce the newly proposed GP-FEA method and model design. Section C will present numerical comparisons between simulation results with the LEFM solution at the crack tip under the plane strain conditions. It will show the LEFM solutions for the displacement field near the crack tip, including the singularity solution and the two-term solutions derived by Rice[24] and Leevers and Radon[66]. The Chapter ends with discussions and conclusions in Section D.

B. GP-FEA Model Design

As shown in Fig. 24, the GP-FEA simulation model includes two inter-connected parts: one is the GP model consisting of multiscale particle domains and the other is a continuum domain made of an FEA mesh. The first of three operations for model design is to develop the GP model to be embedded in the FEA mesh. It is of primary importance, as it includes the atomistic domain where the crack tip will be closely observed. Steps two and three are related to interface design between the GP and FE region as well as the FE mesh design.

In the example shown in Fig. 24b, the GP model consists of three particle scales representing an iron plate with an edge crack. Scale-1 (S1) with scale n=1 is the atomistic domain and is 6 nm wide and 8 nm high around the crack tip. Scale-2 (S2, n=2), the second particle domain, surrounds the atomistic domain whose outer boundary size is about 60 nm in width and 20 nm in height. Scale-3 (S3, n=3), the third particle domain, surrounds the S2 domain space above and below whose size is 60 nm in width and 120 nm in height. The edge crack extends from the left surface in scale-2 into the atomistic domain (S1) having a total crack length of 8 nm. The thickness of each scale and thus the whole GP model is about 4.5 nm.

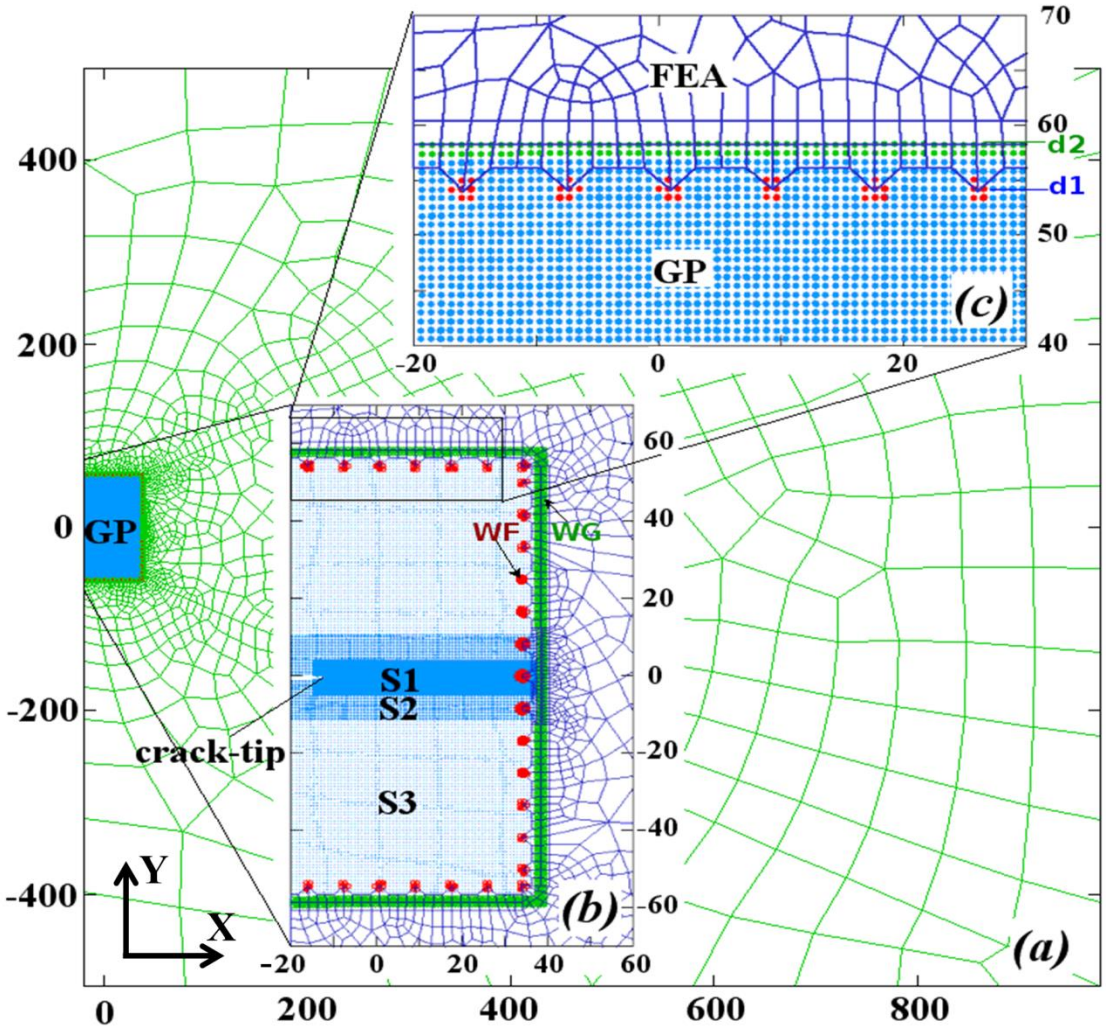


Figure 24. Schematic of a micrometer-size GP-FEA model with an embedded 60×120×4.5 nm^3 three-scale GP model for a thin plate with an edge crack, where the unit is nm. The loading direction [-110] is along the Y-axis and the crack direction [110] is along the X-axis, transverse to the loading direction. The blue/points in the figure represent GP particles and the quadrilaterals around the edges represent finite elements. The separated red dots denote WF domain, the narrow green strip surrounding the GP model is the WG domain.

Following Eqn. (1) and (2) with the scale ratio k equal to 2, the generalized lattice constant for S2 and S3 are, respectively, $a_2=2a_0$, $a_3=4a_0$ and the corresponding number of atoms that a particle represents for S2 and S3 are $\ell_2=8$, $\ell_3=64$. The total number of real particles and atoms is 109,536. If the model was completely filled by atoms, the number

would be 2,865,852, i.e., 26.16 times larger than the GP model's. The bridging between particle scales with bottom-up and top-down schemes is accomplished by imaginary domains $W_{(n+1)\text{image}}$ and $W_{(n)\text{image}}$, as introduced in Section II.A.2. After this GP model is developed the FEA mesh will be generated to fit around the GP model, interfacing at the top, bottom and right side as seen in Figure 25b.

C. Comparison between LEFM solutions and GP-FEA simulation results for the crack-tip displacement

Our experience in using a 2D elastic solution of a cylindrical hole specimen to check the accuracy of the GP-FEA method told us that one may face new challenges along this technical route. Firstly the explicit expression needs to be carefully checked so the direct comparison can be available. Secondly, the data processing for the atomistic scale needs to be done correctly which should involve a group of atoms near a continuum point not a single atom in that position. The third challenge concerns how to estimate the error by the comparison. These issues have been addressed in detail in that work to be published elsewhere and will be discussed again for this particular crack-tip problem.

1. LEFM singularity solution and two-term solutions of crack-tip displacement field

The variables of the comparison are the displacement components not stress tensors. This is because in continuum theory stress is defined locally as the limit of average stress when the area, which subject to the force, approaches zero. This is consistent with the local behavior of the stress-strain relationship that the stress at a material point depends only on the strain at the same point of the continuum. On the other hand, the stress in the atomistic field is the so-called virial stress defined as

$$\underline{\sigma^{\alpha}} = \frac{1}{\Omega^{\alpha}}\left[-m^{\alpha}\bar{v}^{\alpha}\bar{v}^{\alpha} + \frac{1}{2}\sum_{\beta\neq\alpha}\left(\bar{\mathrm{f}}^{\alpha\beta}\right)\left(\bar{\mathrm{r}}^{\alpha\beta}\right)\right] \tag{22}$$

Here, the symbol "-" under a letter denotes a second order tensor and above a letter denotes a vector. There are three vectors: velocity vector, $\bar{v}$, the force vector, $\bar{\mathbf{f}}$, and the position vector, $\bar{\mathbf{r}}$. The superscript α and β denote the atoms and Ω^{α} denotes the volume of atom α. In fact, the stress at atom α depends not only on those atoms closest to it, but also on the atoms, β, within its spherical neighborhood within a cutoff radius through

interatomic forces. Eqn. (22) defines the per-atom stress. To get the corresponding continuum stress a further average for the volume occupied by these atoms should be conducted. The different definition between the stress defined by a continuum and the atomistic field makes direct comparison difficult. On the other hand, the definition of displacement is clear and unified for both continuum and the atomistic field if a certain volume average can be conducted for the atomistic field in order to exclude thermal displacements. Our experience shows the volume average for the virial stress is more sensitive to the volume specified than the volume average for the atomic displacement.

Unlike the 2D plate with a central cylindrical hole, there are explicit LEFM expressions for the displacement field near the crack-tip. For the mode-I opening crack, the leading terms are as follows[83]

$$\sigma_{ij} = \frac{K_I}{\sqrt{2\pi r}} f_{ij}(\theta) \tag{23}$$

$$u_x = \frac{K_I}{2\mu}\sqrt{\frac{r}{2\pi}}\cos\left(\frac{\theta}{2}\right)\left[\kappa - 1 + 2\sin^2\left(\frac{\theta}{2}\right)\right] \tag{24}$$

$$u_y = \frac{K_I}{2\mu}\sqrt{\frac{r}{2\pi}}\sin\left(\frac{\theta}{2}\right)\left[\kappa + 1 - 2\cos^2\left(\frac{\theta}{2}\right)\right] \tag{25}$$

where $\kappa = 3 - 4v$ for plane strain, and $\kappa = (3 - v)/(1 + v)$ for plane stress, μ- shear modulus, K_I-stress intensity factor of Mode I, r and θ are polar coordinates of the point with the displacement.

These equations actually correspond to a singularity solution since the corresponding expressions of stress components are all inversely proportional to r, which will become infinitely large when r approaches zero. This solution is taken from the mathematic solution given by Westergaard[80] and others in which the dominant leading term of the stress at the crack-tip has a singularity and the leading term of the displacement field is shown in Eqn. (24) and (25).

When the r-distance from the crack tip increases these terms decrease fast and the remaining part of the solution cannot be neglected. Rice[24] first suggested to add a second term related to the so-called non-singular stress term "T" that agrees with the results by Larsson and Carlsson.[84] Later Leevers and Radon[66] derived crack-tip deformation with a biaxiality parameter B which relates to the magnitude of the T-stress and can be written

as follows:

$$\sigma_{ij} = Kr^{-\frac{1}{2}} f_{ij}(\theta) + \text{non-singular terms} \tag{26}$$

$$u_x = \frac{K_I}{2\mu}\sqrt{\frac{r}{2\pi}}\cos\left(\frac{\theta}{2}\right)\left[\kappa - 1 + 2\sin^2\left(\frac{\theta}{2}\right)\right] + (1-v^2)\frac{B}{E\sqrt{\pi a}}K_I\cos\theta \tag{27}$$

$$u_y = \frac{K_I}{2\mu}\sqrt{\frac{r}{2\pi}}\sin\left(\frac{\theta}{2}\right)\left[\kappa + 1 - 2\cos^2\left(\frac{\theta}{2}\right)\right] - v(1+v^2)\frac{B}{E\sqrt{\pi a}}K_I\sin\theta \tag{28}$$

B values for the single-edge-cracked specimens are expressed as the following by Kardomateas et al.[85]

$$B = -0.52 - 1.50\left(\frac{a}{w}\right) + 12.70\left(\frac{a}{w}\right)^2 - 20.70\left(\frac{a}{w}\right)^3 \tag{29}$$

K_I solution for single edge notched tension (SENT) is[83]

$$K_I = \frac{P}{b\sqrt{w}}\frac{\sqrt{2\tan\frac{\pi a}{2w}}}{\cos\frac{\pi a}{2w}}\left[0.752 + 2.02\left(\frac{a}{w}\right) + 0.37\left(1 - \sin\frac{\pi a}{2w}\right)^3\right] \tag{30}$$

For the GP-FEA model in this paper, remote strain is applied on the top and bottom edges at 1% engineering strain, which is defined by the elongation between the top and bottom boundary of the model divided by the initial model size L_Y along the Y-axis. This corresponds to a remote average stress 2.24GPa and total boundary tensile force P=10.2 μN when E = 224GPa. In reality, after calculating the remote stress distribution, it is found that the error of average stress ranges from 0.18% to 2.8% by the GP-FEA models (see Figure 32 later).

Before coupling to the FEA mesh, the GP model equilibrated in the NPT ensemble for 100 ps at 100 K and 1 atm to ensure configurational stability. The GP model then had its temperature increased to 300 K and allowed to equilibrate for 20 ps before being attached to the FEA mesh. The loading procedure kept the thickness constant and no deformation was permitted along the thickness direction to mimic plane-strain conditions. Load was applied in two 0.5% strain increments. For each increment the strain was applied at the FEA mesh outer boundary then 20 ps of relaxation at 300 K was allowed for the WG-GP subsystem or until the GP-FEA interface converged triggering the next load step.

2. Comparison between atomistically-based multiscale simulation with both LEFM singularity and two-term solutions

Figure 25 schematically shows three paths in front of the crack-tip on which the eight models' displacement components will be compared with the LEFM solutions. They are paths along the X- and Y- axis and a semi-circle in the first quadrant with radius R=15 Å. To make the GP-FEA calculation result of any generic point, s, in these paths comparable with a continuum solution, the average displacement of atoms within a finite volume surrounding the point at hand should be used. For the circular path, the volume average is over a cylinder with radius of 3.615Å. For the X-axis path a rectangular domain is selected with 2Å width and 8Å height but for the Y-axis path, 8Å width and 2Å height are used instead to maximize the accuracy for this displacement field with a high gradient. All of the domains have the same thickness as the model. As a result, the cylinder contains over 170 atoms and each rectangular have about 60 atoms. The first rectangular domain on X or Y axis is 1Å away from the crack-tip. The obtained average displacements of the atoms will then be compared with the corresponding data calculated from the analytical solution for the plane strain condition.

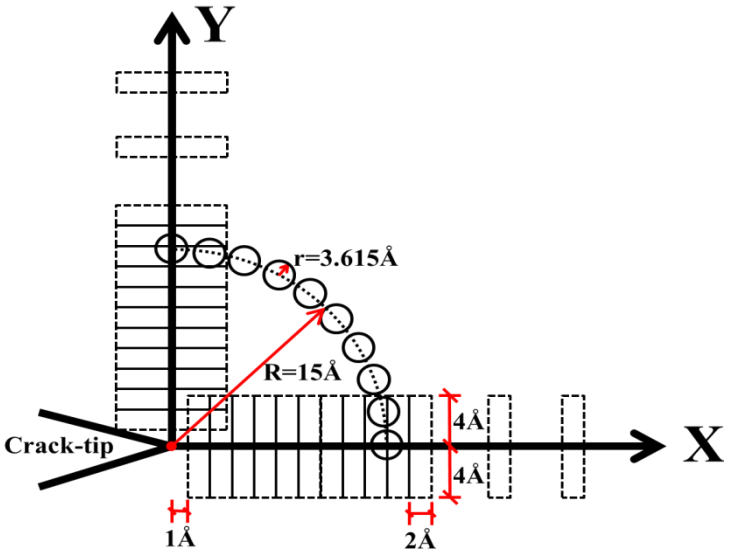


Figure 25. Distribution of data acquisition domains along the paths of the X-axis, Y-axis and a semi-circle with radius of R=15Å

Figures 26 to 28 shows the comparison between simulation results with the LEFM singularity solution and two-term solution, respectively, for the Y-displacement

component, u_y along the Y-axis, the X-displacement component, $u_{x,}$ along the X-axis and the radial displacement component, $u_{r,}$ along the circle of R=15 Å in the first quadrant. The model size in the GP-FEA simulation has 1000nm width and 500nm height. From the comparison between the simulation result with, respectively, the singularity solution and the two term solution in the three figures, it is seen clearly that the LEFM second term solution is more accurate and consistent with the simulation result than the singularity solution beyond the range of r≥2 Å. This is rational in LEFM due to the work by Rice and others. Thus, in the next section we will mainly show the comparison of simulation results with the two-term LEFM solution.

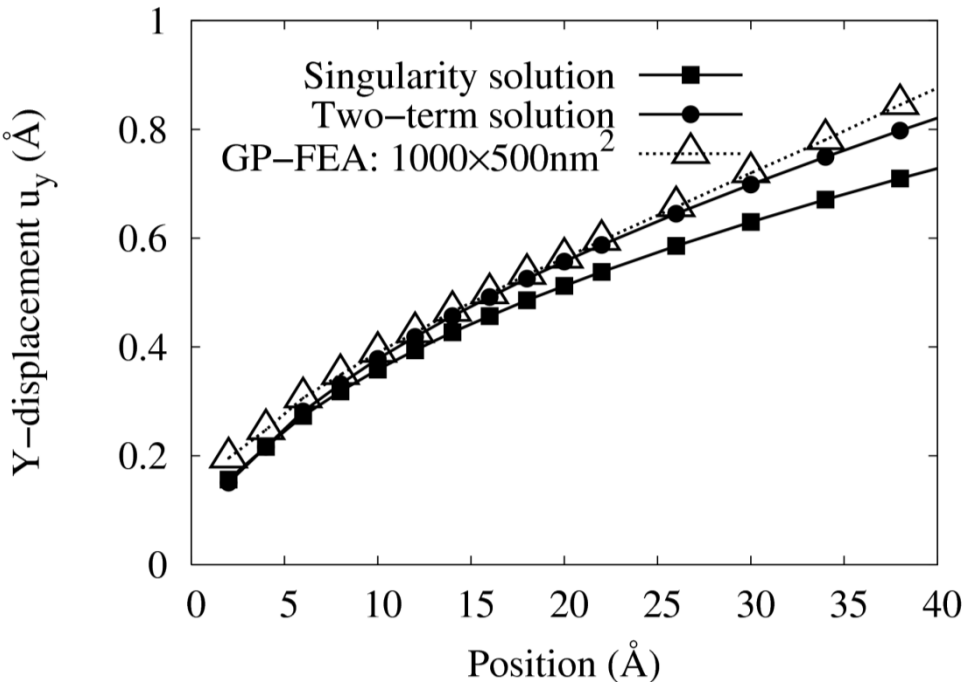


Figure 26. Y-Displacement comparison between results of GP-FEA simulation with LEFM solutions along the Y-axis.

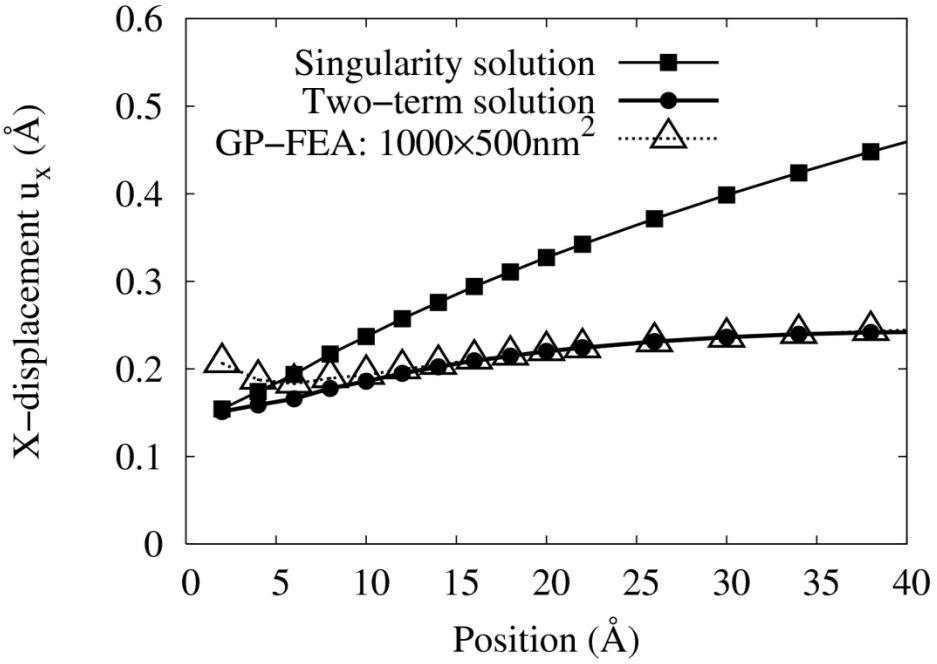


Figure 27. X-Displacement comparison between results of GP-FEA simulation with LEFM solutions along the X-axis.

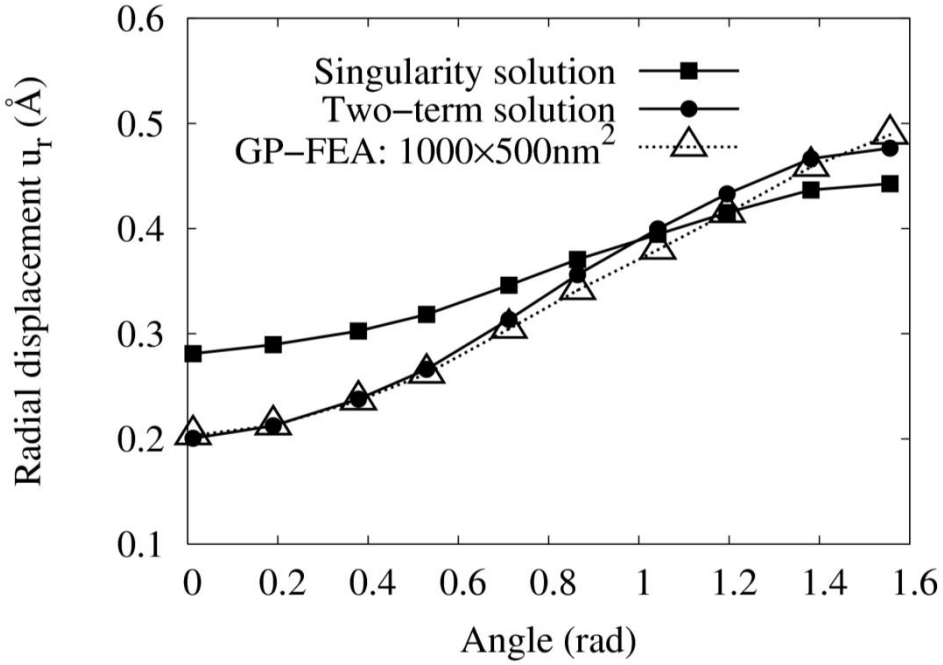


Figure 28. Radial displacement comparison between results of GP-FEA simulations with LEFM solutions along the circle of R=15 Å in the first quadrant.

3. Model size effects on numerical results of displacement distribution near crack-tip in comparison with LEFM two-term solutions

For the case of loading along the Y-direction, the model size L_y along the Y-axis is dominantly important and will be taken as the characteristic model dimension. Thus, the model size effects will be mainly measured by the Y-displacement component, u_y, versus L_y. In view of a minor but not negligible effect of the model size L_x along the X- (or crack line) direction the width of all models will be kept the same as L_x=1000 nm. All the comparisons are carried out by the same condition with strain level ε_y=1%. For the single crystal model it is within the elastic range.

Figure 29 shows the effects of model size L_y on the Y- displacement component, u_y, with eight model sizes for L_y ranged from 120 nm to 5 μm. In the figure, the two-term solution denoted by a bold solid line, will be taken as the exact solution because of its semi-infinite size it is physically sound for an edge crack From this figure one can see that the smaller the model size the less of the displacement u_y at the same location y from the crack tip. This small size causes an error around 50% in comparison with the two-term LEFM solution. It is interesting to note that when the model size L_y increases to about 500 nm and beyond, the simulation results agree well with the two-term solutions. Figure 30 shows the model size effects on the radial displacement component, $u_{r,}$ along the semi-circle of R=15 Å in the first quadrant. Surprisingly, if the model size is too small (i.e., $L_y \leq$250 nm) the trend of the curve $u_r \sim \theta$ is not consistent with the analytical solution.

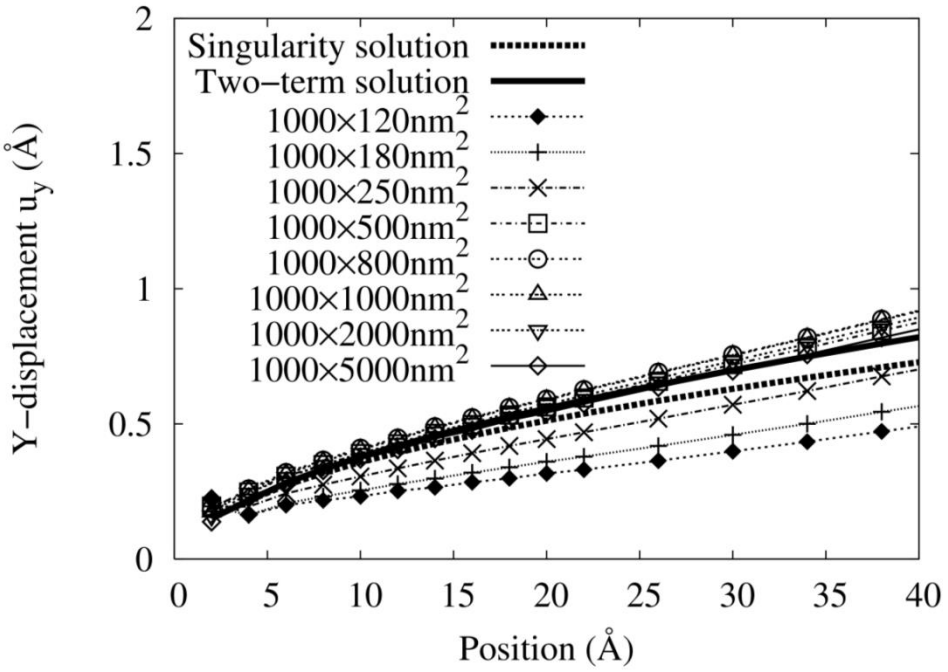


Figure 29. Effects of model size, L_y, on the displacement component u_y for the location along the Y-axis near the crack-tip within 2 to 40 Å.

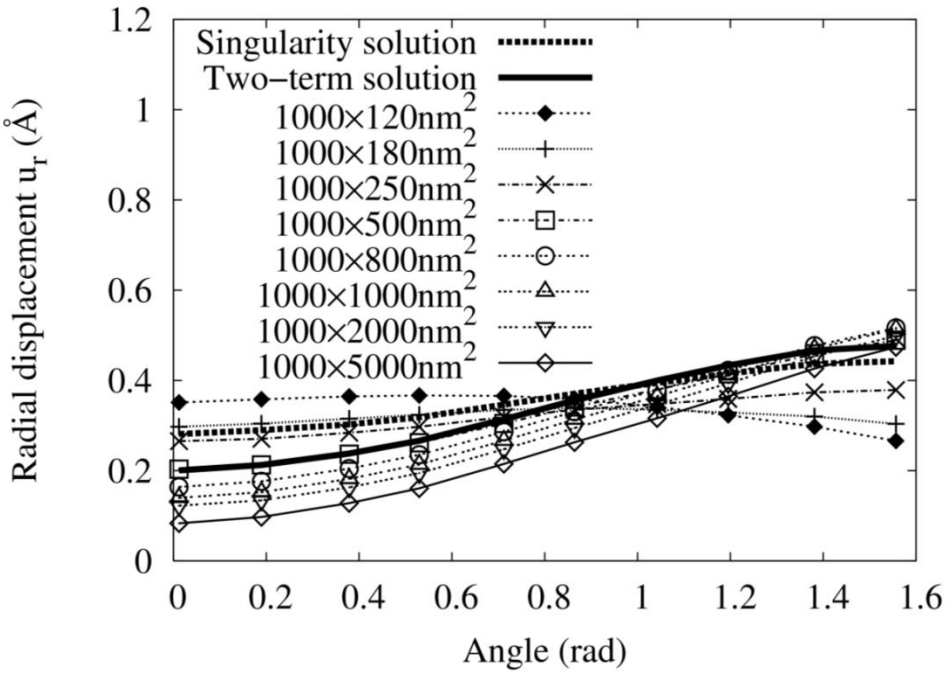


Figure 30. Model size effects on the radial displacement component, u_r, along the circle of R=15 Å in the first quadrant.

To find the reason of model size effects, Fig. 31 shows the σ_y stress distribution of the FE elements along the top boundary (i.e., y= $L_y/2$). It is seen that when the model size L_y < 800 nm, the top boundary stress is seriously affected by the crack-tip with a much lower value than the average one. On the other hand, if $L_y \geq 800$ nm then the average remote stress is about 2.24 GPa. The result indicates that if the BC is too close to the field of interest then it may cause coupling effects between the boundary stress and the crack-tip displacement as we see by the great reduction in the displacement near the crack-tip for small models shown in Fig. 29. For the case of a force boundary, the model size effects will also appear. In that case, if the model size is small then the stress field near the crack-tip will be seriously affected.

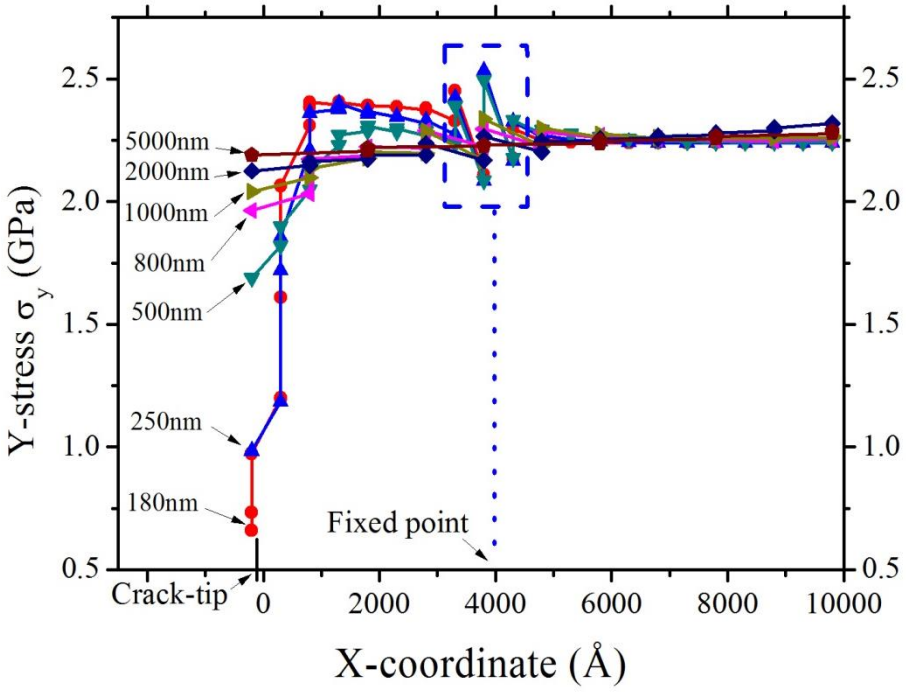


Figure 31. Model size effect on boundary stress component, σy, distribution along the top BC for different models. For these 1000nm width GP-FEA models, edge crack with length of 8 nm is embedded on the left side located at X-coordinate -200. Two fixed points on the top and bottom kept at 400nm away from the left side of the model were applied on each GP-FEA model to avoid rigid body motion, thus, the disturbance can be detected on the Y-stress curve in the dashed box.

D. Summary and discussions

Looking back on the developments of multiscale analysis in the past two decades, two basic issues for extending engineering applications of concurrent multiscale simulations can be addressed. The first is how to quantify the accuracy of atomistically-based multiscale simulation and the second is how to enlarge the model dimension to the minimum size necessary to make the modeling realistic and accurate. While accuracy verification with the well-known continuum field solutions is important for engineering applications, it is seldom performed due to unknown model size effects and the limitation of many multiscale methods in developing a minimum model size necessary to make the comparison meaningful. To address these issues, the key is to develop new multiscale methods which can produce sufficiently large models. Encouraged by the successful comparison of the displacement field predicted by the GP-FEA method with the continuum solution for a 2D plate with a central hole, this newly proposed multiscale method has been further developed and applied to crack-tip analysis. Results show excellent agreement between the simulation and LEFM two-term solutions.

This successful comparison with the continuum solution and the powerful capability of the GP-FEA multiscale method in developing a large micrometer size model is promising. It allows for the investigation of model dimension effects on the accuracy of atomistically-based multiscale methods realistic and attractive. The significance of this investigation should be further emphasized even though the choice of model size is a common problem that appears frequently during model design. It can be further addressed from the following four aspects. Firstly, accuracy verification for low scales is important to find the deformation mechanisms. To make this finding meaningful, however, one must link these low-scale dynamics to material behavior and be characterized in the continuum scale for applications; which requires a relatively large continuum model. Otherwise the approach of predicting material behavior by implicitly averaging atomistic/microscopic dynamics may not be valid. Secondly, if the model size is small the BC disturbances may likely affect the local force and displacement fields which are near the atomistic regions of interest such as interfaces, crack-tips and flaws. In most practical cases, BC cannot be perfectly maintained and will have oscillatory and random perturbations. Various BCs can be accepted if they are sufficiently far away from

the regions of interest, following the concept of Saint Venant's principle.[63] Otherwise, the obtained low-scale phenomena observed can be qualitatively different which can cause instability of atomic motions, microstructural evolutions, and unexpected material failure. Thirdly, some mathematical solutions for the continuum require the medium to be sufficiently large to make the LEFM crack-tip solution realistic for a tiny crack inside of a bulk material. In this case, model size must not be small for a reasonable result. The fourth aspect is that for microsystems and nanotechnology, the model size should be equal or larger than micrometers so the problem of micro- or nano- sensors/activators can be more accurately simulated. For nanotechnology, this is true for some designs since nanotubes, nanofibers, etc. need to be assembled and embedded in a matrix which has a certain size requirement. Thus, investigating the model size effects and choosing a minimum model size necessary for the accuracy requirement is essential.

With the proposed GP-FEA method, the model size effects on the crack-tip displacement fields of a Mode-I edge crack embedded in a single crystal of BCC iron along the X-direction [110] are extensively investigated. All models were subjected to the remote BC displacement along the Y-direction [$\bar{1}$10] with 1% strain. In this problem, the main displacement component is u_y along the loading direction Y and displacement component $u_{x,}$ related to Poisson's effect, has minor effect, thus all the models' widths, $L_{X,}$ were held at a constant 1000 nm but the L_y value was changed from 120, 180, 250, 500, 800, 1000, 2000 to 5000 nm to observe the model size effect. This size effect is measured by the displacement u_y for same material points in different models. They are located on the Y-axis and apart from the crack tip from 2 to about 40 Å. The accuracy is verified by the LEFM two-term solution described in the same figure. From the comparison of the eight simulation curves of these models with the analytical solution shown in Figure 30, one observes the following observations and conclusions.

- It is seen that the smaller the model size the larger the error produced in the simulation-obtained u_y. Specifically, for the case of L_y=120 nm, the error can reach about 50%. This result is consistent with the result of Fan and Yuen;[40] for their small size model a double force is needed to produce the required interfacial displacement. It is also consistent with the result of Dandekar and Shin,[74,77] for their smallest model of 10×15×4 nm^3 the Young's modulus is about 1.25 times

larger than the testing value for the large model, indicating the small model has high rigidity to produce small deformation. Our work shows that using stress intensity factor K to investigate the model size effects is not sufficient since that value is obtained by a model with infinite size. Changing the model size and comparing the behavior with the LEFM solution will show the size effect quantitatively. This result serves as a serious warning: since many existing simulation models are below this size as given in Figure 23, the accuracy of these models may be questionable and need to be carefully verified.

- When the model size increases from 120 nm to 500 nm, the accuracy quickly increases. However, a further increase of the model size from 500 nm to 5000 nm results in basically the same accuracy as the case of 500 nm. This result is significant since it lays a foundation for introduction of a new concept of critical model size, L_{CR}. In fact, the comparison tell us that if the model size is less than L_{CR}, say 500 nm, the results obtained from atomistically-based multiscale simulations will have unrealistic crack-tip behavior, including a large percent of inaccuracy in comparison with the LEFM result. On the other hand, the case for designing the model size larger than L_{CR} should also be avoided since it may not greatly improve the accuracy with the penalty of increasing a large of DOF.
- While this finding may open a new avenue to develop a guideline for the least-required model size, L_{CR}, to improve the accuracy in bridging atomistic and continuum scales, more investigation on the feature of L_{CR} should be conducted. Since L_{CR} is a problem dependent variable, it may relate to material property, environmental conditions, the variables involved, the answer evoked, etc. Thus, it is hard to get a general answer analytically. In many cases, one should carry on numerical simulations for models with different size to find the minimum necessary for the required accuracy. Fortunately, the newly proposed GP-FEA, which can develop large model sizes, has proven so far to be an effective tool to face this challenge. To introduce GP-FEA methods in detail with more examples and their wide applications in crack propagation involving plasticity and failure are prepared to be published elsewhere.

APPLICATION IN CRACK PROPAGATION

The model size was found to affect the deformation field around a pre-crack tip in the last chapter. This chapter takes that work a step further to investigate any size effects for a propagating crack on the energy release rate and atomistic crack-tip phenomena.

A. Introduction

Various work has been done to derive macro-scale traction-separation laws for use in Finite Element Analysis (FEA) from Molecular Dynamics (MD) and multiscale models.

Song et al. studied crack-tip shielding.[86] They found that at the point of fracture there is a unique traction displacement cohesive zone law along the fracture independent of the position of the anti-shielding dislocation. However, for crack propagation, the atomistic model shows that, as an anti-shielding dislocation approaches the crack tip, it causes less anti-shielding than predicted by the singular-crack model. If the cohesive strength is reduced then the cohesive-crack model is consistent. The difference is due to the non-linear deformation of material around the crack tip, which cannot be fully represented by a cohesive zone law. Their simulation method used was CADD and they found excellent agreement with the values obtained from independent atomistic calculations on this material. This shows that crack initiation behavior is different than crack propagation behavior, in the sense that they must be modeled independently.

Yamakov et al. studied the inter-granular failure of a Σ99[110] symmetric tilt boundary in Aluminum.[68] Under hydrostatic tensile load, the crack propagates brittly in one direction and ductily in the other. This is consistent with Rice's criterion for cleavage vs. dislocations blunting. The preference for twinning over dislocation is consistent with the Tadmor and Hai criterion. Two separate traction separation relations are extracted from MD for brittle and ductile decohesion to be used in higher scale FEA models and coupled to MD. Their group then used the ESCM method to couple MD with FEA with the addition of CZM elements near the MD-FEA boundary based on the cohesion measured in the MD domain. In this way, cracks that nucleate or originate in the MD

domain will be able to propagate into the FEA domain.[87] Their coupling scheme is the ESCM method which statistically averages the atomic properties to apply as boundary conditions to the FEA mesh without the mesh being required to be the same size at the atomic lattice constant. Since ESCM provides traction forces to the MD domain that means that the MD boundary has a surface. To compensate for this they partition the local MD surface boundary into domains and provide a correction force that compensates for the different stiffness and surface tension. The statistic nature of ESCM allows for finite temperature of the MD domain.

Choi and Kim used a nanoscale planar field projection of atomic decohesion and slip in crystalline solids based on a new orthogonal eigenfunction expansion of the elastic field around an interfacial cohesive crack.[72] The atomistic fields are obtained from molecular statics simulations of decohesion in a gold single crystal along a $[\overline{11}2]$ direction in a (111) plane, using the EAM potential. The field projection yields the traction and displacement as well as the surface stress of the nascent surface. Thus the traction separation and surface energy gradient can be measured as functions of the cohesive zone displacements. It is shown that there is a nanoscale mechanism of decohesion lattice trapping or hardening caused by the characteristics of non-local atomistic deformations near the crack tip.

Fan and Yuen used hierarchical multiscale analysis for interfacial delamination by using MD to model the chemical phase that bonds the two bulk materials together.[40] From MD they use the obtained traction-displacement information to define cohesive zone parameters for larger scale FEA models. Their results predicted the failure force to be about twice as large as the experimental data. They attributed this discrepancy to be due to an increased interaction cross-link across the interface and the presence of voids and impurities inside the real samples. Their approach for this application is advantageous since it avoids the time-scale problem of concurrent multiscale.

Coffman et al. used cohesive laws derived from atomistic simulations for polycrystalline structures.[88] They found that the levels of external stress are required to fracture GBs. This indicates that fracture initiation is likely dominated by irregular atomic structures along GBs. Thus the cohesive properties alone are not likely to be

sufficient for modeling the fracture of polycrystals using continuum methods. Their explanation is similar to Fan and Yuen's for their adhesion force discrepancy.

Vatne et al. used the QC method to investigate crack propagation in BCC Iron under different crystallographic orientations in mode-I loading with various T-stresses.[89] They found that the mechanisms at the crack tip and the critical stress intensity factor, K_I are sensitive to both the crystallographic orientation and whether or not the boundary conditions were isotropic or anisotropic. Due to their small model size their boundary conditions become very important; in their implementation they provide boundary conditions that match the LEFM displacement field. They observed such mechanisms as cleavage, twinning, and dislocation emission.

These techniques illustrate the possibility of coupling lower scale phenomena and behavior to higher scale models.

B. Simulation Model and Methods

The model used for the crack propagation simulations were essentially the same as described in Chapter IV. Whereas in Chapter IV the model width was the same for each model size in this Chapter the model height and width are the same so as the height changes so does the width preserving the model as a square. Refer to the figure in Chapter IV of the GP-FEA model as the GP model and the interface design is the same. The only changes are in the FEA mesh used. However, it is worth noting that the GP model has crystallographic orientation of (110) [001] such that a horizontal crack propagating along the positive X direction [110] creates a crack plane parallel to (110). This orientation was chosen as it is the most brittle having the lowest Griffith critical stress intensity of about 0.84 MPa $\sqrt{m}$ correlating with the lowest {011} surface energy of about 1.65 J/m^2.[89] The Young's modulus chosen for the FEA continuum given this orientation was 224 GPa.[89]

The Auto-Duality Domains (ADDs) were automatically decomposed when their average maximum principal stress reached 12.0 GPa. There are a total of 25 ADDs each 2 x 8 nm^2 spanning the crack propagation trajectory from X= -14nm to X=36nm. These ADDs functioned the same way that the local domains used by Yamakov et al. were monitored for their stress and displacement values used to derive an atomistically-based

cohesive traction separation (T-S) law.[68] In this study the same concept is used to calculate the critical energy release rate, G_C, for each of these ADDs as the crack propagates through them. This provides a picture of the energy required to further an existing crack configuration; from an ideal initial notch to a nascent brittle crack as it evolves.

C. Crack Propagation Results via GP-FEA

Just as the simulations described in the last chapter, before coupling to the FEA mesh, the GP model equilibrated in the NPT ensemble for 100 ps at 100 K and 1 atm to ensure configurational stability. The GP model then had its temperature increased to 300 K and allowed to equilibrate for 20 ps before being attached to the FEA mesh. The loading procedure kept the thickness constant and no deformation was permitted along the thickness direction to mimic plane-strain conditions. Load was applied in 0.5% strain increments producing a strain rate of about 25% strain/nanosecond. For each increment the strain was applied at the FEA mesh outer boundary then 20 ps of relaxation at 300 K was allowed for the WG-GP subsystem or until the GP-FEA interface converged triggering the next load step.

When each model was at 1% strain, before the initial notch propagated, the stress intensity was calculated based on the formula described in Chapter IV, their values are plotted in Fig. 32. Each data point in the figure is accompanied by an image of the atoms at the crack tip. The atoms of a lighter color (magenta) have local coordination of 13 (CN=13), meaning that they have only 13 nearest neighbors. It can be seen that the smaller the model size the greater the stress intensity, but as the model size increases the stress intensity converges to a constant value. This small variation is due to the different model width sizes, it also corresponds to slightly different patterns of atoms with CN=13.

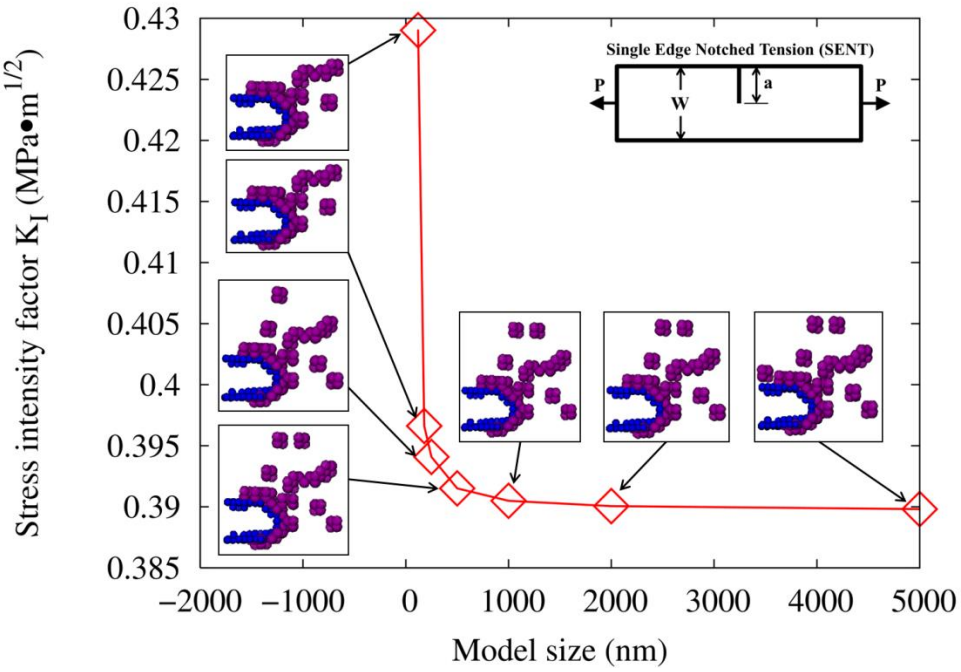


Figure 32. Size effect on the stress intensity factor K_I at 1% strain, including the atoms with CN=13 at the crack tip.

As the crack propagated through the entire GP portion of the model, each ADD recorded the transient stress and separation data in order to derive a cohesive zone model and local fracture toughness. Certain phase transformations occurred at the crack tip and will be discussed as a method of accuracy convergence analysis.

1. Fracture Energy

The stress that was monitored in each of the AD Domains was integrated with respect to the displacement that the domain underwent. When the traction goes down to zero this provides the energy release rate of that material domain. This integration was performed for each of the AD Domains for each model size. Figure 33 shows these values for each AD Domain after the crack had propagated entirely through the model. Since each domain is located at a specific distance from the initial pre-crack their response can be correlated to the distance the crack had to travel before reaching the specific AD Domain. A trend can be seen in Fig. 33 where the energy required to propagate the crack increases farther from the initial pre-crack location. This is consistent

with the three stages of crack growth resistance behavior in small scale yielding.[83] Where during the initial stage the crack is essentially stationary; the constant slope is caused by crack tip blunting. The stage at which the energy release rate increases is the transition between blunting of a stationary crack and crack growth under steady state conditions during which a rising R-curve is possible. For the small scale yielding assumption this function (the J-R curve) is a material property.[83]

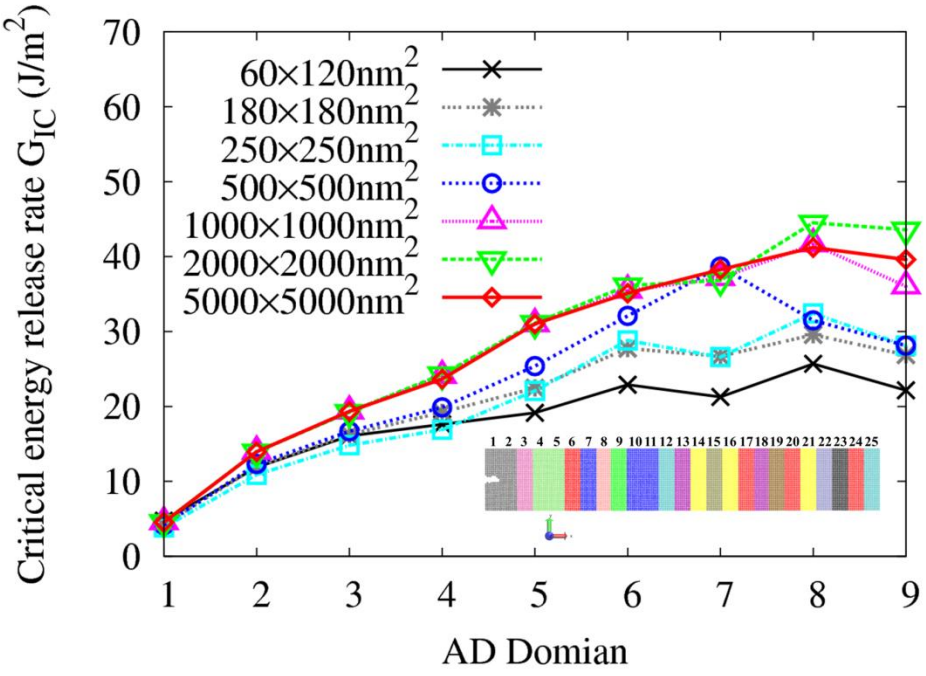


Figure 33. Energy G_{IC} distribution during cracking the local domains.

Another more interesting trend that can be seen in Fig. 33 is that of the model size comparison. Notice how the smaller model sizes have generally lower critical energy release rates than the larger models. Not only do larger models have greater critical energies but they also appear to converge as the model gets larger. This convergence can be visualized in Fig. 34 that plots the Critical energy release rate, G_{IC} for local domains 5 and 6, showing a strong convergence trend.

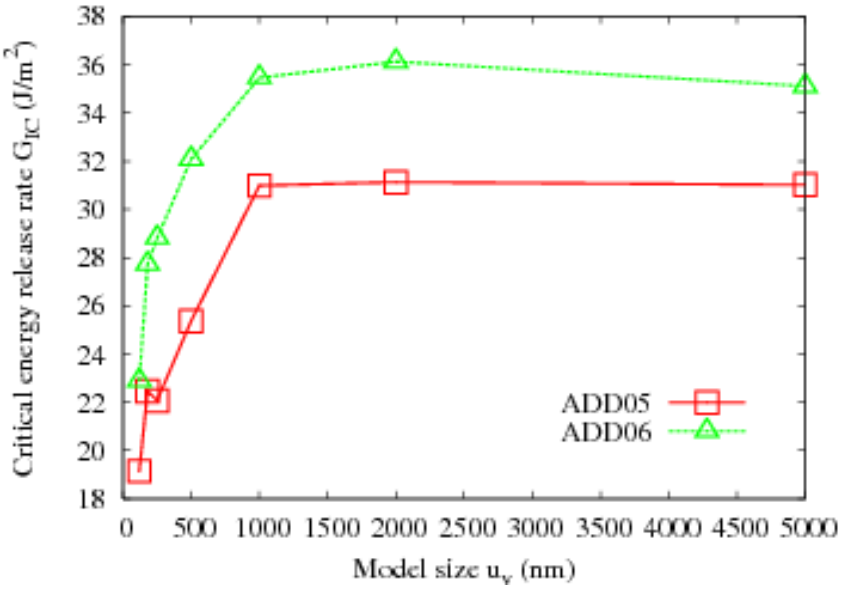


Figure 34. Critical energy release rate, G_{IC} trend with model size for local domains 5 and 6.

Since the GIC values from this figure are calculated by integrating the T-S curve, various T-S curves are plotted in the following two figures below. The first figure shows the T-S curves for ADD 6 and the second for ADD 7. Notice how the smaller models have very sharp T-S relations, this is indicative of brittle fracture as a sudden event would cause a rapid loss of strength, however the larger models have much fatter curves i.e. large stresses continue even after appreciable displacement, indicating more ductile failure, as material deforms it is able to accommodate larger strains at the cost of lower strength.

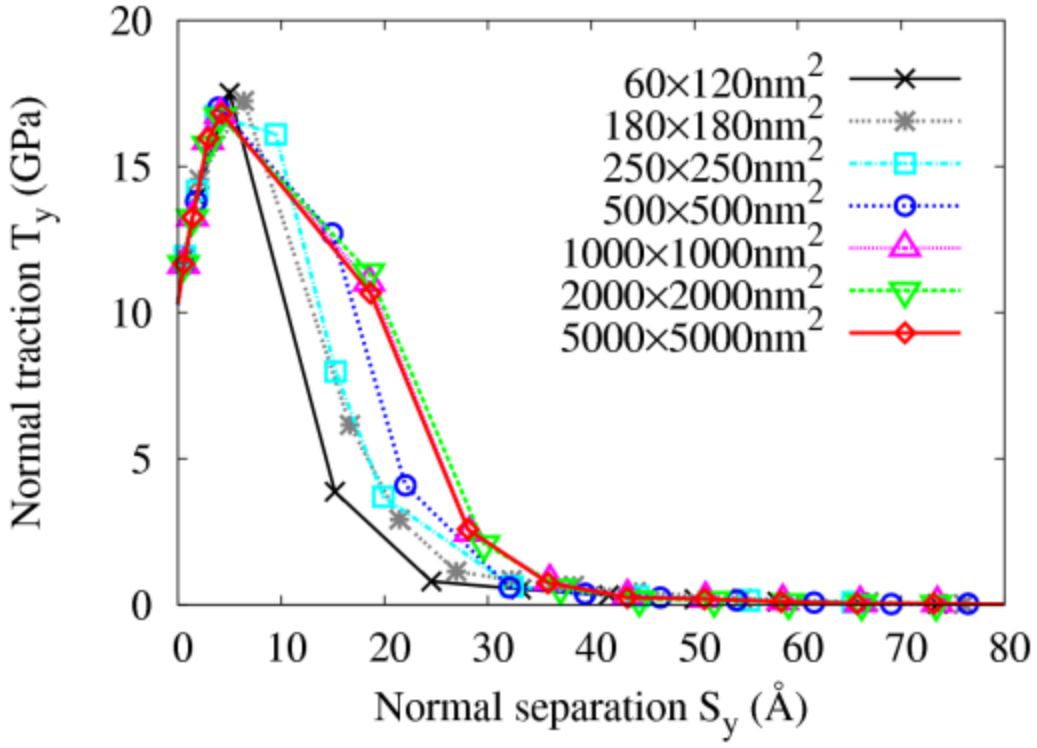


Figure 35. T-S response for domain 6.

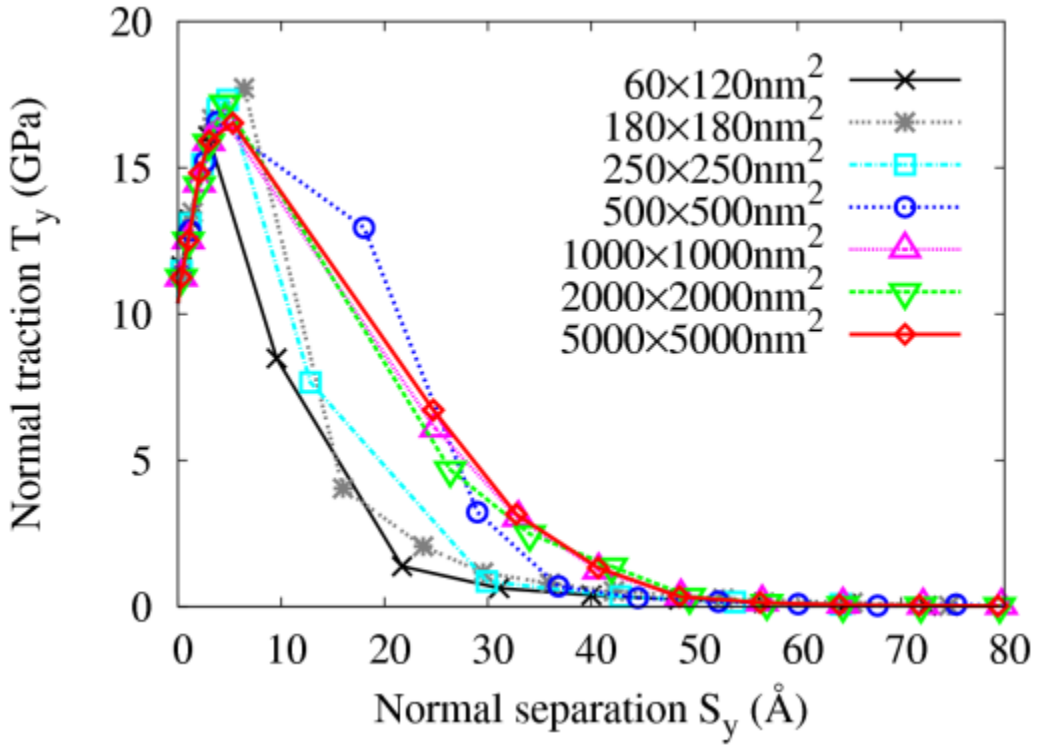


Figure 36. T-S response for domain 7.

The model size effects can also be seen directly in the T-S relations. Notice how the models larger than 500 nm have almost the same T-S relation. This is the case for both AD Domains shown as well as those not shown.

In an attempt to understand why some AD Domains behaved tougher than others, a detailed look at the crystallographic orientations and ductile behavior was conducted.

2. Crack-Tip Phase Transformation

Vatne et al. showed that this crystal orientation produced a BCC to FCC phase transition perpendicular to the crack path symmetric about the crack plane.[89] They mention that the periodicity of this FCC phase is along the [001] direction with respect to the BCC phase and [110] for FCC. The orientation relationship is $(110)_{BCC}//(111)_{FCC}$ and $[001]_{BCC}//[110]_{FCC}$ which coincided with the Nishiyama-Wassermann orientation relationship.[90] This same phase transformation was found during MD simulations as a meta-phase for grain formation within existing BCC grains in nano-structured iron.[91] This same phase transformation can be seen in the results of this study from Fig. 37.

Wang et al. observed the same kind of structural phase transition in BCC Molybdenum under deformation loads within a transmission electron microscope at room temperature.[92] They demonstrated that this transformation is accompanied by shear deformation from the original <001>-oriented BCC structure to a <110>-oriented FCC lattice at crack tips during straining at room temperature. Their crack orientation corresponds to Vatne's orientation-4; (010)[001] with the cleavage plane 45 degrees from the initial crack plane.[89] This BCC to FCC phase is consistent with the Nishiyama-Wassermann orientation relationship.[90] However, Wang et al. also found that this FCC domain reverted back to BCC, however in a <111>-orientation, equivalent to a lattice rotation of 54.7 degrees. Although Vatne's work also illustrates a secondary BCC phase that interacts with the FCC phase in front of the crack tip they do not go into detail about the secondary BCC phase. Wang et al. corroborated their experimental findings with MD simulations and found that the relation between the FCC phase and the secondary BCC phase follows the Kurdjumov-Sachs relationships.[93] This secondary BCC phase is interesting although not observed for the orientation used in this study.

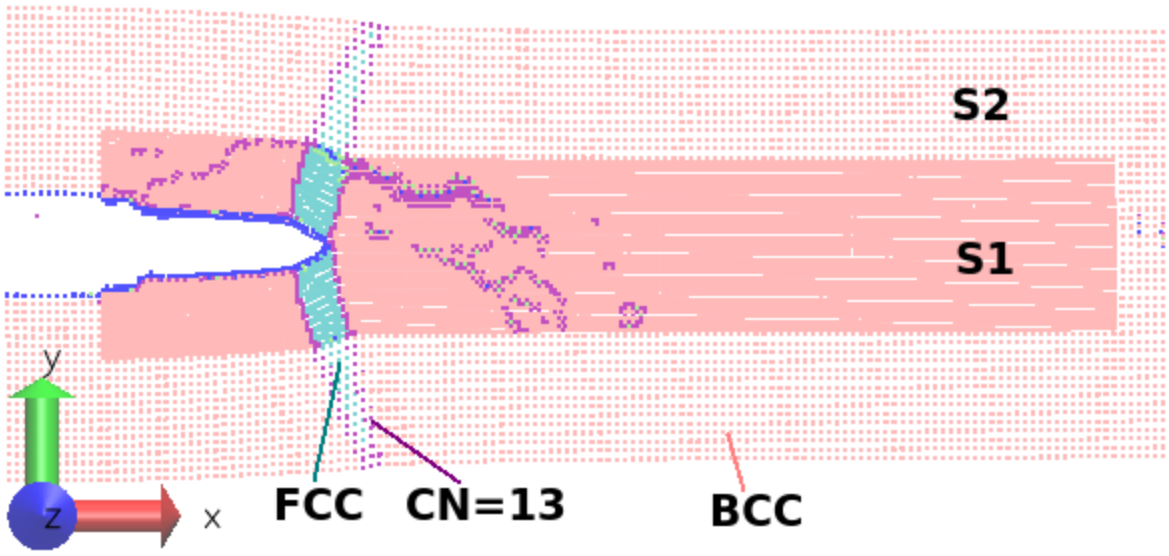


Figure 37. Example of crack-tip phase configuration for the 1 micron sized model. Colored by CNA.

The FCC phase is seen to be shrouded in an atomic layer of atoms with 13 nearest neighbors (CN=13). For the characterization tool used to measure the coordination is set to count the nearest and next nearest neighbors when in a BCC phase yielding a CN=14, however it counts only the nearest neighbors for FCC phases, providing a CN=12. For more information about Coordination Number (CN) and Common Neighbor Analysis (CNA) please refer to Appendix F.2.a. Figure 37 shows the phase pattern within the atomistic (S1) and scale-2 (S2) GP domains. Notice that scale-2 is able to reproduce the phase transition in the appropriate configuration. This indicates that the GP scale interface does not strongly inhibit the growth of the FCC phase.

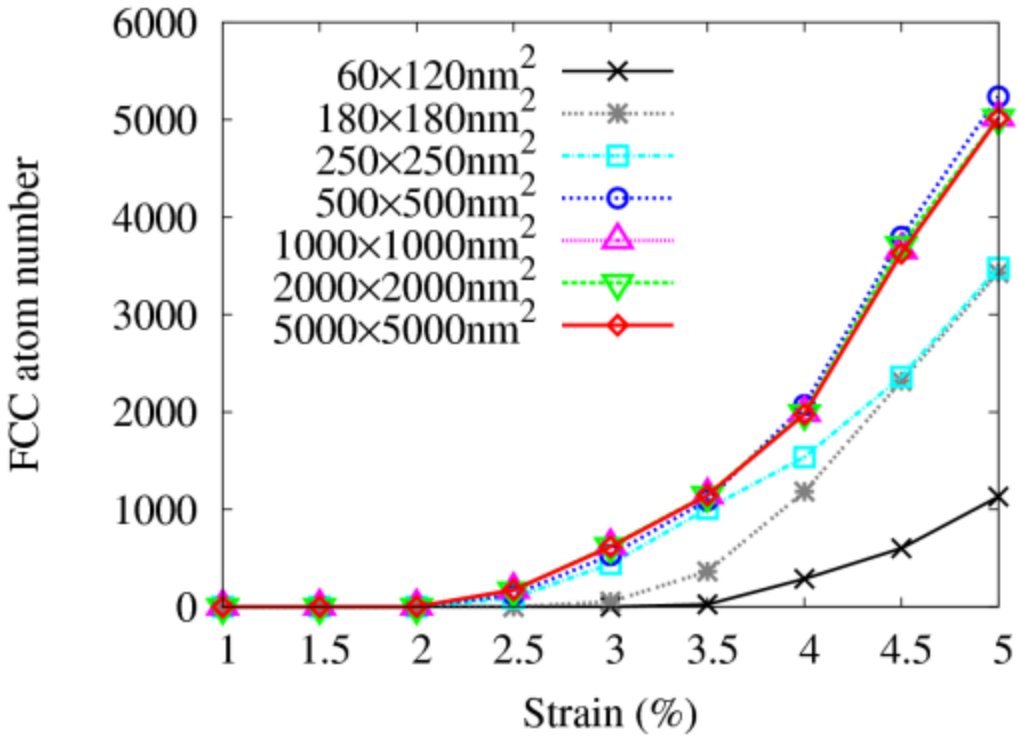


Figure 38. FCC evolution during crack extension.

For different crack propagation lengths the size of the FCC phase varies. For small strain and short propagation distances the FCC phase is small but for longer propagation distances the FCC phase grows until it reaches a steady state size. The growth of this phase may be the reason for the growth in the J-R curve of the material. Figure 38 shows the growth of the FCC phase by counting the number of FCC oriented atoms in the S1 domain for each of the model sizes. Notice how the smallest model has the smallest FCC phase for all load strains; but as the model size increases, the size of the FCC phase converges to the same strain relation. Models 500nm and larger all are seen to have the same FCC size for a given strain level. Figure 39 shows the number of atoms that are part of the transition layer between the BCC and FCC phase. These curves show a similar model size dependence, not so clearly defined by the number of the atoms but rather by the delay. Smaller model sizes do not nucleate the FCC phase as soon as the larger models do. Perhaps the smaller model's boundaries are too close and influence the amount of freedom that the atomistic domain has to nucleate the FCC phase.

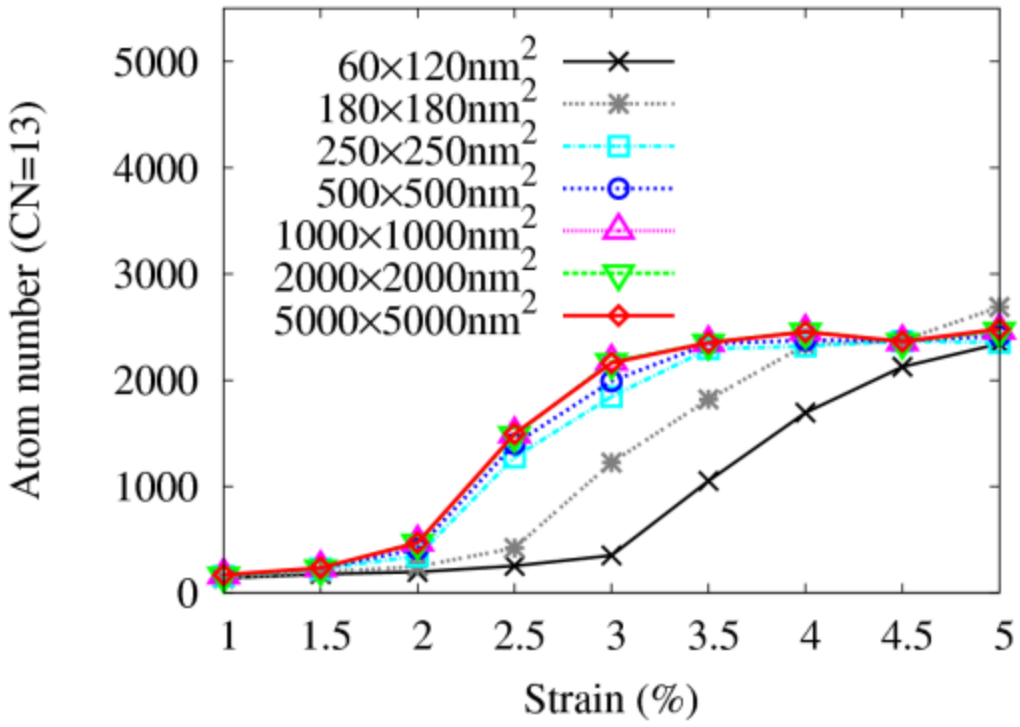


Figure 39. Mid-phase evolution from BCC to FCC.

D. Conclusion

The results of this study show that the size of the model affects the material behavior by influencing the atomistic phenomena at crucial locations as a crack tip. Two effects were found that affect the phase transformation at the crack tip. First, it was seen that smaller models do not have as many atoms in the FCC phase as larger models do for the same loading strain. Second, the smaller models were delayed in their nucleation of the FCC phase.

These effects show that the chosen model size of the simulation can seriously affect atomistic phenomena observed at crack tips. If the researcher is looking to derive critical information from atomistic-based simulations the model size must be carefully chosen as both the size and nucleation strain of the FCC phase is important if characterizing a fracture resistance relation (J-R curve) for use in higher scale modeling.

APPLICATIONS IN IMPACT

Many computational methods work by minimizing the potential energy which yields a quasistatic model. For many cases this is sufficient, however for applications which involve a decent amount of dynamics such as thermal activation, and kinetic energy transfer, this approach is insufficient. Thermal activation can be included in stochastic modeling but not explicitly. For explicit dynamics Newton's equations of motion are usually used to model atoms or material particles. In many ways the propagation of stress waves are very important when considering damage nucleation in materials under high speed loading conditions such as an impact. To understand the material response to such conditions, analysis at the atomic scale should be conducted, to investigate the nucleation of dislocations, zones of amorphousness, nano-cracks, phase changes, etc. This atomic phenomenon can better help us to understand the high scale material behavior. There has been some work in the area of dynamic multiscale analysis with atomistically-based methods. This chapter will describe some of them and show how the GP method may be extended for use in such cases. It will also discuss how the GP method should grow to better account for certain dynamic phenomena.

A. Introduction

Continuum level modeling of dynamic impact of a thin glass sheet was simulated by Hu et al. using the peridynamic method to investigate the dynamic fracture pattern under various boundary conditions.[94] Their model size was the same as the experiments at 10x10x0.3 cm^3, impact velocities ranged from 61 to 200 m/s. Their results show very good qualitative agreement with the experiments. Although their work is very impressive and useful, it does not explain nor imply what atomistic phenomena is involved. Their only measure for fracture is a critical bond-stretch that correlates to the material's critical energy release rate.[95]

Branicio et al. used a 200 million atom MD simulation to model atomistic damage in AlN from a hypervelocity projectile impact at 15 km/s.[96] They found a phase transformation, following the initial elastic compression wave, from the usual wurtzite to

a rocksalt phase which is stable at lower volume and higher energies. Behind this phase transformation wave is a source of nanocavities and kink bands. As the wave returns, being reflected from the other side of the sample, mode-I cracks nucleated from the nanocavities and mode-II from the kink band superdislocation boundaries. Although very interesting results were found there are a couple draw backs to their simulation design. Firstly the size of their impactor had 500,000 atoms which is supposed to represent an armor pricing bullet; but was this sufficient to transfer the correct momentum and energy flux? Secondly in order to have more realistic boundary conditions their model size had to be extremely large; 200 million atom MD simulations still require a long time to run even on the most advanced supercomputers. The authors may benefit from using an atomistically-based concurrent multiscale technique that can handle the dynamics required by their impact problem.

Wang wrote a dissertation about an adaptive multiscale method for modeling nonlinear deformation in nanoscale materials based on the QC method.[97] Wang proposed a remeshing technique using a critical strain or energy criterion when the homogeneity of the microstructure/ deformation was violated and implemented finite temperature into the QC method via a local harmonic approximation which integrates over all available normal modes of vibration to derive the equilibrium entropy. The example application for their method is nano-indentation using the Mixed Penalty functional in the perturbed Lagrangian to implement contact from an impactor. Even though finite temperature is implemented the simulation still maintains the quasi-static nature of the QC method, thus the atomistic domain is not governed by MD or Monte Carlo (MC) but rather Molecular Mechanics (MM). This means that high velocity impact cannot be accurately modeled since stress waves would not propagate through time, an explicit version of QC would be needed.

Lidorikis et al. proposed a concurrent multiscale method that bridges the atomistic domain with the continuum FE mesh through a linear average of each domain's Hamiltonian[98,99] which is similar to the Bridging Domain Method (BDM) by Xiao and Belytschko[100] as they both derive their methodologies from Broughton et al. for wave reflection suppression.[101] They show that their scale coupling method is accurate for both static and dynamic cases. The dynamic case is illustrated with an impact on an Si/Si3N4

interface. Farrell and Park et al.[102,103] used the Bridging scale method (BSM), although without a hand-shake region for blending the energies, to study wave and intersonic crack propagation capturing the formation of daughter cracks. They both found no wave reflection from the scale interface region and both validate their models by comparing them to the full MD simulations. These methods show an overall high accuracy for all of their capabilities; however their methods require the element size to be the same as the atomic lattice in the scale interface domain thus making it a direct coupling method (DC) which is not the most efficient for saving degrees of freedom in the continuum.

Guo et al. implemented a unique hand-shake domain composed of coarse-grain like material points; their Material Point Method (MPM) is able to smoothly transfer the atomistic information to the FE continuum via modified interpolation shape functions to reduce artificial forces on the hierarchical background grids.[104] They tested the MPM by using a step-like wave and a wave packet propagating within a bar. They were able to implicitly include the short-wavelength phonons and their dissipation into the MPM region by weakly coupling the region to a Brownian heat bath whose dynamics are set to the simulation temperature by invoking equipartition and correcting for the lost degrees of freedom.[101] Thus Newton's equations of motion are replaced by Langevin equations for this higher scale.

In the next section the GP method will be investigated for its ability to handle dynamic wave propagation through the use of auto-duality domains to save degree of freedom. Domains are used to decompose higher scale particles into atoms in regions that have high energies in an attempt to maintain a degree of accuracy.

B. GP Model for Wave Propagation

An MD model of two infinite Copper plates impacting each other will be used to validate the auto-duality capability of the GP method for use in dynamic wave propagation. Each plate was 5.262 nm thick, the simulation cell was 4.355 nm wide and broad and periodic in those directions to mimic an infinite plate, see Fig. 40. The impact surfaces of each plate were designed to match each other, such that upon impact they will form a single crystalline copper plate. The plates equilibrated at 10 K about 4 nm apart from one another; the low temperature was used to reduce the thermal noise in the results.

After 15 ps of equilibration each plate was given an initial velocity of v=143 m/s opposing each other such that their impact velocity was the sum of each.

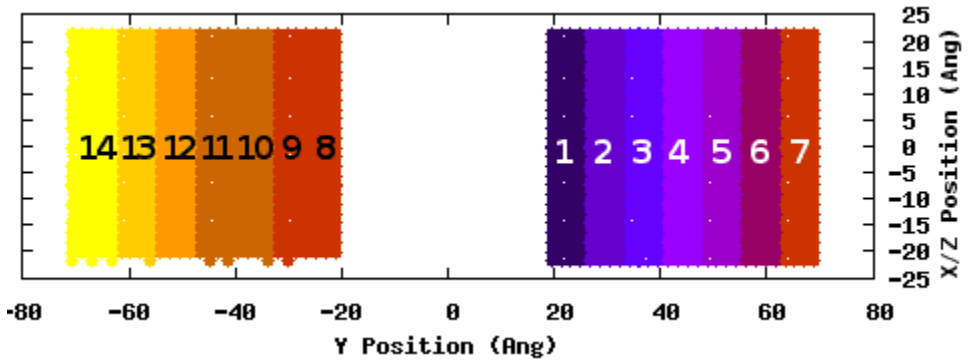


Figure 40. Impact model configuration, each numbered rectangle was individually monitored for stresses and energies and was later used as an Auto-Duality domain.

For the dynamic case, the auto-duality routine as described in Section II.A.4 needs to be augmented. As it was previously described, the imaginary atom domain follows the deformation of the real scale-2 particles, such that when decomposed the newly made real atoms will be in approximately the same deformation field as the now imaginary S2 particles. This works well enough for quasi-static simulations where the primary state variable to be minimized is the potential energy of the system. For the dynamic case, the new atoms, after decomposition, will need to be assigned the proper velocities and accelerations in the same way as they are given positions in the deformation field.

For this initial test case, the newly made real atoms/particles' positions are determined by the average position of the constituents in their former NLC. In the same way as position is determined, their velocity and acceleration is also determined. In this way both the potential and kinetic energy are approximately conserved before and after the decomposition/Lumping process. A word of caution: after decomposition the averaged field variables for the new degrees of freedom will roughly follow the same distribution as the particles had. This will not produce the correct Boltzmann distribution of velocities to maintain the temperature, which is the main reason for running this test at 10 K.

C. MD Results for Wave Propagation

The global pressure average and kinetic energy was monitored to confirm general wave consistency. Figure 41 shows their evolution in time; since negative and positive pressure correspond to greater strain energy it makes sense for the pressure frequency to be half that of the kinetic energy.

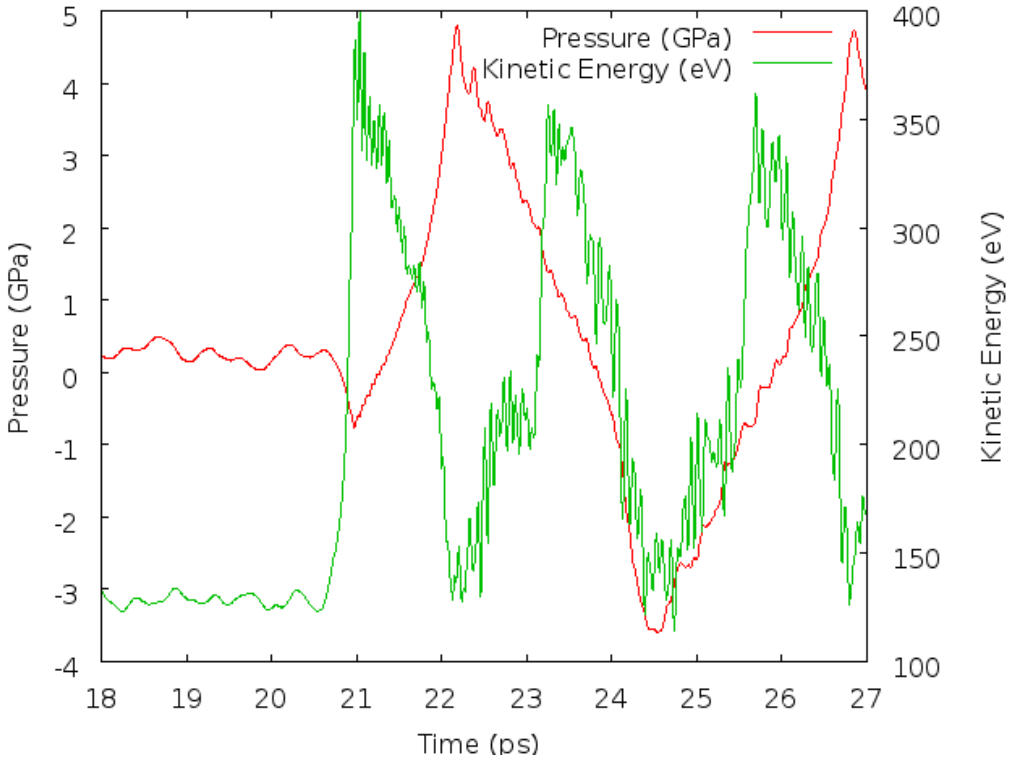


Figure 41. Kinetic energy and pressure evolution for the pure atomistic model at the time of impact. The average kinetic energy oscillates around 250 eV, and the pressure has amplitude around 4 GPa.

Each of the local domains shown in Fig. 40 was monitored for their maximum principal stress evolution and is illustrated in Fig. 42. These values will be compared later and used as criteria for auto-duality. Notice how ADD1 is the first domain to experience the impact; however it first feels a tensile force due to the adhesion forces between the atoms of each plate. This adhesion force adds kinetic energy to the plates hastening their eventual meeting. After this tensile force there is a compressive stress which is from the impact velocity. This wave can be seen to propagate into the neighboring AD Domains at a constant rate and reflect off of the back of the plate in ADD7.

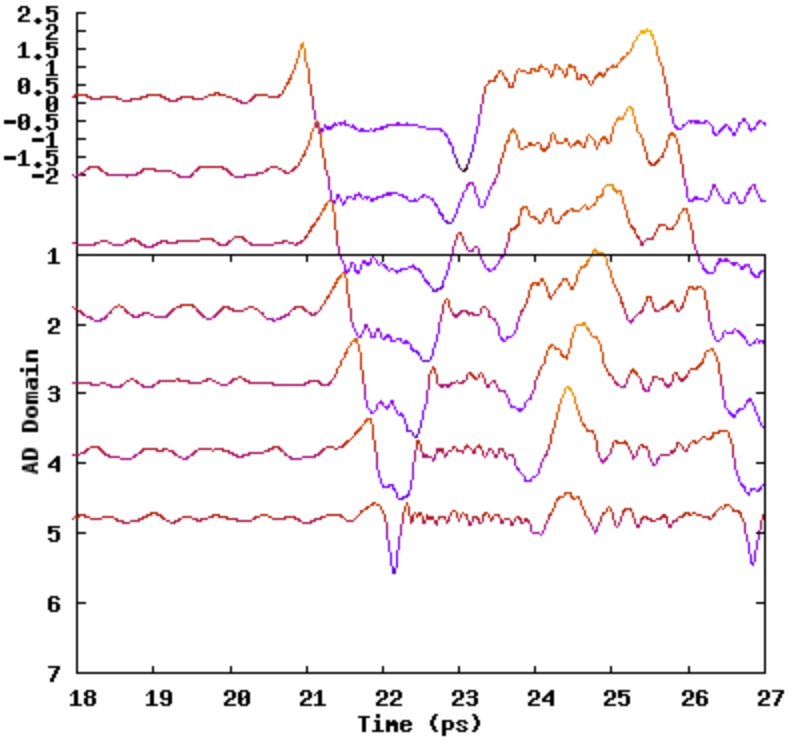


Figure 42. Max principal stress evolution of auto-duality domains 1-7.

For this pure atomistic model the impact configuration is found to be consistent with what was expected, considering that scale-1 of the GP method corresponds directly to Molecular Dynamics. These results set a firm foundation for the subsequent tests.

D. Scale-2 Results for Wave Propagation

A pure scale-2 model is necessary to choose an appropriate decomposition stress for the auto-duality algorithm. The pressure and kinetic energy evolution results for the pure scale-2 impact model are shown in Fig. 43. Two main differences are immediately noticeable, first that the frequency of both pressure and kinetic energy are twice as the atomistic model; this indicates that the elastic wave speed is twice as fast as it should be for this material. The second is that the kinetic energy levels are much higher than the atomistic. However the total energy is roughly consistent as evidenced from the proper relation between pressure and kinetic energy.

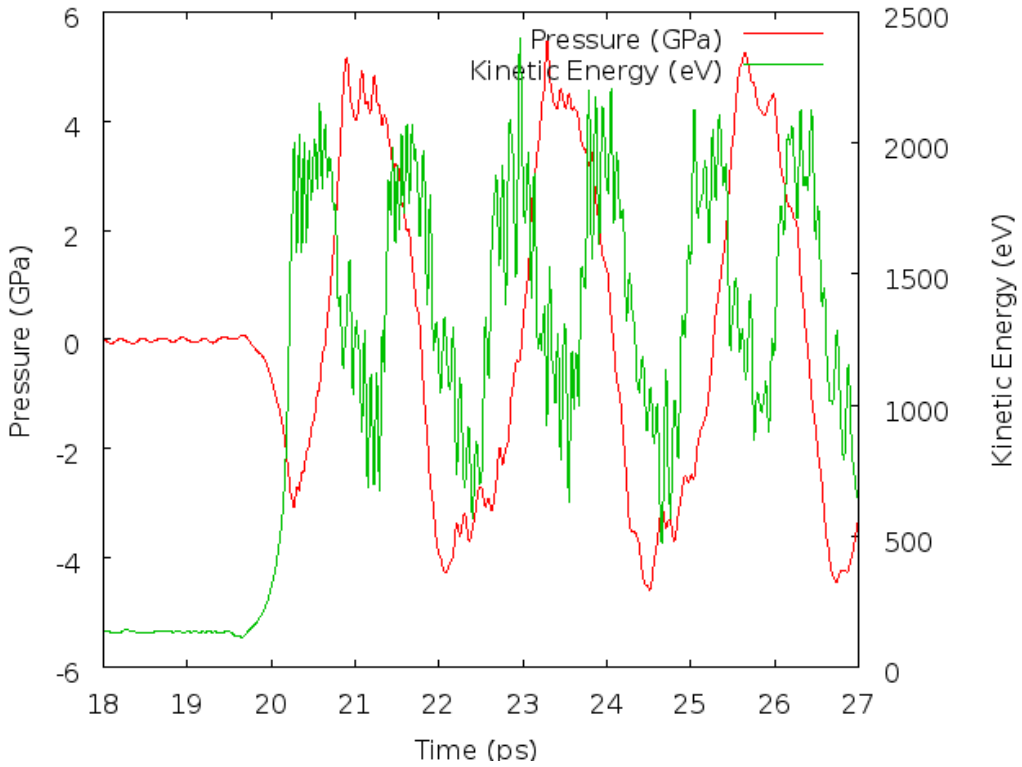


Figure 43. Kinetic and Potential energy evolution for the pure Scale-2 model at the time of impact.

The kinetic energy oscillates around 1300 eV, and potential around -56025 eV. The potential energy deviates by 550 eV from the pure atomistic model at about 1% error, and the kinetic by about 1050 eV, 420% error. It is also seen from the period of the waves that the wave speed in scale-2 is twice as much as the atomistic model. This wave speed discrepancy comes from the difference in strain gradient between GP scales. This causes a factor of two difference between the strain gradient and the acceleration of a given material point for a scale-2 domain, effectively increasing the perceived wave speed.

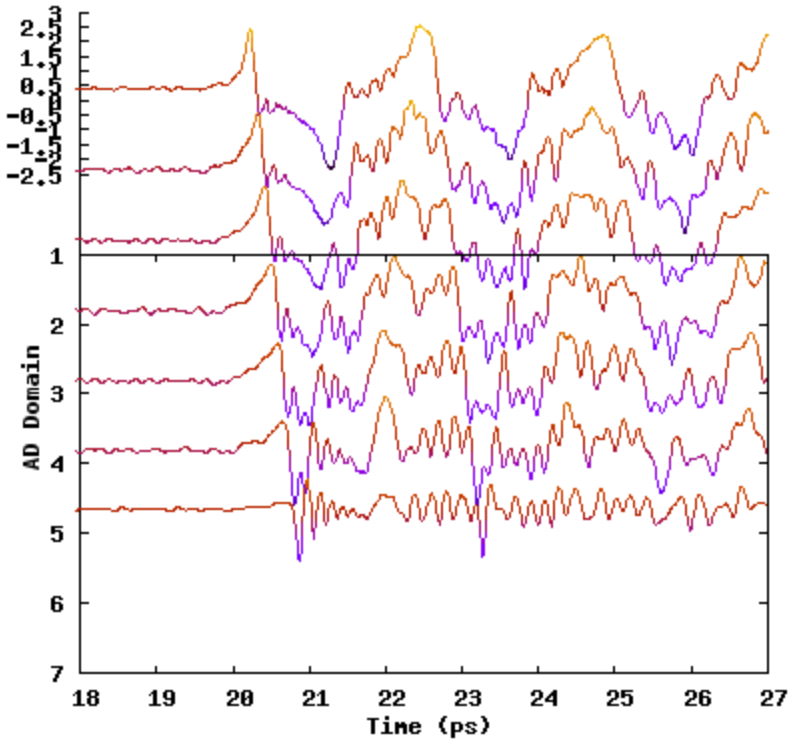


Figure 44. Max principal stress evolution of auto-duality domains 1-7.

The total energy of the scale-two model is about -54725 eV, about a 1600 eV difference from the pure atomic model. Could this difference be explained by surface effects? Since both models impact forming a single crystalline plate the interatomic potentials cause the two approaching surfaces to be attracted to each other, this attraction force causes the plates to accelerate toward each other. However, scale-2 surfaces feel twice the attraction force as an atomic surface, since each scale-2 particle interaction feels the equivalent force of all the atoms it represents. Thus an S2 particle force equals $F_2 = k^3 F_1$ or $8F_1$ when $k=2$. This equates to the same acceleration as an atom would feel due to the larger mass of the particle. Since the acceleration is the same as an atom would feel but the distance to traverse is twice as large the particle's Work of Attraction is twice as much as an atomic surface. This also means that a scale-2 surface spends a longer time under the attraction field as an atomic surface and would have an impact velocity of $v_0+v_1\sqrt{2}$, where v_1 is the velocity from the atomic surface attraction.

These surface factors explain the difference in kinetic energy and illustrate the need for using auto-duality to reduce these effects. It also explains the source of higher

scales' increased surface effects which necessitate the use of surface images as described in Chapter II.A.3. In the next section will show whether the auto-duality concept is sufficient for minimizing these dynamic scale problems.

E. Auto-Duality Results for Wave Propagation

Once the decomposition stress criterion has been chosen the scale-2 model can be simulated again with the auto-duality algorithm switched on. The expectation is that the auto-duality results should better represent the MD results compared to the scale-2 model. The criterion for decomposition was set to 1.0 GPa and the criterion for lumping to 0.5 GPa. Figure 45 shows the lumping and decomposition events during the simulation. The same wave propagation in each AD Domain can be inferred from this figure. Recall that ADD1 impacts ADD8 of the other plate.

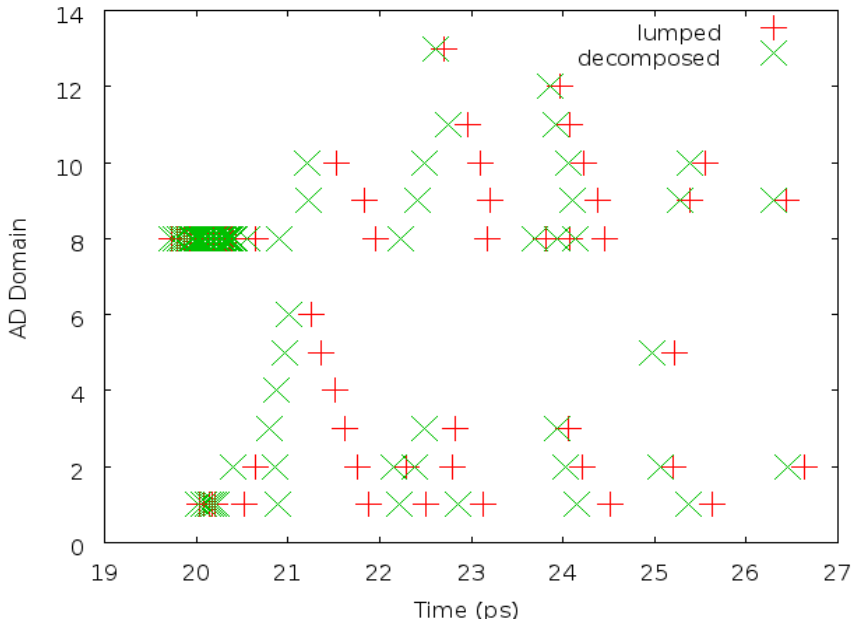


Figure 45. Lumping and Decomposition events for each AD Domain as a function of time. ADD 1 and 8 interfaced at impact.

Notice the rapid lumping and decomposition of ADD8 and ADD1 right before impact. This is due to the previously discussed problem with scale-2 having a longer interaction distance. When the two domains are lumped as S2 particles the two plates can feel each other and feel a tensile force which is large enough to cause decomposition,

however upon decomposition the newly made atoms have a smaller interaction distance and the two plates cannot feel each other and do not have the tensile stress thus they relax and become lumped as S2 particles again. This rapid sequence causes an additional stress oscillation to the surface of the plates. However this effect occurs only initially right before impact and will be neglected for the general comparison.

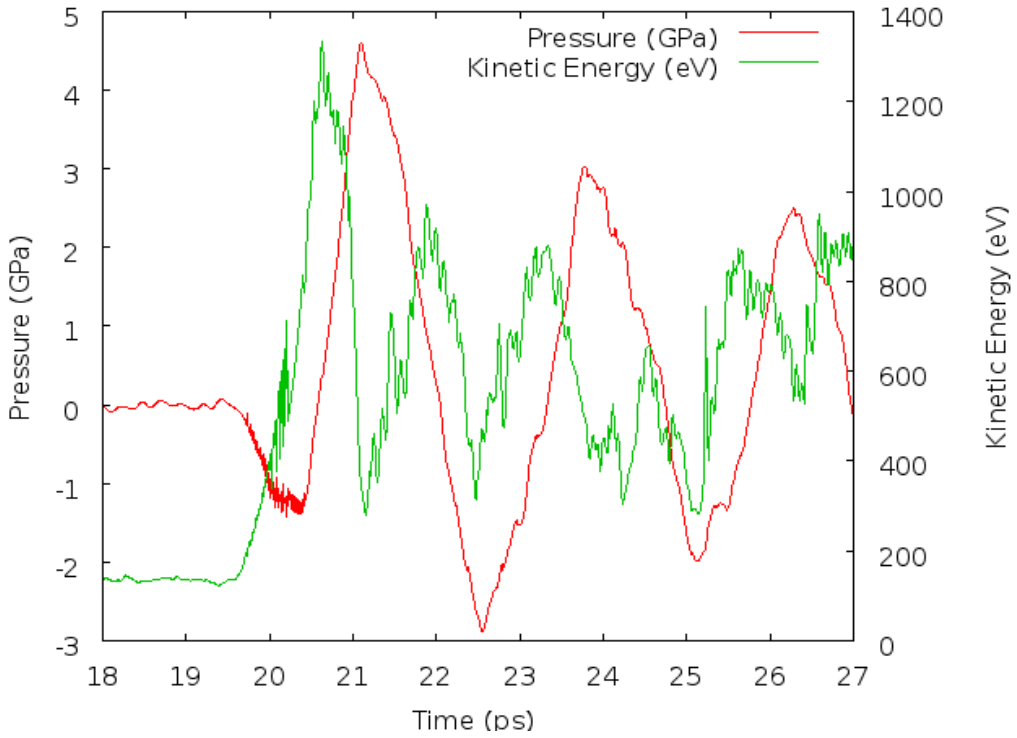


Figure 46. Kinetic energy and Pressure evolution for the auto-duality model at the time of impact.

From a global perspective the oscillation of the pressure and kinetic energy appear to be very similar to the frequency of the pure scale-2 model suggesting that the auto-duality procedure does little to prevent the increased wave speed in the material. The average kinetic energy is about 600 eV which is 140% error from the atomistic, but is much less than the pure scale-2 which had an error of 420%. The auto-duality model still suffers from an increased kinetic energy from the scale-2 surface adhesion, however to a lesser extent due to the decomposition. It is also noticed that pressure oscillations decay which the pure scale impacts do not exhibit. This decay or energy loss is due to conservation errors during the composition processes, i.e. both the decomposition and

lumping procedures. When changing the scale the imaginary become real and the real become imaginary. The newly real particles or atoms are positioned based on a simple geometric average of the positions of their NLC constituents. For a propagating wave this simple average produces a numerical error which manifests as an energy loss. This energy loss may be minimized by improving the averaging algorithm or by applying a correction distribution.

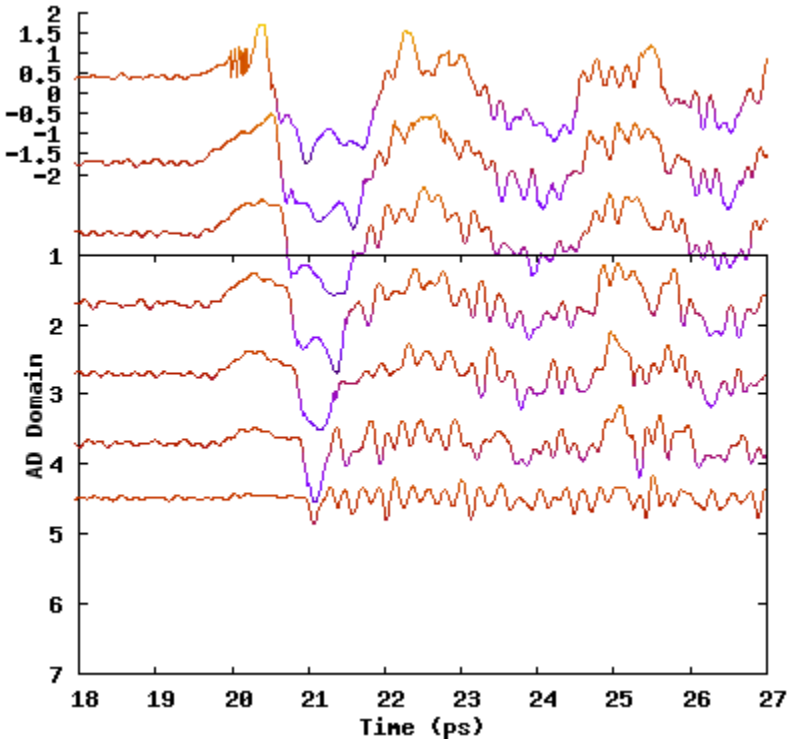


Figure 47. Max principal stress evolution of auto-duality domains 1-7.

F. Summary and Recommendations

It is shown in this Chapter that the auto-duality feature of the GP method can be used to help mitigate the unrealistic wave propagation dynamics caused by higher scale GP representations. This illustrates the need for atoms in locations where the deformation gradient is large and that higher scale particles may be used in areas of small deformations. There are two dynamic problems that occur within GP scales higher than the atomistic scale.

GP Dynamic Problem 1. The wave speed in higher scales is proportional to the scale ratio. This is due to the farther reaching influence of higher scales for the same

reason why higher scales naturally have a proportionally greater surface effect. It causes the gradient of the strain field to be smaller than it would be for a pure atomistic representation while keeping the acceleration consistent. This mismatch between acceleration and strain gradient causes a perceived increase in the wave speed; it changes the wave equation.

In order to address this problem a more accurate way to calculate the strain gradient must be made. The current inverse mapping method is a very fast and convenient method that is appropriate for quasi-static problems due to its assumption based on the Cauchy Born rule. However, when the strain gradient is non-zero it causes a deviation in the wave equation. If there was a way to interpolate the local/atomistic strain gradient at a given particle location then this problem could be mitigated. Put another way, the non-locality of GP high scale domains increases proportionally with the scale ratio. To maintain the atomistic strain gradient, the same degree of non-locality as on the atomistic scale is needed.

GP Dynamic Problem 2. The temperature of high scales is not well defined; using only the inverse mapping method to calculate temperature does not make sense due to the multiscale nature of velocity distributions that compose thermal energy. Since a particle's velocity is the average of the atoms' that it is composed of, there is no guarantee that the kinetic/thermal energy of those atoms is consistent with the particle that represents them.

From this perspective the temperature of a high scale domain could be represented by a sum of two terms, the particles' kinetic temperature plus the internal kinetic energy of the implicit atoms the particles represent. The larger the particle scale the greater the internal thermal contribution of the implicit atoms (lost DOF) to the domain temperature. The heat from the lost DOF due to lumping may be calculated in the same way as is used in the MPM multiscale method.[104]

This brief dynamical study of the GP method for use in dynamic applications is instructive. It clearly illustrates the needs still wanting in the GP method and suggests certain possible solutions to these tough problems. Current work in this field will help to guide the development of the GP method into a more advanced future incarnation.

PARTICLE-BASED MULTISCALE ANALYSIS PROGRAM (PMAP) STRUCTURE

A. Introduction

Many theoretical models seek to investigate the effects of material defects such as pores, dislocations, microcracks, etc. on the actual applications of processing, loading and service conditions. These applications tend to be continuum level fields such as strain, fatigue, electric field, temperature, pressure, etc. To have these material defects evolve naturally in these fields there must be sufficient freedom such that boundary and model size effects do not modify the natural defect evolution. This multiscale capability is the main goal of the GP and GP-FEA methods that differentiate it from traditional Molecular Dynamics (MD). The features of the GP and GP-FEA methods allow them to contend for this goal opening the gateway to pursue other applications such as materials strengthening and toughening by tailoring the design of the nano and microstructure or as a tool for designing nanotechnological devices and microsystems for different functions.

In order to model these multiscale characteristics of materials and to realize multiscale modeling theories, certain numerical methods must be utilized. Computer programs effectuate these methods; quantitatively tracking the motions of atoms and particles as they move through space and time. Most simulation programs are designed for one type of length and time scale, such as molecular dynamic and coarse-graining programs like LAMMPS[105], DLPOLY[106,107], GROMACS[108], NAMD[109], the discrete dislocation dynamics method which can be simulated by software such as, ParaDis[110], microMegas[111], TRIDIS[112] and finite element analysis software like ABAQUS[79], and ANSYS[113]. There are coarse graining methods that can reduce an atomic structure into representative particles or beads, MARTINI[23] is an example and can be run within MD software like NAMD[109] and GROMACS[108]. They are able to handle two different model scales hierarchically, the atomistic and one level up, a coarse-grained representation within a single program framework. These are mainly designed for soft materials in biological applications their accuracy is based on the atomistic structure thus they are a

Class-I multiscale technique. However it is advantageous for all hierarchical scale simulation methods to be able to run within a single framework where one scale's information can be used in higher scale simulations and results can be directly compared in a simple and easy way.

This is the philosophy that caused the development of the OCTA[114] and VOTCA[115] the Versatile Object-oriented Toolkit. The former is an integrated simulation system which utilizes four different meso-scale simulation engines: COGNAC[116] (COarse Grained molecular dynamics program by NAgoya Cooperation), PASTA[117] (Polymer rheology Analyzer with Slip-link model of enTAnglement), SUSHI[118] (Simulation Utilities for Soft and Hard Interfaces), and MUFFIN[119] (MultiFarious FIeld simulator for Non-equilibrium system).

Concurrent multiscale frameworks also exist and commonly come in a single framework. Perhaps the most well-known is the Quasi-Continuum Method (QC).[120] Their code is available publicly and is able to set up the material lattice, grains and the FE mesh, eliminate the ghost force and run the simulation without the need of any third party software. It is currently limited to 2D, crystals with BCC or FCC and the original code does not allow for finite temperature calculation although there has been work to extend it,[121-123] QC is not MD but can do Molecular Mechanics (MM) calculations i.e. energy minimization. There is also LibMultiScale[124] which is a parallel framework for coupled multiscale methods. This framework provides an API which makes it possible to program coupled simulations of pre-existing codes. MD codes such as Stamp from CEA and LAMMPS[105] have been coupled to a unique FEM code libMesh[125]. It currently is based on the Bridging Domain Method (BDM) of T. Belytschko and S. Xiao.[100] Which uses a Lagrange multiplier method or augmented Lagrangian method for enforcing the kinematic constraints in the overlapping subdomain where the total Hamiltonian is a linear combination of the molecular and continuum Hamiltonians which can handle the spurious wave reflection problem in the overlapping domain.[102,103] They have also developed an explicit algorithm and a multi-time step method for BDM. Most simulation codes used in literature are in-house private codes that are designed for specific applications or to prove a concept. Never the less these private codes are extremely important for the advancement of multiscale simulation techniques; one cannot advance

computational simulation methods without writing source code for the computers to execute. Without computer code brilliant ideas will flounder and die; an intimate knowledge of the physical mechanisms, equations, and algorithms are essential for correct simulations.

The program used in this work is a particle-based multiscale analysis program (PMAP) with the capacity of simulating molecular dynamics with numerous interfaces coupling higher-scale particle dynamics concurrently to the atomistic domain via the Generalized Particle Dynamics Method (GP). The model size may be further extended by coupling the high-scale particle domains to finite element meshes. These coupled scales are designed to provide the atomistic domain with realistic boundary conditions so that real world applications may be investigated. This illustrates PMAP as a complete framework for concurrent multiscale analysis, including MD, various particle scales and ultimately an FEM continuum. In this chapter the structure of PMAP will be discussed first as a brief overview, then in more detail about its three parts, initialization, equilibration, and loading. Explaining the structure and process flow of this multiscale framework is important for those who wish to develop their own multiscale simulation code; it is best to begin with the general idea of how the code works before delving into the details of particular subroutines and functions. These details are discussed in the Appendix for instructive purposes.

B. Functionality, constitution and flow charts of three basic processes of PMAP

Molecular Dynamic Simulation codes such as DL_POLY[106,107], LAMMPS[105], and GROMACS[108] have very similar structures; they use essentially the same types of algorithms for atom management, integration of motion, and ensemble controls like thermostats and barostats. The long history of molecular dynamic simulation provides a wealth of information to those who desire to learn the methods and implement them within their own codes.[126]

As is usual for most processes, initial conditions are declared as input and calculated followed by a minimization and/or equilibration procedure, lastly with the obtained equilibrated state particular non-equilibrium loading conditions may be applied,

such as applied strain or electric field. These three processes are shown in Fig. 48 and are contained within three separate subroutines in the code with one following the other.

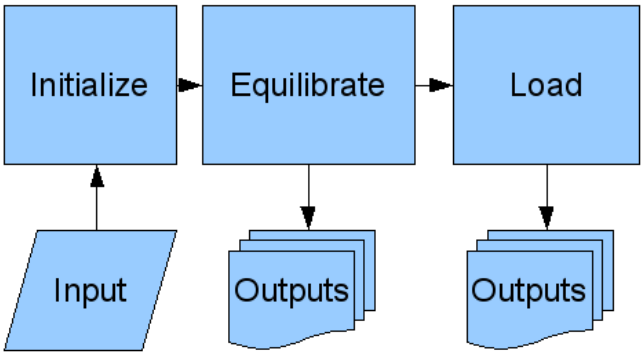


Figure 48. Main flow process of the PMAP.

The initialization and equilibration processes will be described in the next two subsections since these routines are unique to the GP simulation technique. The Load process can have two different process flow styles, one for GP models and another when coupling GP models to FEA meshes. These two styles will be discussed in the subsequent subsections.

1. Initialization Process

The initialization process is vital to any procedure; it is the one time overhead that prepares the tasks to be set into motion. The initialization process flow can be seen in Fig. 49. The first thing that PMAP does is to read the simulation input file. This file contains the ensemble controls for the entire simulation both equilibration and loading procedures. This includes the duration of the time-step used in the integration of motion, the duration of the equilibration process, the temperature if it is to be controlled, and the pressure if that's to be controlled as well. It also specifies configuration output frequencies and the output frequency of general ensemble statistics. It may also have various loading styles for the loading procedure, such as monotonic and cyclic strain rate, electric field, and thermal ramping. Domains may be specified to be fixed with relation to the simulation box, thermostatted locally, or just monitored for the domain's stress. Lastly, but arguably the most important aspect, the atomic species and their inter-atomic potentials must be declared. Any arbitrary atom may be defined given a symbol, number, mass, radius, and

charge. Atoms may interact via analytical potential functions such as the Morse, Lennard-Jones, Buckingham, and Johnson potentials, or from potential tables (in the format used in DL_POLY). The EAM potential can also be used in one of two formats, the *TABEAM* format from DL_POLY or the more commonly used *setfl* format. For the calculation of coulomb potentials the Damped Shifted Coulomb potential algorithm is used, see Appendix A.2. From all of this information the simulation input file is the main control file for the entire simulation process.

Once the simulation process parameters are read from the simulation input file the GP model file is read. This file is typically named "Model.MD" and contains the number of total and real atoms and particles, the simulation boxsize followed by a long list of the atom and particle positions, scales and ID information. This is used as the initial structure or configuration of the material model. This is the next most important thing for a successful simulation. This model file is generated separately before simulation by using lattice programs such as Materials Studio[127], GULP[128], etc. In this work an in-house code was used to generate the crystal structure as well as higher Generalized Particle scales and their coupling domains, more about the model generation procedure please refer to Appendix C. If the simulation is reviving from a previous run the "Revive.MD" file will be read and will overwrite the "Model.MD" configuration including velocities and accelerations of each atom and particle.

After reading the simulation input and the GP model configuration this information is shared among the other compute nodes when running PMAP in parallel across several processors. Most of the details of the parallel process, which utilizes the Message Passing Interface (MPI),[129] will be ignored in this chapter as it is not essential to understand the process flow. The potentials identified from the simulation input file are then used to create tabular arrays that are used during the dynamic processes. Using tabular interpolation for the potentials saves calculation time for the simulation.

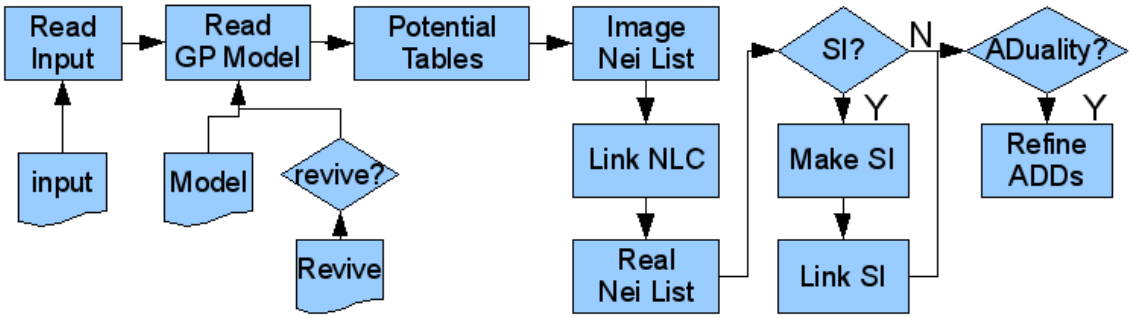


Figure 49. Initialization process flow, where Nei List stands for Verlet Neighbor List, SI for Surface Images, ADuality for Auto-Duality, and ADD for Auto-Duality Domain.

After the potential tables are created Verlet neighbor lists[126] are generated for each imaginary particle and those real particles in ADDs. The neighbors included in the lists are of adjacent scales not of the same scale. For example, an imaginary scale-1 particle, *i*, will have a list of scale-2 neighbors, *J*. The purpose of this kind of neighbor list is for the next process of creating the NLCs for imaginary atoms and particles. The NLCs link images to real particles hence the adjacent scale neighbor lists. Creating these Verlet neighbor lists is important since it considerably speeds the neighbor searching process for a given particle, even if the search process occurs only once for each imaginary particle. The NLC generation procedure collects a list of candidate real particles and sorts them according to distance; such that those real particles that are closest to the image will be included in the NLC and those that are farther away will not be included. The maximum number of real particles allowed in a NLC is twelve. This was decided based on the number of nearest neighbors for FCC and HCP crystal lattice structures. For an imaginary particle, *I*, the real atoms, *j*, in its NLC need not necessarily be the atoms that lump to form the imaginary particle, *I*. Rather, the NLC constituents, *j*, act as positional data points to ensure that the imaginary particle, *I*, follows the same deformation as the real atoms.

After the NLCs are generated all of the real atoms and particles are given new Verlet neighbor lists that contain atoms and particles of the same scale. So that an atom, *i*, will have atom neighbors, *j*. This is a more traditional use of Verlet neighbor lists for use when calculating interatomic forces for dynamic integration. But before the dynamic

integration of the equilibration process, there may still be more things to initialize. If surface images have been declared in the simulation input file then the domains identified to become surface images will be made into images and then linked to the same scale real particles of the model, see Section II.A.3 about the specifics of surface images. Lastly if the model has auto-duality domains (ADD) identified they are checked to make sure that the mass of the imaginary atoms is the same as the real particles in each ADD. If the masses are not consistent the boundary of the ADD is modified gradually until the masses are consistent.

At this point the initialization process is complete and the simulation will continue to the equilibration process.

2. Equilibration Process

The equilibration procedure is the first dynamics routine, the second being part of the loading procedure. These dynamic routines and those that support them are the heart of the simulation. The concept of the equilibration procedure is to allow the forces on each atom and particle to come to zero by relaxing the material structure. The way to do this is to calculate the total force on each particle based on the current configuration. From these forces the acceleration, velocity and finally displacement can be found at a specific later time, defined as the integration timestep. These displacements are then applied to the particle positions to create a new configuration. From this new configuration the process begins again for the next timestep, $t+1$. This calculation cycle is repeated for a particular amount of time by which the structure is determined to have come to equilibrium.

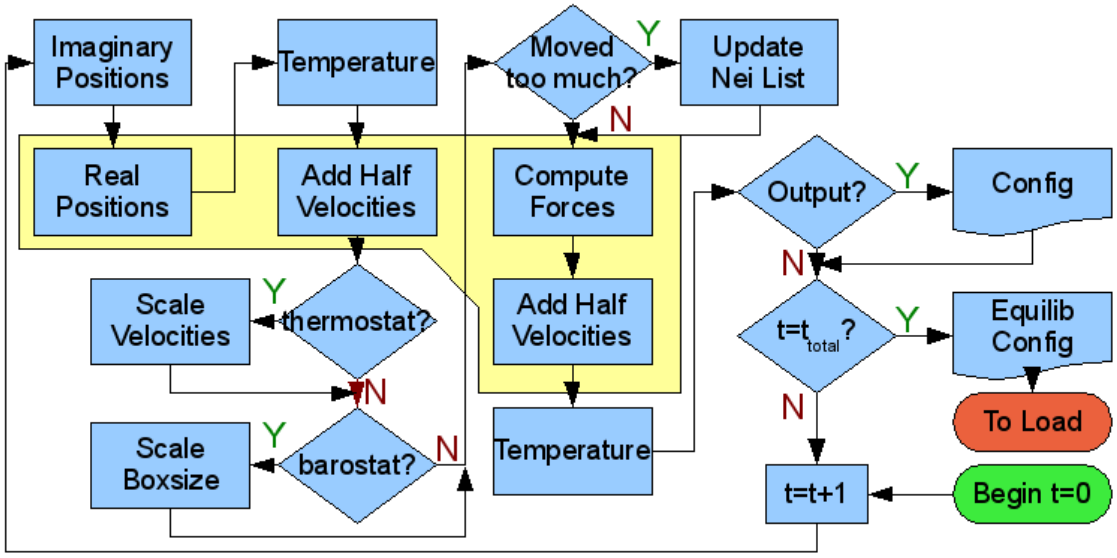


Figure 50. Equilibration process flow. The lightly shaded area is the Velocity Verlet algorithm forming the core of the dynamic algorithm.

The specific subprocesses involved in the equilibration process are shown in Fig. 50. At the bottom right of the figure is where the process begins at time step *t=0*. The main loop is entered and the first subprocess is the determination of the imaginary particle positions, at the top left. As discussed before imaginary particle positions are determined by the position average of the constituents of their NLCs. This requires only the real particle positions. Once the imaginary particle positions are determined the real particle positions are calculated based on the previous timestep's velocities and accelerations. When *t=0* these values are zero. It is possible to prescribe an initial temperature to the model by applying a Boltzmann distribution to the initial velocities, but PMAP does not support this feature at this time, however it does provide initial positional perturbation to make it more realistic when raising the system temperature. The temperature of the system is calculated at this time in preparation for a possible thermostat procedure. Half of the new velocity contribution is added to the existing velocities such that the velocity of atom *i* is:

$$v_i^{t+0.5} = v_i^t + \frac{1}{2} a^t \Delta t \qquad (31)$$

If there is a thermostat on the system, these new velocities will be scaled via a modification to the old velocity term v_i^t. Such that the equation becomes:

$$v_i^{t+0.5} = v_i^t \sqrt{\frac{T_{req}}{T^t}} + \frac{1}{2} a^t \Delta t \tag{32}$$

Where T^t is the temperature at time t and T_{req} is the requested temperature.

After the half velocities are calculated and the system was checked for a thermostat it is then checked for a barostat. If there is a barostat then the simulation boxsize will be scaled to reduce the system pressure. This works because all particle positions are normalized to the simulation boxsize, thus they are essentially a function of the boxsize. The details of this procedure can be seen in Appendix A.5.

Another condition that is checked before the calculation of the interatomic and interparticle forces is whether the particles have moved too much. This condition is to determine whether to regenerate the Verlet neighbor lists. If there is large deformation of local material then the neighbor lists should be regenerated to be sure that the correct neighbors are accounted for. This check helps to speed the calculation by reducing the number of times that the neighbor lists are regenerated.

After these checks are completed the interatomic forces are calculated for all particles based on their inverse mapped atomic distances. These forces provide new accelerations to be used in the second half of the velocity Verlet algorithm.

$$v_i^{t+1} = v_i^{t+0.5} + \frac{1}{2} a^{t+0.5} \Delta t \tag{33}$$

Thus the complete integration for the timestep is realized. Now based on these new velocities the temperature of the system is recalculated and ensemble statistics and configuration files may be written to later data analysis.

When the entire equilibration process is complete, i.e. $t=t_{total}$ the equilibration loop exits and a specific configuration is written of the equilibrated structure.

3. Loading Process

The loading procedure is basically the same as the equilibration procedure; the main difference is the addition of subprocesses to modify the simulation behavior. Loading processes such as monotonic and cyclic strain rate, electric field, and thermal

ramping can be used here. For loads like strain loads are implemented in a step-wise manner with a relaxation time in between loading steps. The load increment and relaxation time are simulation parameters that are specified in the input file.

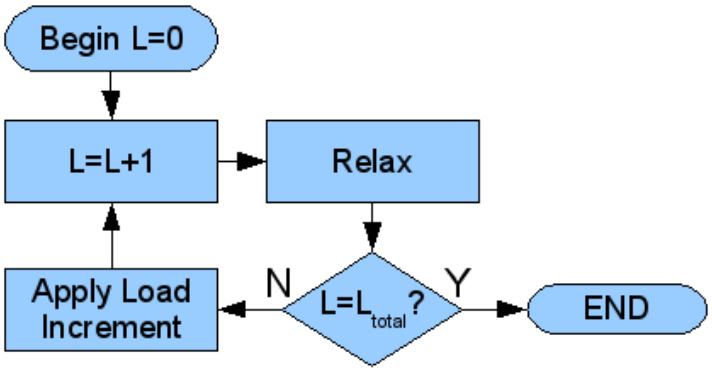


Figure 51. General flow of the GP load process.

From Fig. 51 the load step L begins at zero, so there is a relaxation without any load for the given relaxation time t_{relax}. The relax process follows the same procedures as the equilibration process. However, alternate boundary conditions (e.g. fixed domains) may be applied in the load process. When the relaxation is done it is checked to see if the total load step, L_{total}, has been reached, if not another load increment is applied to the system. If the total load steps have been reached then the simulation is complete.

The main difference in the process flow, when coupling with a finite element mesh, is within the Load procedure. Right before the Load procedure is the initialization routine for the FEA mesh ("FEA_init" of Fig. 52). Another difference occurs periodically within the Loading procedure when the FEA mesh is calculated to provide updated positions to the WG boundary layer to the embedded GP domain ("FEA_calc" of Fig. 52). Figure 52 shows the modified GP load process with the FEA procedures shown as white boxes. The light colored region is the GP-FEA iteration loop wherein the GP-FEA interface is tested for convergence, if it has not converged then the WG-GP subsystem continues to relax and the WF-FEA subsystem is recalculated. When the interface has converged the next load increment is made, unless it was the last load increment in which case the simulation completes.

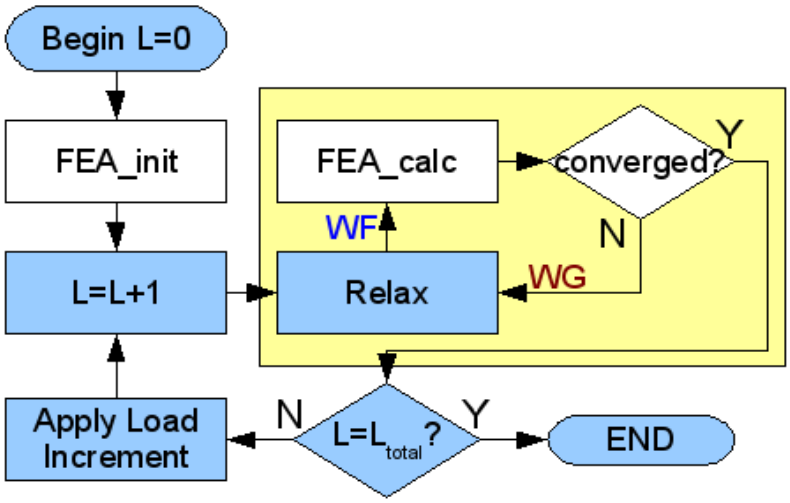


Figure 52. General flow of the GP-FEA load process.

There is another subtlety when coupling with an FEA mesh which involves the WF and WG domains. At the end of the FEA calculation it sends new interpolated WG particle positions to the relaxation process which uses them as a fully fixed boundary layer. While the WG-GP subsystem relaxes with this WG boundary it samples the position average of the WF particles every ten timesteps since the last FEA calculation. When the relaxation process is done the WF particle positions are averaged and applied as WF node displacements for the WF-FEA subsystem boundary condition.

C. Constructions and Executions

There are about three input files that PMAP reads, the model including the atomic and particle coordinate and ID data, if coupling to a FEA mesh the FE mesh data and material properties are read from an FEA input file, and most importantly the global PMAP input file which contains all of the simulation control parameters. The first two input files have been discussed in Section II.B. and Appendix C.6.; the development of the GP model is performed with a separate crystal tessellation code, the usage of which is described in detail in Appendix C.1. The main input file's directives are listed and described in full detail in Appendix D. along with instructions to compile and run PMAP. Also included in Appendix D are the corresponding input files for the examples described in Chapter II. The purpose of this is for the reader to become familiar with the simulation

process and the data files that are required to carry out these kinds of concurrent multiscale simulations. By explaining this we hope that more people will become involved in developing not only new multiscale methods but also to be able to write code to realize their creations.

1. Post-processing

Most of the utility programs used with PMAP are for post processing purposes. These include conversion programs to convert the output files of PMAP to other formats usable by other mainstream programs such as VMD[130], AtomEye[131], and gnuplot[132] for visualization. Other programs are for data analysis, model manipulation and detailed debugging of simulations. A full list of these programs is provided in Table VIII of Appendix E. General post processing program explanations are described in Appendix E and deeper analysis programs are described in Appendix F, such detailed molecular structure analysis which can identify atomic defects using several different techniques including, Coordination Number, Common Neighbor Analysis[133], Near Neighbor Grouping[134], Coordination Vector, and void identification.

With these utility programs the PMAP output data files can easily be manipulated, analyzed, and phenomena debugged and understood. Many of these programs are not restricted for use with PMAP but are programed to have general functionality for use in many other applications.

D. Summary

The PMAP procedures have been described in detail and the capabilities illustrated. The general structure of PMAP is very similar to most MD simulation programs for the GP computational contribution. For FEA coupling a simple FEM solver is used periodically during the iterations. Both of these components are easy to be recognized individually, but the strength of PMAP comes from their combination.

Common topics for code improvement are better memory consumption, more efficient computational techniques, parallel algorithms, and more intuitive model development programs. However the most important improvements to be made are theoretical in nature and will be discussed in the recommendations in Chapter VIII.

CONCLUSIONS AND RECOMMENDATIONS

Although there has been great progress in the use of MD to investigate crack propagation and the origins of failure there are two main shortcomings:

First, most work concentrates on crack propagation rather than crack nucleation. These defects are both theoretically and practically important since they are the key properties for understanding the underlying mechanisms of failure.

Secondly, much work has imposed special treatments to ensure that the crack propagates along a desired path. These treatments are convenient when using the cohesive zone model (CZM) to investigate and model the crack propagation behavior; however their effectiveness must be further validated because the crack propagation essentially depends on how it was nucleated.

From the modeling perspective, the methods used today have a difficult time linking the behaviors of materials on the nanoscale to the large scale bulk properties.[19] This is mainly due to the complicated question: what is the minimum size of material that can be considered a continuum? The answer to this question is highly material dependent. Another reason for ambiguous answers to this question is due to the strong surface effects at the nanoscale. These effects become more significant when the area/volume ratio increases. This indicates that the surface energy is very important to consider on the nanoscale and must be included if an accurate material model is to be used at these small scales. In the elastic range continuum models are incredibly reliable and can be applied to discrete materials having a size of a few nanometers.[47] However these continuum models break down when plasticity is involved at these small scales, such that the minimum required model size for plasticity remains unsolved.

This gap in capability prevents atomistic-based models from predicting large scale material behavior thereby failing to link the two classes of multiscale analysis together. If micron sized models can be developed it would open a door to solving important problems in engineering based on a fundamental atomistic foundation.

The GP method and the related GP-FEA method have been introduced as candidates for this gap in capability. It has been seen to have a natural interface between scale domains where physical variables such as displacement and force may smoothly pass from one scale to another. Atomic phenomena such as dislocations may be passed into higher scale domains via the scale-duality concept[50] by decomposing particles into their constituent atoms. In addition, it has been mathematically proven[48,50] that all calculations in the particle domain can be conducted at the atomistic domain scale using the same potential, parameters and numerical algorithm as is used for the model's atomistic scale. Thus the GP method is essentially an extension of MD and can be easily delivered to applications by modifying existing MD codes.

A. Conclusions

There are two basic issues for extending applications of concurrent multiscale simulations. They include how to quantify the accuracy of atomistically-based multiscale simulation and how to enlarge the model size to the minimum necessary to guarantee the accuracy. These solutions have been discussed with the GP and GP-FEA methods. The GP-FEA method, is a new multiscale method which can make the model size as large as needed in the microsystem. Apart from the conventional verification method with the full atomic solution (e.g. MD), a classic elastic stress solution of a two-dimensional specimen with a central hole under tensile load is used to compare its displacement distribution. This provides an effective tool for accuracy verification of the GP and GP-FEA multiscale methods by comparison of their simulation data with the analytical solution. The result of the comparison is encouraging. Main conclusions from the work include:

- The GP-FEA model embeds an inner multiscale particle system within a surrounding continuum FE domain. It moves the atom-FEA interface of the DC method far away to the particle-FEA interface. This greatly reduces DOF of the system while not disturbing any important phenomena in the focused atomistic domains, thus the artificial forces and deformation caused by, say, ghost forces can be avoided.
- It should be noted that the elastic analytical solution is obtained under the condition that the width of the specimen be larger than four times the hole

diameter to keep the error of the solution, e.g., $(\sigma_\theta)_{max}$ from exceeding 6 per cent. Our design satisfies this requirement. However, model size effects are problem-dependent. It may relate to material property, environmental conditions, the variables involved, etc. Thus, it is difficult to get a general answer analytically. In many cases one should carry on numerical simulations for models with different size to find the minimum necessary for accuracy.

- The satisfactory agreement between the displacement data obtained by the proposed GP-FEA methods with the classical analytical solution establishes a foundation to use these multiscale methods to investigate model size effects.

Encouraged by the successful comparison of the displacement field predicted by the GP-FEA method with the continuum solution for a 2D plate with a central cylinder hole, this newly proposed multiscale method has been further developed and applied to the crack-tip analysis. Results show excellent agreement of the simulation results with the LEFM two-term solutions by Rice and others.

This successful comparison with the continuum solution and the powerful capability of the GP-FEA multiscale method in developing a large micron-scale model are promising. It allows for the investigation of model dimension effects on the accuracy of the atomistically-based multiscale method realistic and attractive. The significance of this investigation should be further emphasized even though the model size choice is a common problem that appears frequently during model design. It can be further addressed from the following four aspects. Firstly, accuracy verification for low scales is important to find the deformation mechanisms. Secondly, if the model size is small the BC disturbances may likely affect the local fields of forces and displacements which are near the atomistic regions of interest. In turn, it will change the behavior of highly important domains such as interfaces, crack-tips and flaws. Thirdly, some mathematical solutions for the continuum require the medium to be sufficiently large to make the LEFM crack-tip solution realistic for a tiny crack inside of a bulk material. In this case, model size must not be small for a reasonable result. The fourth aspect is that for microsystems and nanotechnology, the model size should be equal/larger than micrometers or at least being sub-micrometers so the problem of micro- or nano-

sensors/activators can be more accurately simulated. Thus, investigating the model size effects and choosing a minimum model size necessary for the accuracy requirement is essential.

With the proposed GP-FEA method, the model size effects on the crack-tip displacement fields of a Mode-I edge crack embedded in a single crystal of BCC iron are extensively investigated. All models were subjected to the remote BC displacement along the Y-direction $[\bar{1}10]$ with 1% strain. The accuracy is verified by the LEFM two-term solution from the results the following observations and conclusions can be made.

- It is seen that the smaller the model size the larger the error produced in the simulation-obtained u_y relation. Specifically, for the case of L_y=120 nm, the error can reach about 50%. Indicating the small model has high rigidity to produce small deformation. Our work shows that using stress intensity factor K to investigate the model size effects is not sufficient since that value is obtained by a model with infinite size. Changing the model size and comparing the behavior with the LEFM solution will show the size effect quantitatively. This result serves as a serious warning: since many existing simulation models are below this size, the accuracy of these models may be questionable and need to be carefully verified.
- When the model size increases from 120 nm to 500 nm, the accuracy quickly increases. However, further increases of the model size from 500 nm to 5000 nm results have basically the same accuracy as the case of 500 nm. This result is significant since it lays a foundation for introduction of a new concept of critical model size, L_{CR}. In fact, the comparison tell us that if the model size is less than L_{CR}, say 500 nm, the results obtained from atomistically-based multiscale simulations will have unrealistic crack-tip behavior, including a large percent of inaccuracy in comparison with the LEFM result. On the other hand, the case for designing the model size larger than L_{CR} should also be avoided since it may not greatly improve the accuracy with the penalty of increasing a large of DOF.

The results of this study show that the size of the model affects the material behavior by influencing the atomistic phenomena at crucial locations as a crack tip. Two effects were found that affect the phase transformation at the crack tip. First, it was seen that smaller models do not have as many atoms in the FCC phase as larger models do for the same loading strain. Second, the smaller models were delayed in their nucleation of the FCC phase.

These effects show that the chosen model size of the simulation can seriously affect atomistic phenomena observed at crack tips. If the researcher is looking to derive critical information from atomistic-based simulations the model size must be carefully chosen.

B. Recommendations for Future Work

Most of the recommendations fall under the category of improving the GP and GP-FEA methods within the PMAP. However, there is a clear need to continue to develop a guideline for the least-required model size, L_{CR}, to improve the accuracy in bridging atomistic and continuum scales. Fortunately, the newly proposed GP-FEA, which can develop large model sizes, has proven so far to be an effective tool to face this challenge.

Practice has taught us that the extension of multiscale analysis to more applications requires essential development of computer code. While codes for model generation have also been developed, models still require some manual work to design transition domains such as WF, WG, W_n and W_{n+1}. This shortcoming will be overcome such that these domains can be formed automatically after the boundary line and its width and depth are given, etc.

It has also been shown that the auto-duality feature of the GP method can be used to help mitigate the unrealistic wave propagation dynamics caused by higher scale GP representations. This illustrates the need for atoms in locations where the deformation gradient is large and that higher scale particles may be used in areas of small deformations. There are two dynamic problems that occur within GP scales higher than the atomistic scale.

First, the wave speed in higher scales is proportional to the scale ratio. This is due to the farther reaching influence of higher scales for the same reason why higher scales naturally have a proportionally greater surface effect. In order to address this problem a more accurate way to calculate the strain gradient must be made.

Second, the temperature of high scales is not well defined; using only the inverse mapping method to calculate temperature does not make sense due to the multiscale nature of velocity distributions that compose thermal energy. Since a particle's velocity is the average of the atoms' that it is composed of, there is no guarantee that the kinetic/thermal energy of those atoms is consistent with the particle that represents them.

From this perspective the temperature of a high scale domain could be represented by a sum of two terms, the particles' kinetic temperature plus the internal kinetic energy of the implicit atoms the particles represent. The larger the particle scale the greater the internal thermal contribution of the implicit atoms due to the loss of DOF to the domain temperature. The heat from the lost DOF due to lumping may be calculated in the same way as is used in the MPM multiscale method.[104]

Aside from the obvious improvements to be made to the PMAP code such as better memory consumption, more efficient computational techniques, parallel algorithms, and more intuitive model development programs, the main obstacles are theoretical. For example, how to maintain stability and the general loss of vibrational energy during auto-duality transitions in rapid succession, as described in Chapter VI for the impact application, this is one topic at the front of GP theory development. Another is degree of freedom related; even though the GP method and the GP-FEA methods are able to reduce the system degree of freedom significantly, in some cases it is still not enough. At this time the FEA implementation in PMAP is only two-dimensional, thus the GP embedded domain must be relatively thin to have a meaningful coupling. Three-dimensional FEA is relatively easy to implement, however the cost of a 3D model is much larger than a quasi-2D model for both FEA and GP. When modeling a polycrystalline structure it is almost required to have the atomistic domain all along each of the grain boundaries (GB), the only way to save DOF is to place higher scale domains inside of the bulk of the grains. However, due to the nature of polycrystals the DOF saved is seriously limited by the GB interface to grain volume ratio and to study the plasticity

effects such as dislocation storage and twinning within grains requires atomic resolution to correctly capture the dislocation density.

This brief dynamical study of the GP method for use in dynamic applications is instructive. It clearly illustrates the needs still wanting in the GP method and suggests certain possible solutions to these tough problems. This research work is continuous and further developments to improve the GP and GP-FEA methods as well as efficiency improvements to PMAP will be achieved. However the current capabilities of PMAP have been seen and show promise for a wide range of real world engineering applications. Current work in this field will help to guide the development of the GP method into a more advanced future incarnation.

REFERENCES

1. F. Shackelford, *Introduction to Materials Science for Engineers, 6th ed.*, Prentice Hall., Upper Saddle River, New Jersey, 2004.

2. C. R. A. Catlow, "Point Defect and Electronic Properties of Uranium Dioxide," *Proc. R. Soc. Lond. A*, **353** [1675] 533-561 (1977).

3. V. Amakov, D. Wolf, S. R. Phillpot, A. K. Mukherjee, and H. Gleiter, "Dislocation Processes in the Deformation of Nanocrystalline Aluminum by Molecular Dynamics simulation," *Nat. Mater.*, **1** [1] 45-49 (2002).

4. R. E. Williford, W. J. Weber, R. Devanathan, and A. N. Cormack, "Native Vacancy Migrations in Zircon," *J. Nucl. Mater.*, **273** [2] 164-170 (1999).

5. N. F. Mott and M. J. Littleton, "Conduction in Polar Crystals, I. Electrolytic Conduction in Solid Salts," *Trans. Farad. Soc.*, **34** [1] 485-499 (1938).

6. J. D. Gale and A. L. Rohl, "The General Utility Lattice Program (GULP)," *Mol. Simul.*, **29** [5] 291-341 (2003).

7. D. S. Balint, V. S. Deshpande, A. Needleman, and E. Van der Giessen, "Discrete Dislocation Plasticity Analysis of the Grain Size Dependence of the Flow Strength of Polycrystals," *Int. J. Plasticity*, **24** [12] 2149–2172 (2008).

8. H. D. Espinosa, M. Panico, S. Berbenni, and K. W. Schwarz, "Discrete Dislocation Dynamics Simulations to Interpret Plasticity Size and Surface Effects in Freestanding FCC Thin Films," *Int. J. Plasticity*, **22** [11] 2091–2117 (2006).

9. D. M. Kochmann and K. C. Le, "Dislocation Pile-ups in Bicrystals within Continuum Dislocation Theory," *Int. J. Plasticity*, **24** [12] 2125–2147 (2008).

10. M. Kaluza and K. C. Le, "On Torsion of a Single Crystal Rod," *Int. J. Plasticity*, **27** [3] 460–469 (2011).

11. H. Liang and F. P. E. Dunne, "GND Accumulation in Bi-crystal Deformation: Crystal Plasticity Analysis and Comparison with Experiments," *Int. J. Mech. Sci.*, **51** [4] 326-333 (2009).

12. M. F. Horstemeyer, D. Farkas, S. Kim, T. Tang, and G. Potirniche, "Nanostructurally Small Cracks (NSC): A Review on Atomistic Modeling of Fatigue," *Int. J. Fatigue*, **32** [9] 1473-1502 (2010).

13. T. L. Anderson, *Fracture Mechanics Fundamentals and Applications, 3rd ed.*, CRC Taylor & Francis, London, 2005.

14. Y. Wei and J. W. Hutchinson, "Toughness of Ni/Al_2O_3 Interfaces as Dependent on Micron-scale Plasticity and Atomistic-scale Separation," *Philos. Mag*., **88** [30-32], 3841-3859 (2008).

15. H. B. Fan and M. F. Yuen, "A Multi-scale Approach for Investigation of Interfacial Delamination in Electronic Packages," *Microelectron. Reliab*., **50** [7] 893-899 (2010).

16. W. K. Liu, E. G. Karpov, S. Zhang, and H. S. Park, "An Introduction to Computational Nanomechanics and Materials," *Comput. Method. Appl. Mech. Eng.* **193** [17-20] 1529-1578 (2004).

17. W. K. Liu, E. G. Karpov, and H. S. Park, *Nano Mechanics and Materials: Theory, Multiscale Methods and Applications*, John Wiley & Sons, Hoboken, NJ, 2006.

18. J. Fan, L. He, and R. Stewart, "Concurrent and Hierarchical Multiscale Analysis for Layer-Thickness Effects of Nanoscale Coatings on Interfacial Stress and Fracture Behavior," *J. Eng. Mater. Tech.*, **134** [3] 114-124 (2012).

19. J. Fan, *Multiscale Analysis of Deformation and Failure of Materials*, John Wiley & Sons, Ltd., Chichester, United Kingdom, 2011.

20. J. A. Elliott, "Novel Approaches to Multiscale Modelling in Materials Science," *Int. Mater. Rev*., **56** [4] 207-225 (2011).

21. A. Lyubartsev, A. Tu, and A. Laaksonen, "Hierarchical Multiscale Modelling Scheme from First Principles to Mesoscale," *J. Comput. Theor. Nanos.*, **6** [5] 951-959 (2009).

22. P. Ortoleva, A. Singharoy, and S. Pankavich, "Hierarchical Multiscale Modeling of Macromolecules and their Assemblies," *Soft Matter*, **9** 4319-4335 (2013).

23. L. Monticelli, S. Kandasamy, X. Periole, R. Larson, D. P. Tieleman, and S. Marrink, "The MARTINI Coarse-Grained Force Field: Extension to Protein," *J. Chem. Theory Comput.*, **4** [5] 819-834 (2008).

24. J. R. Rice and R. Thomson, "Ductile Versus Brittle Behavior of Crystals," *Philos. Mag*., **29** [1] 73-97 (1974).

25. J. R. Rice and G. E. Beeltz, "The Activation Energy for Dislocation Nucleation at a Crack," *J. Mech. Phys. Solids*, **42** [2] 333-360 (1994).

26. G. E. Beeltz and J. R. Rice, "Dislocation Nucleation at Metal-ceramic Interfaces," *Acta Metal. Mater.*, **40** [supplement] S321-S331 (1992).

27. R. E. Peierls, "The Size of Dislocation," *Proc. Phys. Soc.*, **52** [1] 34-37 (1940).

28. F. Cleri, S. Yip, D. Wolf, and S. R. Phillpot, "Atomic-scale Mechanism of Crack-tip Plasticity: Dislocation Nucleation and Crack-tip Shielding," *Phys. Rev. Lett.*, **79** [7] 1309-1312 (1997).

29. F. Cleri, D. Wolf, S. Yip, and S. R. Philpot, "Atomistic Simulation of Dislocation of Nucleation and Motion from a Crack Tip," *Acta Mater.*, **45** [12] 4993-5003 (1997).

30. F. Cleri, S. R. Phillpot, D. Wolf, and S. Yip, "Atomistic Simulations of Material Fracture and the Link Between Atomic and Continuum Length Scales," *J. Am. Ceram. Soc.*, **81** [3] 501-516 (1998).

31. L. E. Shilkrot, R. E. Miller, and W. A. Curtin, "Multiscale Plasticity Modeling: Coupled Atomistics and Discrete Dislocation Mechanics," *J. Mech. Phys. Solids*, **52** [4] 755-787 (2004).

32. E. Saether, V. Yamakov, and E. H. Glaessgen, "An Embedded Statistical Method for Coupling Molecular Dynamics and Finite Element Analysis," *Int. J. Numer. Meth. Eng.*, **78** [11] 1292-1319 (2009).

33. E. B. Tadmor, R. Miller, R. Phillips, and M. Ortiz, "Nanoindentation and Incipient Plasticity," *J. Mater. Res.*, **14** [6] 2233-2250, (1999).

34. R. E. Miller and E. B. Tadmor, "The Quasicontinuum Method: Overview, Applications and Current Directions," *J. Comput. Aided Mater. Des.*, **9** [3] 203-239 (2002).

35. L. E. Shilkrot, R. E. Miller, and W. A. Curtin, "Multiscale Plasticity Modeling: Coupled Atomistics and Discrete Dislocation Mechanics," *J. Mech. Phys. Solids*, **52** [4] 755-787 (2004).

36. W. A. Curtin and R. E. Miller, "Atomistic/continuum Coupling in Computational Materials Science," *Modelling Simul. Mater. Sci. Eng.*, **11** [3] R33- R66 (2003).

37. R. E. Miller and D. Rodney, "On the Nonlocal Nature of Dislocation Nucleation During Nanoindentation," *J. Mech. Phys. Solids*, **56** [3] 1203-1223 (2008).

38. Z. Q. Wang and I. J. Beyerlein, "An Atomistically-informed Dislocation Dynamics Model for the Plastic Anisotropy and Tension-compression Asymmetry of BCC metals," *Inter. J. Plasticity*, **27** [10] 1471-1484 (2011).

39. D. Warner, W. Curtin, and S. QU, "Rate Dependence of Crack-tip Processes Predicts Twinning Trends in F.C.C. Metals," *Nature Mater.*, **6** [11] 876-881 (2007).

40. H. Fan and M. Yuen, "A Multi-scale Approach for Investigation of Interfacial Delamination in Electronic Packages," *Microelectron. Reliab.*, **50** [7] 893-899 (2010).

41. J. L. Tsai, S. H. Tzeng, and Y. J. Tzou, "Characterizing the Fracture Parameters of a Graphene Sheet Using Atomistic Simulation and Continuum Mechanics," *Int. J. Solids Struct.*, **47** [3- 4] 503-509 (2010).

42. X. W. Zhou, N. R. Moody, R. E. Jones, J. A. Zimmerman, and E. D. Reedy, "Molecular-Dynamics-Based Cohesive Zone Law for Brittle Interfacial Fracture under Mixed Loading Conditions: Effects of Elastic Constant Mismatch," *Acta Mater.*, **57** [16] 4671-4686 (2009).

43. J. T. Lloyd, J. A. Zimmerman, R. E. Jones, X. W. Zhou, and D. L. McDowell, "Finite Element Analysis of an Atomistically Derived Cohesive Model for Brittle Fracture," *Modelling Simul. Mater. Sci. Eng.*, **19** [6] (065007) 1-18 (2011).

44. K. J. V. Vilet, J. Li, T. Zhu, S. Yip, and S. Suresh, "Quantifying the Early Stages of Plasticity through Nanoscale Experiments and Simulations," *Phys. Rev. B,* **67** [10] (104105) (2003).

45. N. Hansen, "Hall-Petch Relation and Grain Boundary Strengthening," *Scr. Mater.*, **51** [8] 801-806 (2004).

46. H. L. Duan, J. Wang, Z. P. Huang, and B. L. Karihaloo, "Size-dependent Effective Elastic Constants of Solids Containing Nano-inhomogeneities with Interface Stress," *J. Mech. Phys. Solids*, **53** [7] 1574-1596 (2005).

47. B. Yakobson and R. E. Smalley, "Fullerene Nanotubes: C-1000000 and Beyond," *Am. Sci.*, **85** [4] 324-337 (1997).

48. J. Fan, "Multiscale Analysis Across Atoms/continuum by a Generalized Particle Dynamics Method," *Multiscale Model. Simul*., **8** [1] 228-253 (2009).

49. A. Pedone, G. Malavasi, M. C. Menziani, U. Segre, A. N. Cormack, "Molecular Dynamics Studies of Stress-Strain Behavior of Silica Glass under a Tensile Load," *Chem. Mater.*, **20** [13] 4356-4366 (2008).

50. J. Fan, R. J. Stewart, and X. Zeng, "A Multiscale Method for Dislocation Nucleation and Seamlessly Passing Scale Boundaries," *Int. J. Plasticity*, **27** [12] 2103-2124 (2011).

51. T. R. Chandrupatla and A. D. Belegundu, *Introduction to Finite Elements in Engineering, 3rd ed.*, Prentice Hall, Upper Saddle River, NJ, 2002.

52. C. Hua, "An Inverse Transformation for Quadrilateral Isoparametric Elements: Analysis and Application," *Finite Elem. Anal. Des.*, **7** [2] 159-166 (1990).

53. R. E. Miller and E. B. Tadmor, "A Unified Framework and Performance Benchmark of Fourteen Multiscale Atomistic/continuum Coupling Methods," *Modelling Simul. Mater. Sci. Eng.*, **17** [5] 053001-51 (2009).

54. D. L. McDowell and G. B. Olson, "Concurrent Design of Hierarchical Materials and Structures," *Sci. Model. Simul.*, **15** [1-3] 207-240 (2008).

55. E. B. Tadmor, M. Ortiz, and R. Phillips, "Quasicontinuum Analysis of Defects in Solids," *Phil. Mag. A*, **73** [6] 1529-1563 (1996).

56. L. E. Shilkrot, R. Miller, and W. A. Curtin, "Coupled Atomistic and Discrete Dislocation Plasticity," *Phys. Rev. Lett.*, **89** [2] 025501-4 (2002).

57. L. E. Shilkrot, R. Miller, and W. A. Curtin, "A Coupled Atomistic/continuum Model of Defects in Solids," *J. Mech. Phys. Solids*, **50** [10] 2085-2106 (2002).

58. L. E. Shilkrot, R. Miller, and W. A. Curtin, "Multiscale Plasticity Modeling: Coupled Atomistics and Discrete Dislocation Mechanics," *J. Mech. Phys. Solids*, **52** [4] 755-787 (2004).

59. R. E. Rudd. and J. Q. Broughton, "Concurrent Coupling of Length Scales in Solid State Systems," *Phys. Status Solidi B*, **217** [1] 217-251 (2000).

60. V. B. Shenoy, R. Miller, E.B. Tadmor, D. Rodney, R. Phillips, and M. Ortiz, "An Adaptive Methodology for Atomic Scale Mechanics: The Quasicontinuum Method," *J. Mech. Phys. Solids*, **47** [3] 611-642 (1999).

61. S. Kohlhoff, P. Gumbsch, and H. F. Fischmeiser, "Crack Propagation in BCC Crystals Studied with a Combined Finite-element and Atomistic Model," *Phil. Mag. A*, **64** 851-878 (1991).

62. W. K. Liu, E. G. Karpov, S. Zhang, and H. S. Park, "An Introduction to Computational Nanomechanics and Materials," *Comput. Methods in Appl. Mech. Eng.*, **193** [17-20] 1529-1578 (2004).

63. S. Timoshenko, *History of Strength of Materials*, McGraw Hill, New York City, 1953.

64. S. Timoshenko and J. N. Goodier, *Theory of Elasticity*, McGraw Hill, New York City, 1951.

65. M. I. Mendelev, S. Han, D.J. Srolovitz, G. J. Ackland, D. Y. Sun, and M. Asta, "Development of New Interatomic Potentials Appropriate for Crystalline and Liquid Iron," *Phil. Mag.*, **83** [35] 3977-3994 (2003).

66. P. S. Leevers and J. C. Radon, "Inherent Stress Biaxiality in Various Fracture Specimen Geometries," *Int. J. Fracture*, **19** [4] 311-324 (1983).

67. P. Gumbsch, "An Atomistic Study of Brittle Fracture: Toward Explicit Failure Criteria from Atomistic Modeling," *J. Mater. Res.*, **10** [11] 2897-2907 (1995).

68. V. Yamakov, E. Saether, D. R. Phillips, and E. H. Glaessgen, "Molecular-Dynamics Simulation-Based Cohesive Zone Representation of Intergranular Fracture Processes in Aluminum," *J. Mech. Phys. Solids*, **54** [9] 1899-28 (2006).

69. V. Yamakov, E. Saether, and E. H. Glaessgen, "Multiscale Modeling of Intergranular Fracture in Aluminum: Constitutive Relation for Interface Debonding," *J. Mater. Sci.*, **43** [23-24] 7488-94 (2008).

70. D. E. Spearot, K. I. Jacob, and D. L. McDowell. "Non-local Separation Constitutive Laws for Interfaces and their Relation to Nanoscale Simulations," *Mech. Mater.*, **36** [9] 825–847 (2004).

71. S. M. Foiles, M. I. Baskes, and M. S. Daw, "Embedded-Atom-Method Functions for the FCC Metals Cu, Ag, Au, Ni, Pd, Pt, and their Alloys," *Phys. Rev. B*, **33** [12] 7983-7991 (1986).

72. S. T. Choi and K. S. Kim, "Nanoscale Planar Field Projections of Atomic Decohesion and Slip in Crystalline Solids. Part I. A Crack-tip Cohesive Zone," *Phil. Mag.*, **87** [12] 1889-1919 (2007).

73. H. Krull and H. Yuan, "Suggestions to the Cohesive Traction-Separation Law from Atomistic Simulations," *Eng. Fract. Mech.*, **78** [3] 525-33 (2011).

74. C. R. Dandekar and Y. C. Shin, "Molecular Dynamics Based Cohesive Zone Law for Describing Al–SiC Interface Mechanics," *Composites: Part A*, **42**, [4] 355–363 (2011).

75. X. W. Zhou, J. A. Zimmerman, E. D. Reedy, and N. R. Moody, "Molecular Dynamics Simulation Based Cohesive Surface Representation of Mixed Mode Fracture," *Mech. Mater.*, **40** [10] 832-45 (2008).

76. H. B. Fan, C. K. Y. Wong, and M. M. F. Yuen, "Multi-scale Interfacial Delamination Model of CuSAM-Epoxy Systems," In: Proc. International Conference on Electronic packaging technology & High Density Packaging, Shanghai, 2008.

77. C. R. Dandekar and Y. C. Shin, "Effect of Porosity on the Interface Behavior of an Al2O3-aluminum Composite: A Molecular Dynamics Study," *Compos. Sci. Technol.*, **71** [3] 350-356 (2011).

78. V. Yamakov, D. Warner, R. Zamora, E. Saether, W. Curtin, and E. Glaessgen, "Investigation of Crack Tip Dislocation Emission in Aluminum using Multiscale Molecular Dynamics Simulation and Continuum Modeling," *J. Mech. Phys. Solids*, **65** [1] 35-53 (2014).

79. Dassault Systèmes, ABAQUS Documentation. Providence, RI, 2011.

80. H. M. Westergaard, "Bearing Pressure and Cracks," *J. of Appl. Mech.,* **6** [1] 49-53 (1939).

81. G. R. Irwin, "Analysis of Stress and Strains near the End of a Crack Traversing a Plate," *J. Appl. Mech.,* **24** [1] 361-364 (1957).

82. M. L. Williams, "On the Stress Distribution at the Base of a Stationary Crack," *J. Appl. Mech.*, **24** [1] 109-114 (1957).

83. T. L. Anderson, *Fracture Mechanics: Fundamentals and Applications, 3rd ed.,* Taylor & Francis, Florence, Kentucky, 2005.

84. S. G. Larsson and A. J. Carlsson, "Influence of Non-singular Stress Terms and Specimen Geometry on Small-scale Yielding at Crack Tips in Elastic-plastic Materials," *J. Mech. Phys. Solids.,* **21** [4] 263-277 (1973).

85. G. A. Kardomateas, R. L. Carson, A. H. Soediono, and D. P. Schrage, "Near Tip Stress and Strain Fields for Short Elastic Cracks," *Int. J. Fracture,* **62** [3] 219-232 (1993).

86. J. Song, W. Curtin, T. Bhandakkar, and H. Gao, "Dislocation Shielding and Crack Tip Decohesion at the Atomic Scale," *Acta Mater.,* **58** [18] 5933-5940 (2010).

87. E. Saether, V. Yamakov, and E. Glaessgen, "New Developments in the Embedded Statistical Coupling Method: Atomistic/Continuum Crack Propagation," Structures, Structural Dynamics and Materials Conference, AIAA/ASME/ASCE/AHS/ASC, 49th, **11** (2008).

88. V. Coffman, J. Sethna, G. Heber, M. Liu, A. Ingraffea, N. Bailey, and E. Barker, "A Comparison of Finite Element and Atomistic Modelling of Fracture," *Modelling Simul. Mater. Sci. Eng.,* **16** [6] 065008 1-15 (2008).

89. I. Vatne, E. Østby, C. Thaulow, and D. Farkas, "Quasicontinuum Simulation of Crack Propagation in BCC-Fe," *Mat. Sci. Eng. A,* **528** [15] 5122–5134 (2011).

90. Z. Nishiyama, "X-ray Investigation of the Mechanism of the Transformation from Face Centered Cubic Lattice to Body Centered Cubic," *Sci. Rep. Tohoku Imperial University, **23** [1] 637-664 (1934).*

91. A. Latapie and D. Farkas, "Molecular Dynamics Simulations of Stress-induced Phase Transformations and Grain Nucleation at Crack Tips in Fe," *Modelling Simul. Mater. Sci. Eng.,* **11** [5] 745–753 (2003).

92. S. Wang, H. Wang, K. Du, W. Zhang, M. Sui, and S. Mao, "Deformation-induced Structural Transition in Body-Centred Cubic Molybdenum," *Nat. Comm.,* **5** [3433] 1-9 (2014).

93. G. Kurdjumov and G. Sachs, "Over the Mechanisms of Steel Hardening," *Z. Phys.*, **64** [5-6] 325-343 (1930).

94. W. Hu, Y. Wang, J. Yu, C. Yen, and F. Bobaru, "Impact Damage on a Thin Glass Plate with a thin Polycarbonate Backing," *Int. J. Impact Eng.*, **62** [1] 152-165 (2013).

95. S. Silling and E. Askari, "A Meshfree Method Based on the Peridynamic Model of Solid Mechanics," *Comput. Struct.*, **83** [19-20] 1526-1535 (2005).

96. P. Branicio, R. Kalia, A. Nakano, P. Vashishta, F. Shimojo, and J. Rino, "Atomistic Damage Mechanisms during Hypervelocity Projectile Impact on AlN: A large-scale Parallel Molecular Dynamics Simulation Study," *J. Mech. Phys. Solids*, **56** [5] 1955-1988 (2008).

97. W. Wang, "An Adaptive Multi-Scale Computational Method for Modeling Nonlinear Deformation in Nanoscale Materials," The Department of Civil and Environmental Engineering, Louisiana State University, 2006.

98. E. Lidorikis, M. Bachlechner, R. Kalia, R. Kalia, G. Voyiadjis, A. Nakano, and P. Vashishta, "Coupling of Length Scales: Hybrid Molecular Dynamics and Finite Element Approach for Multiscale Nanodevice Simulations," *Mat. Res. Soc. Symp. Proc.*, **653** [Symposium Z] (2000).

99. E. Lidorikis, M. Bachlechner, R. Kalia, A. Nakano, and P. Vashishta, "Coupling Atomistic and Continuum Length Scales in Heteroepitaxial Systems: Multiscale Molecular- Dynamics/Finite- Element Simulations of Strain Relaxation in Si/Si3N4 Nanopixels," *Phys. Rev. B*, **72** [11] 115338 (2005).

100. S. Xiao and T. Belytschko, "A Bridging Domain Method for Coupling Continua with Molecular Dynamics," *Comput. Method Appl. Mech. Eng.*, **193** [17-20] 1645-1669 (2004).

101. J. Broughton, F. Abraham, N. Bernstein, and E. Kaxiras, "Concurrent Coupling of Length Scales: Methodology and Application," *Phys. Rev. B*, **60** [4] 2391-2403 (1999).

102. D. Farrell, H. Park, and W. Liu, "Implementation Aspects of the Bridging Scale Method and Application to Intersonic Crack Propagation," *Int. J. Numer. Meth. Engng.*, **71** [5] 583-605 (2007).

103. H. Park, E. Karpov, W. Liu, and P. Klein, "The Bridging Scale for Two-Dimensional Atomistic/continuum Coupling," *Phil. Mag.*, **85** [1] 79-113 (Jan 2005).

104. Z. Guo and W. Yang, "MPM/MD Handshaking Method for Multiscale Simulation and its Application to High Energy Cluster Impacts," *Int. J. Mech. Sci.*, **48** [2] 145-159 (2006).

105. S. Plimpton, “Fast Parallel Algorithms for Short-Range Molecular Dynamics,” *J. Comp. Phys.*, **117** [1] 1-19 (1995).

106. I. T. Todorov, W. Smith, K. Trachenko, and M. T. Dove, “DL_POLY_3: New Dimensions in Molecular Dynamics Simulations via Massive Parallelism,” *J. Mater. Chem.*, **16** [20] 1911-1918 (2006).

107. W. Smith, T. R. Forester, and I. T. Todorov, The DL_POLY Classic User Manual. Daresbury Laboratory, United Kingdom, 2010.

108. D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. Berendsen, “GROMACS: Fast, Flexible, and Free,” *J. Comput. Chem.*, **26** [16] 1701–18 (2005).

109. J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten. “Scalable Molecular Dynamics with NAMD,” *J. Comput. Chem.*, **26** [16] 1781-1802 (2005).

110. M. Tang, G. Hommes, S. Aubry, and T. Arsenlis, “ParaDiS-FEM Dislocation Dynamics Simulation Code Primer,” LLNL-TR-501662, Lawrence Livermore Nat. Lab., 2011.

111. F. M. Ciorba, S. Groh, and M. F. Horstemeyer, “Parallelizing Discrete Dislocation Dynamics Simulations on Multi-core Systems,” *Int. Conf. on Comput. Sci*., **10th** [1] 2129-2137, (2010).

112. M. C. Fivel, T. J. Gosling, and G. R. Canova, “Implementing Image Stresses in a 3D Dislocation Simulation,” *Modelling Simul. Mater. Sci. Eng.*, **4** [6] 581-596 (1996).

113. ANSYS, Inc., ANSYS Academic Research. Release 13.0, Canonsburg, PA, 2010.

114. M. Doi, The OCTA project, 2002, http://octa.jp.

115. V. Ruhle, C. Junghans, A. Lukyanov, K. Kremer, and D. Andrienko, “Versatile Object-oriented Toolkit for Coarse-Graining Applications,” *J. Chem. Theory Comput.*, **5** [12] 3211–3223 (2009).

116. T. Aoyagia, F. Sawaa, T. Shojia, H. Fukunagaa, J. Takimotob, and M. Doic, “A General-Purpose Coarse-grained Molecular Dynamics Program,” *Comput. Phys. Comm.*, **145** [2] 267–279 (May 2002).

117. J. Takimoto, H. Tasaki, and M. Doi, “Prediction of the Rheological properties of Polymers using a Stochastic Simulation,” *Comput. Phys. Comm.*, **142** [1-3] 136-139 (2001) in Proc. XIIIth Int. Cong. Rheology.

118. T. Honda, S. Urashita, H. Morita, R. Hasegawa, T. Kawakatsu, M. Doi, "Dynamic Mean Field Theory for Mesoscale Polymer Simulations," *Kobunshi Ronbunshu*, **56** [12] 762-771 (1999).

119. T. Aoyagi J. Takimoto, and M. Doi, "Molecular Dynamic Study of Polymer Melt Confined Between Walls," *J. Chem. Phys.*, **115** [1] 552-559 (2001) in Proc. Int. Conf. Adv. Polymers and Processing, 217.

120. R. E. Miller and E. B. Tadmor, QC Reference Manual version 1.3, May 2007, url=www.qcmethod.com

121. W. Wang, "An Adaptive Multi-Scale Computational Method for Modeling Nonlinear Deformation in Nanoscale Materials"; PhD. Thesis, The Department of Civil and Environmental Engineering, Louisiana State University, 2006.

122. S. Li, N. Sheng, and X. Liu, "A Non-equilibrium Multiscale Simulation Paradigm," *Chem. Phys. Lett.*, **451** [4-6] 293-300 (2008).

123. Z. Tang, H. Zhao, G. Li, and N. R. Aluru, "Finite-temperature Quasicontinuum Method for Multiscale Analysis of Silicon Nanostructures," *Phys. Rev. B*, **74** [6] 064110 (2006).

124. G. Anciaux, Libmultiscale. 2009, URL http://libmultiscale.gforge.inria.fr/.

125. B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "libMesh: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations," *Eng. Comput.*, **22** [3-4] 237-254 (2006).

126. M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*. Oxford: Clarendon Press, Gloucestershire, UK, 1991.

127. Accelrys Software Inc., Discovery Studio Modeling Environment, Release 4.0, San Diego, CA, 2013.

128. J. D. Gale and A. L. Rohl, "The General Utility Lattice Program," *Mol. Simul.*, **29** [5] 291-341 (2003).

129. MPI Forum. Message Passing Interface (MPI) Forum Home Page. http://www.mpi-forum.org/ (Dec. 2009)

130. W. Humphrey, A. Dalke, and K. Schulten, "VMD – Visual Molecular Dynamics," *J. Molec. Graphics*, **14** [1] 33-38 (1996).

131. J. Li, "AtomEye: an Efficient Atomistic Configuration Viewer," *Modelling Simul. Mater. Sci. Eng.*, **11** [2] 173- (2003).

132. T. Williams, C. Kelley, et al., Gnuplot 4.4: an interactive plotting program, March 2010, url=http://gnuplot.sourceforge.net/

133. J. D. Honeycutt and H. C. Andersen, "Molecular Dynamics Study of Melting and Freezing of Small Lennard-Jones Clusters," *J. Phys. Chem.,* **91** [19] 4950–4963 (1987).

134. R. Sibson, "SLINK: An Optimally Efficient Algorithm for the Single-link Cluster Method," *Comput. J.,* **16** [1] 30-34 (1972).

Cited in the Appendices

135. M. I. Mendelev, M. J. Kramer, C. A. Becker, and M. Asta, "Analysis of Semi-empirical Interatomic Potentials Appropriate for Simulation of Crystalline and Liquid Al and Cu," *Phil. Mag.,* **88** [12] 1723-1750 (2008).

136. D. Wolf, P. Keblinski, S. R. Phillpot, and J. Eggebrecht, "Exact Method for the Simulation of Coulombic Systems by Spherically Truncated, Pairwise r^{-1} Summation," *J. Chem. Phys.,* **110** [17] 8254-8282 (1999).

137. C. J. Fennell and J. D. Gezelter, "Is the Ewald Summation Still Necessary? Pairwise Alternatives to the Accepted Standard for Long-range Electrostatics," *J. Chem. Phys.,* **124** [23] 234101 (2006).

138. H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, "Molecular Dynamics with Coupling to an External Bath," *J. Chem. Phys.,* **81** [8] 3684-3690 (1984).

139. H. C. Andersen, "Molecular Dynamics Simulation at Constant Pressure and/or Temperature," *J. Chem. Phys.,* **72** [4] 2384-2393 (1980).

140. M. Parrinello and A. Rahman, "Polymorphic Transitions in Single Crystals: A New Molecular Dynamics Method," *J. Appl. Phys.,* **52** [12] 7182-7190 (1981).

141. M. Parrinello and A. Rahman, "Strain Fluctuations and Elastic Constants," *J. Chem. Phys.,* **76** [5] 2662-2666 (1982).

142. A. Pedone, G. Malavasi, M. C. Menziani, U. Segre, and A. N. Cormack, "Molecular Dynamics Studies of Stress-Strain Behavior of Silica Glass Under a Tensile Load," *Chem. Mater.,* **20** [13] 4356-4366 (2008).

143. C. J. Bradley, *The Algebra of Geometry: Cartesian, Areal and Projective Co-ordinates,* Bath: Highperception, Buckinghamshire, UK, 2007.

144. ImageMagick Studio LLC, ImageMagick Studio. 1999-2015. http://imagemagick.org/index.php

145. E. Blaísten-Barojas, "Structural Effects of Three-body Interactions on Atomic Clusters," *Kinem,* **6** [A] 71–84 (1984).

146. S. Plimpton, P. Crozier, and A. Thompson, LAMMPS User Manual: Large-scale Atomic/Molecular Massively Parallel Simulator. Sandia National Laboratories, July 2009.

147. H. Tsuzuki, P. S. Branicio, and J. P. Rino, "Structural Characterization of Deformed Crystals by Analysis of Common Atomic Neighborhood," *Comput. Phys. Comm.*, **177** [6] 518–523 (2007).

148. P. Legendre and L. Legendre, *Numerical Ecology. 2nd English ed.*, 853 pages, Elsevier Science, Melbourne, AU, 1998.

APPENDICES

There are six appendices attached to this thesis that act as supplementary material and guides to understanding the technical aspects of the multiscale analysis used in this and other work. The first Appendix A highlights some of the more interesting algorithms involved in classical Molecular Dynamic simulation. Appendix B goes in depth about the unique features of the GP method such as auto-duality and linking with FEA meshes. Appendix C is a guide for GP and FEA model development including some useful tools for more complicated geometries. The user manual for the multiscale code used in this work: PMAP is given in appendix D along with the input files used for specific examples discussed in Chapter II. The processing of the data files and forming useful results is expounded upon in Appendix E and tools for an in-depth analysis are explained in the last Appendix: F.

A. USUAL MOLECULAR DYNAMIC FEATURES

1. Reading EAM Tables

There are many different EAM potential tables available, the most popular format is “setfl” that is used by LAMMPS. The NIST website has a database of EAM potentials located online at http://www.ctcms.nist.gov/~cbecker/, DL_POLY has a different type of format but the concept is the same. The GP code, as of version 42, can use both formats, the setfl and DL_POLY's format. Before its capability to read EAM tables, it had only the use of Mendelev's 2008 potential for Copper.[135]

Similar functions and subroutines, as DL_POLY uses to read in the “TABEAM” file, were developed in the GP code. Specifically, the subroutine *JohnsonMix_ Potential_Tables* has added to it a block to read the “TABEAM” file that DL_POLY uses (lines 3898-3960 of version 42). Reading the “setfl” formated files another block was added similar to the way that LAMMPS reads the data (lines 3961-3996). The new subroutine *metal_deriv* was added on line 4205 to tabulate the derivatives of the EAM potential functions read from the file by using five-point interpolation. This subroutine is very similar to the one that DL_POLY uses.

Potential arrays named: Tphir, Tpsir, Tfrho are read directly from the potential table file. The arrays named: Tdphir, Tdpsir, and Tdfrho are their derivatives, respectively, generated by the subroutine *metal_deriv*.

The potential functions that the subroutine *Compute_Forces* uses [namely: phir(), psir(), frho(), dphir(), dpsir(), and dfrho()] originally contained the analytical functions of Mendelev's potential,[135] but have been augmented to interpolate the potential arrays if using a table file. It uses three-point interpolation for this. This method required no changes to the *Compute_Forces* subroutine, only to the functions that it uses, so the possibility of introducing errors is minimized.

When using setfl potential files, it is important to make the potential cut off radius consistent or rounded down from the value listed in the potential file, because all of the data listed in the potential file is only for radii less than that value, if a larger cut off radius is set in the GP input file then there will be potential interaction errors.

2. Damped Shifted Coulomb Potential (DSC)

The inclusion of the coulombic potential is essential to accurately simulate ionic materials. The first method that comes to mind when exploring coulomb potential summation techniques is to simply sum the potential of all neighbors within a cutoff; this is the practice for the Van der Waals (VDW) potentials. However, problems arise with this method, most notably that the neighbors being summed, most of the time end up not being charge neutral. Even using a larger cutoff radius does not solve this problem; this is why most summation simulations use the Ewald summation method.[136] The Ewald sum effectively considers all atoms in a simulation by using their reciprocal lattice coordinates, this method requires the system to have 3D periodic boundary conditions and is an $O(N^2)$ algorithm.[137] To compare, simple pair-wise calculations are an O(N) algorithm. This means that the Ewald summation algorithm is much more computationally intensive and a pair-wise solution would be much more efficient. Figure 53 shows the traditional relation between Ewald and pair-wise uses and their differences.

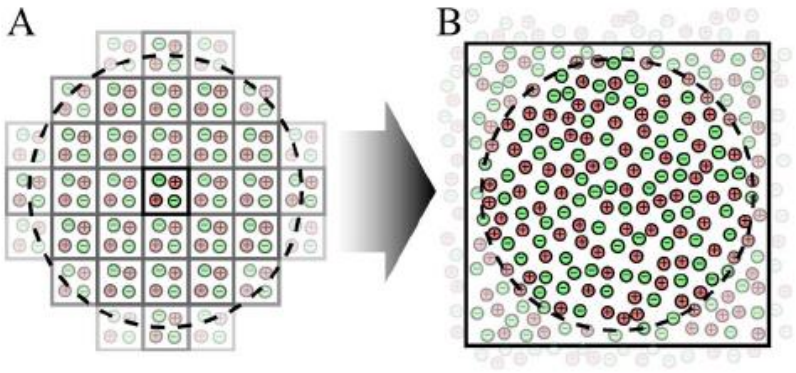


Figure 53. The Ewald sum (A) replicates the simulation box infinitely. Radial cutoff methods (B) should be used for amorphous and geometrically unique systems[3].

The idea behind the pair-wise coulomb summation method is to maintain charge neutrality within the cutoff radius. This can be done by adding a negative charge, $-q_j$ on the surface of the cutoff sphere for every charge, q_j within the sphere, see Fig. 54.

Wolf et al. compared this technique "to the classic problem of determining the potential at the center of a conducting, grounded sphere due to the presence of a point charge, q, at some point **r** = **rń** within the sphere".[136]

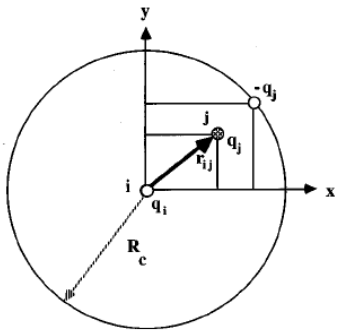


Figure 54. Every charge, q_j has a corresponding charge $-q_j$ located in the same direction but on the cutoff radius' spherical surface.[136]

a. Implementation

The equations describing this method can be seen in Eqs. (34) and (35) for the pair potential and pair force interaction, where R_c is the potential cutoff radius, r is the distance from atom *i* to *j*, and α is the damping parameter.[137,138]

$$V_{DSF}(r) = \frac{q_i q_j}{4\pi\varepsilon_0}\left[\frac{erfc(\alpha r)}{r} - \frac{erfc(\alpha R_c)}{R_c} + \left(\frac{erfc(\alpha R_c)}{R_c^2} + \frac{2\alpha}{\pi^{1/2}}\frac{\exp\left(-\alpha^2 R_c^2\right)}{R_c}\right)\right], r \le R_c \qquad (34)$$

$$F_{DSF}(r) = \frac{q_i q_j}{4\pi\varepsilon_0}\left[\left(\frac{erfc(\alpha r)}{r^2} + \frac{2\alpha}{\pi^{1/2}}\frac{\exp\left(-\alpha^2 r^2\right)}{r}\right) - \left(\frac{erfc(\alpha R_c)}{R_c^2} + \frac{2\alpha}{\pi^{1/2}}\frac{\exp\left(-\alpha^2 R_c^2\right)}{R_c}\right)\right], r \le R_c \qquad (35)$$

Specifically, in the GP code version 42, the potential and force functions are tabulated for each type of atomic interaction. This is conducted in the subroutine *JohnsonMix_Potential_Tables* on line 4015. There are two constants defined, *vcon* and *fcon*. They are shown in Eqns. (36) and (37) and lines 3890-1:

$$vcon = \frac{erfc(\alpha R_c)}{R_c} \qquad (36)$$

$$fcon = \frac{erfc(\alpha R_c)}{R_c^2} + \frac{2\alpha}{\pi^{1/2}}\frac{\exp\left(-\alpha^2 R_c^2\right)}{R_c} \qquad (37)$$

For example, there are arrays called, *PhiTab* and *DPhiTab* for potential and force interactions, respectively (see lines 4015-6). These arrays are populated by calling a function that takes four parameters. First, the type of interaction; this is an integer defined at the beginning of the simulation. Second, the radius between two possible atoms; this is defined to vary as increments of the cutoff radius divided by the TableSize. Third, an

array holding the potential parameters; this is given in the input file. Fourth, the charge product of atom, i and j; this is used in the coulomb part. There are two such functions, the first for the potential and the second for the force. They are, respectively, *pot* and *dpot* located on lines 4093 and 4145. These functions can only be accessed in this subroutine. The potential type must be specified, because these functions can produce three potential equations. First, the Morse potential model as type 10, second, the Buckingham as type 20, and third, Lennard-Jones as type 30. The EAM potentials are not defined by these functions but Mendelev's 2008 EAM potential for Copper has type 40, EAM tables in DL_POLY's format as type 41, and EAM tables in setfl format as type 42. If the coulomb potential is to be included, the potential type will increase by 1, so Morse would be type 11. For these types, the coulomb term will be added to the regular type. This addition can be seen in the *pot* function on lines, 4115, 4126, and 4136. The GP code simply adds the coulomb component to the base VDW potential; for interactions that are neglected for their small VDW potential yet require the coulomb portion, a dummy VDW potential is used with null values so that only the coulombic portion is included.

3. Simulation Revival

When computational environments are unstable or time limits are imposed upon simulations, it becomes desirable to continue the simulation where it left off. For a typical MD simulation the required information to continue, is the initial simulation parameters from a control type of input file, and the position, velocity, and force/acceleration of all of the atoms, since these simulations are not history dependent.

For GP simulations there is additional information needed for Revival. The Neighbor Link Cells (NLCs) also need to be recorded as well as each scale's box size for version 41 of the GP code. During a GP simulation, every time the configuration is written, the Revive.MD file is rewritten with the timestep, loadstep, whether the Revive.MD file has velocity, acceleration and NLCs, the Current Global BoxSize, and the Equilibrated Global BoxSize, on the first line. After this is listed every atom and particle. For example, below is shown the first 5 lines of a sample Revive.MD file generated from GP code version 44.

```
82500  20    T    25681.86523  18561.95117  1197.940308  21581.39062  18561.95117
1197.940308
    5.101561546325684E-03      55.361032485961914E-03    999.718487262725830E-03
5   10029
   -1.113022976000000E+09   -1.516011392000000E+09  947.425088000000000E+06
   -8.760175275863441E+18   -9.640542141595779E+18   38.336949002595992E+18
    5.070656538009644E-03      55.314928293228149E-03    247.275263071060181E-03
5   10029
```

The particle scale and type are listed after the position. If the particle was clamped it will have the clamp domain number multiplied by 10000 added to the particle type, so one can see that the first two particles shown above are clamped. If the particle was in a locally thermostatted domain it would have 9900 added to the type if in the first thermostatted domain, and -100 for every additional local thermostatted domain. If the particle was in an Auto-Decomposition Domain (ADD) also used as a local stress domain, it would have that ADD number multiplied by 100 added to the type, this way ADDomains or thermostatted domains can also be clamped and remain identifiable throughout the simulation.

To revive a simulation, make sure the Revive.MD file is in the simulation directory and toggle the logical value in the input file to ".true.". If the GP code version being used is version 41 or above, the logical value in the input file for relinking the NLCs can be ".false." since the Revive.MD file contains all of the NLCs for every imaginary particle. Version 34 did not have the correct NLCs recorded in the Revive.MD file so when reviving from that version it is necessary for the NLCs to be relinked.

If something happened to the Revive.MD file rendering it useless, it is possible to revive from an MD3 configuration file. All that is required is the proper Revive.MD header, the first line must be placed in the beginning of the MD3 file and the file must be named "Revive.MD". It is important that the logical value be set to "F" for false, since there are no velocities, accelerations, or NLCs in the MD3 configuration files. It's also important to make sure the correct load step is selected; otherwise the simulation could revive at a different strain.

4. Periodic Boundary Conditions (PBC) using Verlet Neighbor Lists

There are two parts to successfully implement Periodic Boundary Conditions (PBCs) in an MD simulation using Link-Cells to find the neighbors of an atom. The first is to refold atom positions if they wander beyond the boundary. The second is to make sure that atoms near the boundary can "sense" the atoms on the other side, in essence, the force interactions must also be refolded.

The first is taken care of with the subroutine Refold_Positions on line 2536 of Version 42. It works on the basis that all particle positions are normalized to the Global boxSize, so all positions range from -0.5 to 0.5, after subtracting 0.5. This is convenient because if a particle is outside of the box its position in that direction will round up to 1.0. When this happens it will be repositioned to the other side of the box simply by subtracting or adding 1.0, depending on which side of the box it was outside of. This algorithm is accomplished following the method of Allen and Tildesley[126] for particle, i, in the X direction:

```
if(PBC(1)) pos(1,i)=(pos(1,i) - anint(pos(1,i)-0.5))
```

This can be seen on line 2559. The position vector originally ranges from 0.0 to 1.0, so to center it, 0.5 is subtracted. This gives -0.5 to 0.5. The function anint() rounds its argument to the nearest integer preserving its REAL data type.

In order to refold the forces across the boundary there needs to be neighbors refolded across the boundary to provide those forces. Neighbors are found by searching through a neighbor list. So these folded neighbors must be present in these lists otherwise they won't even be considered for force calculations. These neighbor lists are generated periodically through the simulation and must be able to add folded neighbors to particles' neighbor lists. The lists are made by searching inside of Verlet Neighbor List Cells. These cells are a little larger than the interatomic cutoff radius and fill the scale's Box size, for version 41, or fill the entire model box size. Every particle in the model is inside of one of these cells. So for any particle, *i*, the particles in the cells around it are considered for its neighbor list. If one of these cells next to particle i's cell is outside of the model, a different cell is used, specifically the cell on the other side of the model is used in its stead, and the particles within are folded across the model to be used in the neighbor list.

The line of code that specifies which link-cell is being considered is line 3208 of version 42:

```
JCELL=ICELLVAL(tmpjcell(1),tmpjcell(2),tmpjcell(3),scl)
```

In this case IXYZ holds the coordinates of the current cell, the one with particle i in it. The variables, k1, k2, and k3 are offset values that when added to IXYZ will specify the neighboring link-cell, array: tmpjcell(:). It is this cell coordinate that must be refolded if it's not part of the model. The procedure for doing this is similar to that in the subroutine Refold_Positions.

To accomplish this, a temporary JCELL array is needed to hold the values of IXYZ(:)+k . This makes it easier to refold. Since the total possible value for a coordinate of JCELL is not 1 as is the case with the position, NCELL must be used in the refolding process. Before line 3208, the tempjcell(:) array should be defined as well as the refolding statement:

```
tmpjcell(1)=IXYZ(1)+k1
tmpjcell(2)=IXYZ(2)+k2
tmpjcell(3)=IXYZ(3)+k3
tmpjcell(:)=tmpjcell(:)-
NCELL(:,scl)*anint(.95*(float(tmpjcell(:)+1)/float(NCELL(:,scl))))
```

Here, in the anint() function tmpjcell(:) is divided by NCELL to normalize it, so the same formula may be used as for refolding position. After the anint() function it is multiplied by NCELL(:,scl) to bring it back to a dimensional value. This statement will reset the JCELL to one that is on the opposite side of the model, rather than empty space which would make it a surface, see Fig. 55.

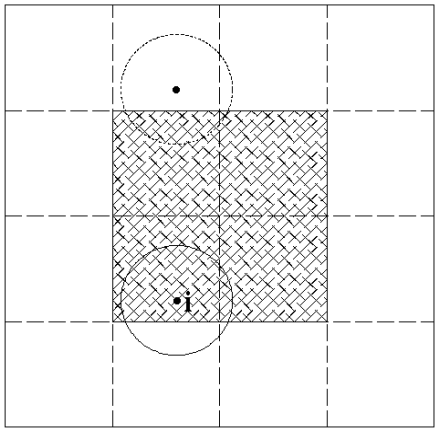


Figure 55. Particle *i* with cut-off radius, including refolded area (dotted circle).

There are two other places that must be altered that relate to the use of forces, they are those that use the distance between an atom and its neighbor, these places should also take into account the situation that an atoms neighbor may be on the other side of the model. In this occasion, the distance must be reduced to a refolded distance. Lines 2750, and 3222 are in subroutines *Compute_Forces* and *UpdateLinkList* respectively, and are shown below for refolding in the X direction:

```
Rij(1) = BoxSize(1)*(Sij(1)-anint(Sij(1)))
```

Where Sij is the normalized distance between atom i and j. The multiplication by BoxSize(:) is to dimensionalize the distance into angstroms. After these three changes, the program should be able to use PBCs in any direction.

5. Barostat (NPT)

The NPT ensemble maintains a constant number of particles, pressure, and temperature. A simulation achieves a constant number of particles easily by explicitly defining each one and not spontaneously creating or destroying them. A simulation maintains a constant pressure and temperature by using a barostat and thermostat respectively. A barostat alters the system pressure by modifying the simulation cell size and hence density according to the current pressure. The same goes for thermostats, but with the velocity of the particles according to the system temperature. Since they are so similar, similar algorithms can be used to control both quantities.

a. Berendsen Barostat

The chosen barostat is one by Berendsen[138] "[this] method does not drastically alter the dynamic trajectories and is easy to program". The equations used are listed below. It begins with the average virial stress, *w* in equation (38).

$$w = \frac{1}{3}\sum_{i=1}^{N} r_i \cdot f_i \tag{38}$$

The instantaneous pressure of a statistical system can be given by equation (40), derived from (39).

$$PV = Nk_B T + w \tag{39}$$

$$P = \rho k_B T + \frac{W}{V} \tag{40}$$

With these equations, the factor for repositioning the cell size and subsequent particle positions is given by equation (41).

$$\chi = 1 - \frac{\delta t}{\tau_P}\left(P_{req} - P\right) \tag{41}$$

How this factor is used to scale the positions and cell size is shown in the next equation.

$$r' = \chi^{1/3} r \tag{42}$$

Implementing the Berendsen barostat was straight forward, by using equations (41) and (42). The rescaling is performed right after the velocity scaling of the thermostat. The coupling parameter, τ_P also called the "rise time" shown in equation (41) is a difficult parameter to use, the larger the parameter the slower the changes to the box size, and the smaller the value the more likely high frequency oscillations are to occur. At the start of the simulation it was found that a very large rise time was needed until the system settled down a bit, if too small of a rise was given, it would inflate the box size very quickly and very largely. This is due to the initial high quantities of stress, pressure and temperature at the early stages of simulation. However, as the simulation progresses it tends to be necessary for the rise time to become smaller, so as to allow a faster pressure equilibration. Using the analogy of a pressure piston, the larger the rise time the more massive and higher inertia the piston has, and the smaller the rise time the more light and less inertia it has.

b. Anisotropic barostat

A closer look into the use of barostats with independent dimensionality was performed. It was found that the Berendsen barostat has the capability to be modified for anisotropic triclinic systems. In this case the instantaneous pressure becomes a tensor.[138] The Andersen barostat[139] is only appropriate for systems assumed to be isotropic. Parrinello and Rahman, providing examples,[140,141] adapted the Andersen method by introducing a time dependent shape using variable crystal lattice vectors and an anisotropic stress tensor.

The Parrinello and Rahman method based on Andersen's is very mathematically intensive and difficult to implement compared to Berendsen's.[126] Pedone et al.[142]

modified DLPOLY to allow for unconstrained simulation of tensile loading under 3D PBCs. The unconstrained aspect was accomplished using the Berendsen barostat to anisotropically relax the strained system. The scaling they used is shown in eqn. 43 and stress in eqn. 44.

$$\eta = 1 - \frac{\beta \Delta t}{\tau_P}\left(P_{ext} 1 - \sigma\right) \tag{43}$$

$$\sigma_{\alpha\beta} = \frac{-1}{V}\left(\sum_i p_{i\alpha} p_{i\beta} / m_i + \sum_i r_{i\alpha} f_{i\beta}\right) \tag{44}$$

Where P_{ext} is the applied pressure and the "1" indicates the loading direction. These equations are consistent with those described by Berendsen[138] where their scaling factor and stress/pressure are as follows.

$$\mu = 1 - \frac{\beta \Delta t}{3\tau_P}\left(P_0 - P\right) \tag{45}$$

$$P = \frac{1}{V}\left\{\sum_i m_i v_i v_i^T + \sum_{i<j} r_{ij} F_{ij}^T\right\} \tag{46}$$

Frequently the β/3 in equation 45 is included with the rise time. The particle coordinates and box size are scaled by equation 14.

$$r_i^{'} = \mu r_i \tag{47}$$

With these two sets of equal equations, it should be apparent how to implement this anisotropic barostat for full 3D freedom.

Detailing the computation involved when using the equations described above, one finds that the scaling parameter **μ** as defined below in equation 48, is a 3x3 matrix that is subsequently multiplied by the position vector to obtain the new positions, see equation 49.

$$\mu = 1 - \frac{\beta \Delta t}{3\tau_P}\left(P_0 - P\right) \tag{48}$$

$$r_i^{'} = \mu r_i \tag{49}$$

$\mathbf{P_0}$ is the requested pressure array, **P** is the current pressure array which comes from the previously calculated stress with an additional kinetic term. **1** is the identity matrix and r_i is the position vector of particle *i*.

In the code, the stress array has 6 components; the full matrix has 9 elements but is symmetric, so it can be compressed into 6 components. Interestingly, the code's matrix equation for equation 48 looks like equation 50, below.

$$\begin{bmatrix} \mu_{(1)}\mu_{(6)} & & \\ & \mu_{(2)}\mu_{(4)} & \\ \mu_{(5)} & & \mu_{(3)} \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix} - \frac{\beta \Delta t}{3\tau_P} \left(\begin{bmatrix} P_{0(1)}P_{0(6)} & & \\ & P_{0(2)}P_{0(4)} & \\ P_{0(5)} & & P_{0(3)} \end{bmatrix} - \begin{bmatrix} P_{(1)}P_{(6)} & & \\ & P_{(2)}P_{(4)} & \\ P_{(5)} & & P_{(3)} \end{bmatrix} \right) \tag{50}$$

Taking advantage of the symmetricalness of $\boldsymbol{\mu}$ we have then, equation 49 looking like:

$$\begin{bmatrix} r'_{ix} \\ r'_{iy} \\ r'_{iz} \end{bmatrix} = \begin{bmatrix} \mu_{(1)}\mu_{(6)}\mu_{(5)} \\ \mu_{(6)}\mu_{(2)}\mu_{(4)} \\ \mu_{(5)}\mu_{(4)}\mu_{(3)} \end{bmatrix} \begin{bmatrix} r_{ix} \\ r_{iy} \\ r_{iz} \end{bmatrix} \tag{51}$$

Equation 51 is used on all particle positions and the Box Size in a decomposed manner. According to matrix multiplication we have the following three equations for each component of the new positions:

$$\begin{aligned} r'_{ix} &= \mu_{(1)}r_{ix} + \mu_{(6)}r_{iy} + \mu_{(5)}r_{iz} \\ r'_{iy} &= \mu_{(6)}r_{ix} + \mu_{(2)}r_{iy} + \mu_{(4)}r_{iz} \\ r'_{iz} &= \mu_{(5)}r_{ix} + \mu_{(4)}r_{iy} + \mu_{(3)}r_{iz} \end{aligned} \tag{52}$$

However the positions in the code are normalized to the current Box Size, so the true particle positions are actually a function of Box Size, so if the Box Size is changed so are all of the particle positions. The code takes advantage of this by only changing the size of the current Box Size to control the pressure.

B. GP FEATURES

The GP code has many unique features that go above and beyond the average Molecular Dynamics Simulator. In this section will be discussed the most interesting and important features. Such as, domains that can automatically decompose into lower scale particles, and connecting high scale particles with multiple FEA domains.

1. Automatic Duality Domains (ADD) with stress and energy (internal duality)

These local domains (ADDomains) are defined at the beginning of the simulation. Everything inside of the domains are given an ID unique to that ADDomain. These particles/atoms are tracked and their energy averaged. At the same time, the initial ADDomains record the local stress, as material passes through them. These ADDomains are specified in the input file and should be specified in a way that may be used for auto-duality, meaning that they can change from being S2 particles to S1 atoms depending on certain threshold values of maximum local Von Mises stress. What will be decomposed are the particles that were assigned the unique ID numbers, identifying them to be a part of that ADDomain. So the initial ADDomain is used for calculating the local stress while the current ADDomain is the collection of atoms/particles that have a certain average energy.

The actual mechanism for auto-duality will be evaluated every output step, usually every 0.5ps. It is a simple evaluation, and can be found on line 2457 of Version 44 of the GP code. If the largest value in the `sum(VMavg(ADD,:))/10` array is larger than the threshold value given in the input file for decomposition, it will decompose that ADDomain which corresponds to the first dimension of the array element of `VMavg`, and if the minimum value is below the lower threshold value, it will lump the domain.

Both lumping and decomposition is performed by the subroutine, *composition*, located on line 4623, it simply changes the correct scale real-particle into an imaginary one, and the imaginary particles into real ones. It then calls the subroutine LinkNLC, (line 4678) which relinks all of the particles in the ADDomain that were just

"composed". The LinkNLC subroutine is called at the beginning of the simulation, before dynamics occur, to relink all of the NLCs properly.

The GP code version 20 was developed to allow all particles, both real and imaginary, to have NLCs. This is advantageous for the automatic scale duality feature. There are two main parts of the auto-duality feature, first the preparation then the detector and process. Separate from this is the ability for the NLCs to break under a specific strain.

a. Preparation

This preparation begins on line 1084, at the end of the Initialize subroutine. What needs to be prepared are the NLCs for all particles. This preparation is only performed when indicated from the input file; it asks if the user wants to relink the NLCs. If this is true, then the preparation is prepared.

i. Special List

To relink the particles, a special neighbor-list must be used. To create this special neighbor-list to link NLCs, an option was added to the subroutine: *Update_List*. This new parameter is a Boolean, when it is true, it creates neighbor-lists for regular dynamics of the simulation, i.e. one scale only interacts with the same scale. When this parameter is true, it creates neighbor-lists for the generation of NLCs. Specifically, one scale looks for adjacent scales, *excluding* its own scale.

The special call to Update_List is on line 1086, and the actual subroutine is on line 3136. The main alteration to this subroutine is to the particle "filter". On line 3258 is the "filter" for neighbor-lists of regular dynamics. It makes sure that the candidate j particles are the same scale regardless of their realness. On line 3275 is the "filter" for neighbor-lists of NLC linking. It makes sure that the pair particles are not of the same scale, and does not care whether they are imaginary or not. This means that an imaginary particle can have an imaginary atom in its NLC, which should be fine.

ii. NLC Linking

The generation of NLCs for either real or imaginary particles, is the same. On line 4678 begins the subroutine LinkNLC. Its general structure is a double loop, the outer loop as particle, i, and the inner loop as particle, j. It acquires candidate j particles using

the neighbor-lists created by the subroutine Update_List(.false.) as previously described. The core of this routine runs on particle pair differences, see line 4726. These particle differences are tested and corrected for Periodic boundary conditions, line 4729. Also, and most likely redundant, line 4743 filters out candidate j particles if they are too far away, according to the NLC strain formula.

This inner loop collects all candidate j particles (SymNei:line 4753), along with their positional differences (banl:line 4751) and scaled distances (dist:line 4752). All three of these collections are sorted from the smallest distance to the greatest, if there exists any, using the sort algorithm by John Mahaffy in March 10, 1995, line 4758. Now that they are sorted, they can be entered into the NLC for particle, i, lines 4775 & 4787. The maximum number of NLC constituents is defined to be 12, line 4777. During the population of the NLC, the positional differences (banl) are being summed into the variable, neighsum, lines 4774 & 4781. This is divided by the number of NLC constituents (CN), line 4788. This is the average position of all of the NLC constituents, the negative of which is the error vector, errvect, line 4790.

iii. NLC Breakage

NLCs can be allowed to break when their constituents separate from each other, in a way, inducing a strain on the NLC. The NLC strain is determined when each imaginary particle's position is evaluated from its NLC constituents' positions.

The subroutine *Evolve_Sample0* (for equilibrium) has additional debugging information that is dumped to Standard Error (File unit=0) because there really should not be any reason for an NLC to break during equilibration, unless under special circumstances, hence the debugging info. There is on line 1835 a test for the option to break NLCs, this option is specified in the input file, whether to allow NLCs to break or not. The strain level of the NLC is a function of the largest scale. For example, whether it's an imaginary atom linking to real particles or if it's an imaginary particle linking to real atoms. It uses whichever is the larger scale. This scale factor is on the next line, 1840. If the NLC constituent is too far away, it will make a call to BreakingNLC. This subroutine takes two arguments, the first is the global ID of the imaginary particle under consideration, and the second is the global ID of the real particle that will be removed from the NLC.

The BreakingNLC subroutine is located on line 4817 and deletes the entry from the NLC by moving all later NLC constituent IDs down, to cover it up, line 4842. Thus removing the offending constituent and subtracting the number of NLC constituents by one.

There is another NLC strain detector in the same spot but in subroutine Evolve_Sample, this is for the evolution between loading steps. It's located on line 2230 with the same parameters, but without any debugging output, since, for example, there was a crack propagating, there would be a very large amount of NLCs breaking and writing data every time slows the simulation tremendously.

b. Detector and Process

The ADDomain energy detector is located in subroutine *Evolve_Sample*, because this is the subroutine that evolves the sample between loading steps. So if an ADDomain was going to change energy it would be after a load step, when the forces and dynamics were being assigned.

i. Detector

The detector is located on line 2457. It compares the maximum VM stress for each ADDomain every Load-output step (LDOuts), which is specified in the input file, when the ADuality variable is set to true, also specified in the input file. If ADuality is true, that means that the simulation will attempt to automatically lump or decompose the ADDomains according to their maximum VM stress values.

It searches through all of the ADDomains' maximum VM stress and compares it to the stress range specified in the input file. The first value is the threshold to lump and the second is the threshold to decompose. After an ADDomain is decomposed or lumped, a small array called ADrecord is set to -1 or 1 respectively, this array keeps track of which domains have or have not been lumped or decomposed. This saves time trying to compose domains that already have been. After the compositions, the Boolean variable ListUpdateRequested is set to true, because after composition there is a different number of real particles, and these new real particles need to be included in the neighbor-lists, so that regular dynamics can be performed on them.

ii. Process

The actual decomposition or lumping process is performed by the subroutine: *composition*, located on line 4623. It takes three arguments, the first is the ADDomain number, the second is the scale of the real particles/atoms, and the third is the composition direction, for example, -1 for decomposition (going down a scale) or +1 for lumping (going up a scale). This is why in the detector on line 2459 the call to composition has "(i,2,-1)" this will decompose the S2 real particles in the ADDomain i; and on line 2462 "(i,1,1)" this will lump the real S1 atoms in the ADDomain i.

The subroutine, *composition*, simply searches all particles/atoms in the model for the ones that are in the ADDomain. This "filter", line 4641, uses implicit integer math; since copper atoms have an atomic number of 29, the copper particles that are in ADDomain 2, for example, will have an atomID of 229. So this filter divides their atomID by 100 to test whether it is in the correct ADDomain. Then it tests whether it is an imaginary particle, and if it is, whether it is of the correct scale. If it is then it turns it into a real particle by making the ID positive. If it was a real particle, it would make sure it is of the right scale, and then make its ID negative; to become an imaginary particle. These determinations are made by the if statement on line 4643.

2. Link GP with FEA

Currently the FEA domains used are two dimensional so coupling with GP requires that the GP model be very thin in the Z direction and best be in a Periodic condition.

The typical equilibration time for GP models is used as equilibration for the GP model without being connected to the FEA mesh. After this separate equilibration the FEA mesh is connected to the GP model. See section 2.2.1 for details about the connexion method. After this connexion the model is "loaded" at zero strain until the FEA-GP interface converges. This acts as another equilibration for the entire global model before any true loading begins. Loading is nonlinear with time, as the interface must converge before the next load step begins. (or else time-out)

The FEM domain is solved using The Cholesky decomposition computed using the subroutine SPBSV of the Linear Algebra Package, LAPACK.

a. FEA-GP connexion

The coupling between the GP and the FE nodes is through a two layered interface. Figure 56 shows that the inner layer (WF) directly modify the nodes that they are connected to, and the outer layer (WG) is defined by the FE nodes. In this way there are no surfaces and the connection condition is through displacement or positions.

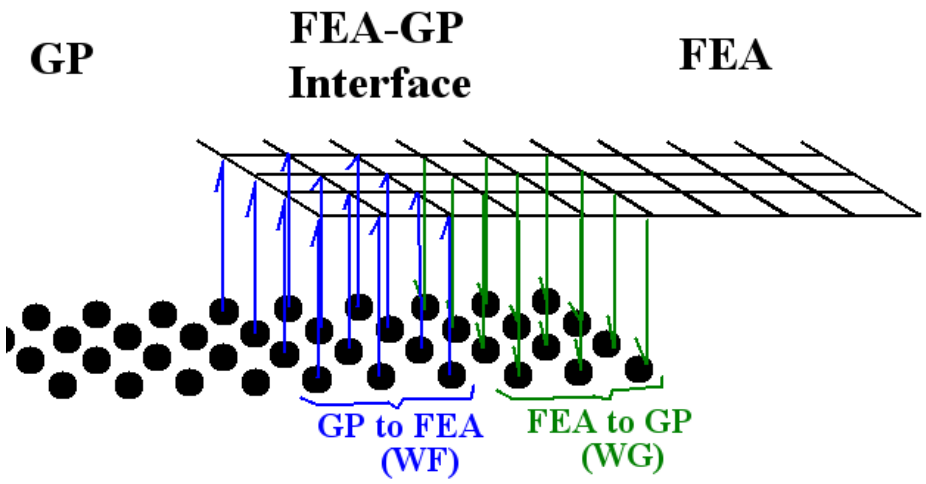


Figure 56. Two layer DC interface.

This coupling can be done by carefully matching the element nodes in the interface with the lattice positions of the material. This is relatively simple for simple cubic structures but becomes much more difficult for more complicated structures. Since this has a one-to-one correspondence between particle and node it would require a very detailed FE pattern to be made in the interface. To avoid this, averaging particles for each node is a better choice.

Of these two interface domains, one is called the “WG” domain. The FE nodes in the WG domain assign the GPs displacement from the FEM calculation. This allows the other free GPs to interact with them. The GPs in the WG domain are fully fixed during GP relaxation and are only moved when FEM gives them new displacements. This fact will cause wave reflexion, but since the connexion is far from the atomistic domain these reflexions' effect to the atomistic domain are minimized.

The other interface domain is called the “WF” domain. This domain is inside of the GP model and averages the GPs position in the WF domain to assign the FE nodes

new displacements. These displacements are used as boundary conditions for the next FEM calculation.

The WG and WF domains are defined at the beginning before loading but after GP equilibration, before the simulation is run. These interface domains are special types of clamping domains. There is an integer after the two clamp vectors for a given clamp domain, this integer is -i if the clamp domain is a WG domain, and i if the clamp domain is a WF domain, and it is 0 if it is not related to an FEA interface domain, also known as a regular clamp domain; neither WG nor WF. The i is the number of the FEA mesh being used, if there are two separate meshes from two separate files, this indicates which file the domain is to be used with. Below are examples of clamp domains (as of Version 56):

```
rec T 0.0 0.0 123.5 142.0 0.0 0.0
1.0 1.0 1.0   0.0 0.0 0.0   -1        !WG (-i_FEA)
rec T 0.0 0.0 105.4 112.6 0.0 0.0
1.0 1.0 1.0   1.0 1.0 1.0    1        !WF (i_FEA)
rec T 0.0 0.0 -30.0 -22.5 0.0 0.0
1.0 1.0 1.0   1.0 0.0 1.0    0        !bottom clamp fixed in Y
```

All “regular” clamp domains are defined based on the initial model's geometry, before equilibration; all particles that start initially inside of any of the clamp domains will be assigned to that clamp domain and remain in that clamp domain for the entirety of the full simulation (both equilibration and loading). WG and WF domains are not, they are determined after GP equilibration so that any surface deformation will not affect the shape of the FEs. This clamp determination is performed in the subroutine *Loading* because it can be easily done after equilibration and before loading, it can be seen on line 3725.

After the GPs are defined to be in either the WG or WF domain they are used to calculate which FE node they are closest to. For Version 14 of the FEA code, line 348 connects all WF GPs to their closest node, and line 378 connects the WG GPs. This will assure the inclusion of all particles in the WG or WF domain.

To interpolate the displacement between FE nodes for each WG GP. Then the question becomes, “what nodes to use for this interpolation? And how do we find them?”. Naturally the nodes around the particle needing to be interpolated should be used, but how to find them? Using a radial search for the nearest three nodes could be inaccurate and time consuming. Since the FEM already uses a nodal connectivity array for each

element, it makes sense to use that to find neighboring nodes. On line 414 of the FEA code, is a routine that finds which element the WG particles are in. This will narrow down the possible nodes to four. When assigning the new positions for the WG GPs, on line 809, a function is called, named *interp()* that will interpolate the WG GP's position given the element that it belongs to, The node that the GP is connected to and the GP's initial position.

When the GP is fully inside of an element, bilinear interpolation is used. This is not trivial as the inverse transformation for quadrilateral isoparametric elements must be performed. It is commonly believed that no explicit solution to the inverse transformation exists.[52] Following the solution technique described in detail by Chongyu Hua[52] we are able to perform the interpolation. However if the GP is on an edge or outside of the element the solution becomes undefined. In such a situation the GP code is designed to fall-back to a less accurate interpolation technique using barycentric coordinates[143] with only three of the closest element nodes. The advantage of this is that it can extrapolate unlike the bilinear solutions.

A position inside of a triangle is very easy to be used to interpolate it's third variable by using Barycentric coordinate transformations,[143] assuming that the third variable is known at each vertex, which it is in this case since this will be interpolating displacement and each vertex is an FE node with displacement, see Fig. 57.

This interpolation procedure was made into a Fortran90 function that returned the new particle position based on the interpolated displacement. This function can be seen on line 934. Barycentric coordinates were introduced (1827) by August Ferdinand Möbius[143] where $T \cdot \lambda = r - r_3$, λ is the vector of barycentric coordinates, r is the vector of Cartesian coordinates, and T is a matrix given by:

$$T = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix} \tag{53}$$

thus:

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{det(T)} = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \tag{54}$$

$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{det(T)} = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)} \tag{55}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 \tag{56}$$

Interpolating a point inside of the triangle for f(r) then becomes:

$$f(r) = \lambda_1 f(r_1) + \lambda_2 f(r_2) + \lambda_3 f(r_3) \tag{57}$$

This linear interpolation is automatically normalized since

$$\lambda_1 + \lambda_2 + \lambda_3 = 1. \tag{58}$$

For our case the function f(r)=u(x_i).

Thus each GP in WG can have an interpolated displacement value.

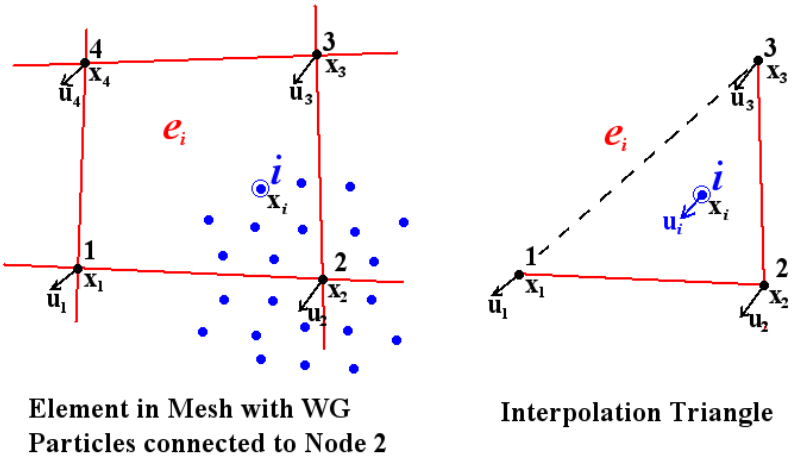


Figure 57. Particle displacement interpolation.

It was determined that using the FEA results to determine the position of the WG domain immediately after applying load was causing compressive stress at the WG-WF interface. The GP code was slightly modified from version 28 to 29 to prevent the WG domain from being redefined after load. Figure 58 details this most current load flow. This allows the WG domain to load with the GP particles and allow relaxation of GP in a loaded state, see point (c) of Fig. 58. Then new WF positions are sent to FEA to recalculate more appropriate positions for WG, see point (d) of Fig. 58. For a more detailed flowchart of the FEA-GP code V29f10 see Fig. 59.

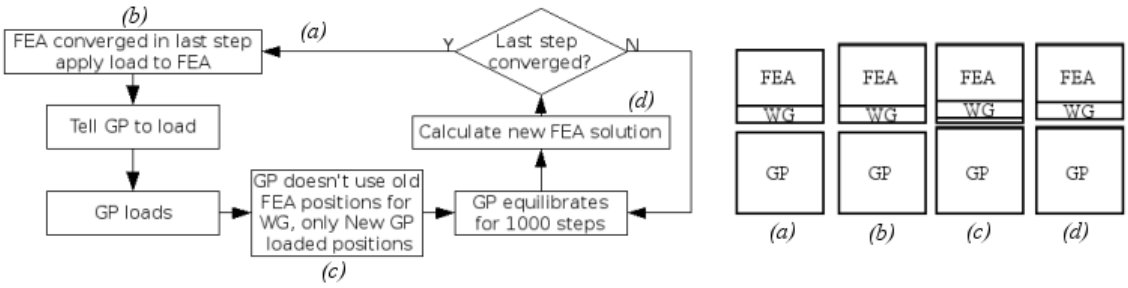


Figure 58. Current loading flow. as of FEA version 10 and GP version ***29***.

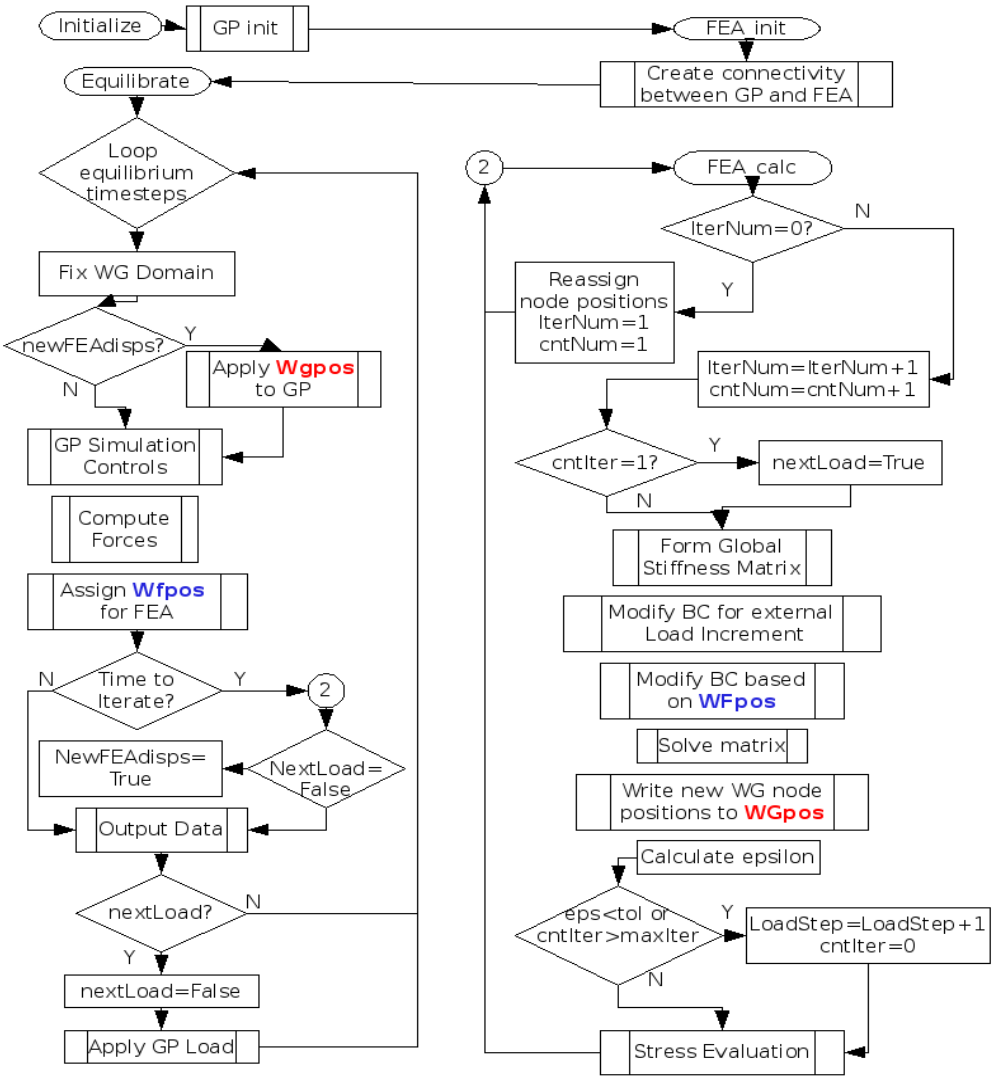


Figure 59. FEA-GP V29f10 Flowchart revision 3.

3. Using local/scale BoxSizes for Verlet neighbor Lists to save memory in large models

Mentioned briefly in Appendix A.4 each scale has its own BoxSize that is used to generate the Neighbor list cells used only for that scale. This boxsize can be any size as long as it is larger than the scale domain. The smaller this boxsize is, the less memory is required for simulation. Prior to this feature, the entire model was used to generate the neighbor list cells, and all cells were sized for the atomic domain. This caused problems when the model size grew to the micron scale, where the atomic domain was very small and did not occupy a very large volume. As of Version 43 of the GP code, the determination of the scale boxsize is automatically calculated based on the maximum position in each direction for each scale. The user does not need to manually determine the boxsize. To prevent the scales from drifting outside of their respective boxsize, the scale's boxsize is recalculated before the neighbor lists are updated. There is a small subroutine in the GP code called *FindBox* that will redefine the scale's box size. This subroutine is called first before the simulation begins on line 940, and before neighbor list updates on line 3530. The subroutine is on line 5054. This change is box size does not change the particle positions, because they are all normalized to the global box size not the scale box size and the global box size is not redetermined in this subroutine.

C. MODEL DEVELOPMENT

The GP simulation code reads the model to be simulated from a file specified in the input file. This file contains all of the coordinates of each atom and particle, what scale it is and what kind of element it is. To generate these models, a material generator is used which reads an input file usually called "model.in" and outputs the model file called "Model.MD" to be used directly by the GP code. When coupling GP to an FEA mesh, a separate input file is needed for the FEA mesh data, these data files are discussed and explained in detail.

1. GP model development program

There are three improvements that were made to the material generator program for efficiency, cleanliness, readability, and configurability.

Source code:	Mater_Model7.f90	# Version 7
Directory:	/shared/DATA/crystals/	

All of the main algorithms in this code came directly from the previous code: "Mater_Multi_2010_4.f90". The improvements in the program are listed below.

1) Only the size of each scale is used when generating the lattice for that respective scale. In other words, lattices do not fill the entire model domain during development, which saves computation time and resources. This also makes it possible to create very large scale GP domains up to the micron scale without suffering from computation lagging.

2) Many processing steps were consolidated and only one temporary file is used. This way, the data remains in memory saving computation time otherwise spent writing to disk. This also makes the code cleaner and more fluent.

3) Previously, all crystal structure data was manually included into the code of the program. This means that a user must modify the source code to add new crystal data. Having the user modifying code is prone to cause errors.

This program uses external crystal data files to assist in crystal structure definition and allows for any number of crystals in any orientation. These crystal files can be located in the same "crystals" directory as the program code. For example FCC aluminum has a crystal file called AlFCC and inside of it is:

```
4.0559          a0
1.0             a
1.0             b
1.0             c
.false.         hexagonal?
1.0 0.0 0.0     Base vectors
0.0 1.0 0.0
0.0 0.0 1.0
1               number of species/different elements
Al 13           element, atomic number
4               ncell, number of atoms in unit cell
0.0 0.0 0.0 13  atom vector and atomic number for each specie cell
0.5 0.5 0.0 13
0.5 0.0 0.5 13
0.0 0.5 0.5 13
```

All of the crystal files follow this format. As a result the input file for the material generator is slightly different. Instead of specifying the material for a block as a single letter it is now the name of the material file. For example an input file for a two scale block of Copper:

```
2  2                    !The number of scales used, GP Scale ratio
177.821 10.887 10.887   !Half model sizes
4                       !number of blocks below
CuFCC   2       1       !material structure,  scale,  number of cuts
-177.821 177.821    -10.887 10.887    -10.887 10.887
```

```
s
-127.015 127.015    -10.887 10.887    -10.887 10.887
CuFCC   1       1
-127.015 127.015    -10.887 10.887    -10.887 10.887
s
127.015 -127.015     10.887 -10.887    10.887 -10.887
CuFCC   -1      1
-130.644 130.644    -10.887 10.887    -10.887 10.887
s
-127.015 127.015    -10.887 10.887    -10.887 10.887
CuFCC   -2       1
-127.015 127.015    -10.887 10.887    -10.887 10.887
s
-119.757 119.757    -10.887 10.887    -10.887 10.887
```

The Model Size is measured from the center of the model, so these values are half-model size values. This means that the model must be centered about the origin to correctly use periodic boundary conditions, also be sure that the largest scale is thick enough in the periodic direction to accommodate two times the interatomic cutoff radius. Also be sure that atomic layers are not being duplicated; in other words a structure that is stacked with the sequence ABABAB does not end with the same type of layer that it begins with, this would make the first layer and the last layer interact directly across the periodic boundary and violate the stacking sequence.

The number of material blocks refers to how many special material domains are listed. For example, it would be 2 if the model was half Al2O3 and Fe. It can also help when building irregularly shaped models and multiple scales.

The scale should be 1 to use atoms, and third is the number of cuts being made in this block. The next line is the full domain of the block in a, x-left, x-right, y-back, y-front, z-lower, z-upper, format. The single letter below this specifies the type of cut to be made (all blocks must have at least one cut). Cut types and their parameters can be seen in the table below. If the lower bound parameter for a given dimension is LARGER than the maximum bound parameter, it will invalidate the cut. In this way, it is possible to circumvent the mandatory one-cut-per block, to allow for a solid block defined by the material block domain. For example, to cut a cylindrical hole down the Y direction, use:

```
c
4.5   6.0   2.0
```

Where the first two parameters are the X and Z coordinates and the third is the radius of the cylinder. The cut-types shown in **bold** are not *cuts* per se, they just modify the lattice within or without the domain.

Cut	type	parameters
Cylinder	c	x-center, z-center, radius of cut
Ellipse	e	x-center, y-center, A-major-axis, B-minor-axis, angle
Pore/Sphere	P,w	x-center, y-center, z-center, radius of pore
Tube	T	x-center, y-center, outer radius, inner radius
Perimeter	p	x-left, x-right, y-back, y-front, z-lower, z-upper, depth, material angle
Rod	r	x-center, y-center, A-major-axis, B-minor-axis, angle,material angle
perimeter	v	x-left, x-right, y-back, y-front, z-lower, z-upper, depth
Rectangular	s	x-left, x-right, y-back, y-front, z-lower, z-upper
Mirror	m	{x,y,z} x-left, x-right, y-back, y-front, z-lower, z-upper

There is a new cut type with identification letter ‘p’ this is a “perimeter” cut type meaning that the first six parameters are the same as a typical rectangular cut, except that the seventh parameter specifies a thickness in the x and y direction. This rectangular perimeter is the material left over. The central part is what is actually cut. The eighth parameter is the angle in radians from the x axis in the XoY plane to rotate the material, this is just the material's orientation, it doesn't rotate the domain. Care must be taken when rotating, since the rotation is made before the cutting, the overall material domain must be able to accommodate the rotation, otherwise the corners may appear to be cut.

The cut type 'v' for perimeter does the inverse of 'p', it actually cuts within the perimeter domain, rather than only leaving the perimeter as material.

The “Mirror” cut is used to make twin boundaries. It reverses, or mirrors, the lattice within the domain and is used, for example, as:

```
m
y
-100.0 100.0 -25.0 25.0 -100.0 100.0
```

This will reverse the lattice across the Y axis following the formula:

```
y = cdom(4)-abs(cdom(3)-y)
```

where cdom(4) is shown above to be 25.0 and cdom(3)=-25.0, for every lattice point, y coordinate.

The cut type “Rod” is designed to be used with the cut type “Ellipse” as the ellipse will cut an elliptical hole and the rod can be used to fill the ellipse with a different material and orientation.

It is important to note that the block and model dimensions should be designed as multiples of the unit-cell lattice constants, a, b, and c. These values can be found in the material files (e.g. AlFCC).

It is important that the imaginary scales, denoted as the negative of their real counterpart, correctly overlap into a real domain. Scales must also be sufficiently large enough to reduce any scale boundary effect.

An example to run this program to generate a model is shown below.

```
$       /shared/DATA/crystals/Mater_Model7.exe  model.in_example
```

The first part is the program executable to run and the first and only parameter is the name of the “model.in” file to be generated from. This program generates a file called: “Model.MD” which can be used immediately for simulation. It is best to rename this file to be more descriptive, such as: “Model.MD_example”. It also makes a file called “Model.xyz” which can be viewed directly with VMD.

a. Auto-Scale Interface Creation

A new version of the model development code was created that will automatically create appropriate scale interfaces given only the inter-atomic cutoff radius. This new code is "Mater_Model8.f90". The main advantage of this code is to make GP model development easy for the user; all that the user needs to do is to define the *real* atom and particle domains. For example the "model.in" input file described before could reduce to the following:

```
2  2                    !The number of scales used, GP Scale ratio
177.821 10.887 10.887   !Half model sizes
5.3                     !Cutoff radius
4                       !number of blocks below
CuFCC   2       1       !material structure,  scale,  number of cuts
-177.821 177.821    -10.887 10.887    -10.887 10.887
s
-127.015 127.015    -10.887 10.887    -10.887 10.887
CuFCC   1       1
-127.015 127.015    -10.887 10.887    -10.887 10.887
s
127.015 -127.015     10.887 -10.887    10.887 -10.887
```

Notice that the third line is new, this is to define the depth of the W_{nimage} domain from the interfaces which should be the same size as the cutoff radius used. Also notice that the previously defined imaginary domains are simply removed from the input file, see how much more simple this file appears!

This automatic capability is based on image particle proximity to real particles of the same scale. Specifically, for every real domain that the user specifies, a corresponding imaginary particle scale is made that has a scale *n+1* and *n-1* of the real scale, *n* (unless it doesn't make sense, if *n=1* then *n-1* is not used, and if *n* is the number of scales then *n+1* is not used). This follows the scale-duality principal described in Section II.A.4. In this example, when the scale-2 domain is generated a corresponding imaginary atomic domain is also made in the same location. Then when the real atomic domain is generated a corresponding imaginary scale-2 domain is made in the same location. At this point the scale interface is already made, however it is grossly over designed. The imaginary

particles in the center and far from the interface are superfluous. Thus after this type of model is made it is filtered into an *image removal routine* that will only keep the imaginary particles and atoms that are within a cutoff radius of a real atom or particle of the same scale. In this way all superfluous images are removed. This routine increases the computational cost and time of model development, but with the utilization of Verlet Neighbor Lists the speed is greatly increased.

The drawback of this automatic scale interface creation is that specific imaginary domains that would be needed for auto-decomposition (see Sections II.A.4.a and Appendix B.1) would be automatically removed since they are not a part of the scale interface. For this situation either the user could use the older version 7 of the code or they could add an imaginary domain to the model separately, outside of the program. The latter process would require two "model.in" files, one for the main GP model and the other for the imaginary domain. The imaginary domain would need to be made as a real domain first, to prevent it from being automatically removed, and then after it's made it should be converted to images with the utility "chScale.exe". This model should then be appended to the GP model. If the user is knowledgeable about proper scale interface design then the former approach is recommended as it is more concise, but for users uncomfortable with scale interface design the latter method is recommended.

2. Nano-structure generation

There is a way to create a simple nanostructure. The easiest way is to cut many holes in a single material crystal and fill the holes with a different material and/or orientation. These holes can be of different geometric shapes randomly distributed and not intersecting. These shapes are ideally elliptical in the XoY plane and extend through the z direction, like a cylinder. In order for them not to intersect, the larger primary axis, a, is assumed to be the radius of a circle for intersection detection. This is an overestimation, meaning that the volume of the shapes cut into the matrix can only easily reach about 50% of the model.

The program to conduct this operation is called: nanoStructGen.f90 it is not an ideal program, meaning that the user must change specific parameters inside of the code for each application. The beginning of the code file has several modules for domain tools

and randomization. At about 85% of the total lines of code begins the actual program. To start, line 521 and 522 specify the volume percent of the model to be cut/filled with nanograins and the other line is the domain array of the model box, as xmin, xmax, ymin ymax, zmin, zmax.

```
PercentVolume=20.0
boxsize=(/-235.2422, 235.2422, -219.0186, 219.0186,-24.82, 24.72/)
```

On line 533-536 are specified the ranges for the a and b axis for the cutting and plugging of the holes. Random values will be chosen within these ranges.

```
mina=5.0
maxa=30.0
minb=5.0
maxb=25.0
```

On line 540-547 define two domains, elliptical shown below, these domains will be avoided when creating the nanostructure. This is useful when generating a bimodal material. These two domains would be two large elliptical grains, clearly Not part of the nanostructure being generated. The variable: e1 and e2 stand for ellipse 1 and 2. These are specially defined variables or objects, as such they have subvariables associated with each object. The ‘ct’ variable of the e1 object, defined as “e1%ct" is for cut type or the shape of the domain. This is a three letter string, “ell" for ellipse and “rec" for rectangular. The ‘axis’ variable, defined as “e1%axis" specifies the direction or plane of the ellipse, here 3 is the z direction meaning that the ellipse is in the XoY plane. The ‘inside’ variable is a logical or boolean type which determines whether the volume defined by the domain is inside or outside. The last variable, 'parm’ of the “e1“ and “e2“ objects is an array that holds the domain values for the specific domain geometry. For ellipses there is needed six values, the x and y coordinates to locate the center of the ellipse, the a and b values of the ellipse, and the angle in radians from the x axis to rotate the ellipse, and the same angle but for the rotation of the material.

```
e1%ct="ell"
e1%axis=3
e1%inside=.true.
e1%parm=(/-96.5,96.5,124.,124.,0.0,0.0,0.0/)
```

```
e2%ct="ell"
e2%axis=3
e2%inside=.true.
e2%parm=(/96.5,-96.5,124.,124.,0.0,0.0,0.0/)
```

On lines 550-552 specifies the domain type for the nanostructures being cut into the crystal matrix.

```
dom(i)%ct="ell"
dom(i)%axis=3
dom(i)%inside=.true.
```

The parameters for these domains are randomly distributed and checked for interaction. For these domains to be useful in model development the cuts into the matrix are output to standard output as a format compatible for “model.in” files. The material to be placed inside of the cuts is output on standard error.

A shell script was designed to more easily use this program to generate a “model.in” file for nanostructure generation. It is simply called “nanoStructGen.sh” and it is listed below:

```
echo "running for 30 seconds..."    #to maximize the volume to the requested
./nanoStructGen.exe 2>err >out & sleep 30s ; kill $!
echo `tail -n2 err | head -n1 | awk '{print $2}'` " percent nanostructured"

echo "Generating the model.in file"
lines=`wc -l out | cut -d " " -f1`
echo "5 3             This value is not used
10582.164 9112.419 587.898  Half model sizes
T T T                   Periodic boundaries in X Y Z
2                          number of blocks below
CuFCC   5       $((lines/2))
-10582.164 10582.164 -9112.419 9112.419 -587.898 587.898" >model.in
cat out >>model.in         #append the cuts to the model.in
lines=`wc -l err | cut -d " " -f1`
echo "CuFCC   5       $((lines/2))
-10582.164 10582.164 -9112.419 9112.419 -587.898 587.898" >> model.in
cat err >>model.in         #append  the plugs to the model.in

echo "Generating the Model.MD file" #run the new model generator
/shared/DATA/crystals/Mater_Model5.exe model.in
```

```
echo "Removing Particles that are too close"
delMDdup.exe 1.0 1 -10582.164 10582.164 -9112.419 9112.419 -587.898 587.898
<Model.MD \ 2>ModelDups.MD >ModelNew.MD
```

The last part is very important, the part about removing the duplicate particles. This is to make sure that the material used to plug the holes cut is not too large for the hole. This program is explained in detail in Appendix C.4.

3. External Decomposition (blind) versions: 2W 3, 5, 6one

External decomposition is performed after a GP simulation to increase the particle resolution in a particular area; this could be to follow the tip of a crack or dislocation propagation. The concept is similar to Auto-Duality discussed in Appendix B.1.

These 'decomp' programs have a duplicate checking routine that is run before any particle is placed in output except for the last version: 6one. This routine searches through all of the currently outputted positions for a possible duplicate. The last version, 6one must have an external duplicate check.

a. decomp2W.f90

This program decomposes a specified domain using shape functions in combination with a deformation gradient as a means of interpolating lower scale positions from higher scale particle positions. The details of how this is done have been explained in previous works.

There are two parameters defined in the code, the first is, a0, the lattice constant for an FCC material such as Copper or Aluminum, the other is, duptol, this is the duplicate tolerance. Meaning that if there is a particle/atom closer than this value to another, their positions will be averaged together to make a single point.

The program takes three arguments. The first is the initial Model.MD file of the simulation, the second is a “current” snapshot of the configuration in an MD3 format and the third is the output; the decomposed model. By default this program will reduce all domains by one scale. So if there is a three scale model, it will keep S1 the same, reduce S2 to S1 and S3 into S2. Needless to say the output has a significantly larger Degree of Freedom. An example command line is shown below:

```
$    ./decomp2W.exe  Model.MD  0001200.MD3  decomped.MD3
```

The program, when executed, will ask, “Decompose a specified local region? (y/n) “. If the user responds with the letter, y, then it will ask, “Please specify box coordinates: xmin xmax ymin ymax zmin zmax”. The user is requested to input a rectangular domain for the local decomposition. Otherwise the entire model domain is used for decomposition.

This program assumes that the scale ratio is equal to two (k=2) and it is designed only for scales two and three. It begins by identifying the eight octants or unit cells around a particle. This octant is used as an element using hexahedral shape functions to use with the deformation matrix to interpolate decomposition points.

b. decomp3.f90

Like the program above, this decomposes a specified domain, or the entire model, using only interpolation or shape functions to estimate lower scale particle positions without the use of the deformation gradient.

Version 3A is able to decompose scale three directly to scale one, and version 3B is able to decompose surfaces by extrapolation rather than the usual bulk interpolation.
This version 3 has an additional feature; it has the option to neglect all scales except the atomic and one of choice. This choice scale may be specified as an optional fourth parameter.

Other than these changes this program is executed in the same way as “decomp2W.f90“

c. decomp5.f90

Regular Inverse Mapping is used to gather information, such as positions and other properties, from high scale particles for use in equations and data processing procedures that are confined to use input at the atomic scale. In other words, Inverse mapping reduces the high values from particles into lower manageable values as if it was the atomic scale. Realizing this, one could draw the conclusion that Inverse Mapping

might be used to decompose particles into lower scales or atoms. This decomposition process is designed to increase the model's "data resolution" by interpolating positions for the particle's constituent atoms. In general, increasing resolution doesn't truly increase resolution; the best that can be done with the finite amount of data is to use an accurate interpolation scheme. Herein will be introduced and described the Enhanced Inverse Mapping Method designed to be used for particle decomposition.

Inverse Mapping uses a one dimensional scheme involving only two particles at a time, and divides that distance by the particle's scaling factor to be used to represent an atom's placement. This would work fine for a perfect crystalline structure, but is too inaccurate to be used with a model under any deformation. The proposed method uses not two, but three particles along a line. For decomposition of a particle 'i', being the first particle in a line, with the second, 'j', being a nearest or next nearest neighbor, the third, 'k', a particle beyond the second by generally the same distance. When finding these particles the initial configuration is used for the simplicity of using a regular structure. Once the particles are defined, their positions at some other time and possibly with some deformation are used.

This method assumes that the strain gradient of the particle domain is the same for the decomposed domain. The principal used is that of proportions, the distance-ratio from particle 'i' to 'j' and the distance from 'i' to 'k' is the same as from particle 'i' to the atom, 'a' to be placed, and from 'i' to 'j', see Fig. 60.

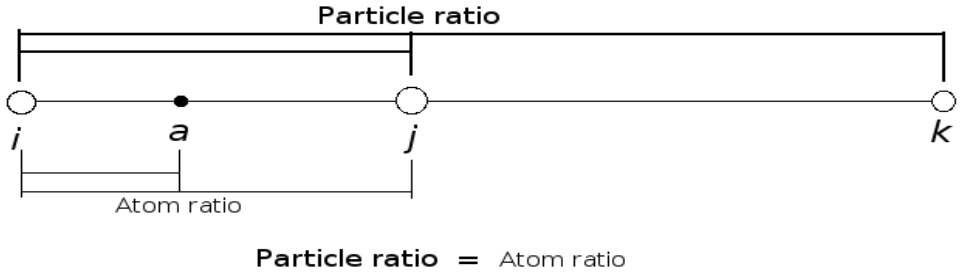


Figure 60. Distance ratios are the same for the atom 'a' as for the particles.

The implementation of this enhanced method is in the developed decomposition code `decomp5.f90`.

Its usage is also the same as “decomp2W.f90”. This decomposition technique only works when the scale ratio is equal to two (k=2).

d. decomp6one.f90

This program uses Inverse Mapping for General Particle decomposition like version 5 but without initial configuration data and only uses particle pairs for interpolation rather than the three of version 5.

This program has three optional parameters but the last one must be the “current” file to decompose. The options are to specify the search radius, -a, the scale ratio, -k, and the periodic boundary conditions, -p. Each of these three options take a parameter. An example command line can be seen below.

```
$    ./decomp6one.exe -a 6.3  -k 3  -p FFT   Equilib.MD3  >decomped.MD3
```

The result from this program does contain duplicates due to the structure and method of looping, so the resulting file should be filtered through the program “delMD3dup.exe”, see Appendix C.4.

For this method there are two ways to achieve decomposition when starting from a high scale model. First, a Fast-Decomposition where the scale is decomposed then the result is decomposed until the atomistic scale is reached. The disadvantage of this is that with every decomposition there is error introduced to the structure, so every scale decomposed from a decomposed scale just accumulates these errors, so much so that the resultant atomic scale may be in a state of disorder. Second, Let the model equilibrate, then decompose one scale, then equilibrate again before decomposing down a further scale. The included equilibration between decompositions in an effort to reduce the errors introduced from the decomposition.

There are two scripts that were made as an example of both ways. The fast decomposition script runs straight through since it decomposes from S5 all the way down to S1. The slow decomposition script just decomposes one scale at a time and is run four times to decompose a S5 model. The fast decomposition script was developed first and is more application specific, below is shown this script. The general layout of the scripts is

to define some helpful variables in the beginning for the location and geometry of the domains to decompose. Then there are subroutines defined for each scale to decompose to the next scale. For this fast decomposition all of these subroutines are run sequentially and depend on one another. After these subroutines the program actually begins by calling these decomposition routines. Then once all of the decomposed domains are created they are compiled into a Model.MD formatted file called: "decompedModel.MD" for simulation. This script takes one parameter, which is the configuration file to be decomposed. For example:

```
$ ./FastDecomp.sh  Equilib.MD3
```

Below is listed the contents of the "FastDecomp.sh" file.

```
#!/bin/bash

#axis direction, central coordinates for left most scales and angle
#xy1="3 -4222.2 2000.0"
xy1="3 -2777.7 -500.0"
#th1="0.610865"  # 35 degs
th1="1.57079632675"  # 90 degs

#axis direction, central coordinates for center most scales
xy2="3 -666.6 666.6"
th2="0.785398"  # 45 degs

# the a and b for the ellipse and the height range
                       # Volume Ang^3        est DOF    ~DOF in file
S4abz="1440.0 720.0 -601.7464 601.7464"   # 3920020675.33609    16668 S4
<33336
Si4abz="1602.0 882.0 -601.7464 601.7464"  # 5342253176.60646    22716
S4all=45432

S3abz="468.0 234.0 -234.0 234.0"        # 161011700.305603      18485 S3
<36970
Si3abz="522.0 288.0 -288.0 288.0"       # 272041616.538188      31232
S3all=62464

S2abz="144.0 72.0 -72.0 72.0"           # 4690372.69893427      14539 S2
<29078
```

```
Si2abz="162.0 90.0 -90.0 90.0"          # 8244795.7598454       25557
S2all=51114

S1abz="36.0 18.0 -18.0 18.0"            # 73287.073420848       6133  S1
=12266
Si1abz="42.0 24.0 -24.0 24.0"           # 152002.818946944      12722
S1all=25444
                                    # over estimation=257583 DOF
file="$1"
echo "Decomposing from file: " $file

# Scale 5 ##########################################3
function makeS5 () {
echo "extract only the final S5 part"
domainExtract.exe ell F $xy1 $S4abz $th1 <"$1" | \
 domainExtract.exe ell F $xy2 $S4abz $th2 >S5
echo "Total Real Scale 5: " $((`wc -l S5 | cut -d " " -f1`-1))
echo "extract only the imaginary S-5 part (in the S4 domain)"
domainExtract.exe ell T $xy1 $S4abz $th1 <"$1" | \
 domainExtract.exe ell F $xy1 $S3abz $th1 | sed 's/\ \ 5\ \ \ /\ -5\ \ \ /' >Si5
domainExtract.exe ell T $xy2 $S4abz $th2 <"$1" | \
 domainExtract.exe ell F $xy2 $S3abz $th2 | sed 's/\ \ 5\ \ \ /\ -5\ \ \ /' |
tail -n +2 \ >>Si5
echo "Total imaginary Scale 5: " $((`wc -l Si5 | cut -d " " -f1`-1))
}

# Scale 4 ##########################################
function makeS4 () {
echo "extract all scale 5 particles in the S4 and S-4 domain to decompose"
domainExtract.exe ell T $xy1 $Si4abz $th1 <"$1" >S4s
domainExtract.exe ell T $xy2 $Si4abz $th2 <"$1" | tail -n +2 >>S4s
echo "decompose Scale 5 to Scale 4; PBC in Z"
decomp6one.exe -a 6.3 -k 3 -p FFT S4s >S4tooMany
#rm S4s         # remove temporary file
echo "check for duplicate Scale 4 particles"
delMD3dup.exe $((`wc -l S4tooMany | cut -d " " -f1`-1)) 1.5 0 <S4tooMany
2>S4dups >S4all
echo "create only S4 domain"
domainExtract.exe ell T $xy1 $S4abz $th1 <S4all | \
 domainExtract.exe ell F $xy1 $S3abz $th1 >S4
domainExtract.exe ell T $xy2 $S4abz $th2 <S4all | \
```

```
 domainExtract.exe ell F $xy2 $S3abz $th2 | tail -n +2 >>S4
echo "Total Real Scale 4: " $((`wc -l S4 | cut -d " " -f1`-1))
echo "create outer S-4 domain"
domainExtract.exe ell F $xy1 $S4abz $th1 <S4all | \
 domainExtract.exe ell F $xy2 $S4abz $th2 |  sed 's/\ \ 4\ \ \ /\ -4\ \ \ /'
>Si4
echo "create inner S-4 domain"
domainExtract.exe ell T $xy1 $S3abz $th1 <S4all | \
 domainExtract.exe ell F $xy1 $S2abz $th1 | sed 's/\ \ 4\ \ \ /\ -4\ \ \ /' |
tail -n +2 \ >>Si4
domainExtract.exe ell T $xy2 $S3abz $th2 <S4all | \
 domainExtract.exe ell F $xy2 $S2abz $th2 | sed 's/\ \ 4\ \ \ /\ -4\ \ \ /' |
tail -n +2 \ >>Si4
echo "Total imaginary Scale 4: " $((`wc -l Si4 | cut -d " " -f1`-1))
}

# Scale 3 ########################################
function makeS3 {
echo "extract all scale 4 particles in the S3 and S-3 domain to decompose"
domainExtract.exe ell T $xy1 $Si3abz $th1 <S4all >S3s
domainExtract.exe ell T $xy2 $Si3abz $th2 <S4all |tail -n +2 >>S3s
echo "decompose Scale 4 to Scale 3"
decomp6one.exe -a 6.3 -k 3 S3s >S3tooMany
#rm S3s         # remove temporary file
echo "check for duplicate Scale 3 particles"
delMD3dup.exe $((`wc -l S3tooMany | cut -d " " -f1`-1)) 1.5 0 <S3tooMany
2>S3dups >S3all
echo "create only S3 domain"
domainExtract.exe ell T $xy1 $S3abz $th1 <S3all | \
 domainExtract.exe ell F $xy1 $S2abz $th1 >S3
domainExtract.exe ell T $xy2 $S3abz $th2 <S3all | \
 domainExtract.exe ell F $xy2 $S2abz $th2 | tail -n +2 >>S3
echo "Total Real Scale 3: " $((`wc -l S3 | cut -d " " -f1`-1))
echo "create outer S-3 domain"
domainExtract.exe ell F $xy1 $S3abz $th1 <S3all | \
 domainExtract.exe ell F $xy2 $S3abz $th2 |  sed 's/\ \ 3\ \ \ /\ -3\ \ \ /'
>Si3
echo "create inner S-3 domain"
domainExtract.exe ell T $xy1 $S2abz $th1 <S3all | \
 domainExtract.exe ell F $xy1 $S1abz $th1 | sed 's/\ \ 3\ \ \ /\ -3\ \ \ /'
|tail -n +2 \ >>Si3
```

```
domainExtract.exe ell T $xy2 $S2abz $th2 <S3all | \
 domainExtract.exe ell F $xy2 $S1abz $th2 | sed 's/\ \ 3\ \ \ /\ -3\ \ \ /'
|tail -n +2 \ >>Si3
echo "Total imaginary Scale 3: " $((`wc -l Si3 | cut -d " " -f1`-1))
}

# Scale 2 ########################################
function makeS2 {
echo "extract all scale 3 particles in the S2 and S-2 domain to decompose"
domainExtract.exe ell T $xy1 $Si2abz $th1 <S3all >S2s
domainExtract.exe ell T $xy2 $Si2abz $th2 <S3all |tail -n +2 >>S2s
echo "decompose Scale 3 to Scale 2"
decomp6one.exe -a 6.3 -k 3 S2s >S2tooMany
#rm S2s         # remove temporary file
echo "check for duplicate Scale 2 particles"
delMD3dup.exe $((`wc -l S2tooMany | cut -d " " -f1`-1)) 1.5 0 <S2tooMany
2>S2dups >S2all
echo "create only S2 domain"
domainExtract.exe ell T $xy1 $S2abz $th1 <S2all | \
 domainExtract.exe ell F $xy1 $S1abz $th1 >S2
domainExtract.exe ell T $xy2 $S2abz $th2 <S2all | \
 domainExtract.exe ell F $xy2 $S1abz $th2 |tail -n +2 >>S2
echo "Total Real Scale 2: " $((`wc -l S2 | cut -d " " -f1`-1))
echo "create outer S-2 domain"
domainExtract.exe ell F $xy1 $S2abz $th1 <S2all | \
 domainExtract.exe ell F $xy2 $S2abz $th2 |  sed 's/\ \ 2\ \ \ /\ -2\ \ \ /'
>Si2
echo "create inner S-2 domain"
domainExtract.exe ell T $xy1 $S1abz $th1 <S2all | sed 's/\ \ 2\ \ \ /\ -2\ \ \
/' |tail -n \ +2 >>Si2
domainExtract.exe ell T $xy2 $S1abz $th2 <S2all | sed 's/\ \ 2\ \ \ /\ -2\ \ \
/' |tail -n \ +2 >>Si2
echo "Total imaginary Scale 2: " $((`wc -l Si2 | cut -d " " -f1`-1))
}

# Scale 1 ########################################
function makeS1 {
echo "extract all scale 2 particles in the S1 and S-1 domain to decompose"
domainExtract.exe ell T $xy1 $Si1abz $th1 <S2all >S1s
domainExtract.exe ell T $xy2 $Si1abz $th2 <S2all |tail -n +2 >>S1s
echo "decompose Scale 2 to Scale 1"
```

```
decomp6one.exe -a 6.3 -k 3 S1s >S1tooMany
#rm S1s         # remove temporary file
echo "check for duplicate Scale 1 particles"
delMD3dup.exe $((`wc -l S1tooMany | cut -d " " -f1`-1)) 1.5 0 <S1tooMany
2>S1dups >S1all
echo "create only S1 domain"
domainExtract.exe ell T $xy1 $S1abz $th1 <S1all >S1
domainExtract.exe ell T $xy2 $S1abz $th2 <S1all |tail -n +2 >>S1
echo "Total Real Scale 1: " $((`wc -l S1 | cut -d " " -f1`-1))
echo "create outer S-1 domain"
domainExtract.exe ell F $xy1 $S1abz $th1 <S1all | \
 domainExtract.exe ell F $xy2 $S1abz $th2 |  sed 's/\ \ 1\ \ \ /\ -1\ \ \ /'
>Si1
echo "Total imaginary Scale 1: " $((`wc -l Si1 | cut -d " " -f1`-1))
}

# main program starts here
############################################################
# These are subroutines
makeS5 $file
makeS4 $file
makeS3
makeS2
makeS1
#
## making the Model.MD file ##############################
# calculate number of reals and imaginary
NReal=$(( `wc -l S1 | cut -d " " -f1`-1+`wc -l S2 | cut -d " " -f1`-1+\
`wc -l S3 | cut -d " " -f1`-1+`wc -l S4 | cut -d " " -f1`-1+`wc -l S5 | cut -d
" " -f1`-1))
NIdeal=$(( `wc -l Si1 | cut -d " " -f1`-1+`wc -l Si2 | cut -d " " -f1`-1+\
`wc -l Si3 | cut -d " " -f1`-1+`wc -l Si4 | cut -d " " -f1`-1+`wc -l Si5 | cut
-d " " \ -f1`-1))
# write the head of the MD file
echo "NTotal $((NReal+NIdeal))  NIdeal $NIdeal"
echo "   5   $((NReal+NIdeal))  $NReal" >decompedModel.MD
echo -e "  F  \c" >>decompedModel.MD   #"
cat S5 >>decompedModel.MD
tail -n +2 Si5 | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
tail -n +2  S4 >> decompedModel.MD
```

```
tail -n +2 Si4 | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
tail -n +2  S3 >> decompedModel.MD
tail -n +2 Si3 | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
tail -n +2  S2 >> decompedModel.MD
tail -n +2 Si2 | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
tail -n +2  S1 >> decompedModel.MD
tail -n +2 Si1 | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
############################################################
```

The slow decomposition file works in generally the same way but only decomposes down one scale specified at a time. For example if decomposing from an S5 model down one scale to S4 the command line would be:

```
$   ./SlowDecomp.sh  4  Equilib >decompedModel.MD
```

Below is the SlowDecomp.sh file.

```
#!/bin/bash
#
################################################################################
###
#
#  Script to automatically decompose a higher scale domain to the one lower.
#  Must specify the decomposition domains i.e. the scale domains to be created
#  Must specify the scale to decompose TO and the file to decompose in MD3
format
#
#  Output is in Model.MD format, ready for simulation
#
#  Usage: ./SlowDecomp.sh 4 Equilib.MD3 2>err >out &
#
#  Author: Ross J. Stewart
#  Date: Friday, November 30, 2012
#
################################################################################
###
```

```
#axis direction, central coordinates for scales and angle
xy="3 -666.6 666.6"
th="0.785398"  # 45 degs

# the a and b for the ellipse and the height range (Vol=pi*a*b*h)
                      # Vol/11.9480285290791*k^3*(scale-1) = CuFCC_DOF
                       # Volume Ang^3        est DOF    ~DOF in file
                       #                           S5    72970

S4abz="1498.0 778.0 -601.7464 601.7464"   # 4406408734.64214    18736 S4   <
Si4abz="1660.0 940.0 -601.7464 601.7464"  # 5899691610.69566    25086 S4all=

S3abz="526.0 292.0 -292.0 292.0"        # 281793723.328496      32352 S3   <
Si3abz="580.0 346.0 -346.0 346.0"       # 436274731.095901      50088 S3all=

S2abz="202.0 130.0 -130.0 130.0"        # 21449538.0016497      66490 S2   <
Si2abz="220.0 148.0 -148.0 148.0"       # 30277916.0130616      93856 S2all=

S1abz="94.0 76.0 -76.0 76.0"            # 3411417.76342131      285521 S1   =
Si1abz="100.0 82.0 -82.0 82.0"          # 4224813.80054755      353599 S1all=
                               # over estimation=595599 DOF
tol="1.5"       # duplicate tolerance
k="3"           # Scale ratio, k, as in k^3*(scale-1)
ra="6.3"        # Decomposition radius used for version 6one of the
decomposition code

scale="$1"      # first parameter: scale to decompose TO
file="$2"       # second parameter: File to decompose
echo "Decomposing from file: " $file

# Scale 5 to 4 ##############################################3
function S5_4 () {
echo "extract only the external final part"
domainExtract.exe ell F $xy $S4abz $th <"$1" >upper
echo "Total Real Upper Scales: " $((`wc -l upper | cut -d " " -f1`-1))
echo "extract only the imaginary S-5 part (in the S4 domain)"
domainExtract.exe ell T $xy $S4abz $th <"$1" | \
 domainExtract.exe ell F $xy $S3abz $th | sed 's/\ \ 5\ \ \ /\ -5\ \ \ /'
>SiUpper
echo "Total imaginary Scale 5: " $((`wc -l SiUpper | cut -d " " -f1`-1))
```

```
echo "extract all scale 5 particles in the S4 and S-4 domain to decompose"
domainExtract.exe ell T $xy $Si4abz $th <"$1" >lowers
echo "decompose Scale 5 to Scale 4; PBC in Z"
decomp6one.exe -a $ra -k $k -p FFT lowers >lowertooMany
#rm lowers      # remove temporary file
echo "check for duplicate Scale 4 particles"
delMD3dup.exe $((`wc -l lowertooMany | cut -d " " -f1`-1)) $tol 0 <lowertooMany
2>lowerdups\
 >allLower
echo "create only S4 domain"
domainExtract.exe ell T $xy $S4abz $th <allLower >lower
echo "Total Real Scale 4: " $((`wc -l lower | cut -d " " -f1`-1))
echo "create outer S-4 domain"
domainExtract.exe ell F $xy $S4abz $th <allLower |  sed 's/\ \ 4\ \ \ /\ -4\ \
\ /' >SiLower
}

# Scale 4 to 3 ##########################################3
function S4_3 () {
echo "extract only the external final part"
domainExtract.exe ell F $xy $S3abz $th <"$1" >upper
echo "Total Real Upper Scales: " $((`wc -l upper | cut -d " " -f1`-1))
echo "extract only the imaginary S-4 part (in the S3 domain)"
domainExtract.exe ell T $xy $S3abz $th <"$1" | \
 domainExtract.exe ell F $xy $S2abz $th | sed 's/\ \ 4\ \ \ /\ -4\ \ \ /'
>SiUpper
echo "Total imaginary Scale 4: " $((`wc -l SiUpper | cut -d " " -f1`-1))

echo "extract all scale 4 particles in the S3 and S-3 domain to decompose"
domainExtract.exe ell T $xy $Si3abz $th <"$1" >lowers
echo "decompose Scale 4 to Scale 3; PBC in Z"
decomp6one.exe -a $ra -k $k -p FFT lowers >lowertooMany
#rm lowers      # remove temporary file
echo "check for duplicate Scale 3 particles"
delMD3dup.exe $((`wc -l lowertooMany | cut -d " " -f1`-1)) $tol 0 <lowertooMany
2>lowerdups\
 >allLower
echo "create only S3 domain"
domainExtract.exe ell T $xy $S3abz $th <allLower >lower
echo "Total Real Scale 3: " $((`wc -l lower | cut -d " " -f1`-1))
```

```
echo "create outer S-3 domain"
domainExtract.exe ell F $xy $S3abz $th <allLower |  sed 's/\ \ 3\ \ \ /\ -3\ \
\ /' >SiLower
}

# Scale 3 to 2 ##########################################3
function S3_2 () {
echo "extract only the external final part"
domainExtract.exe ell F $xy $S2abz $th <"$1" >upper
echo "Total Real Upper Scales: " $((`wc -l upper | cut -d " " -f1`-1))
echo "extract only the imaginary S-3 part (in the S2 domain)"
domainExtract.exe ell T $xy $S2abz $th <"$1" | \
 domainExtract.exe ell F $xy $S1abz $th | sed 's/\ \ 3\ \ \ /\ -3\ \ \ /'
>SiUpper
echo "Total imaginary Scale 3: " $((`wc -l SiUpper | cut -d " " -f1`-1))

echo "extract all scale 3 particles in the S2 and S-2 domain to decompose"
domainExtract.exe ell T $xy $Si2abz $th <"$1" >lowers
echo "decompose Scale 3 to Scale 2; PBC in Z"
decomp6one.exe -a $ra -k $k -p FFT lowers >lowertooMany
#rm lowers      # remove temporary file
echo "check for duplicate Scale 2 particles"
delMD3dup.exe $((`wc -l lowertooMany | cut -d " " -f1`-1)) $tol 0 <lowertooMany
2>lowerdups\
 >allLower
echo "create only S2 domain"
domainExtract.exe ell T $xy $S2abz $th <allLower >lower
echo "Total Real Scale 2: " $((`wc -l lower | cut -d " " -f1`-1))
echo "create outer S-2 domain"
domainExtract.exe ell F $xy $S2abz $th <allLower |  sed 's/\ \ 2\ \ \ /\ -2\ \
\ /' >SiLower
}

# Scale 2 to 1 ##########################################3
function S2_1 () {
echo "extract only the external final part"
domainExtract.exe ell F $xy $S1abz $th <"$1" >upper
echo "Total Real Upper Scales: " $((`wc -l upper | cut -d " " -f1`-1))
echo "extract only the imaginary S-2 part (in the S1 domain)"
domainExtract.exe ell T $xy $S1abz $th <"$1" | sed 's/\ \ 2\ \ \ /\ -2\ \ \ /'
>SiUpper
```

```
echo "Total imaginary Scale 2: " $((`wc -l SiUpper | cut -d " " -f1`-1))

echo "extract all scale 2 particles in the S1 and S-1 domain to decompose"
domainExtract.exe ell T $xy $Si1abz $th <"$1" >lowers
echo "decompose Scale 2 to Scale 1; PBC in Z"
decomp6one.exe -a $ra -k $k -p FFT lowers >lowertooMany
#rm lowers      # remove temporary file
echo "check for duplicate Scale 1 particles"
delMD3dup.exe $((`wc -l lowertooMany | cut -d " " -f1`-1)) $tol 0 <lowertooMany
2>lowerdups\
 >allLower
echo "create only S1 domain"
domainExtract.exe ell T $xy $S1abz $th <allLower >lower
echo "Total Real Scale 1: " $((`wc -l lower | cut -d " " -f1`-1))
echo "create outer S-1 domain"
domainExtract.exe ell F $xy $S1abz $th <allLower |  sed 's/\ \ 1\ \ \ /\ -1\ \
\ /' >SiLower
}

## Make the Model.MD file ##############################
function mkMD () {
grep ".      -.     " upper >>SiUpper        # Append all imaginary in
upper(MD3) to SiUpper
grep -v ".      -.     " upper >>upper.tmp   # extract all real in upper(MD3)
to a temp file
mv upper.tmp upper                              # rename the temp file to upper
NReal=$(( `wc -l upper | cut -d " " -f1`-1+`wc -l lower | cut -d " " -f1`-1))
NIdeal=$(( `wc -l SiUpper | cut -d " " -f1`-1+`wc -l SiLower | cut -d " " -f1`-
1))
echo "NTotal $((NReal+NIdeal))  NIdeal $NIdeal"
echo " `sort -r -k 4 -g $file | head -n1 | awk '{print $4}'`  $((NReal+NIdeal))
$NReal" \  >decompedModel.MD
echo -e "  F  \c" >>decompedModel.MD   #"
cat upper >>decompedModel.MD
tail -n +2 SiUpper | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
tail -n +2  lower >> decompedModel.MD
tail -n +2 SiLower | awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> decompedModel.MD
}
```

```
# MD3 to Model.MD ########################################
function MD3_MD () {
grep ".      -.     " $file >Ideal      # extract imaginary from MD3
grep ".       .     " $file >Real
NReal=`wc -l Real | cut -d " " -f1`
NIdeal=`wc -l Ideal | cut -d " " -f1`
echo "NTotal $((NReal+NIdeal))  NIdeal $NIdeal"
echo " `sort -r -k 4 -g $file | head -n1 | awk '{print $4}'`  $((NReal+NIdeal))
$NReal" \ >Model.MD_${file}
echo "  F  `head -n 1 $file`" >>Model.MD_${file}
cat Real>>Model.MD_${file}
awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' Ideal \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' > Model.MD_${file}
}

##############################################################################
# Main Program Starts Here
##############################################################################

if [ $scale = "4" ]; then
     S5_4 $file
     mkMD
elif [ $scale = "3" ]; then
     S4_3 $file
     mkMD
elif [ $scale = "2" ]; then
     S3_2 $file
     mkMD
elif [ $scale = "1" ]; then
     S2_1 $file
     mkMD
elif [ $scale = "rebuildMD" ]; then  # rebuild the Model.MD output
     mkMD
elif [ $scale = "makeMD" ]; then     # turn an MD3 file into a Model.MD
     MD3_MD $file
fi

##############################################################################
########
```

These two decomposition scripts rely heavily on the external program called, "domainExtract.exe" this is a simple but a very useful utility, and it is described in Appendix E.3.

4. Delete duplicate positions within a cutoff: delMDdup & delMD3dup

This utility is not always necessary but can be useful. It checks an input stream of positions in a domain for duplicate positions according to the given tolerance parameter. Its only parameter is the tolerance in the unit used in the data.

If another position is within this tolerance radius it is output on standard error, the pure results without duplicates is output on standard output. Images are ignored since they have NLCs. The only parameter to be aware of is kScale on line 12 specifying the scale ratio, this is important to make sure the duplicate tolerance is proportionately scaled.

There are two versions of this code, one for Model.MD formatted files and the other for MD3 configuration files. The latter requires an additional parameter for the total number of particles. Their respective usage is as follows:

```
$  delMDdup [dupTol] [numBox] [domains..] < dupFILE.MD > uniqFILE.MD"
$  delMD3dup [NTotal] [dupTol] [numBox] [domains..] < dupFILE.MD > uniqFILE.MD"
```

The parameters needed to execute these programs include the duplicate tolerance (dupTol), an example value is 1.5 Angstrom. These programs also have the ability to operate on local rectangular domains. The number of these domains is specified by "numBox" and are listed in the primitive format of "xmin xmax ymin ymax zmin zmax". The file to be rid of duplicates is read on standard input.

5. Considerations when designing very large micron scale models

The greatest concern when making very large models is the degree of freedom. To most effectively reduce the DOF is to have a large scale ratio. Theoretically this greatly reduces the local resolution and hence accuracy. A recommended scale ratio is k=3. In practice the majority of the micron sized models would best be S5. The domain of highest interest, such as a grain boundary or other interface, should be the atomistic domain. The

intermediate scales between 1 and 5 should be as small as possible, to quickly achieve the fifth scale.

These intermediate scales should ideally be greater than or equal to the size of the cutoff radius for the scale above, so that the higher scale's imaginary domain can fit inside when ideally sized. The Program to generate such a large model must be the model generation program described in Appendix C.1. The reason for this is due to the improved efficiencies in this code compared to the older traditional one which is still suitable for MD and small GP models.

These large models also tend to have large DOFs and are difficult to visualize in programs like VMD. Efficient ways to view the model is scale-by-scale. To extract a specific scale see the code described in Appendix E.3.

6. FE Mesh input file for the GP-FEA simulation.

The FE code currently used within the GP program requires exclusive use of quadrilateral elements, so before meshing the part, be sure that the element type is set to quadrilateral and not set to other types, e.g., not use mixed triangle and quadrilateral elements. What are required in the FEA input file are the FE nodal positions, element connectivity and the BCs. In this study Abaqus was used for mesh generation and a script entitled mkFEAinp.sh has been developed to convert the Abaqus input file xxx.inp into the simple FEA-GP mesh input file. Its format is shown below. Note, the comment lines with symbol "#" are important to show the meaning of the subsequent data and are required even if there are no parameters to list. After each "#" comment line, are generally several lines with numerical data whose meaning and line numbers are given in the previous comment lines. Specifically, the second comment line lists basic parameters in which variables `NN NE NM NDIM NEN NDN` are, respectively, Node Number (NN), Number of Elements (NE), Number of Materials (NM), Number of dimensions (NDIM), Number of Element Nodes (NEN) and Number of Dimensions per Node (NDN). Parameters related to load by displacement and force on the boundary is given in the third comment line where `ND NL NMPC` stand for Number of Displacements (ND), Number of Loads (NL), and Number of Multi-Point Constraints (NMPC). The comment lines of 4 and 5 are common in any FE input file, in turn; they give the node coordinates, element

connectivity conditions including materials and thickness. Line 6 gives the DOF of each node and its constraint condition at the displacement boundary, and line 7 present similar data for the load boundary. Comment line 8 gives the material property for Young's modulus, Poisson's ratio and alpha, the coefficient of thermal expansion for each type of material. Line 9 lists the parameters required for Multi-Point constrains. In this code, NMPC=0 since only displacement BC are applied there is no data used under this line. The last comment line lists the FEA tolerance value that the GP-FEA interface is to converge to and the second number (e.g. number 19) is the maximum allowed number of GP-FEA interface iterations, k_m, after which the model will load regardless of whether the interface converged or not. The third value is multiplied by the WG interpolated displacements as an effort to help accelerate convergence.

```
#

Title: FEA_GP02.inp
# NN NE NM NDIM NEN NDN
3628 3540 1  2   4   2
# ND NL NMPC
20   0   0
# Node#  Coordinates
1   -128.710007  -347.
...
3628 122.416016  -566.217102
# Elem# Nodes(1-4) Mat# Thickness TempRise
1     1   27 849  60  1 45.7    0.00
...
3540  3623  3617 3472 3507  1 45.7 0.00
# DOF# Displacement
37   0.0
38   1.0
51   0.0
52   1.0
40   1.0
...
1694  1.0
1696  1.0
```

```
# DOF# Load
# Mat# E   Nu  Alpha
1    224.0 0.3 12.e-6
# B1 I B2 j B3 MPC
# FEAtoll, MaxIter, kaccel
0.001       19       1.0
```

In this code, only displacement BC are applied, hence NL=0 and NMPC=0. The total number of DOF that are applied for the displacement BC is 20 (i.e., ND=20). They are ranged from 37, 38... 1696 (see lines after the 6th comment line) and they are all even numbers except for 37 and 51. Here odd and even numbers denote the X- and Y- DOF of the node. In the code we use the numbers 0.0 and 1.0 for whether the particular DOF is fixed or to be loaded. If an FE node DOF is not listed it is free to deform without constraint. The fact that only two DOF (i.e., 37 and 51) is zero indicates that only two boundary nodes are fixed along the X-direction. Furthermore, these constraint conditions are re-written from the finite element input file, say, FEAxx.inp. If performing uniaxial load, be sure to set the center top and bottom FE nodes to be fixed (i.e., put "0.0" after that DOF) to prevent rigid body motion in the X direction. For the example here, the X direction is fixed at DOF=37 and 51, which corresponds to the DOF of node 19 and 104.

D. THE PARTICLE-BASED MULTISCALE ANALYSIS PROGRAM (PMAP)

Chapter VII discusses the general structure and flow of the Particle-Based Multiscale Analysis Program (PMAP) to allow the reader to understand the concept of the program. This appendix, goes into the technical details related to the source code of PMAP; conceptually connecting the actual subroutines and functions to the procedures described in Chapter VII.

1. PMAP Structure: Subroutines and functions

Behind all of the processes and procedures described in section VII.B there are specific subroutines and functions called to perform these procedures. Conceptually connecting the subroutines to their respective process is the purpose of this section. The initialization process discussed in section VII.B.1 calls specific subroutines and functions diagrammed on the left of Fig. 61. As mentioned in section VII.B.3 the loading process uses a relaxation procedure almost identical to the equilibration process. This relaxation procedure is "Evolve_Sample" the equilibration procedure is "Evolve_Sample0", as can be seen from Fig. 61 both of these subroutines call most of the same subroutines. These mutually called subroutines, "Compute_Temperature", "Update_List", "Compute_Forces", "print_stresst", and "OutputResult" correspond to the flow procedures shown in Fig. 50. The subroutines that the "Evolve_Sample" or the load relaxation procedure calls that the equilibration process "Evolve_Sample0" does not call include "composition" and "FEA_calc". These subroutines correspond to the auto-duality and FEA mesh procedures.

Table I is a detailed list of the subroutines and functions of PMAP with a brief description. The main routines are shown in Fig. 61. Routines 1-9 are specifically used when parsing the input file to extract information such as local domains. Routines 44-55 are specific to the FEA calculations and are not used at all for pure GP simulations.

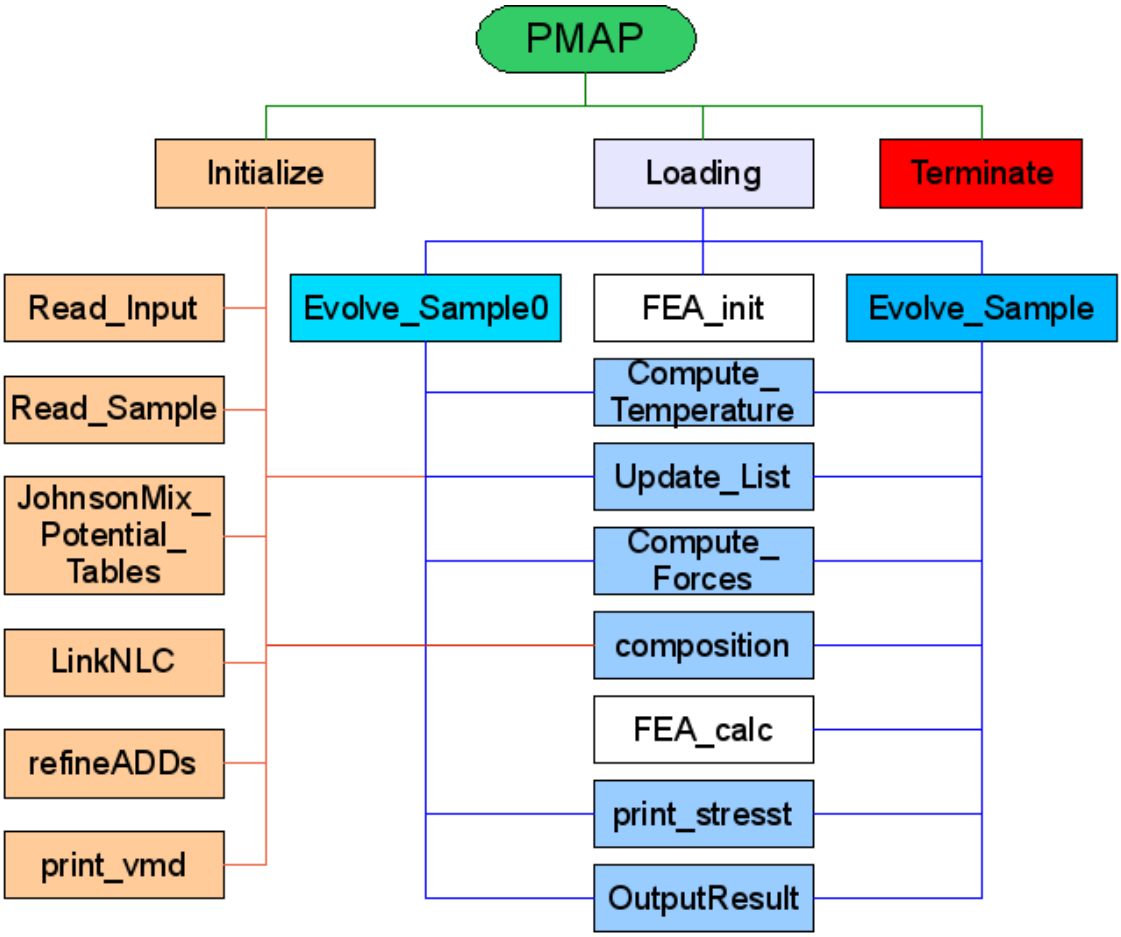


Figure 61. Subroutine tree of PMAP, initialization routines on the left and time evolution routines on the right.

Table I. List of Subroutines and Functions in the PMAP

ID	Routine Name	Description
1	readDomain	Extracts domain information from a string
2	writeDomain	Writes the specified domain to standard error
3	s_word_count*	counts the number of "words" in a string.
4	s_get_word	get the n^{th} word from a string.
5	left_of	returns the string, s, as everything left of the character or

		word, w
6	inDomain	Returns whether position (x,y,z) is within the specified domain
7	DomainVolume	returns the volume of the specified domain
8	DomainCenter	Returns the center position of the specified domain
9	fullangle	Returns the angle from x axis to vector, v
10	Initialize	Initialization procedure, called once at the beginning
11	Read_Sample	Reads the initial GP model file and revives if told
12	Read_Input	Read parameters controlling the simulation from standard input.
13	Initial_Printout	Prints informations on the run parameters on standard output
14	Evolve_Sample0	Equilibration procedure; time evolution of the system.
15	Evolve_Sample	Relaxation procedure between load steps; time evolution
16	Refold_Positions	Particles that left the box are refolded back into the box by PBC
17	Compute_Forces	Computes forces on each particle/atom based on the interatomic potentials.
18	Compute_Temperature	Calculates instantaneous temperature based on total kinetic energy
19	Update_List	Update the Verlet neighbor list
20	MovedTooMuch	True if the sum of the two largest displacements are greater than skin
21	constraint	Modifies displacement vectors according to prescribed constraints
22	Terminate	called once at the end, to deallocate and finalize
23	UpdateLinkList	assigns each particle to a subdomain, ICELL

24	ICELLVAL	Returns the subdomain ID provided a particle position (x,y,z)
25	OutputResult	Writes MD3 and Revive.MD files
26	Loading	Calls first the equilibration then handles the loading procedure
27	JohnsonMix_Potential_Tables	Tabulates all interatomic potentials and their derivatives. Handles analytical forms as well as tabular such as EAM.
28	pot	Returns analytical potential forms
29	dpot	Returns the derivative of analytical potential forms
30	metal_deriv	Calculates derivative of tabular data using five-point interpolation
31	phir	interpolation of EAM pairwise potential
32	dphir	interpolation of EAM pairwise potential derivative
33	psir	interpolation of EAM contribution to the electron charge density
34	dpsir	interpolation of EAM contribution to the electron charge density derivative
35	frho	interpolation of EAM embedding function
36	dfrho	interpolation of EAM embedding function derivative
37	print_vmd	Writes the model configuration to an XYZ file to view in VMD
38	print_stresst	Writes the global and local stresses to the “stresst.out” and “d**stresst.out” files
39	composition	Transforms the real scale particles in the initial ADD, up or down a scale.
40	LinkNLC	Generates the NLC for imaginary particles.

41	BreakingNLC	Severs a constituent of an Imaginary particle's NLC
42	refineADDs	refines the ADDs so that the atom to particle ratio conserves mass
43	FindBox	Determines each Scale's BoxSize for use with Verlet Neighbour Lists
44	natcoord	calculate natural coordinate of an arbitrary point for a given element
45	shapef	Returns quadrilateral shape functions
46	interp	Bilinear interpolate position based on element node displacements
47	FEA_init	initialize the FEA mesh and read the FEAinp file
48	FEA_calc	Solves for the node displacements given boundary conditions
49	interptri	Interpolate the displacement of a position in an element using Barycentric coordinates for half of the element
50	INTEG[#]	Calculates Integration Points
51	DMATX[#]	Creates D Matrix and Element Nodal Coordinates
52	ELSTIF[#]	Creates Element Stiffness and Temperature Load
53	DBMAT[#]	Creates DB MATRIX
54	BAND[#]	Equation Solving Using Banded Storage, forward elimination and back substitution.
55	BandLA	Equation Solving Using LAPack SPBSV routine

**Modified for fortran90 from routines licensed under the GNU LGPL written by John Burkardt*

[#]Modified for fortran90 from Routines from Chandrupatla and Belegundu, "INTRODUCTION TO FINITE ELEMENTS IN ENGINEERING", 4th edn.

a. Compiling and running

PMAP is composed of two source codes, “Multi-Input57.f90” and “FEA19.f90”. The former is the code specific to GP simulations, and the latter is specific to solving FEA models. When compiled together it forms PMAP. The numbers at the end of the source files represents their version number and it typically carries over to the PMAP executable for clarity and to prevent confusion.

The FEA19.f90 code uses the LAPACK routine SPBSV to solve the matrix equation. Only the subroutines required for this routine are included in a subdirectory called “LA”, within which is a compiled library called “libmat.a”. When compiling the code on a new machine this library must also be compiled. In this directory is a script called “makelib.sh” which will compile the library; be sure to check the compiler being used for consistency with the machine being compiled for. This library is reason why the compilation command includes the library path “-LLA/” as well as the name of the library to use “-lmat”.

```
mpif90  Multi-Input57.f90 FEA19.f90 -LLA/ -lmat -o PMAP57f19.exe
```

This command uses the fortran90 compiler for use with the Message Passing Interface (MPI) which enables PMAP to run in parallel, at this time only the GP portion of the code leverages parallel computing, the FEA process runs in serial and does not take much time compared to the GP portion.

2. PMAP Input file

PMAP uses a Free form input file with Key-Value type directives. By Free form, it is meant that directives can appear on any line within the input file or not at all, by virtue of default values. In this way, it is possible to “comment” lines that are unnecessary or not desired, using the Fortran comment character “!” rather than the typical Unix comment character “#”. The Key-value type directives allow this to be possible by pairing a keyword to a property or directive which is modified by parameters and values that follow. For example:

Key Value

specifically could be something like:

CutoffRadius 5.3

Where the Key=CutoffRadius and the Value=5.3. Some keys may take multiple values. Table AIV-II is a list of all directives possible for PMAP, some directives are not recommended for users to use as they may do sensitive or complicated things. The only directives that require a specific sequence are *Species* and *Potentials*, *Species* must be specified before *Potentials*, as the *Potentials* directive requires the data from *Species*. These directives are also both mandatory.

The smallest possible input file would contain only the mandatory directives and be an NVE simulation in a rigid box:

START_GP
Model Model.MD
Species 1
Fe 26 55.847 0.0 1.26
Potentials 1
Fe Fe M 0.4172 2.845 1.389
END_GP

Table II. Available Directives for PMAP

START_GP		Flag to start reading GP input directives
Title	*string*	Set '*string*' to be the title of the simulation. Default=”GPrun”
Model	*file*	Set '*file*' to be the path of the “Model.MD” file. **Mandatory**.
Temperature	*real*	Set the global average temperature of the simulation to '*real*', Default=300.0 K
LocalTemperature	*int*	List '*int*' domains below this that specify domains to have unique temperatures, specify their temperature as the last column after the domain. Default: not present
FEAinput	*filelist*	Connect the GP model to FEA mesh data specified in “*FEA***.inp”* files listed in '*filelist*', Default: no FEA mesh.

Skin *real* Set skin value to '*real*'; allows some deformation before requiring a recalculation of Verlet neighbour lists. Default=0.5 Ang

ScaleRatio *int* Set the GP scale ratio to '*int*'. Default=2

Perturb *real* Perturb the initial structure by a scaled displacement of a random value [0,1] times '*real*'. Default=0.002

PBC *logical(3)* Defines the global box to be periodic along any of the three directions. Default F F F

FixGlobal *real(3)* Globally fix all displacement in any direction, where '*real(3)*' elements can be either 0.0 or 1.0, where 1.0 is free and 0.0 is fixed in the respective direction. Use this to prescribe strict plane strain. Not recommended for more than one direction at a time. Note that a barostat will still be able to scale their positions in that direction. Default=1.0 1.0 1.0

FixLocal *int* Fix '*int*' local domains listed below. Below each domain are two 3-element arrays for equilibration and loading similar in function to the FixGlobal parameters. These can also take two optional angle parameters to change the axis orientation. Default: one spherical domain, includes matter outside of a 1000Ang radius from origin.

WGdomains *int* List '*int*' domains to be WG below, the last column is an integer representing the FEA input file number in the *filelist* of '*FEAinput*', Default: not present

WFdomains *int* List '*int*' domains to be WF below, the last column is an integer representing the FEA input file number in the *filelist* of '*FEAinput*'. Default: not present

EquiTime *real* Time to equilibrate (sec). Default=100.e-12 (100ps)

EquiDataFreq *int* Every '*int*' timesteps a line of data will be written to "Check_out.md", "stresst.out" and the local "d**stresst.out" files during Equilibration. Default=10

LoadDataFreq *int* Every '*int*' timesteps a line of data will be written to "Check_out.md", "stresst.out" and the local "d**stresst.out" files during Load. Default=100

EquiMD3Freq *int* Every '*int*' an MD3 file will be written with the simulation configuration etc. during equilibration. Default=1000

LoadMD3Freq *int* Every '*int*' an MD3 file will be written with the simulation configuration etc. during loading. Default=1000

MD3parameter *char* '*char*' specifies what the sixth column of the MD3 file represents, options are "v,f,e" where 'v' is atomic von Mises stress, 'f' is the atomic force magnitude, 'e' is the atomic potential energy. Default=e

EquiTimestep *real* Set Integrator time step to '*real*' for equilibration Default=1.e-15 sec

LoadTimestep *real* Set Integrator time step to '*real*' for Loading, Default=1.e-15 sec

EquiBarostat [*real*] Use a barostat during equilibration to maintain pressure to '*real*', Default=no barostat, default_*real*=0.0, default isotropic barostatting unless otherwise specified.

LoadBarostat [*real*] Use a barostat during equilibration to maintain pressure to '*real*', Default=no barostat, default_*real*=0.0, default isotropic barostatting unless otherwise specified.

anisotropic Has the barostat function anisotropically, i.e. reduce each direction to the requested pressure. Default=isotropic

isotropic Has the barostat function isotropically, i.e. reduce the average system pressure to the requested, ignoring the stress differences in other directions. Default: present

EquiBaroRiseTime *real* Specify the rise time for the Berendsen barostat during equilibration to be '*real*'. Default=10.e-12.

LoadBaroRiseTime *real* Specify the rise time for the Berendsen barostat during loading along periodic directions and not the loading direction to be '*real*'. Default=10.e-12.

EquiBaroRiseTimeRamp *real int* This ramps the rise time from '*real*' to "EquiBaroRiseTime" within '*int*' timesteps. Not ramping by Default.

LoadSteps *int* Number of loading increments (load is applied in these steps, with relaxation between), Default=0

RelaxationSteps *int* Number of timesteps for relaxation between each load steps, Default=1000

CyclicLoad *real* Applies a cyclic load with one complete cycle every '*real*' Load steps.. Cycle amplitude is equivalent to the Strain specified in "StrainIncrement". Default: not present, Default_*real*=20

StrainIncrement *real(6)* Strain increment for each direction (X Y Z yz zx xy) (strain/loadstep) ex.: yz (y-disp/z-const), Default(6)=0.0.

ElectricField *real(3)* Applies an Electric Field, '*real(3)'*, units of Volt/Ang. Fi=Fi+qi*'*real(3)'*, Default=0.0.

ThermalRamp *real* The thermal ramp will change the temperature of the simulation at the rate of '*real*' degree K per second. Default=0.0

BreakNLC This will allow the links in imaginary particles to break if they become too far away from the central average. (not a very physical criterion) Default: not present

Irms Use the root mean square displacement of the constituents of imaginary particle's NLC instead of the arithmetic mean of constituents' current position.

ADuality *real1 real2* If a local stress domain has a maximum principle stress less than '*real1*' it will recompose or lump into S2 particles, if the maximum principle stress is greater than '*real2*' it will decompose to S1 atoms. Default: not present, Default_*reals*=0.0

LocalStress *int* List '*int*' domains below that should be monitored for local stress and energy, these domains can decompose or recompose if the directive '*ADuality*' is used. Default: one spherical domain, includes matter outside of a 1000Ang radius from origin.

SIdomains *int1 int2 real* List '*int*1' domains below that will be converted to Surface Images (SI) with a maximim of '*int*2' links for each; Default=12. '*real*' specifies the angle (radians) tolerance used when linking surface images; Default=0.15. If 'real' is specified then '*int*2' must also be. Default: not present

FOppDomains *int1 int2* List '*int*1' domains below that will have an opposing force applied to it along the '*int*2' direction, such that acc=acc_new-acc_old along that direction. Used for pseudo 2D models. Default: not present, Default_*int*2=3 the Z direction.

OutputPotTable This will create a file of the inter-atomic potentials used in the simulation, used mainly for debugging purposes. Default: not present

CutoffRadius *real* This specifies the cutoff radius to be used for the atomistic (S1) scale, it is automatically scaled for the higher scale particle interactions. (Make sure that this parameter corresponds to the potential used). Default=10.0 Ang

NeiCutoffRadius *real* This cutoff is used when finding neighbors to use as Neighbor-Link-Cells, also used for linking Surface Images and in the '*BreakNLC*' directive. This value can range from 0.0 to '*CutoffRadius*'+'*Skin*'. When using surface images it is best to keep this value large. The only benefit of this directive is to save a little time when relinking. Default='*CutoffRadius*'

Species*int* List '*int*' rows of the Specie table below following the pattern: ElementSymbol AtomicNumber AtomicMass AtomicCharge AtomicRadius. **Mandatory**.

Potentials *int* '*int*' number of different potential interactions, List '*int*' of them below. **Mandatory** to define after '*Species*' directive. The interaction table takes the pattern: *ElementSymbol1 ElementSymbol2 PotentialType real(3)*.

Where PotentialType can be one of (M,Mc,Mq,B,Bc,L,Lc,Td,J,E,Ed,Es) corresponding to (Morse,Morse with coulomb, Morse with quadratic repulsion,Buckingham,Buckingham with coulomb,Lennard-Jones,LJ with coulomb, TABLE pair-potential in DLPOLY format,Johnson pair-potential,Hard-coded analytical EAM for Cu,EAM from DLPOLY TABEAM file,EAM from setfl file) All potential types must have three parameters, even those that do not need parameters such as those that read table files and those that need only two parameters. Those that require

potantial tables must have the path to the table listed on the line below the interaction.

Revive This will tell the simulation to read the “Revive.MD” file and apply position, velocity, acceleration, and scaleID to the model. If reviving during FEA loading it also reads the domainID, so you cannot add local domains to that revived simulation. This is because the WG and WF domains need to be constant and are based on the equilibrated structure not the initial model. Default: not present

ReLink [*reals*] Forces the relinking of all imaginary particles and even the real particles in local stress domains if '*ADuality*' is present. If [*reals*] is specified it will only relink the real particles in the local stress domains; done automatically if '*ADuality*' and '*Revive*' are present. Relink is done automatically at the start of simulation, i.e. when '*Revive*' is not present. The '*ReLink*' directive is usually not required, it is usually done automatically, and should only be used if you know what you're doing.

NoReNew This will not apply the domains of the input file to the model. However, if domains are present they must be listed in the input file. DEFAULT not present, i.e. ReNew=.true.

DSCalpha *real* The damping parameter for the Damped Shifted Coulomb summation, alpha='*real*'. Default=0.2

ListInputParameters This will output all parameters read from the input file as well as those set to the default. This is used for debugging purposes to make sure that parameters are being set properly.

END_GP flag to end reading GP input file

a. Examples

This section contains example input files that came from a set of simulations designed to teach a student how to use PMAP and to learn the GP method and the capabilities of multiscale analysis. These simulations were also described in Chapter II to

explain the capabilities of PMAP. Here they are below along with the model development input files where relevant.

Table III. Model-1. Input File for the Crystal Generation Code for a Pure Scale 1 Model Specimen. Referred to in Section II.A.1

```
 1  2
21.774 36.29 21.774
1
CuFCC 1  1
-21.774 21.774 -36.29 36.29 -21.774 21.774
s
21.774 -21.774 36.29 -36.29 21.774 -21.774
```

Table IV. Input-1. Scale 1 Model in Tensile Load, the Same Script can be used for any Scale Model the Only Change would be the Model File. Referred to in Section II.A.1

```
START_GP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Model        ../../9_ModelS2/Model.MD_S1  ! file for MD model data
PBC          T T T    !PBC  Periodic Boundary Conditions in X Y Z, DEFAULT=F F F
Temperature    300.0    !Temperature of the simulation
!!!!!!!!!!!!!!!!!!!!!!!!!
EquiTime       15.e-12  !Equilibration time (sec)
EquiBarostat !.000101 !requested pressure for barostat, DEFAULT=0.0 GPa
!!!!!!!!!!!!!!!!!!!!!!!!!
LoadSteps       20      !Number of load steps DEFAULT=0
StrainIncrement 0.0 0.5e-2 0.0 0.0 0.0 0.0 !X Y Z yz zx xy (strain/loadstep) ex.: yz (y-disp/z-const)
!!!!!!!!!!!!!!!!!!!!!!!!!
CutoffRadius    6.5     !cutoff radius (Ang)
Species 1               !number of species; list them below
 Cu 29 63.546  0.0 1.28 !name number mass(amu) charge(e) radius(Ang)
```

```
Potentials 1            !number of interactions; list them below
 Cu Cu M  0.3429   2.866     1.359 !Morse potential
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
ListInputParameters
END_GP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Table V. Model-2. Two Coupled Scale Model Input File, it used the Same PMAP Input File as Input-1. Referred to in Section II.A.2

```
2  2
21.774 36.29 21.774
4
CuFCC 1  1
-21.774 21.774 -36.29 0.0 -21.774 21.774
s
21.774 -21.774 36.29 -36.29 21.774 -21.774
CuFCC 2  1
-21.774 21.774 0.0 36.29 -21.774 21.774
s
21.774 -21.774 36.29 -36.29 21.774 -21.774
CuFCC -1  1
-21.774 21.774 0.0 36.29 -21.774 21.774
s
-21.774 21.774 6.5 29.79 -21.774 21.774
CuFCC -2  1
-21.774 21.774 -36.29 0.0 -21.774 21.774
s
-21.774 21.774 -23.29 -13.0 -21.774 21.774
```

Table VI. Input-2. Scale 1 Model in Tensile Load with Surface Images. Referred to in Section II.A.3

```
START_GP !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Model         ../Model.MD_S1  ! file for MD model data
PBC           F T T    !PBC  Periodic Boundary Conditions in X Y Z, DEFAULT=F F F
Temperature    300.0    !Temperature of the simulation
```

!!!!!!!!!!!!!!!!!!!!!!!!!

EquiTime 15.e-12 !Equilibration time (sec)

EquiBarostat !.000101 !requested pressure for barostat, DEFAULT=0.0 GPa

SIdomains 1 12 1.8 !number of domains, [max links, [tolerance angle(rads)]]

rec F -21.774 21.774 -50.0 50.0 -50.0 50.0

!!!!!!!!!!!!!!!!!!!!!!!!!

LoadSteps 40 !Number of load steps DEFAULT=0

StrainIncrement 0.0 0.5e-2 0.0 0.0 0.0 0.0 !X Y Z yz zx xy (strain/loadstep) ex.: yz (y-disp/z-const)

!!!!!!!!!!!!!!!!!!!!!!!!!

CutoffRadius 6.5 !cutoff radius (Ang)

Species 1 !number of species; list them below

Cu 29 63.546 0.0 1.28 !name number mass(amu) charge(e) radius(Ang)

Potentials 1 !number of interactions; list them below

Cu Cu M 0.3429 2.866 1.359 !Morse potential

!!

ListInputParameters

END_GP !!

Table VII. Model-3. Two Coupled Scale Model Input File, it used the Same PMAP Input File as Input-1. Referred to in Section II.A.2

```
 2  2
43.548 72.58 21.774
4
CuFCC 1 1
-43.548 43.548 -36.29 36.29 -21.774 21.774
T
0.0 0.0 20.0 10.0
CuFCC -2 1
-43.548 43.548 -36.29 36.29 -21.774 21.774
T
0.0 0.0 20.0 10.0
CuFCC -1 1
```

```
-43.548 43.548 -72.58 72.58 -21.774 21.774
T
0.0 0.0 61.6 20.0
CuFCC 2  1
-43.548 43.548 -72.58 72.58 -21.774 21.774
T
0.0 0.0 92.58 20.0
```

E. DATA PROCESSING

After a simulation has been run there are some special programs to assist in the understanding and transformation of the data. If there are several stages to a complicated simulation, there may be a need to alter the configuration after a simulation. For example, the slow decomposition process is just such a case; after each decomposition is required an equilibration since the configuration has become more "resolved" meaning that there is now much more particles and higher DOF.

Here will be discussed configuration format changes for visualization and post-processing, scale and local domain extraction, domain transformation, FEA contour plotting, local stress plotting and movie making. Table VIII lists all the programs available with a brief description of each.

Table VIII. List of Utility Programs to Augment the Capability of PMAP; IDs Suffixes Stand for Topics (a) Analysis, (c) Conversion, (m) Model Manipulation, (d) Debugging Information, and (u) General Utility.

ID	File Name	Description
a	**Analysis**	
1a	avgADD.sh	Takes a given column of the d*stresst.out files and outputs them all together side-by-side with the global strain as the first output column.
2a	Multi_local_stress_ Stoich.f90	Calculates local stress for DLPOLY output as well as MD3 files. Requires an input file to operate, does not include the EAM potential.
3a	matterInvert.f90	Outputs inverse matter file. It removes all matter and includes matter where there was none, i.e. voids and vacuums.
4a	Multi_local_stress_ EAM.f90	Calculates local stress for DLPOLY output as well as MD3 files. Requires an input file to operate. Includes the EAM

		potential **(MAY HAVE BUGS)**
5a	dispMD3_hole.f90	Calculates displacement between two MD3 files for each particle. Also outputs an analytical solution for a plate with a central hole.
6a	ADDxyz.f90	Extracts a range of AD Domains from an MD3 file to view in VMD. Each ADD is given a unique atom type.
7a	CNeval5.f90	Analyzes MD3, and DLPOLY HISTORY files for each particle's Coordination Number (CN), vector (CV), CNA, etc. It outputs these values onto stdout in an xyz format to be viewed in VMD. There is a CN summary on stderr
8a	dCN.f90	determine the change in CN from equilibrium configuration. Reads the output from CNeval.f90.
9a	dispMD3_Bcrack.f90	Calculates displacement between two MD3 files for each particle. Also outputs an analytical solution for a plate with an edge crack using the two-term solution.
10a	dispBatch_crack.sh	Averages particle displacements for a set of small domains around a crack tip. Makes graphs to compare with the analytical solution.
11a	getDs.sh	Concatenates the “d*stresst.out” files from simulation continuation directories into one file “D*”
12a	getTS.sh	Outputs the ADD size for each D* information.
13a	allts.sh	Runs `getTS.sh` for each ADD.
14a	getGe.sh	Outputs the change in energy per unit area from the peak energy to the final energy of each ADD.
15a	getG.sh	Integrates the traction-separation information from `getTS.sh` to produce the energy release rate for each ADD.
16a	getTSGdat.sh	Pastes the Ge, G, and max stress into one file for plotting

		purposes.
17a	TSG.gnu	A gnuplot script to plot the output from `getTSGdat.sh` for each ADD.
c	**Conversion**	
1c	mkMD.sh	Convert MD3 or Revive.MD files to Model.MD format
2c	MD32lammps.sh	Converts an MD3 file into LAMMPS format. Must be programed for specific elements. Currently only Na, Si, Al, and O are supported
3c	mkCONFIG.f90	converts MD3 input into CONFIG file for DL_POLY simulation
4c	mkxyz.f90	Converts Model.MD, CONFIG, and MD3 files to XYZ format for visualization in VMD.
5c	mkMD3.f90	Converts input stream to MD3 format. Readable formats include LAMMPS trajectory, HISTORY, CONFIG, REVCON, Model.MD and XYZ.
6c	mkMD.f90	Convert MD3 files to Model.MD format.
7c	mkFEAinp.sh	Generates an "FEAinp" file from an "abaqus.inp" file for use with GP-FEA, material and fixed point data must be manually modified.
m	**Model Manipulation**	
1m	randComp.f90	Random Composition Generator, good for glasses, makes CONFIG file for DLPOLY. Reads an input file.
2m	delMDdup.f90	checks for duplicate particle positions within a radius of dupTol for Model.MD files.
3m	catREVCON.f90	Concatenates two REVCON files together into one.
4m	domainRotate.f90	Rotates all positions in the XY plane according to angle.

		Coordinates must be first three columns, x y z.
5m	domainMove.f90	Moves all positions according to displacement vector. Can read CONFIG, REVCON, LAMMPS, XYZ, MD3, and Model.MD files.
6m	domainFold.f90	Folds all positions to be within the box assuming PBCs.
7m	delMD3dup.f90	checks for duplicate particle positions within a radius of dupTol for MD3 files
8m	mapADD.f90	maps AD Domain number, ADD, from file1 onto file2
9m	chADD.f90	changes AD Domain number, add0 to add1, from standard input
10m	chScale.f90	changes scale number, s0 to s1, from standard input.
11m	domainScale.f90	Scales (multiplies) all domain positions according to A B C pre-factors
12m	domainMod.f90	Changes atomID properties within specified domain. Such as clamp domain ID, ADD, and element type.
13m	domainRevMod.f90	Adds or sets velocities and accelerations within domains of Revive.MD files. Useful for setting initial velocity and accelerations.
14m	domainExtract4.f90	Outputs points within the given domain, reads MD3 files.
15m	Mater_Model7.f90	Builds Atomic and General Particle Models based on crystalline unit-cells. Reads an input file.
16m	Mater_Model8.f90	Same as above, but does not require the definition of explicit imaginary domains, so it's easier to use.
17m	domainMater.f90	Generates a given crystal structure and scale within a specified domain. Useful for augmenting Model configurations.
d	**Debugging**	
1d	getNeighbours.f90	Outputs all points within a radius, r, centered at coordinates, p.

		Can read from XYZ and MD3 files.
2d	getNLC.f90	outputs NLC information given a Revive.MD file and an MD3.
u	**Utilities**	
1u	mathSub.sh	Reduces any mathematic expressions in a file with its simplified numeric value. Can read variables from file, all variables are considered global. Convenient for model development.
2u	minMaxSumCol.sh	Calculates the minimum, maximum, sum, and average of a given column of data using the shell program 'awk'.
3u	minMaxSumCol.f90	Calculates the minimum, maximum, sum, and average of a given column of data using the fortran90 language.
4u	integrate.f90	Numerically integrates a column of data with respect to another column.
5u	domainFind-rec.f90	Find the X, Y, and Z positional extents of an MD3 file.
6u	getADD.sh	Outputs all particles with given ADD from MD3 files.
7u	getScales.sh	Outputs all particles with given scale from MD3 files.
8u	sets.f90	Partitions a point set A into two subsets: the intersection of volume set B with point set A (B∩A=ADD1) and the complement of B with A (B\A=ADD2). Where volume set B is defined as the union of spheres with radius, r, centered at each positional element in point set C. Like adding a radial fuzziness to set C.
9u	closeTo.f90	prints lines who's value in Column is within a ±Tolerance of a given Value
10u	getRelaxData.f90	prints a number of lines, n, that precede a change in value in the specified column, c.

11u	compress.sh	compresses directory DIR into a gzipped tar archive
12u	uncompress.sh	uncompresses a gzipped tar archive, e.g. archive.tar.gz
13u	matmath.f90	averages every element in the file matrix among several different files, essentially averaging different files together

1. Make XYZ files from CONFIG, MD, MD3 files: mkxyz.exe

This program is used to convert GP Model.MD files that contain imaginary particles and Link cells as well as MD2 and MD3 files, that are output from the Multi-Input GP code, as well as CONFIG or REVCON files from DL_POLY into the xyz format to be visualized with VMD.

The xyz format has the element symbol in the first column followed by the xyz coordinates. Each element takes one line. The very first line of the file is the number of elements in the file followed by an empty line. Since it takes element symbols and the GP configuration files use integers, specifically the element atomic number, rather than the symbol, this code has a few programed symbols on lines 72-77 for the elements: Al, Fe, O, Cu, H. The usage for this program is as follows, the file to be translated is fed to the program as standard input. The converted file is outputted on standard output. Depending on the format of the input depends on what “option" is needed; if converting a Model.MD file the “-i" switch is used, if it’s an MD2 or MD3 file then “-m" is used followed by the number of atoms and particles, NTotal, since it’s not inside the file like it is for MD files. If converting a DLPOLY CONFIG or REVCON file the “-c" switch is used with the total number of atoms afterward, see the usage below.

Usage: mkxyz.exe [OPTION] <inputFile >outputFile.xyz

Options:

-i input file is a GP MD file with imaginary particles and link cells

-m NTotal input file is an MD2 or MD3 file, output of Multi-Input GP code

-c NTotal input file is a CONFIG file of DL_POLY, or like format.

The result can be viewed with VMD such as:

```
$ vmd outputFile.xyz
```

2. Make Model.MD file from MD3 or Revive.MD file: mkMD.sh

Sometimes it is required to run a simulation based on a configuration of a previous simulation. Sometimes it is easiest to make one of the previous MD3 files into a Revive.MD file and "revive" the new simulation. This will work well if immediately beginning loading but if there should be another equilibration it can be cumbersome and the first thing thought of is to just make it a Model.MD file for the new simulation. In this case, this script, "mkMD.sh" can turn an MD3 file or a Revive.MD file into a new Model.MD file for immediate simulation. An example usage can be seen on the last few lines of the script, reproduced below. When executed it will automatically produce the Model.MD file with the name "Model.MD_*inputFileName*" where the *inputFileName* is the name of the file that is being converted.

```
#!/bin/bash

# MD3 to Model.MD ########################################
function MD3_MD () {
grep ".      -.     " $file >Ideal      # extract imaginary from MD3
grep ".       .     " $file >Real
NReal=`wc -l Real | cut -d " " -f1`
NIdeal=`wc -l Ideal | cut -d " " -f1`
echo "NTotal $((NReal+NIdeal))  NIdeal $NIdeal"
echo " `sort -r -k 4 -g $file | head -n1 | awk '{print $4}'`  $((NReal+NIdeal))
$NReal" \    >Model.MD_${file}
echo "  F  `head -n 1 $file`" >>Model.MD_${file}
cat Real>>Model.MD_${file}
awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' Ideal \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> Model.MD_${file}
rm Ideal Real
}

# Revive.MD to Model.MD ##################################
# does not maintain NLCs
function rev_MD () {
boxX="`head -n 1 $file | awk '{print $4}'`"
boxY="`head -n 1 $file | awk '{print $5}'`"
boxZ="`head -n 1 $file | awk '{print $6}'`"
```

```
# extract imaginary from MD3 and normalize it
echo "$boxX $boxY $boxZ" >ideal
grep ".      -.     " $file >>ideal
domainMove.exe -0.5 -0.5 -0.5 <ideal | domainScale.exe $boxX $boxY $boxZ |tail
-n +2 >Ideal
# extract real from MD3 and normalize it
echo "$boxX $boxY $boxZ" >real
grep ".       .     " $file >>real
domainMove.exe -0.5 -0.5 -0.5 <real | domainScale.exe $boxX $boxY $boxZ |tail -
n +2 >Real
rm real ideal
NReal=`wc -l Real | cut -d " " -f1`
NIdeal=`wc -l Ideal | cut -d " " -f1`
echo "NTotal $((NReal+NIdeal))  NIdeal $NIdeal"
echo " `sort -r -k 4 -g Real | head -n1 | awk '{print $4}'`  $((NReal+NIdeal))
$NReal" \    >Model.MD_${file}
echo "  F  $boxX $boxY $boxZ" >>Model.MD_${file}
cat Real>>Model.MD_${file}
awk '{print $1" "$2" "$3" "($4<0?-$4:$4)" -"$5}' Ideal \
| sed -e 's/$/\n0\ 0\.000\ 0\.000\ 0\.000/' >> Model.MD_${file}
rm Ideal Real
}

if [ $# -eq "2" ]; then # if two parameters given
 echo "Converting Revive.MD to MD"
 file="$2"
 rev_MD $file
elif [ $# -eq "1" ]; then # no option, assume MD3
 echo "Converting MD3 to MD"
 file="$1"
 MD3_MD $file
else # no options
 echo "Convert MD3 or Revive.MD files to Model.MD format"
 echo " will produce output file Model.MD_FILE"
 echo "Usage: mkMD.sh FILE.MD3 "
 echo “     will produce: Model.MD_FILE.MD3”
 echo "Usage: mkMD.sh -r Revive.MD "
 echo “     will produce: Model.MD_Revive.MD”
fi
```

3. Extract a specific scale or local domain from MD3 files: getScale.sh getADD.sh

Sometimes it is convenient to do analysis or visualization on a particular subdomain one at a time. To do this two simple shell scripts were developed to extract a particular scale or local domain from an MD3 file. They take two arguments: the scale or local domain number to extract and the file to extract it from. For example:

```
$  ./getScale.sh  1  Equilib.MD3  >Equilib.S1.MD3
$  ./getADD.sh  1  Equilib.MD3  >Equilib.d01.MD3
```

The “getScale” script is very short (4 lines) and simple, it is shown below. The number of spaces in quotes is very important.

```
#!/bin/bash
head -n1 ${2}
grep ".       ${1}     " ${2}
grep ".      -${1}     " ${2}
```

The “getADD” script is even shorter (3 lines) and shown below:

```
#!/bin/bash

head -n1 ${2}
grep "\ \ ..${1}[0-9][0-9]\ " ${2}
```

4. Domain utilities

When dealing with complicated models or to make a complicated model it becomes necessary to manipulate certain geometric domains. To do this there are some programs that were developed to do such things as move domains, extract, transform the coordinates, fit data into a box, or determine if a point is inside of a domain.

a. domainMove and domainScale

These read a stream of coordinates from an MD3 formatted file into standard input and outputs the file with translated or scaled coordinates, respectively, according to a translation-vector, delta x, delta y, and delta z or Scaling factors, A, B, and C. This will move or scale the inputted domain according to these components, for example:

```
$ ./domainMove.exe  -20.0  3.629  20.0   <domain0   >domain.moved
$ ./domainScale.exe  2.0  3.0  2.0   <domain0   >domain.scaled
```

The first example moves 'domain0' -20 units in the x direction and 3.629 in the y and 20 in z. The second example multiplies the domain coordinates by 2 in the X and Z directions and 3 in the Y. These two programs can be used together to convert normalized coordinates into real coordinates, such is needed for transforming a Revive.MD file into an MD file, as discussed in Appendix E.2 with the command:

```
$ domainMove.exe -0.5 -0.5 -0.5 <real | domainScale.exe $boxX $boxY
      $boxZ |tail -n +2 >Real
```

If a Model.MD file is needed to be moved or scaled, these programs just need to read four command line options, the fourth option can be anything, for example:

```
$ ./domainMove.exe  -20.0  3.629  20.0  MD  <domain0   >domain.moved
```

The will be able to transform Model.MD formatted files and keep them in the same format.

b. domainExtract

This is a very simple program that reads a stream of coordinates from standard input and outputs the lines that are within a specified domain. Its parameters are the domain values following those used for the GP simulation code. (note: the version before this this one could only use rectangular domains)

```
$  domainExtract2.exe rec T -40. -20. 0. 0. -20. 20. <0210000.MD >210k.domain
```

The "<" will stream its file: "0210000.MD" as input to the program, and the program's output will be streamed into the "210k.domain" file. The domain shown includes particles between x=-40. and x=-20. Every particle along the y direction and all particles between z=-20 and z=20. This program is extremely useful, as any transformation or operation performed, must be performed on an extracted domain, separate from the entire model. The output from this program can be piped into another program for transformation or filtering or addition, et cetera.

c. inDomain

This program demonstrates the routines used to determine what points are inside of a specified domain. These routines are used in the domainExtract code in Appendix E.4.b. The routines used are: readDomain, writeDomain, s_word_count, s_get_word, left_of, inDomain, DomainVolume, DomainCenter, DomainResize. This program is used to test these domain routines. There are two parameters set inside of the program on lines 508 and 509 to specify a data point and the boxsize. It takes an input file that has the number of domains to test followed by the domains. An example input file is shown below:

```
5      ! number of domains
rec T -10. 10. -10. 10. -10. 10 !rectangular, inside domain, xmin xmax, ymin
ymax, zmin zmax
per T -10. 10. -10. 10. -10. 10    2.0 !perimeter, inside domain, xmin xmax,
ymin ymax, zmin                                    ! zmax, only go inside
domain this much
sph T 0.0 0.0 0.0   2.0                !sphere, inside, x y z, radius
ell T 2   0.0 0.0   2.0 1.7            !ellipse, inside, axial direction i.e.
2=Y, x z, a b
tub T 2   0.0 0.0   2.0 1.7   1.5 1.2   !tube, inside, axial direction, 2 of x
y or z, outer                                      ! a b, inner a b
```

The program will read each domain with "readDomain", calculate its volume with "DomainVolume" and its center with "DomainCenter", then it will say whether the point is inside of the domain or not with "inDomain" and will write out the domain to confirm with "writeDomain". Lastly it will output the total volume for all of the domains and their

inverted volume. The output for this input file is shown below when the point is: 0.0, 0.0, 0.0 and boxsize: 519.1552, 519.1552, 519.1552.

```
 point:    0.0000000       0.0000000       0.0000000
 Domain Volume:   8000.0000
 Domain Center:   0.0000000       0.0000000       0.0000000
 in domain:
rec T  -10.000  10.000 -10.000  10.000 -10.000  10.000
 Domain Double:
rec T  -20.000  20.000 -20.000  20.000 -20.000  20.000
 Domain Volume:   3904.0000
 Domain Center:   0.0000000       0.0000000       0.0000000
 NOT in domain:
per T  -10.000  10.000 -10.000  10.000 -10.000  10.000   2.000
 Domain Double:
per T  -20.000  20.000 -20.000  20.000 -20.000  20.000   2.000
 Domain Volume:   33.510323
 Domain Center:   0.0000000       0.0000000       0.0000000
 in domain:
sph T    0.000   0.000   0.000   2.000
 Domain Double:
sph T    0.000   0.000   0.000   4.000
 Domain Volume:   5545.3125
 Domain Center:   0.0000000       0.0000000       0.0000000
 in domain:
ell T  2    0.000   0.000   2.000   1.700
 Domain Double:
ell T  2    0.000   0.000   4.000   3.400
 Domain Volume:   2609.5588
 Domain Center:   0.0000000       0.0000000       0.0000000
 NOT in domain:
tub T  2    0.000   0.000   2.000   1.700   1.500   1.200
 Domain Double:
tub T  2    0.000   0.000   4.000   3.400   3.000   2.400
 Total volume of all domains:    20092.383      1.43595152E-02 %
  inverse volume:   1.39903728E+08   99.985641     %
```

d. domainCoordTrans

This program was never fully tested and never used but is included here for completeness. Its purpose is to transform an MD2 file from rectangular coordinates to either cylindrical or spherical and vice versa. Its usage is described below.

```
Usage: $ ./domainCoordTrans.exe Coord1 Coord2 < FILE >newFILE
```

It transforms all domain positions from Coord1 to Coord2. Coordinates must be the first three columns, x y z. Parameters may be the following strings:

rec Rectangular or Cartesian Coordinates, x y z

cyl Cylindrical Coordinates, r theta z. theta: angle on plane perpendicular to Z-axis

sph Spherical Coordinates, r theta phi. theta: angle from Z-axis. phi: angle on plane perpendicular to Z-axis

e. domainFind-rec

This program will fit all input data into a rectangular domain oriented to the axial planes. Even though this program is not very flexible, it can give a general idea of where a domain is located and its dimensions. For example, if there was evaporation in a simulation, this program would show that the domain has significantly increased its dimensions. It also can give an initial guess for scale/domain extraction. It takes one argument, the scale to fit. Its usage is shown below.

```
$  ./domainFind-rec.exe  2  <Equilib.MD3
```

It will output the six rectangular parameters which correspond to the minimum and maximum particle position for all three directions.

5. Plot stress contour from FEA: meshplot.exe

The FEA subroutines output node displacements and stresses every time the GP code outputs a configuration MD3 file. The FEA output files are named: FEA14.out0022 where the ‘14‘ denotes the FEA version number and the ‘0022‘ is the calculation number.

The corresponding GP output file would be 0053000.MD3 if there was a 30000 step equilibration and if the FEA calculations were every 1000. This is because the FEA step 0 is completed at 31000 steps.

This program will plot the mesh of the FEA part of the multiscale model and color the element boundaries according to the Von Mises stress of the integration points. It begins by reading the FEA input file, assuming that its name matches the first 6 letters of the FEA output file and ending with “inp”. It reads this file to find the initial node positions. It then reads the FEA file that records the equilibrated WG and WF node positions; the file ending in “init” This file is written after the FEA step 0. It reads this to update these positions. Then it reads the node displacements and element VM Stresses from the FEA output file provided. It uses this data to write a data file with the new node positions and VM Stress in GPa for each element's node. This data file is named: “0022.meshdat” numbered for the FEA calculation step. It also writes a GnuPlot script to plot this data file called: “0022.gnu”, it makes a picture named: “0022.png” the gnu script file can be modified to plot the data however the user wishes. An example output picture is shown in Figure 62, this has a single FEA domain around a central GP model.

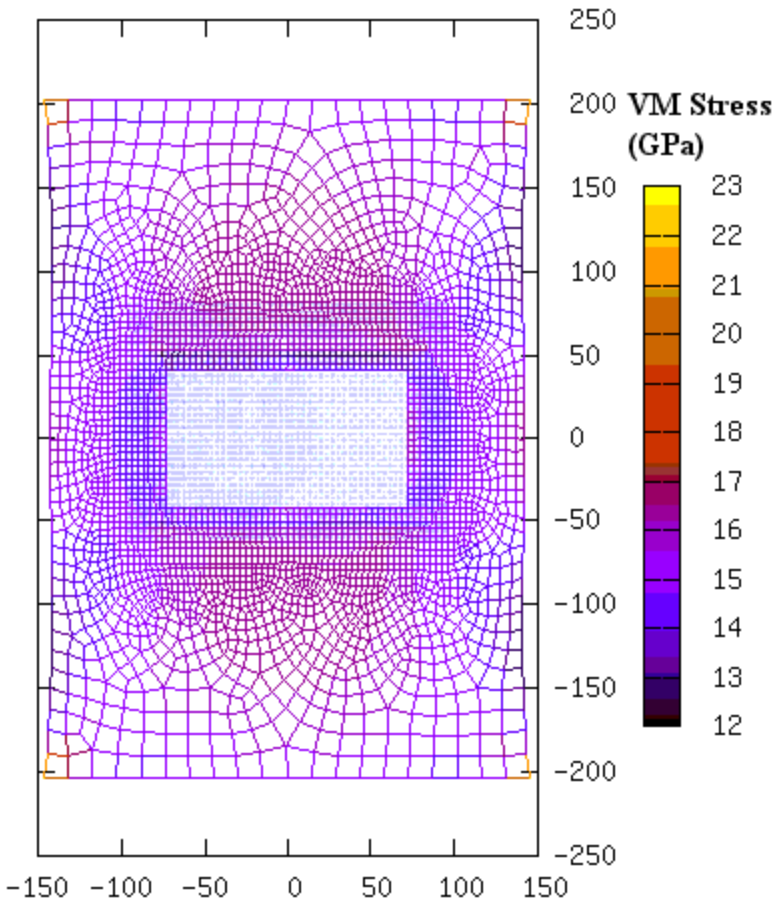


Figure 62. FEA-GP Cu model at 13% strain and 384 ps, colored for element VM Stress.

To use this program an example command would look like:

```
$ ./meshplot.exe  FEA14.out0022  0053000.MD3
```

6. Plot FEA output and plotfiles: plot.sh

This script averages the FEA VM stress, X stress, Y stress, X displacement, and Y displacement for each FEA calculation and puts this data into the file named: “plot”. This is so that these quantities can be plotted through time or strain. Even though it's a global FEA average it can show possible trends, for the distribution of stresses the “meshplot.exe” program is better. Below is the script.

```
#!/bin/bash
```

```
j=0
#find length of plot files
len=`wc -l FEA*.plot0001 | cut -d " " -f 1`
for file in `ls -1tr FEA*.plot*`; do
 j=`expr $j + 1`
# calculation step, avg VM stress, avg X stress, avg Y Stress
 echo $j `awk -v ln="${len}" '{ SUM += $1}{ SUMb += $2}{ SUMc += $3} END {
print SUM / ln \
       " " SUMb / ln " " SUMc / ln }' $file` >>tmp1
done

j=0
#find the number of nodes from the input file
NN=`sed -n '4p' FEA*.inp | cut -d " " -f 1`
for file in `ls -1tr FEA*.out*`; do
 j=`expr $j + 1`
# average X displacement
 tail -n +4 $file | head -n ${NN} | awk -v nn="${NN}" '{ SUM += $2} END { print
SUM / nn }'\
   >>tmp2
# average Y displacement
 tail -n +4 $file | head -n ${NN} | awk -v nn="${NN}" '{ SUM += $3} END { print
SUM / nn }'\
   >>tmp3
done

paste tmp1 tmp2 tmp3 >plot       #puts files together as columns
rm tmp1 tmp2 tmp3
```

7. Plot local ADDomains: plotLS.sh

Sometimes it is useful to plot the local stresses and compare them to one another. This script takes a parameter from the local stress data files “d**stresst.out” and puts them all together so it can be easily plotted and compared. It takes one optional parameter, the column in the local stress file to plot, if neglected it will plot the local energy, column 3. The usage is:

```
$  ./plotLS.sh 6
```

This will compare all the local stress domains' Y stress component by making a GnuPlot script file called: “temp.gnu” which will plot all of the local domains. Below is the “plotLS.sh” script.

```
#!/bin/bash

if [ -z $1 ]; then
 col=3
elif [ $1 = "--help" ]; then
 echo "run with no parameters will graph all local domain's energy"
 echo "options are the column to graph in the files"
else
 col=$1
fi

echo "set xlabel 'time (ps)'" >temp.gnu
echo "set ylabel 'Average atomic energy (eV)'" >>temp.gnu
#echo "set output 'dEnergy.png'" >>temp.gnu
#echo "set term png" >>temp.gnu

cnt=1
echo -e "p 'd01stresst.out' u (\$1/1000):$col t \"ADD $cnt\" w l\c" >>temp.gnu

for file in $(ls -1 d*stresst.out)
do
 if [ $file = "d01stresst.out" ]; then
  continue
 fi
 echo ', \' >> temp.gnu
 cnt=$((cnt+1))
 echo -e " '"$file"' u (\$1/1000):$col t \"ADD $cnt\" w l\c" >>temp.gnu
done
echo "" >>temp.gnu
echo "pause -1" >>temp.gnu

gnuplot temp.gnu

#rm temp.gnu
```

8. Movie Generation

To make a movie there must be a series of pictures that can be used as each frame in the animation. These pictures should all be the same size and minimize the white space around the image; VMD[130] usually has a lot of white space around the model. It is also helpful to label each frame to denote the change in time or strain or whatever variable is being animated across.

There are two ways to make movies from a series of pictures, each have their pros and cons. The easiest way is to make an animated GIF image. This format is good if there are only a small number of frames, if image quality is required to be high, and portability and ease of use is needed. It's not so good if there are a lot of frames since it will make the GIF file very large and may not load very well. The other way is to transcode the frames into a video format like FLV or MPEG. These formats are good if there are a lot of frames since they use compression algorithms to keep the file size down. The disadvantage is that FLV compresses the information in the same way that JPEG images do, so there is a loss of data, this means that FLV videos are blurrier and lack the crisp detail that a GIF image has.

To prepare the pictures to be used as frames for animation the ImageMagick Suite[144] of tools is used to modify the pictures in batch for consistency, specifically the command "convert". Below is the script used to perform these operations. The uncommented lines are for a GIF image, to make an FLV some of the lines should be uncommented:

```
#!/bin/bash
i=0   # initialize counter

# process every PNG picture in the current directory
for line in `ls *.png`; do

#increment counter
 i=`expr $i + 1`

#crop the original image to get rid of white space and rotate counterclockwise
90 degrees
#convert -crop 460x445\!\+160\+5 -rotate \-90 ${line} ${line}t.png
convert -crop 822x652\!\+100\+90 ${line} ${line}t.png
# The first two numbers specify the wanted image size, the last two numbers is
the
```

```
#  coordinate of the wanted top left corner. the new cropped image is named the
same
#  but has t.png at the end

#annotation to add the strain value to each frame
#must convert to jpg for the conversion into a movie
#convert -quality 100 -pointsize 20 -annotate +5+25 "$(((i-1)/2)).$(((i-
1)%2*5))%" \
#       ${line}t.png `printf %02d ${i%} ${i##*}`.jpg
convert -quality 100 -pointsize 20 -annotate +5+25 "$(((i-1)/2)).$(((i-
1)%2*5))%" \
       ${line}t.png `printf %02d ${i%} ${i##*}`g.png
# Pointsize is the font size of the annotation text. The annotate option sets
the
#  coordinate to place th text, remember: image coordinates begin from the top
left, X goes
#  from left to right and Y goes from top to bottom. The text to be annotated
is in
#  quotation marks, here two integer calculations are made for the whole number
and decimal
#  according to the frame/image number counter variable: $i. The resulting
frame is named
#  according to the frame number with lading zeros, this is important for the
creation of
#  an FLV file.

# remove temporary file
rm ${line}t.png

done    # end loop
echo "processed $i files"

#Make the animated GIF file, pausing 0.25 seconds between frames, and set the
animation to
#  loop forever. The resulting file is “test.gif"
convert -delay 25 -loop 1 *g.png test.gif
#rm *g.png

#combine all of the jpg files together at a frame rate of 5 per second and
bitrate of 1800
#ffmpeg -r 5 -b 1800 -i %02d.jpg test.flv
```

```
#play the resultant movie with mplayer
#mplayer test.flv
mplayer test.gif
```

F. ANALYSIS

This section is about the analysis of the data from the GP simulations after any processing of Appendix E. This includes void detection, dislocation identification, and local structure analysis.

1. Void detection: matterInvert.exe

The Matter Inverse method creates an empty grid over the entire model domain; this grid should have units larger than the nearest neighbor distance. The grids are populated by values that correspond to the number of atoms/particles within them. All of the grids that have nothing in them are output to a file containing the grid locations, see Fig. 63. In essence providing a map of the void or vacancy locations. This is helpful in many ways. First, voids are usually less numerous than material models, this greatly reduces the amount of data to analyze. Second, the grid locations are perfectly spaced so perfect structural values can be obtained. This output is designed to be fed into the NNg algorithm of the CNeval program described in Appendix F.2, and will be able to determine the surface of the inverted voids. Since the grid is perfectly structured, the CM_{cutoff} and CN_{radius} can be confidently defined. Below is an example command to invert the model.

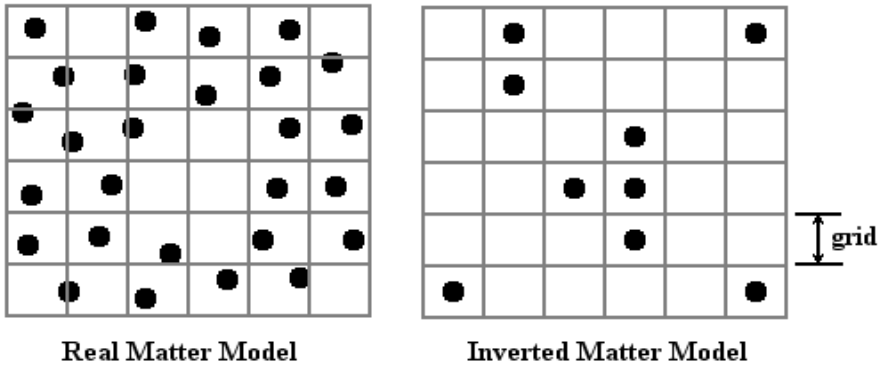


Figure 63. The relationship between real matter and inverted mater models.

```
$ matterInvert.exe -g 2.8 <ModelPore.MD3 >ModelPore.inv.MD3
```

The output is on standard output and in an MD3 format so that the mkxyz.exe program can easily be used to make it viewable in VMD. The model to be inverted is fed to the program on standard input. Certain information is output on standard error such as the percent of the model volume that is vacant and the plane area that is vacant. There are three command line options:

-g [grid] grid spacing (default=1.0) units in Ang

-h [frame] history file frame number

-b [domain(6)] override boxsize with this domain

With the "-h" option HISTORY files from DLPOLY[106,107] can be inverted, the option requires the frame number in the file. The "-b" option allows a rectangular domain to be inverted instead of the entire model's boxsize. The grid spacing specified with the "-g" option should increase with the GP scale being analyzed. An example of the output on standard error is shown below from the command:

```
$matterInvert.exe -g 3.2 -b -186.918 186.918 -186.918 186.918 -20.0 10.0
<0102500.MD3 >inverted.MD3

 Using unit size:   3.2000000
 Using Box Domain:   -186.91800       186.91800      -186.91800       186.91800
-20.000000      10.0000000
 Using grid size:         116         116           9
 Using new grid unit:   3.2227242       3.2227242       3.3333333
      358613  counted
 Void Fraction  0.19024970
 Plane Void Fraction  0.53160918      0.95881224      0.48625147
```

Tip: Sometimes the domain that is to be analyzed is not a convenient shape, there are three ways to deal with this, 1) under-sample the domain, by reducing the Inverse-domain size to fit inside of the domain to be analyzed. 2) Over-sample the domain, this required the grid size to be larger which reduces the resolution of the inverted result but is necessary since the domain will include, most likely, a higher scale. 3) Over-sample with post-cut, the same as making the domain larger, but keeping the grid size at a high resolution, the results will have "*false vacancies*" which should be neglected, the result

can then go through the *domainExtract* program (See Appendix E.4.b) to cut the results to a desired shape domain. The disadvantage of this is clear; that it requires much more analysis to properly operate.

2. Structure analysis: CNeval.exe

This is the most advanced program for GP configuration analysis. For analyzing MD, MD2, and DLPOLY HISTORY files for each particle's Coordination Number (CN), Coordination Vector (CV), Common Neighbor Analysis (CNA) and near-neighbor grouping (NNg). The "CNeval.f90" code can be used. There are actually three different versions to this code; the most current is "CNeval3.f90". The computer cluster uses a link to the most recent version so that the user can ignore the version number and just use "CNeval.exe". The program takes sixteen optional arguments and one mandatory; the mandatory argument is the file name of the configuration, such as "0007500.MD3". It reads by default MD2 and MD3 files.

The program outputs to four different file descriptors, Standard Error (stderr), Standard Output (stdout), *filename*.CNdens, and *filename*.densV. It is customary to redirect the stderr and stdout to files with the extensions ".CNdat" and "xyz", respectively.

Examples of usage can be seen below, followed by program options. The third example is rather specific; it reads and analyzes the file "0550000.MD2" and redirects the CN summary into "550000.CNdat" then pipes the stdout into **egrep** which is a special kind of **grep** that can search more than one thing at a time, in this case it filters out only the particles with CN greater than 12, or having element type, "Al" or "Si", then it puts this data into "550000gt12.xyz" to be viewed with VMD.

Usage: CNeval.exe [Options] [inputfile]

```
 example: CNeval.exe ' 550000.MD2' > 550000.xyz
 example: CNeval.exe ' 550000.MD2' 2> 550000.CNdat > 550000.xyz
 example: CNeval.exe " 550000.MD2" 2> 550000.CNdat | egrep "Al|Si" >
550000gt12.xyz
 example: CNeval.exe -d 3 '55000.MD2' > 55000.xyz
 example: CNeval.exe -n 453 '55000.MD2'
 example: CNeval.exe -d 2 -h 21 -a 2.87 'HISTORY' > hist.xyz
```

```
 example: CNeval.exe -d 2 -a 3.09 -g 0.2 '55000.MD2' 2> 55000.MD2.CNdat >
55000.MD2.xyz
 example: CNeval.exe -a 3.09 -g 0.16 -r 1.26 0055000.MD2 2> 55000.MD2.CNdat >
55000.MD2.xyz
```

Options:

-d dim The "-d" switch is used to specify which direction to integrate across when doing the CN density and CV summation. The values acceptable for *dim* are from 1 to 3 as integers representing the X Y and Z directions. The default value is 1 for the X direction.

-l len The "-l" switch sets the thickness or resolution of the directional analyses, for use with CN densities and CV densities. By default it is 1Å. This is generally the smallest value necessary.

-p PBC The "-p" switch is used when the model has some kind of periodic boundary condition. The parameter it takes is a string of three letters, they are boolean, so they can be either true or false. Each letter's position corresponds to each of the three dimensions, for example, having periodic boundaries in the X and Z direction would have a parameter of: TFT; "T" for true and "F" for false.

-L param The "-L" switch tells the program that the input MD file contains Link-cells. Usually the only file that contains Link-cells are the initial GP configuration file. If the parameter is True, it will output the imaginary particles along with the real ones. All imaginary particles are represented by Hydrogen, to easily identify them from the other particles in the "xyz" file.

-i param The "-i" switch is used to output imaginary particles from the input file, its parameter must be True (I.e. "-i T"). As for the "-L T" option the imaginary particles will be seen as Hydrogen in the "xyz" file.

-n atom The "-n" switch will dump the positions and ID of *atom*'s nearest neighbors and give the *atoms*'s CN, CV and CM, as well as output a simple RDF to the file called "RDF*atom*.dat" where *atom* is the atom number in the configuration, or more specifically the atom number according to the sequential list in the input file.

-a CNcutoff The "-a" switch specifies the lattice constant to use for analysis. The units are in Ångström. The default is 3.629 Å used for copper.

-h frame The "-h" switch specifies that the input file is a DLPOLY HISTORY file and its argument, *frame* is the frame number in the HISTORY file to use for the configuration. When using this switch the output files are called explicitly, "hist*frame*.CNdens" and "hist*frame*.densV*"*. When using a DLPOLY HISTORY file, the "-h" switch must be used, otherwise the program will read the input file incorrectly and it will not run.

-g CMcutoff The "-g" switch will perform Nearest-Neighbor grouping (NNg), with CMcutoff to filter out the low CM particles so NNg will only be done on the high CM particles. The individual particle results from the NNg analysis is output to another separate file ending in "NNg.xyz". See Appendix F.2.d.

-c specie The "-c" switch performs the CNA analysis and NNg to group 'specie'. The specie by default is "Cu". The NNg is only used to group those particles that are found to be in the local structure identified by *specie*. The NNg results are outputted to a file ending with "NNgCNA.xyz". See Appendix F.2.a.

-r atomr The "-r" switch will redefine the atomic radius from a0/2.25 to *atomr* for use with NNg surface area calculation based on the CM. See Appendix F.2.d.

-b box(6) The "-b" switch identifies a local domain of the model to analyze.

-k scale The "-k" switch defines the scale ratio, k. The default is k=2

-o NN The "-o" switch specifies the Nearest neighbor distance used for an experimental local distortion vector. The default is 2.566Å. And the vector is defined as: $os_i = \sum_{j=1}^{N} \frac{(Rij)}{NN} - \sum_{j=1}^{N} \frac{(Rij)}{\sqrt{dist} \cdot CN_i}$ This vector is listed at the end of the lines in the output xyz file.

-t types The "-t" switch filters for elements included in the analysis, i.e. atoms i example: '-t Al,Fe,O,' for HISTORY or '-t 13,26,8,' for MD3. This can be helpful for very complicated models with many different types of elements.

-N scale The "-N" switch makes the program use Verlet Neighbor Lists for the neighbor analysis. This is very efficient if the model involves a very large DOF.

a. Common Neighbour Analysis (CNA)

The Common Neighbor Analysis technique identifies individual atoms' local structure by comparing the atoms' nearest and next-nearest neighbors. This method yields highly detailed results uniquely identifying specific atomic structures. This section discusses how to implement this method to identify atoms in FCC, HCP and tentatively BCC orientations.

i. Method

The Common Neighbor Analysis (CNA) is not a new concept.[145] However, it wasn't until Honeycutt and Andersen[133] that the possible local structures were given quantifiable values and clearly explained. In their words: *"...pairs of atoms are classified by (i) whether or not they are near-neighbors, (ii) the number of near-neighbors they have in common, and (iii) the near-neighbor relationships among the shared neighbors. Two atoms are said to be near-neighbors if they are within a specified cutoff distance of each other. We typically used a cutoff distance of 1.4σ in our analysis, which is roughly the distance to the minimum in the pair correlation function of a solid-like cluster."* Their "near-neighbor" cutoff distance is the same that is used for finding Coordination Number (CN).

The real analysis and key to this method is the description of the common near-neighbors. These are the atoms that are near-neighbors of both the i^{th} and j^{th} atoms. The configuration of these common atoms define the structure as it relates to the i − j atom pair.

(a)CNA Values

There are two types of i − j pairs. The first (I) is when i and j are near-neighbors and the second (II) is when they are next-near-neighbors. Examples of these two types can be seen in Fig. 64 where the open or unfilled circles represent i and j and the black filled circles represent the atoms they have in common, excluding j. Near-neighbors are

represented with a black line between them or "bond". For convenience, the i − j pair will no longer be included in the diagrams as they are implied.

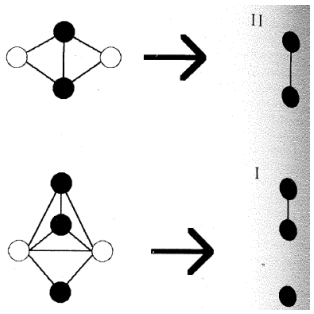


Figure 64. Examples of the two types of pair relationships, I, when the pair are near-neighbors, and II, when next-near-neighbors.

There are many different configurations that the common atoms may have. A few, most frequent configurations are shown in Fig. 65 along with their CNA value. Each i − j pair yields a four digit number, which will be called, CNA value. The first digit is either 1 or 2 indicating the diagram "Type" explained in Fig. 64. The second digit represents the number of near neighbors shared by the i − j pair, or number of common atoms. The third digit represents the number of bonds among the shared neighbors, or how many common atom pairs are near-neighbors.

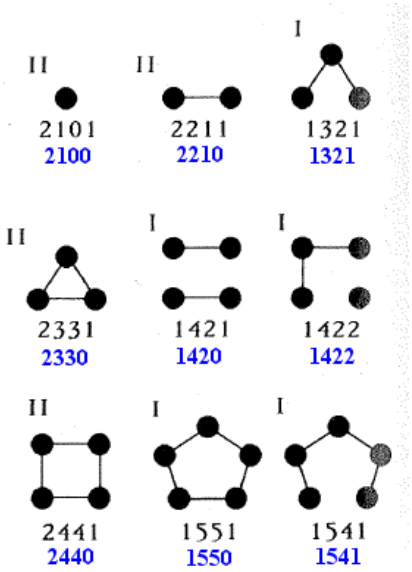


Figure 65. Examples of common atom configurations, omitting atoms i and j.

These three numbers are not sufficient to characterize a diagram uniquely, so a fourth integer, whose value is arbitrary as long as it is consistently used,[133] is added to provide a unique correspondence between numbers and diagrams. Most users of the CNA[146] arbitrarily keep Honeycutt's fourth digit scheme. However there is another quantity that could be used as the fourth digit that would also preserve the uniqueness. The range of the number of bonds per common neighbor can be used as the fourth digit, these CNA values are in blue, below the first set in Fig. 65. Looking at the diagram for the CNA value 1421 (1420) one can see that each common atom has only one bond, so the range is 0. Looking at the diagram for the CNA value 1321 (1321) one can see that two atoms have one bond and one atom has two, so the range is 1, being the difference in bond numbers, $2 - 1 = 1$. Lastly, looking at the diagram for the CNA value 1422 (1422) one can see that there is a lonely atom with no bonds, two atoms with one bond, and one atom with two bonds, so the range is 2; $2 - 0 = 2$. This method of numbering the last digit is useful in the program implementation discussed in section 5.2.

(b)Characterization

Honeycutt and others have researched structures that form in small atomic clusters[133,145] from their findings they could identify atoms in FCC and HCP local structures, along with multilayer icosahedral clusters. All twelve of the nearest neighbors in the FCC structure yield a CNA value of 1421 (1420). For the twelve nearest neighbors in the HCP structure only six have CNA values of 1421 (1420), the remaining six are 1422 (1422). Tsuzuki et al.[147] used a combination of CN, CNA, Centrosymmetry Parameter (CSP) and their own Common Neighborhood Parameter (CNP) to analyze crystal deformation and structural transformation. They illustrated the actual CNA values of the BCC structure which are 1441 and 1661 in an unexplained ratio.

ii. Code Specifics

The code that was written for the structure analysis is simply a slight modification to the CNeval.f90 program – hence making version 2 i.e. CNeval2.f90, the inclusion of Verlet Neighbor lists makes version 3, CNeval3.f90 -- that has been used successfully to analyze the CN of different scales of GP models. The modified code for the CNA is located in: /shared/DATA/Multi_GP_Cu/NPT/CNeval3.f90 The main modification is the

addition of an array called “SO” that contains the CNA of each nearest neighbor pair. This means that it only finds Type I pairs, those that are near-neighbors. This is sufficient to distinguish between FCC, HCP, and BCC structures. Line 624 is where the second digit is calculated for the i − j pair; the number of mutual neighbors. Line 631 accumulates the number of mutual bonds or bonds of the common neighbors and it is added to “SO” on line 635. Lastly, line 637 adds the difference of mutual bond numbers, the range used as the fourth digit. This entire “SO” array is written in the produced xyz file along with the other CN data.

In order to determine what structure atom i is in, all of the CNA values from all of i’s near-neighbors must be analyzed somehow. In the code, they are all summed together, so an atom in an FCC structure will have a value of 12 × 1420 = 17040 (line 643) and HCP is 6 × 1420 + 6 × 1422 = 17052 (line 644).

Table IX. Element Types Used in CNA Visualization

Condition	Element
CN=12	Mg
CN>12	Al
CN<12	Na
FCC	Cu
HCP	B
BCC	Fe

A potential problem occurs when thinking about BCC, certainly we know what CNA values it should be, but in a mixed element model, the cutoff radius is different for different materials. For example, Aluminum has a lattice constant of 4.0559Ang and has an FCC structure, this means that the distance to the first minimum of the pair distribution function is about 3.46Ang, For Iron, which has a BCC structure and lattice constant of 2.92Ang, the CNcutoff radius is 2.72Ang. What results is that Al with the larger cutoff radius is used in the model, which makes the regular Iron atom have a CN of 14 which happens because the CNcutoff is large enough to count both nearest and next nearest neighbors. In the code it is assumed that a BCC structure will have a CN of 14,

and that six of its i − j pairs will have a CNA value of 1440 and eight with 1660 yielding 6 × 1440 + 8 × 1660 = 21920. In order to visualize this data, the structures were assigned element names that correspond to the CNA structure types, this allows the model to be viewed with VMD easily, see table 1 for the specifics.

b. Coordination Number (CN)

The Coordination Number (CN) as defined for crystal structures is the number of atoms that are touching the atom of interest. For bulk BCC metals this number is four and twelve for FCC metals. Another way of thinking about CN is by considering the number of nearest neighbors. Due to temperature and other effects, the position of the nearest neighbors may vary. To ensure that they are counted, a radius larger than the ideal nearest neighbor distance is used to "collect" these atoms. Traditionally, the radius used is the distance of the first minimum in the radial distribution function (RDF) or pair correlation function *g(r)*.[142] This distance represents the least probable radius to find occupied, located between the nearest and next-nearest neighbors. In an FCC metal this distance is between $a_0\sqrt{2}/2$ and a_0, where a_0 is the lattice constant. Finding the RDF is usually performed over an entire simulation collecting data all the while, to more adequately supply its function with a statistic distribution.

The xyz output file appends the CN, CM, and CV after each line, see Table X for a sample of this file. For VMD to show different CNs from the "xyz" file, each value of CN is assigned to an element, for example if a particle has a CN equal to 1, it would be represented by Hydrogen, and if it had a CN of 13 it would be shown as Aluminum.

There is a CN summary output to stderr, an example of which is shown in Table XI and two density files are created with extensions "CNdens" and "densV" the former is the CN density across a given dimension, this dimension can be specified with the "-d" switch on the command line, an example of this output can be seen in Table XII, and the latter, the average CV across the same given dimension. This can be used to search for structural irregularities such as voids and pores. A sample output file is listed in Table XIII. All output files can be used with "gnuplot".

Table X. Sample “xyz” File

The first number is the number of particles. The columns are: (1) The elemental representation of the particle's CN. (2-4) the position. (5) particle scale. (6) Particle's atomic number, its real element type. (7) CN. (8) CM. (9-11) CV.

----------0005000.MD2.xyz----------

```
        9600

  C  -19.740402  -19.944939  -24.442770   1  13   6    0.358683    0.000014   -0.000043   -0.358683
 Be  -18.985060  -21.253231  -25.818981   1   8   4    0.391581   -0.195795    0.339117   -0.000011
 Be  -18.985058  -18.636736  -25.818981   1   8   4    0.391511   -0.195797   -0.339035   -0.000011
  C  -19.740402  -19.944939  -17.707396   1  13   6    0.358684    0.000014   -0.000043   -0.358684
  C  -19.740402  -19.944939  -10.972019   1  13   6    0.358683    0.000014   -0.000043   -0.358683
  C  -19.740402  -19.944939  -20.459738   1  13   6    0.358662    0.000124   -0.000042    0.358662
 Be  -18.229767  -19.944939  -19.083609   1   8   4    0.391529   -0.391529   -0.000017    0.000078
 Be  -18.985060  -21.253231  -12.348231   1   8   4    0.391581   -0.195795    0.339117    0.000061
```

Table XI. CN Summary Output on Stderr

There are four sets of columns, the first is the total model statistics, the last three are statistics about each scale from the atomic, S1, to S3. The last two groups show a “0” and “NaN” because this example model has only scale-1 particles.

Each column group lists the CN frequency or actual number of particles with that CN, and its percentage. The very first column is the CN corresponding to the second column which shows the element type that is given for that CN. A careful look at these numbers can determine whether the model is amorphous or crystalline.

----------0005000.MD2.CNdat----------

```
Using total options:
number of particles per scale:         6911           0           0
total:        6911
number of particles per scale:         6911           0           0
total real particles:        6911
 CN  elem  freq    %       S1 freq            S2 freq            S3 freq
                          Rcut=   3.4733405       6.9466810       13.893362
  1    H      0   0.000000      0   0.000000      0        NaN      0        NaN
  2   He      0   0.000000      0   0.000000      0        NaN      0        NaN
```

3	Li	3	0.043409	3	0.043409	0	NaN	0	NaN
4	Be	1	0.014470	1	0.014470	0	NaN	0	NaN
5	B	89	1.287802	89	1.287802	0	NaN	0	NaN
6	C	37	0.535378	37	0.535378	0	NaN	0	NaN
7	N	6	0.086818	6	0.086818	0	NaN	0	NaN
8	O	1301	18.825062	1301	18.825062	0	NaN	0	NaN
9	F	148	2.141514	148	2.141514	0	NaN	0	NaN
10	Ne	3	0.043409	3	0.043409	0	NaN	0	NaN
11	Na	12	0.173636	12	0.173636	0	NaN	0	NaN
12	Mg	5208	75.358124	5208	75.358124	0	NaN	0	NaN
13	Al	102	1.475908	102	1.475908	0	NaN	0	NaN
14	Si	1	0.014470	1	0.014470	0	NaN	0	NaN

Table XII. CN Density Across the Model

The first column is position; the remaining columns are the percentage of each CN. Column 2 is for CN=1, column 3 for CN=2, etc.

----------0005000.MD2.CNdens----------

```
 # Pos(,i)  CN=1 CN=2 CN=3 ... CN=20 CN=0&CN>20
   -19.88149      0.00000      0.00000      0.00180      0.00180      0.04317      0.02878
0.00719       0.41727       0.07374       0.00180       0.00000       0.41547       0.00899
0.00000     ...
   -18.86193      0.00000      0.00000      0.00000      0.00000      0.00000      0.05000
0.05000       0.40000       0.15000       0.05000       0.00000       0.30000       0.00000
0.00000     ...
   -17.84237      0.00000      0.00000      0.00000      0.00000      0.00694      0.00000
0.00000       0.09722       0.05556       0.00000       0.00000       0.73264       0.10764
0.00000     ...
   -15.80324      0.00000      0.00000      0.00000      0.00000      0.00347      0.00347
0.00000       0.13542       0.01736       0.00000       0.00000       0.82639       0.01389
0.00000     ...
   -13.76411      0.00000      0.00000      0.00000      0.00000      0.00347      0.00347
0.00000       0.13889       0.01389       0.00000       0.00000       0.82986       0.01042
0.00000     ...
   -11.72498      0.00000      0.00000      0.00000      0.00000      0.00694      0.00000
0.00000       0.14931       0.00347       0.00000       0.00000       0.83681       0.00347
0.00000     ...
   -10.70542      0.00000      0.00000      0.00000      0.00000      0.00347      0.00347
0.00000       0.14583       0.00694       0.00000       0.00000       0.83681       0.00347
0.00000     ...
    -8.66629      0.00000      0.00000      0.00000      0.00000      0.00347      0.00347
0.00000       0.13889       0.01389       0.00000       0.00000       0.82986       0.01042
0.00000     ...
```

```
    -6.62716      0.00000      0.00000      0.00000      0.00000      0.00694      0.00000
0.00000       0.14236       0.01042       0.00000       0.00000       0.82986       0.01042
0.00000     ...
```

Table XIII. Coordination Vector Across the Model

The first column is the position; the last three are the Coordination Vector (CV).

----------0005000.MD2.densV----------

# Pos(,i)	average Coordination Vector (X,Y,Z)		
-19.88149	2128.21167	-1.04622	-11.24967
-18.86193	3.91419	-1.46443	7.96096
-17.84237	-117.07870	0.58429	1.35901
-15.80324	-3.73650	0.53191	1.16895
-13.76411	-4.46231	0.88720	0.42965
-11.72498	-3.34580	0.01192	0.52085
-10.70542	-2.39000	-0.00161	-0.74924
-8.66629	0.15434	0.53781	0.56996
-6.62716	-0.41944	0.78974	-0.40410

c. Coordination Vector (CV)

CN is a useful quantity for identifying many attributes about a system. For example, it can identify surfaces and edges of a model, vacancies, voids, dislocations and other structural deviations. Some limitations come to mind when thinking about the CN as a measure of local density. For example, CNs are scalar and integers; there exists a vector quantity that shall be termed the Coordination Vector (CV). This vector is defined by the sum of all the atomic distances of the neighbors that were counted to equal the CN.

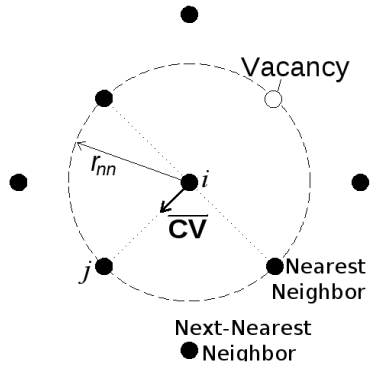


Figure 66. $\overline{CV}$ concept for atom *i* with a vacancy.

The Coordination Vector, $\overline{CV}$, for atom *i*, is related to its coordination number (CN) within its NN radius, r_{nn}. It is defined as a non-dimensional vector by an average covering all its NN atoms (i.e., j=1, 2…CN). The $\overline{CV}'s$ geometric and vector expressions of atom *i* are shown in Fig. 66 in terms of the position of the vacancy, neighbor atom *j*, and r_{nn}. Its mathematical definition is given as follows:

$$\overline{CV} \equiv \frac{1}{r_{nn}.CN}\sum_{j=1}^{CN}(\bar{r}_j - \bar{r}_i) \tag{59}$$

It is easy to see the following properties of the Coordination Vector, $\overline{CV}$:

I If there is no vacancy, $\overline{CV} = 0$

II $\overline{CV}$ points in the direction of the largest density of NN atoms (or particles) and thus quantifies the asymmetry of the NN atom distribution around atom *i*.

III An atom with a larger CV value indicates more vacancies on its one side, thus it may have a greater probability to be close to a pore or be a site for tiny pore nucleation.

d. Near-Neighbor Grouping (NNg)

Identifying atoms that might be at the surface of a void or vacancy is very useful, however rather difficult to use in a meaningful way. What would be desirable would be to filter out all of the atoms that have large CMs and group them according to the void they are affected by. The question then becomes how to group them? Or what criteria to use

that would specify whether an atom belongs to one group over another? If the large CM is not just a single phenomenon, there should be others beside it. In the case of a surface, all of the surface atoms could be nearest neighbors of each other, and if connected would create a sheet. In this manner groups represent continuous surfaces of a model. Discontinuities may be caused by an atom having unusual geometric symmetry near the surface.

The method used to group points together, if the points are close enough, is called a *connectivity-based single-link agglomerating cluster algorithm*.[148] These methods are used when the clusters are based on connectivity and distance.[134] In this case the distance is the same as used for determining CN, i.e. the distance to the first minimum of the radial distribution function, or about half way between the nearest and next-nearest neighbors. The agglomerative type of algorithm starts by assigning every atom to its own cluster and gathers all of the atoms that are connected to it. It does this by using a nearest-neighbor list and reassigning its neighbors and every atom in its neighbor's cluster to its own cluster. In this way a Nearest-Neighbor grouping (NNg) algorithm is employed. Many times, there will be numerous remaining clusters that contain only one or two atoms, these clusters are neglected for most of the further analysis. Since a single vacancy ideally has about twelve surface atoms, all clusters containing twelve atoms or more are filtered for further analysis. Figure 67 shows steps on how this is accomplished.

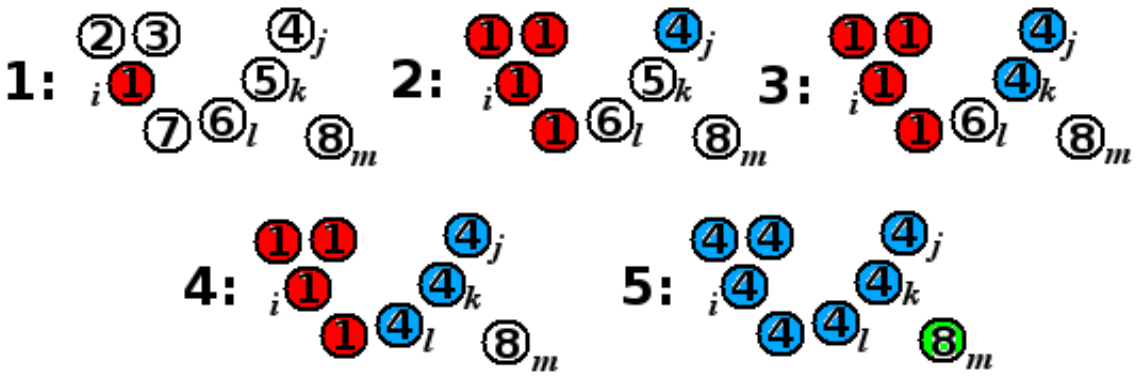


Figure 67. Five step process of grouping particles by a single-link method.

Not all of the atoms with large CM values will be part of a vacancy or void surface, but some may be the surface of a dense object, such as a particulate or surface of

a larger massive object. The quantitative difference lies in the direction of the CVs in the cluster or group. A method was developed to compare each group member's CV in relation to the geometric center of the group. The CV was projected onto the difference vector from the center to the group member; these projections are all averaged together from each group member. Theoretically, if this value is negative then the density is located outside of the group (fig. 68b), which represents the group as a void surface, and if it's positive, the density is inside of the group (fig. 68a), representing a dense volume or possible particulate. These two types of groups are filtered and assigned element types to be viewed in VMD, all other atoms are assigned the element type: “Na”, giving rise to only three different colors for identifying types of groups or clusters.

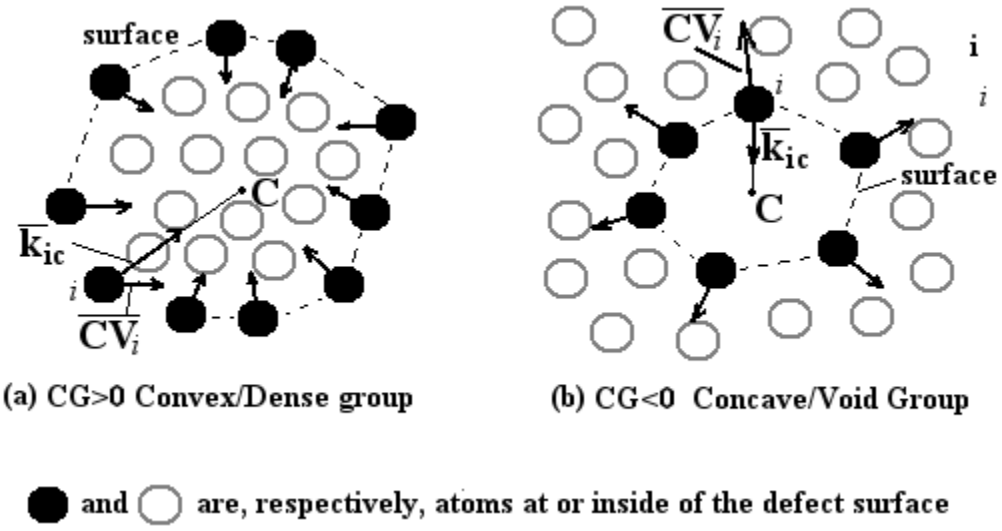
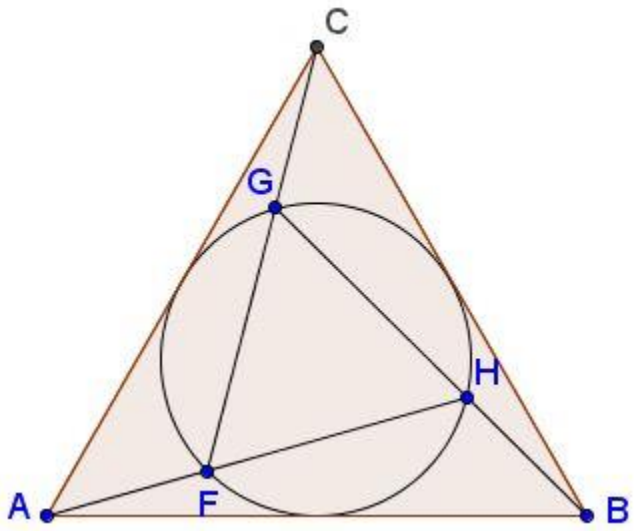


Figure 68. Two types of clusters, (a) a dense group or (b) a Void group.


C
G
H
A
F
B