Parallel Parameter Study of the Wigner-Poisson Equations for RTDs

M. S. Lasater, C. T. Kelley[1]

Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, Raleigh, NC, 27695-8205

A. G. Salinger

Sandia[2] National Laboratories P.O. Box 5800, MS-1111 Albuquerque, New Mexico, 87185, USA

D. L. Woolard

U. S. Army Research Office U. S. Army Research Laboratory, RTP, North Carolina, 27709-2211, USA

P. Zhao

Electrical and Computer Engineering Department North Carolina State University, Raleigh, North Carolina, 27695-8205, USA

Abstract

We will discuss a parametric study of the solution of the Wigner-Poisson equations for resonant tunneling diodes. These structures exhibit self-sustaining oscillations in certain operating regimes. We will describe the engineering consequences of our study and how it is a significant advance from some previous work, which used much coarser grids. We use LOCA and other packages in the Trilinos framework from Sandia National Laboratory to enable efficient parallelization of the solution methods and to perform bifurcation analysis of this model. We report on the parallel efficiency and scalability of our implementation.

[1] Corresponding author

[2] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000

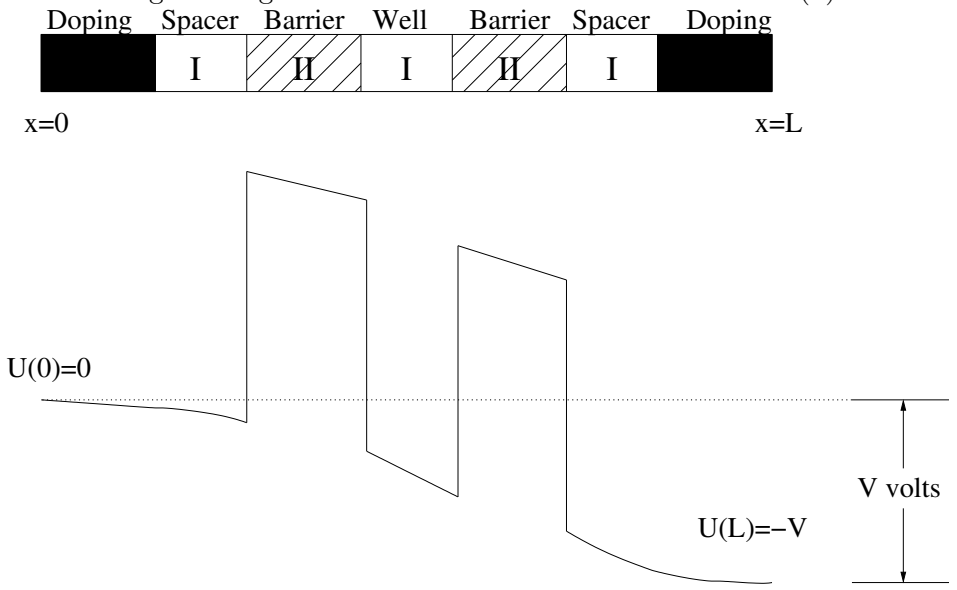
[3] *E-mail addresses:* mslasate@unity.ncsu.edu (M. S. Lasater), tim_kelley@ncsu.edu (C. T. Kelley), agsalin@sandia.gov (A. G. Salinger), dwight.woolard@us.army.mil (D. L. Woolard), pzhao@eos.ncsu.edu (P. Zhao),

1 Introduction

Semiconductor technology has developed to the point where the next generation of electronic devices will operate at the nanoscale level (10^{-9} meters). One particular prototypical nanostructure under current investigation is the resonant tunneling diode (RTD). Engineers and physicists researching this quantum device need accurate and efficient models for electron transport derived from first-principle physics in order to completely understand the quantum mechanical effects which will dominate the device physics.

A RTD structure is typically created by growing alternating layers of two different semiconductor materials (e.g., types I and II) where one material system has a significantly larger bandgap than the other. When heterostructures of this type are formed as illustrated in Figure 1, the conduction band alignments can lead to the formation of potential barriers (i.e., regions labeled II) between regions of narrow bandgap material (i.e., regions labeled I). Note that Figure 1 correlates the alternating layers of materials to the resultant potential energy profile for the case of a large applied bias.

Fig. 1. Diagram of RTD and the Electric Potential U(x)



In our simulations, the semiconductor material in the I-region is gallium arsenide (GaAs), and the semiconductor material in the II-region is aluminum gallium arsenide (AlGaAs). The wider bandgap material in II-regions creates *barriers* in the potential energy profile. This results in the formation of a *quantum-well* in the center section that prohibits classical electron transport (i.e., drift or diffusion). On either end of the RTD structure the I-regions are doped (represented by dark regions) to provide for electron reservoirs for injections and collection processes. Doping refers to the procedure where atoms are seeded into the base material that contain more (or less) electrons and

that leads to a situation where the number of free electrons are increased (or decreased). In our studies, only the reservoir regions are doped which means that all electrons will transverse *spacer* I-regions on either side of the barrier-well-barrier active regions during the transport processes.

Classically, if a particle runs into a potential barrier and it does not have enough momentum, it will be reflected back. Since quantum mechanics treats electrons as waves instead of particles, an electron at any speed that encounters a barrier still has some probability of passing through the barrier. This effect is known as quantum tunneling and is the basis of this device. If a voltage difference is applied across the device, electrons will start to move along the device, tunnel through the barriers, and reach the other side, thus creating a current.

Numerical simulations [14], [15], have shown that current oscillation can be expected for certain voltage differences, and that these current oscillations occur within the terahertz (THz) regime. With these numerical simulations, engineers and physicists are hoping to understand what physical mechanisms create these intrinsic oscillations and determine what physical parameters (i.e. doping profile, barrier height and width, well width, etc.) are conducive to sustaining and controlling these oscillations in hopes of producing a viable high frequency power source. This work is an attempt to create a faster and more accurate RTD simulator to aid the engineers in these goals.

2 Model Description

The model used to describe the electron transport in these devices is the Wigner-Poisson equations [2], [13]. These equations consist of a nonlinear integro-partial differential equation (IPDE) (2) which describes the time-evolution of the distribution of the electrons in the device and Poisson's equation (9) for the electrostatic potential.

The IPDE is

$$\frac{\partial f}{\partial t} = W(f) = Kf + P(f) + S(f). \tag{1}$$

Here, $f = f(x, k, t)$, is the distribution of the electrons. It is a function of the position of the electron $x \in [0, L]$, momentum of the electron, $k \in (-\infty, \infty)$, and time, $t > 0$. L is the length of the device.

In this paper we consider only the time-independent problem

$$W(f) = Kf + P(f) + S(f) = 0. \tag{2}$$

A study of the time-independent system leads to important dynamic information about the time-dependent system, which we will explore in subsequent work. Earlier work, which focused on the time-dependent system for coarse grids, can be found [7], [8].

The linear term Kf on the right side of (2) represents the kinetic energy effects on the distribution and is given by

$$Kf = -\frac{hk}{2\pi m^*}\frac{\partial f}{\partial x}. \tag{3}$$

Here, h is Planck's constant and m^* is the effective mass of the electron. The second term, $P(f)$, is the nonlinear term in the equation and accounts for the potential energy effects on the distribution

$$P(f) = -\frac{4}{h}\int_{-\infty}^{\infty} f(x,k')T(x,k-k')dk'. \tag{4}$$

The function $T(x,k)$ is defined by

$$T(x,k) = \int_0^{\frac{L_c}{2}} [U(x+y) - U(x-y)]\sin(2xk)dy. \tag{5}$$

In this equation, $U(x)$ is the electric potential as a function of position, and L_c is the correlation length. This term is nonlinear in f since $U(x)$ depends on f through Poisson's equation.

In (4) and (5), there are convolutions to compute in both x and k space, which can be efficiently numerically computed using Fast Fourier Transforms (FFTs). We found, however, when we compared a FFT-routine, written using FFTPACK [11], to one that uses standard BLAS routines for computing the numerical approximations to (4) and (5), we discovered that the FFT routine was slower on the parallel machines which we used to compute the numerical results reported in this paper. Therefore, the results in this paper do not use FFT-based routines to compute the convolutions in (4) and (5).

The last term describes electron-electron scattering

$$S(f) = \frac{1}{\tau}\left[\frac{\int_{-\infty}^{\infty} f(x,k')dk'}{\int_{-\infty}^{\infty} f_0(x,k')dk'}f_0(x,k) - f(x,k)\right]. \tag{6}$$

In (6), τ is the relaxation time, and $f_0(x,k)$ is the equilibrium Wigner distribution. This is the steady-state solution to Equation (2) when there is no

voltage difference across the device.

Boundary conditions are imposed at the device edges. On the left ($x = 0$), we have for $k > 0$ (electrons with positive momentum that are moving right)

$$f_0(0,k) = \frac{4\pi m^* k_B T}{h^2} \ln\left(1 + \exp\left[\frac{1}{k_B T}\left(\frac{h^2 k^2}{8\pi^2 m^*} - \mu_0\right)\right]\right). \tag{7}$$

Similarly on the right ($x = L$) we specify f for $k < 0$ (electrons with negative momentum that are moving left)

$$f_0(L,k) = \frac{4\pi m^* k_B T}{h^2} \ln\left(1 + \exp\left[\frac{1}{k_B T}\left(\frac{h^2 k^2}{8\pi^2 m^*} - \mu_L\right)\right]\right). \tag{8}$$

In (7) and (8), k_B is Boltzmann's constant, T is the temperature, μ_0 is the Fermi energy at $x = 0$, and μ_L is the Fermi energy at $x = L$.

The electric potential $U(x)$ is the sum of the potential barrier $\Delta_c(x)$ that arises from the heterojunction of the two different semiconductor materials and the electrostatic potential $u(x)$. The electrostatic potential is the solution of Poisson's equation

$$\frac{d^2u}{dx^2} = \frac{q^2}{\epsilon}\left[N_d(x) - \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x,k')dk'\right]. \tag{9}$$

In (9), q is the charge of the electron, ϵ is the dielectric constant, and $N_d(x)$ is the doping profile. The boundary conditions for (9) are

$$u(0) = 0, u(L) = -V, \tag{10}$$

where $V \geq 0$ is the applied voltage. The dependence of $U(x) = u(x) + \Delta_c(x)$ on f is through $u(x)$.

3 Discretization

We discretize the IPDE by a finite difference method for the derivative and by the composite trapezoid rule for the integrals. We will denote the solution of the discrete problem by $\vec{f} \in R^n$, where $n = N_x \times N_k$ is the number of grid points in the discrete (x,k) domain.

For the x-domain, we use N_x equally spaced grid points $x_i = (i-1)\Delta x$, $i = 1, 2, \ldots, N_x$ and $\Delta x = \frac{L}{N_x - 1}$. For the k-domain, we first truncate from $-\infty$ to ∞ to $-K_{Max}$ to K_{Max}, where K_{Max} is the maximum momentum we consider. We use N_k equally spaced grid points where $k_j = (2j - N_k - 1) * \frac{\Delta k}{2}$, $j = 1, 2, \ldots, N_k$ and $\Delta k = \frac{2}{N_k K_{Max}}$. The number of grid points is then $n = N_x \times N_k$ and $\vec{f}$ is the approximation to f at the grid points (x_i, k_j) for $i = 1$, 2, ..., N_x and $j = 1$, 2, ..., N_k.

For the $K\vec{f}$ term, we use an upwind differencing scheme to approximate $\frac{\partial f}{\partial x}$. So we get for $k_j > 0$,

$$K\vec{f}\Big|_{(x_m,k_j)} \approx \frac{hk_j}{2\pi m^*}\left(\frac{-3\vec{f}_{mj} + 4\vec{f}_{m-1,j} - \vec{f}_{m-2,j}}{2\Delta x}\right), \tag{11}$$

and for $k_j < 0$,

$$K\vec{f}\Big|_{(x_m,k_j)} \approx \frac{hk_j}{2\pi m^*}\left(\frac{3\vec{f}_{mj} - 4\vec{f}_{m+1,j} + \vec{f}_{m+2,j}}{2\Delta x}\right). \tag{12}$$

The integrals in the $P(\vec{f})$ and $S(\vec{f})$ terms are approximated with a sum:

$$P(\vec{f})\Big|_{(x_m,k_j)} \approx \frac{1}{h}\sum_{j'=1}^{N_k} \vec{f}_{mj'} T(x_m, k_j - k_{j'})\Delta k. \tag{13}$$

Here, $T(x_m, k_j - k_{j'})$ is also approximated with a sum,

$$T(x_m, k_j - k_{j'}) \approx \sum_{m'=1}^{N_c} [U(x_m + x_{m'}) - U(x_m - x_k)] \sin(2x_{m'}(k_j - k_{j'}))\Delta x. \tag{14}$$

For $S(\vec{f})$, we have

$$S(\vec{f})\Big|_{(x_m,k_j)} \approx \frac{1}{\tau}\left[\frac{f_0(x_m, k_j)\sum_{j'=1}^{N_k} \vec{f}_{mj'}}{\sum_{j'=1}^{N_k} f_0(x_m, k_{j'})} - \vec{f}_{mj}\right]. \tag{15}$$

We use a standard three-point differencing scheme to discretize Poisson's equation. For $m = 2, 3, \ldots, N_x - 1$ it is:

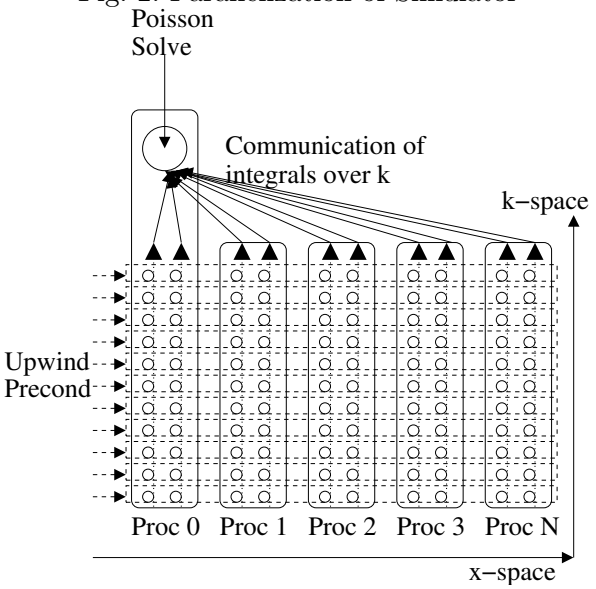
$$\frac{u(x_{m-1}) - 2u(x_m) + u(x_{m+1})}{\Delta x^2} = \frac{q^2}{\epsilon}\left[N_d(x_m) - n(x_m)\right] \tag{16}$$

with $u(x_1) = 0$ and $u(x_{N_x}) = -V$ set for the boundary conditions.

4 Parallel Simulator

To parallelize our evaluation of $W(\vec{f})$, we partition (x, k) space among different processors. In this paper each processor gets a contiguous block of x-space and all of the corresponding k-space. By splitting the data between the processors this way, we ensure that the integrals in k-space can be performed by each processor independently. This splitting, though, will require communication between the processors when the spatial derivative term in Equation (3) is calculated. The Poisson solve was not parallelized and is performed by the main processor before everything else is calculated. Once $U(x)$ is known, the main processor sends out a copy to rest of the processors. The processors then compute their part of $W(\vec{f})$ and return this to the main processor. Figure 2 gives a visual description of how the data is split among the processors and how the processors perform the computations.

Fig. 2. Parallelization of Simulator



5 Continuation Methods

Continuation methods are used to solve nonlinear equations that depend on a parameter. Let $G : R^n \times R \to R^n$ be a nonlinear equation, $\vec{z} \in R^n$ be the solution vector, and $\lambda \in R$ the parameter. We would like to find a curve in R^n parameterized by λ (denoted by $(\vec{z}, \lambda) \in R^n \times R$) that solves the nonlinear equation

$$G(\vec{z}, \lambda) = 0 \tag{17}$$

Continuation methods numerically generate a sequence $\{(\vec{z}_m, \lambda_m)\}$ that satisfy $G(\vec{z}_m, \lambda_m) = 0$.

We will now present two common continuation methods. A standard technique for solving nonlinear equations is through Newton's method [5], and each continuation method uses it to solve the nonlinear equation. Newton's method is an iterative method for solving a nonlinear equation. Each iteration (or Newton step) corrects the previous iteration and requires a linear solve for this correction where the Jacobian matrix ($G'(\vec{z}, \lambda) = \frac{\partial G}{\partial \vec{z}}(\vec{z}, \lambda) \in R^{n \times n}$) is the coefficient matrix. Analysis of the method shows that if the initial iterate is taken close enough to the solution and the function is well-behaved, the Newton iteration is guaranteed to converge to the solution quickly [5]. The two continuations we will discuss are natural continuation and pseudo arclength continuation.

We will first describe natural continuation. This is the easiest continuation method to understand. Here, one finds an initial point on the solution curve and then monotonically increases or decreases the parameter as Newton's method is used to compute points on the solution curve.

Assume we have just computed $(\vec{z}_m, \lambda_m)$, and now we want to compute the next solution $(\vec{z}_{m+1}, \lambda_{m+1})$ for a parameter value λ_{m+1} that is close to the previous value λ_m.

The simplest natural continuation uses Newton's method to solve $G(\vec{z}_{m+1}, \lambda_{m+1}) = 0$ using $\vec{z}_m$ as an initial iterate. This method does not attempt to incorporate the effects of changing the parameter λ_m to λ_{m+1} in our initial iterate for $\vec{z}_{m+1}$.

The second continuation method, pseudo arclength continuation [4], is useful when continuing around turning points. Turning points are parts of the solution branch where the branch turns around on itself. When a turning point occurs, the Jacobian matrix becomes singular. So applying Newton's method is difficult as we approach the turning point since the Jacobian matrix is becoming singular, making the linear solves for the Newton steps harder. Pseudo arclength continuation handles this problem by augmenting the nonlinear equation $G(\vec{z}, \lambda)$ with an artificial parameter s (the arclength parameter) and an additional arclength equation. So the augmented system we are solving now is

$$\begin{pmatrix} G(\vec{z}(s), \lambda(s)) \\ N(\vec{z}(s), \lambda(s), s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{18}$$

where the first equation specifies that we are on the solution branch, and the second equation specifies the step to take in the parameter s. Suppose we have the point $(\vec{z}_m,\ \lambda_m)$ on the solution curve and the next solution point to be computed is $(\vec{z}_{m+1},\ \lambda_{m+1})$. For the next continuation step, the arclength equation is given by

$$N(\vec{z}(s), \lambda(s), s) = \frac{\partial \vec{z}}{\partial s}(\vec{z}(s) - \vec{z}_m) + \frac{\partial \lambda}{\partial s}(\lambda(s) - \lambda_m) - \Delta s = 0 \tag{19}$$

where Δs is the step taken in the parameter s. A geometric interpretation of the $(\vec{z},\ \lambda)$ points that satisfy $N(\vec{z}(s), \lambda(s), s) = 0$ can be given. Suppose a, b, and c, and are real numbers and (α_0, β_0) is a two-dimensional vector. It is a result from analytic geometry that the two-dimensional vectors (α, β) that satisfy $a(\alpha - \alpha_0) + b(\beta - \beta_0) - c = 0$ lie in the plane perpendicular to the two-dimensional vector (a, b) at a distance away from (α_0, β_0) which is determined by the size of c. Similarly, if $(\vec{z}_{m+1},\ \lambda_{m+1})$ satisfy the arclength equation, then the point will lie in the $(\vec{z},\lambda)$ plane perpendicular to the gradient of $(\vec{z}(s),\ \lambda(s))$ at some distance away from $(\vec{z}_m,\ \lambda_m)$ which is determined by the size of Δs.

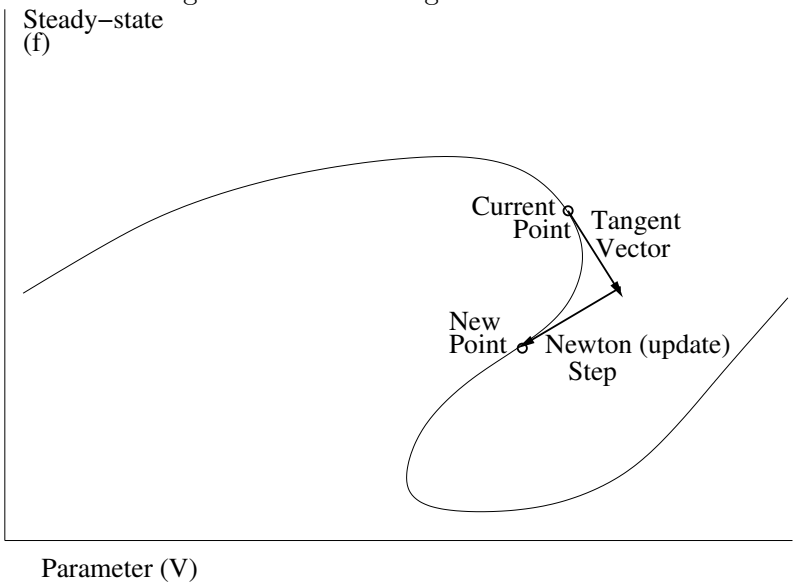
Pseudo arclength continuation solves for $n+1$ variables each time. For natural continuation, the variable we were varying was λ, and for each λ we would compute an n-dimensional solution state $\vec{z}$. This time our parameter is the arclength s, and both the solution state $\vec{z}$ and parameter λ are simultaneously being found. In this way, pseudo arclength continuation is well-suited for handling turning points. The additional orthogonality constraint given in Equation (19) also allows the Newton step to return to the solution curve near a turning point, as shown in Figure 3. Figure 3 shows an example of tracing a one-dimensional system f with a one dimensional parameter v using pseudo arclength continuation.

Returning to our particular application, we can think of the solution vector $\vec{z}$ being $\vec{f} \in R^n$, the finite-dimensional numerical approximation to the Wigner distribution f, the parameter λ as being v, and the nonlinear equation G as $W(\vec{f}(v))$. Since we know that when $v = 0$, the steady-state solution is given by the equilibrium Wigner distribution $f_0(x, k)$, then the first terms in these sequences are $v_1 = 0$ and $\vec{f_1} = f_0(x_i, k_j)$.

6 Linear Solver

The nonlinear solver in the continuation method used for our application was Newton-GMRES. This is an inexact Newton method, where the linear systems for the Newton steps are solved with the Krylov iterative method GMRES [5].

Fig. 3. Pseudo arclength Continuation



One of the advantages of GMRES is that it does not require the storage of the coefficient matrix of the linear system it is solving. GMRES only needs to know the action of the matrix on a vector. At each iteration, GMRES solves a linear least squares problem to compute the next iterate, and this requires GMRES to store a vector at each iteration. To reduce the number of iterations GMRES takes and therefore reduce the computational burden of the simulation, a preconditioner was developed. When solving the linear equation $A\vec{x} = \vec{g}$, where A is a n by n matrix and $\vec{x}$,$\vec{g}$ are n-dimensional vectors, a preconditioner is another matrix M multiplied into the equation (so now we solve $MA\vec{x} = M\vec{b}$) where the new coefficient matrix MA is an easier matrix for an iterative method to handle. Usually, M is an approximate inverse to A. When solving the linear equations in Newton's method, the coefficient matrix is the Jacobian matrix. If we look at Equation (2),and ignore the last two terms, we get the approximation that $W(f) \approx Kf$, leading to $\frac{\partial K}{\partial f} = K$. So an approximation to the Jacobian is $W'(f) \approx K$. Therefore, the preconditioner we use is $M = K^{-1} \approx W'(f)^{-1}$.

The discretized version of K is a block triangular matrix with only three nonzero subdiagonals. So matrix-vector products of K^{-1} can be performed with a vector cost $O(n)$.

K^{-1} is a non-local integral operator, and its discretization, which we do not compute, is a dense block triangular matrix. Since the (x, k) domain is distributed among the processors so that each processor only gets a small portion of x-space and the application of the preconditioner K^{-1} is sequential in x, we should expect to see a performance penalty.

7 LOCA: Library of Continuation Algorithms

To apply these continuation methods to our RTD simulator, we use LOCA (Library of Continuation Algorithms), a software library developed at Sandia National Laboratories [10, 9]. This software library was created for large scale bifurcation and stability analysis. It is a part of Sandia's Trilinos project [3], a collection of Sandia's parallel solver algorithms. Other packages of the Trilinos framework used in this work are the Epetra data structure that aids in parallelization of the code, the NOX nonlinear solver to implement the matrix-free Newton-Krylov algorithm, the AztecOO package for the preconditioned Krylov solver, and the Anasazi iterative eigensolver package.

The implementation of pseudo arclength continuation in LOCA includes a dynamically calculated scaling factor to pre-multiply the second (parameter) term in Equation(19). This is added to assure that both the solution step and parameter step contribute appreciably to the arclength step Δs. There are also several algorithms that directly influence the control of the step size (Δs) at each subsequent step in the continuation procedure so that the solution branch is adequately resolved without requiring too many solutions or experiencing too many convergence failures.

To solve the augmented system of equations in the pseudo arclength algorithm Equation(18), LOCA uses a Newton method that results in a system of order $n + 1$. To keep LOCA independent of any application code's matrix solution algorithm, it uses by default a block decomposition requiring two solves of the order n Jacobian matrix in place of the order $n + 1$ solve of the augmented system. The details can be seen in the manual [10]. A recent implementation of the algorithm in Walker (1999) [12] using a Householder transformation to enforce the linear constraint has been successfully implemented for one common data structure, requiring just one solve of the order n Jacobian matrix to solve the order $n + 1$ system.

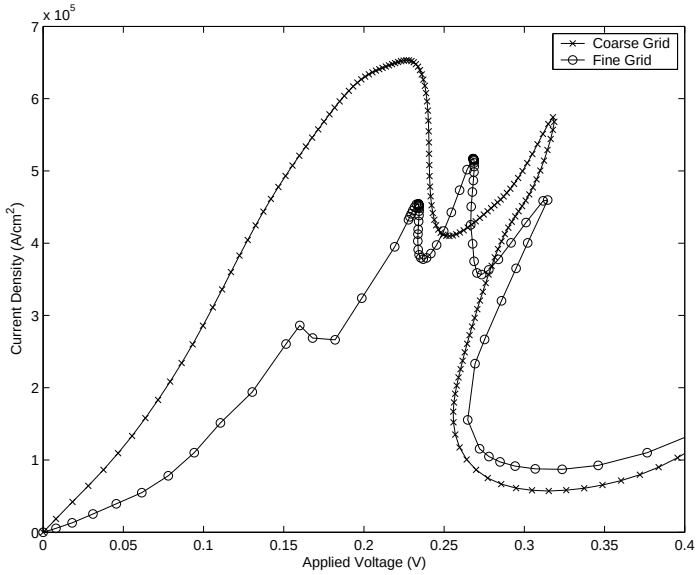
8 Numerical Results

We will now present parallel efficiency and scalability results of our simulation. The runs reported in this section were performed on processors of a Linux cluster at Sandia National Laboratories. This cluster has a total of 236 computing nodes. The nodes are duel 3.06 GHz Xenon processors, each with 2 GB of RAM.

The results for the finer grids yield new and unusual resonant-like features that were not present in the coarse grid studies [14]. Figure 4 is a plot of

the current output versus applied voltage for the coarse grid with a similar plot for a finer grid.The emergence of these features in the more numerically accurate simulations is important because they are not normally observed in experimental measurements on RTD's. Furthermore, we presently believe that this phenomenon is an indicator that parametric values for correlation effects that are presently being used in the code may not be adequate. Therefore, this subject has been targeted for future investigation when time-dependent versions of our parallel code are fully developed.

Fig. 4. Coarse Mesh and Fine Mesh Simulation

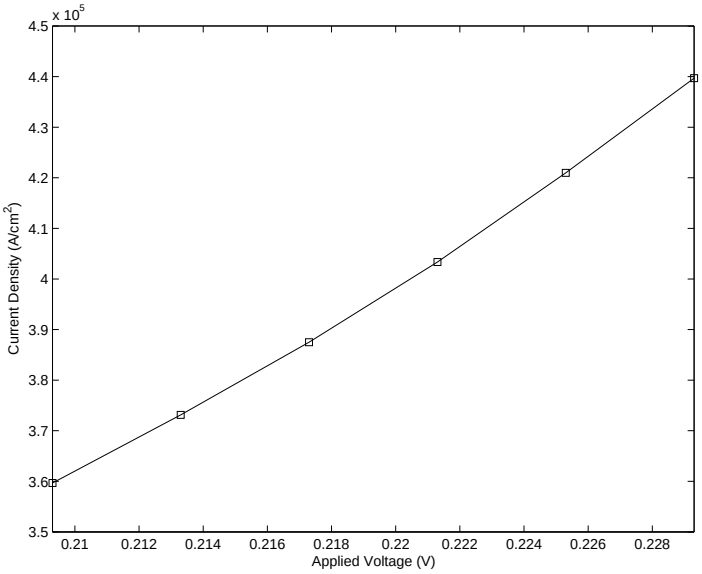


The previous simulation [14] missed the unstable branch between $V = .313$ moving toward $V = .25$ and then on to $V = .318$. The reason was that the work in [14] used a low-accuracy temporal integration to compute steady state solutions, and could therefore only resolve those which were dynamically stable.

Table 1 reports on the parallel efficiency of RTD simulator. The results were computed on for the grid $N_x = 512, N_k = 1024$ and parameter values ranging between $V = 0.2093$ and $V = 0.2293$. We did compare the run times for the entire continuation run ($V = 0$ to $V = 0.4$) since the fine grid requires too much time for a smaller number of processors. Figure 5 shows the part of the current-voltage curve we are using in our efficiency calculation.

To compare the efficiency of the application, we looked at the total linear solve times taken to compute 5 points along the solution-curve, i.e. take 5 continuation steps in the parameter as the number of processors applied to the job is varied. The 5 continuation steps required 11 Newton iterations and, therefore, 11 linear solves. The results produced by the various number of processors were all identical. Since the nodes used to perform the efficiency study are dual processor, we decided a fair evaluation of the efficiency required a

Fig. 5. Efficiency Calculation for Fine Mesh Simulation



base case of 2 processors instead of the normal 1 processor. The communication between 2 processors on the same node would be more efficient than the communication between processors across distinct nodes.

Table 1
Computation of Parallel Efficiency

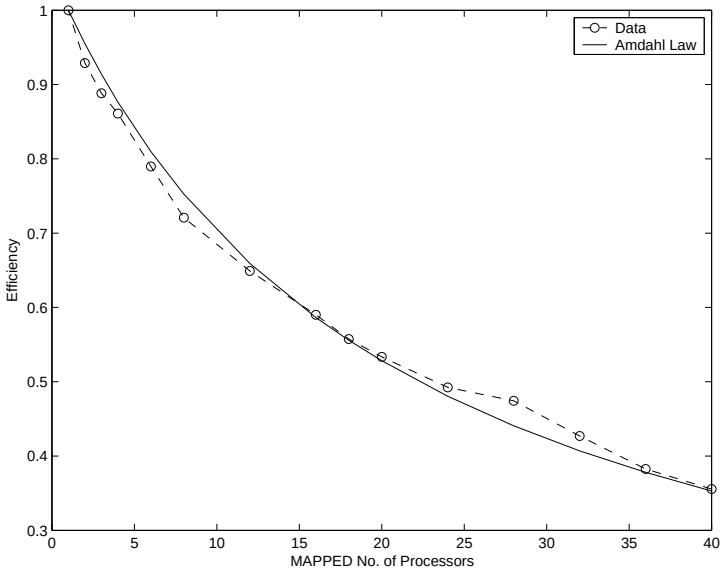
# of Procs.	Linear Solve Time (s)	Efficiency (Percent)
2	9120.61	100 (Base case)
4	4904.46	92.88
8	3422.43	88.83
12	1925.05	78.96
16	1581.53	72.09
24	1171.00	64.91
32	966.06	59.01
40	908.92	55.75
48	771.91	53.34
56	712.25	47.43
64	667.62	42.69
72	662.24	38.26
80	641.39	35.55

Amdahl's Law [1], relates the percent of a code that is serial, the number of processors used, and the corresponding parallel efficiency for these processors. If N is the number of processors, E is the parallel efficiency, and S is the percent of the code that is serial, then Amdahl's Law is

$$E = \frac{1}{NS + (1 - S)} \tag{20}$$

Therefore, Amdahl's Law predicts an inverse relationship between the parallel efficiency of an application and the number of processors used when $S \neq 0$. We can use Amdahl's Law and Table 1 to estimate the fraction of our application that is serial. To do this, we found the value S in Equation(20) that minimizes the sum of the squares of the differences between the efficiency data and the efficiency predicted by Amdahl's Law. This is a nonlinear least squares problem, and a Levenberg-Marquardt code from [6] was used to determine the optimal value of S. This value was $S = 0.047$. So Amdahl's Law predicts we have 4.7 percent serial code. Figure 6 plots the parallel efficiency of our application against the mapped number of processors used along with Amdahl's Law prediction of efficiency when there is 4.7 percent serial code. The figure shows this estimate is accurate.

Fig. 6. Plot of Efficiency and Amdahl's Law Prediction



For the scalability results, we compared the performance of our simulator with a continuation run from $V = 0$ to $V = 0.33$ for three different grids, each using a different number of processors. We start with a base case of $N_x = 172$, $N_k = 144$ and 2 processors, and then simultaneously quadruple both the number of unknowns and the number of processors applied to the problem. We also want to emphasis that at the grids we are simulating, we do not have grid convergence. Therefore, we are not solving the same problem as

the number of unknowns varies, and this must be considered when evaluating these results.

Table 2 shows that the preconditioner is scalable. The number of GMRES iterations for each Newton step and the number of Newton iterations for each continuation step are essentially independent of the mesh.

Table 2
Krylov and Newton Iterations as Mesh Is Refined

N_x	N_k	Avg. Newton Its. Per Cont.Step	Avg. Krylov Its. Per Newton
172	144	3.20	174
344	288	3.26	194
688	576	3.21	198

As we refine the grids, the number of Newton iterations per continuation step and the number of Krylov iterations per Newton iteration are remaining relatively constant which we expect of a scalable preconditioner. Table 3 reports on the scalability of the simulator.

Table 3
Scalability of RTD Simulation

N_x	N_k	Run Time (min)	No. of Cont. Steps	Avg. $W(f)$ Evaluation Time (sec)
172	144	30	34	0.0345
344	288	51	34	0.0591
688	576	112	38	0.1330

Let $n = N_x \times N_k$ be the number of unknowns, and we will assume the serial and parallel work both scale as $O(N^\alpha)$, where α will be a parameter we fit from our data. If we let N be the number of processors, S be the serial fraction of our code, and scale the base time to be 1, then we can model the relative time $T(N)$ for the scalability runs as

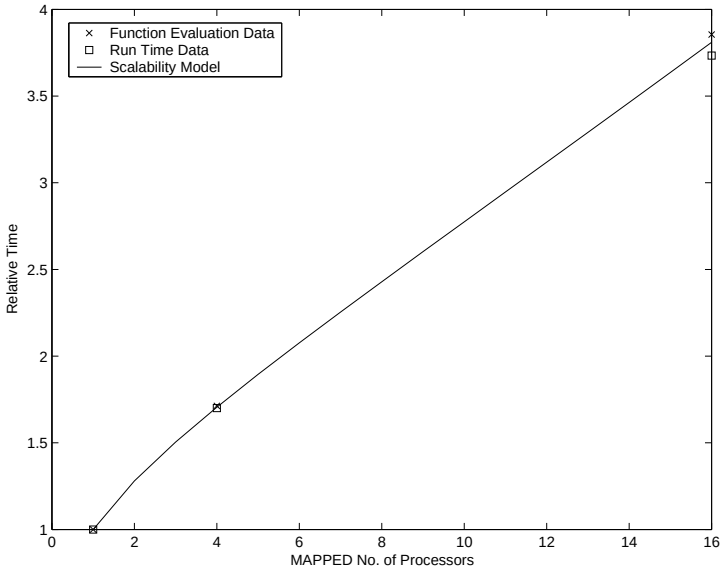
$$T(N) = SN^\alpha + (1 - S)N^{\alpha-1}$$

which we can then use to get an estimate of the serial fraction of our code from the scalability data.

From the scalability data, we have two data points (since the base case will be scaled to $T = 1$ and for all possible parameter values, the model has $T(1) = 1$). Since we have two data points and two parameters to fit, the parameter

estimation is a solution to a nonlinear equation instead of a nonlinear least squares problem. Therefore, we used a Newton code from [5] to solve the two-dimensional problem. We did two separate nonlinear solves with this code, using in one case the total run times and in the second case using the average function evaluation times. For the total run time case, the values of S and α were $S = 4.33$ percent and $\alpha = 1.2948$. The values of S and α were $S = 4.78$ percent and $\alpha = 1.2915$ in the average function evaluation time case. Figure 7 shows the relative times calculated from both the run time data and the function evaluation time data and the model with $S = 4.7$ percent and $\alpha = 1.29$.

Fig. 7. Plot of Relative Time and Model Prediction



9 Conclusion

We have coupled an parallel RTD simulator with LOCA, a continuation package from the Trilinos framework. We have designed a preconditioner that is mathematically scalable, in that the Krylov work for each nonlinear iteration and step in arclength is independent of the mesh. We reported on the parallel efficiency and scalability of our RTD simulator, and by utilizing Amdahl's Law and a model for scalability timings, we estimate that about 5 percent of our code is serial. With the parallel code, we have been able to examine far finer grids than were studied in the previous investigations. Furthermore, these new studies reveal unusual phenomenon and motivate more investigations into underlying physical mechanisms such as electron correlation effects. Indeed, plans are already underway to utilize parallel codes that are being developed under this research project to pursue such physics-based investigations in the future.

Acknowledgements

The research of MSL, CTK, and PZ was supported by a U. S. Army DURINT grant and U. S. Army Research Office grant #W911NF-04-1-0276. The research of CTK was also supported by National Science Foundation grants #DMS-0070641, #DMS-0209695, and #DMS-0404537. Part of this work was done while MSL was an intern at Sandia National Laboratory in Albuquerque, New Mexico.

References

[1] G. M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings vol. 30, Altantic City, NJ, April 18-20, 1967*, pages 483–485. AFIPS, 1967.

[2] F. A. Buot and K. L. Jensen. Lattice Weyl-Wigner formulation of exact many-body quantum-transport theory and applications to novel solid-state quantum-based devices. *Phys. Rev. B*, 42:9429–9457, 1990.

[3] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[4] H. B. Keller. *Lectures on Numerical Methods in Bifurcation Theory*. Tata Institute of Fundamental Research, Lectures on Mathematics and Physics. Springer-Verlag, New York, 1987.

[5] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, PA, first edition, 1995. ISBN 0-89871-352-8 (pbk.).

[6] C. T. Kelley. *Iterative Methods for Optimization*, volume 18 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, PA, first edition, 1999. ISBN 0-89871-433-8 (pbk.).

[7] C.T. Kelley, D.L. Woolard, P. Zhao, M. Kerr, and M.S. Lasater. Parallel-platform bases numerical simulation of instabilities in nanoscale tunneling devices. In *Proceedings of 2002 2nd IEEE Conference on Nanotechnology, Washington DC, August 26-28, 2002*, pages 417–421. IEEE, 2002.

[8] M.S. Lasater, P. Zhao, C.T. Kelley, and D.L. Woolard. Numerical tools for the study of instabilities within the positive-differential-resistance regions of tunneling devices. In *Proceedings of 2003 3rd IEEE Conference on Nanotechnology, San Francisco, CA, August 12-14, 2003*, pages 390–393. IEEE, 2003.

[9] A.G. Salinger, E.A. Burroughs, R.P. Pawlowski, E.T. Phipps, and L.A.

Romero. Bifurcation tracking algorithms and software for large-scale applications. *International Journal of Bifurcation and Chaos*, 2005. in press.

[10] Andrew G. Salinger, Nawaf M. Bou-Rabee, Roger P. Pawlowski, Edward D. Wilkes, Elizabeth A. Burroughs, Richard B. Lehoucq, and Louis A. Romero. Loca 1.0 library of continuation algorithms: Theory and implementation manual. Technical Report SAND2002-0396, Sandia National Laboratory, March 2002.

[11] P. N. Swarztrauber. *Vectorizing the FFTs, in Parallel Computations*. Academic Press, 1982.

[12] H.F. Walker. An adaptation of krylov subspace methods to path following problems. *SIAM Journal on Scientific Computing*, 21:1191–1198, 1999.

[13] E. Wigner. On the quantum correction for thermodynamic equilbrium. *Phys. Rev.*, 40:749–759, 1932.

[14] P. Zhao, H. L. Cui, and D. L. Woolard. Dynamical instabilities and I-V characteristics in resonant tunneling through double barrier quantum well systems. *Phys. Rev. B*, 63:75302, 2001.

[15] P. Zhao, H. L. Cui, D. L. Woolard, K. L. Jensen, and F. A. Buot. Simulation of resonant tunneling structures: Origin of I-V hysteresis and plateau-like structure. *J. Appl. Phys*, 87:1337–1349, 2000.