AN ABSTRACT OF THE THESIS OF

Timothy B. Costa for the degree of Doctor of Philosophy in Mathematics presented on May 31, 2016.

Title: Hybrid Multiscale Methods with Applications to Semiconductors, Porous Media, and Materials Science

Abstract approved: Malgorzata S. Peszynska

In this work we consider two multiscale applications with tremendous computational complexity at the lower scale. First, we examine a model for charge transport in semiconductor structures with heterojunction interfaces. Due to the complex physical phenomena at the interface, the model at the design scale is unable to adequately capture the behavior of the structure in the interface region. Simultaneously it is computationally intractable to simulate the full heterostructure on the scale required near the interface. Second, we consider the problem of the simulation of fluid flow in a dynamically evolving porous medium. The evolution of the medium strongly couples the porescale flow solutions and the macroscale model, requiring a novel approach to communicate the porescale evolution to the macroscale without resorting to the intractable simulation of the fluid flow problem directly on the porescale geometry. We formulate novel methods for these two applications in the multiscale framework. For the semiconductor problem we present iterative substructuring domain decomposition methods that decouple the interface computation from the macroscale model. For the fluid flow problem we develop a reduced order three-scale fluid flow model based on a spatial decomposition of the porescale geometry and the offline approximation of a stochastic process describing macroscale permeability parameterized by the volume fraction of the evolved geometry.
Hybrid Multiscale Methods with Applications to Semiconductors, Porous Media, and Materials Science

by

Timothy B. Costa

A THESIS

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Presented May 31, 2016
Commencement June 2016
Doctor of Philosophy thesis of Timothy B. Costa presented on May 31, 2016

APPROVED:

______________________________
Major Professor, representing Mathematics

______________________________
Chair of the Department of Mathematics

______________________________
Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

______________________________
Timothy B. Costa, Author
I’ll begin by thanking my advisor, Dr. Malgorzata Peszynska. There is no doubt that my successes in graduate school were due in no small part to her guidance and model of persistent enthusiasm for applied and computational mathematics. I would also like to thank my committee members Dr. Ralph Showalter, Dr. Robert Higdon, Dr. Mina Ossiander and Dr. Dorthe Wildenschild.

Further I’d like to thank Dr. Anthony Costa, Dr. Justin Webster and Dr. Duncan McGregor for many useful discussions throughout graduate school, without which my research progress would certainly have been slower.

I’d also like to acknowledge Dr. David Foster and Dr. Guenter Schneider from the Department of Physics at Oregon State for their collaboration on the semiconductor portion of this thesis.

My time at Sandia National Laboratories, while not directly related to this thesis, was nonetheless instrumental in my development as a researcher. I’d like to thank Dr. David Littlewood and Dr. Veena Tikare for providing me the opportunity to participate in research at the laboratory, as well as Dr. Stephen Bond for many productive conversations and collaborations.

The research in this work was funded by the National Science Foundation (NSF) on several grants. The heterostructure semiconductor work was funded by NSF-DMS 1035513, “Enhanced Photovoltaic Efficiency Through Heterojunction Assisted Impact Ionization,” (Principal Investigator Stephen Kevan, Co-Principal Investigator Malgorzata Peszynska, Co-Principal Investigator Geraldine Richmond). The work in modeling fluid flow in an evolving porous media was supported by NSF-DMS 1115827, “Hybrid Modeling in Porous Media,” (Principal Investigator Malgorzata Peszynska) and NSF-DMS 1522734,
"Phase Transitions in Porous Media Across Multiple Scales,” (Principal Investigator Malgorzata Peszynska).

I’d also like to thank my parents, John and Denise Costa. None of this work would have been possible without decades of their love and support.

Finally, to those to whom this thesis is dedicated. Thank you to my wife Jen, and my daughters Emily and Olivia for years of patience, love and support before and throughout graduate school. I can’t wait to start the next chapter of our lives together.
# TABLE OF CONTENTS

1. INTRODUCTION ................................................................. 1

2. MATHEMATICAL PRELIMINARIES ........................................ 7
   2.1. Functional Analysis.................................................... 7
       2.1.1. Lebesgue and Sobolev Spaces .............................. 7
       2.1.2. Banach and Hilbert Space Methods ..................... 12
   2.2. Probability ............................................................. 17
   2.3. Numerical Analysis ................................................. 25
       2.3.1. Finite Difference Method ................................. 27
       2.3.2. Finite Volume Method ...................................... 30
       2.3.3. Richardson Iteration ...................................... 36
       2.3.4. Krylov Methods ............................................. 39
       2.3.5. Preconditioners ............................................ 44

3. CHARGE TRANSPORT IN HETEROSTRUCTURE SEMICONDUCTORS . 58
   3.1. Modeling Charge Transport Across Heterojunction Interfaces . 61
       3.1.1. Drift-Diffusion ............................................. 61
       3.1.2. Thermionic Emission ...................................... 62
       3.1.3. Analysis: Single Material Domain ..................... 64
   3.2. Domain Decomposition ............................................. 69
       3.2.1. Mathematical Background .................................. 69
       3.2.2. DDM for Potential Equation ............................. 73
       3.2.3. DDM for Carrier Transport .............................. 83
       3.2.4. Heterojunction Model Analysis .......................... 98
   3.3. Heterojunction Uncertainty ..................................... 111
   3.4. Numerical Results ............................................... 113
       3.4.1. Algorithm DDP ............................................. 113
       3.4.2. Algorithm DDC ............................................ 117
       3.4.3. Algorithm DDN ............................................ 118
       3.4.4. Heterostructure Semiconductors ....................... 121
### TABLE OF CONTENTS (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.5. Angstrom Scale Sensitivity Analysis</td>
<td>124</td>
</tr>
<tr>
<td>3.5. Conclusions</td>
<td>128</td>
</tr>
<tr>
<td><strong>4. MODELING FLUID FLOW IN COMPLEX MEDIA</strong></td>
<td>130</td>
</tr>
<tr>
<td>4.1. Fluid Flow Models</td>
<td>133</td>
</tr>
<tr>
<td>4.1.1. Fluid Flow Models</td>
<td>133</td>
</tr>
<tr>
<td>4.1.2. Multiscale Fluid Flow</td>
<td>138</td>
</tr>
<tr>
<td>4.2. Fluid Flow in a Stochastic Geometry</td>
<td>143</td>
</tr>
<tr>
<td>4.2.1. Literature Remarks</td>
<td>144</td>
</tr>
<tr>
<td>4.2.2. Stochastic Immersed Boundary Stokes Model</td>
<td>145</td>
</tr>
<tr>
<td>4.3. Three-scale Fluid Flow Methods</td>
<td>149</td>
</tr>
<tr>
<td>4.3.1. Direct Three-scale Method</td>
<td>149</td>
</tr>
<tr>
<td>4.3.2. Reduced Order Three-scale Fluid Flow Method</td>
<td>156</td>
</tr>
<tr>
<td>4.4. Software: HybGe-Flow3d</td>
<td>160</td>
</tr>
<tr>
<td>4.5. Numerical Results</td>
<td>165</td>
</tr>
<tr>
<td>4.5.1. Upscaling with Porescale Evolution</td>
<td>165</td>
</tr>
<tr>
<td>4.5.2. Model 2D Geometry</td>
<td>169</td>
</tr>
<tr>
<td>4.5.3. Sandstone</td>
<td>175</td>
</tr>
<tr>
<td>4.7. Conclusions</td>
<td>192</td>
</tr>
<tr>
<td><strong>BIBLIOGRAPHY</strong></td>
<td></td>
</tr>
<tr>
<td><strong>APPENDICES</strong></td>
<td>204</td>
</tr>
<tr>
<td>A. Density Functional Theory</td>
<td>205</td>
</tr>
<tr>
<td>B. HybGe-Flow3D User Manual, V1.0.1</td>
<td>207</td>
</tr>
<tr>
<td>B1. HybGe-Flow3D Overview</td>
<td>207</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>B2. Installation</td>
<td>208</td>
</tr>
<tr>
<td>B3. Running HybGe-Flow3D</td>
<td>215</td>
</tr>
<tr>
<td>C Geometry.dat</td>
<td>215</td>
</tr>
<tr>
<td>D Hanford Sandstone Permeability Distributions</td>
<td>220</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Illustration of cell indices for finite volume discretization</td>
</tr>
<tr>
<td>2.2</td>
<td>Sparsity pattern, finite volume discretization of the Laplacian in 3d</td>
</tr>
<tr>
<td>2.3</td>
<td>Staggered grid illustration in $\mathbb{R}^3$</td>
</tr>
<tr>
<td>2.4</td>
<td>Sparsity pattern, finite volume discretization of the Stokes’ equations</td>
</tr>
<tr>
<td>2.5</td>
<td>ILU(0) sparsity patterns for 5-point Laplacian matrix, image credit [[95] p. 308]</td>
</tr>
<tr>
<td>2.6</td>
<td>ILU(1) sparsity patterns for 5-point Laplacian matrix, image credit [[95] p.312]</td>
</tr>
<tr>
<td>3.1</td>
<td>Top: schematic plot of potential across 1D interface region. Bottom: interface region shrunk to a point resulting in discontinuous potential</td>
</tr>
<tr>
<td>3.2</td>
<td>Illustration of domain division.</td>
</tr>
<tr>
<td>3.3</td>
<td>Illustration of boundary setup in 2d for heterojunction model analysis</td>
</tr>
<tr>
<td>3.4</td>
<td>Solution to 1d model problem with algorithm DDP.</td>
</tr>
<tr>
<td>3.5</td>
<td>Convergence of discrete estimate $\theta^{\psi,h}_{\max}$ as $h \to 0$. $N_1 = N_2$ denotes the number of grid points in each domain.</td>
</tr>
<tr>
<td>3.6</td>
<td>Algorithm DDP residual reduction for various $\theta$.</td>
</tr>
<tr>
<td>3.7</td>
<td>Solution to model problem for algorithm DDC.</td>
</tr>
<tr>
<td>3.8</td>
<td>Convergence of discrete estimate $\theta^{u,h}_{\max}$ as $h \to 0$. $N_1 = N_2$ denotes the number of grid points in each domain.</td>
</tr>
<tr>
<td>3.9</td>
<td>Algorithm DDC residual reduction for various $\theta$.</td>
</tr>
<tr>
<td>3.10</td>
<td>Algorithm DDC solves problem algorithm DDN cannot.</td>
</tr>
<tr>
<td>3.11</td>
<td>CdS/CIGS heterostructure with algorithms DDP and DDC.</td>
</tr>
<tr>
<td>3.12</td>
<td>$J_{sc}$ against $V_{oc}$ for the ZnS/Si heterojunction at each collocation point</td>
</tr>
<tr>
<td>3.13</td>
<td>$J_{sc}$ against $V_{oc}$ for CdS/CIGS heterojunction at each collocation point</td>
</tr>
<tr>
<td>4.1</td>
<td>Illustration of pore-network representation of a simple 2d porous medium.</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.2</td>
<td>Pore-network representation of sandstone geometry</td>
</tr>
<tr>
<td>4.3</td>
<td>Illustration of geometry in 2d.</td>
</tr>
<tr>
<td>4.4</td>
<td>Illustration of volume averaging in 2d.</td>
</tr>
<tr>
<td>4.5</td>
<td>Illustration of domain division for $d = 2$.</td>
</tr>
<tr>
<td>4.6</td>
<td>Illustration of a porescale subdivision into throats and their connected</td>
</tr>
<tr>
<td></td>
<td>pores according to pore-network model.</td>
</tr>
<tr>
<td>4.7</td>
<td>Pressure and $x$-component velocity of flow around a sphere.</td>
</tr>
<tr>
<td>4.8</td>
<td>Porescale pressure and $x$-component velocity in a synthetic glassbead</td>
</tr>
<tr>
<td></td>
<td>geometry</td>
</tr>
<tr>
<td>4.9</td>
<td>Porescale pressure solution in an imaged glassbead geometry</td>
</tr>
<tr>
<td>4.10</td>
<td>(TOP) $\kappa$ as a function of the placement of a 5x5 object in void space (BOTTOM) $\kappa$ compared to velocity magnitude at location where object is placed - from Section 4.5.1.</td>
</tr>
<tr>
<td>4.11</td>
<td>$\kappa$ as a function of the placement of a 10x10 object in void space (BOTTOM) $\kappa$ compared to velocity magnitude at location where object is placed - from Section 4.5.1.</td>
</tr>
<tr>
<td>4.12</td>
<td>$\kappa$ as a function of the placement of a 15x15 object in void space (BOTTOM) $\kappa$ compared to velocity magnitude at location where object is placed - from Section 4.5.1.</td>
</tr>
<tr>
<td>4.13</td>
<td>$\kappa$ as a function of the placement of a 20x20 object in void space (BOTTOM) $\kappa$ compared to velocity magnitude at location where object is placed - from Section 4.5.1.</td>
</tr>
<tr>
<td>4.14</td>
<td>Flow in an idealized porous geometry in 2D from Section 4.5.2.</td>
</tr>
<tr>
<td>4.15</td>
<td>Full domain with material growth from Section 4.5.2.</td>
</tr>
<tr>
<td>4.16</td>
<td>Simple throat subdomain of idealized porous medium from Section 4.5.2.</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.17</td>
<td>Empirical probability density functions (top) and cumulative distribution functions (bottom) for experiments on simple throat geometry from Section 4.5.2.</td>
</tr>
<tr>
<td>4.18</td>
<td>Example domain with $N = 5$ growths from Section 4.5.2.</td>
</tr>
<tr>
<td>4.19</td>
<td>Pressure solution for a single realization of pore-network model for the modified domain from Section 4.5.2.</td>
</tr>
<tr>
<td>4.20</td>
<td>Empirical probability density function for $\kappa$ on the modified domain, computed with the synthetic structured pore-network model, which samples, for each throat, the porescale empirical distributions based on the amount of growth present from Section 4.5.2.</td>
</tr>
<tr>
<td>4.21</td>
<td>Porescale velocity magnitude for sandstone geometry from Section 4.5.3.</td>
</tr>
<tr>
<td>4.22</td>
<td>Empirical densities, sandstone geometry subdomain 0 from Section 4.5.3.</td>
</tr>
<tr>
<td>4.23</td>
<td>Empirical densities, sandstone geometry subdomain 4 from Section 4.5.3.</td>
</tr>
<tr>
<td>4.24</td>
<td>Empirical densities, sandstone geometry subdomain 9 from Section 4.5.3.</td>
</tr>
<tr>
<td>4.25</td>
<td>Empirical densities, sandstone geometry subdomain 17 from Section 4.5.3.</td>
</tr>
<tr>
<td>4.26</td>
<td>Velocity magnitude in modified (obstacles in green) sandstone geometry from Section 4.5.3.</td>
</tr>
<tr>
<td>4.27</td>
<td>Growth pattern (in green) for modified sandstone geometry from Section 4.5.3.</td>
</tr>
<tr>
<td>4.28</td>
<td>Empirical probability density function for $\kappa$ on the modified sandstone geometry computed with pore-network model, which samples, for each throat, the porescale empirical distributions based on the volume of obstacles present from Section 4.5.3.</td>
</tr>
<tr>
<td>4.29</td>
<td>Pressure solution for a single realization of the pore-network model for the modified sandstone geometry from Section 4.5.3.</td>
</tr>
<tr>
<td>4.30</td>
<td>Snapshots of biofilm growth from Section 4.6.</td>
</tr>
<tr>
<td>4.31</td>
<td>Illustration of PCA results for biofilm growth from Section 4.6.</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>0.32</td>
<td>Empirical densities, sandstone geometry subdomain 1 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.33</td>
<td>Empirical densities, sandstone geometry subdomain 3 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.34</td>
<td>Empirical densities, sandstone geometry subdomain 5 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.35</td>
<td>Empirical densities, sandstone geometry subdomain 6 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.36</td>
<td>Empirical densities, sandstone geometry subdomain 7 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.37</td>
<td>Empirical densities, sandstone geometry subdomain 8 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.38</td>
<td>Empirical densities, sandstone geometry subdomain 10 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.39</td>
<td>Empirical densities, sandstone geometry subdomain 11 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.40</td>
<td>Empirical densities, sandstone geometry subdomain 12 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.41</td>
<td>Empirical densities, sandstone geometry subdomain 13 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.42</td>
<td>Empirical densities, sandstone geometry subdomain 14 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.43</td>
<td>Empirical densities, sandstone geometry subdomain 15 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.44</td>
<td>Empirical densities, sandstone geometry subdomain 16 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.45</td>
<td>Empirical densities, sandstone geometry subdomain 18 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.46</td>
<td>Empirical densities, sandstone geometry subdomain 19 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.47</td>
<td>Empirical densities, sandstone geometry subdomain 20 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.48</td>
<td>Empirical densities, sandstone geometry subdomain 21 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.49</td>
<td>Empirical densities, sandstone geometry subdomain 22 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.50</td>
<td>Empirical densities, sandstone geometry subdomain 23 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.51</td>
<td>Empirical densities, sandstone geometry subdomain 24 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.52</td>
<td>Empirical densities, sandstone geometry subdomain 26 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.53</td>
<td>Empirical densities, sandstone geometry subdomain 27 from Section 4.5.3.</td>
</tr>
<tr>
<td>0.54</td>
<td>Empirical densities, sandstone geometry subdomain 28 from Section 4.5.3.</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>0.55</td>
<td>244</td>
</tr>
<tr>
<td>0.56</td>
<td>245</td>
</tr>
<tr>
<td>0.57</td>
<td>246</td>
</tr>
<tr>
<td>0.58</td>
<td>247</td>
</tr>
<tr>
<td>0.59</td>
<td>248</td>
</tr>
<tr>
<td>0.60</td>
<td>249</td>
</tr>
<tr>
<td>0.61</td>
<td>250</td>
</tr>
<tr>
<td>0.62</td>
<td>251</td>
</tr>
<tr>
<td>0.63</td>
<td>252</td>
</tr>
<tr>
<td>0.64</td>
<td>253</td>
</tr>
<tr>
<td>0.65</td>
<td>254</td>
</tr>
<tr>
<td>0.66</td>
<td>255</td>
</tr>
<tr>
<td>0.67</td>
<td>256</td>
</tr>
<tr>
<td>0.68</td>
<td>257</td>
</tr>
<tr>
<td>0.69</td>
<td>258</td>
</tr>
<tr>
<td>0.70</td>
<td>259</td>
</tr>
<tr>
<td>0.71</td>
<td>260</td>
</tr>
<tr>
<td>0.72</td>
<td>261</td>
</tr>
<tr>
<td>0.73</td>
<td>262</td>
</tr>
<tr>
<td>0.74</td>
<td>263</td>
</tr>
<tr>
<td>0.75</td>
<td>264</td>
</tr>
<tr>
<td>0.76</td>
<td>265</td>
</tr>
<tr>
<td>0.77</td>
<td>266</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.78</td>
<td>Empirical densities, sandstone geometry subdomain 53 from Section 4.5.3.</td>
<td>267</td>
</tr>
<tr>
<td>0.79</td>
<td>Empirical densities, sandstone geometry subdomain 54 from Section 4.5.3.</td>
<td>268</td>
</tr>
<tr>
<td>0.80</td>
<td>Empirical densities, sandstone geometry subdomain 55 from Section 4.5.3.</td>
<td>269</td>
</tr>
<tr>
<td>0.81</td>
<td>Empirical densities, sandstone geometry subdomain 56 from Section 4.5.3.</td>
<td>270</td>
</tr>
<tr>
<td>0.82</td>
<td>Empirical densities, sandstone geometry subdomain 57 from Section 4.5.3.</td>
<td>271</td>
</tr>
<tr>
<td>0.83</td>
<td>Empirical densities, sandstone geometry subdomain 58 from Section 4.5.3.</td>
<td>272</td>
</tr>
<tr>
<td>0.84</td>
<td>Empirical densities, sandstone geometry subdomain 59 from Section 4.5.3.</td>
<td>273</td>
</tr>
<tr>
<td>0.85</td>
<td>Empirical densities, sandstone geometry subdomain 60 from Section 4.5.3.</td>
<td>274</td>
</tr>
<tr>
<td>0.86</td>
<td>Empirical densities, sandstone geometry subdomain 61 from Section 4.5.3.</td>
<td>275</td>
</tr>
<tr>
<td>0.87</td>
<td>Empirical densities, sandstone geometry subdomain 62 from Section 4.5.3.</td>
<td>276</td>
</tr>
<tr>
<td>0.88</td>
<td>Empirical densities, sandstone geometry subdomain 63 from Section 4.5.3.</td>
<td>277</td>
</tr>
<tr>
<td>3.1</td>
<td>Algorithm DDP convergence.</td>
<td>115</td>
</tr>
<tr>
<td>3.2</td>
<td>Algorithm DDP mesh independence, ( \theta = 0.25 ).</td>
<td>115</td>
</tr>
<tr>
<td>3.3</td>
<td>Algorithm DDC convergence.</td>
<td>118</td>
</tr>
<tr>
<td>3.4</td>
<td>Algorithm DDC mesh independence, ( \theta = 0.335 ).</td>
<td>118</td>
</tr>
<tr>
<td>3.5</td>
<td>Algorithm DDN convergence.</td>
<td>120</td>
</tr>
<tr>
<td>3.6</td>
<td>Algorithm DDN mesh independence, ( \theta = 0.25 ).</td>
<td>120</td>
</tr>
<tr>
<td>3.7</td>
<td>Material parameter values</td>
<td>122</td>
</tr>
<tr>
<td>3.8</td>
<td>Interface and structure parameter values</td>
<td>122</td>
</tr>
<tr>
<td>3.9</td>
<td>DDM iterations at each Gummel step for the CdS/CIGS heterostructure.</td>
<td>124</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.10</td>
<td>Uncertainty study: ZnS/Si</td>
<td>125</td>
</tr>
<tr>
<td>3.11</td>
<td>Uncertainty study: CdS/CIGS</td>
<td>125</td>
</tr>
<tr>
<td>3.12</td>
<td>Means and variances for heterojunction parameter uncertainty studies</td>
<td>126</td>
</tr>
<tr>
<td>4.1</td>
<td>Permeabilities ($\times 10^{-3}(\mu m)^2$) in direct subdivision strategy from Section 4.5.2</td>
<td>172</td>
</tr>
</tbody>
</table>
TO JEN, EMILY AND OLIVIA
1. INTRODUCTION

Human progress is strongly coupled to understanding of the physical universe. Scientific discovery fuels innovation and progress across all facets of human civilization. A fundamental piece of scientific discovery is experimentation. We learn about the universe by formulating hypotheses and testing them. As the questions we ask become larger and more complicated, the cost associated to answering these questions with physical experimentation likewise becomes greater and greater, to the point that full physical experimentation has become cost prohibitive in many important areas. Physical experimentation is also often destructive, and sometimes dangerous.

Scientific computing offers an alternative to physical experimentation with several important advantages; most notably significantly lower cost, non-destructive experimentation, and the ability to experiment with dangerous materials at no risk to the scientist or the surrounding community. The aerospace community, for example, long ago accepted computational fluid dynamics as the first experimentation grounds for the design of better airplanes, thus drastically reducing the cost associated to testing of new ideas compared to building all prototypes for testing in wind tunnels. Moreover, problems in subsurface resource extraction, groundwater remediation, or storage in the subsurface often exist on scales too large for consideration in a physical laboratory. Certainly important questions can be asked and answered with physical experimentation on smaller scales for these
problems, but consideration of the ‘engineering scale’ can only be accomplished in silico.

Despite the tremendous power of scientific computing to ask and answer important questions problematic for physical experimentation, human society remains ready to ask questions larger still, at the very edge of, and well beyond, what modern computers are capable of simulating. Detailed three-dimensional simulations of material behavior and subsurface flow and transport often result in computational models comprised of millions or billions of degrees of freedom occupying immense amounts of memory and requiring very large computational time, often beyond the physical resources of the available computer and beyond the time-frame to solution required by the researcher. For the community of researchers designing and delivering computers the answer is clear: larger and more capable hardware. But for the computational scientist working on today’s hardware and needing answers to today’s questions this answer is woefully inadequate. Further, history is clear that scientists have no shortage of larger problems to tackle as the hardware becomes more and more capable. Thus the mathematical models and numerical methods used to attack the problems need their own examination in order to bring larger problems into focus on today’s resources as well as tomorrow’s.

Historically, the answer to the problem of insufficient computational power for investigation of a mathematical model describing physical phenomena has been to trade resolution for scale. This phrasing encompasses both the need to solve for fewer degrees of freedom within a given model (coarsen the resolution within a computational model at a given scale), as well as the need to trade models entirely for a mathematical description of larger scale phenomena (change scales) which sacrifices fine scale accuracy. For example, in the porous media community, laminar flow in the subsurface is accurately modeled by the Stokes equations, which explicitly resolve the geometry of every grain of sand and piece of rock and model the fluid’s pressure and velocity in the space between this solid matrix. This level of detail is a tremendous burden for simulations of large sections of the
Subsurface, and the typical strategy is to instead model the fluid’s velocity and pressure at a higher scale with Darcy’s model, which averages the geometric detail into a tensor describing the freedom for fluid to travel through the geometry. Similar strategies exist in other application areas. It is from within this general background that multiscale methods appear.

In some sense the field of multiscale methods is a direct nod to the limitations of scientific computing power. With unlimited resources, in many applications one would simply solve the finest scale model available to obtain the most accurate solution. Methods that adaptively track features do so to avoid resolving the entire computational domain to the level required near the feature. Methods that compute effective upscaled parameters typically do so because the fine-scale model is intractable, and so accuracy is traded for feasibility. On the other hand, the field of multiscale methods is a celebration of the successful endeavor to increase the power of modern computing systems, and the certainty that this success will continue. From this standpoint, a multiscale method takes advantage of evolving and increasingly powerful computing architectures to perform simulations of increasing fidelity by including lower scale phenomena into the engineering scale simulation.

In this work we consider two application areas with tremendous computational overhead at the lower scale. First, we examine a model for charge transport in a heterostructure semiconductor; that is, a semiconductor structure comprised of two distinct semiconductor materials connected by an interface, referred to as a heterojunction. Due to the physics at the interface the models at the 'engineering scale,' heretofore referred to as the continuum scale, are unable to adequately capture the behavior of the structure in the interface region. Simultaneously it is a hopeless endeavor to model, computationally, the behavior on the full structure at the scale required near the interface. Thus we model the full structure in a multiscale framework, capturing the necessary physics at the angstrom scale near the
interface and including this information in the continuum scale model for full device simulation. The resulting continuum scale model presents tremendous challenges numerically, and so we formulate and present novel domain decomposition methods which decouple the troublesome interface model from the bulk simulations and provide a robust method for simulating the multiscale heterostructure model.

We then turn our attention to the simulation of flow and transport in evolving porous media. Here we are interested in simulating fluid flow in a porous media that is dynamically evolving according to some reactive transport process within the fluid. This may be due to, e.g. biofilm growth, the formulation of methane hydrate, or reactive transport with precipitation - dissolution. In this setting there is a dynamic coupling between the porescale fluid and pressure and higher scale models through the permeability tensor describing the capacity of fluid to move through the geometry. As the reactive transport proceeds and the geometry evolves, the most accurate model would involve solving for the porescale fluid and pressure on the computational domain and computing an upscaled permeability tensor for the higher scale models at each time step, or whenever the geometry has changed significantly enough. To make matters worse, the appropriate time scale for flow and transport simulations in porous media is on the order of 1 second or less, while the integration time to study biofilm growth or other reactive processes is on the order of several days. This problem – wedged between constraints of computational feasibility and active processes at the porescale – necessarily requires a novel multiscale strategy. To address these difficulties we present a new three-scale method based on spatial decomposition of the porescale domain together with a reduced order offline/online strategy describing the response of the locally defined permeability tensor to the reactive transport process. This strategy results in modeling of the permeability tensor as a conditional random vector, and the formulation of the offline porescale problem on a stochastic geometry.
In order to encompass more general physics in the porescale evolution we formulate a novel model that translates the geometric uncertainty into a stochastic resistive term, similar to the immersed boundary method. On each subdomain we approximate the stochastic permeability and then in the online stage sample the resulting random vectors according to the volume fraction of material growth given by the reactive transport process. This strategy is validated on static flow problems in randomly modified porous media. In practice, we anticipate this offline approximated random permeability replacing the physically unreasonable Hagen-Poiseuille assumption for flow and transport models in pore-networks, thus allowing a full transient simulation of flow and transport, accurate to the porescale, without resorting to an online porescale simulation. Further, we demonstrate that with porescale accurate permeabilities a pore-network model may be constructed without faithfully representing the underlying porous structure, while still resulting in accurate macroscale properties.

This work is organized as follows. In Chapter 2 we review mathematical preliminaries necessary for the rest of the thesis. In Chapter 3 we consider charge transport in a heterostructure semiconductor. In particular we review the heterostructure model, and describe its multiscale character. We then develop novel domain decomposition methods designed for the unusual conditions at the heterostructure interface stemming from the multiscale phenomena in the model. We prove convergence theorems for the methods and then consider the mathematical analysis of the heterostructure model leveraging the domain decomposition formulation. We then examine the sensitivity of the continuum scale model to the data produced by the angstrom scale interface calculation. Finally we present numerical results validating the methods including mesh independence tests and convergence tests on model problems, and simulate several true heterostructure models and perform sensitivity analysis to the angstrom scale data.

In Chapter 4 we consider fluid flow in an evolving porous medium. We first review
fluid flow models at various scales, as well as their relationships through the hydraulic permeability. Next we present our stochastic porescale flow model and examine how the stochastic geometry is represented by a resistive term. We then present our reduced order three-scale flow method. We validate the method first on a synthetic 2d geometry, and conclude with simulations on real porescale images of sandstone.

In Appendix A we present a review of Density Functional Theory, the angstrom scale model for the heterostructure semiconductor problem. In Appendix B we provide an abbreviated user manual for HybGe-Flow3D, a freely available and open source software package developed for multiscale fluid flow in complex stochastic porous media. Finally in Appendix D we present permeability distributions for the sandstone geometry considered in Chapter 4.
2. MATHEMATICAL PRELIMINARIES

2.1. Functional Analysis

In this section we present background definitions and results from functional analysis. We begin by discussing the Lebesgue and Sobolev spaces, followed by a review of the results related to linear operators on Banach and Hilbert spaces. We then present various results needed for convergence analysis in iterative methods. The results in this section, unless otherwise marked, follow the texts [3, 13, 14, 101].

Throughout this section we let $U \subset \mathbb{R}^d$ be an open bounded domain, where $d$ is a positive integer.

2.1.1. Lebesgue and Sobolev Spaces

**Definition 2.1.1** (Lebesgue Spaces, $p \in [1, \infty)$). For $1 \leq p < \infty$, the Lebesgue space $L^p(U)$ is the set of equivalence classes of measurable functions $v$ defined on $U$ which are equal almost everywhere s.t.

$$\left( \int_U |v|^p \right)^{\frac{1}{p}} < \infty$$

where the integration is with respect to the Lebesgue measure. The space $L^p(U)$ is equipped with the norm

$$\|v\|_{L^p(U)} = \left( \int_U |v|^p \right)^{\frac{1}{p}}.$$  \hspace{1cm} (2.1.2)

**Definition 2.1.2** (Lesbesgue Space $L^\infty(U)$). The space $L^\infty(U)$ is defined as the set of those functions $v$ defined on $U$ s.t. $v$ is bounded almost everywhere on $U$, with norm given by the essential supremum,

$$\|v\|_{L^\infty(U)} = \inf\{K : |v| \leq K \text{ a. e. on } U\}.$$  \hspace{1cm} (2.1.3)
A Lebesgue space appearing frequently which is of particular importance in analysis and numerical approximation of boundary value problems is the Hilbert space $L^2(U)$, with the inner product

$$(u, v)_{L^2(U)} = \int_U u \, v.$$  

(2.1.4)

When working on boundary value problems, the space $L^p(U)$ is often 'too big,' in the sense that it contains functions that are incompatible with the partial differential equations we are interested in analyzing and solving. This is because $L^p(U)$ contains functions with no well-defined derivatives. The appropriate vector spaces are restrictions of $L^p(U)$ containing functions with well-defined weak derivatives, and are called Sobolev spaces. Before defining the Sobolev spaces, we define the notion of a weak derivative. First we need to introduce the locally $p$-integrable spaces, the multi-index and associated derivative, and the space of smooth functions with compact support, $C^\infty_0(U)$.

**Definition 2.1.3** (Locally $p$-integrable). A function $v : U \to \mathbb{R}$ is called locally $p$-integrable in $U$ if $v \in L^p(K)$ for every compact subset $K \subset U$. In this case we write $v \in L^p_{\text{loc}}(U)$.

We note that $L^p(U) \subset L^p_{\text{loc}}(U)$ for each open subset $U \in \mathbb{R}^d$ and for $p \in [1, \infty)$.

A multi-index $\alpha = (\alpha_1, \ldots, \alpha_d)$ is a vector of $d$ nonnegative integers. Define the length of the multi-index $|\alpha| = \sum_{i=1}^d \alpha_i$, and the notation for the $\alpha$th partial derivative $D^\alpha$

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \ldots \partial x_d^{\alpha_d}}.$$  

(2.1.5)

We denote by $C^k(U)$ the space of continuous functions on $U$ whose $\alpha$th partial derivatives exist and are continuous for all $\alpha$ with $|\alpha| \leq k$. We note that $C^k(U)$ is a Banach space with respect to the norm

$$\|v\|_{C^k(U)} = \max_{0 \leq |\alpha| \leq k} \sup_{x \in U} |D^\alpha v(x)|.$$  

(2.1.6)
Analogously, the space $C^\infty(U)$ denotes the space of infinitely smooth functions on $U$, i.e. the space of functions with continuous partial derivatives of any order. Finally, $C_0^\infty(U)$ denotes the set of infinitely differentiable functions with compact support in $U$.

**Definition 2.1.4 (Weak $\alpha$th Derivative).** If $u \in L^1_{loc}$ and $\alpha$ is a multi-index, then $v$ is called a weak $\alpha$th derivative of $u$, denoted by $\partial^\alpha u = v$, if

$$\int_U u \ D^\alpha \phi = (-1)^{|\alpha|} \int_U v \phi, \quad \forall \phi \in C_0^\infty(U).$$

(2.1.7)

It is well known that the weak $\alpha$th derivative is unique, and for sufficiently smooth functions, agrees with the classical derivative. More precisely, for $v \in L^1_{loc}(U)$ and $\alpha$ a multi-index, the weak $\alpha$th derivative $\partial^\alpha v$, if it exists, is defined in $U$ up to a set of measure zero. Additionally, if $v \in C^k(U)$ and $\alpha$ a multi-index with $|\alpha| \leq k$, then the classical partial derivative $D^\alpha v$ is equal to the weak $\alpha$th derivative $\partial^\alpha v$.

We are now prepared to define the Sobolev spaces $W^{k,p}(U)$.

**Definition 2.1.5.** Let $k \geq 1$ be an integer and $p \in [1, \infty]$. The Sobolev space $W^{k,p}(U)$ is the set of all functions $v \in L^p(U)$ such that for each multi-index $\alpha$ with $|\alpha| \leq k$, the weak derivative $\partial^\alpha v$ exists and is a member of $L^p(U)$. The space $W^{k,p}$ is equipped with the norm,

$$\|v\|_{W^{k,p}(U)} = \begin{cases} 
\left( \int_U \sum_{|\alpha| \leq k} |\partial^\alpha u|^p \right)^{\frac{1}{p}}, & \text{if } 1 \leq p < \infty, \\
\max_{|\alpha| \leq k} \|\partial^\alpha v\|_{L^\infty(U)}, & \text{if } p = \infty.
\end{cases}$$

(2.1.8)

Of particular importance are the Sobolev spaces where $p = 2$. In this case we write $H^k(U) = W^{k,2}(U)$.

Our interest in Sobolev spaces are within the context of analysis and numerical approximation of boundary value problems. Thus, a natural question arises: what does it mean to ask for the boundary value of a function $v \in W^{k,p}(U)$, or even $L^p(U)$? Since the elements of these spaces are equivalence classes of functions who differ on a set of measure zero, this question is nontrivial. The boundary of the domain $U$ is a $d$-1 dimensional
subset of $\mathbb{R}^d$, and so has measure zero. So, asking for a boundary value of a function in $L^p(U)$ makes no sense. Instead, we consider the trace of a function $v \in L^p(U)$ along the boundary $\partial U$ of the domain $U$. On the way to a definition of the trace operator, we first introduce the concept of a Lipschitz domain.

**Definition 2.1.6** ([3] p. 283). Let $V$ be a space of Lipschitz continuous functions on $\mathbb{R}^{d-1}$. Then the domain $U$ is said to be a Lipschitz domain if for every point $x_0 \in \partial U$ there exists $r > 0$ and $f \in V$ such that, with transformation of the coordinate system if necessary, we have

$$U \cap B_r(x_0) = \{x \in B_r(x_0) : x_d > f(x_1, \ldots, x_{d-1})\}, \quad (2.1.9)$$

where $B_r(x_0)$ is the ball centered at $x_0$ with radius $r$.

**Theorem 2.1.7** (Trace Theorem, [3] p. 297). Assume $U$ is a Lipschitz domain, and $p \in [1, \infty)$. Then there exists a compact bounded linear operator $\gamma : W^{1,p} \to L^p(\partial U)$ such that

1. For all $x \in \partial U$, $\gamma u(x) = u(x)$ if $u \in W^{1,p}(U) \cap C(\overline{U})$.

2. For some constant $C > 0$, $\|\gamma u\|_{L^2(\partial U)} \leq C\|u\|_{W^{1,p}(U)}$, for all $u \in W^{1,p}(U)$.

The operator $\gamma$ is called the trace operator.

The trace is a generalization of the boundary value of a function which is well defined for the Sobolev spaces $W^{1,p}$; for continuous functions in $W^{1,p}$ the trace agrees with the boundary value of the function. The range of the trace operator acting on $W^{1,p}$ is a subspace of $L^2(\partial U)$ denoted $W^{1-1/p,p}(\partial U)$, and is itself a positive order Sobolev space. Of particular importance in this work is the trace of the space $H^1(U) = W^{1,2}(U)$, which we denote by $\gamma(H^1(U)) = H^{1/2}(\partial U)$.

For boundary value problems with homogeneous Dirichlet conditions we are interested in the subspace of $W^{k,p}$ with 0 trace, which we denote by $W^{k,p}_0(U)$. In particular
for the case \( p = 2 \) we denote this space,

\[
H^k_0(U) = \{ u \in H^k(U) : \gamma(\partial^\alpha u) = 0 \text{ on } \partial U \text{ for all } |\alpha| < k \}. \tag{2.1.10}
\]

Then for \( k = 1 \),

\[
H^1_0(U) = \{ u \in H^1(U) : \gamma u = 0 \text{ on } \partial U \}. \tag{2.1.11}
\]

The following two theorems will be used to obtain estimates necessary for convergence arguments in this work.

**Theorem 2.1.8** (Trace Inequality [90]). There exists a constant \( C > 0 \) such that

\[
\| \gamma v \|_{H^{1/2}(\partial U)} \leq C \| v \|_{H^1(U)}, \quad \forall v \in H^1(U). \tag{2.1.12}
\]

**Theorem 2.1.9** (Poincaré Inequality). Let \( p \in [1, \infty) \). Then there exists a constant \( C \) depending only on \( U \) and \( p \) such that for each \( u \in W^{1,p}_0(U) \),

\[
\| u \|_{L^p(U)} \leq C \| \nabla u \|_{L^p(U)}. \tag{2.1.13}
\]

Next we review several results that are useful for the weak formulation of boundary value problems. We now assume \( U \) is a Lipschitz domain. We denote by

\[
\nu(x) = (\nu_1, \ldots, \nu_d)^T(x) \tag{2.1.14}
\]

the outward unit normal to the boundary \( \partial U \).

**Theorem 2.1.10** (Generalized Green’s Theorem). Let \( u, v \in H^1(U) \). Then

\[
\int_U \frac{\partial v}{\partial x_i} u = - \int_U v \frac{\partial u}{\partial x_i} + \int_{\partial U} u \nu_i v. \tag{2.1.15}
\]

The Generalized Green’s Theorem gives us the corollary,

**Corollary 2.1.11.** If \( u \in H^1(U) \) and \( v \in H^2(U) \), then

\[
\int_U u \Delta v = - \int_U \nabla u \cdot \nabla v + \int_{\partial U} u \nabla v \cdot \nu. \tag{2.1.16}
\]
2.1.2. Banach and Hilbert Space Methods

The variational, or weak, formulation of the boundary value problems considered in this work are formulations of the differential operator as maps from a Hilbert space to its dual space. Next we define some of the basic terminology and concepts necessary to formulate and work with the weak statements of elliptic boundary value problems.

Let $V, W$ be Banach spaces, and denote by $V', W'$ their dual spaces, i.e. the space of all bounded linear functionals acting on $V$ and $W$ respectively:

$$V' := \{ f : V \to \mathbb{R} : f \text{ is continuous and linear} \}. \quad (2.1.17)$$

**Definition 2.1.12 (Continuous Linear Operator).** We say that a linear operator $A : V \to W'$ is continuous if there exists some constant $C > 0$ such that for each $v \in V$ and $w \in W$ we have

$$|Av(w)| \leq C\|v\|_V \|w\|_W. \quad (2.1.18)$$

We denote $\mathcal{L}(V, W')$ the space of continuous linear operators mapping $V$ to $W'$. This space is equipped with the norm,

$$\|A\|_{\mathcal{L}(V, W')} = \sup_{\|v\|_V \leq 1} \|Av\|_{W'}. \quad (2.1.19)$$

**Definition 2.1.13 (Coercive Linear Operator).** We say that a linear operator $A : V \to V'$ is coercive if there exists some constant $\alpha > 0$ such that for each $u \in V$ we have

$$Au(u) \geq \alpha\|u\|_V. \quad (2.1.20)$$

We note that a linear operator $A : V \to W'$ is equivalent to the bilinear form $a : V \times W \to \mathbb{R}$, given by

$$Au(v) = a(u, v), \quad u \in V, v \in W. \quad (2.1.21)$$

An important concept is that of the annihilator of a subspace of a Banach space.
Definition 2.1.14 (Annihilator). Let \( K \subset V \) be a subspace of the Banach space \( V \). We denote the annihilator of \( K \) by \( K^o \), which is defined as the set of continuous linear functionals whose kernel contains \( K \):

\[
K^o = \{ f \in V' : f(u) = 0 \text{ for all } u \in K \}.
\] (2.1.22)

We note that \( K^o \) is a closed subspace of \( V' \).

We now assume \( V \) and \( W \) are Hilbert spaces with inner products \((\cdot, \cdot)_V\) and \((\cdot, \cdot)_W\), respectively. A fundamental result for Hilbert spaces is the Riesz Representation Theorem.

Theorem 2.1.15 (Riesz Representation Theorem). Let \( V \) be a Hilbert space. For each \( f \in V' \) there exists a unique \( v \in V \) such that

\[
f(u) = (v, u)_V, \quad \forall u \in V.
\] (2.1.23)

A result of fundamental importance in the analysis of partial differential equations is the Lax-Milgram Theorem.

Theorem 2.1.16 (Lax-Milgram). Let \( V \) be a Hilbert space. Assume the linear operator \( A : V \to V' \) is continuous and \( V \)-coercive with coercivity constant \( \alpha \). Then \( A : V \to V' \) is an isomorphism: for each \( f \in V' \) there is a unique \( u \in V \) for which \( Au = f \in V' \), and moreover, \( \|u\|_V \leq \alpha^{-1}\|f\|_{V'} \). If, in addition, \( A \) is symmetric, then \( u \) is the solution of the minimization problem,

\[
u \in V : \frac{1}{2}Au(u) - f(u) \leq \frac{1}{2}Av(v) - f(v), \quad \forall v \in V.
\] (2.1.24)

Next we present results necessary to consider a partial differential equation with constraints. Let \( B : V \to W' \) be continuous and linear.

Definition 2.1.17 (Adjoint (dual) Operator). We define the adjoint, or dual, of \( B \), \( B' : W \to V' \), by

\[
B'w(v) = Bv(w), \quad \forall v \in V, w \in W.
\] (2.1.25)
We note that $B'$ is continuous, and its adjoint is given by $B'' = B$.

In this work we will take advantage of the equivalence between the dual of an operator being bounding and the surjectivity of the operator. The following theorem and corollary are used to establish this equivalence.

**Theorem 2.1.18.** Let $V$ and $W$ be Hilbert spaces. If $B : V \to W'$ is continuous and linear, then the closure of the range of $B$ is the annihilator of the kernel of $B'$. That is,

$$
\overline{B(V)} = (\text{Ker } B')^\circ.
$$

(2.1.26)

**Corollary 2.1.19.** Let $V$ and $W$ be Hilbert spaces. Let $B : V \to W'$ be continuous and linear. Then there exists a $\beta > 0$ such that

$$
\|Bv\|_{W'} \geq \beta \|v\|_V, \quad \forall v \in V,
$$

(2.1.27)

if and only if $B$ is an isomorphism of $V$ onto $(\text{Ker } B')^\circ$.

These results motivates a definition.

**Definition 2.1.20 (Bounding Operator).** Let $B \in \mathcal{L}(V,W')$. We say that the operator $B$ is bounding if there exists $\beta > 0$ such that

$$
\|Bv\|_{W'} \geq \beta \|v\|_V \forall v \in V.
$$

(2.1.28)

The following theorem establishes the equivalence we will require in this work.

**Theorem 2.1.21.** Assume $V$ and $W$ are Hilbert spaces, and that $B \in \mathcal{L}(V,W')$. Then the following are equivalent.

- The adjoint $B' : W \to (\text{Ker } B)^\circ$ is an isomorphism.
- $B'$ is bounding,

$$
\inf_{w \in W} \sup_{v \in V} \frac{B'w(v)}{\|v\|_V \|w\|_W} \geq \beta > 0.
$$

(2.1.29)
The restriction to the orthogonal complement $B : (\text{Ker } B)^\perp \to W'$ is an isomorphism, and
\[ \|Bv\|_{W'} \geq \beta \|v\|_V, \quad v \in (\text{Ker } B)^\perp. \] (2.1.30)

$B : V \to W'$ is a surjection.

Before moving on to the mixed formulation appropriate for analyzing constrained problems, we present the Closed Range Theorem.

**Theorem 2.1.22** (Closed Range Theorem). Let $V$ and $W$ be Hilbert spaces. The following are equivalent.

- $B : V \to W'$ has closed range.
- $B(V) = (\text{Ker } B')^\circ$.
- $\sup_{w \in W} \frac{Bv(w)}{\|w\|_W} \geq \beta \inf_{z \in \text{Ker } B} \|v + z\|_V, \quad v \in V$.
- The adjoint $B' : V \to W'$ has closed range.
- $B'(W) = (\text{Ker } B)^\circ$.
- $\sup_{v \in V} \frac{Bv(w)}{\|v\|_V} \geq \beta \inf_{z \in \text{Ker } B'} \|w + z\|_V, \quad w \in W$.

The Lax-Milgram Theorem 2.1.16 is, in fact, a special case of the Closed Range Theorem.

Suppose now that $A : V \to V'$ is symmetric and nonnegative,
\[ Au(v) = Av(u), \quad \forall u, v \in V, \] (2.1.31)
\[ Au(u) \geq 0, \quad \forall u \in V, \] (2.1.32)
and that $f \in V'$. As the Lax-Milgram Theorem stated, the problem $Au = f \in V'$ characterizes the solution of the minimization problem
\[ u \in V : \frac{1}{2} Au(u) - f(u) \leq \frac{1}{2} Av(v) - f(v), \quad \forall v \in V. \] (2.1.33)
Let $B : V \to W'$ be linear and continuous, $g \in B(V) \in W'$, and introduce the constraint $Bv = g$. Then this becomes

$$u \in V, \quad Bu = g : \frac{1}{2} Au(u) - f(u) \leq \frac{1}{2} Av(v) - f(v), \quad \forall v \in V \text{ with } Bv = g. \quad (2.1.34)$$

The constrained minimum to this problem is characterized by

$$u \in V : \text{Bu} = g \text{ and } Au - f \in (\text{Ker } B)^{\circ}. \quad (2.1.35)$$

If we drop the assumption that $A$ is symmetric, and assume $A$ is $V$-coercive on Ker $B$, then there exists a unique solution to this problem through application of the Lax-Milgram Theorem.

We introduce the Lagrange Multiplier $p \in W$, which realizes the relaxation of the equation in $V'$, to present the mixed formulation of the constrained problem,

$$[u, p] \in V \times W : \text{Bu} - g = 0 \in W', \text{ and } Au + B'p - f = 0 \in V'. \quad (2.1.36)$$

**Theorem 2.1.23.** Assume that the linear operators $A : V \to V'$, $B : V \to W'$ are continuous from the indicated Hilbert spaces $V, W$ to their duals, and

- $A$ is non-negative and $V$-coercive on Ker $B$.
- $B'$ is bounding.

Then for every $f \in V'$ and $g \in W'$ the mixed system

$$u \in V, \quad p \in W : \quad Au + B'p = f \in V', \quad \text{(2.1.37)}$$

$$Bu = g \in W', \quad \text{(2.1.38)}$$

has a unique solution in $V \times W$.

In this work we will prove an extension of this theorem containing two constraints for the analysis of the heterostructure semiconductor model in Section 3.2..

Several methods in this work are iterative in nature and their convergence analysis employs one of the following two classic results.
Theorem 2.1.24 (Banach Contraction Mapping Principle). Let \((X, d)\) be a complete metric space, and let \(f : X \to X\) be a contraction. That is, there exists \(0 < K < 1\) such that \(d(f(x), f(y)) < Kd(x, y)\) for all \(x, y \in X\). Then there exists a unique fixed point \(x_0 \in X\) such that \(f(x_0) = x_0\). Further, for any \(x \in X\), \(\lim_{n \to \infty} f^n(x) = x_0\).

Theorem 2.1.25 (Schauder Fixed Point Theorem). Let \(X\) be a normed vector space, and \(K \subset X\) a non-empty, compact, and convex set. Then given any continuous mapping \(f : K \to K\) there exists \(x \in K\) such that \(f(x) = x\).

2.2. Probability

In Chapter 3 we will perform sensitivity analysis relating the moments of the quantities of interest in the semiconductor model to the angstrom scale data produced by a density functional theory calculation. Also, in Chapter 4 we will introduce a model for laminar flow in stochastic complex media. Thus in this section we give a review of some basic elements of probability theory in preparation for the the probabilistic elements of the work in this thesis.

For further background on probability theory we point the reader to the text [11], and for further discussion of stochastic modeling of fluid flow problems we point the reader to the dissertation [108].

Throughout this section we denote by \(\Omega\) the sample space of all possible outcomes of a random phenomenon, \(\mathcal{F}\) the collection of permissible events, and \(P : \mathcal{F} \to [0, 1]\) a probability measure.

Definition 2.2.1. A probability space is a triple \((\Omega, \mathcal{F}, P)\).

Definition 2.2.2. A probability space \((\Omega, \mathcal{F}, P)\) is complete if all subsets of probability 0 are events in \(\mathcal{F}\). That is, for \(A \subset \Omega\) with \(P(A) = 0\), \(B \in \mathcal{F}\) for all \(B \subset A\).

Next we define a random variable. First we introduce the notion of a Borel set.
Definition 2.2.3. The Borel σ-algebra, $\mathcal{B}(\mathbb{R})$, is the σ-algebra generated by the open intervals of $\mathbb{R}$.

Definition 2.2.4. Let $(\Omega, \mathcal{F}, P)$ be a probability space. A real-valued function $X : \Omega \to \mathbb{R}$ is called a random variable if for all Borel sets $B \in \mathcal{B}(\mathbb{R})$, the set $X^{-1}(B) = \{\omega \in \Omega \mid X(\omega) \in B\}$ satisfies

$$X^{-1}(B) \in \mathcal{F}.$$ (2.2.1)

In this case, $X$ is called an $\mathcal{F}$-measurable random variable.

Definition 2.2.5. Assume $X$ is a random variable on the probability space $(\Omega, \mathcal{F}, P)$. Then the probability law of $X$ on $\mathbb{R}$, denoted $P_X$, is a function defined for every Borel set $B \in \mathcal{B}$ by

$$P_X(B) = P(X^{-1}(B)) = P([X \in B]).$$ (2.2.2)

In particular, every random variable $X$ defines a probability measure given by (2.2.2).

We are interested in considering a stochastic process. We start by discussing collections of random variables. Later we will discuss random vectors and finally, stochastic processes.

Definition 2.2.6. Let random variables $X_1, X_2, \ldots, X_n$ be defined on the same probability space $(\Omega, \mathcal{F}, P)$. Then their joint probability law is a probability measure $P_{X_1, X_2, \ldots, X_n}$ on $\mathbb{R}^n$ such that for any Borel set $B \in \mathbb{R}^n$

$$P_{X_1, X_2, \ldots, X_n}(B) = P([X_1, X_2, \ldots, X_n] \in B)).$$ (2.2.3)

Definition 2.2.7. The (cumulative) distribution function $F_X : \mathbb{R} \to [0, 1]$ of a random variable $X$ is defined by

$$F_X(x) = P_X((\infty, x]) = P([X \leq x]).$$ (2.2.4)
We recall the relationship between the distribution $F_X$ of a random variable and the probability law of $X$, as well as some basic facts about the distribution function.

**Lemma 2.2.8.** A distribution function $F_X$ satisfies the following conditions:

1. $F_X(x) \leq F_X(y)$ for every pair $(x, y)$ such that $x < y$.
2. $F_X(b) - F_X(a) = P_X((a, b])$, for $a < b$.
3. $\lim_{x \to \infty} F_X(x) = 1$, $\lim_{x \to -\infty} F_X(x) = 0$.
4. $F_X$ is right continuous, i.e. $\lim_{n \to \infty} F_X(x_n) = F_X(x)$ whenever the sequence $\{x_n\}_{n=1}^{\infty}$ approaches $x$ from the right.

**Definition 2.2.9.** Let the random variables $X_1, X_2, \ldots, X_n$ be defined on a common probability space $(\Omega, \mathcal{F}, P)$. Then their joint distribution function $F_{X_1,\ldots,X_n}$ is given for $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ by

$$F_{X_1,\ldots,X_n}(x_1, x_2, \ldots, x_n) = P([X_1 \leq x, X_2 \leq x_2, \ldots, X_n \leq x_n]). \quad (2.2.5)$$

**Definition 2.2.10.** A random variable $X$ is called continuous if there is a nonnegative integrable function $f_X : \mathbb{R} \to [0, \infty]$, called the probability density function of $X$, such that

$$F_X(x) = \int_{-\infty}^{x} f_X(u) \, du. \quad (2.2.6)$$

**Definition 2.2.11.** The random variables $X_1, X_2, \ldots, X_n$, given on the same probability space $(\Omega, \mathcal{F}, P)$, are jointly continuous with density $f_{X_1,\ldots,X_n}$ if there exists a function $f_{X_1,\ldots,X_n} : \mathbb{R}^n \to [0, \infty]$ such that for any Borel set $B \in \mathbb{R}^n$

$$P_{X_1,\ldots,X_n}(B) = \int_B f_{X_1,\ldots,X_n}(x_1, x_2, \ldots, x_n) \, dx_1dx_2\ldots dx_n. \quad (2.2.7)$$

**Definition 2.2.12.** A random variable $X$ is called discrete if the set of values $S$ it can take is finite or countably infinite.
Definition 2.2.13. The discrete random variable $X$ has a probability mass function $p_X : \mathbb{R} \to [0, 1]$ defined by
$$p_X(x) = P([X = x]). \quad (2.2.8)$$

Definition 2.2.14. Assume $X$ and $Y$ are discrete random variables given on the same probability space $(\Omega, \mathcal{F}, P)$. Then their joint probability mass function is defined by
$$p_{X,Y}(x,y) = P([X = x] \cap [Y = y]). \quad (2.2.9)$$

An important idea related to random variables is the expected value of the random variable.

Definition 2.2.15. Let $X$ be a random variable on the probability space $(\Omega, \mathcal{F}, P)$. Then the integral of $X$ with respect to the probability measure $P$ is given by
$$\int_{\Omega} X \, dP = \int_{\Omega} X^+ \, dP - \int_{\Omega} X^- \, dP, \quad (2.2.10)$$
where $X^+ = \max(0, X)$, $X^- = -\min(0, X)$, $X = X^+ - X^-$, $|X| = X^+ + X^-.$

Definition 2.2.16. A random variable $X$ is called integrable if $\int_{\Omega} |X| \, dP$ exists and is finite. In this case, we say that $X$ belongs to $L^1(\Omega, \mathcal{F}, P)$.

Definition 2.2.17. If a random variable $X \in L^1(\Omega, \mathcal{F}, P)$ then it has finite expected value given by
$$E[X] = \int_{\Omega} X \, dP. \quad (2.2.11)$$

Proposition 2.2.18. If $X$ is an integrable discrete random variable, then its expected value is given by
$$E[X] = \sum x p_X(x). \quad (2.2.12)$$
If $X$ is an integrable continuous random variable, then its expected value is given by
$$E[X] = \int_{\mathbb{R}} x f(x) \, dx. \quad (2.2.13)$$
In general, if $X$ is integrable

$$E[X] = \int_{\mathbb{R}} x dF_X(x). \quad (2.2.14)$$

An important related concept is the variance of the random variable $X$.

**Definition 2.2.19.** A random variable $X$ is called square-integrable if $\int_{\Omega} |X|^2 \, dP$ exists and is finite. In this case we say that $X$ belongs to $L^2(\Omega, \mathcal{F}, P)$.

**Definition 2.2.20.** Let the random variable $X \in L^2(\Omega, \mathcal{F}, P)$. Then we define the variance of the random variable $X$ by

$$\text{Var}[X] = \int_{\Omega} (X - E[X])^2 \, dP. \quad (2.2.15)$$

**Definition 2.2.21.** Let the random variables $X, Y \in L^2(\Omega, \mathcal{F}, P)$. Then the covariance of $X$ and $Y$ is defined by

$$\text{Cov}[X,Y] = E[(X - E[X])(Y - E[Y])]. \quad (2.2.16)$$

**Proposition 2.2.22.** Let $X_1, X_2, \ldots, X_n \in L^2(\Omega, \mathcal{F}, P)$. Then $\text{Var}[\sum_{i=1}^n X_i]$ is finite and given by

$$\text{Var}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \text{Var}[X_i] + 2 \sum_{i<j} \text{Cov}[X_i, X_j]. \quad (2.2.17)$$

**Definition 2.2.23.** Let the random variables $X, Y \in L^2(\Omega, \mathcal{F}, P)$. If $\text{Cov}[X,Y] = 0$ then we say that $X$ and $Y$ are uncorrelated random variables.

A related concept is the characteristic function of $X$, 

**Definition 2.2.24.** The characteristic function of $X$ is the function $\phi : \mathbb{R} \to \mathbb{C}$ given by

$$\phi(t) = E[e^{itX}]. \quad (2.2.18)$$

An important concept for random variables is the notion of independence.
**Definition 2.2.25.** Let $A, B \in \mathcal{F}$. Then if $P(B) \neq 0$ we define the conditional probability of $A$ given $B$ to be

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$  

(2.2.19)

**Proposition 2.2.26.** Let $B \in \mathcal{F}$, then the function $P_B : \mathcal{F} \to \mathbb{R}$ given by

$$P_B(A) = P(A|B)$$  

(2.2.20)

defines a probability measure on $\mathcal{F}$.

**Definition 2.2.27.** Let $A, B \in \mathcal{F}$. Then $A$ and $B$ are independent if

$$P(A \cap B) = P(A)P(B).$$  

(2.2.21)

We note that if $P(A|B)$ and $P(B|A)$ are defined, then $A$ and $B$ are independent if $P(A|B) = P(A)$ and $P(B|A) = P(B)$. Next we define the independence of random variables $X$ and $Y$.

**Definition 2.2.28.** Let $X$ and $Y$ be random variables given on the same probability space $(\Omega, \mathcal{F}, P)$. Random variables $X$ and $Y$ are said to be independent if for any Borel sets $A$ and $B$ the events $[X \in A]$ and $[Y \in B]$ are independent.

**Definition 2.2.29.** Random variables $X_1, X_2, \ldots, X_n$ are independent if for any ordered $m$-tuple $(i_1, \ldots, i_m)$ with $\{i_1, \ldots, i_m\} \subset \{1, \ldots, n\}$, $m \leq n$, and for any $(x_{i_1}, \ldots, x_{i_m}) \in \mathbb{R}^m$,

$$F_{X_{i_1}, \ldots, X_{i_m}}(x_{i_1}, \ldots, x_{i_m}) = F_{X_{i_1}}(x_{i_1}) \ldots F_{X_{i_m}}(x_{i_m}),$$  

(2.2.22)

or, equivalently, if for any $(t_1, t_2, \ldots, t_n) \in \mathbb{R}^n$

$$\phi_{X_1, X_2, \ldots, X_n}(t_1, t_2, \ldots, t_n) = \phi_{X_1}(t_1)\phi_{X_2}(t_2) \ldots \phi_{X_n}(t_n).$$  

(2.2.23)

Next we define a random vector, which is nothing more than a finite collection of random variables.
Definition 2.2.30. \( \mathbf{X} = (X_1, \ldots, X_n) \) is an \( n \)-dimensional random vector if its components \( X_1, \ldots, X_n \) are one-dimensional random variables.

Definition 2.2.31. The collection of probabilities
\[
F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1, \ldots, X_n \leq x_n), \quad \mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n,
\]
(2.2.24)
is the distribution function \( F_{\mathbf{X}} \) of \( \mathbf{X} \).

Three important statistical quantities of a random vector with all components in \( L^2(\Omega, \mathcal{F}, P) \) are its expectation, variance, and covariance matrix.

Definition 2.2.32. The expectation of the \( n \)-dimensional random vector \( \mathbf{X} \) is given by
\[
E[\mathbf{X}] = (E[X_1], \ldots, E[X_n]).
\]
(2.2.25)

Definition 2.2.33. The covariance matrix \( \mathbf{C}_X \) of the \( n \)-dimensional random vector \( \mathbf{X} \) is defined by
\[
(C_{\mathbf{X}})_{ij} = \text{Cov}(X_i, X_j), \quad 1 \leq i, j \leq n.
\]
(2.2.26)

We note that the diagonal elements of the covariance matrix \( (C_{\mathbf{X}})_{ii} = \text{Var}(X_i) \). It is often convenient to normalize the covariance matrix by dividing the random variables by their standard deviations,
\[
\sigma_{X_i} = \sqrt{\text{Var}(X_i)}.
\]
(2.2.27)
The resulting matrix is referred to as the correlation matrix.

Definition 2.2.34. The correlation matrix \( \mathbf{C}_X \) of the \( n \)-dimensional random vector \( \mathbf{X} \) is defined by
\[
(C_{\mathbf{X}})_{ij} = \frac{\text{Cov}(X_i, X_j)}{\sigma_i \sigma_j}.
\]
(2.2.28)

We require the Cauchy-Schwarz Inequality.
Proposition 2.2.35 \hspace{0.2cm} (Cauchy-Schwarz Inequality). Let the random variables \( X, Y \in L^2(\Omega, \mathcal{F}, P) \). Then

\[
(E[XY])^2 \leq E[X^2]E[Y^2].
\] (2.2.29)

The Cauchy-Schwarz Inequality then tells us that

\[-1 \leq (C_X)_{ij} \leq 1 \] (2.2.30)

for \(1 \leq i, j \leq n\). We say that \( X_i \) and \( X_j \) are uncorrelated if \((C_X)_{ij} = 0\) and strongly correlated if \(|(C_X)_{ij}| \approx 1\).

Finally we turn our attention to stochastic processes.

**Definition 2.2.36.** Let \((\Omega, \mathcal{F}, P)\) be a complete probability space, \((S, \mathcal{S})\) be a measurable space, and \((D, d)\) be a metric space. A family \(\{Z(x, \omega), x \in D\}\) of random variables defined on \((\Omega, \mathcal{F}, P)\) taking values in \(S\) is called a stochastic process. The spaces \(S\) and \(D\) are referred to as the state space and the index set, respectively.

It is common to introduce stochastic processes with \(D = T \subset \mathbb{R}^+\) denoting time. However, the stochastic process in the fluid flow section of this work is indexed by the spatial domain, and so we introduce the stochastic process in more generality. Next we define some particular types of stochastic processes.

**Definition 2.2.37.** If the index set \(D\) is finite or countably infinite (respectively, uncountably infinite), then the stochastic process \(\{Z(x, \omega), x \in D\}\) is called a discrete-parameter (respectively, continuous-parameter) stochastic process.

**Definition 2.2.38.** If the state space \(S\) is finite or countably infinite (respectively, uncountably infinite), then a stochastic process \(\{Z(x, \omega), x \in D\}\) is called a discrete-state (respectively, continuous-state) stochastic process.

It is useful to note that the stochastic process may be viewed as a function of two variables, \(x \in D\) and \(\omega \in \Omega\), \((x, \omega) \mapsto Z(x, \omega)\). We note that it is common to use a subscript to denote the index parameter, i.e. to write \(Z_x(\omega)\).
Definition 2.2.39. For each \( \omega \in \Omega \), \( Z(\cdot, \omega) : D \rightarrow S \) is a function of \( x \) that represents a possible realization of the stochastic process \( \{Z(x), x \in D\} \). This function is called the sample path of the process.

Definition 2.2.40. Let \( n \) be an integer, \( \mathbf{x} = (x_1, x_2, \ldots, x_n) \in D^n \), and \( \mathbf{z} = (z_1, z_2, \ldots, z_n) \in \mathbb{R}^n \). Then the probabilities

\[
P(\{Z_{x_1} \leq z_1, Z_{x_2} \leq z_2, \ldots, Z_{x_n} \leq z_n\}),
\]

are called the finite-dimensional distributions of \( Z_x \), and denoted by

\[
F_{Z_x}(\mathbf{z}) = F_{Z_{x_1}, Z_{x_2}, \ldots, Z_{x_n}}(z_1, z_2, \ldots, z_n).
\]

Definition 2.2.41. The covariance function of the stochastic process \( Z_x(\omega) \) is the function \( C : D \times D \rightarrow \mathbb{R} \) defined by

\[
C(x, y) = \text{Cov}(Z_x, Z_y).
\]

2.3. Numerical Analysis

Each of the applications we consider in ensuing chapters will involve the numerical solution of systems of partial differential equations. The heterostructure semiconductor model considered in Chapter 3 will be discretized by centered finite differences. The porescale fluid flow model presented in Chapter 4 will be discretized by a staggered grid cell-centered finite volume method. In this section we begin by reviewing each of these discretizations.

We then turn our attention to analysis of the Richardson iterative scheme for solving linear systems of equations and more generally, linear equations posed on Hilbert spaces. The methods developed in Chapter 3 for the heterostructure semiconductor model can be
analyzed as Richardson schemes for equations developed on the interface between semiconductor domains. Thus we review the convergence analysis of the Richardson scheme here.

Solving a large sparse linear system is central to many numerical simulations in science and engineering. Direct methods which factor the matrix $A$ into easily invertible matrices are widely used, but scale poorly with system size in terms of operation counts and memory requirements. When detailed three-dimensional simulations are required one often ends up with problem sizes in the areas of millions or billions of degrees of freedom. On these large problems iterative methods for the solution of the linear system resulting from discretization of the partial differential equation model are the only option, but stationary methods like the Richardson scheme often converge too slowly. Stronger methods fall in the category of "Krylov subspace methods.” Thus we review two methods, the Conjugate Gradient Method (CG) and the Generalized Minimum Residual Method (GMRES). These methods are ubiquitous in scientific computing codes, and in particular are used in the software HybGe-Flow3D [24] presented in Chapter 4. The presentation of the Krylov methods closely follows the presentation from [[53] Chapter 2].

Iterative methods, like the Krylov subspace methods, for large sparse linear systems have several advantages. Iterative methods tend to have lower memory requirements and generally arrive at a solution in fewer operation counts than direct solvers. However, their performance is less robust and reliable. In the case of the Krylov methods it is well known that the Conjugate Gradient and Generalized Minimum Residual methods will, in fact, arrive at the solution within at most $N$ iterations, where $N$ is the system size. However, when the system is very large, this performance is far too poor, and we require a solution within a certain tolerance in far fewer iterations. The performance of these methods can be drastically improved by preconditioning the linear system, to arrive at a system that is equivalent, but with a better conditioned matrix. To quote Trefethen
and Bau [107] "In ending this book with the subject of preconditioners, we find ourselves at the philosophical center of the scientific computing of the future... Nothing will be more central to computational science in the next century than the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly. For Krylov subspace matrix iterations, this is preconditioning.” We end this section by by describing two common preconditioners, as well as the strategy we employ in the software HybGe-Flow3D [24] in Chapter 4.

2.3.1. Finite Difference Method

Consider the linear Poisson equation posed on \((a, b)\),

\[-u'' = f, \ x \in (a, b),\]  
\[u(a) = \alpha, \ u(b) = \beta.\]  

(2.3.1)  
(2.3.2)

We seek to approximate \(u(x_j)\) by values \(U_j\) at grid points \(x_j, j = 1, \ldots, M\) where \(a = x_0\) and \(b = x_{M+1}\). We will assume a uniform grid, so that \(x_{j+1} - x_j = h, j = 0, \ldots, M\).

Replace \(u''\) in (2.3.1) by a centered finite difference approximation,

\[D^2U_j = \frac{1}{h^2}(U_{j-1} - 2U_j + U_{j+1}),\]  

(2.3.3)

to obtain a set of algebraic equations

\[-\frac{1}{h^2}(U_{j-1} - 2U_j + U_{j+1}) = f(x_j), \ \text{for} \ j = 1, 2, \ldots, M.\]  

(2.3.4)

Incorporating the boundary conditions we obtain a linear system

\[AU = F,\]  

(2.3.5)
The values $U_j$ are an approximation of the true solution $u$ at grid points $x_j$ depending on the width of the mesh $h$. Thus we expect the error between $U_j$ and $u(x_j)$ to converge to 0 as $h \to 0$. In Chapter 3 we develop domain decomposition methods for heterostructure semiconductor simulation, and will mesh convergence to ensure that the expected order of convergence of the discretization is not effected by the domain decomposition methods. Thus we review here the expected order of convergence of the centered finite difference approximation.

We begin by recalling Taylor expansions up to fourth order,

$$u(x + h) = u(x) + hu'(x) + \frac{h^2}{2} u''(x) + \frac{h^3}{6} u^{(3)}(x) + \frac{h^4}{24} u^{(4)} + O(h^5),$$

(2.3.7)

$$u(x - h) = u(x) - hu'(x) + \frac{h^2}{2} u''(x) - \frac{h^3}{6} u^{(3)}(x) + \frac{h^4}{24} u^{(4)} + O(h^5)$$

(2.3.8)

Then plugging (2.3.7)-(2.3.8) into (2.3.3) we have

$$D^2 u(x) = \frac{u(x + h) - 2u(x) + u(x - h)}{h^2}$$

(2.3.9)

$$= \frac{h^2 u''(x) + \frac{1}{12} h^4 u^{(4)} + O(h^5)}{h^2}$$

(2.3.10)

$$= u''(x) + O(h^2),$$

(2.3.11)

where $O(h^2)$ denotes the term whose size is controlled by $h^2$.

This shows that the pointwise error $|D^2 U_j - u''(x_j)|$ is controlled by $h^2$. However,
this does not tell us about how the global error

$$\|E\|_\infty = \|U - u\|_\infty = \max_{1 \leq j \leq M} |U_j - u(x_j)|$$  \hspace{1cm} (2.3.12)$$
is controlled by the mesh size $h$.

To obtain a bound on $\|E\|_\infty$ we compute the local truncation error (LTE), and then we use stability to show that the global error can be bounded in terms of the LTE.

Replacing $U_j$ by $u(x_j)$ in (2.3.4) we define the local truncation error $\tau_j$ by

$$\tau_j = \frac{1}{h^2} (u(x_{j-1}) - 2u(x_j) + u(x_{j+1})) + f(x_j) \text{ for } j = 1, 2, \ldots, M.$$  \hspace{1cm} (2.3.13)$$

It follows from (2.3.11) that

$$\tau_j = \frac{1}{12} h^2 u'''(x_j) + O(h^4).$$  \hspace{1cm} (2.3.14)$$

Since $u'''$ is independent of $h$, we have $\tau_j = O(h^2)$ as $h \to 0$.

Define $u$ to be the vector consisting of the true solution $u$ evaluated at grid points, so that $u_j = u(x_j)$. Then let $E = U - u$, and $\tau$ be the vector with components $\tau_j$, the local truncation error $x_j$. Then we subtract $AU = F + \tau$ from $Au = F$ and consider the equation,

$$A^h E^h = -\tau^h,$$  \hspace{1cm} (2.3.15)$$

where the superscript $h$ denotes the dependence on $h$.

Solving (2.3.15) for $E^h$ we have

$$E^h = -(A^h)^{-1} \tau^h.$$  \hspace{1cm} (2.3.16)$$

Thus,

$$\|E^h\| \leq \|(A^h)^{-1}\| \|\tau^h\|.$$  \hspace{1cm} (2.3.17)$$

Then if there is some constant $K$ s.t.

$$\|(A^h)^{-1}\| \leq K \text{ for sufficiently small } h,$$  \hspace{1cm} (2.3.18)$$
we have
\[ \| E^h \| \leq K \| r^h \| \]  
(2.3.19)
in whichever norm we choose.

The existence of such a $K$ is referred to as the stability of the method. In [[58], p.20] it is shown that $A$ is stable in both the 2-norm and the $\infty$-norm. Thus we have
\[ \| E^h \|_\infty = O(h^2). \]  
(2.3.20)

\section*{2.3.2. Finite Volume Method}

Recall the linear Poisson equation posed on a bounded open domain $D \subset \mathbb{R}^3$,
\[-\nabla \cdot (k \nabla u) = f, \quad x \in D, \]  
(2.3.21)
\[ u = u_D, \quad x \in \partial D_D, \]  
(2.3.22)
\[ k \frac{\partial u}{\partial \nu} = u_N, \quad x \in \partial D_N, \]  
(2.3.23)
where we split the boundary $\partial D$ of $D$ into two components $\partial D = \partial D_D \cup \partial D_N$ consisting of components with Dirichlet and Neumann data prescribed, respectively. Here $\nu$ denotes the unit outward normal to $D$ along $\partial D$.

We begin by discretizing the domain $D$ into $N$ hexahedrons $\{D_i\}_{i=1}^N$. In this exposition we restrict attention to a cartesian grid, though the basic ideas here extend to much more general meshings. The finite volume method seeks the vector $U = [U_1, \ldots, U_N]^T$, where $U_j$ denotes the value of the primary variable $u$ on cell $D_j$, which is assumed constant within the cell.

We will require the divergence theorem,

\textbf{Theorem 2.3.1} (Divergence Theorem). \textit{Let $V \subset \mathbb{R}^d$ be compact and have a piecewise smooth boundary. If $F$ is a continuously differentiable vector field defined on a neighborhood of $V$, then we have:}
\[ \int_V \nabla \cdot F = \int_{\partial V} (F \cdot \nu), \]  
(2.3.24)
where $\nu$ denotes the unit outward normal to $V$ along $\partial V$.

Let $C$ denote an interior cell in the discretization. We integrate and apply the divergence theorem,

$$- \int_C \nabla \cdot (k \nabla u) = - \int_{\partial C} (k \nabla u) \cdot \nu, \quad (2.3.25)$$

where $\nu$ denotes the unit outward normal to $C$ along $\partial C$.

Next we denote by $C_S$ the index of the cell below ($z$-axis) cell $C$, $C_N$ the index of the cell above ($z$-axis) cell $C$, $C_W$ the index of the cell to the left ($x$-axis) of cell $C$, $C_E$ the index of the cell to the right ($x$-axis) of cell $C$, $C_F$ the index of the cell in front ($y$-axis) of cell $C$, and $C_B$ the index of the cell behind ($y$-axis) cell $C$. Analogously we have the primary variable values $U_C$ on cell $C$ and $U_S$, $U_N$, $U_W$, $U_E$, $U_F$, $U_B$ on these neighbor cells. Finally we denote by $\partial_S$, $\partial_N$, $\partial_W$, $\partial_E$, $\partial_F$, and $\partial_B$ the cell face in each direction. Figure 2.1 illustrates this setup. Then we have

$$- \int_{\partial C} (k \nabla u) \cdot \nu = - \sum_{i \in \{S,N,W,E,F,B\}} \int_{\partial_i} (k \nabla u) \cdot \nu. \quad (2.3.26)$$

If we assume linear variation between cell centers for the data $k$, we then have

$$- \sum_{i \in \{S,N,W,E,F,B\}} \int_{\partial_i} \left( \frac{k_C + k_i}{2} \nabla u \right) \cdot \nu. \quad (2.3.27)$$

Then assuming linear variation in the primary variable $u$ between cell centers, we have

$$\sum_{i \in \{S,N,W,E,F,B\}} \int_{\partial_i} \left( \frac{k_C + k_i}{2} \frac{U_C - U_i}{\triangle i} \right), \quad (2.3.28)$$

where $\triangle i$ denote the distance between the cell centers $C$ and $C_i$. Finally, integrating this constant value along the boundary we obtain,

$$\sum_{i \in \{S,N,W,E,F,B\}} m^*(\partial_i) \frac{k_C + k_i}{2} \frac{U_C - U_i}{\triangle i}, \quad (2.3.29)$$

where $m^*$ denotes the Lebesgue measure in $\mathbb{R}^2$. 
FIGURE 2.1: Illustration of cell indices for finite volume discretization
Next we consider a boundary term. Suppose the face $i$ is on $\partial D$, the component of the boundary with Dirichlet data prescribed. Then we need to re-consider the term

$$\int_{\partial_i} (k \nabla u) \cdot \nu. $$

(2.3.30)

To approximate $k \nabla u \cdot \nu$ along $\partial_i$ we must use the Dirichlet data $u_D$ rather than the cell data $U_i$. We also assume that $k|_{\partial_i} = k_C$. Thus we have

$$\int_{\partial_i} (k \nabla u) \cdot \nu \approx m^*(\partial_i) k_C \frac{U_C - u_D|_{\partial_i}}{0.5 \Delta_i C}$$

(2.3.31)

where $\Delta_i C$ denotes the cell length of cell $C$ along the axis heading towards $\partial_i$, and $u_D|_{\partial_i}$ denotes the value of $u_D$ at the midpoint of the boundary face $\partial_i$.

For Neumann data the scenario is more straightforward since we are approximating $k \nabla u \cdot \nu$ along the boundary which is precisely the data prescribed by the Neumann condition. So we have,

$$\int_{\partial_i} (k \nabla u) \cdot \nu \approx m^*(\partial_i) u_N|_{\partial_i}. $$

(2.3.32)

Note that if $u_N$ is linear along the cell wall $\partial_i$, this formula is exact. This results in a sparse linear system

$$AU = F$$

(2.3.33)

where $A \in \mathbb{R}^{N \times N}$, $U \in \mathbb{R}^N$ and $F \in \mathbb{R}^N$, and row $i$ of the system corresponds to the finite volume discretization of cell $i$. The sparsity pattern is determined by neighboring cells, i.e.

$$A_{ij} \neq 0 \text{ when cells } i \text{ and } j \text{ share a face. }$$

(2.3.34)

The resulting linear system is symmetric and has a structured sparsity pattern that can be exploited for solving the system. Figure 2.2 presents this pattern.
2.3.2.1 Stokes Equations

Next we turn to discretizing the Stokes equations,

\begin{align*}
-\mu \Delta u + \nabla p &= f, \quad (2.3.35) \\
\nabla \cdot u &= 0. \quad (2.3.36)
\end{align*}

Here \( u : \mathbb{R}^3 \to \mathbb{R}^3 \) is the vector valued velocity, \( p : \mathbb{R}^3 \to \mathbb{R} \) is the pressure, \( f : \mathbb{R}^3 \to \mathbb{R}^3 \) corresponds to external forces, and \( \mu \in \mathbb{R} \) is the fluid viscosity. We notice immediately that there is no coupling in the momentum equation between the components of the velocity.
To see this, we write out an equivalent system that decouples the velocity components,

\[-\mu \Delta U + \nabla_x p = f_1,\]  
\[-\mu \Delta V + \nabla_y p = f_2,\]  
\[-\mu \Delta W + \nabla_z p = f_3,\]  
\[\nabla_x u + \nabla_y u + \nabla_z u = 0.\]  

Here we denote by $U$ the $x$-component of the velocity, $V$ the $y$-component, and $W$ the $z$-component. The Laplacian in each of these equations may be treated independently similarly to how the Poisson equation was discretized earlier. However, care must be taken in determining the grid, as unphysical pressure solutions may occur when all velocity components and the pressure are defined at the same cell centers. A common strategy is called a 'staggered grid discretization.' In this setting, one first discretizes the domain into $N$ cells corresponding to pressure unknowns. The velocity degrees of freedom are then taken at appropriate cell faces, resulting in staggered grids. In particular, the $x$ component of the velocity, $U$, is defined on the cell faces $\partial_W$ and $\partial_E$, the $y$ component of the velocity $V$, is defined on the cell faces $\partial_F$ and $\partial_B$, while the $z$ component of the velocity $W$ is defined on the cell faces $\partial_N$ and $\partial_B$. Figure 2.3 illustrates the mesh by looking at the $U$ grid.

We notice immediately that it is straightforward to approximate the $x$ component of the pressure gradient across the $U$ cell since the pressure is defined on cell faces to the 'left' and 'right' of the $U$ value. Analogously, it is straightforward to define the approximation to the $y$ and $z$ components of the pressure gradient on the $V$ and $W$ cells, respectively. One issue that arises for the staggered grid discretization is the ordering of the velocity cells and their relationship to the ordering of the pressure cells. If care is not taken, here, the condition number and sparsity pattern of the resulting matrix suffers. To account for this, when defining each velocity grid we reorder the pressure cells to produce the standard discrete Laplacian stencil illustrated in Figure 2.2 for each velocity component.
This results in an undesirable pattern in the portions of the gradient and divergence matrices associated to the $V$ and $W$ cells, but this is strongly preferable to losing the Laplacian structure in the matrices corresponding to $\Delta V$ and $\Delta W$. Figure 2.4 presents the sparsity pattern for the resulting linear system.

### 2.3.3. Richardson Iteration

Let $A : \mathbb{R}^N \to \mathbb{R}^N$, and $b \in \mathbb{R}^N$, and consider the problem,

$$A x = b. \quad (2.3.41)$$

Write (2.3.41) as

$$x = (I - A)x + b, \quad (2.3.42)$$
and define the Richardson iteration by: given $x^0$, for each $k \geq 1$,

$$x^{k+1} = (I - A)x^k + b.$$  \hspace{1cm} (2.3.43)

More generally, we consider the iteration matrix $M$,

$$x^{k+1} = Mx^k + b,$$  \hspace{1cm} (2.3.44)

while in this case we have $M = (I - A)$.

**Theorem 2.3.2** ([53] p. 6). *If $\|M\| < 1$ then the iteration (2.3.44) converges to*

$$x = (I - M)^{-1}b.$$
for all initial iterates $x^0$.

While it is easiest to present this theorem in finite dimension, the infinite dimensional case can be handled similarly. Convergence relies on the contraction property $\|M\| < 1$. We will encounter an infinite dimensional problem of the following form. Let $H$ be a Hilbert space and let $A \in \mathcal{L}(H, H')$. Further, let $b \in H'$. Find $x \in H$ such that

$$Ax - b = 0 \in V'.$$  \hfill (2.3.45)

Applying the Riesz Representation Theorem 2.1.15, let $A : H \to H$ such that for each $x \in H$,

$$(y, Ax) = Ax(y) \quad \forall y \in H. \hfill (2.3.46)$$

Also, abusing notation, denote by $b \in H$ the element such that

$$(y, b) = b(y) \quad \forall y \in H. \hfill (2.3.47)$$

Then our problem is stated as, find $x \in H$ such that

$$Ax - b = 0 \in H. \hfill (2.3.48)$$

The importance of the Riesz Representation Theorem 2.1.15 here is that it allows us to consider an operator mapping $H$ to itself equivalent to the original operator mapping $H$ to its dual. This will allow us to apply the Banach Contraction Mapping Principle 2.1.24. Similar to the finite dimensional case, we add and subtract $x$ and rearrange to obtain,

$$x = (I - A)x + b, \hfill (2.3.49)$$

and define the Richardson iteration by

$$x^{k+1} = (I - A)x^k + b. \hfill (2.3.50)$$
Then, similar to the proof given in the finite dimensional case, by application of the Banach Contraction Mapping Principle 2.1.24, if the map $T : H \to H$ defined by

$$T(x) = (I - A)x + b$$  \hspace{1cm} (2.3.51)

is a contraction on $H$, the Richardson scheme (2.3.50) will converge to the solution $x$ of (2.3.48). Note that

$$\|T(x) - T(y)\| \leq \|I - A\|\|x - y\|. \hspace{1cm} (2.3.52)$$

Thus if the operator $M = (I - A)$ has norm strictly less than 1, the Richardson iteration converges.

### 2.3.4. Krylov Methods

Unlike the Richardson scheme previously introduced, which falls into the category of a 'stationary' method, the Krylov methods do not have an iteration matrix, and their analysis is more involved. We define

$$x^* = A^{-1}b. \hspace{1cm} (2.3.53)$$

Krylov methods attempt to minimize, at the $k$th iteration, the error over the affine space

$$x_0 + \mathcal{K}_k, \hspace{1cm} (2.3.54)$$

where $x_0$ is the initial iterate and the $k$th Krylov subspace $\mathcal{K}_k$ is

$$\mathcal{K}_k = \text{span} \,(r_0, Ar_0, \ldots, A^{k-1}r_0), \hspace{1cm} (2.3.55)$$

for $k \geq 1$. Here $r_0 = b - Ax_0$ is the initial residual and $\{r_k\}_{k \geq 0}$ denotes the sequence of residuals

$$r_k = b - Ax_k. \hspace{1cm} (2.3.56)$$
The Conjugate Gradient method is intended to solve the system (2.3.41) when $A$ is symmetric positive definite. We recall that $A$ is symmetric when $A^T = A$ and positive definite when

$$x^T Ax > 0, \quad \forall x \neq 0.$$ (2.3.57)

Since $A$ is symmetric positive definite we may define a norm,

$$\|x\|_A = \sqrt{x^T Ax},$$ (2.3.58)

called the $A$-norm.

The $k$-th iterate of the CG method minimizes

$$\phi(x) = \frac{1}{2} x^T Ax - x^T b,$$ (2.3.59)

over the space $x_o + \mathcal{K}_k$. We note that if $\phi(\tilde{x})$ is the minimal value (in $\mathbb{R}^N$) then

$$\nabla \phi(\tilde{x}) = A\tilde{x} - b = 0,$$ (2.3.60)

and so $\tilde{x} = x^*$.

Lemma 2.3.3 ([53] p. 12). Let $S \subset \mathbb{R}^N$. If $x_k$ minimizes $\phi$ over $S$ then $x_k$ also minimizes $\|x^* - x\|_A = \|r\|_{A^{-1}}$ over $S$.

Lemma 2.3.3 implies that

$$\|x^* - x_k\|_A \leq \|x^* - w\|_A$$ (2.3.61)

for all $w \in x_0 + \mathcal{K}$.

The symmetry of the matrix $A$ allows the Spectral Theorem to play a crucial role in analysis of the Conjugate Gradient method.

Theorem 2.3.4 (Spectral Theorem [44] p. 335). Let $T$ be a normal operator on a finite dimensional complex inner product space $V$ or a self-adjoint operator on a finite dimensional real inner product space $V$. Let $c_1, \ldots, c_k$ be the distinct eigenvalues values of $T$. 


Let $W_j$ be the eigenspace associated with $c_j$ and $E_j$ the orthogonal projection of $V$ onto $W_j$. Then $W_j$ is orthogonal to $W_i$ when $i \neq j$, $V$ is the direct sum of $W_1, \ldots, W_k$, and

$$T = c_1E_1 + \ldots + c_kE_k.$$  \hspace{1cm} (2.3.62)

Together with Lemma 2.3.3, the Spectral Theorem 2.3.4 provides that,

$$\|x_k - x^*\|_A \leq \|x_0 - x^*\|_A \min_{p \in \mathbb{P}_k} \max_{z \in \sigma(A)} |p(z)|,$$  \hspace{1cm} (2.3.63)

where $\sigma(A)$ denotes the set of all eigenvalues of $A$.

The following corollary is a consequence of (2.3.63), and will imply several important results for the Conjugate Gradient method.

**Corollary 2.3.5** ([53] p.14). Let $A$ be symmetric positive definite and let $\{x_k\}$ be the CG iterates. Let $k$ be given and let $\{\overline{p}_k\}$ be any $k$-th degree polynomial such that $\overline{p}_k(0) = 1$. Then

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \leq \max_{z \in \sigma(A)} |\overline{p}_k(z)|.$$  \hspace{1cm} (2.3.64)

**Theorem 2.3.6** ([53] p.14). Let $A$ be symmetric positive definite. Then the CG algorithm will find the solution within $N$ iterations.

**Theorem 2.3.7** ([53] p.15). Let $A$ be symmetric positive definite with eigenvectors $\{u_i\}_{i=1}^N$. Let $b$ be a linear combination of $k$ of the eigenvectors of $A$

$$b = \sum_{l=1}^k \gamma_l u_i.$$ \hspace{1cm} (2.3.65)

Then the CG iteration for $Ax = b$ with $x_0 = 0$ will terminate in at most $k$ iterations.

**Theorem 2.3.8** ([53] p.15). Let $A$ be symmetric positive definite. Assume that there are exactly $k \leq N$ distinct eigenvalues of $A$. Then the CG iteration terminates in at most $k$ iterations.
In practice the CG method, and more broadly any iterative method for the solution of a linear system of equations, is not run until an exact solution is found. Rather, some measure of how close the current iterate is to the true solution is used to define a stopping criteria. A common choice is the "relative residual,"

\[ \| b - Ax_k \|_2 \leq \eta \| b \|_2, \quad (2.3.66) \]

where \( \eta \) is the tolerance.

Throughout the exposition on the Conjugate Gradient method we made the assumption that the matrix \( A \) is symmetric. For many systems resulting from applications, this assumption holds, but there are also many systems which are not symmetric, and for these problems the CG method may diverge. The Generalized Minimum Residual method (GMRES) was proposed in 1986 [94] as a Krylov subspace method for nonsymmetric systems. The GMRES method is more difficult to analyze since we will not have the Spectral Theorem at our disposal.

The \( k \)-th iterate of the GMRES method is the solution of the least-squares problem

\[ \arg \min_{x \in x_0 + \mathbb{K}_k} \| b - Ax \|_2. \quad (2.3.67) \]

Notice already the difference between GMRES and the CG method. Here we are minimizing over the 2-norm since \( A \) does not induce a norm due to lack of symmetry. Still, we have the following result

**Theorem 2.3.9** ([53] p.33). *Let \( A \) be nonsingular and let \( x_k \) be the \( k \)-th GMRES iterate. Then for all \( p_k \in \mathbb{P}_k \) with \( p(0) = 1 \),

\[ \| r_k \|_2 = \min_{p \in \mathbb{P}_k} \| p(A)r_0 \|_2 \leq \| p_k(A)r_0 \|_2, \quad (2.3.68) \]

Then we have the corollary
Corollary 2.3.10 ([53] p.33). Let $A$ be nonsingular and let $x_k$ be the $k$-th GMRES iterate. Then for all $p_k \in P_k$ with $p(0) = 1$,

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \|p_k(A)\|_2.$$  \hspace{1cm} (2.3.69)

From this corollary we have a theorem proving finite termination of the GMRES method.

Theorem 2.3.11 ([53] p.34). Let $A \in \mathbb{R}^{N\times N}$ be nonsingular. Then the GMRES algorithm will find the solution within $N$ iterations.

In the case that $A$ is diagonalizable, i.e. there exists a nonsingular matrix (possibly complex) $V$ such that

$$A = VAV^{-1},$$  \hspace{1cm} (2.3.70)

where $\Lambda$ is a diagonal matrix containing the (possibly complex) eigenvalues of $A$ along the main diagonal, we have theorems for the convergence behavior similar to those presented for the CG method. We omit the technical detail here, and point the reader to [53].

Theorem 2.3.12 ([53] p. 35). Let $A \in \mathbb{R}^{N\times N}$ be a nonsingular, diagonalizable matrix. Assume that $A$ has only $k$ distinct eigenvalues. Then GMRES will terminate in at most $k$ iterations.

Theorem 2.3.13 ([53] p. 35). Let $A \in \mathbb{R}^{N\times N}$ be a nonsingular, diagonalizable matrix. Let $b$ be a linear combination of $k$ of the eigenvectors of $A$. Then the GMRES iteration, with $x_0 = 0$, for $Ax = b$ will terminate in at most $k$ iterations.

Again we note that one does not typically solve the problem exactly by waiting $N$ (or $k$) iterations, but rather defines a stopping criteria which measures how far the iterate is from the true solution and stops when a tolerance is satisfied.

Implementation of the GMRES method is much more involved than the CG method. In particular, one must build an orthogonal basis for the Krylov spaces at each iteration.
to be stored, which requires the introduction of the Arnoldi algorithm, which implements Gramm-Schmidt orthogonalization for the space $x_0 + \mathcal{X}_k$. This is beyond the scope of this work and so we point the reader to [53] Chapter 3 for details on implementing GMRES.

### 2.3.5. Preconditioners

As the sparsity pattern illustrated in Figure 2.4 suggests, the linear system resulting from a finite volume discretization of the Stokes’ equations is a saddle-point system. In particular, we note that we have

$$
\begin{bmatrix}
-\mu \triangle & 0 & 0 & \nabla_x \\
0 & -\mu \triangle & 0 & \nabla_y \\
0 & 0 & -\mu \triangle & \nabla_z \\
\nabla_x & \nabla_y & \nabla_z & 0
\end{bmatrix}
\begin{bmatrix}
U \\
V \\
W \\
P
\end{bmatrix} =
\begin{bmatrix}
F_x \\
F_y \\
F_z \\
0
\end{bmatrix}
$$

or, equivalently,

$$
\begin{bmatrix}
A & B \\
B^T & 0
\end{bmatrix}
\begin{bmatrix}
\vec{U} \\
P
\end{bmatrix} =
\begin{bmatrix}
F \\
0
\end{bmatrix}
$$

where $A$ corresponds to the block-Laplacian operator acting on the velocity components $\vec{U} = [U \ V \ W]^T$, $B$ is the discrete gradient operator acting on the pressure $P$ and $B^T$ is the discrete divergence operator acting on the velocity $\vec{U} = [U \ V \ W]^T$.

It is well known that saddle-point systems are very poorly conditioned, therefore linear solver performance relies heavily on appropriately preconditioning the system. We begin by discussing the basic ideas behind preconditioning for the linear system (2.3.41), where $A \in \mathbb{R}^{N \times N}$ is a matrix, $b \in \mathbb{R}^{N}$ is a known vector and $x \in \mathbb{R}^{N}$ is a sought vector of unknowns. Preconditioning refers to transforming the system (2.3.41) into a system with more favorable properties for iterative solution. A preconditioner is a matrix that performs that transformation. Most often, preconditioning attempts to improve the spectral properties of the matrix, i.e. to improve the condition number of the matrix. Next we make some of these terms precise, following [107].
Conditioning refers to the perturbation behavior of a mathematical problem. In the abstract, we can consider the problem given by a function \( f : Y \rightarrow X \) from the normed vector space \( Y \) of data into the normed vector space \( X \) of solutions. An instance of this problem is then given by the data \( y \in Y \). A well-conditioned problem is one with the property that all small perturbations of \( y \) result in only small changes in \( f(y) \). An ill-conditioned problem is one with the property that some small perturbation in \( y \) results in a large change in \( f(y) \). The meaning of the terms "small" and "large" often depend on the problem. In particular, we can consider perturbations on an absolute scale or on a relative scale.

**Definition 2.3.14** (Absolute Condition Number). Let \( \delta y \) denote a small perturbation of \( y \), and write \( \delta f = f(y + \delta y) - f(y) \). The absolute condition number of the problem \( f \) at \( y \) is defined by

\[
\hat{\kappa} = \lim_{\delta \rightarrow 0} \sup_{\|\delta y\| \leq \delta} \frac{\|\delta f\|}{\|\delta x\|}. \tag{2.3.73}
\]

**Definition 2.3.15** (Relative Condition Number). The relative condition number \( \kappa = \kappa(y) \) is defined by

\[
\kappa = \lim_{\delta \rightarrow 0} \sup_{\|\delta y\| \leq \delta} \left( \frac{\|\delta f\|}{\|\delta y\|} \right) \left( \frac{\|\delta y\|}{\|y\|} \right). \tag{2.3.74}
\]

The next theorem removes the additional layer of abstraction and deals with the condition number for a problem of the type (2.3.41) over the complex number field.

**Theorem 2.3.16** ([107] p.94). Let \( A \in \mathbb{C}^{m \times m} \) be nonsingular and consider the equation \( Ax = b \). The problem of computing \( b \), given \( x \), has condition number

\[
\kappa = \|A\| \frac{\|x\|}{\|b\|} \leq \|A\| \|A^{-1}\| \tag{2.3.75}
\]

with respect to perturbations of \( x \). The problem of computing \( x \), given \( b \), has condition number

\[
\kappa = \|A^{-1}\| \frac{\|b\|}{\|x\|} \leq \|A\| \|A^{-1}\| \tag{2.3.76}
\]
with respect to perturbations of $b$.

The ubiquity of the term $\|A\|\|A^{-1}\|$ is such that it is given its own name.

**Definition 2.3.17** (Condition Number of a Matrix). The condition number $\kappa(A)$ of the non-singular matrix $A$ relative to the norm $\| \cdot \|$ is the number

$$\kappa(A) = \|A\|\|A^{-1}\|. \quad (2.3.77)$$

If $\kappa(A)$ is small, $A$ is said to be well-conditioned while if $\kappa(A)$ is large, $A$ is ill-conditioned. If $A$ is singular, than it is customary to write $\kappa(A) = \infty$.

Notice that this definition refers only to the matrix $A$ and not to the problem (2.3.41). Also in Theorem 2.3.16 we held $A$ fixed and perturbed $x$ or $b$. In the context of solving a linear system of equations we are interested in fixing $b$ and examining the condition number of the problem of finding $x$. That is, we consider the behavior of the problem $A \mapsto x = A^{-1}b$ when $A$ is perturbed by infinitesimal $\delta A$.

**Theorem 2.3.18** ([107] p. 95). Let $b$ be fixed and consider the problem of computing $x = A^{-1}b$, where $A$ is square and nonsingular. The condition number of this problem with respect to perturbations in $A$ is

$$\kappa = \|A\|\|A^{-1}\| = \kappa(A). \quad (2.3.78)$$

Thus we see that for our problem the condition number of the system (2.3.41) is determined by the condition number of the matrix $A$.

The idea behind preconditioning is to define a preconditioner $M$ and premultiply the equation by a matrix $M^{-1}$, to arrive at the system

$$M^{-1}Ax = M^{-1}b. \quad (2.3.79)$$

If we have done our job well, then $M^{-1} \approx A^{-1}$ and the matrix $M^{-1}A$ is a well-conditioned matrix. We can then apply an iterative method like the Conjugate Gradient method or the
GMRES method to the solution of the system (2.3.79) and expect improved performance. Typically one does not explicitly construct $M^{-1}$, but rather solves a system of linear equations

$$M y = c.$$  (2.3.80)

However, one must proceed with caution. The application of the preconditioner comes with a computational cost, and one must be careful not to choose a preconditioner whose cost outweighs its benefits. Two extreme examples commonly given to illustrate this balance are $M_1 = A$ and $M_2 = I$. In the first case, $M_1^{-1}A = I$, which is of course extremely well-conditioned.

$$\kappa(M_1^{-1}A) = \|I\|\|I^{-1}\| = 1.$$  (2.3.81)

But, this requires solving a system $Ay = c$, and so the solution time is the same as solving the original system. On the other hand, with $M_2 = I$, construction time is trivial, but the condition number of the resulting linear system

$$\kappa(M_2^{-1}A) = \|IA\|\|(IA)^{-1}\| = \|A\|\|A^{-1}\| = \kappa(A)$$  (2.3.82)

is unchanged. Thus the set of useful preconditioners live somewhere in between these two extreme cases, structured enough that the system $My = c$ can be solved quickly, but close enough to $A$ in some sense such that an iterative method for (2.3.79) converges quickly.

The meaning of $M$ being 'close enough to $A$' is vague. In general if the eigenvalues of $M^{-1}A$ are close to 1 and $\|M^{-1}A - I\|_2$ is small, many standard iterative techniques will perform well. However, preconditioners that do not satisfy these conditions can perform well, as well. In general [107] p. 314 the following rule of thumb is adequate: A preconditioner $M$ is good if $M^{-1}A$ is not too far from normal and its eigenvalues are clustered.

Next we review the specific preconditioners that are employed by default in the software HybGe-Flow3D [24] described in Section 4.4.. We will take advantage of the block
structure of the system (2.3.71), but first must describe the component preconditioners of the block-strategy.

### 2.3.5.1 Jacobi Preconditioning

One of the simplest preconditioning strategies is the Jacobi, or Diagonal preconditioning strategy [5]. This method uses the diagonal entries of $A$ to precondition the system.

$$M_{ij} = \begin{cases} A_{i,i} & i = j, \\ 0 & i \neq j \end{cases}.$$  \hfill (2.3.83)

The inverse $M^{-1}$ is then easily written down explicitly,

$$M_{ij}^{-1} = \begin{cases} \frac{1}{A_{i,i}} & i = j, \\ 0 & i \neq j \end{cases}.$$  \hfill (2.3.84)

In addition to being easily computable, this preconditioner requires no additional storage beyond what is required for the matrix $A$, which we will see is a considerable advantage over other preconditioners.

A common extension of this strategy is the so-called Block Jacobi preconditioner. Let $S = \{1, \ldots, N\}$ be partitioned as $S = \bigcup_i S_i$ with the sets $S_i$ mutually disjoint, then we define

$$M_{i,j} = \begin{cases} A_{i,j} & \{i, j\} \subset S_k \text{ for some } k, \\ 0 & \text{otherwise} \end{cases}.$$  \hfill (2.3.85)

The resulting matrix $M$ is block-diagonal.

Jacobi type preconditioners require very little storage and are easy to implement. Additionally, their application is trivially parallelizable. In particular a common strategy in the Block Jacobi method is to define the partitioning of blocks to coincide with the division of variables over processors. However, more sophisticated preconditioners typically result in larger performance gains.
2.3.5.2 Incomplete LU factorization

A general Incomplete LU (ILU) factorization computes a sparse lower triangular matrix $L$ and a sparse upper triangular matrix $U$ so that the residual matrix $R = LU - A$ satisfies some constraints, e.g. having zero entries in some pre-defined locations [95]. Because incomplete factorization is a much more complicated process than, e.g. the Jacobi type strategy, several variants are worth considering. In addition, it is not guaranteed for all matrices that for a given $ILU$ strategy the incomplete factorization will even exist or that the strategy will not break down (due, e.g. to division by zero). Following [95] we first describe an ILU preconditioner built for $M$-matrices. We then discuss the $p$-sweep style ILU factorization, beginning with $ILU(0)$, the simplest form of the ILU preconditioner. We will then mention the power($q$)-pattern [62] strategy, which is an advanced technique designed to overcome some of the difficulties present in a simple ILU preconditioner.

A general strategy for building ILU factorizations can be obtained from performing Gaussian elimination and dropping some elements in predetermined nondiagonal positions. First we must define $M$-matrices.

**Definition 2.3.19** (M-matrix [95] p. 30). A matrix is said to be an $M$-matrix if it satisfies the following four properties:

1. $A_{i,i} > 0$ for $i = 1, \ldots, N$.

2. $A_{i,j} \leq 0$ for $i \neq j$, $i, j = 1, \ldots, N$.

3. $A$ is nonsingular.

4. The entries of $A^{-1}$ are nonnegative.

**Theorem 2.3.20** ([95] p. 302). Let $A$ be an $M$-matrix and let $A_1$ be the matrix obtained from the first step of Gaussian elimination. Then $A_1$ is an $M$-matrix.
If we now drop certain elements from the result of the Gaussian elimination step off of the main diagonal, any element that is dropped is a nonpositive entry transformed into a zero. Then the resulting matrix $\tilde{A}_1$ can be written

$$\tilde{A}_1 = A_1 + R,$$  \hfill (2.3.86)

where the entries in $R$ satisfy $r_{ii} = 0$, $r_{ij} \geq 0$, and we have

$$(A_1)_{ij} \leq (\tilde{A}_1)_{ij}$$  \hfill (2.3.87)

for each $i, j \in 1, \ldots, N$. Thus $\tilde{A}_1$ is also an $M$-matrix. Proceeding inductively, we see that $\tilde{A}_m$ is an $M$-matrix for the $m$-th step of the Gaussian elimination process, all the way until the factorization is complete. In the simplest case, we choose some non-zero pattern in advance for the factored matrix to determine the entries which will be dropped at each step of the Gaussian elimination. The only restriction being that the diagonal entries should be not be included in the set of indices to drop. Defining the zero pattern set $P$

$$P \subset \{(i, j) \mid i \neq j; 1 \leq i, j \leq N\},$$  \hfill (2.3.88)

an Incomplete LU factorization $ILU_P$ can be computed as presented in Algorithm 1.

**Theorem 2.3.21** ([95] P. 303). Let $A$ be an $M$-matrix and $P$ a given zero pattern. Then Algorithm 1 does not break down and produces an incomplete factorization

$$A = LU - R$$  \hfill (2.3.89)

which is a regular splitting of $A$.

It is worth noting that the Static Pattern ILU algorithm presented in Algorithm 1 assumes a certain loop pattern in the Gaussian elimination process. Algorithm 1, in particular, assumes the $k, i, j$ variant. However, the most commonly employed Gaussian
elimination is designed for continuous row access, and is referred to as the $i,k,j$ variant. In the high performance computing context this is an important consideration. Row continuous memory access allows each row to be computed one at a time and accumulated in a row-oriented data structure such as the CSR format. In this setting, the general static ILU factorization takes the form presented in Algorithm 2. The following proposition

Algorithm 2: General ILU Factorization, IKJ Version [[95] p. 305]

<table>
<thead>
<tr>
<th>for $i = 2, \ldots, N$ do</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $k = 1, \ldots, i - 1$ and if $(i, k) \notin P$ do</td>
</tr>
<tr>
<td>$A_{ik} := A_{ik}/A_{kk}$</td>
</tr>
<tr>
<td>for $j = k + 1, \ldots, N$ and for $(i, j) \notin P$ do</td>
</tr>
<tr>
<td>$A_{ij} := A_{ij} - A_{ik}A_{kj}$</td>
</tr>
</tbody>
</table>

states that the $IKJ$ variant Algorithm 2 produces the same factorization as the original procedure in Algorithm 1.

**Proposition 2.3.22** ([95] p. 305). Let $P$ be a zero pattern. Then the ILU factors produced by the $KIJ$-based Algorithm 1 and the $IKJ$-based Algorithm 2 are identical if
they can both be computed.

We note that this equivalence is a special property of the static pattern ILU. If the nonzero pattern is determined dynamically as the elimination proceeds then the resulting ILU factorizations may be different.

We now turn to the Zero Fill-in ILU, also referred to as $ILU(0)$. This factorization technique is a static pattern technique where the zero pattern $P$ is precisely the zero pattern of the matrix $A$. This pattern has the advantage of restricting the memory requirement for the preconditioner $M = LU$ to near the memory required for $A$. In particular, Figure 2.5 shows the nonzero pattern of the resulting matrices when $A$ is the 5-point Laplacian stencil.

Unfortunately, the $ILU(0)$ factorization may be insufficient to obtain adequate rates of convergence for iterative methods such as CG or GMRES applied to difficult large problems. Extending this method leads to the class of methods labeled $ILU(p)$, which can be understood as "$ILU$ with $p$th-order fill-ins." To understand $ILU(p)$, we first introduce $ILU(1)$. We first define the nonzero set of a matrix $B$, $NZ(B)$

$$NZ(B) = \{(i, j) \mid 1 \leq i, j \leq N \text{ such that } B_{ij} \neq 0\}.$$  

(2.3.90)

Additionally, we refer to row $i$ of a matrix $B$ by $B_{i\ast}$. We then note that $ILU(0)$ is precisely the general static pattern ILU algorithm with nonzero pattern $NZ(A)$. Notice that the resulting matrix $M = LU$ has nonzero pattern $NZ(LU) \supset NZ(A)$. Thus we see that the ILU method increases the size of the nonzero set when going from $A$ to $M = LU$. The $ILU(1)$ method allows "1st order fill-ins" by setting the nonzero pattern to $NZ_1(A)$ where $NZ_1(A) = NZ(L_0U_0)$ and $L_0, U_0$ are the result of applying $ILU(0)$ to $A$. Figure 2.6 shows the nonzero pattern of the resulting matrices where Augmented $A$ refers to the sparsity pattern $NZ_1(A)$. Thus intuitively when going from $p = 0$ to $p = 1$ we are making $M$ closer to $A$ by allowing more nonzero entries. This intuition will hold for $p > 1$ as well, but we will require a more careful definition of "fill-ins."
Definition 2.3.23 ([95] p. 312). The initial level of fill of an element $A_{ij}$ of a sparse matrix $A$ is defined by

$$lev_{ij} = \begin{cases} 
0, & \text{if } A_{ij} \neq 0, \text{ or } i = j, \\
\infty, & \text{otherwise}
\end{cases} \quad (2.3.91)$$

Each time this element is modified in line 5 of Algorithm 2 its level of fill must be updated by

$$lev_{ij} = \min\{lev_{ij}, lev_{ik} + lev_{kj} + 1\}. \quad (2.3.92)$$
FIGURE 2.6: ILU(1) sparsity patterns for 5-point Laplacian matrix, image credit ([95] p.312)
We note that this definition can be understood in terms of the graph of the matrix, and refer the reader to [[95] p. 312-314] for the exposition. With this definition at hand, we can define the $ILU(p)$ method, outlined in Algorithm 3.

<table>
<thead>
<tr>
<th>Algorithm 3: ILU(p) [[95] p. 314]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  for $1 \leq i, j \leq N$ with $A_{ij} \neq 0$ do</td>
</tr>
<tr>
<td>2       Define $lev(A_{ij}) = 0$</td>
</tr>
<tr>
<td>3  for $i = 2, \ldots, N$ do</td>
</tr>
<tr>
<td>4        for $k = 1, \ldots, i - 1$ and for $lev(A_{ik}) \leq p$ do</td>
</tr>
<tr>
<td>5            Compute $A_{ik} := A_{ik}/A_{kk}$</td>
</tr>
<tr>
<td>6            Compute $A_{i*} := A_{i*} - A_{ik}A_{k*}$</td>
</tr>
<tr>
<td>7        Update the levels of fill of the nonzeros $A_{ij}$s by (2.3.91)</td>
</tr>
<tr>
<td>8        Replace any element in row $i$ with $lev(A_{ij}) &gt; p$ by zero</td>
</tr>
</tbody>
</table>

There are many shortcomings of the $ILU(p)$ algorithm. First, in general if $p > q$ then preconditioning with $ILU(p)$ will use more memory and occupy more compute time than $ILU(q)$. These differences can be dramatic. Even more crucially, if one is working with an indefinite matrix the level of fill-in is not necessarily a good indicator of the size of the elements being dropped. The algorithm may drop large elements and result in an inaccurate factorization, in the sense that $R = LU - A$ is large. Several techniques have been designed to remedy this, but in our work we deal with definite matrices, and so we will not explore these variants here.

More relevant to our work is that applying the $ILU(p)$ preconditioner can be compute intensive and is simultaneously difficult to parallelize effectively. A variation of the $ILU(p)$ method called the Power($q$)-pattern enhanced $ILU(p,q)$ is presented in [62] which provides for increased parallelism in the application of an $ILU(p)$ preconditioner. This is also sometimes referred to as "Multicolored-ILU," and we will employ this method as
a component of our preconditioning strategy in the software HybGe-Flow3D presented in Section 4.4.

2.3.5.3 Saddle-Point Preconditioning

Consider the saddle-point system (2.3.72). Block preconditioning for saddle-point systems is based on preconditioning for $A$ and the Schur complement of the system separately.

If we multiply the top block line of the system (2.3.72) by $B^T A^{-1}$ and subtract the result from the second row, we obtain the system,

$$
\begin{bmatrix}
A & B \\
0 & -B^T A^{-1} B \\
\end{bmatrix} 
\begin{bmatrix}
\bar{U} \\
P \\
\end{bmatrix} =
\begin{bmatrix}
F \\
-B^T A^{-1} F \\
\end{bmatrix}
$$

(2.3.93)

**Definition 2.3.24 (Schur Complement for a Saddle-point System).** Given a saddle-point system (2.3.72), the Schur complement of the system is the matrix

$$
S = B^T A^{-1} F.
$$

(2.3.94)

A common strategy, then, is to produce a preconditioner,

$$
M \approx
\begin{bmatrix}
A & B \\
0 & -S \\
\end{bmatrix}.
$$

(2.3.95)

The strategy we employ is to create a block diagonal preconditioner

$$
M \approx
\begin{bmatrix}
A & 0 \\
0 & -S \\
\end{bmatrix}.
$$

(2.3.96)

This involves the creation of two preconditioners,

$$
P_A \approx A,
$$

(2.3.97)

and

$$
P_S \approx -S,
$$

(2.3.98)
which together form

\[
M = \begin{bmatrix}
P_A & 0 \\
0 & P_S
\end{bmatrix}
\] (2.3.99)

Denote by \( D_A \) the Jacobi preconditioner for the matrix \( A \). Then for \( P_S \) we take,

\[
P_S = -B^T D_A B.
\] (2.3.100)

For \( P_A \), we notice that the upper left matrix in (2.3.71) is block diagonal,

\[
\begin{bmatrix}
-\mu \Delta & 0 & 0 \\
0 & -\mu \Delta & 0 \\
0 & 0 & -\mu \Delta
\end{bmatrix}
\] (2.3.101)

We would like to take advantage of this fact in devising a preconditioning strategy. In particular, denote by \( P_{i,q,p} \) the \( ILU(p,q) \) preconditioner for the discrete Laplacian for velocity component \( i \). Then we take

\[
P_A = \begin{bmatrix}
P_{1,p,q} & 0 & 0 \\
0 & P_{2,p,q} & 0 \\
0 & 0 & P_{3,p,q}
\end{bmatrix}
\] (2.3.102)

and so we use the preconditioner

\[
M = \begin{bmatrix}
P_{1,p,q} & 0 & 0 & 0 \\
0 & P_{2,p,q} & 0 & 0 \\
0 & 0 & P_{3,p,q} & 0 \\
0 & 0 & 0 & P_S
\end{bmatrix}
\] (2.3.103)

which has shown strong results for the modified Stokes model we present in Chapter 4.
3. CHARGE TRANSPORT IN HETEROSTRUCTURE SEMICONDUCTORS

In this chapter we consider the problem of modeling charge transport in semiconductor structures with heterojunction interfaces. In particular, we develop and analyze domain decomposition techniques designed for partial differential equation models with non-standard transmission conditions, resulting from multiscale phenomena. This work is motivated by the collaboration of computational mathematicians, physicists, and material scientists who are interested in such structures for the purpose of building more efficient solar cells. Typically, when a photon with excess energy strikes a solar cell the excess energy is wasted. However, occasionally, a photon with enough can cause impact ionization to occur, essentially producing two conductive electrons and thus not wasting the excess energy. It is hypothesized that the presence of a heterojunction can increase the likelihood of impact ionization occurring. Much of the work appearing in this section was first presented in publications [32, 22, 23].

Understanding charge transport in a heterostructure semiconductor at the design scale is a fundamentally multiscale problem. The physics at the heterojunction strictly require a multiscale approach. A heterojunction is an interface between distinct semiconductor materials. This interface has a positive width on the angstrom scale. However, the scale of the bulk semiconductor for many applications, e.g. solar cell design and optimization, is that of microns. Furthermore, we anticipate that in the heterojunction interface region the primary variables of a continuum description of the bulk material vary widely. Thus the challenge is to correctly account for the physics at the interface without making the global problem computationally intractable. Our approach is to model the design scale, bulk semiconductor region with the well-known drift-diffusion equations (DD), connected across the heterojunction interface by a set of transmission conditions known as
the thermionic emission equations (TEM). In this model, the heterojunction region is approximated as an idealized abrupt interface. To account for the physics at the microscale in the heterojunction region, a density functional theory (DFT) calculation is employed, which provides data to the TEM model.

Before resting confident that the multiscale issues have been addressed, the issue of numerical simulation must be addressed. It is well known in the semiconductor community that the drift-diffusion system, even for a homojunction (single material), is notoriously difficult to simulate due to the non-linear coupled nature of the model as well as the presence of boundary and interior layer behavior stemming from wide variation in data within the materials. To make matters worse, one must have a method for dealing with the presence of the transmission conditions from the TEM equations. A naive attempt to address the TEM model would be to insert the transmission conditions into the linear system resulting from a discretization of the drift-diffusion system in the material; however, this has the predictable effect of worsening an already poorly conditioned and highly non-linear problem. A better approach begins by noticing that the design scale model, given as partial differential equations posed on subdomains connected by transmission conditions, is a natural candidate for a domain decomposition approach. The difficulty, then, lies in the development of domain decomposition methods capable of handling the unusual transmission conditions present in the heterostructure model. In this work we develop iterative substructuring domain decomposition techniques designed for the unusual transmission conditions in the thermionic emissions model. We provide convergence analysis and numerical demonstration of the methods. We extend this previous work with a generalization of the transmission conditions present in the TEM in order to open the algorithms up to other applications. In addition we present analysis of the equations in the heterostructure model based on the domain decomposition strategy that represents a first step towards the first analysis of the heterostructure drift-diffusion model.
The work in this section is organized as follows. In Section 3.1, we introduce the continuum model used in simulating charge transport in heterostructure semiconductors including the drift-diffusion system and the thermionic emission model. Additionally we outline the analysis of the single domain drift-diffusion model. In Section 3.2, we present iterative substructuring methods designed for the heterostructure model. We present a background on the mathematical aspects of iterative substructuring methods, and prove convergence results for our algorithms. Additionally, we present analysis of the component equations of the heterostructure model and outline a roadmap to a complete analysis of the coupled model. In Section 3.3, we describe a sensitivity study that considers the impact of error in the density functional model on the quantities of interest in the continuum scale model. In Section 3.4, we present numerical results including convergence studies, mesh independence studies and iteration counts on model problems, as well as results from simulation of true heterostructure models. Finally we perform sensitivity studies on two heterostructure models.
3.1. Modeling Charge Transport Across Heterojunction Interfaces

In this section we describe the models used to simulate charge transport in semiconductor structures with heterojunctions including the drift-diffusion model and the thermionic emission equations. The angstrom scale model, density functional theory, is reviewed in Appendix A. Additionally, we review the mathematical analysis of the single domain drift-diffusion equations.

Let \( \Omega \subset \mathbb{R}^N \), \( N \in \{1, 2, 3\} \), be an open connected set with a Lipschitz boundary \( \partial \Omega \). Let \( \Omega_i, i \in \{1, 2\} \) be two non-overlapping subsets of \( \Omega \) s.t. \( \Omega_1 \cup \Omega_2 = \Omega \), \( \Omega_1 \cap \Omega_2 = \emptyset \), and denote \( \Gamma := \Omega_1 \cap \Omega_2 \). We assume \( \Gamma \) is an \( N-1 \) dimensional manifold and \( m(\Gamma \cap \partial \Omega) = 0 \), where \( m \) denotes the \( N-1 \) dimensional Lebesgue measure. We adopt the following usual notation: \( w_i = w_{\mid \Omega_i} \), \( w^{\Gamma}_i = \lim_{x \to \Gamma} w_i \), and \( [w]_{\Gamma} = w^{\Gamma}_2 - w^{\Gamma}_1 \) denotes the jump of \( w \). We denote by \( \nu^i \) the outward normal of \( \Omega_i \) on \( \Gamma \), and define \( \nu := \nu^1 \).

3.1.1. Drift-Diffusion

In the bulk semiconductor domains \( \Omega_i \), the charge transport is described by the drift diffusion system, consisting of a Poisson equation solved for electrostatic potential \( \psi \) and two continuity equations solved for electron and hole densities \( n \) and \( p \).

\[
-\nabla \cdot (\epsilon_i \nabla \psi_i) = p_i - n_i + N_T, \tag{3.1.1}
\]

\[
-\nabla \cdot D_{n,i}(n_i \nabla \psi_i + \nabla n_i) = -R_i(n_i, p_i), \tag{3.1.2}
\]

\[
-\nabla \cdot D_{p,i}(p_i \nabla \psi_i + \nabla p_i) = R_i(n_i, p_i). \tag{3.1.3}
\]

Here \( N_T(x) = N_D(x) - N_A(x) \) is the given net doping profile including the donor \( N_D \) and acceptor \( N_A \) doping. \( R_i(n_i, p_i) \) is a model for electron and hole generation and recombination, typically a rational function of carrier densities. \( \epsilon_i \) denotes electrical permittivity, and \( D_{n,i} \) and \( D_{p,i} \) are the electron and hole diffusivities, respectively. We require that \( \epsilon_i \) is non-degenerate and bounded from above, i.e. there exists \( \epsilon^0_i > 0 \) such that \( \epsilon_i \geq \epsilon^0_i \) on \( \Omega_i \).
and $\epsilon_i^1 > 0$ such that $\epsilon_i < \epsilon_i^1$ on $\Omega_i$. Similarly, we assume there exists constants $D_{n,i}^j$, $D_{p,i}^j$, $i = 1, 2$, $j = 0, 1$, ensuring non-degeneracy and boundedness of the continuity equations (3.1.2)-(3.1.3).

For analysis and computation it is convenient to introduce an alternative set of variables. The Slotboom variables $u, v$ are related to $n, p$ by

\begin{align*}
    n &= \delta_n^2 e^{\psi} u, \\
    p &= \delta_p^2 e^{-\psi} v.
\end{align*}

The scaling parameters $\delta_n^2$ and $\delta_p^2$ depend on the material and doping profile. In Slotboom variables the continuity equations (3.1.2)-(3.1.3) are self-adjoint, though we pay for this improvement in the Poisson equation (3.1.1) which is then semi-linear. Henceforth we will consider the system in Slotboom variables,

\begin{align*}
    -\nabla \cdot (\epsilon_i \nabla \psi_i) &= \delta_p^2 e^{-\psi} v_i - \delta_n^2 e^{\psi} u_i + N_T := q(\psi_i, p_i, n_i) \quad (3.1.6) \\
    -\nabla \cdot (D_{n,i} \delta_n^2 e^{\psi} \nabla u_i) &= R(\psi_i, u_i, v_i), \quad (3.1.7) \\
    -\nabla \cdot (D_{p,i} \delta_p^2 e^{-\psi} \nabla v_i) &= -R(\psi_i, u_i, v_i). \quad (3.1.8)
\end{align*}

### 3.1.2. Thermionic Emission

We are interested in simulating charge transport in a semiconductor structure consisting of distinct semiconductor materials connected by a heterojunction interface. A true heterojunction interface is a region of positive width on the angstrom scale. However, a typical device scale for many applications, e.g. solar cells, is that of microns. Together with the fact that we expect large variation in the primary variables of the charge transport models in the heterojunction region means we cannot simulate transport in the heterojunction region with the drift diffusion system. Instead we adopt, at the macroscale, the thermionic emission model introduced in [46] where the interface is treated as an idealized abrupt interface. As Figure 3.1 illustrates, this results in discontinuities in the primary variables of the drift-diffusion system.
The thermionic emission model thus consists of transmission conditions for the primary variables across the idealized abrupt interface. These are a jump discontinuity $\psi_{\Delta}$ in the potential and unusual Robin-like conditions relating electron and hole densities across the interface,

\begin{align}
[\psi]\Gamma &= \psi_{\Delta} \\
\left[\frac{\partial \psi}{\partial \nu}\right]_{\Gamma} &= 0, \\
(D_n \delta_n^2 e^\psi \nabla u)_{\Gamma} &= a_2^n (e^\psi u)_2 - a_1^n (e^\psi u)_1, \\
\left[D_n \delta_n^2 e^\psi \nabla u\right]_\Gamma &= 0, \\
(D_p \delta_p^2 e^{-\psi} \nabla v)_{\Gamma} &= a_2^p (e^{-\psi} v)_2 - a_1^p (e^{-\psi} v)_1, \\
\left[D_p \delta_p^2 e^{-\psi} \nabla v\right]_\Gamma &= 0.
\end{align}

Here the coefficients $a_i^n$ and $a_i^p$ are, up to scaling, mean electron thermal velocities that depend on the temperature $T$ and the sign of $\psi_{\Delta}$. The jump $\psi_{\Delta}$ must be calculated at the
angstrom scale in the heterojunction region. Typically this is accomplished via density functional theory, described in Appendix A. More details on the coefficients can be found in [32].

Note that the homogeneous jumps (3.1.10), (3.1.12), (3.1.14) are an assumption based on observation of the shape of the flux across a heterojunction interface region. Due to the loss of scale we have no a-priori reason to expect this homogeneity, and thus future models may need to account for a non-homogeneous jump in fluxes. The algorithms developed in this work are capable of handling this extension.

For each equation in the drift-diffusion system the thermionic emission model provides a pair of transmission conditions. For the potential equation (3.1.6) we have (3.1.9) and (3.1.10), for the equation for electron transport (3.1.7) we have (3.1.11) and (3.1.12), and for the equation for hole transport (3.1.8) we have (3.1.13) and (3.1.14).

Finally, the system (3.1.6)-(3.1.14) are completed by appropriate boundary conditions. Appropriate boundary conditions for the full system are a Dirichlet condition for the potential and Robin conditions for the carrier densities [32].

3.1.3. Analysis: Single Material Domain

In a single semiconductor domain the well-posedness of the drift-diffusion system is well understood. We review this analysis following [64]. Later we will consider the use of our domain decomposition framework to extend the model analysis to the two-domain problem.

Following [[64], Sec 3.2] we consider the drift diffusion system in one material domain $U$ (bounded in $\mathbb{R}^d$, $d \in \{1, 2, 3\}$ of class $C^{0,1}$) with no heterojunction in Slotboom variables
with mixed Dirichlet and homogeneous Neumann boundary conditions.

\[-\nabla \cdot (\epsilon \nabla \psi) = q(\psi, u, v), \quad x \in U, \quad (3.1.15)\]
\[-\nabla \cdot (D_n \delta_n^2 e^\psi \nabla u) = R(\psi, u, v), \quad x \in U, \quad (3.1.16)\]
\[-\nabla \cdot (D_p \delta_p^2 e^{-\psi} \nabla v) = -R(\psi, u, v), \quad x \in U, \quad (3.1.17)\]
\[\psi|_{\partial U_D} = \psi_D, \quad u|_{\partial U_D} = u_D, \quad v|_{\partial U_D} = v_D, \quad (3.1.18)\]
\[\frac{\partial \psi}{\partial \nu}|_{\partial U_N} = \frac{\partial u}{\partial \nu}|_{\partial U_N} = \frac{\partial v}{\partial \nu}|_{\partial U_N} = 0. \quad (3.1.19)\]

We will be interested in the weak formulation of the system \((3.1.15)-(3.1.19)\). We write \(\psi = \psi_0 + \psi_D, \ u = u_0 + u_D, \) and \(v = v_0 + v_D\) and wish to find \((\psi, u, v) \in (H^1_0)^3\) such that

\[\int_U \epsilon \nabla \psi_0 \cdot \nabla \phi = \int_U q(\psi_0 + \psi) \phi - \int_U \epsilon \nabla \psi_D \cdot \nabla \phi, \quad (3.1.20)\]
\[\int_U D_n \delta_n^2 e^\psi \nabla u_0 \cdot \nabla \phi = \int_U R(\psi, u, v) \phi - \int_U D_n \delta_n e^\psi u_D \cdot \nabla \phi, \quad (3.1.21)\]
\[\int_U D_p \delta_p^2 e^{-\psi} \nabla v_0 \cdot \nabla \phi = \int_U -R(\psi, u, v) \phi - \int_U D_p \delta_p^2 e^{-\psi} v_D \cdot \nabla \phi, \quad (3.1.22)\]

for all \(\phi \in H^1_0\).

We make the following assumptions:

(A1) The \(N - 1\) dimensional Lebesgue measure of \(\partial U_D\) is positive.

(A2) The Dirichlet boundary data satisfies

\[(\psi_D, u_D, v_D) \in (H^1(U))^3, \quad (\psi_D, u_D, v_D)|_{\partial U_D} \in (L^\infty(\partial U_D))^3 \quad (3.1.23)\]

and there is a \(K \geq 0\) s.t.

\[e^{-K} \leq \inf_{\partial U_D} u_D, \inf_{\partial U_D} v_D; \quad \sup_{\partial U_D} u_D, \sup_{\partial U_D} v_D \leq e^K. \quad (3.1.24)\]

(A3) The recombination-generation rate \(R\) satisfies \(R = F(x, \psi, u, v)(uv - 1)\), where \(F \in C^1\) for all \(x \in U; \ F \in L^\infty, \nabla_{(\psi, u, v)} F \in L^\infty, \) and \(F \geq 0\).
(A4) The mobilities $D_n, D_p$ satisfy:

(i) $D_n = D_n(x, \nabla \psi), D_p = D_p(x, \nabla \psi), D_n, D_p \colon U \times \mathbb{R}^N \to \mathbb{R}$.

(ii) $0 < D_{n,0} \leq D_n \leq D_{n,1}$ for some $D_{n,0}, D_{n,1} \in \mathbb{R}$.

(iii) $0 < D_{p,0} \leq D_p \leq D_{p,1}$ for some $D_{p,0}, D_{p,1} \in \mathbb{R}$.

(iv) $|D_n(x, y_1) - D_n(x, y_2)| + |D_p(x, y_1) - D_p(x, y_2)| \leq L|y_1 - y_2|$ for some $L > 0$.

(A5) The doping profile $N_T$ satisfies $N_T \in L^\infty(U)$.

**Theorem 3.1.1** ([64], p. 35). Let assumptions (A1) – (A5) hold. Then the problem (3.1.15)-(3.1.19) has a weak solution $(\psi, u, v) \in (H^1(U) \cap L^\infty(U))^3$, which satisfies the $L^\infty$ estimates,

$$e^{-K} \leq u \leq e^K \text{ a.e. in } U,$$

$$e^{-K} \leq v \leq e^K \text{ a.e. in } U.$$

Proof of this theorem has two components. The system is decoupled by an iterative procedure, resulting in a fixed point analysis for the so-called Gummel Map while existence of solutions is demonstrated for each of the component equations independently.

For analysis of the component equations, the reader is refered to [64], Sec. 3.2. We note that under physically admittable data it is well known that solutions to the system are not unique. Under more severe restrictions on data solutions can be shown to have $H^2$ regularity [64] p. Sec. 3.3.

One of the difficulties with the drift-diffusion system is its coupled nature. The Gummel Map is thus important for both analysis and computation [49, 64], and is used in the simulations in this paper. Let $\mathcal{H} := H^1(U) \cap L^\infty(U)$ with norm inherited from $H^1$. The Gummel Map, $G : \mathcal{H}^2 \to \mathcal{H}^2$, is applied iteratively until a fixed point $(u^*, v^*)$ is reached. The map proceeds by: given $(u^0, v^0) \in \mathcal{H}^2$, for each $k \geq 0$, 


1. solve (3.1.15) with data \((u^k, v^k)\),

\[-\nabla \cdot (\epsilon \nabla \psi^{k+1}) = q(\psi^{k+1}, u^k, v^k) \quad \text{on } U,\]

\[\psi^{k+1} = \psi_D \quad \text{on } \partial U_D,\]

\[\frac{\partial \psi^{k+1}}{\partial \nu} = 0 \quad \text{on } \partial U_N\]

for \(\psi^{k+1} \in \mathcal{H}\).

2. Then solve (3.1.16) with data \((\psi^{k+1}, u^k, v^k)\) iteration lagging the right hand side semilinearity,

\[-\nabla \cdot (D_n e^{\psi^{k+1}} \nabla u^{k+1}) = R(\psi^{k+1}, u^k, v^k) \quad \text{on } U,\]

\[u^{k+1} = u_D \quad \text{on } \partial U_D,\]

\[\frac{\partial u^{k+1}}{\partial \nu} = 0 \quad \text{on } \partial U_N\]

for \(u^{k+1} \in \mathcal{H}\) and solve (3.1.17) similarly iteration lagging \(v^k\) on the right hand side,

\[-\nabla \cdot (D_p e^{-\psi^{k+1}} \nabla v^{k+1}) = R(\psi^{k+1}, u^k, v^k) \quad \text{on } U,\]

\[v^{k+1} = v_D \quad \text{on } \partial U_D,\]

\[\frac{\partial v^{k+1}}{\partial \nu} = 0 \quad \text{on } \partial U_N\]

for \(v^{k+1} \in \mathcal{H}\).

A fixed point corresponds to a weak solution \((\psi^*, u^*, v^*) \in \mathcal{H}^3\) of the drift diffusion system.

When the Gummel Map can be shown to be a contraction, the Banach Contraction Mapping Principle 2.1.24 is applied, and uniqueness of the solution is guaranteed. However, under some data the Gummel Map fails to be contraction, and the Schauder Fixed Point Theorem 2.1.25 is applied, guaranteeing existence, but not uniqueness of the solution. In our case, \(X = \mathcal{H}^2\) and \(K \subset X\) is a subset satisfying bounds on \(u\) and \(v\) guaranteed by the analysis of the component equations of the drift-diffusion system. Thus a fixed point
is guaranteed by showing that $G$ is continuous and maps a non-empty, convex, compact subset $K$ into itself where $K$ is determined by bounds on solutions to the component equations.
3.2. Domain Decomposition

Domain decomposition methods provide a framework for solving a PDE independently on multiple subdomains. Originally these methods were developed in the interest of parallelization for PDE models with smooth solutions that were too large to be solved on a single processor. More recently, domain decomposition has developed as a technique for solving problems with discontinuous coefficients or interfaces separating different materials or physical models [17, 19, 27, 79, 80, 89, 114, 117].

In this section we first review the basic concepts in domain decomposition methods for partial differential equations, focusing on the linear Poisson equation, following [90]. We then present two algorithms designed for models with non-standard transmission conditions, such as the model for charge transport in semiconductor heterostructures, and perform convergence analysis on the methods. Finally, we use the domain decomposition setup as a framework for the analysis of the component equations in the heterostructure model.

3.2.1. Mathematical Background

While the linear Poisson equation has no jumps or discontinuities, our methods can be seen as extensions of a classical method, the Neumann-Neumann method, which is developed for the linear Poisson equation. Thus the simple model here serves as a template for the development of more advanced domain decomposition techniques capable of handling multiphysics and multiscale phenomena.
3.2.1.1 Continuous Formulation

Recall the linear Poisson equation posed on the whole domain $\Omega$, with homogeneous Dirichlet boundary conditions,

$$-\nabla \cdot (\nabla \phi) = f, \quad x \in \Omega,$$

$$\phi = 0, \quad x \in \partial \Omega.$$  \hfill (3.2.1)

We divide the domain $\Omega$ into two non-overlapping subdomains $\Omega_i$, $i = 1, 2$ and let $\Gamma$ denote the interface between the two domains. Figure 3.2 illustrates this setup in $\mathbb{R}^2$.

Next we write (3.2.1)-(3.2.2) in a multidomain formulation. For $i = 1, 2$, we have,

$$-\nabla \cdot (\nabla \phi_i) = f, \quad x \in \Omega_i,$$

$$\phi_i = 0, \quad x \in \partial \Omega_i \cap \partial \Omega.$$  \hfill (3.2.3)

(3.2.4)

together with the transmission conditions,

$$\phi_1 = \phi_2, \quad x \in \Gamma,$$

$$\frac{\partial \phi_1}{\partial \nu} = \frac{\partial \phi_2}{\partial \nu}, \quad x \in \Gamma.$$  \hfill (3.2.5)

(3.2.6)

The goal in the domain decomposition setting is to solve the PDE independently on the two subdomains $\Omega_1$ and $\Omega_2$. This would be trivial if instead of the transmission conditions (3.2.5)-(3.2.6), we had boundary conditions for the variables $\phi_1$ and $\phi_2$. In a nut-shell,
the domain decomposition techniques we will consider, known as ‘iterative substructuring’ methods, are iterative techniques to establish boundary conditions from the transmission conditions (3.2.5)-(3.2.6).

Consider the independent subdomain problems,

\[-\nabla \cdot (\nabla w_i) = f, \quad x \in \Omega_i, \quad (3.2.7)\]

\[w_i = 0, \quad x \in \partial \Omega_i \cap \partial \Omega, \quad (3.2.8)\]

\[w_i = \lambda, \quad x \in \Gamma. \quad (3.2.9)\]

It is clear that for any data \(\lambda\), the condition (3.2.5) is satisfied. However there is no reason to expect that (3.2.6) will hold for any particular \(\lambda\). But, if \(\lambda = \phi_1|\Gamma = \phi_2|\Gamma\), where \(\phi_i\) solves (3.2.1)-(3.2.2), then (3.2.6) will hold. Thus in this setting \(\lambda\) is not actually data, but rather is an additional unknown for the two-domain problem, and the model requires an additional equation to avoid being under-determined. This additional equation is called the ‘Steklov-Poincaré’ interface equation, and its iterative solution is the crux of iterative substructuring methods. Next we develop the equation for the linear Poisson problem.

Write \(w_i = J_i \lambda + P_i f\), where \(J_i \lambda\) denotes the harmonic extension of \(\lambda\) into subdomain \(\Omega_i\) and \(P_i f\) describes the response to the force. \(J_i \lambda\) is defined as the solution to \n
\[-\nabla \cdot (\nabla v_i) = 0, \quad x \in \Omega_i, \quad v_i = \lambda, \quad x \in \Gamma, \quad v_i = 0, \quad x \in \partial \Omega_i \cap \partial \Omega, \quad (3.2.10)\]

while \(P_i f\) is defined as the solution to \n
\[-\nabla \cdot (\nabla v_i) = f, \quad x \in \Omega_i, \quad v_i = 0, \quad x \in \Gamma, \quad v_i = 0, \quad x \in \partial \Omega_i \cap \partial \Omega, \quad (3.2.11)\]

We then define the operators \(S \lambda\) and \(\chi\) by,

\[S \lambda := \sum_{i=1}^{2} \frac{\partial J_i \lambda}{\partial \nu_i}, \quad (3.2.12)\]

and

\[\chi := -\sum_{i=1}^{2} \frac{\partial P_i f}{\partial \nu_i}. \quad (3.2.13)\]
We then state the Steklov-Poincaré interface equation,

\[ S\lambda = \chi, \quad \text{on } \Gamma. \quad (3.2.14) \]

If \( \lambda \) satisfies (3.2.14), then \( w_i = \phi_i \) for \( i = 1, 2 \).

### 3.2.1.2 Neumann-Neumann Algorithm

While there are several well-established iterative substructuring algorithms, the Neumann-Neumann algorithm is perhaps the most well-known. Additionally, the methods developed in [30, 22, 23], and in this work can be viewed as extensions of the Neumann-Neumann algorithm. Thus we review the Neumann-Neumann algorithm here.

#### Neumann-Neumann Algorithm

Given \( \lambda^0 \), for each \( k \geq 0 \),

1. For \( i = 1, 2 \) solve the subdomain problems

   \[- \Delta \phi_i^{k+1} = f, \quad x \in \Omega_i, \quad (3.2.15)\]

   \[ \phi_i^{k+1} = \lambda, \quad \text{on } \Gamma, \quad (3.2.16)\]

   \[ \phi_i^{k+1} = 0, \quad \text{on } \partial \Omega_i \cap \partial \Omega, \quad (3.2.17)\]

   for \( \phi_1^{k+1} \) and \( \phi_2^{k+1} \).

2. Then solve in subdomains \( i = 1, 2 \),

   \[- \Delta v_i^{k+1} = 0, \quad x \in \Omega_i \quad (3.2.18)\]

   \[ v_i^{k+1} = 0 \quad \text{on } \partial \Omega \cap \partial \Omega_i \quad (3.2.19)\]

   \[ \frac{\partial v_i^{k+1}}{\partial \nu} = \left[ \frac{\partial \phi}{\partial \nu} \right]_{\Gamma} \quad \text{on } \Gamma \quad (3.2.20)\]

3. Update \( \lambda \) by

   \[ \lambda^{k+1} = \lambda^k - \theta \left[ v_i^{k+1} \right]_{\Gamma}, \quad (3.2.21)\]
4. Continue with (1) unless stopping criteria $\left\| \left[ \frac{\partial \phi^{k+1}}{\partial \nu} \right]_{\Gamma} \right\|$ holds.

As it is shown in [90] the Neumann-Neumann algorithm can be analyzed as a preconditioned Richardson scheme for the solution of (3.2.14), and there exists $\theta_{\text{max}} > 0$ s.t. if $\theta \in (0, \theta_{\text{max}})$, the Neumann-Neumann algorithm converges to a solution of (3.2.1)-(3.2.2).

3.2.2. DDM for Potential Equation

In this section we extend the techniques introduced previously for the Poisson equation to the Potential equation in the heterostructure model with homogeneous Dirichlet boundary conditions. Our method, called algorithm DDP (Domain Decomposition for Potential equation), is based on the Neumann-Neumann method but will allow for the non-homogeneous jump (3.1.9). Extending our previous work [30, 22, 23], we allow for a non-homogeneous jump in the flux (3.1.10). Thus we consider (3.1.6) with (3.1.9) and

$$\left[ \epsilon \frac{\partial \psi}{\partial \nu} \right]_{\Gamma} = \psi_\theta. \tag{3.2.22}$$

3.2.2.1 Domain Decomposition Formulation

We consider the following independent subdomain problems,

$$- \nabla \cdot (\epsilon_1 \nabla \psi_1) = q_1, \ x \in \Omega_1, \ \psi_1|_{\Gamma} = \lambda, \ \psi_1|_{\partial \Omega_1 \cap \partial \Omega} = 0. \tag{3.2.23}$$

$$- \nabla \cdot (\epsilon_2 \nabla \psi_2) = q_2, \ x \in \Omega_2, \ \psi_2|_{\Gamma} = \lambda + \psi_\Delta, \ \psi_2|_{\partial \Omega_2 \cap \partial \Omega} = 0. \tag{3.2.24}$$

We drop the dependence on the carrier transport variables $u, v$ in the force $q$ since due to the Gummel decoupling we are able to consider the potential equation independently from the two continuity equations for carrier transport, thus allowing the development of iterative substructuring methods for each equation type independently.

In (3.2.23)-(3.2.24), it is clear that for any $\lambda$, (3.1.9) will be satisfied since

$$[\psi]|_{\Gamma} = \psi_2^\Gamma - \psi_1^\Gamma = \psi_1|_{\Gamma} - \psi_2|_{\Gamma} = \lambda - \lambda - \psi_\Delta = \psi_\Delta. \tag{3.2.25}$$
However, there is no reason to expect that for any particular \( \lambda \) (3.2.22) will hold. Thus, algorithm DDP will be designed to determine, iteratively, the appropriate \( \lambda \) which will ensure (3.2.22) holds.

Recalling the definitions of \( J_1 \) (3.2.10) and \( P_1 \) (3.2.11), we write

\[
\psi_1 = J_1 \lambda + P_1 q, \quad \psi_2 = J_2 \lambda + J_2 \psi_\Delta + P_2 q. \tag{3.2.26}
\]

For \( \psi_1 \) this is the standard separation of the solution to (3.2.23) into components responding to the boundary data \( \lambda \) and the forcing term \( q_1 \). For \( \psi_2 \) we are additionally separating the response to the data \( \lambda + \psi_\Delta \) into components responding to \( \lambda \) and to \( \psi_\Delta \). We then define \( K \) and \( \Upsilon \),

\[
K \lambda := \epsilon_1^\Gamma \frac{\partial J_1 \lambda}{\partial \nu} - \epsilon_2^\Gamma \frac{\partial J_2 \lambda}{\partial \nu}, \tag{3.2.27}
\]

\[
\Upsilon := \epsilon_2^\Gamma \frac{\partial P_2 q}{\partial \nu} - \epsilon_1^\Gamma \frac{\partial P_1 q}{\partial \nu} + \epsilon_2^\Gamma \frac{\partial J_2 \psi_\Delta}{\partial \nu} - \psi_\beta. \tag{3.2.28}
\]

The Steklov-Poincaré interface equation associated to the Poisson equation in the heterostructure model (3.1.6), (3.1.9)-(3.1.10) is then

\[
K \lambda = \Upsilon. \tag{3.2.29}
\]

**Lemma 3.2.1.** Suppose \( \lambda \) satisfies the equation \( K \lambda = \Upsilon \). Then a solution of (3.2.23)-(3.2.24) also solves (3.1.6) with (3.1.9) and (3.2.22) and homogeneous Dirichlet boundary conditions.

**Proof.** The proof of this lemma is a straightforward calculation. Assume \( \lambda \) satisfies (3.2.29) and assume \( \psi \) satisfies (3.2.23)-(3.2.24) with this \( \lambda \). Clearly (3.1.6) is satisfied for \( i = 1, 2 \). Further, as previously noted, (3.1.9) is satisfied by construction. Thus it remains to check (3.2.22). We begin with (3.2.29), and inserting definitions for \( K \) and \( \Upsilon \), have

\[
\epsilon_1^\Gamma \frac{\partial J_1 \lambda}{\partial \nu} - \epsilon_2^\Gamma \frac{\partial J_2 \lambda}{\partial \nu} = \epsilon_2^\Gamma \frac{\partial P_2 q}{\partial \nu} - \epsilon_1^\Gamma \frac{\partial P_1 q}{\partial \nu} + \epsilon_2^\Gamma \frac{\partial J_2 \psi_\Delta}{\partial \nu} - \psi_\beta. \tag{3.2.30}
\]
Next we group terms by subdomain,
\[
\epsilon_1 \left( \frac{\partial J_1 \lambda}{\partial \nu} + \frac{\partial P_1 q}{\partial \nu} \right) - \epsilon_2 \left( \frac{\partial J_2 \lambda}{\partial \nu} + \frac{\partial J_2 \psi \Delta}{\partial \nu} + \frac{\partial P_2 q}{\partial \nu} \right) = -\psi \vartheta, \quad (3.2.31)
\]
and so,
\[
\epsilon_1 \frac{\partial (J_1 \lambda + P_1 q)}{\partial \nu} - \epsilon_2 \frac{\partial (J_2 \lambda + J_2 \psi \Delta + P_2 q)}{\partial \nu} = -\psi \vartheta. \quad (3.2.32)
\]
Recalling (3.2.26) we have
\[
\epsilon_1 \frac{\partial \psi_1}{\partial \nu} - \epsilon_2 \frac{\partial \psi_2}{\partial \nu} = -\psi \vartheta, \quad (3.2.33)
\]
and so we conclude
\[
\left[ \frac{\partial \psi}{\partial \nu} \right]_{\Gamma} = \psi \vartheta. \quad (3.2.34)
\]

### 3.2.2.2 Algorithm DDP

To find an appropriate $\lambda$ and solve (3.2.23)-(3.2.24) with (3.2.29) we present algorithm DDP.

![Algorithm DDP to solve (3.1.6) with (3.1.9) and (3.2.22)]

Given $\lambda^0$, for each $k \geq 0$,

1. Solve (3.2.23) for $\psi^{k+1}_1$ and (3.2.24) for $\psi^{k+1}_2$.

2. Solve the following problem in subdomains $i = 1, 2$,

\[
- \nabla \cdot (\epsilon_i \nabla \phi^{k+1}_i) = 0, \quad x \in \Omega_i \quad (3.2.35)
\]
\[
\phi^{k+1}_i = 0 \text{ on } \partial \Omega \cap \partial \Omega_i \quad (3.2.36)
\]
\[
\epsilon_i \frac{\partial \phi^{k+1}_i}{\partial \nu} = \left[ \epsilon_i \frac{\partial \psi}{\partial \nu} \right]_{\Gamma} - \psi \vartheta \text{ on } \Gamma \quad (3.2.37)
\]
3. Update $\lambda$ by

$$\lambda^{k+1} = \lambda^k - \theta [\phi]_\Gamma$$  \hspace{1cm} (3.2.38)

4. Continue with (1) unless stopping criterium $\| \frac{\partial \psi}{\partial \nu} \|_\Gamma - \psi \theta \|$ holds.

### 3.2.2.3 Convergence Analysis

To show that algorithm DDP converges we will analyze the weak form of (3.2.29). Step 3 of the algorithm will be analyzed as a Richardson iteration preconditioned by steps 1 and 2. This will be shown to be a contraction on a complete matrix space and the Banach Contraction Mapping Principle 2.1.24 will be applied to guarantee convergence.

Let $\gamma_i : H^1(\Omega_i) \to H^\frac{1}{2}(\partial \Omega_i)$ be the trace operator, and

$$V_i := H^1(\Omega_i),$$

$$V_i^0 := \{ v \in V_i \mid \gamma_i v|_{\partial \Omega_i \cap \partial \Omega} = 0 \},$$

$$V := \{ v \in L^2(\Omega) \mid v|_{\Omega_i} \in V_i^0 \text{ for } i = 1, 2 \},$$

$$\Lambda := H^\frac{1}{2}(\Gamma),$$

$$(u, v)_{\Omega_i} := \int_{\Omega_i} uv,$$

$$a_i(u, v) := \int_{\Omega_i} \epsilon_i \nabla u \cdot \nabla v.$$  \hspace{1cm} (3.2.44)

Additionally, let $R_i$ be any continuous extension operator from $\Lambda$ into $V_i^0$.

First we characterize $K$ as an operator from $\Lambda$ to $\Lambda'$, where $\Lambda'$ denotes the space of continuous linear functionals on $\Lambda$. Let $\mu \in \Lambda$, multiply by $\mu$, and note that both $\epsilon_i J_i \lambda$ is
divergence free in $\Omega$, and $R_i \mu|_{\partial \Omega_i \cap \partial \Omega} = 0$ for any $\mu \in \Lambda$.

$$
\langle K\lambda, \mu \rangle = \int_\Gamma \left( \epsilon_1 \frac{\partial J_1 \lambda}{\partial \nu^1} + \epsilon_2 \frac{\partial J_2 \lambda}{\partial \nu^2} \right) \mu
$$

$$
= \int_\Gamma \epsilon_1 \frac{\partial J_1 \lambda}{\partial \nu^1} R_1 \mu + \epsilon_2 \frac{\partial J_2 \lambda}{\partial \nu^2} R_2 \mu
$$

$$
= \int_{\partial \Omega_1} \epsilon_1 \frac{\partial J_1 \lambda}{\partial \nu^1} R_1 \mu + \int_{\partial \Omega_2} \epsilon_2 \frac{\partial J_1 \lambda}{\partial \nu^2} R_2 \mu
$$

$$
= \int_{\Omega_1} \nabla J_1 \lambda \cdot \nabla R_1 \mu + \int_{\Omega_2} \nabla J_2 \lambda \cdot \nabla R_2 \mu
$$

$$
= a_1(J_1 \lambda, R_1 \mu) + a_2(J_2 \lambda, R_2 \mu)
$$

Since $R_i$ can be any possible extension operator, we take $R_i = J_i$ in this case, and have

$$
\langle K\lambda, \mu \rangle = \sum_{i=1}^{2} a_i(J_i \lambda, J_i \mu).
$$

The fact that for each $\lambda \in \Lambda$ $\langle K\lambda, \mu \rangle$ is a linear function of $\mu$ follows from the linearity of integration. In the proof of the upcoming Theorem 3.2.3 continuity will be shown. Thus for each $\lambda \in \Lambda$, $\langle K\lambda, \cdot \rangle \in \mathcal{L}$.

Next we proceed similarly for the right hand side of (3.2.29), recalling the definition of $P_i$ (3.2.11),

$$
\langle \Upsilon, \mu \rangle = - \int_\Gamma \epsilon_2 \frac{\partial P_2 q}{\partial \nu^2} \mu - \int_\Gamma \epsilon_1 \frac{\partial P_1 q}{\partial \nu^2} \mu - \int_\Gamma \epsilon_1 \frac{\partial J_2 \psi_\Delta}{\partial \nu^2} \mu - \int_\Gamma \psi \partial \mu
$$

$$
= - \int_\Gamma \epsilon_2 \frac{\partial P_2 \psi}{\partial \nu^2} R_2 \mu - \int_\Gamma \epsilon_1 \frac{\partial P_1 \psi}{\partial \nu^2} R_1 \mu - \int_\Gamma \epsilon_1 \frac{\partial J_2 \psi_\Delta}{\partial \nu^2} R_2 \mu - \int_\Gamma \psi \partial \mu
$$

$$
= - \int_{\partial \Omega_2} \epsilon_2 \frac{\partial P_2 \mu}{\partial \nu^2} R_2 \mu - \int_{\partial \Omega_1} \epsilon_1 \frac{\partial P_1 \psi}{\partial \nu^2} R_1 \mu - \int_{\partial \Omega_2} \epsilon_1 \frac{\partial J_2 \psi_\Delta}{\partial \nu^2} R_2 \mu - \int_\Gamma \psi \partial \mu
$$

$$
= \nabla \cdot (\epsilon_2 \nabla P_2 q) R_2 \mu - \int_{\Omega_2} \epsilon_2 \nabla P_2 \mu \cdot \nabla R_2 \mu
$$

$$
+ \int_{\Omega_1} \nabla \cdot (\epsilon_1 \nabla P_2 q) R_1 \mu - \int_{\Omega_1} \epsilon_1 \nabla P_1 q \cdot \nabla R_1 \mu - \int_{\Omega_2} \epsilon_2 \nabla J_2 \psi_\Delta \cdot \nabla R_2 \mu - \int_\Gamma \psi \partial \mu
$$

$$
= (q_1, R_1 \mu)_{\Omega_i} - a_1(P_1 q, R_1 \mu) - (q_2, R_2 \mu)_{\Omega_2} - a_2(P_2 q, R_2 \mu) - a_2(J_2 \psi_\Delta, R_2 \mu) - \langle \psi \partial, \mu \rangle.
$$

And so,

$$
\langle \Upsilon, \mu \rangle = \sum_{i=1}^{2} [(q_i, R_i \mu)_{\Omega_i} - a_1(P_1 q, R_i \mu)] - a_2(J_2 \psi_\Delta, R_2 \mu) - \langle \psi \partial, \mu \rangle.
$$

(3.2.46)
Linearity of $\langle \Upsilon, \cdot \rangle$ follows from the linearity of integration. Continuity of the $a_i(\cdot, \cdot)$ terms is demonstrated in the proof of Theorem 3.2.3. Continuity of the $\langle \psi_{\theta}, \cdot \rangle$ term is a consequence of the Cauchy-Schwarz Inequality. The continuity of the $(q_i, R_i \mu)_{\Omega_i}$ terms remain to be demonstrated. Let $\mu \in \Lambda$, 

$$
|\langle q_i, R_i \mu \rangle_{\Omega_i}| = \left| \int_{\Omega_i} q_i R_i \mu \right| 
\leq \|q_i\|_{L^2(\Omega_i)} \|R_i \mu\|_{L^2(\Omega_i)} 
\leq \|q_i\|_{L^2(\Omega_i)} \|R_i \mu\|_{H^1(\Omega_i)},
$$

(3.2.47) 

(3.2.48) 

(3.2.49) 
as long as $q_i \in L^2(\Omega_i)$. Then since $R_i : \Lambda \rightarrow V_i^0$ is continuous, there exists $C > 0$ such that

$$
|\langle q_i, R_i \mu \rangle| \leq C \|q_i\|_{L^2(\Omega_i)} \|\mu\|_{\Lambda}.
$$

(3.2.50) 

So, $\langle \Upsilon, \cdot \rangle \in \Lambda'$. 

With (3.2.45) and (3.2.46) we have the weak form of (3.2.29)

$$
\text{find } \lambda \in \Lambda : \langle K \lambda, \mu \rangle = \langle \Upsilon, \mu \rangle \forall \mu \in \Lambda.
$$

(3.2.51) 

To prove the convergence of algorithm DDP we use the following lemma, contained in a theorem from [90]. 

**Lemma 3.2.2** ([90] p. 120). Let $X$ be a Hilbert space and $A : X \rightarrow X'$ be an operator. Suppose $A$ can be split into $A = A_1 + A_2$, and that each $A_i$ is continuous and coercive with continuity constant $C_i$ and coercivity constant $\alpha_i$, respectively. That is,

$$
|\langle A_i x, y \rangle| \leq C_i \|x\|_X \|y\|_X \forall x, y \in X, \text{ and}
$$

$$
\langle A_i x, x \rangle \geq \alpha_i \|x\|_X^2 \forall x \in X.
$$

(3.2.52) 

(3.2.53) 

Define $N = (A_2^{-1} + A_1^{-1})^{-1}$. Note that $N$ is continuous and coercive since $A_i$ is continuous and coercive. Let $\alpha_N$ denote the coercivity constant for $N$ and let $C_N$ denote the continuity
constant for $N$. Further assume $N$ satisfies the condition that there exists $k^* > 0$ such that

$$\langle N\lambda, N^{-1}A\lambda \rangle + \langle A\lambda, \lambda \rangle \geq k^*\|\lambda\|_X^2 \quad \forall \lambda \in X. \quad (3.2.54)$$

Define $\theta_{\text{max}}$ by

$$\theta_{\text{max}} = \frac{k^*\alpha_N}{C_N \left( \frac{1}{\alpha_1} + \frac{1}{\alpha_2} \right) (C_1 + C_2)^2} \quad (3.2.55)$$

Then for any $\theta \in (0, \theta_{\text{max}})$ the operator $T_{\theta} = I - \theta N^{-1}A$ is a contraction on the space $X$.

We also note that if $A_i$ are symmetric, condition (3.2.54) is equivalent to the coercivity of $A$ [[90] p. 121].

**Theorem 3.2.3 (Convergence of Algorithm DDP).** There exists $\theta^\psi_{\text{max}} > 0$ such that for any $0 < \theta < \theta^\psi_{\text{max}}$, and for any initial guess $\lambda^0$, algorithm DDP converges to the solution $\lambda$ of (3.2.29).

**Proof.** A proof of this theorem takes the following steps:

1. First we define an operator splitting for $K$ and show that algorithm DDP is equivalent to the following preconditioned Richardson scheme for (3.2.29),

$$\left( K_1^{-1} + K_2^{-1} \right)^{-1} (\lambda^{k+1} - \lambda^k) = \theta (\Upsilon - K\lambda^k). \quad (3.2.56)$$

2. Next we show that

$$N_p := (K_1^{-1} + K_2^{-1})^{-1} \quad (3.2.57)$$

satisfies the criteria from Lemma 3.2.2 on the Hilbert space $\Lambda$.

3. Finally we conclude by application of the Banach Contraction Mapping Principle 2.1.24 and Lemma 3.2.2 that the algorithm converges.
3.2.2.4 Step 1

We define the operator splitting of \( K \) by
\[
K_1 \lambda := \frac{\partial J_1 \lambda}{\partial \nu},
\]
\[
K_2 \lambda := -\frac{\partial J_2 \lambda}{\partial \nu}.
\]
(3.2.58)
(3.2.59)

Recall that \( \psi_1^{k+1} = J_1 \lambda^k + P_1 q \) and \( \psi_2^{k+1} = J_2 \lambda^k + J_2 \psi_\triangle + P_2 q \), therefore,
\[
\phi_1^{k+1}|_\Gamma = K_1^{-1} \left( \frac{\partial J_1 \lambda}{\partial \nu} + \frac{\partial P_1 q}{\partial \nu} - \frac{\partial J_2 \lambda^k}{\partial \nu} + \frac{\partial J_2 \psi_\triangle}{\partial \nu} - \frac{\partial P_2 q}{\partial \nu} - \psi_\theta \right)
\]
(3.2.60)
\[
= -K_1^{-1}(\Upsilon - K \lambda^k).
\]
(3.2.61)

Similarly,
\[
\phi_2^{k+1}|_\Gamma = K_2^{-1}(\Upsilon - K \lambda^k).
\]
(3.2.62)

And so we have,
\[
\lambda^{k+1} = \lambda^k + \theta(K_1^{-1} + K_2^{-1})(\Upsilon - K \lambda^k).
\]
(3.2.63)

Or, equivalently,
\[
N_p(\lambda^{k+1} - \lambda^k) = \theta(\Upsilon - K \lambda^k).
\]
(3.2.64)

3.2.2.5 Step 2

To show that \( N_p \) satisfies the criteria of Lemma 3.2.2 we must show that \( K_i \) is continuous and coercive on \( \Lambda \) for each \( i = 1, 2 \) and that \( N_p \) satisfies condition (3.2.54).

In the weak setting we have
\[
\langle K_1 \lambda, \mu \rangle = a_1(J_1 \lambda, J_1 \mu),
\]
(3.2.65)
\[
\langle K_2 \lambda, \mu \rangle = a_2(J_2 \lambda, J_2 \mu).
\]
(3.2.66)
Then to check coercivity,

\[ \langle K_1 \lambda, \lambda \rangle \geq \epsilon_1^0 \| \nabla J_1 \lambda \|_{L^2(\Omega)}^2 \]  \hspace{1cm} (3.2.67)

\[ \geq \epsilon_1^0 P_1^2 \| J_1 \lambda \|_{H^1(\Omega)} \]  \hspace{1cm} (3.2.68)

\[ \geq \epsilon_1^0 P_1^2 C_2 \| J_1 \lambda \|_{\Lambda}^2 \]  \hspace{1cm} (3.2.69)

where \( P_1 \) is the constant guaranteed by the Poincaré Inequality 2.1.9 and \( C_1 \) is the constant guaranteed by the Trace Inequality 2.1.8. The calculation for \( K_2 \) follows, thus \( K_i \) is coercive with constant \( \alpha_i = \epsilon_i^0 P_i^2 C_i^2 \).

To check continuity,

\[ \langle K_i \lambda, \mu \rangle \leq \epsilon_i^1 \| J_i \lambda \|_{H^1(\Omega)} \| J_i \mu \|_{H^1(\Omega)}. \]  \hspace{1cm} (3.2.70)

From well known estimates for elliptic boundary value problems [90], p. 9 we know there exists \( \beta_i > 0 \) such that

\[ \| J_i \lambda \|_{H^1(\Omega)} \leq \beta_i \| \lambda \|_{\Lambda}. \]  \hspace{1cm} (3.2.71)

Thus \( K_i \) is continuous with constant \( C_i = \epsilon_i^1 \beta_i \).

Now we simply note that the \( K_i \) are clearly symmetric, and thus condition (3.2.54) is equivalent to the coercivity of \( K \), which is a direct consequence of the coercivity of \( K_i, i = 1, 2 \). Thus condition (3.2.54) is satisfied, and the conditions of Lemma 3.2.2 are satisfied.

### 3.2.2.6 Step 3

Let \( \varphi_{max}^\psi \) be the constant guaranteed by Lemma 3.2.2. Let \( 0 < \theta < \theta_{max}^\psi \). From Lemma 3.2.2 we know that the operator

\[ T_\theta^\psi := I - \theta N_p^{-1} K \]  \hspace{1cm} (3.2.72)

is a contraction on \( \Lambda \). Notice,

\[ N_p(\lambda^{k+1} - \lambda^k) = \theta(\Upsilon - K \lambda^k), \]  \hspace{1cm} (3.2.73)
then solving for $\lambda^{k+1}$

$$\lambda^{k+1} = \lambda^k - \theta N_p^{-1}(K\lambda^k - \Upsilon),$$

(3.2.74)

and applying (3.2.72),

$$\lambda^{k+1} = T_{\theta}^\psi + \theta N_p^{-1}\Upsilon.$$  

(3.2.75)

Then since $\theta N_p^{-1}\Upsilon$ does not depend on $\lambda^k$, $T_{\theta}^\psi$ a contraction implies that the operator $G_{\theta}$, defined by

$$G_{\theta}^\psi \lambda = T_{\theta}^\psi \lambda + \theta N_p^{-1}\Upsilon$$

(3.2.76)

is a contraction on the Hilbert space $\Lambda$.

Now applying the Banach Contraction Mapping Principle 2.1.24 we know that algorithm DDP converges to a fixed point $\lambda_0$, of the map $G_{\theta}^\psi$. Then we have

$$G_{\theta}^\psi \lambda_0 = \lambda_0,$$

(3.2.77)

then recalling (3.2.76) and (3.2.72),

$$\lambda_0 - \theta N_p^{-1}(K\lambda_0 - \Upsilon) = \lambda_0.$$  

(3.2.78)

Next we isolate the two free $\lambda^0$ terms,

$$\lambda_0 - \lambda_0 = \theta N_p^{-1}(K\lambda_0 - \Upsilon),$$

(3.2.79)

and so,

$$0 = K\lambda_0 - \Upsilon.$$  

(3.2.80)
3.2.2.7 Relaxation Parameter Analysis

Next we consider the maximum relaxation parameter $\theta_{\psi max}$. This requires estimates for the continuity and coercivity constants for the operator $N$ ($C_N$ and $\alpha_N$, respectively) as well as those for the operators $K_i$, $i = 1, 2$ ($C_i$ and $\alpha_i$, respectively).

We already saw in the proof of Theorem 3.2.3 that

$$\alpha_i = \epsilon_i^0 P_i^2 C_i^2,$$  \hspace{1cm} (3.2.81)
$$C_i = \epsilon_i^1 \beta_i^2.$$  \hspace{1cm} (3.2.82)

Next, since $K_i$ is symmetric for $i = 1, 2$ we have $k^* = 2(\alpha_1 + \alpha_2)$. In Ref. [90], p. 108 it is shown that for the Neumann-Neumann type preconditioner $N$, we have

$$\alpha_N = \frac{(\alpha_1 C_2^2 + \alpha_2 C_1^2)\alpha_1^2 \alpha_2^2}{C_1^2 C_2^2 (\alpha_1 + \alpha_2)^2},$$  \hspace{1cm} (3.2.83)
$$C_N = \frac{C_1^2 C_2^2}{\alpha C_1^2 + \alpha_2 C_2^2}.$$  \hspace{1cm} (3.2.84)

Thus, $\theta_{\psi max}$ can be computed in terms of $P_i$, $C_i$, $\beta_i$ and $\epsilon$. In Section 3.4. we will see for a simple problem that this analytical estimate does not perform well; i.e. that the resulting $\theta_{\psi max}$, while correct in the sense that for $\theta \in (0, \theta_{\psi max})$ algorithm DDP converges, is too small and the resulting method converges too slowly. Thus we will examine a computational technique for obtaining a much better result.

3.2.3. DDM for Carrier Transport

In this section we extend the techniques introduced for the Poisson equation to the equations for carrier transport in the heterostructure model, again with homogeneous Dirichlet boundary conditions. Our method, called algorithm DDC (Domain Decomposition for Carrier Transport), is based on the Neumann-Neumann method but will allow for the unusual Robin-like transmission conditions (3.1.11)-(3.1.12) for electron transport, or (3.1.13)-(3.1.14) for hole transport.
Rather than separately considering the equations for electron and hole transport, we recognize their similar structure and develop the algorithm for the following model problem,

\[-\nabla \cdot (k_i \nabla u_i) = f_i, \quad x \in \Omega_i, \quad (3.2.85)\]

\[k_1^\Gamma \frac{\partial u_1}{\partial \nu} = a_2 u_2^\Gamma - a_1 u_1^\Gamma, \quad (3.2.86)\]

\[k_1^\Gamma \frac{\partial u_1}{\partial \nu} = k_2^\Gamma \frac{\partial u_2}{\partial \nu}, \quad (3.2.87)\]

\[u_i|_{\partial \Omega_i \cap \partial \Omega} = 0. \quad (3.2.88)\]

We assume \(k\) is data such that \(0 < k_{\text{min}} < k < k_{\text{max}}\) on \(\Omega\) for some constants \(k_{\text{min}}\) and \(k_{\text{max}}\), \(f \in L^2(\Omega_i)\) for \(i = 1, 2\), and that the \(a_i\) are positive and bounded.

### 3.2.3.1 Domain Decomposition Formulation

We consider the following subdomain problems,

\[-\nabla \cdot (k_1 \nabla u_1) = f_1, \quad x \in \Omega_1, \quad u_1^\Gamma = \lambda, \quad u_1|_{\partial \Omega_1 \cap \partial \Omega} = 0, \quad (3.2.89)\]

\[-\nabla \cdot (k_2 \nabla u_2) = f_2, \quad x \in \Omega_2, \quad u_2^\Gamma = \frac{a_1}{a_2} \lambda + \frac{1}{a_2} k_1^\Gamma \frac{\partial u_1}{\partial \nu}, \quad u_2|_{\partial \Omega_2 \cap \partial \Omega} = 0. \quad (3.2.90)\]

In (3.2.89)-(3.2.90), it is clear that for any \(\lambda\), (3.2.86) will be satisfied since

\[u_2^\Gamma = \frac{a_1}{a_2} \lambda + \frac{1}{a_2} k_1^\Gamma \frac{\partial u_1}{\partial \nu} \quad (3.2.91)\]

\[= \frac{a_1}{a_2} u_1^\Gamma + \frac{1}{a_2} k_1^\Gamma \frac{\partial u_1}{\partial \nu}, \quad (3.2.92)\]

and so

\[k_1^\Gamma \frac{\partial u_1}{\partial \nu} = a_2 u_2^\Gamma - a_1 u_1^\Gamma. \quad (3.2.93)\]

However, we have no reason to assume that (3.2.87) is satisfied for any particular \(\lambda\). Thus, similarly to the case of the potential equation, the task is to determine the correct \(\lambda\) that ensure (3.2.87) holds. However, the transmission conditions (3.2.86)-(3.2.87) present challenges not present in the case of the potential equation. In particular, notice that the
"Dirichlet condition" on the interface Γ in (3.2.90) depends on the value of the flux of the solution from (3.2.89). A more natural choice would be to prescribe Neumann conditions for an unknown \( \lambda \). This method, which we call algorithm DDN will be developed towards the end of this section since it may be suitable for other applications. However, as we will show, this seemingly more natural method has tremendous difficulty with the pathological nature of the data in heterostructure problems. Thus it is not a strong candidate for this application, and we must deal with the oddity of the dependence of the interface condition in domain two on the flux of the solution in domain one.

To develop a method for finding a suitable \( \lambda \), we write

\[ u_1 = H_1 \lambda + P_1 f, \quad u_2 = H_2 \lambda + H_2 \left( \frac{1}{a_2} k_1^\Gamma \left( \frac{\partial H_1 \lambda}{\partial \nu} + \frac{\partial P_1 f}{\partial \nu} \right) \right) + P_2 f, \quad (3.2.94) \]

where \( u \) is the solution of (3.2.89)-(3.2.90). Here we recall the definition of \( P_i f \) (3.2.11) and define \( H_i \lambda \) as the solution of the problems,

\[ -\nabla \cdot (k_i \nabla v_i) = 0, \quad x \in \Omega_i, \quad (3.2.95) \]
\[ v_1 = \lambda, \quad x \in \Gamma, \quad (3.2.96) \]
\[ v_2 = \frac{a_1}{a_2} \lambda, \quad x \in \Gamma, \quad (3.2.97) \]
\[ v_i = 0, \quad x \in \partial \Omega_i \cap \partial \Omega. \quad (3.2.98) \]

For \( u_1 \) this is the standard separation of the solution to (3.2.89) into components responding to the boundary data \( \lambda \) and the forcing term \( f_1 \). For \( u_2 \) we additionally separate the response to the data \( \frac{a_1}{a_2} + \frac{1}{a_2} k_1^\Gamma \frac{\partial u_1}{\partial \nu} \) into components responding to \( \frac{a_1}{a_2} \lambda \) and \( \frac{1}{a_2} k_1^\Gamma \frac{\partial u_1}{\partial \nu} \). The multiplicative factor \( \frac{a_1}{a_2} \) is included in the definition of \( H_2 \lambda \) to simplify calculations.

We define \( \Xi \) and \( \Sigma \),

\[ \Xi := k_1^\Gamma \frac{\partial H_1 \lambda}{\partial \nu} - k_2^\Gamma \frac{\partial H_2 \lambda}{\partial \nu} - k_2^\Gamma \frac{\partial H_2}{\partial \nu} \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), \quad (3.2.99) \]
\[ \Sigma := k_2^\Gamma \frac{\partial P_2 f}{\partial \nu} - k_1^\Gamma \frac{\partial P_1 f}{\partial \nu} + k_2^\Gamma \frac{\partial H_2}{\partial \nu} \left( \frac{1}{a_1} \frac{\partial P_1 f}{\partial \nu} \right), \quad (3.2.100) \]
and consider the Steklov-Poincaré equation

\[ \Xi \lambda = \Sigma. \] (3.2.101)

**Lemma 3.2.4.** Suppose \( \lambda \) satisfies the equation \( \Xi \lambda = \Sigma \). Then a solution to (3.2.89)-(3.2.90) also solves (3.2.85)-(3.2.88).

**Proof.** Assume \( \lambda \) satisfies \( \Xi \lambda = \Sigma \). Assume \( u \) satisfies (3.2.89)-(3.2.90) with this \( \lambda \). Clearly

\[
-\nabla \cdot (k_i \nabla u_i) = f_i, \quad x \in \Omega_i.
\] (3.2.102)

Further, (3.2.86) is satisfied by construction. It remains to check the condition (3.2.87).

So, we calculate, beginning from (3.2.101) and recalling (3.2.99)-(3.2.100),

\[
k_1 \frac{\partial H_1 \lambda}{\partial \nu} - k_2 \frac{\partial H_2 \lambda}{\partial \nu} - k_2^\Gamma \frac{\partial H_2}{\partial \nu} \left( \frac{1}{a_i} \frac{\partial H_1 \lambda}{\partial \nu} \right) = k_2 \frac{\partial P_2 f}{\partial \nu} - k_1^\Gamma \frac{\partial P_2 f}{\partial \nu} + k_2^\Gamma \frac{\partial P_2}{\partial \nu} \left( \frac{1}{a_i} \frac{\partial P_1 f}{\partial \nu} \right).
\]

Then we group terms by domain,

\[
k_1^\Gamma \left( \frac{\partial H_1 \lambda}{\partial \nu} + \frac{\partial P_1 f}{\partial \nu} \right) = k_2^\Gamma \left( \frac{\partial H_2}{\partial \nu} \left( \lambda + \frac{1}{a_i} \left( \frac{\partial H_1 \lambda}{\partial \nu} + \frac{\partial P_1 f}{\partial \nu} \right) \right) + \frac{\partial P_2 f}{\partial \nu} \right).
\]

Then recalling (3.2.94) we conclude

\[
k_1^\Gamma \frac{\partial u_1}{\partial \nu} = k_2^\Gamma \frac{\partial u_2}{\partial \nu}.
\]

\[ \square \]

### 3.2.3.2 Algorithm DDC

To find an appropriate \( \lambda \) and solve (3.2.89)-(3.2.90) with (3.2.101) we present algorithm DDC.
Algorithm DDC to solve (3.2.85)-(3.2.88)

Given $\lambda^0$, for each $k \geq 0$,

1. Solve (3.2.89) for $u_1^{k+1}$ and (3.2.90) for $u_2^{k+1}$.

2. Solve the following problem in subdomains $i = 1, 2$,

\[- \nabla \cdot (k \nabla \phi_i^{k+1}) = 0, \quad x \in \Omega_i \tag{3.2.103}\]

\[\phi_i^{k+1} = 0 \text{ on } \partial \Omega \cap \partial \Omega_i \tag{3.2.104}\]

\[k_i \frac{\partial \phi_i^{k+1}}{\partial \nu} = \left[ k_i \frac{\partial u_i}{\partial \nu} \right]_{\Gamma} \text{ on } \Gamma \tag{3.2.105}\]

3. Update $\lambda$ by

\[\lambda^{k+1} = \lambda^k - \theta \left[ \phi_i^{k+1} \right]_{\Gamma} \tag{3.2.106}\]

4. Continue with (1) unless stopping criterium $\| [k \frac{\partial u_i}{\partial \nu}]_{\Gamma} \|$ holds.

3.2.3.3 Convergence Analysis

To show that algorithm DDC converges we will analyze the weak form of (3.2.101). This analysis is very similar to that of algorithm DDC. This is not surprising since both methods are designed as extensions of the Neumann-Neumann method, and thus have similar structures. However, the calculations do differ due to the difference in the definitions of the interface equations. Step 3 of the algorithm will be analyzed as a Richardson iteration preconditioned by steps 1 and 2. This will be shown to be a contraction on a complete metric space and the Banach Contraction Mapping Principle 2.1.24 will be applied to guarantee convergence.

Due to the presence of the normal derivative in the problem on domain 1, we will
require an additional constraint on the space \( V^0_1 \). We define
\[
V^0_{1,\Delta} = \{ v \in H^1(\Omega_1) \mid \gamma_1 v|_{\partial \Omega_1 \cap \partial \Omega} = 0, \Delta v \in H^{-1}(\Omega_1) \}, \tag{3.2.107}
\]
\[
V_\Delta = \{ v \in L^2(\Omega) \mid v_1 \in V^0_{1,\Delta}, v_2 \in V^0_2 \}. \tag{3.2.108}
\]

Let \( R_i \) and \( R^*_i \) denote any possible continuous extension operators from \( \Lambda \) to \( V^0_{1,\Delta} \), \( V^0_2 \). Also define the bilinear forms,
\[
b_i(u, v) := \int_{\Omega_i} k_i \nabla u \cdot \nabla v. \tag{3.2.109}
\]

To characterize \( \Xi : \Lambda \to \Lambda' \) we multiply by \( \mu \in \Lambda \), and note both that \( k_k H_i \lambda \) is divergence free in \( \Omega_i \) and \( R_i \mu|_{\partial \Omega_1 \cap \partial \Omega} = 0 \) for any \( \mu \in \Lambda \),
\[
\langle \Xi \lambda, \mu \rangle = \sum_{i=1}^{2} b_i(H_i \lambda, R_i \mu) + b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), R^*_2 \mu \right).
\]

Since \( R_i \) and \( R^*_2 \) can be any possible continuous extension operators, we will take \( R_i = H_i \) and \( R^*_2 = H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \mu}{\partial \nu} \right) \) in this case. These choices ensure symmetry of the operators, which is useful in this analysis. Then we have
\[
\langle \Xi \lambda, \mu \rangle = \sum_{i=1}^{2} b_i(H_i \lambda, H_i \mu) + b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \mu}{\partial \nu} \right) \right). \tag{3.2.110}
\]

The fact that for each \( \lambda \in \Lambda \), \( \langle \Xi \lambda, \mu \rangle \) is linear as a function of \( \mu \) follows from the linearity of integration. In the proof of the upcoming Theorem 3.2.5 it will be shown that \( \Xi \) is continuous. Thus for each \( \lambda \in \Lambda \), \( \langle \Xi \lambda, \mu \rangle \in \Lambda' \).
Next we proceed similarly for the right hand side of (3.2.101), recalling the definition of $P_i f$:

$$\langle \Sigma, \mu \rangle = - \int \Omega_2 \frac{\partial P_2 f}{\partial v^2} \mu - \int \Omega_1 \frac{\partial P_1 f}{\partial \nu} \mu - \int \Omega_2 \frac{\partial H_2 \left( \frac{1}{a_1} \frac{\partial P_1 f}{\partial \nu} \right)}{\partial v^2} \mu$$

$$= - \int \Omega_2 \frac{\partial P_2 f}{\partial v^2} R_2 \mu - \int \Omega_1 \frac{\partial P_1 f}{\partial \nu} R_1 \mu - \int \Omega_2 \frac{\partial H_2 \left( \frac{1}{a_1} \frac{\partial P_1 f}{\partial \nu} \right)}{\partial v^2} R_2 \mu$$

$$= \int \Omega_2 \nabla \cdot \left( k_2 \nabla P_2 f \right) R_2 \mu - \int \Omega_2 k_2 \nabla P_2 f \cdot \nabla R_2 \mu$$

$$+ \int \Omega_1 \nabla \cdot \left( k_1 \nabla P_1 f \right) R_1 \mu - \int \Omega_1 k_1 \nabla P_1 f \cdot \nabla R_1 \mu - \int \Omega_2 \frac{\partial H_2 \left( \frac{1}{a_1} \frac{\partial P_1 f}{\partial \nu} \right)}{\partial v^2} \cdot \nabla R_2 \mu$$

$$= \sum_{i=1}^2 \left( f_i, R_i \mu \right)_{\Omega_i} - b_i \left( P_i f, R_i \mu \right) - b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial P_1 f}{\partial \nu} \right), R_2 \mu \right).$$

Linearity of $\langle \Sigma, \mu \rangle$ follows from linearity of integration. Continuity of the $b_i(\cdot, \cdot)$ terms is demonstrated in the proof of Theorem 3.2.5. In the analysis of algorithm DDP we saw that as long as $f_i \in L^2(\Omega_i)$, the terms $(f_i, R_i \mu)_{\Omega_i}$ are continuous. Thus $\langle \Sigma, \mu \rangle \in \Lambda'$.

Now we can write the weak form of (3.2.101) as:

$$\text{find } \lambda \in \Lambda : \langle \Xi \lambda, \mu \rangle = \langle \Sigma, \mu \rangle \quad \forall \mu \in \Lambda. \quad (3.2.111)$$

**Theorem 3.2.5.** There exists $\theta_{\text{max}}^u > 0$ such that for any $0 < \theta < \theta_{\text{max}}^u$, and for any initial guess $\lambda^0$, algorithm DDC converges to the solution to (3.2.101).

**Proof.** Similarly to the proof for algorithm DDP, the proof of this theorem takes the following steps:
1. First we define an operator splitting for $\Xi$, and show that algorithm DDC is equivalent to the following preconditioned Richardson scheme for (3.2.101),

$$(\Xi_1^{-1} + \Xi_2^{-1})^{-1}(\lambda^{k+1} - \lambda^k) = \theta(\Xi\lambda - \Sigma).$$  \hspace{1cm} \text{(3.2.112)}

2. Next we show that

$$N_c := (\Xi_1^{-1} + \Xi_2^{-1})^{-1}$$ \hspace{1cm} \text{(3.2.113)}

satisfies the criteria from Lemma 3.2.2 on the Hilbert space $\Lambda$.

3. Finally we conclude via application of the Banach Contraction Mapping Principle 2.1.24 that the algorithm converges.

**Step 1**

We define the operator splitting of $\Xi$ by

$$\Xi_1\lambda := k_1^\Gamma \frac{\partial H_1\lambda}{\partial \nu}$$ \hspace{1cm} \text{(3.2.114)}

$$\Xi_2\lambda := k_2^\Gamma \frac{\partial H_2\lambda}{\partial \nu^2} + k_2^\Gamma \frac{\partial H_2}{\partial \nu} \left(\frac{1}{a_1} k_1^\Gamma \frac{\partial H_1\lambda}{\partial \nu} + \frac{\partial P_1f}{\partial \nu}\right).$$ \hspace{1cm} \text{(3.2.115)}

Recall that $u_1^{k+1} = H_1\lambda^k + P_1f$ and $u_2^{k+1} = H_2\lambda^k + H_2 \left(\frac{1}{a_1} k_1^\Gamma \left(\frac{\partial H_1\lambda^k}{\partial \nu} + \frac{\partial P_1f}{\partial \nu}\right)\right) + P_2f$

therefore,

$$\phi_1^{k+1}|_{\Gamma} = \Xi_1^{-1} \left(\frac{\partial H_1\lambda}{\partial \nu} + \frac{\partial P_1f}{\partial \nu}\right)$$
$$- \frac{\partial}{\partial \nu} \left(\frac{H_2\lambda^k + H_2 \left(\frac{1}{a_1} k_1^\Gamma \left(\frac{\partial H_1\lambda^k}{\partial \nu} + \frac{\partial P_1f}{\partial \nu}\right)\right)}{\partial \nu} - \frac{\partial P_2f}{\partial \nu}\right)$$ \hspace{1cm} \text{(3.2.116)}

$$= -\Xi_1^{-1}(\Sigma - \Xi \lambda^k)$$ \hspace{1cm} \text{(3.2.117)}

Similarly,

$$\phi_2^{k+1}|_{\Gamma} = \Xi_2^{-1}(\Sigma - \Xi \lambda^k).$$ \hspace{1cm} \text{(3.2.118)}
And so we have,
\[ \lambda^{k+1} = \lambda^k + \theta(\Xi^{-1} + \Xi^{-1})(\Sigma - \Xi \lambda^k). \]  
(3.2.119)

Or equivalently,
\[ N_c(\lambda^{k+1} - \lambda^k) = \theta(\Sigma - \Xi \lambda^k). \]  
(3.2.120)

**Step 2**

To show that \( N_c \) satisfies the criteria of Lemma 3.2.2, we must show that \( \Xi_i \) is continuous and coercive on \( \Lambda \) for \( i = 1, 2 \) and that \( N_c \) satisfies condition (3.2.54).

To show that \( \Xi_i \) is coercive we will again use the trace inequality. In the weak setting we have
\[
\langle \Xi_1 \lambda, \mu \rangle = b_1(H_1 \lambda, H_1 \mu),
\]  
(3.2.121)
\[
\langle \Xi_2 \lambda, \mu \rangle = b_2(H_2 \lambda, H_2 \mu) + b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \mu}{\partial \nu} \right) \right). 
\]  
(3.2.122)

Then we calculate,
\[
\langle \Xi_1 \lambda, \lambda \rangle \geq k_0^0 \|
abla H_1 \lambda \|^2_{L^2(\Omega_1)} \]  
(3.2.123)
\[
\geq k_0^0 P_1^2 \| H_1 \lambda \|^2_{H^1(\Omega_i)} \]  
(3.2.124)
\[
\geq k_0^0 P_1^2 C_1^2 \| H_1 \lambda \|^2_{\Lambda} \]  
(3.2.125)

where \( P_1 \) is the constant guaranteed by the Poincaré Inequality 2.1.9 and \( C_1 \) is the constant guaranteed by the Trace Inequality 2.1.8. Thus \( \Xi_1 \) is coercive with \( \alpha_1 = k_0^0 P_1^2 C_1^2 \).

To see that \( \Xi_2 \) is also coercive, note that
\[
b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \mu}{\partial \nu} \right) \right) \geq 0. \]  
(3.2.126)

So,
\[
\langle \Xi_2 \lambda, \lambda \rangle = b_2(H_2 \lambda, H_2 \lambda) + b_2 \left( H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right), H_2 \left( \frac{1}{a_1} \frac{\partial H_1 \lambda}{\partial \nu} \right) \right) \]  
(3.2.127)
\[
\geq b_2(H_2 \lambda, H_2 \lambda). \]  
(3.2.128)
The calculation to show that $b_2(H_2\lambda, H_2\lambda)$ is coercive is identical to the calculation for $\Xi_1$. Thus we conclude that $\Xi_2$ is coercive on $\Lambda$.

Next we calculate,

$$\langle \Xi_1\mu, \mu \rangle \leq k_1\|H_1\lambda\|_{H^1(\Omega_1)}\|H_1\mu\|_{H^1(\Omega_1)} \tag{3.2.129}$$

From the well known estimates for elliptic boundary value problems [[90], p. 9] we know there exists $\beta > 0$ such that

$$\|H_1\lambda\|_{H^1(\Omega_1)} \leq \beta\|\lambda\|_{\Lambda}. \tag{3.2.130}$$

Thus $\Xi_1$ is continuous with constant $C_1 = \epsilon_1 \beta$. Again the calculation for $\Xi_2$ follows.

Now we simply note that the $\Xi_i$ are clearly symmetric, and thus condition (3.2.54) is equivalent to the coercivity of $\Xi$. But the coercivity of $\Xi$ is a direct consequence of the coercivity of the $\Xi_i$. Thus condition (3.2.54) is satisfied, and the conditions of Lemma 3.2.2 are satisfied.

**Step 3**

Let $\theta_{max}^u$ be the constant guaranteed by Lemma 3.2.2. Let $0 < \theta < \theta_{max}^u$. From Lemma 3.2.2 we know that the operator

$$T_\theta^u := I - \theta N_c^{-1}\Xi \tag{3.2.131}$$

is a contraction on $\Lambda$. Notice,

$$N_c(\lambda^{k+1} - \lambda^k) = \theta(\Sigma - \Xi\lambda^k),$$

then solving for $\lambda^{k+1}$,

$$\lambda^{k+1} = \lambda^k - \theta N_c^{-1}(\Xi\lambda^k - \Sigma),$$

and applying (3.2.131),

$$\lambda^{k+1} = T_\theta^u \lambda^k + \theta N_c^{-1}\Sigma.$$
Then since \( \theta N^{-1}_c \Sigma \) does not depend on \( \lambda^k \), \( T^u_{\theta} \) a contraction implies

\[
G^u_{\theta} \lambda = T^u_{\theta} \lambda + \theta N^{-1}_c \Sigma
\]

(3.2.132)

is a contraction on the Hilbert space \( \Lambda \).

Now applying the Banach Contraction Mapping Principle 2.1.24 we know that algorithm DDC converges to a fixed point, \( \lambda_0 \) of the map \( G^u_{\theta} \). Then we have

\[
G^u_{\theta} \lambda_0 = \lambda_0,
\]

then recalling (3.2.132),

\[
T^u_{\theta} \lambda_0 + \theta N^{-1}_c \Sigma = \lambda_0,
\]

and recalling (3.2.131),

\[
\lambda_0 - \theta N^{-1}_c (\Xi \lambda_0 - \Sigma) = \lambda_0,
\]

then isolating the two free \( \lambda_0 \) terms

\[
\lambda_0 - \lambda_0 = \theta N^{-1}_c (\Xi \lambda_0 - \Sigma).
\]

Hence

\[
0 = \Xi \lambda_0 - \Sigma
\]

\( \Box \)

3.2.3.4 Relaxation Parameter Analysis

Next we consider the maximum relaxation parameter \( \theta^u_{\max} \). Similarly to the relaxation parameter analysis for algorithm DDP, we require estimates for the continuity and coercivity constants for the operator \( N (C_N \text{ and } \alpha_N, \text{ respectively}) \) as well as those for the operators \( \Xi_i, i = 1, 2 \) (\( C_i \) and \( \alpha_i \), respectively). We note that (3.2.83) and (3.2.84) hold in this case, as well. Recalling the continuity and coercivity estimates from the previous
section, we can again estimate $\theta_{\text{max}}^u$ via the constants $P_i$, $C_i$, $\beta_i$, and $k$. In Section 3.4, we will see for a simple problem that this analytical estimate does not perform well; i.e. that the resulting $\theta_{\text{max}}^u$, while correct in the sense that for each $\theta \in (0, \theta_{\text{max}}^u)$ algorithm DDC converges, is too small and the resulting method converges too slowly. Thus we will examine a computational technique for obtaining a much better result.

3.2.3.5 Algorithm DDN

When we formulated independent subdomain problems (3.2.89)-(3.2.90) from (3.2.85)-(3.2.88), considerable effort went into translating information from the Robin-type interface condition (3.2.86) to a Dirichlet type boundary condition at the interface for the subdomain problems. A more natural plan of attack is to formulate subdomain problems with a Neumann boundary condition at the interface built from (3.2.87), and to iterate to resolve the Robin condition (3.2.86). In this section we present an algorithm that follows this strategy. However, we will see that this formulation has tremendous difficulty with the data that is present in true heterostructure devices despite showing very strong performance on model problems. We present the method, since its more natural setup may be desirable for other applications with similar structure in their transmission conditions, but without the troublesome data in a heterostructure device.

We consider the problems,

$$-\nabla \cdot (k_i \nabla y_i) = f_i, \quad x \in \Omega_i, \quad k_i \frac{\partial u_i}{\partial \nu} = \lambda, \quad u_i|_{\partial \Omega_i \cap \partial \Omega} = 0.$$  

(3.2.133)

Note that, by construction, for any $\lambda$, (3.2.87) is satisfied. We are left with finding an appropriate $\lambda$ such that (3.2.86) is also enforced.

To develop a method for finding a suitable $\lambda$, we write,

$$u_i = Q_i \lambda + Y_i f.$$  

(3.2.134)
Here $Q_i \lambda$ denotes the harmonic extension of the Neumann data $\lambda$ into domain $i$,

$$-\nabla \cdot (k_i \nabla Q_i \lambda) = 0, \ x \in \Omega_i,$$  
(3.2.135)

$$\frac{\partial Q_i \lambda}{\partial \nu} = \lambda, \ x \in \Gamma,$$  
(3.2.136)

$$Q_i \lambda = 0, \ x \in \partial \Omega_i \setminus \Gamma,$$  
(3.2.137)

and $Y_i f$ denotes the response to the force,

$$-\nabla \cdot (k_i \nabla Y_i f) = f, \ x \in \Omega_i,$$  
(3.2.138)

$$\frac{\partial Y_i f}{\partial \nu} = 0, \ x \in \Gamma,$$  
(3.2.139)

$$Y_i f = 0, \ x \in \partial \Omega_i \setminus \Gamma.$$  
(3.2.140)

We then define $L$ and $\Theta$,

$$L \lambda := k_1^\Gamma \frac{\partial Q_1 \lambda}{\partial \nu} |_{\Gamma} + a_1 Q_1 \lambda |_{\Gamma} - a_2 Q_2 \lambda |_{\Gamma},$$  
(3.2.141)

$$\Theta := a_2 Y_2 f |_{\Gamma} - a_1 Y_1 f |_{\Gamma} - k_1^\Gamma \frac{\partial Y_1 f}{\partial \nu} |_{\Gamma},$$  
(3.2.142)

and the Steklov-Poincaré interface equation for algorithm DDN,

$$L \lambda = \Theta.$$  
(3.2.143)

**Lemma 3.2.6.** Suppose $\lambda$ satisfies the equation $L \lambda = \Theta$. Then a solution of (3.2.133) also solves (3.2.85)-(3.2.88).

**Proof.** Assume $\lambda$ satisfies $L \lambda = \Theta$ and that $u$ satisfies (3.2.133) with this $\lambda$. By construction we have,

$$-\nabla \cdot (k_i \nabla u_i) = f_i, \ x \in \Omega_i,$$  
(3.2.144)

and (3.2.87) is satisfied. We are left with showing that with this $\lambda$ (3.2.86) is also satisfied. Beginning from (3.2.143) and rearranging terms we have

$$k_1^\Gamma \frac{\partial (Q_1 \lambda + Y_1 f)}{\partial \nu} = a_2 (Q_2 \lambda + Y_2 f) - a_1 (Q_1 \lambda + Y_2 f).$$  
(3.2.145)

Then recalling (3.2.134) we see that (3.2.86) is satisfied.  \(\square\)
3.2.3.6 Algorithm DDN

To find an appropriate $\lambda$ and solve (3.2.133) with (3.2.143) we present algorithm DDN.

\textbf{Algorithm DDN to solve} (3.2.85)-(3.2.88)

Given $\lambda^0$, for each $k \geq 0$,

1. Solve (3.2.133) for $u_1^{k+1}$ and $u_2^{k+1}$.

2. Update $\lambda$ by

$$
\lambda^{k+1} = \lambda^k - \theta \left( k_1 \frac{\partial u_1^{k+1}}{\partial \nu}\big|_{\Gamma} - a_2 u_2^{k+1}\big|_{\Gamma} + a_1 u_1^{k+1}\big|_{\Gamma} \right). 
$$

(3.2.146)

3. Continue with (1) unless stopping criteria

$$
\left\| \left[ \frac{\partial u_1^{k+1}}{\partial \nu} |_{\Gamma} - a_2 u_2^{k+1}\big|_{\Gamma} + a_1 u_1^{k+1}\big|_{\Gamma} \right]_{\Gamma} \right\|. 
$$

(3.2.147)

holds.

3.2.3.7 Difficulties: A 1d Case Study

Algorithm DDN has a much more natural structure than algorithm DDC. However, the character of data in semiconductor problems make algorithm DDN an unsuitable candidate for this problem. To illustrate this we examine the properties of a suitable relaxation parameter for algorithm DDN applied to a Zinc Sulfide - Silicon heterojunction.

Similarly to algorithm DDC, algorithm DDN may be viewed as a Richardson scheme for the interface equation (3.2.143). To see this, we first establish the weak form of the
interface equation. Multiply $L\lambda$ by $\mu \in \lambda$ and integrate over $\Gamma$,

\[
\langle L\lambda, \mu \rangle = \int_{\Gamma} \left( k_1 \frac{\partial Q_1\lambda}{\partial \nu} + a_1 Q_1\lambda - a_2 Q_2\lambda \right) \mu \quad (3.2.148)
\]

\[
= \int_{\partial\Omega_1} k_1 \frac{\partial Q_1\lambda}{\partial \nu} \mathcal{R}_1 \mu + \int_{\Gamma} (a_1 Q_1\lambda - a_2 Q_2\lambda) \mu \quad (3.2.149)
\]

\[
= \int_{\Omega_1} k_1 \nabla Q_1\lambda \cdot \nabla \mathcal{R}_1 \mu + \int_{\Gamma} (a_1 Q_1\lambda - a_2 Q_2\lambda) \mu \quad (3.2.150)
\]

\[
= b_1(Q_1\lambda, \mathcal{R}_1 \mu) + \int_{\Gamma} (a_1 Q_1\lambda - a_2 Q_2\lambda) \mu. \quad (3.2.151)
\]

Here $\mathcal{R}_1$ denotes any continuous extension operator from $\Gamma$ into $V_{1,\triangle}$. Similarly we have

\[
\langle \Theta, \mu \rangle = \int_{\Gamma} (a_2 Y_2 f - a_1 Y_1 f) \mu - \int_{\partial\Omega_1} k_1 \frac{\partial Y_1 f}{\partial \nu} \mathcal{R}_1 \mu \quad (3.2.152)
\]

\[
= \int_{\Gamma} (a_2 Y_2 f - a_1 Y_1 f) \mu - b_1(Y_1 f, \mathcal{R}_1 \mu). \quad (3.2.153)
\]

Then choosing $\mathcal{R}_1 \mu = Q_1 \mu$, we have

\[
\langle L\lambda, \mu \rangle = b_1(Q_1\lambda, Q_1 \mu) + \int_{\Gamma} (a_1 Q_1\lambda - a_2 Q_2\lambda) \mu, \quad (3.2.154)
\]

and the weak form of the interface equation,

\[
\text{find } \lambda \in \Lambda : \langle L\lambda, \mu \rangle = \langle \Theta, \mu \rangle, \quad \mu \in \Lambda. \quad (3.2.155)
\]

To estimate the required bound on $\theta$, we consider the interface update step as a Richardson scheme on $\Lambda$.

\[
\lambda^{k+1} = \lambda^k - \theta (L\lambda^k - Y). \quad (3.2.156)
\]

Then rewriting this in the common presentation,

\[
\lambda^{k+1} = (I - \theta L)\lambda^k + \theta Y. \quad (3.2.157)
\]

Then our iteration operator is $I - \theta L$, and if $\|I - \theta L\| < 1$, the Richardson scheme will converge. To find the necessary bound on $\theta$ we let $J$ denote the Riesz map and consider the weak operator $\langle (J - \theta L)\lambda, \mu \rangle$.

\[
\langle (J - \theta L)\lambda, \mu \rangle = \left| \int_{\Gamma} \left( \lambda + \theta a_2 Q_2\lambda - \theta a_1 Q_1\lambda - \theta k_1 \frac{\partial Q_1\lambda}{\partial \nu} \right) \mu \right|. \quad (3.2.158)
\]
For exposition, we assume $k_i$ is constant for each $i$, and see that in 1$d$,

\[
|\langle(\beta - \theta L)\lambda, \mu\rangle| = \left|1 - \left(\theta \frac{a_2 d_2}{k_2} + \theta \frac{a_1 d_1}{k_1} + \theta\right)\lambda \right| \mu,
\]

where $d_i$ is the length of domain $i$. Taking the domain sizes to be the lengths of the domains in the ZnS/Si heterojunction we simulate later in this work we have $d_1 = 0.1 \mu m = 10^{-7} m$ and $d_2 = 2 \mu m = 2 \times 10^{-6} m$. Then applying the values $a_i$ from the ZnS/Si heterojunction, to enforce a contraction property on the iteration operator we require

\[
|1 - \theta(1.205 \times 10^{50})| < 1,
\]

and so, we require,

\[
0 < \theta < 1.66 \times 10^{-57}
\]

which is clearly well below any reasonably obtained precision for computation. So while this algorithm may seem more natural, and we may prove a similar convergence result to those for algorithms DDP and DDC, the data requires an unrealistic relaxation parameter for heterojunction semiconductor simulation.

### 3.2.4. Heterojunction Model Analysis

To conclude this section we present preliminary work towards analysis of the heterostructure model. Our scheme is to analyze the DDM formulations of the component equations of the heterostructure model, obtaining a priori estimates on their solutions. Then to complete the analysis a fixed point analysis of the Gummel Map in the heterostructure model is necessary. This fixed point analysis will be the subject of future work, building upon the analysis in this section. However, we will outline the full analysis here.

We consider here the heterostructure model in Slotboom variables with the Thermionic Emission model. Additionally, we lag the semilinearity in each equation; i.e. the Gummel map will be employed to linearize the component equations as well as to decouple the
A lagged term, or previous iterate, is identified by a tilde, $\tilde{u}$. For boundary conditions we consider conditions appropriate for inducing current in the direction orthogonal to the heterostructure interface. To illustrate this we divide the boundaries $\Gamma_i = \partial \Omega_i$ into components $\Gamma_{i,D}$ and $\Gamma_{i,N}$. Here $\Gamma_{i,D}$ represents contacts and $\Gamma_{i,N}$ represents insulating portions of the boundary. Figure 3.3 illustrates this domain in 2d. We note that this setup represents a heterojunction diode. The full model with notation in place for identifying the lagged terms is,

$$-\nabla \cdot (\epsilon_i \nabla \psi_i) := \delta_p^2 e^{-\psi_i} \tilde{v}_i - \delta_n^2 \bar{\psi}_i \tilde{u}_i + N_T$$  \hspace{1cm} (3.2.162)

$$[\psi]_\Gamma = \psi_\Delta$$  \hspace{1cm} (3.2.163)

$$\left[ \epsilon \frac{\partial \psi}{\partial \nu} \right]_\Gamma = 0,$$  \hspace{1cm} (3.2.164)

$$\psi_i|_{\Gamma_{i,D}} = \psi_i, \hspace{1cm} \frac{\partial \psi_i}{\partial \nu}|_{\Gamma_{i,N}} = 0,$$  \hspace{1cm} (3.2.165)

$$-\nabla \cdot (D_n \delta_n^2 e^{\psi_i} \nabla u_i) = R(\psi_i, \tilde{u}_i, \tilde{v}_i),$$  \hspace{1cm} (3.2.166)

$$(D_n \delta_n^2 e^{\psi} \nabla u)_1 = a_n^1 (e^{\psi} u)_1^2 - a_n^2 (e^{\psi} u)_1^1,$$  \hspace{1cm} (3.2.167)

$$\left[ D_n \delta_n^2 e^\psi \nabla u \right]_\Gamma = 0,$$  \hspace{1cm} (3.2.168)

$$u_i|_{\Gamma_{i,D}} = u_i, \hspace{1cm} \frac{\partial u_i}{\partial \nu}|_{\Gamma_{i,N}} = 0,$$  \hspace{1cm} (3.2.169)

$$-\nabla \cdot (D_p \delta_p^2 e^{-\psi_i} \nabla v_i) = -R(\psi_i, \tilde{u}_i, \tilde{v}_i),$$  \hspace{1cm} (3.2.170)

$$(D_p \delta_p^2 e^{-\psi} \nabla v)_1^1 = a_p^1 (e^{-\psi} u)_1^2 - a_p^2 (e^{-\psi} u)_1^1,$$  \hspace{1cm} (3.2.171)

$$\left[ D_p \delta_p^2 e^{-\psi} \nabla v \right]_\Gamma = 0,$$  \hspace{1cm} (3.2.172)

$$v_i|_{\Gamma_{i,D}} = v_i, \hspace{1cm} \frac{\partial v_i}{\partial \nu}|_{\Gamma_{i,N}} = 0.$$  \hspace{1cm} (3.2.173)

Here $u_{i,D}$ and $v_{i,D}$ are given by the thermal equilibrium values obtained by setting the
quasi-Fermi potentials to 0. Similarly \( \psi_{i,D} = \psi_{i,D}^{TE} + V_{ext} \) where \( \psi_{i,D}^{TE} \) is the thermal equilibrium value and \( V_{ext} \) is an externally applied voltage.

![Illustration of boundary setup in 2d for heterojunction model analysis](image)

**FIGURE 3.3:** Illustration of boundary setup in 2d for heterojunction model analysis

We make the following assumptions, some of which are restatements of earlier assumptions,

**Assumption 3.2.7.**

(A1) The data \( \psi_{i,D}, u_{i,D}, \) and \( v_{i,D} \) satisfy:

(i) \( \psi_{i,D} \in H^1(\Omega_i) \cap L^\infty(\Omega_i) \) and there exist constants \( \psi_{lb} \) and \( \psi_{ub} \) such that

\[
\psi_{ub} > \psi_{i,D}|_{\Gamma_{i,D}} > \psi_{lb} > 0. \tag{3.2.177}
\]

(ii) \( u_{i,D} \in H^1(\Omega_i) \cap L^\infty(\Omega_i) \) and there exist constants \( u_{lb} \) and \( u_{ub} \) such that

\[
u_{ub} > u_{i,D}|_{\Gamma_{i,D}} > u_{lb} > 0. \tag{3.2.178}
\]

(iii) \( v_{i,D} \in H^1(\Omega_i) \cap L^\infty(\Omega_i) \) and there exist constants \( v_{lb} \) and \( v_{ub} \) such that

\[
u_{ub} > v_{i,D}|_{\Gamma_{i,D}} > v_{lb} > 0. \tag{3.2.179}
\]
(A2) The data $\tilde{\psi}, \tilde{u}, \tilde{v}, N_T,$ and $R$ satisfy,

(i) $\tilde{\psi} \in V \cap L^\infty(\Omega)$.

(ii) $\tilde{u} \in V, \tilde{v} \in V, N_T \in V$.

(iii) $R : V^3 \rightarrow V$.

(A3) (coercivity assumptions)

(i) There exists $\epsilon_i^0 > 0$ and $\epsilon_i^1 > 0$ such that

$$0 < \epsilon_i^0 < \epsilon_i < \epsilon_i^1$$

on $\Omega_i$, for each $i$.

(ii) There exists $D_{n,i}^0 > 0$ and $D_{n,i}^1 > 0$ such that

$$0 < D_{n,i}^0 < D_{n,i} < D_{n,i}^1$$

on $\Omega_i$, for each $i$.

(iii) There exists $D_{p,i}^0 > 0$ and $D_{p,i}^1 > 0$ such that

$$0 < D_{p,i}^0 < D_{p,i} < D_{p,i}^1$$

on $\Omega$, for each $i$.

(A4) The domain $\Omega$ satisfies:

(i) $\Gamma \cap \Gamma_{i,D} = \emptyset$ for $i = 1, 2$.

(ii) The $N$-1 dimensional Lebesgue measure of $\Gamma_{i,D}$ is positive for $i = 1, 2$.

Also we define the following vector spaces and bilinear forms, some of which are
restatements from earlier analysis,

\[ V_i := H^1(\Omega_i) \quad (3.2.183) \]

\[ V_i^0 := \{ v \in V_i \mid \gamma_i v|_{\Gamma_{i,i}} = 0 \}, \quad (3.2.184) \]

\[ V := \{ v \in L^2(\Omega) \mid v|_{\Omega_i} \in V_i^0 \text{ for } i = 1, 2 \}, \quad (3.2.185) \]

\[ V_{i,\Delta}^0 := \{ v \in H^1(\Omega_i) \mid \gamma_{i1} v|_{\Gamma_{1,i}} = 0, \ \triangle v \in H^{-1}(\Omega_i) \}, \quad (3.2.186) \]

\[ V_{\Delta} := \{ v \in L^2(\Omega) \mid v_1 \in V_{1,\Delta}^0, \ v_2 \in V_2^0 \}, \quad (3.2.187) \]

\[ \Lambda := H^{\frac{1}{2}}(\Gamma), \quad (3.2.188) \]

\[ a_i(\cdot, \cdot) : V_i \times V_i \to \mathbb{R}, \quad a_i(u, v) = \int_{\Omega_i} \epsilon_i \nabla u \cdot \nabla v, \quad (3.2.189) \]

\[ c_i(\cdot, \cdot, \psi) : V_i \times V_i \to \mathbb{R}, \quad c_i(u, v, \psi) = \int_{\Omega_i} D_{n,i} \delta_n^2 e^{\psi} \nabla u \cdot \nabla v, \quad (3.2.190) \]

\[ d_i(\cdot, \cdot, \psi) : V_i \times V_i \to \mathbb{R}, \quad d_i(u, v, \psi) = \int_{\Omega_i} D_{p,i} \delta_p^2 e^{-\psi} \nabla u \cdot \nabla v, \quad (3.2.191) \]

\[ (\cdot, \cdot)_i : V_i \times V_i \to \mathbb{R}, \quad (u, v)_i = \int_{\Omega_i} uv. \quad (3.2.192) \]

### 3.2.4.1 Heterostructure Gummel Map

To analyze the heterostructure model we will need to be more precise in defining the Gummel Map for the heterostructure model. We define the map \( G = UV \circ \Psi : L^2(\Omega)^3 \to L^2(\Omega)^3 \) as the composition of two maps \( UV : L^2(\Omega)^3 \to L^2(\Omega)^3 \) and \( \Psi : L^2(\Omega)^3 \to L^2(\Omega)^3 \).

**Definition 3.2.8.** Given \((\tilde{\psi}, \tilde{u}, \tilde{v}) \in L^2(\Omega)^3\), solve (3.2.162)-(3.2.166) for \( \psi \in L^2(\Omega) \).

Define,

\[ \Psi(\tilde{\psi}, \tilde{u}, \tilde{v}) = (\psi, \tilde{u}, \tilde{v}). \quad (3.2.193) \]

**Definition 3.2.9.** Given \((\psi, \tilde{u}, \tilde{v}) \in L^2(\Omega)^3\), solve (3.2.167)-(3.2.171) for \( u \in L^2(\Omega) \) and solve (3.2.172)-(3.2.176) for \( v \in L^2(\Omega) \). Define,

\[ UV(\psi, \tilde{u}, \tilde{v}) = (\psi, u, v). \quad (3.2.194) \]
Definition 3.2.10. Given $(\tilde{\psi}, \tilde{u}, \tilde{v}) \in L^2(\Omega)^3$, the Gummel Map $G$ is defined by

\[ G(\tilde{\psi}, \tilde{u}, \tilde{v}) = \mathbf{UV}(\Psi(\tilde{\psi}, \tilde{u}, \tilde{v})) \]
\[ = \mathbf{UV}(\psi, \tilde{u}, \tilde{v}) \]
\[ = (\psi, u, v). \]

The map $G$ linearizes and decouples the system (3.2.162)-(3.2.176). A fixed point of the map $G$ corresponds to a solution of the system $(\psi, u, v) \in L^2(\Omega)^3$.

3.2.4.2 Component Equation Analysis

To analyze the heterostructure model the following theorem will be applied to each component equation to prove their decoupled well-posedness and to obtain an a priori estimate on each solution.

Theorem 3.2.11. Let $V$ and $W$ be Hilbert spaces. Let $A \in \mathcal{L}(V, V')$ and $B_i \in \mathcal{L}(V, W')$, $i = 1, 2$. Assume

- $A$ is non-negative and it is $V$-coercive on $\mathbf{Ker} \ B_1 \cap \mathbf{Ker} \ B_2$,
- $B_i'$ is bounding for $i = 1, 2$.

Then for every $f \in V'$ and for every $g_1 \in W'$, $g_2 \in W'$, $g_i \neq 0$ such that

\[ \{ u \in V : B_1 u - g_1 \in W^o \& B_2 u - g_2 \in W^o \} \neq \emptyset, \]

the system

\[ u \in V : B_1 u - g_1 \in W^o, B_2 u - g_2 \in W^o, \text{ and } Au - f \in (\mathbf{Ker} \ B_1 \cap \mathbf{Ker} \ B_2)^o, \]

has a unique solution $u \in V$ which satisfies the estimate

\[ \|u\|_V \leq \frac{1}{\alpha} \left( \|f\|_{V'} + \frac{1}{\beta} (\|A\|_{\mathcal{L}(V, V')} + \alpha) \max(\|g_1\|_{W'}, \|g_2\|_{W'}) \right). \]
Proof. Let \( u_g \in (\text{Ker } \mathcal{B}_1 \cap \text{Ker } \mathcal{B}_2)^\perp \) with \( \mathcal{B}_1 u = g_1 \) and \( \mathcal{B}_2 u = g_2 \). Then we seek

\[ u_0 \in \text{Ker } \mathcal{B}_1 \cap \text{Ker } \mathcal{B}_2 : \mathcal{A}u_0 + \mathcal{A}u_g - f \in (\text{Ker } \mathcal{B}_1 \cap \text{Ker } \mathcal{B}_2)^\circ. \quad (3.2.201) \]

Since \( \mathcal{B}_i \) is continuous and linear for each \( i = 1, 2 \), Ker \( \mathcal{B}_i \) is a closed subspace of the Hilbert space \( V \) for each \( i \). Thus Ker \( \mathcal{B}_i \) is a Hilbert space, and consequently so is Ker \( \mathcal{B}_1 \cap \text{Ker } \mathcal{B}_2 \). Then the Lax-Milgram Theorem 2.1.16 guarantees a unique solution to this problem.

Next we turn to the estimate (3.2.200). Our first step is to write down the problem in a 'mixed' formulation; i.e. to formulate the constraints \( \mathcal{B}_i u = g_i \) as Lagrange multipliers for the system,

\[
[u, p_1, p_2] \in V \times W \times W : \\
\mathcal{A}v - f + \mathcal{B}_1' q_1 + \mathcal{B}_2' q_2 \in V^\circ, \\
\mathcal{B}_1 v - g_1 \in W^\circ, \\
\mathcal{B}_2 v - g_2 \in W^\circ. 
\quad (3.2.202, 3.2.203, 3.2.204)
\]

Since \( \mathcal{B}_i' : W \to V' \) is bounding, we know there exists \( \beta = \min_i \beta_i \) such that

\[ \| \mathcal{B}_i' q_i \|_{V'} \geq \beta \| q_i \|_W, \quad q_i \in W, \ i = 1, 2. \quad (3.2.205) \]

In particular, we may choose \( u_g \) such that

\[ \| u_g \|_V \leq \frac{1}{\beta} \max_i \| g_i \|_{W'}. \quad (3.2.206) \]

Then the Lax-Milgram Theorem 2.1.16 gives us the estimate,

\[ \| u_0 \|_V \leq \frac{1}{\alpha} (\| f \|_{V'} + \| \mathcal{A}u_g \|_{V'}). \quad (3.2.207) \]
Combining these, we have

\[ \|u\|_V = \|u_0 + u_g\|_V \]

\[ \leq \|u_0\|_V + \|u_g\|_V \]

\[ \leq \frac{1}{\alpha}(\|f\|_V + \|Au_g\|_V) + \frac{1}{\beta} \max_i \|g_i\|_{W'} \]

\[ \leq \frac{1}{\alpha}(\|f\|_V + \|A\xi(V,V')u_g\|_V) + \frac{1}{\beta} \max_i \|g_i\|_{W'} \]

\[ \leq \frac{1}{\alpha}\left(\|f\|_V + \max_i \|g_i\|_{W'} \right) + \frac{1}{\beta} \max_i \|g_i\|_{W'} \]

\[ = \frac{1}{\alpha}\left(\|f\|_V + \frac{1}{\beta}(\|A\xi(V,V') + \alpha) \max_i \|g_i\|_{W'} \right), \]

which is the desired estimate.

3.2.4.3 Potential Equation Analysis

To perform analysis on the potential equation in the heterostructure model we write the weak form of the potential equation in the domain decomposition formulation coupled to the weak Steklov-Poincaré operator so that we may apply Theorem 3.2.11. Define

\[ A_w(y) = \sum_{i=1}^2 a_i(w_i, y_i), \] (3.2.208)

\[ q_i = \delta^i e^{-\tilde{\psi}_i} \tilde{v}_i - \delta^i e^{\tilde{\psi}_i} \tilde{u}_i + N_T, \] (3.2.209)

\[ f(y) = \sum_{i=1}^2 \int_{\Omega_i} [g_i y_i - \epsilon_i \nabla \psi_{i,D} \cdot \nabla y_i], \] (3.2.210)

\[ B_1w = \gamma(w_1)|_\Gamma, \] (3.2.211)

\[ B_2w = \gamma(w_2)|_\Gamma, \] (3.2.212)

\[ g_1(\lambda) = \lambda - \gamma(\psi_{1,D})|_\Gamma, \] (3.2.213)

\[ g_2(\lambda) = \lambda + \psi_\Delta - \gamma(\psi_{2,D})|_\Gamma. \] (3.2.214)
Then we consider the problem,

\[
\psi \in V, \ \lambda \in \Lambda,
\]

\[
\mathcal{B}_1 \psi - g_1(\lambda) = 0 \in \Lambda', \quad (3.2.215)
\]

\[
\mathcal{B}_2 \psi - g_2(\lambda) = 0 \in \Lambda', \quad (3.2.216)
\]

\[
\mathcal{A} \psi - f \in (\text{Ker} \ \mathcal{B}_1 \cap \text{Ker} \ \mathcal{B}_2)^o, \quad (3.2.217)
\]

\[
K \lambda - \Upsilon = 0 \in \Lambda'. \quad (3.2.218)
\]

We saw already in previous analysis that the operator \( K : \Lambda \to \Lambda' \) is linear, continuous and coercive on the Hilbert space \( \Lambda \), and that \( \Upsilon \in \Lambda' \). Thus the Lax-Milgram Theorem 2.1.16 guarantees a unique solution \( \lambda \) satisfying

\[
\|\lambda\|_{\Lambda} \leq \frac{1}{\alpha_K} \|\Upsilon\|_{\Lambda'}, \quad (3.2.219)
\]

where \( \alpha_K \) is the coercivity constant for the operator \( K \).

With \( \lambda \) fixed as the solution to the weak Steklov-Poincaré equation for the potential, we have a problem exactly in the form appropriate for application of Theorem 3.2.11, and are left to establish that the operators \( \mathcal{A} \) and \( \mathcal{B}_i \) satisfy the hypotheses of the theorem.

The linearity of the operator \( \mathcal{A} \) is clear. Establishing the continuity and coercivity is a direct result of the coercivity and continuity of the component bilinear forms \( a_i \). In each case this is a standard result due to the assumed bounds on the data \( \epsilon_i \). The non-degeneracy of the data \( \epsilon_i \) provides coercivity while the bound from above guarantees continuity.

Establishing that \( \mathcal{B}_i \) are bounding isn’t as pedestrian, though is straightforward after application of Theorem 2.1.21. In particular, we leverage the equivalence between items 1 and 4 in the theorem. \( \mathcal{B}_i w \) is, for each \( i \), the restriction of the image of the trace operator acting on \( w \) onto the interface \( \Gamma \). Citing the Trace Theorem 2.1.7, we know that \( \mathcal{B}_i \) is surjective for each \( i \), and thus \( \mathcal{B}_i' \) is bounding, for each \( i \).
Finally we must establish
\[
\{ u \in V : \mathcal{B}_1 u - g_1 \in \Lambda^o & \mathcal{B}_2 u - g_2 \in \Lambda^o \} \neq \emptyset. \tag{3.2.220}
\]

But since \( V \) is a broken Sobolev space, we may simply take any continuous extension of \( g_i \) into \( V_i \) for each \( i \) and paste these together to establish that this set is nonempty. Thus given the assumption that \( g_i \in \Lambda \), this condition is satisfied.

The preceding discussion establishes the following theorem.

**Theorem 3.2.12.** Under Assumptions 3.2.7 there exists a unique solution to the lagged potential equation in the heterostructure model (3.2.215)-(3.2.218) satisfying the estimate
\[
\|\psi\|_V \leq \frac{1}{\alpha} \left( \|f\|_{V'} + \frac{1}{\beta} \left( \|A\|_{\mathcal{L}(V,V')} + \alpha \right) \max_i \|g_i\|_{\Lambda'} \right). \tag{3.2.221}
\]

Here \( \alpha \) is the coercivity constant associated to the operator \( A \) and \( \beta = \min_i \beta_i \) where \( \beta_i \) is the bounding constant for the dual operator \( \mathcal{B}'_i \).

### 3.2.4.4 Continuity Equations Analysis

Analyses of the equations for electron density and hole density are, unsurprisingly, identical. Thus we restrict attention to the equation for electron density.

We consider the weak form of the equation for electron density in Slotboom variables coupled to the weak Steklov-Poincaré equation (3.2.111). Define,

\[
\mathcal{C} w(y) = \sum_{i=1}^{2} c_i(w_i, y_i), \tag{3.2.222}
\]
\[
\tau(y) = \sum_{i=1}^{2} (R_i(\psi_i, \tilde{u}_i, \tilde{v}_i), y_i)_i, \tag{3.2.223}
\]
\[
\mathcal{D}_1 w = \gamma(w_1)_{\Gamma}, \tag{3.2.224}
\]
\[
\mathcal{D}_2 w = \frac{a_2}{a_1} e^{\gamma(\psi_2)_{\Gamma} - \gamma(\psi_1)_{\Gamma}} \gamma(w_2)_{\Gamma} - \frac{1}{a_1} \frac{\partial u_1}{\partial \nu^1}, \tag{3.2.225}
\]
\[
h_i(\lambda) = \lambda - \gamma(u_i, \nu)_{\Gamma}. \tag{3.2.226}
\]
Then we consider the problem,

\[ u \in V_\Delta, \lambda \in \Lambda, \]
\[ D_1 u - h_1(\lambda) = 0 \in \Lambda', \tag{3.2.227} \]
\[ D_2 u - h_2(\lambda) = 0 \in \Lambda', \tag{3.2.228} \]
\[ \mathcal{C} u - r \in (\text{Ker } D_1 \cap \text{Ker } D_2)^\circ, \tag{3.2.229} \]
\[ \Xi \lambda - \Sigma = 0 \in \Lambda'. \tag{3.2.230} \]

We saw already in previous analysis that the operator \( \Xi : \Lambda \to \Lambda' \) is linear, continuous and coercive on the Hilbert space \( \Lambda \), and that \( \Sigma \in \Lambda' \). Thus the Lax-Milgram Theorem guarantees a unique solution satisfying

\[ \| \lambda \|_\Lambda \leq \frac{1}{\alpha_\Xi} \| \Sigma \|_{\Lambda'}, \tag{3.2.231} \]

where \( \alpha_\Xi \) is the coercivity constant for the operator \( \Xi \).

With \( \lambda \) fixed as the solution to the Steklov-Poincaré equation for the equation for electron density, we have a problem exactly in the form appropriate for application of Theorem 3.2.11, and are left to establish that the operators \( \mathcal{C} \) and \( D_i \) satisfy the hypotheses of the theorem. This process is identical to that for the potential equation. For the operator \( \mathcal{C} \) we have component weighted bilinear forms whose continuity and coercivity depend on the boundedness and non-degeneracy of the weights. Then, due to assumptions on data in Assumption 3.2.7, we have continuity and coercivity for the operator \( \mathcal{C} \). To see that the dual operators \( D_i^\prime \) are bounding, we again cite the surjectivity of the trace operator.

Finally, the condition

\[ \{ u \in V_\Delta : D_1 u - h_1 \in \Lambda^\circ & D_2 u - h_2 \in \Lambda^\circ \} \neq \emptyset. \tag{3.2.232} \]

is again a consequence of the broken Sobolev space construction. We take any continuous extension of the data \( h_i \) into the spaces \( V_{1,\Delta}^\circ \) and \( V_2^\circ \), and together have an element of this set.
The preceding discussion establishes the following theorem.

**Theorem 3.2.13.** Under Assumptions 3.2.7 there exists a unique solution to the lagged equation for electron density in the heterostructure model (3.2.227)-(3.2.230) satisfying the estimate

$$\|u\|_V \leq \frac{1}{\alpha} \left( \|r\|_{V'} + \frac{1}{\beta} (\|\mathcal{C}\|_{\mathcal{L}(V,V')} + \alpha) \max_i \|h_i\|_{\Lambda'} \right).$$  \hspace{1cm} (3.2.233)

Here $\alpha$ is the coercivity constant associated to the operator $\mathcal{C}$ and $\beta = \min_i \beta$ where $\beta_i$ is the bounding constant for the dual operator $\mathcal{D}_i'$.

### 3.2.4.5 Roadmap to Heterostructure Analysis

To complete analysis of the heterostructure model a fixed point analysis of the Gummel map $\mathbf{G}$ is required – a fixed point $\mathbf{G}(\psi, u, v) = (\psi, u, v)$ corresponds to a weak solution of the full heterostructure model. This step would apply the estimates from the theorems in the previous section, and depending on the data, would fall in one of two categories.

- Establish a strong contraction property for $\mathbf{G}$ and apply the Banach Contraction Mapping Principle 2.1.24 to guarantee the existence of a unique fixed point to the model, and thus a unique solution.

- Establish that the map $\mathbf{G}$ is a continuous operator that maps a nonempty convex subset $K \subset V^3$ into a compact subset of $K$, and apply the Schauder Fixed Point Theorem 2.1.25 to guarantee the existence of a fixed point, and thus a solution to the model.

The difference between these two cases is not simply a matter of choice in analysis. It is well known that with certain data solutions to the drift-diffusion model are not unique. Establishing strict rules on the data that distinguishes these cases is thus a matter of practical importance. When dealing with data that is suitable to application of the Banach
Contraction Mapping principle the computational scientist can rest assured that beginning from any data the unique solution will be found by the Gummel iteration. However, when the Banach Contraction Mapping Principle cannot be applied greater care is required during simulation to guarantee that a solution is found, and that it is the appropriate solution. Further, in the latter situation, the Schauder Fixed Point Theorem not only does not guarantee uniqueness of the solution but also has nothing to say regarding the convergence of the iterative process.
3.3. Heterojunction Uncertainty

The results of any computational simulation depend on the reliability of the parameters used. Since parameters are frequently determined experimentally, it is important to know how sensitive the solutions, or quantities of interest computed from solutions, are to parameter variation.

For the heterostructure semiconductor model the quantities of interest are, among others, the $J-V$ curve. In 1d the current $J = J_n + J_p$ is the total (constant) current, and $V$ is the external voltage drop used to determine external boundary conditions for the potential. The $J-V$ curve is a set of values of $J$ obtained for prescribed values of $V$. Next, $P_{\text{max}}$ is the point on the $J-V$ curve which corresponds to the maximum value of the power $P = JV$. Finally, the values $I_{\text{sc}}$ and $V_{\text{oc}}$ are, respectively, the maximum current, and the maximum voltage that a system (device + solar) can produce.

In this section we address the dependence of the quantities of interest in the heterostructure model on the parameter $\psi_\Delta$, obtained from DFT calculations. We first address the error associated with DFT at the atomic scale, and then consider its propagation through the continuum model. In Section 3.4. we perform numerical experiments as described in this section.

3.3.0.6 Error at the Atomic Scale

The error associated with the calculation of the potential jump $\psi_\Delta$, for a given interface (heterojunction) microstructure is typically quoted as $\pm0.1$ Volts (V), which at room temperature is equivalent to an error of approximately $\pm4$ imposed on $\psi_\Delta$. The uncertainty arises from error within the approximations used in the DFT calculations themselves, as well as from error associated with the imposition of imperfect boundary conditions at finite distances from the interface. The former error is systematically reducible only by the use of various computationally expensive electronic structure calculations, such as the
GW calculation [47]. The latter error ("finite length" error) can, in principle, be reduced arbitrarily using longer quasi-1d modeling cells for the DFT calculation.

3.3.0.7 Propagation Through Continuum Model

To study the propagation of the error in $\psi_\Delta$ from the DFT calculation to the quantities of interest (QoIs), we set up a simple stochastic model. We assume the heterojunction parameter is a random variable $\Psi_\Delta$ with mean $\psi_\Delta$ obtained by DFT simulations and that $\Psi_\Delta$ is uniformly distributed

$$\Psi_\Delta = U(\psi_\Delta - \vartheta, \psi_\Delta + \vartheta).$$

(3.3.1)

The heterojunction parameter controls the band offsets, as described in Ref. [30]. The band offsets, in turn, affect the quantities of interest $V_{oc}, J_{sc},$ and $P_{max}$. Since $\Psi_\Delta$ is a random variable, so are all computed quantities of interest. We are interested in the dependence of the distributions of these quantities upon $\Psi_\Delta$.

We would like to obtain moments of the functions defined by the dependence of the QoIs on the random variable $\Psi_\Delta$. To this end we apply a stochastic collocation approach by simulating the deterministic model at the Gaussian quadrature points associated to the integral for the expectation $E[Q(\Psi_\Delta)]$ of $Q$,

$$E[Q(\Psi_\Delta)] := \int_{\psi_\Delta-\vartheta}^{\psi_\Delta+\vartheta} Q(\Psi_\Delta) f_{\Psi_\Delta} d\Psi_\Delta$$

(3.3.2)

where $f_{\Psi_\Delta}$ is the (uniform) probability density function associated to the random variable $\Psi_\Delta$ and $Q$ is one of $J_{sc}, V_{oc}, P_{max}$. In stochastic collocation one approximates the expectation of $Q$ using numerical integration

$$E[Q(\Psi_\Delta)] \approx \sum_m w_m Q(\psi_m)$$

(3.3.3)

where the collocation points $\psi_m$ and weights $w_m$ are optimal for the (uniform) density. We then use solutions to obtain statistics other than $E[Q]$ on the quantities of interest.

In Section 3.4. we report our findings for two heterostructures: ZnS/Si (Zinc Sulfide / Silicon) and CdS/CIGS (Cadmium Sulfide / Copper Indium Gallium Selenide).
3.4. Numerical Results

In this section we perform numerical tests of the domain decomposition algorithms presented in this work. We first consider convergence and other performance aspects of algorithms DDP and DDC, respectively. We then examine the difficulty algorithm DDN experiences with true heterostructures, while noting its good performance on model problems. We then apply algorithms DDP and DDC to true heterostructure semiconductors. Finally, we perform sensitivity analysis to the uncertainty in the angstrom scale DFT calculations.

3.4.1. Algorithm DDP

In this section we first perform convergence and mesh independence tests for algorithm DDP on a 1d model problem. We then address the practical choice of the relaxation parameter $\theta$. In all simulations we set the tolerance for convergence of algorithm DDP to $\tau = 10^{-12}$.

The 1d model problem we consider is,

$$-\triangle u_i = \frac{2x}{(x^2 + 1)^2}, \quad x \in \Omega_i, \quad (3.4.1)$$

$$[u]_{\Gamma=[0]} = 0.1, \quad \left[ \frac{\partial u}{\partial v} \right]_{\Gamma=[0]} = 0, \quad (3.4.2)$$

$$u_1(-1) = 0, \quad u_2(1) = \frac{1}{4}, \quad (3.4.3)$$

where $\Omega_1 = (-1,0)$, $\Omega_2 = (0,1)$. Figure 3.4 shows the solution to this problem obtained with algorithm DDP. For convergence testing, we compare algorithm DDP to the analytical solution to this problem,

$$u(x) = \begin{cases} u_1(x) = \hat{u}(x), & x \in \Omega_1, \\ u_2(x) = \hat{u}(x) + 0.1, & x \in \Omega_2 \end{cases} \quad (3.4.4)$$

$$\hat{u}(x) = \arctan(x) - 0.710398x + 0.075. \quad (3.4.5)$$
Figure 3.4: Solution to 1d model problem with algorithm DDP.

Table 3.1 demonstrates order 2 convergence for this problem using centered finite differences and algorithm DDP. We see here that the expected order of convergence for the discretization is obtained while using algorithm DDP. Table 3.2 demonstrates the mesh independence of algorithm DDP. This table shows that the number of domain decomposition iterations is independent of the resolution of the mesh in each subdomain; an important consideration for the efficiency of the method. In these tables $N = N_1 + N_2$, $N_1 = N_2$, where $Ni$ denotes the number of grid points in domain $Ω_i$.

Since we know that the number of algorithm DDP iterations is not influenced by the mesh resolution, we are left with determining an appropriate relaxation parameter $θ^ψ$. The goal is, of course, to minimize the number of iterations needed to converge in algorithm DDP by a good choice of the relaxation parameter. For the model 1d problem, the analytical $θ^ψ_{max}$ from Lemma 3.2.2 is estimated to be

$$θ^ψ_{max} \geq 0.008.$$  (3.4.6)
We will see now that this estimate is too small; i.e. not only is there a larger maximum value such that the algorithm converges for all \( \theta \) between that value and 0, but the optimal \( \theta \) is larger, as well.

To estimate \( \theta_{\psi,\text{max}} \) computationally, we use estimates on the eigenvalues of the matrices associated to the discrete realization of \( K = K_1 + K_2 \). Denoting by \( e_i^- \) and \( e_i^+ \) the minimum and maximum eigenvalues corresponding to \( K_i \), respectively, we compute

\[
\theta_{\psi,h,\text{max}} \approx \frac{2(e_1^- e_2^-)^2(e_1^+ e_2^+)^2 + e_1^- e_2^- (e_1^+)^2 + (e_2^-)^2 + (e_1^-)^2 + (e_2^-)^2 + (e_1^+)^2 + (e_2^+)^2}{(e_1^+ e_2^+)^4(e_1^- + e_2^-)((e_1^-)^{-1} + (e_2^-)^{-1})^2(e_1^+ + e_2^+)^2}.
\]  

(3.4.7)

As \( h \to 0 \), we expect the estimates \( \theta_{\psi,h,\text{max}} \) to approach the true value for the continuous case. Figure 3.5 shows the convergence of the estimated values to \( \theta_{\psi,\text{max}} = 0.5 \) for the model problem. Figure 3.6 shows the reduction in the residual for various \( \theta \) values. We see here that convergence appears to break at \( \theta = 0.5 \), as predicted by the estimate. We see, as well, that the optimal value is below the maximum value, at \( \theta = 0.25 \).

In summary, algorithm DDP shows promising performance on model problems. Below we test it for true heterostructures.

<table>
<thead>
<tr>
<th>( N ) / ( h )</th>
<th>( L^2 ) error</th>
<th>Observed order</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 / 1e-02</td>
<td>2.32173e-06</td>
<td>–</td>
</tr>
<tr>
<td>400 / 5e-03</td>
<td>5.80366e-07</td>
<td>2.0002</td>
</tr>
<tr>
<td>600 / 3.33e-03</td>
<td>2.57998e-07</td>
<td>1.9995</td>
</tr>
<tr>
<td>800 / 2.5e-03</td>
<td>1.45047e-07</td>
<td>2.0018</td>
</tr>
<tr>
<td>1000 / 2e-03</td>
<td>9.2809e-08</td>
<td>2.001</td>
</tr>
</tbody>
</table>

TABLE 3.1: Algorithm DDP convergence.

<table>
<thead>
<tr>
<th>( N )</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

TABLE 3.2: Algorithm DDP mesh independence, \( \theta = 0.25 \).
FIGURE 3.5: Convergence of discrete estimate $\theta_{\text{max}}^{\psi,h}$ as $h \to 0$. $N_1 = N_2$ denotes the number of grid points in each domain.

FIGURE 3.6: Algorithm DDP residual reduction for various $\theta$. 

$\theta = 8.446 \times 10^{-5}$  
$\theta = 2.221 \times 10^{-2}$  
$\theta = 0.145$  
$\theta = 0.25$  
$\theta = 0.4$  
$\theta = 0.45$  
$\theta = 0.49$  
$\theta = 0.499$  
$\theta = 0.5$
3.4.2. Algorithm DDC

In this section we first perform convergence and mesh independence tests for algorithm DDC on a 1d model problem. We then address the practical choice of the relaxation parameter $\theta^u$. In all simulations we set the tolerance for convergence of algorithm DDC to $\tau = 10^{-12}$.

The 1d model problem we consider is,

$$-	riangle u_i = \frac{2x}{(x^2 + 1)^2}, \quad x \in \Omega_i, \quad (3.4.8)$$

$$\frac{\partial u_1}{\partial \nu} = u_1(0) - u_2(0), \quad \left[ \frac{\partial u}{\partial \nu} \right]_{\Gamma=\{0\}} = 0, \quad (3.4.9)$$

$$u_1(-1) = 1, \quad u_2(1) = 1, \quad (3.4.10)$$

designed similarly to the model problem for algorithm DDP. Figure 3.7 shows the solution to this problem, while Tabes 3.3 and 3.4 demonstrate order 2 convergence and mesh independence for a centered finite difference discretization with algorithm DDC.

To estimate $\theta^u_{max}$ we proceed similarly to algorithm DDP, denoting by $e^-_i$ and $e^+_i$
TABLE 3.3: Algorithm DDC convergence.

<table>
<thead>
<tr>
<th>N / h</th>
<th>L² error</th>
<th>Observed order</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 / 1e-02</td>
<td>1.72797e-05</td>
<td>–</td>
</tr>
<tr>
<td>400 / 5e-03</td>
<td>4.30769e-06</td>
<td>2.0041</td>
</tr>
<tr>
<td>600 / 3.33e-03</td>
<td>1.91255e-06</td>
<td>2.0026</td>
</tr>
<tr>
<td>800 / 2.5e-03</td>
<td>1.07507e-06</td>
<td>2.0024</td>
</tr>
<tr>
<td>1000 / 2e-03</td>
<td>6.87690e-07</td>
<td>2.0023</td>
</tr>
</tbody>
</table>

TABLE 3.4: Algorithm DDC mesh independence, θ = 0.335.

<table>
<thead>
<tr>
<th>N</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

As $h \rightarrow 0$, we expect the estimates $\theta_{max}^{u,h}$ to approach the true value for the continuous case. Figure 3.8 shows the convergence of the estimated values to $\theta_{max}^u = 0.635$. Figure 3.9 shows the reduction in residual for various $\theta$ values. We see here that convergence slows near the estimated value. This is less clear and dramatic than in the DDP case, though we note that the optimal value, $\theta = 0.335$, is clearly below the estimated value.

3.4.3. Algorithm DDN

As mentioned above, algorithm DDN originally proposed in Ref. [30] gave promising results for simple model problems such as (3.4.8)-(3.4.10). We see in Tables 3.5 and 3.6 that for the model problem algorithm DDN shows order 2 convergence, mesh independence, and generally low iteration counts.
FIGURE 3.8: Convergence of discrete estimate $\theta_{\text{max}}^{u,h}$ as $h \to 0$. $N_1 = N_2$ denotes the number of grid points in each domain.

FIGURE 3.9: Algorithm DDC residual reduction for various $\theta$. 

$\theta = 3.35 \times 10^{-5}$  \hspace{1cm} $\theta = 0.335$

$\theta = 3.35 \times 10^{-4}$  \hspace{1cm} $\theta = 0.35$

$\theta = 3.35 \times 10^{-3}$  \hspace{1cm} $\theta = 0.6$

$\theta = 3.35 \times 10^{-2}$  \hspace{1cm} $\theta = 0.65$
TABLE 3.5: Algorithm DDN convergence.

<table>
<thead>
<tr>
<th>N / h</th>
<th>L² error</th>
<th>Observed order</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 / 1e-02</td>
<td>2.4936e-05</td>
<td>–</td>
</tr>
<tr>
<td>400 / 5e-03</td>
<td>6.21489e-06</td>
<td>2.0044</td>
</tr>
<tr>
<td>600 / 3.33e-03</td>
<td>2.75907e-06</td>
<td>2.0028</td>
</tr>
<tr>
<td>800 / 2.5e-03</td>
<td>1.55086e-06</td>
<td>2.0025</td>
</tr>
<tr>
<td>1000 / 2e-03</td>
<td>9.92019e-07</td>
<td>2.0024</td>
</tr>
</tbody>
</table>

TABLE 3.6: Algorithm DDN mesh independence, \( \theta = 0.25 \).

<table>
<thead>
<tr>
<th>N</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

However, algorithm DDN has tremendous difficulties when data more typical of a true heterostructure is used in the model. To see this we compare algorithms DDN and DDC in a real heterostructure simulation. Specifically, we use transmission conditions accurate to a ZnS/Si heterostructure

\[
\frac{\partial u_1}{\partial \nu} = a_{n_1} u_1(0) - a_{n_2} u_2(0) \quad (3.4.12)
\]

where \( a_{n_1} = a_{n_2} = 1.20476 \times 10^{57} \). For this case algorithm DDC completes in 5 iterations with the relaxation parameter \( \theta = 6 \times 10^{56} \); the solution is shown in Figure 3.10.

While algorithm DDC handles large coefficients in the transmission condition by using a larger relaxation parameter, algorithm DDN requires just the opposite. Even with \( \theta = 1e - 16 \), i.e., machine zero, by the 2nd iteration the interface condition \( \lambda \) in algorithm DDN for this problem grows to machine infinity and the algorithm breaks. Thus, to make algorithm DDN productive for heterostructure problems further study is required. Perhaps scaling according to the transmission condition coefficient would be productive. However, other applications that may have unusual transmission conditions may also not
have the pathological data present in heterostructure semiconductors, and thus the more natural presentation of algorithm DDN over algorithm DDC may be preferred in these cases.

### 3.4.4. Heterostructure Semiconductors

Now that we have shown appropriate convergence and mesh independence for simple model problems for algorithms DDP and DDC, we turn our attention to the simulation of a full heterostructure. We will simulate a CdS/CIGS (Cadmium Sulfide / Copper Indium Gallium Selenide) heterostructure and examine the performance of algorithms DDP and DDC on this material. In the next section we will consider both the CdS/CIGS heterostructure and a ZnS/Si (Zinc Sulfide / Silicon) heterostructure in an uncertainty study for the interface parameter $\psi_{\Delta}$, coming from the lower scale DFT simulations. Tables 3.7 and 3.8 show the data for these heterostructures.

Figure 3.11 shows the primary variables $\psi$, $n$, and $p$ as well as the currents $J_n$, $J_p$, and $J = J_n + J_p$ resulting from simulation of the CdS/CIGS heterostructure. We see
<table>
<thead>
<tr>
<th>Material</th>
<th>Si</th>
<th>ZnS</th>
<th>CIGS</th>
<th>CdS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>11.9</td>
<td>8.4</td>
<td>13.5</td>
<td>10</td>
</tr>
<tr>
<td>$\chi$ (eV)</td>
<td>4.27</td>
<td>3.17</td>
<td>4.35</td>
<td>4.79</td>
</tr>
<tr>
<td>$E_g$ (eV)</td>
<td>1.12</td>
<td>3.54</td>
<td>1.16</td>
<td>2.42</td>
</tr>
<tr>
<td>$N_C$ (cm$^{-3}$)</td>
<td>2.83e19</td>
<td>4.3e18</td>
<td>2.2e18</td>
<td>2.2e18</td>
</tr>
<tr>
<td>$N_V$ (cm$^{-3}$)</td>
<td>1e19</td>
<td>6e19</td>
<td>1.8e19</td>
<td>1.8e19</td>
</tr>
<tr>
<td>$D_n$ (cm$^2$/s)</td>
<td>37.6</td>
<td>15.5</td>
<td>2.6</td>
<td>2.6</td>
</tr>
<tr>
<td>$D_p$ (cm$^2$/s)</td>
<td>12.9</td>
<td>1.0</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>SRH $\tau_n$ (s)</td>
<td>1e-07</td>
<td>1e-07</td>
<td>2e-09</td>
<td>1e-08</td>
</tr>
<tr>
<td>SRH $\tau_p$ (s)</td>
<td>1e-07</td>
<td>1e-07</td>
<td>1e-06</td>
<td>1e-12</td>
</tr>
<tr>
<td>$R_{dc}$ (cm$^3$/s)</td>
<td>1e-15</td>
<td>1e-10</td>
<td>1e-10</td>
<td>1e-10</td>
</tr>
</tbody>
</table>

**TABLE 3.7: Material parameter values**

<table>
<thead>
<tr>
<th>Interface</th>
<th>ZnS/Si</th>
<th>CdS/CIGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(A_n T^2)$ (A cm$^{-2}$)</td>
<td>3.34e06</td>
<td>2.13e06</td>
</tr>
<tr>
<td>$(A_p T^2)$ (A cm$^{-2}$)</td>
<td>5.86e06</td>
<td>8.67e06</td>
</tr>
<tr>
<td>$N_{T,n}$ (cm$^{-3}$)</td>
<td>5e15</td>
<td>-2e16</td>
</tr>
<tr>
<td>$N_{T,p}$ (cm$^{-3}$)</td>
<td>-1e17</td>
<td>2e17</td>
</tr>
<tr>
<td>$\Psi_\Delta$ (eV)</td>
<td>+0.76</td>
<td>+0.74</td>
</tr>
</tbody>
</table>

**TABLE 3.8: Interface and structure parameter values**
FIGURE 3.11: CdS/CIGS heterostructure with algorithms DDP and DDC.
Table 3.9: DDM iterations at each Gummel step for the CdS/CIGS heterostructure.

<table>
<thead>
<tr>
<th>Gummel Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ψ (DDP iterations)</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>u (DDC iterations)</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>v (DDC iterations)</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

good performance for the algorithms; Table 3.9 shows a modest number of iterations are needed at each Gummel iterations on the CdS/CIGS heterojunction.

3.4.5. Angstrom Scale Sensitivity Analysis

In this section we perform uncertainty analysis for the parameter $\psi_\Delta$, as described in Section 3.3. We simulate the model at the points associated to a 5-point Gaussian quadrature rule for the integral,

$$E[Q(\Psi_\Delta)] := \int_{\psi_\Delta-\vartheta}^{\psi_\Delta+\vartheta} Q(\Psi_\Delta) f(\psi_\Delta) d\psi_\Delta$$  

(3.4.13)

for both the ZnS/Si and CdS/CIGS heterojunctions.

Tables 3.10 and 3.11 list the collocation points associated to the 5-point Gaussian quadrature rule as well as the open circuit voltage $V_{oc}$, the short circuit current $J_{sc}$, and the maximum power $P_{max}$, resulting from a simulation of each material at the collocation points for the ZnS/Si and CdS/CIGS heterojunctions, respectively.

Next, from the results in Tables 3.10 and 3.11 we can compute the mean and variance for each distribution with

$$E[Q(\Psi_\Delta)] \approx \sum_m w_m Q(\psi_m).$$  

(3.4.14)

We present these results in Table 3.12. Overall, the current $J_{sc}$ appears less sensitive than $V_{oc}$, and tho CdS/CIGS heterojunction appears to be resilient to uncertainty at the atomic scale.
<table>
<thead>
<tr>
<th>Collocation Point</th>
<th>$J_{sc}$ (A/cm$^2$)</th>
<th>$V_{oc}$ (V)</th>
<th>$P_{max}$ (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.8958</td>
<td>2.192e-04</td>
<td>0.524</td>
<td>6.473e-05</td>
</tr>
<tr>
<td>-0.8408</td>
<td>1.496e-04</td>
<td>0.575</td>
<td>6.03e-05</td>
</tr>
<tr>
<td>-0.76</td>
<td>1.404e-04</td>
<td>0.655</td>
<td>6.851e-05</td>
</tr>
<tr>
<td>-0.6792</td>
<td>1.407e-04</td>
<td>0.736</td>
<td>7.831e-05</td>
</tr>
<tr>
<td>-0.6241</td>
<td>1.411e-04</td>
<td>0.791</td>
<td>8.487e-05</td>
</tr>
</tbody>
</table>

TABLE 3.10: Uncertainty study: ZnS/Si

<table>
<thead>
<tr>
<th>Collocation Point</th>
<th>$J_{sc}$ (A/cm$^2$)</th>
<th>$V_{oc}$ (V)</th>
<th>$P_{max}$ (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.8759</td>
<td>2.480e-02</td>
<td>0.640</td>
<td>1.234e-02</td>
</tr>
<tr>
<td>-0.8208</td>
<td>2.478e-02</td>
<td>0.640</td>
<td>1.239e-02</td>
</tr>
<tr>
<td>-0.74</td>
<td>2.475e-02</td>
<td>0.640</td>
<td>1.238e-02</td>
</tr>
<tr>
<td>-0.6592</td>
<td>2.471e-02</td>
<td>0.640</td>
<td>1.236e-02</td>
</tr>
<tr>
<td>-0.6041</td>
<td>2.467e-02</td>
<td>0.640</td>
<td>1.578e-02</td>
</tr>
</tbody>
</table>

TABLE 3.11: Uncertainty study: CdS/CIGS
### TABLE 3.12: Means and variances for heterojunction parameter uncertainty studies

<table>
<thead>
<tr>
<th>Structure</th>
<th>Mean $J_{sc}$</th>
<th>Var $J_{sc}$</th>
<th>Mean $V_{oc}$</th>
<th>Var $V_{oc}$</th>
<th>Mean $P_{max}$</th>
<th>Var $P_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZnS/Si</td>
<td>1.56e-04</td>
<td>8.5435e-10</td>
<td>0.6561</td>
<td>8.6e-03</td>
<td>7.0994e-05</td>
<td>7.2954e-11</td>
</tr>
<tr>
<td>CdS/CIGS</td>
<td>2.472e-02</td>
<td>1.9597e-09</td>
<td>0.64</td>
<td>1.6653e-16</td>
<td>1.296e-02</td>
<td>1.673989e-06</td>
</tr>
</tbody>
</table>

However, this may be misleading, as the ZnS/Si heterojunction displays meaningful differences in the quantities of interest when the heterojunction parameter is changed within the region of expected error. This is again reflected in Figures 3.12 and 3.13, which plot the current $J_{sc}$ against the voltage $V_{oc}$ for each quadrature point for the ZnS/Si and CdS/CIGS heterojunctions, respectively.
FIGURE 3.13: $J_{sc}$ against $V_{oc}$ for CdS/CIGS heterojunction at each collocation point
3.5. Conclusions

In this chapter a multiscale model for charge transport in semiconductor structures with heterojunctions was considered. The heterojunction model consists of a coupled system of partial differential equations posed in subdomains connected by unusual transmission conditions involving a jump discontinuity in electrostatic potential and Robin-like internal boundary conditions at the interface for the carrier densities, with data coming from angstrom scale density functional theory simulations. This setup lends itself naturally to a domain decomposition approach at the continuum scale.

We proposed novel iterative substructuring methods designed for these unusual transmission conditions, and performed convergence analysis for these methods. For the potential equation we presented a natural extension of the Neumann-Neumann algorithm to the transmission conditions in the heterostructure model, and additionally extended this method to include nonhomogeneous jumps in the flux. For the carrier densities, we presented two algorithms: a natural algorithm, DDN, which shows strong performance on model problems but faces difficulty when presented with the pathological data in a true heterostructure device and a less natural algorithm, DDC, which shows strong performance on both model problems and true heterostructure devices. We performed analysis on the relaxation parameters in the substructuring methods, and presented a computational technique for obtaining optimal values.

Additionally, we applied the domain decomposition formulation to the analysis of the component equations of the heterostructure model, thus presenting the first step towards a complete analysis of the heterostructure model. We discussed the practical importance of this analysis, and provided a road map for analysis of the full heterostructure model by fixed point analysis using estimates obtained in this work.

Finally, we performed sensitivity analysis for the data from the DFT simulations in the continuum scale model.
The results in this work are promising. The algorithms show very strong performance for difficult heterostructure problems. However, the application motivating this work was effectively 1d, and therefore the simulations in this work were restricted to 1d problems. Certainly other applications of the heterostructure model require investigations in 2d and 3d. In future work the methods developed here will be implemented in higher dimension and on time-dependent problems. Additionally, attention was restricted to the case where there was single heterojunction interface. In future work we would like to consider several interfaces, as well as the mixture of a single, physics based heterjunction interface with several artificial homojunction interfaces to allow for scaling these methods in an high performance computing environment.
4. MODELING FLUID FLOW IN COMPLEX MEDIA

Understanding fluid flow in a complex media is an area of vital importance to future energy resources as well as to the health and safety of human beings. The extraction of energy resources as well as the need to protect drinking water from contaminants both require a firm understanding of how fluid flows in the subsurface. Computational simulations provide a method for exploring fluid flows in complex media at a feasible cost, e.g., to investigate the potential impact of energy exploration on water resources, to investigate novel methods of extracting resources from the subsurface, or to understand how and where non-traditional forms of subsurface energy resources form.

Historically, due to the prohibitive cost of resolving subsurface geometries at the porescale, subsurface flow has been modeled with Darcy’s Law, which averages the geometry into a hydraulic permeability tensor describing the capacity for fluid to travel through the porous medium. However, as computational resources have become more capable, there has been an increasing interest in considering porescale models, either in their own right, or as tools to enhance the precision of models at the Darcy scale. Our work falls into the latter category. At the porescale, fluid flow is modeled by the Navier-Stokes equations, or the Stokes equations in the laminar regime. These models require that the geometry is explicitly resolved, resulting in an enormous computational burden if any reasonable expanse of the subsurface is to be simulated simultaneously. An increasingly popular alternative are the so-called pore-network models, which approximate the geometry as a collection of pores and throats. This models flow at the pore scale while easing the burden of resolving the porous medium geometry.

The situation becomes more complicated when one expects the geometry of the subsurface to evolve dynamically throughout the simulation. This geometric evolution may be due to, e.g., the growth of biofilm, the formulation of methane hydrate, or precipitation-
dissolution phenomena. For example, subsurface sequestration of supercritical CO$_2$ is an important area of research for mitigating the effects of carbon based energy consumption. For this method to be effective, the potential for CO$_2$ to escape back into the atmosphere must be understood and mitigated. Recently it has been proposed that biofilm may be engineered to reduce the permeability of the subsurface and thus mitigate the potential for sequestered CO$_2$ to escape [69]. Precipitation-dissolution phenomena are also crucial in modeling geothermal energy resources.

We see, then, that there is a dynamic coupling between the porescale and the corescale, as one expects the averaged geometry, i.e. the hydraulic permeability tensor, to change as the porescale geometry evolves. Thus a computational investigation of flow and transport processes in an evolving geometry requires a methodology for updating the hydraulic permeability as the porescale geometry evolves. To make matters worse, the appropriate time scale for flow and transport simulations in porous media is on the order of 1 second or less, while the integration time to study processes like biofilm growth or methane hydrate formulation are on the order of many days or more. This problem, – wedged between constraints of computational feasibility and active processes at the porescale – necessarily requires a novel multiscale strategy.

In this work we consider a reduced order, three-scale numerical method that leverages the computational advantages of Darcy’s law and pore-network models with the accuracy of confined porescale flow models, i.e., the Stokes’ equations, to communicate the evolution of the hydraulic permeability from the porescale. The core of our strategy is the subdivision of the computational domain followed by the offline computation of locally defined empirical probability distributions relating the absolute permeability to the volume fraction of material, e.g. hydrate or biofilm, in the void space of each subdomain. The online simulation then informs a pore-network model by sampling the appropriate empirical distributions to obtain the absolute permeability for each throat. Finally, the
pore-network model is solved to produce a full domain permeability appropriate for simulation at the Darcy scale.

Our methodology at the porescale is a reduced order model, where we divide the work into offline and online stages. A key component of this strategy is the formulation of a novel representation of stochastic geometries in the Stokes’ equations. This model utilizes a stochastic resistive term similar to an immersed boundary model for the growth of material in the fluid domain, which allows for efficient sampling of the stochastic process describing material growth, as well as facilitates the generation of realizations that are accurate to relevant physics through, e.g., optimization based principal component analysis [109]. In practice, we anticipate the offline approximated stochastic permeability replacing the physically unreasonable Hague-Poiseuille assumption for flow and transport models in pore-networks, thus allowing a full transient simulation of flow and transport, accurate to the porescale, without resorting to an online porescale simulation. Further, we demonstrate that with porescale accurate permeabilities a pore-network model may be constructed with no concern for the underlying porous structure, while still resulting in accurate macroscale properties.

The rest of this paper is organized as follows. In Section 4.1. we review porescale and corescale fluid flow models, as well as the connections between them. In Section 4.2. we introduce a stochastic immersed boundary Stokes’ model. In Section 4.3. we describe our method in detail. In Section 4.4. we introduce a free, open source software package, HybGe-Flow3D that we developed for simulating flow in complex, hybrid geometries. In Section 4.5. we present numerical results. Finally in Section 4.7. we present concluding remarks.
4.1. Fluid Flow Models

In this section we review several models for fluid flow in a complex media: the Stokes’ equations, Darcy’s law, and the pore-network model. In each model we are concerned with the relationship between two intensive properties, the fluid’s velocity and the fluid’s pressure. At all scales we consider we assume the fluid is a continuum; i.e. the fluid is a continuous substance rather than a collection of discrete particles. We distinguish between microscopic velocity and pressure, \((u, p)\), defined at the porescale in the void space of the geometry, and macroscopic velocity and pressure \((U, P)\), defined at the corescale on the total geometry consisting of void space and the solid matrix.

4.1.1. Fluid Flow Models

Throughout this section we denote by \(D\) a bounded open subset of \(\mathbb{R}^d\), \(d \in \{2, 3\}\), representing a porous medium. We denote by \(D_f\) the void space, or fluid subdomain and by \(D_s\) the ‘rock’ subdomain, such that \(D_f \cup D_s = D\). Additionally we have the interface \(\Gamma = D_f \cap D_s\).

4.1.1.1 Darcy’s Law

In the mid 19th century, Henri Darcy famously investigated, experimentally, the flow of water through saturated, homogeneous sand columns for the water of the city’s fountains. From his investigations, the empirical law,

\[ q = -K \cdot \nabla h, \quad \text{in } D, \]

was derived. Here \(q\) is the specific discharge, defined as the volume of water passing through a unit area of porous medium cross-section, in the normal direction, per unit time. \(K \in \mathbb{R}^{d \times d}\) denotes the hydraulic conductivity, and describes the potential for fluid flow through the porous medium. \(K_{ij}\) may be viewed as the contribution to the specific
discharge in the $i$-th direction produced by a unit component of the hydraulic gradient in the $j$-th direction. Finally, $h$ denotes the piezometric head, defined as

$$h = z + \frac{P}{\rho g},$$

where $z$ denotes elevation, $P$ and $\rho$ denote the fluid’s pressure and mass, respectively, and $g$ denotes acceleration due to gravity.

In this work, for simplicity we neglect gravity and consider the flow of an incompressible fluid, thus arriving at the common presentation of Darcy’s law for fluid flow through a porous medium,

$$U = -K \cdot \nabla P, \quad \text{in } D.$$  \hspace{1cm} (4.1.3)

Here $U : D \to \mathbb{R}^d$ denotes the fluid velocity, $P : D \to \mathbb{R}$ denotes the fluid pressure, and $K = \frac{\kappa}{\mu}$ denotes the hydraulic conductivity, or permeability. $\kappa \in \mathbb{R}^{d \times d}$ denotes the absolute permeability, which in the laminar regime depends only on the geometry, and $\mu \in \mathbb{R}$ denotes the fluid viscosity.

We emphasize that $P$ and $U$ are the macroscopic velocity and pressure, defined on the entire domain $D$, including the solid matrix, $D_s$. $K$, in turn, averages the porescale geometry by describing the ability of the fluid to flow through the medium. This averaging enables numerical simulations of regions of a porous medium that would be impossible if one had to resolve the void space explicitly.

While Darcy’s law was originally derived experimentally, it is interesting to note that this empirical law can also be derived through rigorous mathematical homogenization of the momentum balance equation, and also by volume averaging techniques [7].
4.1.1.2 Stokes’ Equations

At the porescale, in the void space $D_f$ the pressure and velocity of the fluid are described by the Navier-Stokes equations,

\begin{align}
  \rho u \cdot \nabla u - \mu \Delta u + \nabla p &= 0, \quad \text{in } D_f, \\
  \nabla \cdot u &= 0, \quad \text{in } D_f.
\end{align}

(4.1.4)

(4.1.5)

Here we have neglected volume forces, and in particular, ignored gravity. Here $\mu \in \mathbb{R}$ is the fluid’s viscosity. As mentioned previously, the porescale velocity and pressure $u : D_f \to \mathbb{R}^3$ and $p : D_f \to \mathbb{R}$ are defined in the void space, and the geometry $D_f$ must be explicitly resolved in numerical simulations.

We will be interested in flows in the laminar regime. At low velocities, the Navier-Stokes equations are well approximated by the Stokes’ equations,

\begin{align}
  -\mu \Delta u + \nabla p &= 0, \quad \text{in } D_f, \\
  \nabla \cdot u &= 0, \quad \text{in } D_f.
\end{align}

(4.1.6)

(4.1.7)

As boundary conditions we enforce a combination of no-slip conditions,

\begin{align}
  u &= 0, \quad \text{in } \Gamma_s, \\
\end{align}

(4.1.8)

inflow conditions,

\begin{align}
  u &= u_D, \quad \text{in } \Gamma_{in},
\end{align}

(4.1.9)

and outflow conditions,

\begin{align}
  \nabla u \cdot n &= 0, \quad \text{in } \Gamma_{out},
\end{align}

(4.1.10)

where $n$ is the unit outward normal to the void space $D_f$ along $\Gamma$. The specifics of the boundary segments will be made clear in Section 4.1.2.1.
4.1.1.3 Pore-Network Models

Pore-Network models, as the name suggests, are models that represent the porous medium as a network of pores. The connections between pores are called throats. The distribution of pores, as well as their volumes, connectivity, and the permeability assigned to each throat are the major elements of a pore-network description of a porous medium. Figure 4.1 illustrates the description of a simple 2d porous medium and a corresponding pore-network model. Of course for a realistic porous geometry the pore-network model is much more complicated. Figure 4.2 shows the solution of a pore-network representation of a sandstone geometry. In the sandstone image, the pore locations were computed by Ken Kennedy using 3DMA-Rock, courtesy of Masa Prodanovic [60]. However, in sandstone the throat and pore volumes are very similar, and 3DMA-Rock did not produce a reasonable network of throats. To overcome this, we define the mean coordination number for sandstone $\sigma$. We then find the $2\sigma$ nearest pores for each pore, and randomly select approximately $\sigma$ connected pores.

Assume the porous medium $D$ is represented by $N_p$ pores, $\{P_i\}_{i=1}^{N_p}$. For each pore
$P_i$, we denote by $I_i$ the collection of indices of pores connected to pore $P_i$. For each $j \in I_i$, we denote by $T_{ij}$ the throat connecting pore $P_i$ and pore $P_j$. We denote by $p_i$ the pressure in pore $P_i$, and by $c_{ij}$ the hydraulic conductivity in the throat connecting pores $P_i$ and $P_j$; i.e. $c_{ij}$ denotes the permeability in throat $T_{ij}$. In the simplest case of a single component, single phase flow, the pore-network model is given by,

$$\sum_{j \in I_i} c_{ij}(p_i - p_j) = 0, \quad \forall i. \quad (4.1.11)$$

Equation (4.1.11) must be completed by appropriate boundary conditions. In this work we prescribe a Dirichlet condition on the pressure on inflow and outflow pores, thus enforcing a pressure drop across the domain. We also impose a periodic boundary condition for the
pressure along remaining boundary pores.

If we assume the location of pores and the list of connecting throats are a good approximation of the porous medium, then the accuracy of the model (4.1.11) is determined by the accuracy of the permeability coefficients \{c_{ij}\}_{1 \leq i \leq N_p, j \in I_i}, and their ability to encompass the fine geometric details about the local pore geometry. A common approach for calculating these coefficients is to assume a Hagen-Poiseuille flow in the throats. Independently derived, experimentally, by Hagen in 1839 and Poiseuille in 1838, the Hagen-Poiseuille equation gives the pressure drop in an incompressible, Newtonian fluid in the laminar regime flowing through a cylindrical pipe of constant cross-sectional area,

\[ \Delta p = \frac{8\mu LQ}{\pi r^4}. \] (4.1.12)

Here \( \Delta p = p_i - p_j \) denotes the pressure loss across the throat, \( L \) the length of the throat, \( Q \) the volumetric flow-rate in the throat, and \( r \) the radius of the throat. To determine the permeability coefficient \( c_{ij} \), a pressure drop is prescribed along the throat, and \( Q \) is computed from the formula (4.1.12). Then by Darcy’s law,

\[ c_{ij} = \frac{A \Delta p}{\mu L Q}, \] (4.1.13)

where \( A \) denotes the cross-sectional area of the throat, which was assumed constant.

While the Hagen-Poiseuille flow has the advantage of being simple and easy to calculate, the assumption that the geometry of a porous medium is accurately and completely described by a collection of throats and pores of constant cross-sectional area is extremely unrealistic. Additionally since we are interested in the evolution of the geometry driven by a reactive transport process it is too restrictive to assume that the geometry evolves only by a reduction in a constant cross-sectional area in the throats.

4.1.2. Multiscale Fluid Flow

As may be clear already, a key to connecting flow models at the various scales is the permeability. In this section we review how to compute a permeability for a domain using
the Stokes’ equations (4.1.6), as well as how to compute a permeability for a domain from the solution to the pore-network model (4.1.11).

4.1.2.1 Permeability from the Porescale

Because we are interested in improving upon the Hagen-Poiseuille throat conductivities for the pore-network model, we need to make a distinction between the permeability tensor $\kappa$ discussed above and the permeability in a particular direction, $\kappa_{\text{dir}}$. Clearly, the full tensor in Darcy’s law (4.1.3) makes little sense for a pipe or throat between pores, as fluid can only flow, macroscopically, in the direction defined by the throat. To compute $\kappa_{\text{dir}}$ we employ a volume averaging technique. We note that volume averaging can be used to compute the full tensor $\kappa$, but since that is not relevant to throat conductivities, we point the reader to papers by Peszynska et al. describing a volume averaging technique for upscaled conductivity tensors [85, 86, 87].

Suppose we are interested in the conductivity $\kappa_{xx}$ describing the capacity for fluid to flow through a porous medium $D$ defined by a void space $D_f$ and a solid matrix $D_s$ in the direction defined by the $x$-axis. We will describe the fluid flow in the void space by the Stokes’ equations (4.1.6), with boundary conditions appropriate to induce flow in the $x$-direction. Define $\underline{x} = \{(x,y,z) \in D : x \leq \hat{x} \ \forall (\hat{x}, \hat{y}, \hat{z}) \in D\}$, and $\underline{\bar{x}} = \{(x,y,z) \in D : x \geq \hat{x} \ \forall (\hat{x}, \hat{y}, \hat{z}) \in D\}$. We enforce the inflow condition (4.1.9) on $\Gamma_{\text{in}} = \Gamma \cap \underline{x}$, the outflow condition (4.1.10) on $\Gamma_{\text{out}} = \Gamma \cap \underline{\bar{x}}$, and the no-slip condition (4.1.8) on the remaining boundary $\Gamma_s = \Gamma \setminus (\Gamma_{\text{in}} \cup \Gamma_{\text{out}})$. An example in 2D is given in Figure 4.3.

![Figure 4.3: Illustration of geometry in 2d.](image)

To compute the permeability $\kappa_{xx}$ we consider Darcy’s law (4.1.3), and note that
\( \kappa_{xx} \) is determined by a macroscopic velocity in the \( x \)-direction, \( U_x \), and the \( x \)-component of the macroscopic pressure gradient, \( \nabla P \), along with the fluid property, \( \mu \). However, solving the Stokes’ equations at porescale produces microscopic velocity and pressure \((u, p)\). Thus computing an upscaled permeability requires a method for obtaining the macroscopic properties \((U, P)\) from the microscopic properties \((u, p)\). To compute \((U, P)\) from \((u, p)\) we employ a volume averaging technique. To avoid possible recirculation, we will compute averages in a box \( B \subset D \), whose boundary is a small distance from the boundary \( \partial D \). To compute the pressure gradient, we split the box \( B \) into two halves, \( B_L \) and \( B_R \), where the interface \( \overline{B_L} \cap \overline{B_R} \) is defined by the midpoint in the \( x \)-direction of the domain. An illustration of these subdivisions in 2D is given in Figure 4.4.

![Figure 4.4: Illustration of volume averaging in 2d.](image)

We then compute the average \( x \)-component of the velocity in the box \( B \),

\[
\langle u_x \rangle = \frac{1}{|B \cap D_f|} \int_{B \cap D_f} U_x. \tag{4.1.14}
\]

and the average pressure in each sub-box,

\[
\langle p \rangle_L = \frac{1}{|B_L \cap D_f|} \int_{B_L \cap D_f} P, \quad \langle p \rangle_R = \frac{1}{|B_R \cap D_f|} \int_{B_R \cap D_f} P. \tag{4.1.15}
\]

We define the center of mass of the two boxes \( B_L \) and \( B_R \),

\[
c_L = \frac{1}{|B_L|} \int_{B_L} x \, dx, \quad c_R = \frac{1}{|B_R|} \int_{B_R} x \, dx. \tag{4.1.16}
\]

Then we have the volume-averaged \( x \)-component of the pressure gradient,

\[
\nabla_x \hat{P} = \frac{\langle p \rangle_R - \langle p \rangle_L}{|c_R - c_L|}. \tag{4.1.17}
\]
Next we recall that while $U$ is defined everywhere on $D$, the microscopic velocity $u$ is only defined in the void space. To account for this in our average, we multiply $\langle u_x \rangle$ by the porosity,

$$\phi = \frac{|D_f|}{|D|} \quad (4.1.18)$$

and obtain the volume averaged $x$-component of the velocity,

$$\hat{U}_x = \phi \langle u_x \rangle. \quad (4.1.19)$$

Then from Darcy’s law we have

$$\kappa_{xx} = -\mu \frac{\hat{U}_x}{\nabla_x \hat{P}}. \quad (4.1.20)$$

### 4.1.2.2 Permeability from the Pore-Network

Previously we reviewed how the permeability in the principle flow direction is computed from the microscopic velocity and pressure $(u, p)$. Next we review how the permeability is computed from the solution to the pore-network model (4.1.11) following [21].

We begin by distinguishing the boundary pores in the pore-network representation of the porous medium $D$. Since we are interested in the permeability in the $x$-axis direction, we define $I_L$ as the collection of indices of pores along the left face of the cubic volume $D$ and $I_R$ the collection of indices of pores along the right face. We enforce a pressure drop across the domain by assigning a Dirichlet condition $P_L$ on nodes $I_L$ and a Dirichlet condition $P_R$ on $I_R$. Remaining boundary pores (along the top, bottom, front, and back faces of the cubic volume in 3d, or along the top and bottom of the volume in 2d) are prescribed periodic boundary conditions.

By conservation of mass the total flux through the left face of the volume is equal to the total flux through the right face. Denoting this flux by $f$, we have

$$f = \sum_{i \in I_L} \sum_{j \in I_i} c_{ij} (p_i - p_j) = -\sum_{i \in I_R} \sum_{j \in I_i} c_{ij} (p_i - p_j). \quad (4.1.21)$$
When there is no source or sink present, we expect the minimum and maximum pressure to be obtained on the boundary of the network. This implies (details in [21]) the existence of a nonnegative function $q$ such that

$$f = -(P_R - P_L)q(P_L, P_R)$$  \hspace{1cm} (4.1.22)

when $f$ is differentiable with respect to $P_R$ and $P_L$. If the throat permeabilities $c_{ij}$ are independent of the boundary conditions and the pressures in the pores, the system (4.1.11) is always solvable, and the function $q(P_L, P_R)$ can be expressed as a constant,

$$q = \frac{k}{L}$$  \hspace{1cm} (4.1.23)

where $L$ is the distance between the left and right faces. The absolute permeability $\kappa$ (in the principle flow direction), is then given by

$$\kappa = \frac{qL}{A},$$  \hspace{1cm} (4.1.24)

where $A$ is the area of the left-side face of the volume.

We note that a more general method due to Peszynska and Trykozko [86] allows for anisotropic values of $\kappa$. 


4.2. Fluid Flow in a Stochastic Geometry

A key component of our method is the offline computation of empirical distribution functions describing the relationship between the volume of growth material in the void space of a porous medium and the upscaled absolute permeability. Thus we assume we are given a porous geometry, $D_0$, with fluid subdomain $D_{f,0}$, rock subdomain $D_{s,0}$, and we define $\Gamma_0 = \overline{D_{f,0}} \cap \overline{D_{s,0}}$. We are interested in modeling the impact of the stochastic growth of material within the void space on the absolute permeability. The natural setting for this problem is the Stokes’ equations posed on a stochastic geometry.

Let $(\Omega, \mathcal{F}, P)$ be a complete probability space, and we consider,

$$-\mu \Delta u + \nabla p = 0, \quad \text{on } D_f(\omega), \ \omega \in \Omega. \quad (4.2.1)$$

Here $D_f(\omega)$ denotes a realization of the void space with the growth of material. In this work we restrict our consideration to applications where $D_f(\omega) \subset D_{f,0}$, i.e. the void space in a sample of the stochastic geometry represents a restriction of the void space in the base geometry. Applications in which the void space ‘grows,’ e.g. hydraulic fracture of the subdomain $D_{s,0}$, are beyond our current investigations.

We denote by $D_g(\omega)$ the growth domain for realization $\omega$, i.e. the portion of $D_{f,0}$ upon which material has grown. Thus we have, for each $\omega \in \Omega$, $D_{f,0} = D_g(\omega) \cup D_f(\omega)$.

While intuitively simple, this model is computationally impractical. For each realization $\omega \in \Omega$ one must remesh the entire domain, and it is not at all obvious how one would parameterize the stochastic process describing the growth of the material. Thus we require a different representation of the stochastic geometry more suitable to efficient computation and parameterization.
4.2.1. Literature Remarks

Research in modeling of partial differential equations on stochastic geometries is a relatively young field, with most work appearing in the last 15 years. Methods in this area can loosely be placed into three categories; domain mapping [115, 116], perturbation methods [41, 42], and fictitious domain methods [16, 40]. However, each of these methods have short-comings for our application. Both domain mapping and perturbation methods rely on continuity of the map $\Gamma_0 \mapsto \Gamma(\omega)$; i.e. the boundary is a continuous deformation of the original boundary. We make no such assumption, a priori, on the growth pattern of the material in the void space. For biofilm growth this assumption may be somewhat realistic, as film seems to grow attached to the rock wall in the subsurface. However, in other applications, e.g. methane hydrate formulation, there is evidence that hydrate forms fully within the void space and without contact to the solid matrix [111]. Thus realizations of the geometry cannot, in general, be expected to be continuous deformations of the boundary $\Gamma_0$.

Fictitious domain methods, while not requiring a continuous deformation of the boundary, enforce the geometry through a Lagrange multiplier the represents a surface penalty along the new material’s boundary. These methods have two drawbacks: (1) it has been shown, in the deterministic context, that surface integral approaches are less robust at high flow rates than volume penalty approaches, and (2) fictitious domain methods do not allow for an immersed object with positive permeability, thus restricting the allowable physics for the application.

To summarize, we seek a model that satisfies the following:

1. There should be no remeshing required for computational sampling of $\Omega$, thus facilitating efficient investigations when many simulations are necessary.

2. Discontinuity of the map $\Gamma_0 \mapsto \Gamma(\omega)$ should be allowed, thus placing no restriction on the growth pattern of material.
3. The model should be tunable in the sense that permeability in obstructed subdomains should be controlled by a parameter, thus allowing for more general material physics.

In the next section we describe a model that enforces material growth through a stochastic resistive term, similar to an immersed boundary method [70, 78].

4.2.2. Stochastic Immersed Boundary Stokes Model

In this work we use the following novel stochastic immersed boundary Stokes model,

\[-\mu \nabla \cdot \nabla u + Z_x(\omega)u + \nabla p = 0, \quad \text{on } D_{f,0}, \, \omega \in \Omega, \quad (4.2.2)\]

\[\nabla \cdot u = 0, \quad \text{on } D_{f,0}, \quad (4.2.3)\]

\[u = u_D, \quad \text{on } \Gamma_{D,0}, \quad (4.2.4)\]

\[\mu \frac{\partial u}{\partial n} - np = 0, \quad \text{on } \Gamma_{N,0}. \quad (4.2.5)\]

Here \(Z\) is a stochastic process defined by,

\[Z_x(\omega) = \begin{cases} 
0, & x \in D_f(\omega), \\
\eta^{-1}, & x \in D_g(\omega).
\end{cases} \quad (4.2.6)\]

Here \(\eta > 0\) is a penalization term. In the simplest case where one seeks to stop fluid from flowing through \(D_g(\omega)\) entirely, we may consider \(Z\) as a binary field multiplied by the penalization parameter,

\[Z_x(\omega) = \eta^{-1} I_{D_g(\omega)}, \quad (4.2.7)\]

where

\[I_{D_g(\omega)} = \begin{cases} 
0, & x \in D_f(\omega), \\
1, & x \in D_g(\omega).
\end{cases} \quad (4.2.8)\]

However, the presentation in (4.2.2) is more general, and in particular the model allows \(\eta\) to be spatially dependent in order to enforce permeability appropriate to the different physics associated to different material growths in the void space.
In the simplest case, $\eta$ is taken constant and very close to 0 to stop fluid from flowing in $D_g(\omega)$. To see how this works, we compute an a priori estimate for the velocity in the growth domain using the pressure-free weak formulation.

Define

$$V := H^1_0(D_{f,0})^3.$$  \hfill (4.2.9)

$$V_0 := \{ v \in V \mid \nabla \cdot v = 0 \}.$$ \hfill (4.2.10)

**Proposition 4.2.1.** Fix $\omega \in \Omega$, let $f \in V'$, and let $u_\eta$ be the solution of the problem,

$$\text{find } u_\eta \in V_0 \text{ s.t. } \int_{D_{f,0}} \mu \nabla u_\eta \cdot \nabla v + \int_{D_{f,0}} \eta^{-1} Z_x(\omega) u_\eta v = \int_{D_{f,0}} f v, \, \forall v \in V_0.$$ \hfill (4.2.11)

As $\eta \to 0$ we have $\|u_\eta\|_{L^2(D_g(\omega))} \to 0$.

**Proof.** First, test (4.2.11) against the solution $u_\eta$ to obtain,

$$\mu \|\nabla u_\eta\|_{L^2(D_{f,0})}^2 + \eta^{-1} \|u_\eta\|_{L^2(D_g(\omega))}^2 = \int_{D_{f,0}} f u_\eta. \quad (4.2.12)$$

Dropping the first term on the left hand side and applying Hölder’s Inequality to the right hand side, we obtain

$$\eta^{-1} \|u_\eta\|_{L^2(D_g(\omega))}^2 \leq \|f\|_{L^2(D_{f,0})} \|u_\eta\|_{L^2(D_{f,0})}.$$ \hfill (4.2.13)

Returning to (4.2.12), dropping the second term on the left hand side and applying Hölder’s Inequality to the right hand side, we obtain

$$\mu \|\nabla u_\eta\|_{L^2(D_{f,0})}^2 \leq \|f\|_{L^2(D_{f,0})} \|u_\eta\|_{L^2(D_{f,0})}.$$ \hfill (4.2.14)

Applying the Poincaré Inequality and denoting by $C$ the Poincaré constant, we obtain,

$$\|u_\eta\|_{L^2(D_{f,0})} \leq \frac{C^2}{\mu} \|f\|_{L^2(D_{f,0})}.$$ \hfill (4.2.15)
Inserting (4.2.15) into the right side of (4.2.13), we have,
\[ \|u_\eta\|_{L^2(D_g(\omega))} \leq C \left( \frac{\eta}{\mu} \right)^{\frac{1}{2}} \|f\|_{L^2(D_{f,0})}. \] (4.2.16)

In the simulations in this work we set \( \eta \) to be constant and enforce no flow through the obstacles. In this setting the stochastic process \( Z \) is a continuous-parameter, discrete-state stochastic process with index set \( D_{f,0} \subset \mathbb{R}^3 \) and state space \( \{0, \eta^{-1}\} \). For a fixed \( \omega \in \Omega \), the sample path of the stochastic process \( Z(\cdot, \omega) : D_{f,0} \rightarrow \{0, \eta^{-1}\} \) completely describes the realization of the geometry \( D_g(\omega) \). Of course for numerical simulation we must approximate \( Z(\cdot, \omega) \) at finitely many index values. Let \( \{D_i\}_{i=1}^N \) be the pressure cells in a staggered-grid finite volume discretization of the domain \( D_{f,0} \), and let \( \{d_i\}_{i=1}^N \) be the associated cell centers. We approximate \( Z \) by the discrete-parameter, discrete-state stochastic process
\[
Z_i(\omega) = \begin{cases} 
0, & d_i \in D_f(\omega), \\
\eta^{-1}, & d_i \in D_g(\omega). 
\end{cases}
\] (4.2.17)

We now have the index set \( I = \{1, \ldots, N\} \). Now for a fixed \( \omega \) the sample path of the process \( \{Z_i(\omega)\}_{i=1}^N \) describes the discrete realization of the geometry. In the linear system resulting from discretization \( Z_i(\omega) \) is given by a diagonal matrix with \( Z_i(\omega) \) along the diagonal.

We are ultimately interested in modeling the impact of material growth on the quantity of interest, the permeability \( \kappa \). Clearly \( \kappa \) depends on \( \omega \in \Omega \). Moreover, we parameterize \( \kappa \) by the volume fraction of material growth,
\[
V(\omega) = \frac{|D_g(\omega)|}{|D_{f,0}|}. \] (4.2.18)

Then for each \( v \in [0, 1] \), \{\( \kappa_v(\omega); \omega \in \Omega, V(\omega) = v \} \) is a random vector describing the permeability for realizations \( \omega \in \Omega \) satisfying \( V(\omega) = v \). Thus \( \kappa_v(\omega) \) is a conditional
random vector describing the permeability of the domain for samples with fixed volume fraction of material growth. In the next section we will describe how we discretize the index set \([0, 1]\) appropriate for numerical simulation.
4.3. Three-scale Fluid Flow Methods

In this section we describe our Reduced Order Three-scale Fluid Flow Method. The method is based on the subdivision of the porescale geometry and the independent computation of permeability distributions on each subdomain. These permeabilities are then used in a pore-network model for the throat conductivities. The solution of the pore-network model then communicates the macroscale permeability to the Darcy scale.

We begin by considering a three-scale method with no stochastic component. The direct method will be useful for illustration, but we will see that an expensive – albeit cheaper than solving the porescale problem as a whole – porescale simulation is required in real time as the geometry evolves. In the direct method, we solve the porescale model in each subdomain in a deterministic setting and use the resulting permeabilities as throat conductivities for the pore-network model. Despite the reduced complexity stemming from the subdivision of the domain, for many realistic problems this will still be computationally intractable. Thus after reviewing the method in the deterministic setting we will introduce our reduced order modeling approach which approximates, in an offline stage, the conditional random vector $\kappa_\omega(\omega)$, which can then be sampled in an online stage without requiring new flow solutions at porescale.

4.3.1. Direct Three-scale Method

Given a porous geometry $D$, we begin by performing a structured subdivision of the domain into $N_X \times N_Y \times N_Z = N_p \times N_p$ subdomains. For $j \in \{1, \ldots, N_X\}$, $k \in \{1, \ldots, N_Y\}$, and $l \in \{1, \ldots, N_Z\}$ we denote by $D^{jkl} \subset D$ subdomain $jkl$ ($D^{jk}$ in $\mathbb{R}^2$). We prefer to have a single index to run through the subdomains. So, let $i$ denote the standard lexographical ordering of the indices $jkl$ or $jk$. 
We then define
\[ P_i = \frac{1}{|D_i|} \int_{D_i} x \, dx \]  
(4.3.1)
so that the pore location for pore \( i \) is the center of mass of subdomain \( i \). We then have
\[ I_i = \left\{ j : m^* \left( D_j \cap D_i \right) > 0 \right\}, \]  
(4.3.2)
where \( m^* : \mathbb{R}^{d-1} \rightarrow \mathbb{R} \) denotes the \((d-1)\)-dimensional Lebesgue measure.

Denote
\[ P = \left\{ \{P_i, I_i\}_{i=1}^{N_p}, T \right\}, \]  
(4.3.3)
where
\[ T = \{T_{ij} \mid i < j, \ j \in I_i\}. \]  
(4.3.4)
In addition, we define
\[ C = \{c_{ij}\}_{j \in I_i}. \]  
(4.3.5)
Here \( c_{ij} \) denotes the permeability describing fluid flow from pore \( i \) to pore \( j \). That is, \( P \) consists of the collection of pore locations, pore connectivities and throats, and \( C \) is the collection of throat connectivities for the pore-network description of \( D \).

**Remark 4.3.1.** In a typical pore-network model, where the Hagen-Poiseuille, or some other analytical model for permeability in the throat is used, one would typically need to compute and store pore volumes, throat radii, and throat length in addition to the data listed above. Because we will rely on direct porescale simulation in subdomains to determine (either directly or by sampling \( \kappa_\omega(\omega) \)) the permeabilities, we do not need this extra information.

The top image in Figure 4.5 illustrates a 4x4 subdivision for a 2d domain. As we see here, the pore locations described thus far (pores 1-16 in Figure 4.5) do not include
boundary pores. To properly set boundary information, we introduce additional pores. Define the sets

\[ I_{in} = \{ i : m^* (D^i \cap \Gamma_{in}) > 0 \}, \quad (4.3.6) \]
\[ I_{out} = \{ i : m^* (D^i \cap \Gamma_{out}) > 0 \}. \quad (4.3.7) \]

Then, for each \( i \in I_{in} \) we define the inflow pore

\[ P^I_i = \arg \min_{x \in \Gamma_{in}} \text{dist} (P_i, x), \quad (4.3.8) \]

and for each \( i \in I_{out} \) we define

\[ P^O_i = \arg \min_{x \in \Gamma_{out}} \text{dist} (P_i, x). \quad (4.3.9) \]

Because of our definition of the inflow and outflow boundaries together with the fact that the immersed boundary representation in the Stokes’ model allows the consideration of exclusively rectangular grids these minimums always exist and are unique. These pores will not show up as degrees of freedom in the model, but will be included in the force acting on their respective pores within the computational domain to set the pressure boundary conditions. Additionally, we enforce periodic conditions as previously described. This setup is illustrated in the bottom image in Figure 4.5.

On this network we solve the problem,

\[ \sum_{i \in I_{in}, j \in I_i} \sum_{i \not\in I_{out}} c_{ij} (p_i - p_j) = 0, \quad (4.3.10) \]
\[ \sum_{i \in I_{in}} \sum_{j \in I_i} c_{ij} (p_i - p_j) + c_{in}^i p_i = c_{in}^i p_{in}, \quad (4.3.11) \]
\[ \sum_{i \in I_{out}} \sum_{j \in I_i} c_{ij} (p_i - p_j) + c_{out}^i p_i = c_{out}^i p_{out}. \quad (4.3.12) \]

Here \( c_{out}^i \) and \( c_{in}^i \) are the permeabilities in the shortened throats connecting the ghost boundary pores \( P^I_i \) and \( P^I_i \) to \( P_i \), respectively. \( p_{in} \) and \( p_{out} \) are the pressures prescribed on the inflow and outflow boundaries to induce flow in the direction from \( p_{in} \) to \( p_{out} \).
FIGURE 4.5: Illustration of domain division for $d = 2$.

We are now ready to describe our method, which is presented as Algorithm 4. With the pore-network structure as described, we are left with determining the set of values
\( C \) as well as \( \{ c_i^{\text{in}} \mid i \in I_{\text{in}} \} \) and \( \{ c_i^{\text{out}} \mid i \in I_{\text{out}} \} \). To do this, we turn to the immersed boundary Stokes’ model on the subdomains \( D^i \),

\[
-\mu \Delta u + \chi_{D^g} \eta^{-1} u + \nabla p = 0, \quad \text{on } D_f,
\]

\[
\nabla \cdot u = 0, \quad \text{on } D_f,
\]

\[
u = u_D, \quad \text{on } \Gamma_D,
\]

\[
\mu \frac{\partial u}{\partial n} - np = 0, \quad \text{on } \Gamma_N.
\]

Here \( \chi_{D^g} \) denotes the indicator function for the growth domain \( D^g \). On each subdomain \( D^i \), we solve (4.3.13)-(4.3.16) for flows in each principal axis direction, and compute the local subdomain permeabilities \( \kappa^{i_{\text{dir}}} \), for \( \text{dir} \in \{xx, yy, zz\} \) (\( \text{dir} \in \{xx, yy\} \) in 2d). For \( j \in I_i \), we denote by \( \kappa^{i_{\text{dir}}} \) the permeability computed in domain \( D^i \) in the direction of the axis connecting \( D^i \) and \( D^j \). For example, for the geometry in Figure 4.5, we have, \( \kappa^{1_{\text{dir}}} = \kappa^{1_{xx}} \) while \( \kappa^{1_{\text{dir}}} = \kappa^{1_{yy}} \). We then define, for each \( i \) and each \( j \in I_i \),

\[
c_{ij} = 0.5(\kappa^{i_{\text{dir}}} + \kappa^{j_{\text{dir}}}).
\]

We then set

\[
c_i^{\text{in}} = 0.5\kappa_{i-\text{in}},
\]

\[
c_i^{\text{out}} = 0.5\kappa_{i-\text{out}}.
\]

Notice that if \( \kappa \) were constant the throat connecting a boundary pore to an interior pore would have \( \kappa \) exactly one half of the permeability in throats connecting interior pores. This stems from the fact that the length of the throat is in a multiplicative factor (i.e. we divide by the pressure gradient) in the computation for an upscaled permeability from a pore-scale flow solution (4.1.20). Since these boundary pores are on the boundary of their subdomains, the distance from the boundary pore to the pore it connects to is half that of the distance between two pores in the computational domain. Thus halving its value has the desired effect. We solve the resulting pore-network model (4.3.10)-(4.3.11), and compute a permeability on the entire domain from the pore-network solution.
Algorithm 4: Direct Three-scale Method

1. for $i \in \{1, \ldots, N_p\}$ do
2.     Solve (4.3.13)-(4.3.16) on subdomain $D^i$ for flows in each axis direction.
3.     Compute $\kappa_{\text{dir}}^i \forall \text{dir} \in \{xx, yy, zz\}$ ($\{xx, yy\}$ in 2d).
4. for $i \in \{1, \ldots N_p\}$ do
5.     for $j \in I_i$ do
6.         $c_{ij} \leftarrow 0.5(\kappa_{i \to j}^i + \kappa_{j \to i}^j)$
7.         if $i \in I_{\text{in}}$ then
8.             $c_{i\text{in}}^i \leftarrow 0.5\kappa_{i \to \text{in}}^i$
9.         if $i \in I_{\text{out}}$ then
10.        $c_{i\text{out}}^i \leftarrow 0.5\kappa_{i \to \text{out}}^i$
11. Solve (4.3.10)-(4.3.11).
12. Compute $\kappa$ from pore-network solution according to (4.1.24).
13. Perform Darcy scale simulations.

4.3.1.1 Interpretation as Multiscale Finite Volume for Darcy

When the subdomains are rectangles this method may be viewed as a multiscale finite volume discretization for the velocity-free formulation of Darcy’s Law,

$$-\nabla \cdot (\kappa \nabla p) = 0, \ x \in D. \quad (4.3.20)$$

Here $\kappa : \mathbb{R}^d \to \mathbb{R}^{d \times d}$ is the full permeability tensor. Consider this equation posed on subdomain $D^i$. Applying the Divergence Theorem we have,

$$-\int_{\partial D^i} \kappa \nabla P \cdot n = 0, \quad (4.3.21)$$
where $n$ denotes the unit outward normal along the surface $\partial D^i$. We then note that by construction,

$$\partial D^i = \bigcup_{j \in I_i} D^i \cap D^j. \quad (4.3.22)$$

Define $\partial_j^i := D^i \cap D^j$. So, we have

$$-\sum_{j \in I_i} \int_{\partial_j^i} \kappa \nabla P \cdot n = 0. \quad (4.3.23)$$

Using the midpoint flux, we define $\kappa|_{\partial_j^i} = 0.5(\kappa^i + \kappa^j)$, where $\kappa^i$ is the constant value of the permeability tensor in domain $i$. Then,

$$-\sum_{j \in I_i} \int_{\partial_j^i} 0.5(\kappa^i + \kappa^j) \nabla P \cdot n = 0. \quad (4.3.24)$$

Then approximating the gradient in the $j$ direction, since we are on a uniform cartesian grid,

$$\nabla_{i\rightarrow j} P \cdot n \approx \frac{p_j - p_i}{d_{i\rightarrow j}}. \quad (4.3.25)$$

Here $d_{i\rightarrow j}$ is the distance between the cell centers for cells $i$ and $j$. Then we observe,

$$(\kappa^i + \kappa^j) \nabla P \cdot n = ((\kappa^i_{i\rightarrow j} + \kappa^j_{i\rightarrow j})(\nabla P)_{i\rightarrow j}).$$

Then noting that the distance $d_{i\rightarrow j}$ will be canceled by term resulting from the integration along the surface $\partial_j^i$, we have

$$-\sum_{j \in I_i} 0.5(\kappa^i_{i\rightarrow j} + \kappa^j_{i\rightarrow j})(p_j - p_i) = 0, \quad (4.3.26)$$

or equivalently,

$$\sum_{j \in I_i} 0.5(\kappa^i_{i\rightarrow j} + \kappa^j_{i\rightarrow j})(p_i - p_j) = 0, \quad (4.3.27)$$

which is precisely the pore-network model defined above.

The relationship between the multiscale pore-network model and the finite volume discretization of the velocity-free Darcy’s Law provides intuition why, with accurate porescale computed conductivities the accuracy of the structure of the pores and throats...
to the actual porous geometry may not be required. In Section 4.5, we show excellent agreement between the macroscale permeability computed from (i) a full porescale simulation and (ii) the direct method while the underlying porous structure is not considered in the construction of the pore-network model.

4.3.2. Reduced Order Three-scale Fluid Flow Method

Notice that in a transient simulation with an evolving porous geometry, employing the direct three-scale method described by Algorithm 4 requires a domain decomposed porescale simulation at each time that the geometry has changed sufficiently to warrant updating the macroscale permeability. While this is an improvement over requiring global porescale simulations, for a large problem this strategy can still be computationally intractable. Next we consider a reduced-order method that considers porescale geometric evolution as a stochastic process. This new method allows for efficient transient and transport simulations at higher scale (pore-network or Darcy) without sacrificing the accuracy of porescale simulation for determining conductivities in the pore-network model, though we consider here a static simulation for proof of concept.

To achieve this, the conditional random vector \( \kappa_v(\omega) \) is pre-computed in an offline stage and then sampled based on the growth \( v \) in each throat to evolve the macroscale permeability in the online stage. In this section we assume that for data we have a pore-network representation of the geometry, \( \mathbb{P} \). This representation may come externally as an accurate representation of the porescale geometry, or from the structured decomposition of the domain described in the previous section.

Define \( B^{ij} \subset D \) to be the portion of the geometry \( D \) containing \( P_i \), \( P_j \), and \( T_{ij} \). Clearly we have

\[
\bigcup_{i,j=1, i<j}^{N_p} B^{ij} \supset D_f. \tag{4.3.28}
\]

Figure 4.6 illustrates this decomposition for the simple geometry presented earlier in Fig-
ure 4.1. Consistent with the notation for the full domain, we denote by $B_0^{ij}$ the base geometry for the domain $B_{ij}$ (i.e. only rock and void space). We denote by $B_{f,0}^{ij}$, $B_{s,0}^{ij}$, and $\Gamma_0^{ij}$ the void space, solid matrix, and the interface between them for the base geometry on subdomain $ij$, respectively. Similarly, for an event $\omega \in \Omega$ we have $B_f^{ij}(\omega)$, $B_s^{ij}(\omega)$, $B_g^{ij}(\omega)$, and $\Gamma^{ij}(\omega)$ denoting the fluid subdomain, solid matrix, growth domain, and boundary for the fluid subdomain, respectively, for the subdomain realization $B^{ij}(\omega)$.

We are now ready to describe our method, which is presented in pseudo-code in Algorithm 5. The first step is to approximate the conditional random vector $\kappa_v(\omega)$. To
do this, we define a set of values \( \{v_l\}_{l=1}^{N_v} \), \( v_l \in (0,1) \) for each \( l \). These serve as the ’discretization’ of the mass fraction of growth, \( v \), which parameterizes the stochastic process \( \kappa_v(\omega) \). Next, for each \( l \in \{1, \ldots, N_v\} \) we sample the geometry \( N_\omega \) times, generating \( \{Z_x(\omega^l_k)\}_{k=1,l=1}^{N_\omega,N_v} \). We then solve the immersed boundary Stokes’ model (4.3.13)-(4.3.16) on each \( B^{ij}(\omega^l_k) \). Effectively, we have prescribed a collocation method for the stochastic model (4.2.2)-(4.2.5). At this point the reader may be imagining a large computational overhead depending on the sizes of the numbers \( N_\omega \) and \( N_v \). However, we note that these computations are independent, and can be performed concurrently. In addition, in a transient simulation, this process is performed only one time as an offline, preprocessing step. For each flow solution \( (u^{ij}(\omega^l_k), p^{ij}(\omega^l_k)) \), we compute a permeability \( \kappa^{ij}_{v_l}(\omega_k) \). These permeabilities \( \{\kappa^{ij}_{v_l}(\omega_k)\}_{l=1,k=1}^{N_v,N_\omega} \) are then used to compute empirical probability density functions \( F^{ij}_{l}(\omega) \).

Next we turn to the pore-network \( \mathbb{P} \). To define \( C \), we choose each \( c_{ij} \) by sampling the densities \( F^{ij}_{l}(\omega) \) according to the amount of obstruction present in the throat \( T_{ij} \). We then solve the system (4.1.11), and compute the permeability for the full domain \( D \) from this pore-network solution as a proof of concept for the sampling method. In practice we envision the sampling to take place at each time step in a transient simulation with transport simulations determining the mass fraction of obstacles.

To validate the method, in Section 4.5, we perform steps 9-12 in Algorithm 5 repeatedly to compute a distribution of full domain \( \kappa \) stemming from this method for a geometry containing material growth. We see excellent agreement between the true value of \( \kappa \) for the geometry and the distribution of predicted values based on subdomain mass fraction sampling.
Algorithm 5: Reduced Order Three-scale Method

1 Define $N_v$ values $\{v_l\}_{l=1}^{N_v}$.

2 for $T_{ij} \in T$ do

3     for $l \in 1, \ldots, N_v$ do

4         for $k \in 1, \ldots, N_\omega$ do

5             Sample $\omega_k \in \Omega$ satisfying $|B_{g}^{ij}(\omega_k)/B_f^{ij}(\omega_k)| = v_l$.

6             Solve (4.3.13)-(4.3.16) on $B^{ij}(\omega_k)$.

7             Compute $\kappa_{ij}^{ij}(\omega_k)$.

8             Compute an empirical density function $F_{ij}^{ij}(\kappa)$ from $\{\kappa_{ij}^{ij}(\omega_k)\}_{k=1}^{N_\omega}$.

9 for $T_{ij} \in T$ do

10    $c_{ij} \leftarrow F_{ij}^{ij}(\kappa)$

11 Solve (4.1.11) with pressure drop and periodic boundary conditions.

12 Compute $\kappa$ from pore-network solution.

13 Perform Darcy simulations.
4.4. Software: HybGe-Flow3d

HybGe-Flow3D (HGF) [24] is a software package that solves multiscale laminar fluid flow problems in complex, uncertain geometries. HybGe-Flow3D is written primarily in C++, with OpenMP parallelism and CUDA acceleration available for meshing bottlenecks as well as linear algebra. HybGe-Flow3D was developed with a Linux workstation in mind, though thanks to the cross-platform CMake build system it has been successfully built in a Windows environment, as well.

The core dependencies of HybGe-Flow3D include

- GCC (tested with 4.8.5),
- CMake,
- OpenMP,
- BOOST C++ Libraries [1],
- Paralution Linear Algebra [56],
- CUDA (optional) [74],

The core feature of HybGe-Flow3D is the simulation of laminar flow on a stochastic geometry and the computation of upscaled permeabilities by sampling of the stochastic geometry. This is done efficiently by enforcing the changes in the geometry, as compared to a base, reference geometry, through the resistive immersed boundary term. HybGe-Flow3D, as of version 1.0.1 also contains a pore-network solver as well as modules to upscale the full permeability tensor from the porescale. In future versions more models will be implemented across all scales.

The porescale problem is discretized by a finite volume discretization on a staggered grid. Several options are available for linear algebra. The default is to solve the immersed
boundary Stokes’ system directly as a single linear system using the Paralution [56] package for linear algebra. A Generalized Minimum Residual (GMRES) solver is employed with a block multi-colored ILU and Jacobi type preconditioning strategy, as described in Section 2.3.5. Also available, but minimally tested are a Richardson type decoupling scheme for the continuity and momentum equations, as well as a Krylov-accelerated Uzawa scheme. Also available, but currently experimental is the use of the AMGX library, which is NVIDIA’s package for algebraic multi-grid on GPUs.

From the user’s perspective, using HybGe-Flow3D is as simple as pointing the compiled binary at a directory containing a geometry file and a problem parameters file. After a mesh has been built, the mesh is saved in the directory. If the user investigates further problems on this geometry, the mesh is loaded, rather than rebuilt. After simulations have completed, output folders are created in the problem directory and the requested output data is saved for the user. Flow solutions can be output either in Tecplot formatting or in VTK formatting for visualization with Paraview. We conclude this section with several example simulations. In each case the geometry is enforced via the immersed boundary. Figure 4.7 shows flow around a sphere using Tecplot for visualization. A flow solution on a synthetic porous geometry is given in Figure 4.8 using Tecplot for visualization. An example flow solution on an imaged glassbead geometry, courtesy of Dorthe Wildenschild [88], is given in Figure 4.9 using Paraview for visualization.

For full build and use instructions the reader is referred to the user manual. A reduced version of the current user manual is included in Appendix B. An up to date, full manual is available, along with the software [24] at www.github.com/costat/hybge-flow3d.
FIGURE 4.7: Pressure and $x$-component velocity of flow around a sphere
FIGURE 4.8: Porescale pressure and $x$-component velocity in a synthetic glassbead geometry
FIGURE 4.9: Porescale pressure solution in an imaged glassbead geometry
4.5. Numerical Results

In this section we present numerical simulations using the methods described in Section 4.3. We begin by motivating the methods. In particular, we examine the influence of the locations of clumps of material growth in a 2d geometry and take note of the strong dependence of the permeability computed from the porescale solution on the location of the growth. In particular, this highlights that assuming Hagen-Poiseuille flow in a pore-network model with reactive transport is a poor choice. We then present results applying the direct three-scale method and the reduced order sampling method for a model geometry with $d = 2$ as a proof of concept. Finally, we simulate a large 3d sandstone geometry with the methods.

4.5.1. Upscaling with Porescale Evolution

In this section we show an example of the dependence of upscaled permeability computed from porescale solutions on the pattern of the porescale evolution. We consider an idealized pore geometry in 2d. The discrete geometry is given by a 100x100 voxel array. On the top in Figures 4.10-4.13 we present plots of $\kappa(x)$ where $x$ denotes the location of the center of mass where we eliminate 5x5, 10x10, 15x15 and 20x20 voxels to obstruct the flow, respectively. In each of the experiments the principle flow direction is along the $x$-axis. On the bottom of each of these figures is a scatter plot comparing the permeability ($y$-axis) with the velocity magnitude at the center of mass of the obstacle ($x$-axis). We notice that for each obstacle size we have a strong dependence of the upscaled permeability on the location of the obstacle. In particular, obstacles in the center-north and center-south of the domain are hidden from the primary flow path and have very little influence on the resulting permeability, while growths in the center of the domain latitudinally exhibit tremendous influence, reducing the geometry’s permeability considerably. Also, interestingly, there is a strong relationship in the permeability between the velocity of the
unobstructed flow solution at a point and the presence of an obstacle at that point. In particular if material grows in high velocity regions the resulting permeability is reduced dramatically, while if material grows in low velocity regions there is little influence of the
growth on the permeability. This relationship is noticeable at small obstacle sizes but becomes extremely clear at higher obstacle sizes.
FIGURE 4.12: $\kappa$ as a function of the placement of a 15x15 object in void space (BOTTOM) $\kappa$ compared to velocity magnitude at location where object is placed - from Section 4.5.1.
4.5.2. Model 2D Geometry

In this section we consider the idealized porous medium, $D$, in 2d depicted in Figure 4.14. The physical dimensions in these experiments are given in $\mu$m. To test the meth-
ods we generate a realization $D(\omega)$ with material growth, depicted in Figure 4.15. Directly computed at the porescale, this geometry has a permeability $\kappa_{xx} = 1.27706 \times 10^{-3}(\mu m)^2$. 

FIGURE 4.14: Flow in an idealized porous geometry in 2D from Section 4.5.2.
4.5.2.1 Direct Three-Scale Method

We first consider the direct method discussed in Section 4.3. We begin by instructing HybGe-Flow3D to cut the geometry with material growth given in Figure 4.15 into 4x4 subdomains and to compute permeabilities for each subdomain in the $x$ and $y$ directions. Table 4.1 lists the resulting values.
<table>
<thead>
<tr>
<th>Subdomain</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_{xx}$</td>
<td>0.664</td>
<td>2.146</td>
<td>1.845</td>
<td>1.589</td>
<td>3.317</td>
<td>1.874</td>
<td>1.346</td>
<td>1.292</td>
</tr>
<tr>
<td>$\kappa_{yy}$</td>
<td>1.421</td>
<td>1.148</td>
<td>1.078</td>
<td>2.025</td>
<td>1.649</td>
<td>1.428</td>
<td>1.850</td>
<td>1.163</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\kappa_{xx}$</td>
<td>1.486</td>
<td>1.886</td>
<td>2.565</td>
<td>1.524</td>
<td>2.495</td>
<td>1.892</td>
<td>2.390</td>
<td>3.219</td>
</tr>
<tr>
<td>$\kappa_{yy}$</td>
<td>1.701</td>
<td>1.533</td>
<td>1.454</td>
<td>2.472</td>
<td>1.282</td>
<td>2.226</td>
<td>2.133</td>
<td>2.473</td>
</tr>
</tbody>
</table>

TABLE 4.1: Permeabilities ($\times 10^{-3}(\mu m)^2$) in direct subdivision strategy from Section 4.5.2.

We then define the pore-network according to the 4x4 subdivision as described in Section 4.3., and compute the macroscopic conductivity from the pore-network solution, obtaining $\kappa_{PN} = 1.2717 \times 10^{-3}(\mu m)^2$, which differs from the porescale result by approximately 0.42%.

### 4.5.2.2 Reduced Order Sampling Method

In this section we again consider the idealized porous medium in 2D depicted in Figure 4.14, and apply the reduced-order sampling method introduced in Section 4.3..

We begin by noting that when subdividing this domain to compute permeability distributions, we can consider a single throat connecting two pores, as depicted in Figure 4.16. We note that the full geometry can be recreated by tiling this simple throat geometry. This mesh is given on a 100x100 voxel rectangle with total dimensions $1\mu m \times 1\mu m$.

To compute empirical probability density functions and empirical cumulative distribution functions we place, at random, growths in the throat domain’s void space and compute the resulting absolute permeability. Five experiments are performed. For experiment $N$, 1000 geometries are produced with $N$ growths, each with volume 1.62% of the
void space. Empirical probability density functions and cumulative distribution functions are given in Figure 4.17. An example solution for \( N = 5 \) is presented in Figure 4.18.

![Figure 4.16: Simple throat subdomain of idealized porous medium from Section 4.5.2.](image)

Next we return to the geometry with obstacles given in Figure 4.15. We build and inform a pore-network model in subregions by sampling the distributions computed in the previous section, as described in Section 4.3. In this case, due to the simple structure of the idealized geometry it is straightforward to label regions as pores and throats and to approximate the geometry with the pore-network model. Figure 4.15 presents the porescale flow solution on the modified domain. As mentioned previously, the absolute permeability of the domain, computed directly, is \( \kappa = 1.27706 \times 10^{-3} (\mu m)^2 \). To verify the method we perform the pore-network sampling step 1000 times and produce a distribution of permeabilities for the full domain computed from the pore-network simulations. One realization of the pore-network solution is given in Figure 4.19. Figure 4.20 shows the empirical probability distribution of \( \kappa \) computed with the pore-network model which sam-
FIGURE 4.17: Empirical probability density functions (top) and cumulative distribution functions (bottom) for experiments on simple throat geometry from Section 4.5.2.

For each throat, the empirical distributions from the porescale based on the amount of growth present in that throat. In particular, the "peak" of the density function is very near the true permeability of the domain, computed directly from the porescale.
4.5.3. Sandstone

In this section we consider a 3d porous geometry, courtesy of Brett Lindquist, which came from imaging sandstone [15] consisting of a grain-size mixture of gravel, sand, silt, and clay. The porescale velocity magnitude for this geometry is given in Figure 4.21. We note that this geometry was also investigated by Peszynska and Trykozko in [86].

In the synthetic 2d geometry previously presented the porescale structure is periodic, thus we are able to approximate $\kappa_v(o)$ on a single representative subdomain and sample the single subdomain approximate random vector for pore-network throat permeabilities. Additionally, in that simple geometry the porescale structure is isotropic, i.e.,

$$\kappa_v(o)_{yy} \approx \kappa_v(o)_{zz} \approx \kappa_v(o)_{xx}. \quad (4.5.1)$$

In the sandstone the situation is more complicated. The porescale structure is not periodic and thus we have no reason to suspect that we can isolate a representative subdomain. Additionally we have no reason to expect (4.5.1). Thus we must approximate the conditional random vector $\kappa_v(o)$ in each subdomain and for each flow direction. We divide the domain into 4x4x4 (64) subdomains and approximate $\kappa_v(o)$ on each subdomain for sampling at the higher scale. Figures 4.22-4.25 display the approximated empirical density

FIGURE 4.18: Example domain with $N = 5$ growths from Section 4.5.2.
FIGURE 4.19: Pressure solution for a single realization of pore-network model for the modified domain from Section 4.5.2.

FIGURE 4.20: Empirical probability density function for $\kappa$ on the modified domain, computed with the synthetic structured pore-network model, which samples, for each throat, the porescale empirical distributions based on the amount of growth present from Section 4.5.2.
functions for 4 subdomains, illustrating the local dependence of the permeability response to porescale evolution for the sandstone geometry, as well as the anisotropy of $\kappa_v(\omega)$ for the sandstone. The permeabilities for the remaining subdomains are given in Appendix D. Missing subdomains or volumes of growth are due to lack of percolation, i.e. where there is no connected component of the void space connecting the inflow boundary to the outflow boundary.

Next we generate a geometry containing obstacles for validation of the sampling method. Figure 4.26 displays the velocity magnitude from the porescale solution on this geometry, and Figure 4.27 displays the obstacle growth over a transparent geometry. The absolute permeability for this modified domain computed directly from the porescale is
\[ \kappa = 4.5018 \times 10^{-5} (\mu m)^2. \]

Finally we solve the pore-network model \( M = 5000 \) times, each time sampling in each throat the permeabilities from the offline stage distributions according to the sampling method described in Section 4.3.2. The distribution of resulting permeabilities for the full domain is given in Figure 4.28. We note that the true permeability of the domain, \( \kappa = 4.5018 \times 10^{-5} \) is located within the large 'hump' in the density function, which suggests good agreement between the true value and the range of values stemming from the sampling method. An example pore-network solution for the flow is given in Figure 4.29.
FIGURE 4.22: Empirical densities, sandstone geometry subdomain 0 from Section 4.5.3.
FIGURE 4.23: Empirical densities, sandstone geometry subdomain 4 from Section 4.5.3.
FIGURE 4.24: Empirical densities, sandstone geometry subdomain 9 from Section 4.5.3.
FIGURE 4.25: Empirical densities, sandstone geometry subdomain 17 from Section 4.5.3.
FIGURE 4.26: Velocity magnitude in modified (obstacles in green) sandstone geometry from Section 4.5.3.
FIGURE 4.27: Growth pattern (in green) for modified sandstone geometry from Section 4.5.3.
FIGURE 4.28: Empirical probability density function for $\kappa$ on the modified sandstone geometry computed with pore-network model, which samples, for each throat, the porescale empirical distributions based on the volume of obstacles present from Section 4.5.3.
FIGURE 4.29: Pressure solution for a single realization of the pore-network model for the modified sandstone geometry from Section 4.5.3.
4.6. Sampling a Stochastic Geometry: An Open Direction

The numerical results in the previous section show promising results for the ability of the reduced order three-scale method to compute accurately, and very efficiently, macroscale permeabilities according to the evolution of the porescale geometry. In order to apply this method to a full transient model coupling porescale evolution to flow simulations, we require an efficient method for sampling the geometry for the offline stage in a way appropriate to desired physics. In Section 4.5. we made no restriction on the physics of the growth. In other words, we sampled a uniform distribution for each volume fraction. This illustrates that the model can handle any geometric configuration and that the three-scale method produces an accurate macroscale permeability for any type of porescale evolution. However, to apply the method to a specific application, we will need to sample the geometry appropriately.

One candidate for this open question is to develop penalty functionals for the optimization based principal component analysis method (OPCA) [109] appropriate to various physics. To understand this method we first introduce the traditional principal component analysis (PCA). We begin by assuming that we have a set of realizations for the discrete-parameter stochastic process $Z_i(\omega)$ (4.2.17) stemming from a spatial discretization of the continuous parameter stochastic process $Z_x(\omega)$ in the stochastic immersed boundary model (4.2.2). We denote this set of realizations by the collection of vectors $\{z_i\}_{i=1}^M$, where for each $i$, $z_i \in \mathbb{R}^N$ is referred to as a snapshot of the stochastic process, and is a sample path which contains the true physics of the evolution of the discrete geometry. These snapshots may come from imaging data or from porescale simulations of an appropriate reactive transport model. We note that porescale reactive transport simulations are computationally very expensive, and so the goal with principal component analysis in this context is to produce, in a computationally tractable way, new realizations which
adhere to the physics of the evolution in the snapshots. We denote by \( \bar{z} \) the \( N \)-length vector of mean values, i.e. \( \bar{z}_i \) is the mean value of the collection \( \{ [z_i(\omega)]_j \}_{j=1}^M \). We then define the data matrix

\[
X_C = [z_1 - \bar{z}, \ldots, z_M - \bar{z}] \in \mathbb{R}^{M \times N}.
\]

(4.6.1)

Then the covariance matrix for the data matrix is given by,

\[
C = \frac{1}{M-1} X_C X_C^T \in \mathbb{R}^{N \times N}.
\]

(4.6.2)

Next we compute the singular value decomposition (SVD) of the matrix

\[
Y = \frac{1}{\sqrt{M-1}} X_C,
\]

(4.6.3)

denoted by

\[
Y = U \Sigma V^T.
\]

(4.6.4)

We note that the SVD takes \( O(N^2M) \) operations. We then generate new realizations by

\[
z = U \Sigma \xi + \bar{z}
\]

(4.6.5)

where \( \xi \) is a \( N(0, 1) \) random vector of length \( N \). The resulting realization is Gaussian, and must be truncated to obtain the binary field needed for the process \( Z_i(\omega) \). We note that more efficient versions of this method exist, in particular the Kernel PCA method [109].

With a large enough set of snapshots, the PCA method reliably computes new realizations for a given stochastic process. However, in applications marked by expensive computations for the production of a snapshot, one may end up with too few snapshots, which can result in physically unreasonable realizations from the standard PCA method. To illustrate this, we begin with a set of \( M = 20 \) snapshots resulting from porescale reactive transport simulations describing the growth of biofilm in a 2d geometry and use PCA to generate new realizations. A selection of the snapshots are given in Figure 4.30, courtesy
of Malgorzata Peszynska. Here black denotes the solid matrix, blue the void space, and red the biofilm. Figure 4.31 show the results from generating new realizations by the PCA method. We see that two physical characteristics have degraded: (1) the biofilm grows in large connected components, whereas in the results from the PCA method we see several locations where individual cells, or small clumps of cells, show biofilm growth isolated from the larger sections; (2) the biofilm prefers to grow with a connection to the rock wall, whereas in the results from the PCA method we see several growths that are not a connected component with the rock wall.

FIGURE 4.30: Snapshots of biofilm growth from Section 4.6.

Vo and Durlofsky introduced in 2014 [109] a method designed to overcome the
shortcomings of the PCA method for generating realizations of binary fields. This method, the optimization based principal component analysis (OPCA) reformulates the final step of the PCA process (4.6.5) as an optimization problem, allowing the introduction of a penalty functional to discourage undesirable characteristics in the results of the PCA process. For Vo and Durlofsky, this penalty enforced a binary random field. The reformulation is given by replacing (4.6.5) by

$$z = \arg \min_{x \in \mathbb{R}^N} \| U \Sigma \xi + \bar{\Sigma} - x \|^2_2. \quad (4.6.6)$$

This formulation can now be viewed as the minimization of an objective function $f$, where
for the standard PCA method we have

\[ f(x) = \|U\Sigma \xi + z - x\|_2^2. \] (4.6.7)

When the PCA method fails to adhere to the proper physics due to too few snapshots, we can now consider adding penalty terms within the function \( f \). For example, for biofilm, one might add a penalty to the surface area of the growth domain, thus encouraging larger clumps, rather than many small growths.
4.7. Conclusions

In this chapter a reduced order three-scale fluid flow model was introduced to reduce the computational complexity of simulating fluid flow in evolving complex geometries, whose evolution is driven by some reactive transport phenomenon, e.g., biofilm growth, methane hydrate formulation or precipitation-dissolution phenomena.

The reduced order three-scale method is based on the reformulation of the geometric evolution as a stochastic process and the offline approximation of the quantity of interest, a conditional random vector describing the absolute permeability of the stochastic geometry. With our approach, transient fluid flow simulations may be performed at the pore-network and Darcy scales, coupled to a reactive transport model accounting for the physics of the geometric evolution, with the absolute permeability sampled from the offline approximated random vector $\kappa_v(\omega)$ describing the permeability depending on the volume fraction of obstructed domain.

To formulate the method for the most general physics we presented a novel formulation of fluid flow on a stochastic geometry based on the immersed boundary method. This formulation is robust at higher flow rates and allows for geometries that are not continuous deformations of the original geometry. Additionally, the penalization term in the immersed boundary region is allowed to vary spatially, accounting for multiple processes and for those that have reduced but non-zero permeabilities.

Additionally, a freely available, open source multiscale fluid solver, HybGe-Flow3D designed for the methods in this chapter was developed and is available for download from GitHub [24]. To illustrate the method we first performed simulations on an idealized 2d geometry with a periodic pore structure. We generated a sample of this geometry with obstructed flow and saw excellent agreement between the true permeability of the domain as computed directly at the porescale and the result of the reduced order three-scale
method. These results are very promising, but the simplicity of the geometry resulted in a periodic and isotropic random permeability. To test the method in a more realistic setting we then turned our attention to an imaged sandstone geometry. In this case we again generated a sample geometry with obstructed flow and saw excellent agreement between the true permeability computed at the porescale and the result of the reduced order three-scale method.

Our results shown here are promising for the reduced order three-scale method, though many open areas of research remain. At the front of the list we note that in order to apply the method to a full transient model the question of sampling the stochastic geometry according to appropriate physics in an efficient way during the offline stage needs to be addressed. There are several candidates for this question, including the development of penalty functionals to be used alongside the optimization based principal component analysis [109]. This strategy would entail several simulations of a porescale model for the evolution process to generate a training set, followed by low cost production of new realizations by the OPCA method applying the penalty functional to ensure the correct physics.
BIBLIOGRAPHY


41. Helmut Harbrecht and Michael Peters. Combination technique based second moment analysis for elliptic pdes on random domains.


APPENDICES
A Density Functional Theory

Quantum mechanics of electrons governs the properties of matter. The fundamental equation describing quantum behavior is the Schrödinger equation. However, the direct solution of a quantum mechanical problem of $N$ interacting electrons is an intractable problem on virtually any scale. A reformulation of this problem in terms of the electron density of Hohenberg and Kohn, and Kohn and Sham [45, 54] provides a tractable alternative, and density functional theory is based on this theory. Density functional theory formulates a minimization problem in terms of electron density $n(r)$, that is equivalent to the Schrödinger equation for the ground state of the material. For a thorough exposition, the reader is referred to [29]. In this section we provide a brief review.

We consider here the standard Hamiltonian, $\hat{H} = \hat{T} + \hat{V}_{ee} + \hat{V}_{ext}$ of $N$ interacting electrons, ignoring spin. Here $\hat{T}$ is the kinetic energy operator, $\hat{V}_{ee}$ is the Coulomb interaction between electrons, and $\hat{V}_{ext}$ the interaction of electrons with an external potential $v_{ext}(r)$.

$$\hat{T} = \sum_{i=1}^{N} -\frac{\hbar^2}{2m} \nabla_i^2, \quad \hat{V}_{ee} = \sum_{i<j}^{N} \frac{e^2}{|r_i - r_j|}, \quad \hat{V}_{ext} = \int v_{ext}(r) n(r) d^3r. \quad (A.1)$$

In (A.1) $\hbar = \frac{h}{2\pi}$ where $h$ is the Planck constant, $m$ is the electron mass, $-e = -|e|$ is the charge of the electron, and $r_j$ is the quantum mechanical position operator for electron $j$.

The solutions of the stationary Schrödinger equation

$$\hat{H}\Psi_n = E_n \Psi_n \quad (A.2)$$

are the many-electron wave functions $|\Psi_n\rangle$ with energy $E_n$. For the ground state $|\Psi_0\rangle$ with energy $E_0$ (A.2) is equivalent to the minimization problem

$$E = \min_{\Psi} \langle \Psi | \hat{H} | \Psi \rangle. \quad (A.3)$$
In [45] Hohenberg and Kohn establish the existence of a density variational principle for the ground state,

$$E = \min_n \left\{ F[n] + \int v_{\text{ext}}(r)n(r)d^3r \right\}, \quad (A.4)$$

$$F[n] = \min_{\Psi \to n} \langle \Psi | \hat{T} + \hat{V}_{\text{ee}} | \Psi \rangle. \quad (A.5)$$

The solution to (A.4) is computationally a considerable improvement over the solution of the Schrödinger equation; in (A.4), density is a function in $\mathbb{R}^3$, while in (A.2), $\Psi$ is posed in $\mathbb{C}^{3N}$. However, the functional $F[n]$ is unknown, and DFT requires approximate solutions for (A.5). The Kohn-Sham equations provide a framework for the solution of the minimization problem (A.4) as well as a basis for approximating density functionals.

For a system of $N$ non-interacting electrons we may find a solution to (A.5) by considering single electron Schrödinger equations,

$$\left(-\frac{\hbar^2}{2m} \nabla^2 + v_{\text{es}}(r) + v_{\text{xc}}([n]; r)\right) \psi_k(r) = \epsilon_k \psi_k(r) \quad (A.6)$$

where $\psi_k$, $\epsilon_k$ denote the eigenstate and energy of a single particle, and $v_{\text{es}}(r)$ the electrostatic potential. The exchange-correlation potential $v_{\text{xc}}([n]; r)$ is the functional derivative of the exchange correlation energy $E_{\text{xc}}[n]$ with respect to the electron density. $E_{\text{xc}}[n]$ is the remainder of the functional $F[n]$ after the kinetic energy of the $N$ electrons $T_{ni}[n]$, and the Hartree energy $U[n]$ have been removed:

$$E_{\text{xc}}[n] = F[n] - T_{ni}[n] - U[n]. \quad (A.7)$$

A solution to the Kohn-Sham equations (A.6)-(A.7) can be found iteratively for a suitable choice of approximation of the exchange correlation energy $E_{\text{xc}}$. 


B HybGe-Flow3D User Manual, V1.0.1

B1. HybGe-Flow3D Overview

HybGe-Flow3D (HGF) is a software package that solves multiscale laminar fluid flow problems in complex, uncertain geometries. In this appendix we describe how to compile and run Hybge-Flow3D.

B1.1 Model & Discretization

At the porescale, the basic equations solved by HGF are the Stokes equations, modified with a resistive term corresponding to an immersed boundary representation of complex geometry.

\[-\mu \nabla^2 u + \frac{1}{\eta} \chi_{\Omega_g} u = -\nabla p, \quad x \in \Omega,\]

\[u = u_D, \quad x \in \partial \Omega_D\]

\[\nabla u \cdot n = 0, \quad x \in \partial \Omega_N.\]

Here, \(\Omega \subset \mathbb{R}^d\) with \(d = 3\) or \(d = 2\) is the simulation domain, and \(\partial \Omega\) is its boundary. \(\partial \Omega_D\) refers to the 'Dirichlet boundary' where the fluid velocity is prescribed. This is used for no-slip and inflow boundary conditions. \(\partial \Omega_N\) refers to the 'Neumann boundary' where a homogeneous diffusive flux is prescribed. This is used for outflow conditions. Additionally we have \(\Omega = \Omega_g \cup \Omega_f, \Omega_g \cap \Omega_f = \emptyset\), where \(\Omega_g\) refers to the immersed boundary and \(\Omega_f\) the fluid flow domain. In these equations \(u\) is the fluid velocity, \(p\) is the fluid pressure, and \(\mu\) is the fluid viscosity. \(\chi_{\Omega_g}\) is the indicator function for the immersed boundary domain \(\Omega_g\), and \(\eta\) is a penalization parameter, taken to be very small. HybGe-Flow3D solves the above fluid flow model by a staggered-grid finite volume discretization.

HybGe-Flow3D has several problem scenarios built in. In the simplest case, HGF will simply solve the above model on a given geometry, and produce a constant upscaled absolute permeability based on the flow direction chosen. Additionally, HGF can be
instructed to solve the flow problem in all principal axis directions and produce the full upscaled absolute permeability tensor.

More interesting are the full multiscale simulation capabilities of HGF. In later sections these will be examined in detail. Briefly, HGF can be instructed to cut a geometry into subdomains and solve local porescale flow problems. On these subdomains absolute permeabilities are computed, and used in a constructed pore-network model. This pore-network model is then solved to produce a permeability for the full domain.

Additionally, HGF can be instructed to sample a stochastic resistive term on each subdomain and produce empirical probability density functions relating the permeability to the presence of growth of an obstruction in the void space. These densities are then sampled for throat permeabilities in a pore-network solve. In future work we anticipate this being useful for transient simulations at pore-network or corescale with reactive transport.

B2. Installation

HybGe-Flow3D has two variants. The first uses Paralution for linear algebra, and the second uses AMGX. The AMGX version is currently experimental, and thus we will not overview the build process for AMGX in this document.

In addition to requiring either Paralution or AMGX. HybGe-Flow3D requires the BOOST C++ Libraries, and (optional) CUDA. In particular, HGF requires that the system, filesystem, and serialization BOOST libraries be compiled. In this section we review the build process for Paralution, BOOST, CUDA, and HybGe-Flow3D on a linux machine. We make no promises that these steps work universally, but they have worked reliably for the author on a Debian based or Fedora machine.

B2.1 Building BOOST

Many BOOST libraries do not require separate compilation, as they are header-only libraries. In our case, though, we require the following compiled libraries:
• serialization

• filesystem

• system

We will describe here how to build these 3 libraries and provide a template module file. These instructions can be found at http://www.boost.org/doc/libs/1_60_0/more/getting_started/unix-variants.html.

First, download the boost_1_60_0.tar.gz from above site. We then open the tar with

```bash
$ tar -bzip2 -xf /path/to/boost_1_60_0.tar.gz
```

Then navigate into the unpacked directory

```bash
$ cd /path/to/boost_1_60_0
```

Then create a build directory where the libraries and header files will be installed. We assume this is located at /path/to/boost/build. We then issue the following command from within the boost_1_60_0 directory, setting prefix to our build directory

```bash
$ ./bootstrap.sh --prefix=/path/to/boost/build --with-libraries=serialization,system,filesystem
```

Then
This installs the two directories /path/to/boostbuild/lib and /path/to/boostbuild/include. The –with-libraries flag is optional. Without it boost will build all compiled libraries. However, this takes some time, and HGF uses only the three compiled libraries listed above.

Next we write an environment module file for BOOST. We assume for exposition that this file is placed at /path/to/modulefiles/boost/1.60.0, and describe the process of adding the folder /path/to/modulefiles to the module path using a bash terminal.

```
#%Module 1.0
#
# Boost module
#

module-whatis "boost/1.60.0"

set boost_home /path/to/boostbuild/

prepend-path PATH $boost_home
prepend-path LIBRARY_PATH $boost_home/lib
prepend-path LD_LIBRARY_PATH $boost_home/lib
prepend-path CMAKE_LIBRARY_PATH $boost_home/lib
prepend-path INCLUDE_PATH $boost_home/include
prepend-path C_INCLUDE_PATH $boost_home/include
prepend-path CPLUS_INCLUDE_PATH $boost_home/include
```
setenv BOOST_HOME $boost_home
setenv BOOST_DIR $boost_home

To see the modulefiles within /path/to/modulefiles we add the following line to our .bashrc:

```
module use-append /path/to/modulefiles
```

Open a new terminal or type

```
$ source ~/.bashrc
```

Finally, type

```
$ module avail
```

and verify that boost/1.60.0 is an available module.

### B2.2 Building CUDA

Instructions for building CUDA can be found at http://docs.nvidia.com/cuda/cuda-getting-started-guide-for-linux. We prefer not to include a link to /usr/bin/, opting instead for a cuda modulefile. Assuming the version of CUDA built is 7.5.18, the following is placed at /path/to/modulefiles/cuda/7.5.

```bash
#%Module

proc ModulesHelp { } {

    puts stderr "CUDA 7.5.18"

```
paralution is straightforward to build on Linux machines using cmake. The instructions contained here can be found at http://www.paralution.com/download/.

First, navigate to the directory in which you will build paralution, which we will denote by /path/to/paralution. We create a build directory for Paralution,
$ mkdir ./1.1.0

Then grab the tar file containing Paralution.

$ wget http://www.paralution.com/downloads/paralution-1.1.0.tar.gz

The following steps build paralution using cmake.

$ tar zxvf paralution-1.1.0.tar.gz
$ cd 1.1.0
$ cmake ../paralution-1.1.0/
$ make

Next we write an environment module file for Paralution. This file should be located at /path/to/modulefiles/paralution/1.1.0.

```
#%Module 1.0
#
# Paralution module
#
module-whatis "paralution/1.1.0"

set paralution_home /path/to/paralution/1.1.0
prepend-path PATH $paralution_home/bin
prepend-path LIBRARY_PATH $paralution_home/lib
```
prepend-path LD_LIBRARY_PATH $paralution_home/lib
prepend-path CMAKE_LIBRARY_PATH $paralution_home/lib
prepend-path INCLUDE_PATH $paralution_home/inc
prepend-path C_INCLUDE_PATH $paralution_home/inc
prepend-path CPLUS_INCLUDE_PATH $paralution_home/inc
prepend-path CMAKE_INCLUDE_PATH $paralution_home/inc
setenv PARALUTION_HOME $paralution_home
setenv PARALUTION_DIR $paralution_home

B2.4 Building HybGe-Flow3D

Once the dependencies are built and module files are in place, building HybGe-Flow3D is simple. First, clone (or download from www.github.com/costat/HybGe-Flow3D/) the repository.

```
$ cd /path/to/build/hgf
$ git clone git@github.com:costat/HybGe-Flow3D
```

Then navigate into the PARALUTION subfolder of the repository.

```
$ cd ./HybGe-Flow3D/PARALUTION
```

Next, load the appropriate modules,

```
$ module load paralution/1.1.0
```
$ module load cuda/7.5.18
$ module load boost/1.60.0

Next, execute the cmake script and make,

$ cmake .
$ make

And, done! The binary `hgf` is now located in the PARALUTION folder.

**B3. Running HybGe-Flow3D**

To run HGF, the binary must be pointed at a directory containing, at minimum, two files: a geometry file "Geometry.dat," and a parameter file "Parameters.dat." The format of these files is most easily explained by example. HGF ships with 3 examples located at `/path/to/hgf/examples/`. After running a simulation with a full porescale solve a file "Mesh.dat" is created and placed in the directory. If the file "Mesh.dat" is present, then for global porescale solves the "Geometry.dat" file is ignored and the "Mesh.dat" file is loaded. All output is placed in a directory `/path/to/problem/directory/Output/`. If this directory does not exist hgf creates the directory.

**C Geometry.dat**

Opening `../examples/3dCube/Geometry.dat`, we see

```
nx= 30
ny= 30
nz= 30
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```
The first three lines of the Geometry.dat file tell HGF the number of voxels in the x, y, and z direction in the geometry data file. The formatting is important here. HGF reads the first integer found after a space in line 1 as the value of nx, likewise for line 2 with ny, and line 3 with nz. Thus we must have nx= with no spaces, followed by a space, followed by the integer value of nx. Similarly for ny and nz. Each voxel must be a '0', '1', or '2'. In this trivial example, all voxels are '0', which refers to void space in the domain. A value of '1' identifies the solid matrix, which will not be meshed. Finally, a value '2' identifies part of the immersed boundary. This voxel will be in the mesh, but flow will be restricted by the immersed boundary term. The voxel array is formatted by the following rules:

- A row of voxels in Geometry.dat corresponds to a line of voxels in the x-direction in the domain, and is nx entries long.
- Each block of voxels corresponds to a z-slice, i.e. an x-y plane in the geometry, and has size nx×ny.
- Each z-slice is separated by an empty row.
- The first line of voxels must be in line 4 of the file. There is no vertical space between nz and the beginning of the voxel array.

C0.1 Parameters.dat

Opening ../examples/3dCube/Geometry.dat, we see

```
length = 1.0
width = 1.0
height = 1.0
visc = 1.0
```
direction = 0
output = 1
nThreads = 4
prec = 0
tolAbs = 1e-8
tolRel = 1e-8
maxIt = 3000
nCuts = 4

This file contains the problem and solver parameters needed to run HGF. Similarly to the formatting for nx, ny, and nz in the Geometry.dat file, it is important that each entry contains "valuelabel=" followed by a space, followed by the value (double or int, case depending). Additionally, these values are read in by their order, not by their label. So, entries must be provided in the order:

1. length ← the length of the domain, i.e. size in the x-direction.

2. width ← the width of the domain, i.e. the size in the y-direction.

3. height ← the height of the domain, i.e. the size in the z-direction.

4. visc ← the fluid viscosity.

5. direction ← this is an integer which determines the type of problem HGF will solve.
   The possible values are identified below.

6. output ← this is an integer that determines the format of the file containing flow solutions. 0 for tecplot formatting, and 1 for paraview vtk formatting.

7. nThreads ← this is an integer that tells Paralution how many shared memory threads to use. If this is greater than the machine’s maximum threads, Paralution will use all threads available. Must still be an entry in the Parameters.dat file even if AMGX is used, since ordering determines the read in of the parameters.
8. prec ← this is an integer that tells Paralution how many p-levels to use in the ILU preconditioner.

9. tolAbs ← the absolute tolerance for the linear solver residual.

10. tolRel ← the relative tolerance for the linear solver residual.

11. maxIt ← the maximum number of iterations to be used by the linear solver.

12. nCuts ← the number of cuts to make in each axis direction for the definition of subdomains in the subdivide multiscale strategies.

The direction parameter requires further discussion. If direction ∈ \{0, 1, 2\}, then HGF will solve a problem in a single principal axis flow direction, and compute the absolute permeability in that direction from the solution. For these cases, direction 0 corresponds to an x-flow problem, 1 to a y-flow problem, and 2 to a z-flow problem. If direction = 3, then all flow directions are solved and a permeability tensor is computed. If direction = -1, hgf simply meshes the geometry and saves the mesh to /problemdirectory/Mesh.dat.

Directions 4 and 5 define scenarios which are experimental, and for research purposes related to upcoming papers. If direction = 4 the domain is cut nCuts times in each axis direction to define subdomains. On each subdomain, permeabilities are computed from porescale simulations in each axis direction. These values are then used to define a pore-network model, and a global permeability in the x-direction is computed from the solution to the pore-network model. If direction = 5, the domain is subdivided similarly to the direction = 4 case. This time, though, hgf randomly samples the immersed boundary vector to compute distributions of permeabilities for each subdomain based on the mass fraction of immersed boundary. This distributions are then sampled to define a pore-network model and a global permeability in the x-direction is computed.
D  Hanford Sandstone Permeability Distributions

In this appendix we present the remaining subdomain permeability distributions for the sandstone geometry from Section 4.5.3.
FIGURE 0.32: Empirical densities, sandstone geometry subdomain 1 from Section 4.5.3.
FIGURE 0.33: Empirical densities, sandstone geometry subdomain 3 from Section 4.5.3.
FIGURE 0.34: Empirical densities, sandstone geometry subdomain 5 from Section 4.5.3.
FIGURE 0.35: Empirical densities, sandstone geometry subdomain 6 from Section 4.5.3.
FIGURE 0.36: Empirical densities, sandstone geometry subdomain 7 from Section 4.5.3.
FIGURE 0.37: Empirical densities, sandstone geometry subdomain 8 from Section 4.5.3.
FIGURE 0.38: Empirical densities, sandstone geometry subdomain 10 from Section 4.5.3.
FIGURE 0.39: Empirical densities, sandstone geometry subdomain 11 from Section 4.5.3.
FIGURE 0.40: Empirical densities, sandstone geometry subdomain 12 from Section 4.5.3.
FIGURE 0.41: Empirical densities, sandstone geometry subdomain 13 from Section 4.5.3.
FIGURE 0.42: Empirical densities, sandstone geometry subdomain 14 from Section 4.5.3.
FIGURE 0.43: Empirical densities, sandstone geometry subdomain 15 from Section 4.5.3.
FIGURE 0.44: Empirical densities, sandstone geometry subdomain 16 from Section 4.5.3.
FIGURE 0.45: Empirical densities, sandstone geometry subdomain 18 from Section 4.5.3.
FIGURE 0.46: Empirical densities, sandstone geometry subdomain 19 from Section 4.5.3.
FIGURE 0.47: Empirical densities, sandstone geometry subdomain 20 from Section 4.5.3.
FIGURE 0.48: Empirical densities, sandstone geometry subdomain 21 from Section 4.5.3.
FIGURE 0.49: Empirical densities, sandstone geometry subdomain 22 from Section 4.5.3.
FIGURE 0.50: Empirical densities, sandstone geometry subdomain 23 from Section 4.5.3.
FIGURE 0.51: Empirical densities, sandstone geometry subdomain 24 from Section 4.5.3.
FIGURE 0.52: Empirical densities, sandstone geometry subdomain 26 from Section 4.5.3.
Figure 0.53: Empirical densities, sandstone geometry subdomain 27 from Section 4.5.3.
FIGURE 0.54: Empirical densities, sandstone geometry subdomain 28 from Section 4.5.3.
FIGURE 0.55: Empirical densities, sandstone geometry subdomain 29 from Section 4.5.3.
FIGURE 0.56: Empirical densities, sandstone geometry subdomain 30 from Section 4.5.3.
FIGURE 0.57: Empirical densities, sandstone geometry subdomain 31 from Section 4.5.3.
FIGURE 0.58: Empirical densities, sandstone geometry subdomain 32 from Section 4.5.3.
FIGURE 0.59: Empirical densities, sandstone geometry subdomain 33 from Section 4.5.3.
FIGURE 0.60: Empirical densities, sandstone geometry subdomain 34 from Section 4.5.3.
FIGURE 0.61: Empirical densities, sandstone geometry subdomain 35 from Section 4.5.3.
FIGURE 0.62: Empirical densities, sandstone geometry subdomain 36 from Section 4.5.3.
FIGURE 0.63: Empirical densities, sandstone geometry subdomain 37 from Section 4.5.3.
FIGURE 0.64: Empirical densities, sandstone geometry subdomain 38 from Section 4.5.3.
FIGURE 0.65: Empirical densities, sandstone geometry subdomain 39 from Section 4.5.3.
FIGURE 0.66: Empirical densities, sandstone geometry subdomain 40 from Section 4.5.3.
FIGURE 0.67: Empirical densities, sandstone geometry subdomain 41 from Section 4.5.3.
FIGURE 0.68: Empirical densities, sandstone geometry subdomain 42 from Section 4.5.3.
FIGURE 0.69: Empirical densities, sandstone geometry subdomain 43 from Section 4.5.3.
FIGURE 0.70: Empirical densities, sandstone geometry subdomain 44 from Section 4.5.3.
FIGURE 0.71: Empirical densities, sandstone geometry subdomain 45 from Section 4.5.3.
FIGURE 0.72: Empirical densities, sandstone geometry subdomain 47 from Section 4.5.3.
FIGURE 0.73: Empirical densities, sandstone geometry subdomain 48 from Section 4.5.3.
FIGURE 0.74: Empirical densities, sandstone geometry subdomain 49 from Section 4.5.3.
FIGURE 0.75: Empirical densities, sandstone geometry subdomain 50 from Section 4.5.3.
FIGURE 0.76: Empirical densities, sandstone geometry subdomain 51 from Section 4.5.3.
FIGURE 0.77: Empirical densities, sandstone geometry subdomain 52 from Section 4.5.3.
FIGURE 0.78: Empirical densities, sandstone geometry subdomain 53 from Section 4.5.3.
FIGURE 0.79: Empirical densities, sandstone geometry subdomain 54 from Section 4.5.3.
FIGURE 0.80: Empirical densities, sandstone geometry subdomain 55 from Section 4.5.3.
FIGURE 0.81: Empirical densities, sandstone geometry subdomain 56 from Section 4.5.3.
FIGURE 0.82: Empirical densities, sandstone geometry subdomain 57 from Section 4.5.3.
FIGURE 0.83: Empirical densities, sandstone geometry subdomain 58 from Section 4.5.3.
FIGURE 0.84: Empirical densities, sandstone geometry subdomain 59 from Section 4.5.3.
FIGURE 0.85: Empirical densities, sandstone geometry subdomain 60 from Section 4.5.3.
FIGURE 0.86: Empirical densities, sandstone geometry subdomain 61 from Section 4.5.3.
FIGURE 0.87: Empirical densities, sandstone geometry subdomain 62 from Section 4.5.3.
FIGURE 0.88: Empirical densities, sandstone geometry subdomain 63 from Section 4.5.3.