Solving Device PDE's with the PROPHET Simulator

http://www-tcad.stanford.edu/~prophet/

Dan Yergeau   Stanford University

With contributions from
Zhiping Yu and Gaofeng Wang (Stanford)
Paco Leon (Mixed Technology Associates)
Conor Rafferty (Agere Systems)

Outline

- What is PROPHET?
- Specifying PDE's for PROPHET to solve
- Discretization on a mesh
- Applications
- Summary

What is PROPHET?

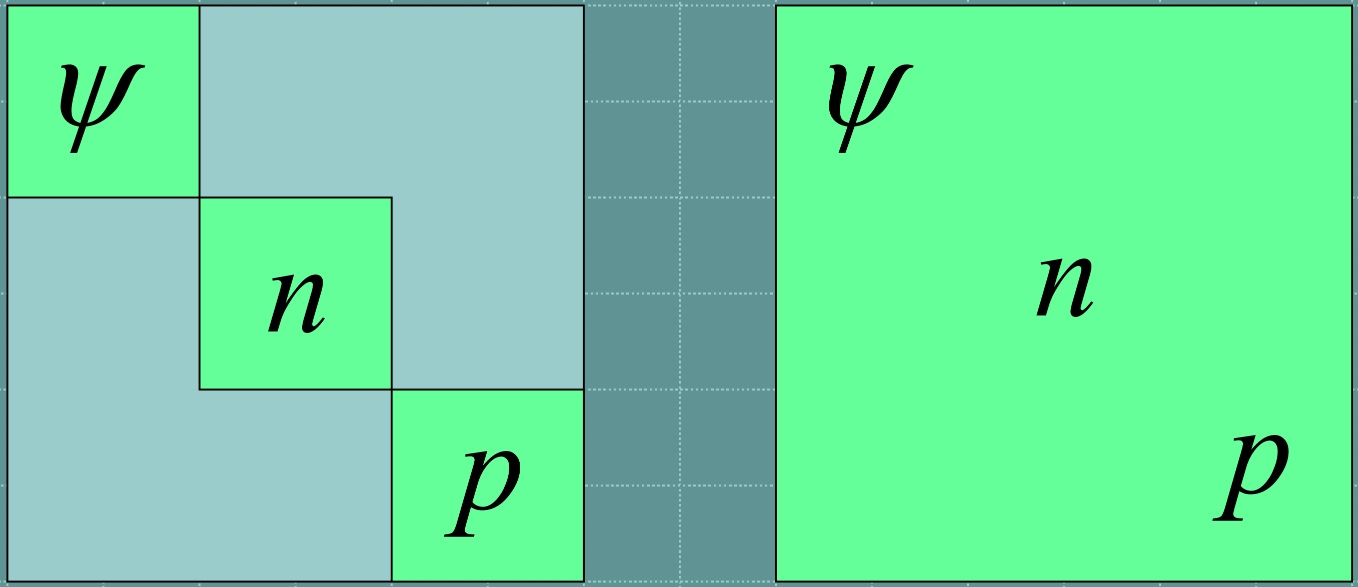
- Developed at Bell Labs as a process simulator
- Released externally as a TCAD simulation *platform* (Stanford, UT Austin, and Conor Rafferty added device simulation capabilities)
- Permits user-level specification of PDE's created from reusable operators

PROPHET Overview

User interface:	Input parser	Graphics/postprocessing			
Modules:	Solve	Grid	Field	Bias	…
Libraries:	Database	Structure	Linear solver		
PDE Engine:	Assembly	Solvers	Discretization (geometric)	Models (physical)	

Representing Physical Systems in PROPHET

Global system is composed of blocks of PDE's



Loosely Coupled

Tightly Coupled

Representing PDE's in PROPHET

- PDE is a sum of terms (well, almost)
- Terms are a combination of geometric and physical operators: $\mathrm{PDE} = \mathrm{G}_1\mathrm{P}_1 + \mathrm{G}_2\mathrm{P}_2 + \mathrm{G}_3\mathrm{P}_3$
- Geometric operator: $\nabla\times,\ \ \nabla\cdot,\ \partial/\partial\mathrm{t}$
- Physical operator: $\mathrm{flux} = f\left(A, \nabla A, X, \nabla X, \ldots\right)$
- Functions permit evaluation of intermediate values with chaining back to solution variables

Geometric Operators

- Spatial operators: divergence, nodal
- Differentiation wrt time
- Interface flux
- Dirichlet
- Constraint
- Interface algebraic

Physical Operators

- Algebraic building blocks: +, -, *, /, sqrt, exp, etc.
- Fluxes: $-\varepsilon\nabla\psi,\ n\mu\nabla\psi - D\nabla n$
- Many domain-specific expressions:
  - space charge density
  - mobility
  - device contact boundary conditions
  - Shockley-Reed-Hall and Auger recombination

Specifying Terms

<geo_op>.<phy_op>(<inputs>|<outputs>)@{<where>}

$\nabla \cdot (\varepsilon \psi)$	box_div.lapflux(psi|psi)@{silicon,poly,oxide}
$q\left(p - n + N_D^+ - N_A^-\right)$	nodal.nscd(electrons,holes,netdope|psi)@{silicon,poly}
	constraint.continuity(0|psi)@{silicon/oxide,poly/oxide}
	dirichlet.device_dirichlet(netdope|psi)@{CONTACTS}

Specifying Systems (PDE block)

```
system name=silicon_poisson
+ sysvars=psi
+ term0=ndiv_fbm.lapflux(psi|psi)@{SEMICONDUCTORS,INSULATORS}
+ term1=nodal.nscd(electrons,holes,netdope|psi)@{SEMICONDUCTORS}
+ term2=dirichlet.device_dirichlet(netdope|psi)@{CONTACTS}
+ term3=constraint.continuity(psi|psi)@{ALL_INTERFACES}
+ tmpvars=electrons,holes
+ func0=quasiFermi(psi|electrons,holes)@{SEMICONDUCTORS}
```

Solution Methods

- Timestepping
  - typically TR/BDF2 with LTE-based timestep control
- PDE block staggering
- Newton nonlinear algorithm on a single block
  - convergence detection in residual and/or update norm
  - range clamping
  - damping
- Small signal AC, preliminary harmonic balance

Discretization

Apply Gauss' Theorem

$$\nabla \cdot \mathbf{F} = U = \frac{\partial u}{\partial t} + G - R$$

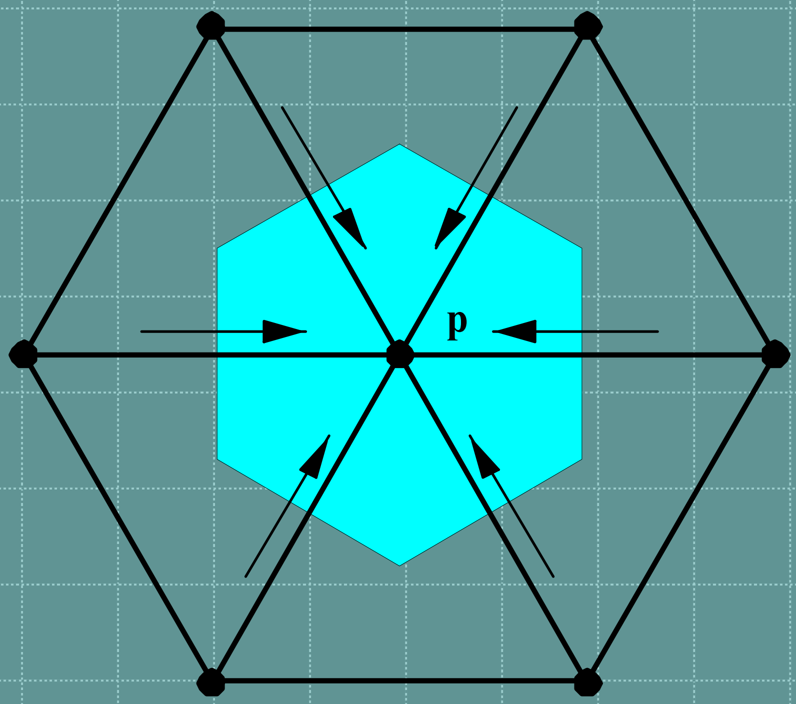
$$\oint_{\Omega_{CV}} \nabla \cdot \mathbf{F} d\Omega - \oint_{\Omega_{CV}} U d\Omega = 0$$

$$\oint_{S_{CV}} \mathbf{F} \cdot \hat{n} dS - \oint_{\Omega_{CV}} U d\Omega = 0$$

Approximate on a mesh

$$\oint_{S_p} \mathbf{F} \cdot \hat{n} dS = \sum_i F_{i,p} l_i$$

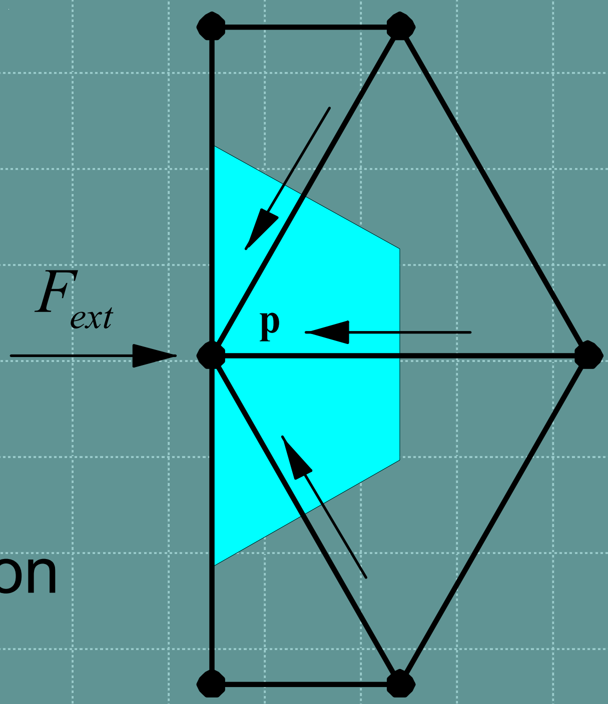
$$\oint_{\Omega_p} U d\Omega = U_p V_p$$



Interfaces

Control volume integration is "closed" by including external flux

$$\oint_{S_p} \mathbf{F} \cdot \hat{n} dS = F_{ext} l_p + \sum_i F_{i,p} l_i$$



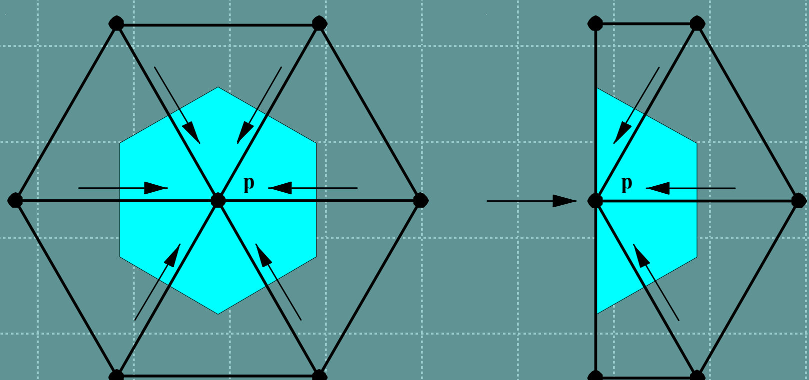
Thus, the natural boundary condition is zero-flux if equation has no interface flux terms

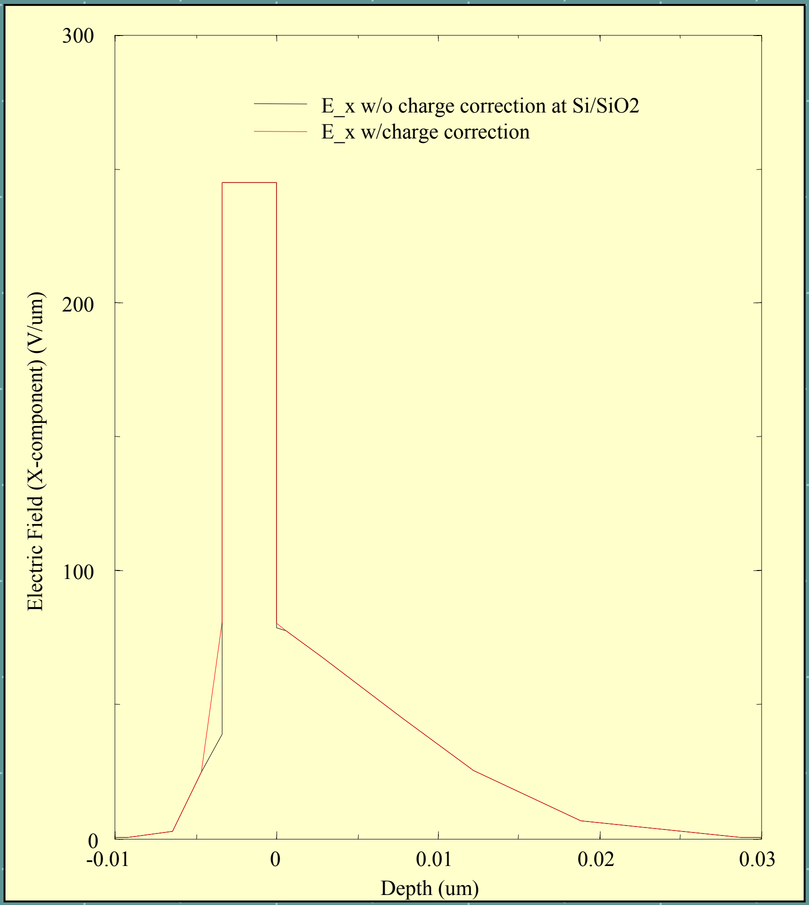
Evaluating the "Real" Flux

$$\nabla \cdot \mathbf{j}_n = -q\nabla \cdot (n\mu\nabla\psi - D\nabla n)$$

$$\mu = \mu(N, T_L, E_{\parallel}, E_{\perp})$$

$$E_{\parallel} = \frac{|\mathbf{E}\cdot\mathbf{j}|}{|\mathbf{j}|},\ E_{\perp} = \frac{|\mathbf{E}\times\mathbf{j}|}{|\mathbf{j}|}$$





Applications

- Modeling quantum effects via Density Gradient
- Laser simulation
- Interconnect interactions with substrate (device level frequency domain)

Density Gradient

Models quantum confinement of particles adjacent to silicon/oxide interface. The large chemical potential barrier of the insulator requires the wave function to vanish at the interface. Continuity of this wave function pushes carriers away from the barrier.

$$\mathbf{F}_n = -D_n \nabla n + \mu_n n \nabla \psi + 2\mu_n n \frac{\hbar^2}{4lqm_n^*} \nabla \left(\frac{\nabla^2 \sqrt{n}}{\sqrt{n}} \right)$$

Quantum potential: $2b_n \dfrac{\nabla^2 \sqrt{n}}{\sqrt{n}}$

Density Gradient System

$$\nabla \cdot (\varepsilon \psi) + q\left(p - n + N_D^+ - N_A^-\right) = 0$$

$$\nabla \cdot \left(b_n \nabla \sqrt{n}\right) + \frac{\sqrt{n}}{2}\left(\psi - \frac{kT}{q}\ln\frac{n}{n_i} - \phi_n\right) = 0$$

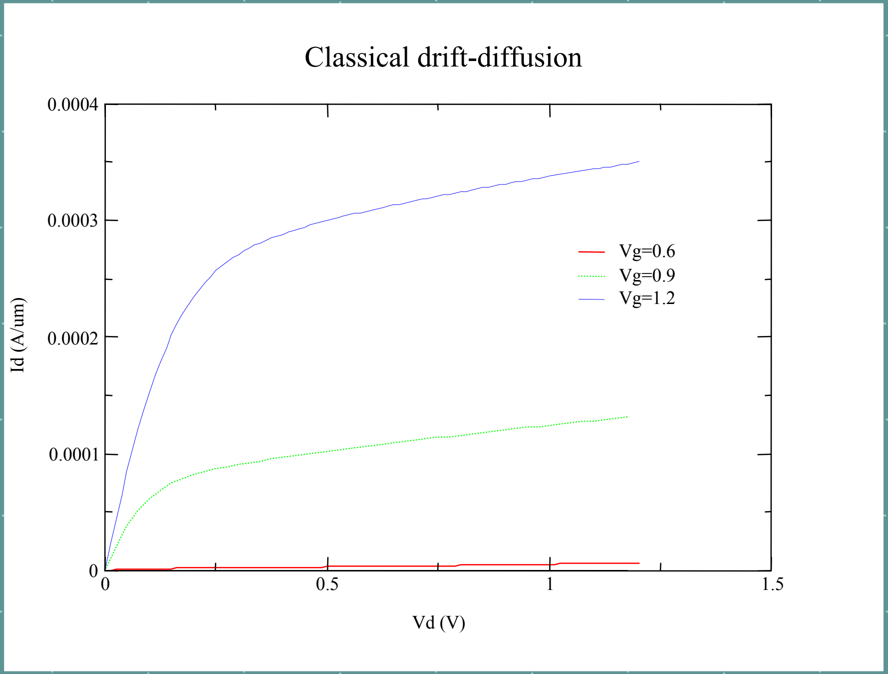
$$\nabla \cdot \left(b_p \nabla \sqrt{p}\right) + \frac{\sqrt{p}}{2}\left(\psi + \frac{kT}{q}\ln\frac{p}{p_i} - \phi_p\right) = 0$$

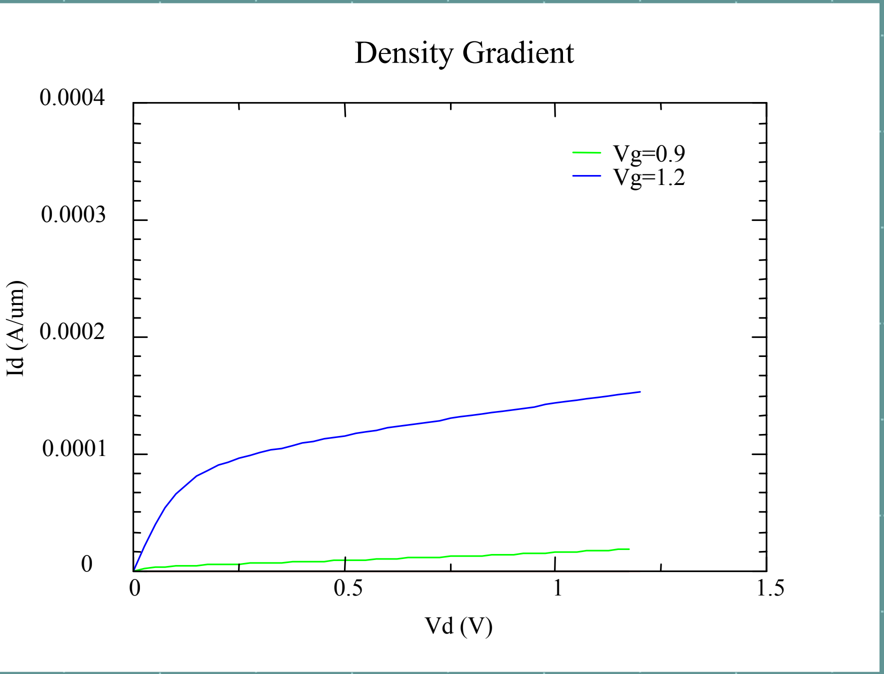
$$\frac{\partial n}{\partial t} + \nabla \cdot \left(\mu_n n \nabla \phi_n\right) + r = 0$$

$$\frac{\partial p}{\partial t} - \nabla \cdot \left(\mu_p p \nabla \phi_p\right) + r = 0$$

Classical DD vs. Density Gradient

MIT 50nm well-tempered MOSFET

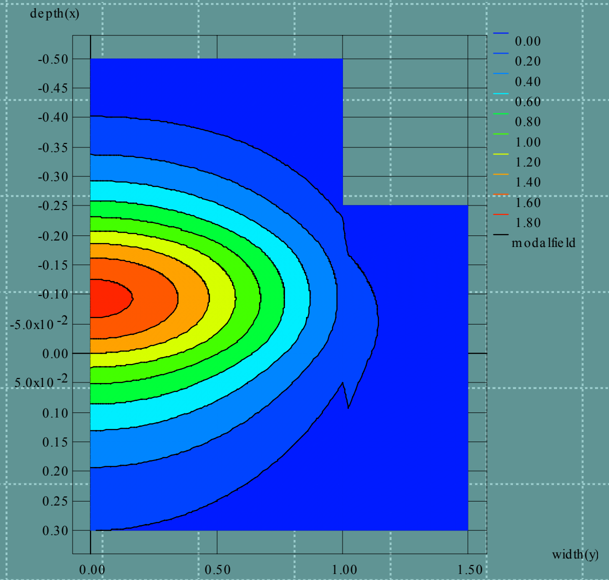




Photon Generation in a Laser

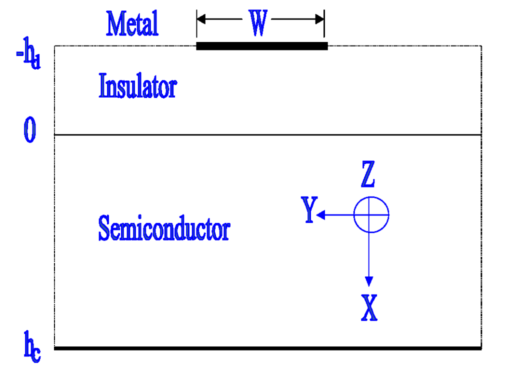
$$\nabla\cdot(\varepsilon\nabla\psi)+q\left(p-n+N_D^+-N_A^-\right)=0$$
$$-\nabla\cdot(-\mathbf{F}_n)+r_{SRH}+r_{Auger}+B_{\text{field}}np+r_{st}=0$$
$$-\nabla\cdot(-\mathbf{F}_p)+r_{SRH}+r_{Auger}+B_{\text{field}}np+r_{st}=0$$
$$-\nabla\cdot(\kappa\nabla T_L)-\nabla\cdot(-\psi\mathbf{j})=0$$
$$\left(v_gG-1/\tau\right)N_p+\beta_{sp}R_{sp}=0$$

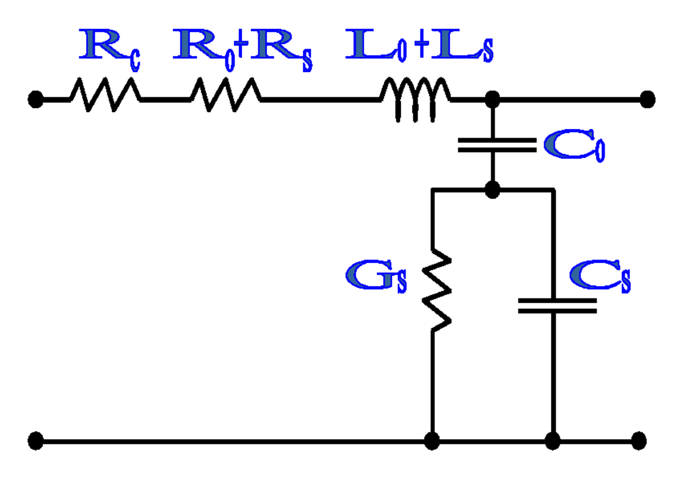
1.55 micron InGaAs/InP
Edge Emitting Laser
Intensity of dominant mode

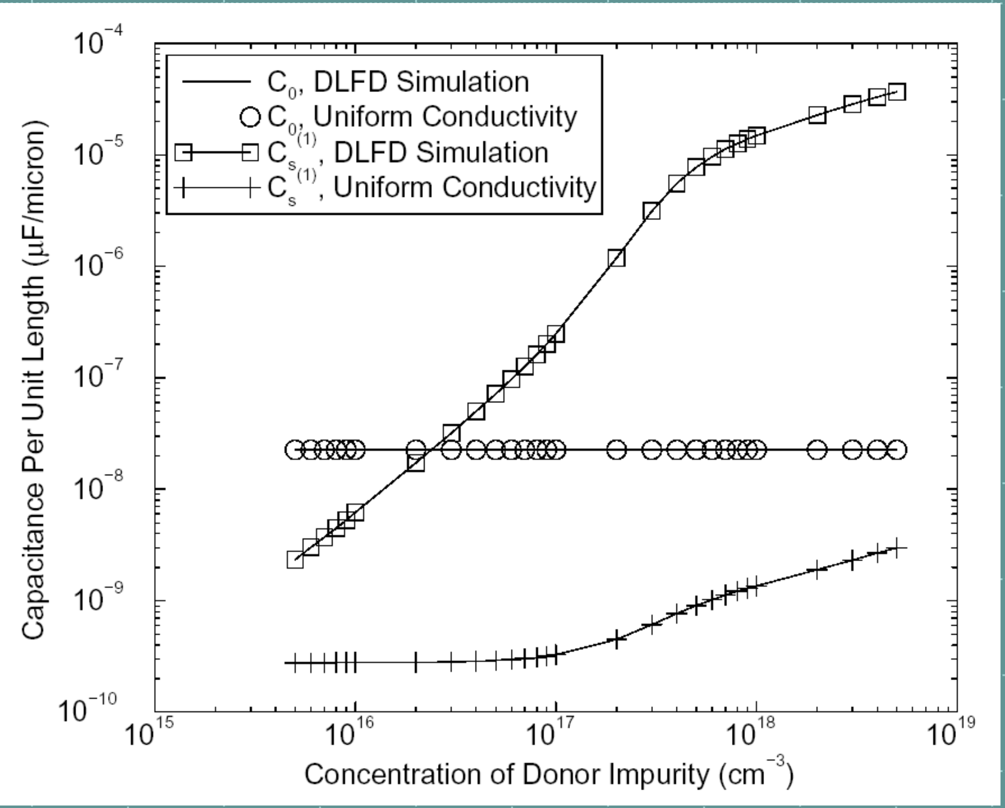


Last equation is lumped (scalar), but dependent on distributed spontaneous emission rate

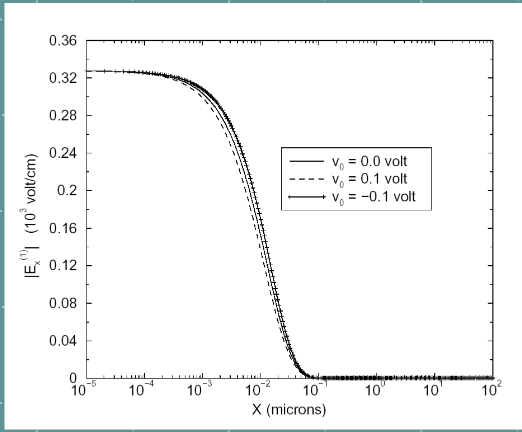
Interconnect/substrate

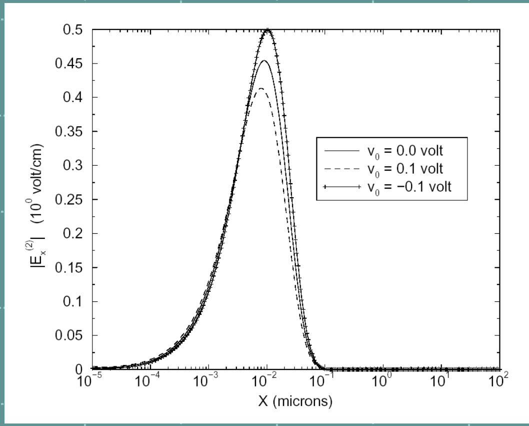


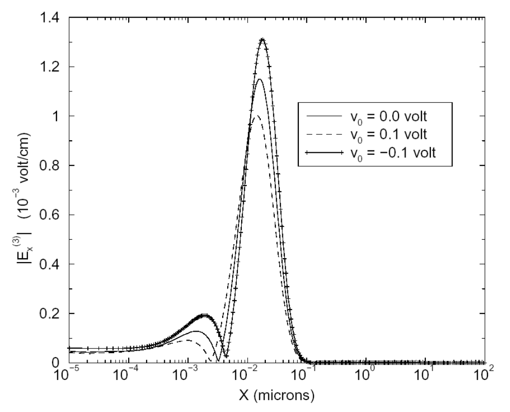


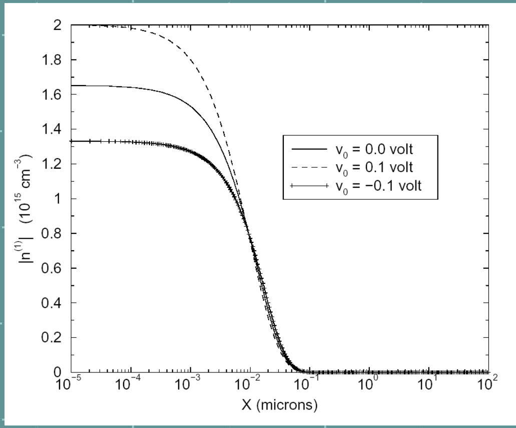


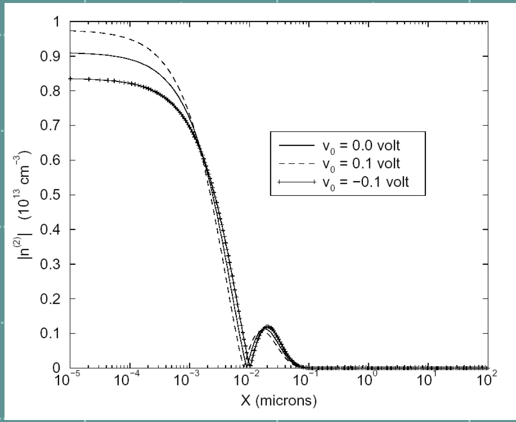
Bias Effects on the Substrate

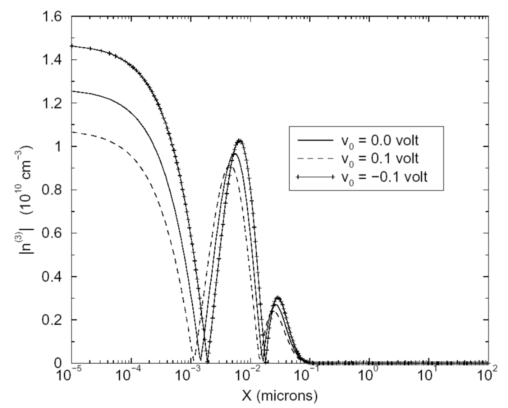












Summary -- Advantages

- Rapid prototyping
- Simulation on real structures
- Reasonable efficiency (extra assembly overhead is fairly minimal)
- Partial box method
- Code reuse means fewer errors
- Model debugging aids

Drawbacks and Needs

- Diagnostics
- Still looking for faster and more robust linear solvers. We use Berkeley Sparse and PETSc (sparse direct, ILU+GMRES, ILU+BiCGstab)
- Recovery from failed Newton is through load control (reduce time step or bias change). What alternatives have been tried and how well do they work?