An Efficient Parallel Solution to the Wigner-Poisson Equations

A. S. Costolanski[1], C. T. Kelley[2], G. W. Howell[3] and A. G. Salinger[4]
[1]Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205
Email: ascostol@ncsu.edu
[2]Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205
Email: tim_kelley@ncsu.edu
[3]Advanced Computing, Office of Information Technology, North Carolina State University, Raleigh, NC 27695-7109
Email: gary_howell@ncsu.edu
[4]Computer Science Research Institute, Sandia National Laboratories, Albuquerque, NM 87123
Email: agsalin@sandia.gov

Keywords: resonant tunneling, nanoscale devices, parallel computation

Abstract

A new model for studying the behavior of nanoscale tunneling devices has been developed in C++ using the Wigner-Poisson formulation. This model incorporates the parallel solvers of Sandia National Lab's Trilinos software with the efficient use of parallel data structures to create a code that scales well to a high number of processors. It also incorporates non-uniform meshes to discretize the solution space and higher order numerical methods to reduce simulation run times and increase numerical accuracy. The improvements inherent in the new C++ model will improve the quality of numerical simulations, and allow longer and more complex nanoscale devices to be modeled.

1. INTRODUCTION

Over the past few decades, electronic devices have become smaller, and the functional demands placed on them greater. The need for efficient micro and nanoscale components to power these devices is immediate, and a variety of research on simulating these ultra-small devices has been undertaken. One such nanoscale device that has the potential to be used as a power source for devices on the micro and nanoscales is the resonant tunneling diode (RTD). RTDs have numerous characteristics that make them interesting to study, and their high speed operation makes them candidates for use as components in high-speed electronic devices. Various models have been developed to study their behavior using both classical and quantum mechanics [1–5]. One of the main models that properly incorporates the quantum effects inherent in these nanoscale devices is the Wigner-Poisson formulation [6–22, 24].

The Wigner equation [25] is an integro-differential equation for which a solution cannot be computed analytically, so numerical techniques must be used [26]. The Wigner equation is often coupled with the Poisson equation [6] to account for potential effects in the device. Solving the non-linear Wigner equation using a traditional Newton method is extremely difficult, since the Jacobian is dense and requires significant memory usage as finer grids are implemented. Previous attempts to model resonant tunneling behavior using the Wigner-Poisson formulation have produced simulation results with either a low degree of computational accuracy or simulation run times that are unreasonably long [12–24]. The goal of this work is to develop a research model that will produce simulation results with a high degree of numerical accuracy combined with reasonable run times, so that a larger variety of devices can be modeled.

This paper will present details of a new Wigner-Poisson model we have written in C++ and built with the Trilinos [27] package. The use of Trilinos allows a distributed memory implementation without explicit use of MPI message passing. The new model uses higher order numerical methods than previous FORTRAN versions, and improves run times over previous parallel implementations by the use of a new non-uniform grid structure. We also improved upon previous scalability results by effectively partitioning the solution vector into two different formats rather than one, allowing every piece of the Wigner and Poisson equations to be computed in parallel.

1.1. Resonant Tunneling Diode Structure

Resonant tunneling diodes have been studied for over 30 years [7, 28–30] due to their interesting physical characteristics and their potential device applications. They have response times on the order of picoseconds and high frequency output (on the order of THz), but their ability to produce negative differential resistance led to an increase in research on these devices.

Negative differential resistance is the effect of quantum tunneling in the middle of the device that allows current to flow more freely in proportion to an increase in voltage, up to a certain critical voltage value. As the voltage is increased past the critical value, quantum effects bar the passage of electrons between the semiconductor materials in the well, and the corresponding current values decrease drastically down to a relative minimum before increasing again. This effect is seen experimentally for voltage changes in both directions: voltage increases (from $V = 0$ to a critical value V_I) and decreases (from beyond a critical value $V = V_D$ down to 0) where $V_I > V_D$. Plotting the steady-state current-voltage relationship shows multiple values for the current in the voltage range $V \in (V_D, V_I)$. This effect is called hysteresis, and this behavior should be present in RTD simulations in order for a numerical model to be valid. (See figure 5 in section 3. that shows these hysteretic effects are present in the RTD simulation model results.)

The structure of a typical RTD incorporates two different types of semiconductor materials with differing energy band gaps to create a quantum well in the center of the device. A large region at each end of the device is infused with dopants to facilitate current flow when a

voltage is applied across the device. The material parameters shown in figure 1 are those used for the numerical simulations discussed in this paper.

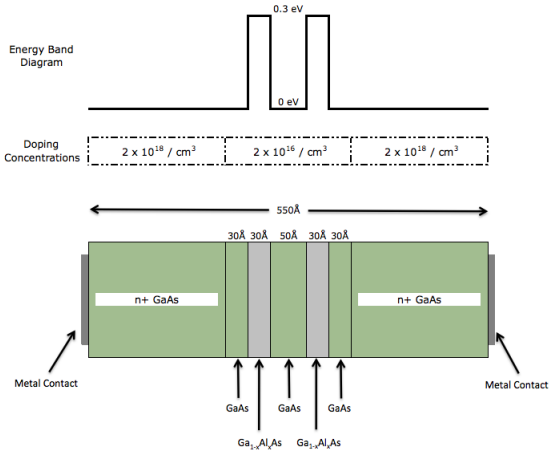

Figure 1. Sample material parameters and device structure for a two barrier resonant tunneling diode.

1.2. The Wigner-Poisson Equations

The Wigner equation models electron transport and is used along with Poisson's equation as a method to approximate device behavior. The RTD is modeled as a two-dimensional system, with a spatial dimension and a momentum dimension. The Wigner equation is

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

where $f = f(x,k,t)$ is the distribution of electrons in the device, x is the spatial parameter, k is the momentum parameter, and t represents time. The components of the Wigner equation are $K(f)$, which represents the kinetic effect of the electrons in the device; $P(f)$, which models the potential energy effects inside the device; and $S(f)$, the scattering term, which takes into account the interactions between electrons within the device. The kinetic term $K(f)$ is

$$K(f) = -\frac{hk}{2\pi m^*}\frac{\partial f(x,k,t)}{\partial x} \tag{2}$$

where h is Planck's constant and m^* is the effective mass of an electron in the device. The potential term is

$$P(f) = -\frac{4}{h}\int_{-\infty}^{\infty} f(x,k',t)T(x,k-k',t)dk' \tag{3}$$

with

$$T(x,z,t) = \int_0^{\frac{L_c}{2}} [U(x+y,t) - U(x-y,t)]\sin(2xz)dy. \tag{4}$$

In $T(x,z,t)$, L_c is called the correlation length of the device and satisfies $L_c \leq L$, where L is the total device length. The correlation length represents the maximum distance that an electron feels the effects of other electrons in the device. $U(x,t)$ is the potential energy and can be written as

$$U(x,t) = \Delta_c(x) + u_p(x,t) \tag{5}$$

where $\Delta_c(x)$ is the time-independent energy band function defined by the barriers in the device, and $u_p(x,t)$ is the electrostatic potential, which is the solution to Poisson's equation

$$\frac{\partial^2 u_p(x,t)}{\partial x^2} = \frac{q^2}{\varepsilon}[N_d(x) - n(x,t)]. \tag{6}$$

In Poisson's equation, q represents the charge on an electron; ε is the dielectric constant of the primary semiconductor material; $N_d(x)$ is the concentration of the ionized dopants in the device (assumed to be time-independent); and $n(x,t)$ is the electron density in the device, defined by

$$n(x,t) = \int_{-\infty}^{\infty} f(x,k,t)dk. \tag{7}$$

Poisson's equation imposes boundary conditions to account for the voltage change V across the device:

$$u_p(0,t) = 0, \;\; u_p(L,t) = -V. \tag{8}$$

Finally, the last term in the Wigner equation is the scattering term $S(f)$, defined from the first-order relaxation time approximation [8] as

$$S(f) = \frac{1}{\tau}\left[\frac{\int_{-\infty}^{\infty} f(x,k,t)dk}{\int_{-\infty}^{\infty} f_0(x,k)dk} f_0(x,k) - f(x,k,t)\right] \tag{9}$$

where τ is the relaxation time constant for the semiconductor material, and $f_0(x,k)$ is the initial Wigner distribution at $V = 0$ and is assumed to be time-independent.

The Wigner equation also has boundary conditions imposed which depend on the sign of the momentum:

$$f(0,k) = \frac{4\pi m^* k_B T}{h^2}\ln\left\{1 + e^{-\frac{1}{k_B T}\left(\frac{\hbar^2 k^2}{8\pi^2 m^*} - \mu_0\right)}\right\}, \; k > 0 \tag{10}$$

$$f(L,k) = \frac{4\pi m^* k_B T}{h^2}\ln\left\{1 + e^{-\frac{1}{k_B T}\left(\frac{\hbar^2 k^2}{8\pi^2 m^*} - \mu_L\right)}\right\}, \; k < 0 \tag{11}$$

where k_B is Boltzmann's constant, T represents the temperature of the device, and μ_0 and μ_L are the fermi energies at each end of the device.

1.3. Previous Versions

Earlier versions of the Wigner-Poisson model [9, 14] were written in FORTRAN and used second-order numerical approximations to simulate device performance. Due to computational limitations of the hardware available at the time, these versions only used very coarse grids, which limited numerical accuracy. A later version of the FORTRAN code [19–22] incorporated the Trilinos software [27], which was a significant enhancement over previous versions. Trilinos' parallel capabilities and more advanced solver methods significantly decreased run times, which in turn allowed finer grids to be used.

However, low-order numerical methods continued to limit computational accuracy. In addition, the uniform grids used by the FORTRAN model limited how far the mesh could be refined if reasonable run times were desired. Even for a large number of processors

(over 100), run times could stretch from hours into weeks as the grids grew progressively finer.

For a large portion of the domain, the Wigner function $f \approx 0$. This can be seen in figure 2, which shows a sample Wigner distribution function f with zero bias applied across the device. For $|k| \geq 0.15\mathring{A}^{-1}$, $f \approx 0$. Thus, when a uniform grid is used, a large percentage of the calculations to compute the solution to the Wigner equation provides no meaningful information.

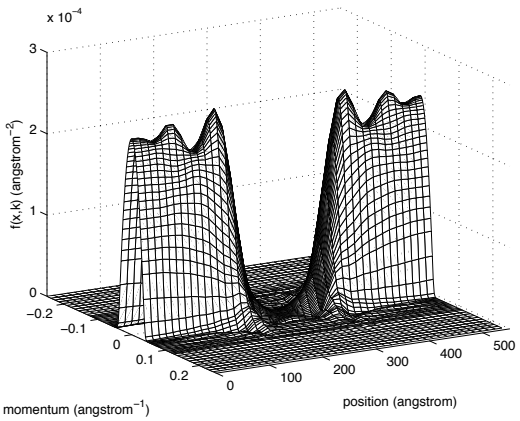

Figure 2. Zero bias Wigner distribution for a 550 Å device.

A more recent version of the Wigner-Poisson model [24], written in MATLAB, attempted to fix this limitation by employing non-uniform grids to discretize the domain. The MATLAB code also incorporated higher-order numerical methods and solved several integrals analytically to increase numerical accuracy. These improvements reduced run times significantly over the FORTRAN version and allowed finer grids to be simulated. However, the MATLAB code is serial and limited to specific device types, so a newer version of the Wigner-Poisson model was needed.

1.4. New C++ model

Thus, we have written a new Wigner-Poisson model in C++ that includes the higher-order numerical methods and analytical solutions from the previous MATLAB code, along with parallel computation and a new non-uniform grid implementation to decrease run times significantly as compared to all previous versions of the Wigner-Poisson formulation. Unlike previous parallel implementations [19–22] which applied the preconditioner serially, the new C++ model parallelizes every piece of the Wigner and Poisson equations. This required discretizing the solution vector in two separate ways and coordinating the transfer of data between the two discretizations, which resulted in improved scalability over the previous parallel versions.

The Trilinos software [27], developed by Sandia National Laboratories, was incorporated into the C++ model to solve the non-linear Wigner equation due to the variety of capabilities it includes, such as flexible parallel data structures, non-linear solvers and continuation. Also incorporated is the interpolation package from Alglib [31], an open-source numerical analysis library that supports several programming languages, including C++. Alglib interpolation handles conversions from a non-uniform grid to a uniform grid, and was chosen due to its ease of implementation and ability to be compiled across multiple Unix and Linux platforms.

1.5. Numerical Discretizations

Since the coupling of the Wigner equation with Poisson's equation yields a complex system which cannot be solved analytically, appropriate numerical methods must be employed to solve the system and produce a physically realistic numerical solution.

1.5.1. Discretization of the domain

The momentum domain is truncated from $(-\infty,\infty)$ to $(-K_{max},K_{max})$, where K_{max} is chosen such that for $|k| > K_{max}$, $f(x,k,t) \approx 0$ for all x,t. For the simulations discussed in this paper, we have used $K_{max} = 0.25$ Å. This choice is based on earlier work [19, 26] and is validated by figure 2, which demonstrates for $|k| > 0.15$ Å, $f \approx 0$.

Next, the non-uniform meshes for the spatial and momentum domain are created. We denote N_x as the number of grid points for a uniform grid in the x dimension, and N_k as the number of grid points for a uniform grid in the k dimension. The actual number of grid points for the non-uniform meshes will be less than or equal to N_x and N_k respectively.

The spatial mesh grid points (discretized into n points from $x = 0$ to $x = L$) are $\{0 = x_1,\ x_2,\ \ldots,\ x_{n-1},\ x_n = L\}$ where $n \leq N_x$, and the momentum mesh grid points (discretized into m mesh points) are $\{-K_{max} = k_1,\ k_2,\ \ldots,\ k_{m-1},\ k_m = K_{max}\}$ where $m \leq N_k$. Dropping the time dependence for the Wigner distribution function f, we denote the grid points for f as $f_{ij},\ 1 \leq i \leq n,\ 1 \leq j \leq m$ where the first index represents the spatial dimension and the second represents momentum. The details of how the non-uniform grids are created will be discussed in section 1.5.3.

1.5.2. Discretization of the Equations

The Wigner-Poisson equations are discretized using finite difference methods and Newton-Cotes quadrature rules. Wherever possible, each term uses a fourth order approximation.

The kinetic term, $K(f)$ (equation 2), is approximated using a fourth order upwinding approximation for the first derivative. For $k_j > 0$, it is:

$$K(f_{ij}) = -\frac{hk_j}{2\pi m^*}\left(\frac{25f_{ij} - 48f_{i-1,j} + 36f_{i-2,j} - 16f_{i-3,j} + 3f_{i-4,j}}{12\Delta x}\right) \quad (12)$$

and for $k_j < 0$,

$$K(f_{ij}) = -\frac{hk_j}{2\pi m^*}\left(\frac{-25f_{ij} + 48f_{i+1,j} - 36f_{i+2,j} + 16f_{i+3,j} - 3f_{i+4,j}}{12\Delta x}\right) \quad (13)$$

where Δx depends on the size of the spatial mesh at each value of k. However, close to the boundaries at $x = 0$ and $x = L$, there are not enough grid points for a full fourth order stencil, so decreasing order upwinding approximations are used.

Note that the differing approximations for the kinetic term produce a discontinuity in the solution at $k = 0$, so the discretization of the momentum domain must be chosen so that $k = 0$ is not a grid point. The discretization of the spatial domain can change for

different values of k, but the mesh size Δx at a particular k must be constant due to the finite differencing of the kinetic term. These conditions will be used to determine the non-uniform grid points, and the specific structure of the non-uniform grids will be described in the next section.

The potential $P(f)$ term (equation 3) is discretized using composite Newton-Cotes quadrature rules

$$P(f_{ij}) \approx -\frac{4}{h}\sum_{j'=1}^{N_k} f_{ij'}T(x_i, k_j - k_{j'})w_{j'} \tag{14}$$

The $w_{j'}$ are the weights of the appropriate quadrature rule in each grid region. Since $k = 0$ cannot be a grid point, the composite midpoint rule is used in the mesh region in the interior of the device around $k = 0$. Away from the $k = 0$ line, however, the grid points in the region(s) can be chosen so that a fourth order composite Simpson's rule can be implemented.

To compute $T(x_i, k_j - k_{j'})$ (equation 4), we split the term into two pieces so $T(x_i, z) = T_1(x_i, z) + T_2(x_i, z)$ where $z = k_j - k_{j'}$ and

$$T_1(x_i, z) = \int_0^{\frac{L_c}{2}} [\Delta_c(x_i + y) - \Delta_c(x_i - y)] \sin(2x_i z) dy \tag{15}$$

$$T_2(x_i, z) = \int_0^{\frac{L_c}{2}} [u_p(x_i + y) - u_p(x_i - y)] \sin(2x_i z) dy \tag{16}$$

Since the energy band function $\Delta_c(x) = 0$ for a large portion of the x domain, the integral T_1 can be computed analytically. T_2 must be computed numerically using quadrature methods. While composite Simpson's rule is a preferred method, note that the upper limit of the integrand involves L_c, which may or may not correspond to a grid point in the spatial domain. Thus, the weights for the discretization must be modified to take this into account.

Assume the value of L_c falls between the values corresponding to grid points N_c and $N_c + 2$, where N_c is even. We can divide the integral into two pieces, an integral from 0 to x_{N_c} and another from x_{N_c} to L_c, and then use composite Simpson's rule to approximate the integral over $[0, x_{N_c}]$ and Simpson's rule for the integral over $[x_{N_c}, L_c]$. To compute the weights for the integral over $[x_{N_c}, L_c]$, Taylor expansions are used to compute fourth order approximations to the weights between $N_c - 1$ and $N_c + 2$, depending on exactly where L_c falls in the $[x_{N_c}, x_{N_c+2}]$ interval. So equation (16) can be approximated as

$$T_2(x_i, z) \approx \sum_{i'=1}^{N_c+1} \left[u_p(x_i + x_{i'}) - u_p(x_i - x_{i'})\right] \sin(2x_{i'} z) w_{i'} \tag{17}$$

where the $w_{i'}$ are the modified composite Simpson's rule weights for $1 \le i' \le N_c + 1$, with additional fourth order weighting terms added for $N_c - 1 \le i' \le N_c + 2$.

The scattering term $S(f)$ (equation 9) can be discretized using composite Newton-Cotes quadrature rules as

$$S(f_{ij}) \approx \frac{1}{\tau}\left[\frac{f_0(x_i, k_j)}{\sum_{j'=1}^{N_k} f_0(x_i, k_{j'}) w_{j'}} \sum_{j'=1}^{N_k} f_{ij'} w_{j'} - f_{ij}\right] \tag{18}$$

where the $w_{j'}$ are the composite midpoint and Simpson's rule weights as described for the $P(f_{ij})$ term (equation 14).

To solve Poisson's equation (6), we can split $u_p(x,t)$ into two pieces, $u_p(x,t) = u_p^a(x,t) + u_p^b(x,t)$, such that

$$\frac{d^2 u_p^a(x,t)}{dx^2} = \frac{q^2}{\varepsilon} N_d(x) \quad \text{and} \quad \frac{d^2 u_p^b(x,t)}{dx^2} = -\frac{q^2}{\varepsilon} n(x,t) \tag{19}$$

solve the first part of equation (19) analytically and the second part using a finite difference approximation, and then sum the two pieces to obtain $u_p(x,t)$. This method not only saves computational time but produces a more accurate solution. To solve the second part of equation (19), a fourth order center difference formula is used to approximate the second derivative of $u_p^b(x,t)$ except at the end points, where one-sided fourth order methods are used with $u_p(x_1) = 0$ and $u_p(x_n) = -V$. The electron density $n(x)$ is approximated using composite Newton-Cotes quadrature:

$$n(x_i, t) \approx \frac{1}{2\pi}\sum_{j=1}^{N_k} f_{ij} w_j \tag{20}$$

where the weights are as listed for the potential term $P(f)$ (equation 14).

Finally, the current density $j(x)$ can also be approximated using composite Newton-Cotes quadrature:

$$j(x_i) \approx \frac{h}{2\pi m^*}\sum_{j=1}^{N_k} k_j f_{ij} w_j \tag{21}$$

where the weights are again as listed for the potential term $P(f)$ (equation 14).

1.5.3. Non-Uniform Grids

As noted above, the discretization of the kinetic term forces certain requirements for constructing a non-uniform mesh in both the spatial and momentum dimensions. Namely, (1) $k = 0$ cannot be used as a grid point, and (2) the spatial grid must be uniform for a given value of k.

Since the spatial grids depend on the value of k, the discretization of the momentum space will be constructed first, and then the spatial grids will be determined.

Momentum non-uniform grid The non-uniform momentum grid must contain a region symmetric about $k = 0$ that does not include $k = 0$. The size of this region is minimized so that the impact of the error associated with the second order quadrature rule is minimized.

The remaining momentum regions are symmetric about $k = 0$. The grid spacing Δk in an interior (closer to $k = 0$) region is equal to $\frac{1}{2}$ the grid spacing in the adjacent exterior (away from $k = 0$) region. The k grid points are assigned to each region to minimize the total number of grid points without sacrificing accuracy of the solution.

Spatial non-uniform grid Once the momentum regions are chosen, the spatial grids are determined for each momentum region based on similar rules: (1) a minimum number of spatial grid points per region is chosen, (2) each spatial region must include at least the minimum number of k grid points, and (3) the mesh spacing Δx for an interior grid region equals $\frac{1}{2}$ the grid spacing in the adjacent exterior region.

Note that for a single momentum region, it is possible to have two different spatial grid regions if there are a large number of k grid points in the momentum region.

An example of the non-uniform mesh is given in figure 3.

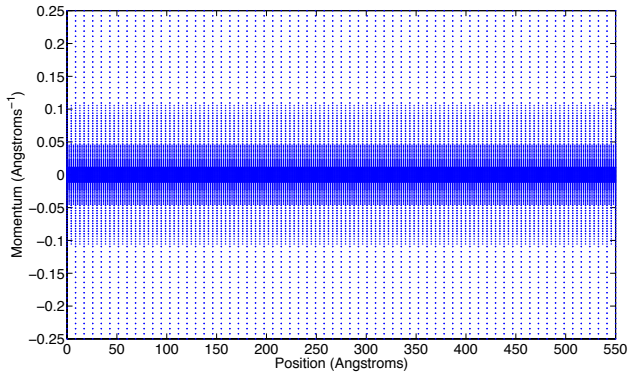

Figure 3. Example of the C++ model non-uniform grid.

1.6. Solution Method

To begin the simulation, an initial guess is made for the Wigner distribution function f at $V = 0$, which is then fed to the nonlinear solver package NOX to compute the initial Wigner distribution function f_0. To compute the steady-state simulation and produce a current-voltage (I-V) curve, continuation on the voltage parameter V is used to compute f for various values of the voltage from $V = 0$ to $V = 0.45$.

2. PARALLEL IMPLEMENTATION

One goal of the C++ Wigner-Poisson model is to decrease run times by using parallel computation. We used three approaches: (1) use of the Trilinos software; (2) balanced allocation of vectors across processors; and (3) parallel computation of terms in the Wigner and Poisson equations.

2.1. Trilinos Software

The Trilinos software [27] offers many parallel packages that allow the user to distribute computations without having to explicitly use MPI message passing calls. We used version 10.6.0, which was released in September 2010 and includes 48 packages. The primary packages incorporated in the Wigner-Poisson model were Epetra (data structures), Amesos (direct solvers for sparse matrices), NOX (non-linear solvers), and LOCA (continuation methods).

Epetra: Epetra is the core linear algebra package upon which several other packages are based. The Epetra data structures are designed to efficiently use BLAS and LAPACK routines while simplifying parallel computation. Thus, most of the vector and matrix elements used by the C++ model were defined using Epetra data structures.

Amesos: The Amesos package provides a variety of sparse direct solvers [32] which can be used for Poisson's equation, and a serial implementation proved to be the most efficient method due to the relatively smaller problem size.

NOX: The NOX package uses Newton's method to solve nonlinear equations, and provides line search and trust region methods as well as exact and inexact Newton methods to tailor the choice of algorithm. In addition, Jacobians can be evaluated via several finite differencing methods or via a Jacobian-free implementation, the latter of which was used in the Wigner-Poisson code due to the dense nature of the Jacobian.

LOCA: The continuation package LOCA [33, 34] is built using the non-linear solver package NOX and provides both natural and arc length continuation methods [35] as well as the ability to locate and track several types of bifurcations. For the Wigner-Poisson model, hysteretic effects are present in the I-V curve [36, 37], so LOCA uses an arc length continuation method to choose the voltage step.

2.2. Parallelization of the Wigner vector

For a non-uniform grid in both space and momentum as described in section 1.5.3. (see figure 3 for an example), the number of x grid points corresponding to each value of k (and visa versa) will change depending on the value of k (or x). Thus care must be taken to balance the number of calculations (i.e., total grid points) assigned to each processor.

The solution vector f is partitioned in two ways, once across k, and separately across x. Due to the discretization of the equations, the vector f partitioned across k is used to compute the kinetic term (equations 12 and 13), with the other partitioning (f across x) used in preconditioning and also for internal portions of the potential term (equations 14 through 17). Cross-processor communication is necessary to switch from one set of partitioning to the other, but this takes place only once per computation of the Wigner function. The efficiency of performing all major calculations in parallel outweighs the additional communication time.

A standard method for distributing elements across processors is to allocate the elements evenly, with any remainder allocated to the first processors. However, this method may lead to additional cross-processor communication as neighboring elements in the grid end up on different processors. This can be seen in figure 4, where the left picture shows how the grid points associated with the k_4 and k_5 values end up on two different processors. Thus, to partition f across each grid efficiently, we allocated the elements so that no cross-processor communication was necessary, while keeping the total number of grid points on each processor as balanced as possible.

To parallelize over k, we calculated an average number of total grid points per processor, and then allocated the k grid points to each processor so that the total number of associated x grid points would be as close to the average as possible. The right picture of figure 4 demonstrates not only the lack of cross-processor communication, but also an even distribution of elements. A similar method is employed to distribute the Wigner vector over the x grid. This rounding method based on the total grid points in the solution domain (rather than just the number of k or x grid points) ensures that the number of calculations will be as balanced as possible across processors.

2.3. Efficient Parallel Code

Conversions between the momentum k and x space parallel forms of the Wigner function f are necessary but minimized in the computation of $W(f) = 0$. NOX and LOCA use the distributed (over k)

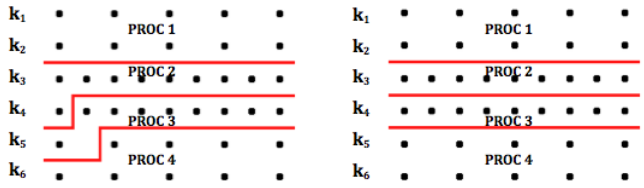

Figure 4. A sample non-uniform grid allocated across processors. The figure on the left represents an allocation that leads to additional cross-processor communication; the figure on the right demonstrates a more balanced distribution.

vector f to locally compute successive f iterates.

Equation (15) and the interior portion of equation (16) involving only the sine term are computed and stored for later use. As the meshes are refined and the overall problem size becomes exceedingly large, memory limitations may increase run times if too much information is stored on each processor, so these terms as well as most components of $W(f) = 0$ are computed in parallel. The solution to Poisson's equation, due to the small problem size, is computed on each processor.

3. NUMERICAL RESULTS

We tested the accuracy, speed and scalability of the C++ code. Simulation runs were performed on a 240 processor cluster at the NCSU High Performance Computing Center. Quad core Xeon blades were used with Infiniband switches, along with Intel compilers for optimum performance. Trilinos version 10.6.0 was incorporated into the C++ code along with Alglib version 3.6.0.

To ensure the accuracy of the C++ model, we compared steady state I-V curves for the previous FORTRAN and MATLAB codes against the C++ model to determine whether the hysteretic effects seen in previous versions between $V \approx 0.25$ and $V \approx 0.30$ were still present. The agreement between the solution curves is very close, especially between MATLAB and C++. The largest difference between the MATLAB and C++ curves is due to the more stringent parameter settings for the C++ code, which limited the voltage step size for the continuation, revealing greater detail. Note that the current vs. voltage curve can be expressed neither as a function of current nor of voltage, nor do tangent lines appear to exist at all points of the curve. See figure 5.

Simulation run times for the C++ code were compared against those for FORTRAN, MATLAB, and the uniform grid C++ model to determine how much the incorporation of non-uniform grids improved computation time. For the parallel FORTRAN model, the decrease in run times using the same number of processors was significant (see table 1 for an example). The MATLAB code is a highly efficient serial code, and while run times were lower for the non-uniform MATLAB code (on the order of $\approx$ 15 hours) than for the serial non-uniform C++ code (on the order of $\approx$ 45 hours), the ability to increase the number of processors allows the C++ code to return results more quickly by choosing an appropriate number of cores. And for the C++ code comparison, although the incorporation of a non-uniform spatial grid requires interpolation for the Wigner vector not required with a uniform grid, the decreased number of computations more than offsets the addition of the interpolation. The results shown in table 2 for different mesh sizes demonstrate these results were not grid-dependent.

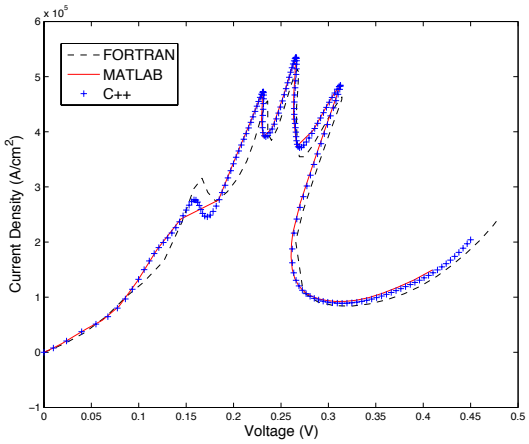

Figure 5. Comparison of I-V curves using the FORTRAN, MATLAB, and C++ models.

TABLE 1 – FORTRAN vs C++						
No.	*Grid*		*Time (hr:min)*		*% Reduction*	
Procs	N_k	N_x	FORT	C++	Time	Grid Pts
20	512	513	17:17	2:52	83.4	90.4

TABLE 2 – Uniform vs Non-uniform C++						
No.	*Grid*		*Time (hr:min)*		*% Reduction*	
Procs	N_k	N_x	Unif	NonU	Time	Grid Pts
48	512	513	5:39	1:44	69.3	90.4
24	256	257	1:11	0:22	69.0	85.2

We also analyzed the scalability of the C++ model. The study shown in figure 6 demonstrates the strong scalability of the C++ code for simulation runs using a fine mesh of $N_x = 513$, $N_k = 512$ with the number of processors increased from 2 up to 64. Using 2 processors as a base, the speedups calculated from the simulation runs show very good adherence to Amdahl's law

$$\text{speedup} = \frac{1}{\alpha + \frac{(1-\alpha)}{P}} \tag{22}$$

where α represents the amount of serial code and P the number of processors.

Additional studies were run with much finer grids to determine if Amdahl's law continues to be accurate in predicting the speedup of the C++ code as the problem size is increased. The results in figure 7 using a very fine mesh of $N_x = 1025$, $N_k = 2048$ and 8 processors as a base show that adherence to Amdahl's law is still good, although superlinear convergence is noted when the number of processors used is less than 32. This most likely indicates that the size of the problem exceeds efficient memory capabilities of the blades, and that larger numbers of processors should be used for very fine meshes. We feel there is room for improvement and are examining trade-offs between redundant calculations and lessened global communications.

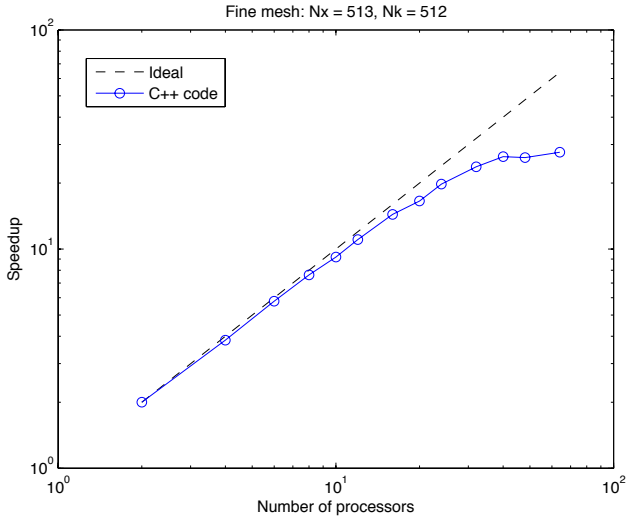

Figure 6. Ideal vs. computed speedup (using 2 processors as a base) for a fine mesh of $N_x = 513$, $N_k = 512$. The maximum speedup is around 28, achieved at 64 processors.

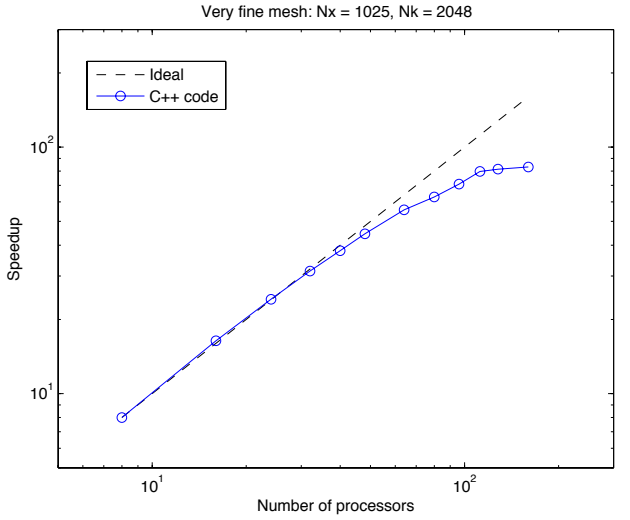

Figure 7. Ideal vs. computed speedup (using 8 processors as a base) for a fine mesh of $N_x = 1025$, $N_k = 2048$. The maximum speedup is around 83, achieved at 160 processors.

4. CONCLUSIONS AND FUTURE WORK

A more efficient Wigner-Poisson model has been developed in C++ that effectively utilizes parallel computation, non-uniform meshes, and higher order numerical methods to produce solutions that are more numerically accurate and have decreased run times as compared to previous versions. Using different partitioning to solve the Wigner equation, along with the incorporation of the Trilinos data structures and highly efficient solvers, results in a research model that scales well.

Further work is in improving scalability to allow efficient use of thousands of processors. Also, this code will allow additional flexibility in the types of device structures being modeled, which provides more insight into the design and behavior of nanoscale tunneling devices. Possible future work includes modeling longer device lengths similar to or exceeding those in [24] as well as multiple barrier structures similar to those in [23].

5. ACKNOWLEDGEMENT

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF0710112.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

REFERENCES

[1] M.G. Ancona and G.J. Iafrate. "Quantum correction of the equation of state of an electron gas in a semiconductor". Physical Review B, v.39 (1989).

[2] P. Degond, S. Gallego, and F. Méhats. "Simulation of a resonant tunneling diode using an entropic quantum drift-diffusion model". J. Comput. Electron, v. 6 (2007), pp. 133–136.

[3] C. de Falco, E. Gatti, A.L. Lacaita, and R. Sacco. "Quantum-corrected drift-diffusion models for transport in semiconductor devices". Journal of Computational Physics, v. 204 (2005).

[4] A. Jungel and R. Pinnau. "A positivity preserving numerical scheme for a fourth order parabolic equation". SIAM Journal of Numerical Analysis, v. 39 (2001).

[5] A. Pirovano and A. Lacaita and A. Spinelli. "Two-dimensional quantum effects in nanoscale MOSFETs". IEEE Tran. Electr. Devices, v. 47 (2002).

[6] N. C. Kluksdahl, A. M. Kriman, D. K. Ferry and C. Ringhofer. "Self-consistent study of the resonant-tunneling diode". Phys. Rev. B, v. 39 (1989), pp. 7720–7735.

[7] L.L. Chang and L. Esaki and R. Tsu. "Resonant tunneling in semiconductor double barriers". Applied Physics Letters, v. 24 (1974), pp. 593–595.

[8] P. Borbone, M. Pascoli, R. Brunetti, A. Bertoni, and C. Jacoboni. "Quantum transport of electrons in open nanostructures with the Wigner-function formalism". Phys. Rev. B, v. 59 (1999), pp. 3060–3069.

[9] 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., v.42 (1990), pp. 9429–9457.

[10] K. L. Jensen and F. A. Buot. “Numerical Simulation of Intrinsic Bistability and High-Frequency Current Oscillations in Resonant Tunneling Structures”. Physical Review Letters, v. 66, no. 8 (1991), pp. 1078–1081.

[11] B. A. Biegel and J. D. Plummer. “Comparison of self-consistency iteration options for the Wigner function method of quantum device simulation”. Phys. Rev. B, v. 54 (1996), pp. 8070–8082.

[12] D. L. Woolard, P. Zhao, and H. L. Cui. “THz-Frequency Intrinsic Oscillations in Double-Barrier Quantum Well Systems”. Physica B, v. 314 (2002), pp. 108–112.

[13] 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, v. 63 (2001), p. 75302.

[14] P. Zhao, H. L. Cui, D. L. Woolard, K. L. Jensen and F. A. Buot. “Simulation of resonant tunneling structures: Origin of the I-V hysteresis and plateau-like structure”. Journal of Applied Physics, v.87 (2000), pp. 1337–1349.

[15] P. Zhao, D. L. Woolard, and H. L. Cui. “Multisubband Theory for the Origination of Intrinsic Oscillations Within Double-Barrier Quantum Well Systems”. Physical Review B, v. 67 (2003), pp. 085312.

[16] P. Zhao, D. L. Woolard, B. L. Gelmont, and H. L. Cui. “Creation and quenching of interference-induced emitter-quantum wells within double-barrier tunneling structures”. Journal of Applied Physics, v. 94, no. 3 (2003), pp. 1833–1849.

[17] P. Zhao, D. L. Woolard, M. S. Lasater, C. T. Kelley, and R. Trew. “Terahertz-Frequency Quantum Oscillator Operating in the Positive Differential Resistance Region”. Proceedings of SPIE, v. 5790 (2005), pp. 289–300.

[18] P. Zhao, D. L. Woolard, H. L. Cui, and N. J. M. Horing. “Origin of Intrinsic Oscillations in Double-Barrier Quantum Well Systems”. Physics Letters A, v. 311 (2003), pp. 432–437.

[19] M. S. Lasater. “Numerical Methods for the Wigner-Poisson Equations”. Ph.D. dissertation, North Carolina State University, Raleigh, North Carolina, 2005.

[20] M. S. Lasater, C. T. Kelley, A. G. Salinger, D. L. Woolard, and P. Zhao. “Simulating Nanoscale Semiconductor Devices”. Intl. Jour. High Speed Elect. & Sys., v. 16, no. 2 (2006), pp. 677–690.

[21] M. S. Lasater, C. T. Kelley, A. G. Salinger, D. L. Woolard, and P. Zhao. “Enhancement of Numerical Computations of the Wigner-Poisson Equations for Application to the Simulation of THz-Frequency RTD Oscillations”. Proceedings of SPIE, v. 5584 (2004), pp. 42–51.

[22] M. S. Lasater, C. T. Kelley, A. G. Salinger, D. L. Woolard, and P. Zhao. “Parallel Parameter Study of the Wigner-Poisson Equations for RTDs”. Computers and Mathematics with Applications, v. 51 (2006), pp. 1677–1688.

[23] G. J. Recine. *Numerical Simulation of Quantum Electron Transport in Nanoscale Resonant Tunneling Structures*. PhD thesis, Stevens Institute of Technology, Hoboken, NJ, 2004.

[24] A. S. Costolanski and C. T. Kelley. “Efficient Solution of the Wigner-Poisson Equations for Modeling Resonant Tunneling Diodes”. IEEE Trans. on Nano., v. 9 (2010), pp. 708–715.

[25] E. Wigner. “On the quantum correction for thermodynamic equilibrium”. Phys. Rev., v. 40 (1932), pp. 749–759.

[26] W. R. Frensley. “Wigner function model of a resonant-tunneling semiconductor device”. Phys. Rev. B, v.36, no. 3 (1987), pp. 1570–1580.

[27] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. “An overview of the Trilinos project”. ACM Trans. Math. Softw., v. 31, no. 3 (2005), pp. 397–423.

[28] R. Tsu and L. Esaki. “Tunneling in a finite superlattice”. Applied Physics Letters, v. 22 (1973), pp. 562–564.

[29] T.C.L.G. Sollner, W.D. Goodhue, P.E. Tannenwald, C.D. Parker, and D.D. Peck. “Resonant tunneling through quantum wells at frequencies up to 2.5THz”. Applied Physics Letters, v. 43 (1983), pp. 588–590.

[30] T.C.L.G. Sollner, E.R. Brown, W.D. Goodhue, and H.Q. Le. “Observation of millimeter-wave oscillations from resonant tunneling diodes and some theoretical considerations of ultimate frequency limits”. Applied Physics Letters, v. 50 (1987), pp. 332–334.

[31] ALGLIB (http://www.alglib.net), Sergey Bochkanov and Vladimir Bystritsky.

[32] M. Sala, K. Stanley, and M. Heroux. “Amesos: A Set of General Interfaces to Sparse Direct Solver Libraries”. Proceedings of PARA’06 Conference, Umea, Sweden, 2006.

[33] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. D. Wilkes, E. A. Burroughs, R. B. Lehoucq, and L. A. Romero. *LOCA 1.0 Library of Continuation Algorithms: Theory and Implementation Manual*. Sandia National Laboratory, SAND2002-0396, 2002.

[34] 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”. Int. J. Bifurcation Chaos, v. 15, no. 3 (2005), pp.1015-1032.

[35] H.B. Keller. *Lectures on Numerical Methods in Bifurcation Problems*. Applied Mathematics 217 (1987), 50.

[36] V.J. Goldman, D.C. Tsui, and J.E. Cunningham. “Observation of Intrinsic Bistability in Resonant-Tunneling Structures”. Physical Review Letters, v. 58, no. 12 (1987), pp. 1256–1259.

[37] T.C.L.G. Sollner. Comment on “Observation of Intrinsic Bistability in Resonant-Tunneling Structures”. Physical Review Letters, v. 59, no. 14 (1987), p. 1622.