NEMO1D: Software Highlights

Gerhard Klimeck
Who are the Users?

Users: Engineers
Interest: Advanced Dev. Design
Shrinking Devices
Problems: Transport with Particle Interaction
Approach: Drift Diffusion, Boltzmann Eq.

Users: Scientists
Interest: Solid State Physics
New Device Materials
Problems: Coherent Quantum Mechanics
Approach: Schrödinger Eq.

NEMO
Approach: Non-equilibrium Green Functions
Includes: Quantum Mechanics, Particle Interaction, Transport
Models: Charging, Bandstructure, Scattering
Access: User friendly GUI

TI, Raytheon, Hughes, Motorola, IBM, Intel
Universities
National Labs.

Students
Universities
NCN
User Requirements

• Simple Data Input:
  – Device Design
    • No restrictions on geometry.
    • III-V and Si.
  – Material Database
    • Provide material parameters.
    • User can modify material parameters.
    • User can define new materials.
  – Simulation Parameters
    • Guide user through hierarchy during the selection.

• Simple Data Analysis:
  – Built-in graphics plots.
  – Allow access to all computed data.
  – Minimize disk clutter.

Multiple platforms: SUN, HP, IBM, SGI
NEMO: Current vs. Voltage

Calculation Display

Calculation Options
- Current vs Voltage
- Resonances
- Band Structure

Test Conditions
- Start (V): 0
- Stop (V): 0.5
- Num. Points: 21
- T (K): 300
NEMO: Resonances

![Graph showing energy vs position with highlighted resonances and Fermi level values.]

NCN
Interactive Data Visualization

- During an I-V run all 2D data files are stored in one “slicer”
- All of the data can be examined after the run.
- All slicers can be linked to examine the correlation of the contained data.
- Hot Buttons Jump to Min/Max Extrema on I-V

Experimentalists Demanded Fast / Interactive Simulation Must Be Faster than Experiment
Software Structure: Hybrid C, FORTRAN, and FORTRAN 90

- Advantages of C:
  - flexible data design, fast prototyping.
  - simple hook-up of the graphical user interface.
  - close to the UNIX system level for job scheduling.

- Advantages of FORTRAN
  - speed for floating point arithmetic.
  - 4x speed-up for the same algorithm comparing C and F77 on HP.

- Advantages FORTRAN90 = FORTRAN + ...
  - standardized vector and matrix arithmetic.
  - data structures.
  - simple parallelization.
Programmer Requirements

• Theory:
  – Fast prototypeing.
  – Change input and output often.
  – Need to visualize detailed (deep down in the code) data.

• GUI:
  – 20/80 rule: 20% of the simulator is used 80% of the time.
    -> Need to put the 20% within easy reach of the user.
  – Keep up with ever changing simulation parameters.
  – Static GUI design is “simple”:
    • Canned software products like X-Designer create a GUI.
    • Windows are defined at compile time.
    • Put in call-backs to the functions which perform a certain action once a button is pushed.
  – Dynamic GUI design is “tough”:
    • Create windows “on-the-fly” depending on the user choices.
Simulation Parameter Entry

• What are simulation parameters?
  – Current and potential models
  – Energy and momentum grid specifications
  – Iteration numbers, required accuracy, etc.

• Design Premise:
  – Make **ALL** parameters available to the user

• Problems:
  – **User**: Large number of parameters
    • not all are needed for a specific simulation
    • they may even be mutually exclusive and conflicting!
    • some of them may only be for expert use
  => Hierarchical Ordering
  – **Programmer**: Need to be able to add parameters very quickly, painlessly, without reworking the GUI.
  => Dynamic widget creation, reflecting abstract data structures.
Hierarchical Ordering of User Input

<table>
<thead>
<tr>
<th>Semi-class. self-cons. pot. &amp; single band current</th>
<th>Quantum self-cons. potential &amp; single band current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify desired outputs</td>
<td>exchange &amp; correlation?</td>
</tr>
<tr>
<td>Quantum region: “Where are wave-functions?”</td>
<td>how to go from bias to bias?</td>
</tr>
<tr>
<td>Non-equilibrium region: “Where are the reservoirs?”</td>
<td>Specify desired outputs</td>
</tr>
<tr>
<td>Adaptive energy grid</td>
<td>Quantum region: “Where are wave-functions?”</td>
</tr>
<tr>
<td></td>
<td>Non-equilibrium region: “Where are the reservoirs?”</td>
</tr>
<tr>
<td></td>
<td>Quantum Charge region: “Where is the charge quantum mechanically calculated?”</td>
</tr>
<tr>
<td></td>
<td>Resonances-based energy grid</td>
</tr>
</tbody>
</table>

Ask user for input that is really needed.

> User input determines the sequence of simulation parameter windows.

NCN
Generic Data Structure I/O

Dynamic GUI Design.
- data structure
- member descriptor
-> I/O for GUI or files

Data Structure
PotType potential
real hbarovertau
Boolean Ec
RangeStruct NonEq

Translator

Graphical User Interface
File/Batch User Interface
potential=Hartree
hbarovertau=0.0066
Ec=FALSE

Such Modular Approach is Germaine to Simulation
It Enables Development with Uncertain Specs.
Software Structure: Hybrid C, FORTRAN, and FORTRAN 90

• Advantages of C:
  • flexible data design, fast prototyping.
  • simple hook-up of the graphical user interface.
  • close to the UNIX system level for job scheduling.

• Advantages of FORTRAN
  • speed for floating point arithmetic.
  • 4x speed-up for the same algorithm comparing C and F77 on HP.

• Advantages FORTRAN90 = FORTRAN + ...
  • standardized vector and matrix arithmetic.
  • data structures.
  • simple parallelization.

• NEMO 3-D design:
  • XML handles all I/O including computational data structures

• Scriptable Interfaces:
  • Key element to integration into other programs
Eigenenergies contain a lot of information!!

- Poles correspond to resonances
- Zeros correspond to anti-resonances
  - Real part resonance energy
  - Imaginary part resonance width

- Resonance width $\sim 0.1$ meV
- Energy Range 0.4eV
- Each computation of $G^R$ at each energy is expensive
- Need to use an inhomogeneous energy grid.
\[ \left( E - H - \sum_{\text{bound}(E)} R \right) \Psi = 0 \]

**Shifted and Inverted Eigenspectrum**

- Hamiltonian is non-Hermitian
- Boundary self-energy is energy dependent
  - Strong dependence at band edge
  - Weak dependence between band edges
- Utilize Lanczos on shifted and inverted, energy independent system.
- Refine resonances using Newton iterations on the energy dependent system.
• Utilize eigenvalue solver to locate resonance energies and resonance widths
  » Need to resolve the real part and imaginary part of the associated Green fct.
  » Can solve integral analytically for real and imaginary part
• Need to resolve band edges
  » Integrate over 1-D density of states
• Need to resolve Fermi energies
  » Utilize Fermi function directly
Novel Energy Grid Generator

- Problem: need to integrate over typically very sharp spectral features
- Assume we know where the resonances are -> toy problem

Reduction of mesh points here: 4x
=> Speedup: 4x-16x
=> Memory Savings: 4x, 16x, 64x
High Level Simulator Tasks:
1) Calculate potential profile:
   charge < -self-cons.- > potential
2) Calculate current:
   may or may not be self-cons. w/ potential.

Low Level Simulator Tasks:
1) Generate microscopic quantity
2) Integrate (1,2,3 dimensions) into observables (charge, current).

\[ \int dE dk \int dE dk \]
\[ n(k, E; z) \quad J(k, E; z) \]
\[ n(z) \quad I \]
\[ \text{Potential} \quad \text{Current} \]
How do we Integrate $G^<$? Different Integration Algorithms.

**Single Band**

- No Scattering
- Analytic transverse $k$
  
  $\int dE \int dk G^<(E,k)$

- Finite Order Scattering
- Numerical transverse $k$
  
  $\int dE \int dE_z$

**Multiband**

- No Scattering
- Analytic transverse $k$
  
  $\int dE G^<(E,k=0)$

- No Scattering
- Numerical transverse $k$
  
  $\int dk \int dE$

- Infinite Order Scattering

- Finite Order Scattering
  
  $\int dE \int dk$
- Code modularity provides **simulator flexibility**.
- User specifies independently
  - Model for $G^<(E,k)$
    - $1,2,10$ etc. band model without scattering
  - Integration target
    - Electron density or current density
  - Integration Scheme
    - $k=0$ analytic transverse integration
    - Full band numerical transverse momentum integration
  - Energy grid type:
    - Fixed with resonance finding
    - Adaptive
    - Mixed
- Addition of models is as simple as adding a function name to a list.
• Pseudocode is a step by step description of a program in a natural language (high level and low level)
• Maintain pseudocode and code in one document -> easy to keep it up-to-date
• Generated automatically from comments in the code
• Cross references to
  » theory description
  » further pseudocode
  » the actual code

Such Documentation Approach is Germaine to Simulation
A Similar Approach is in Wide use in Some Other Tools Now
Revision Control Systems

- Several people - one piece of code?
- Break up the source code into (many) files.
- Use a central library system.
- All developers can look at the latest versions.
- The files cannot be changed.
- Must check out and lock a file before editing.

FORGET about PRIVATE CODE VERSIONS!!!

User gekco: locks and edits file.c r1.3 he creates the new r1.4

Users lake, blanks, etc.: can look at the latest public revision of file.c (r1.3)

Emacs facilitates the check-out & check-in