Experiments in DIY TCAD

Andrei Taropa

project supervised by Dr. Dan Weatherill

About me

- Andrei Taropa
- 3rd year Physics undergraduate
- University of Oxford, Lincoln College
- Interested in Computer Science and Physics
- andrei-nicolae.taropa@lincoln.ox.ac.uk

Agenda

1. Motivation
2. Development
3. Theory
4. Implementation
5. Further work
6. Integration with other software

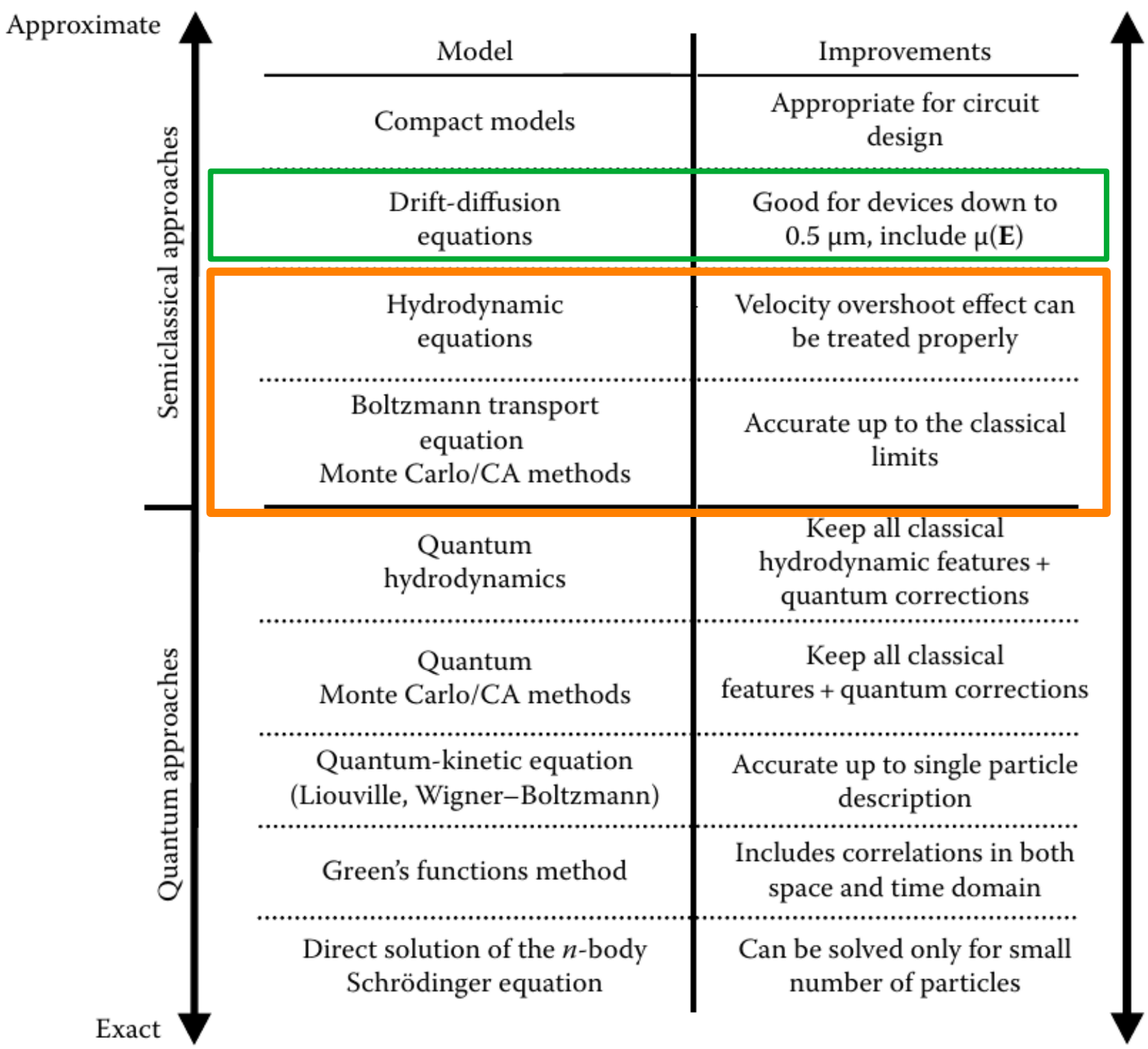
Computational Project

- It is worth 17% of the 3rd year grade
- There are around 10 computational projects allocated each year

- This project was supervised by Dr. Dan Weatherill and Prof. Ian Shipsey from the OPMD group
- **The aim:** implement a simulation to calculate the shape of a stored cloud of electrons confined in a CCD potential well

Motivation

- Develop a fully open source TCAD-like simulator
- Drift-diffusion equations model
- Use the 3D Finite Volume Method
- Calculating the E-field structure from doping & geometry



SPICE is here

We are here

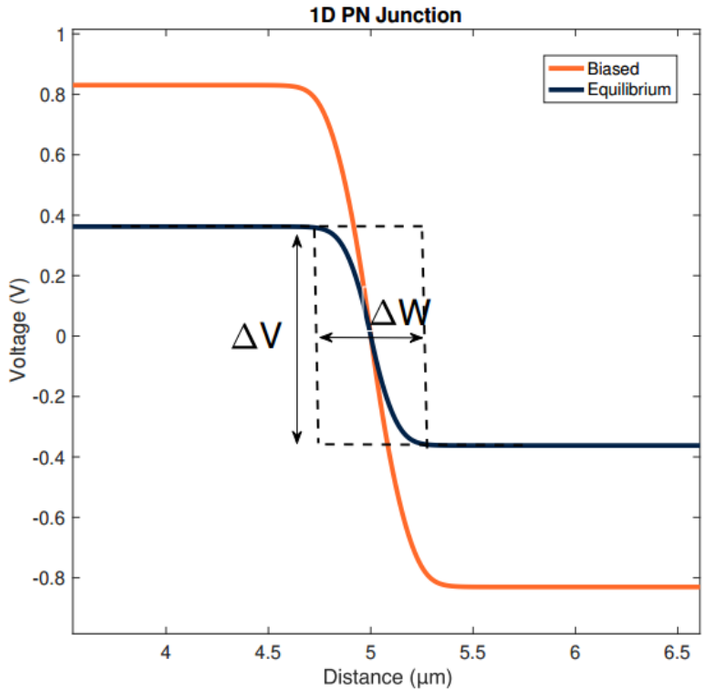
Silvaco/Sentaurus are here

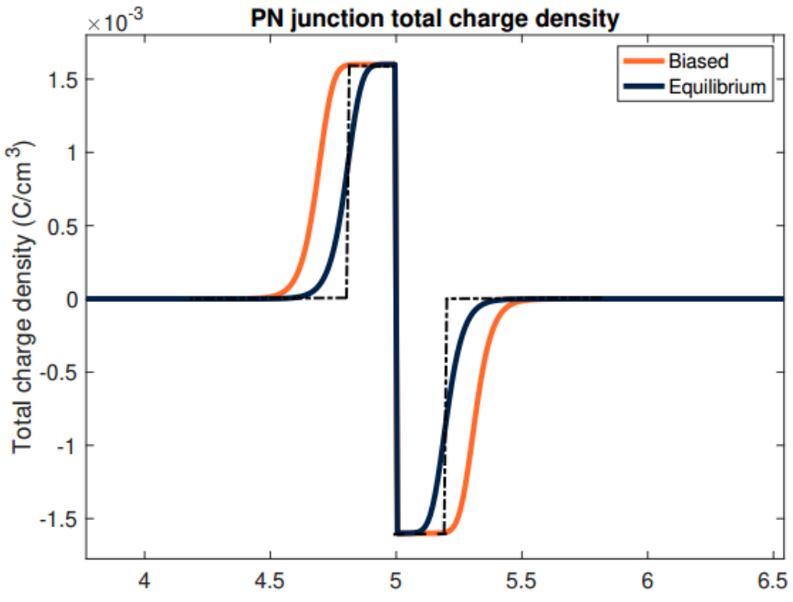
From "Computational Electronics" by Vasileska, Goodnick, Klimeck Chapter 5

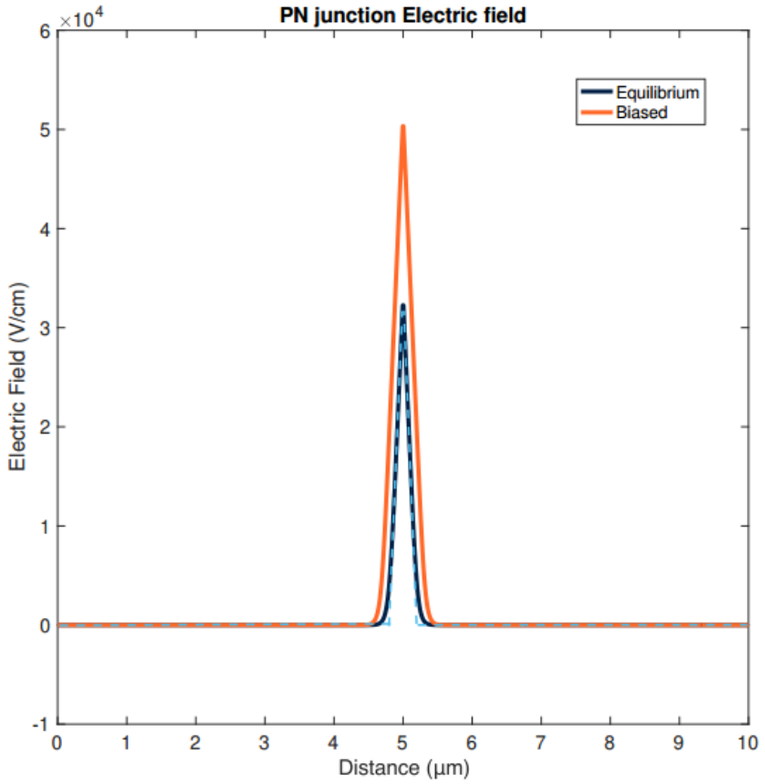
Development

- (2019/20) – Guo-Zheng “Theo” Yoong & Taavet Kalda first implementations of pn junction
- (2020/21) – Mac Zhou & Xihan Deng – performance improvements, addition of mobility & recombination models
- (2021/22) – Megan Evans –first “CCD charge packet” simulation
- (2022/23) – Tevz Lotric – comparisons against commercial and analytic models from 1D simulator
- (2023/24) – Andrei Taropa – rewrite in c++ using the DUNE numerics framework. Our first inherently 2D devices simulated!

Results on pn junction simulation



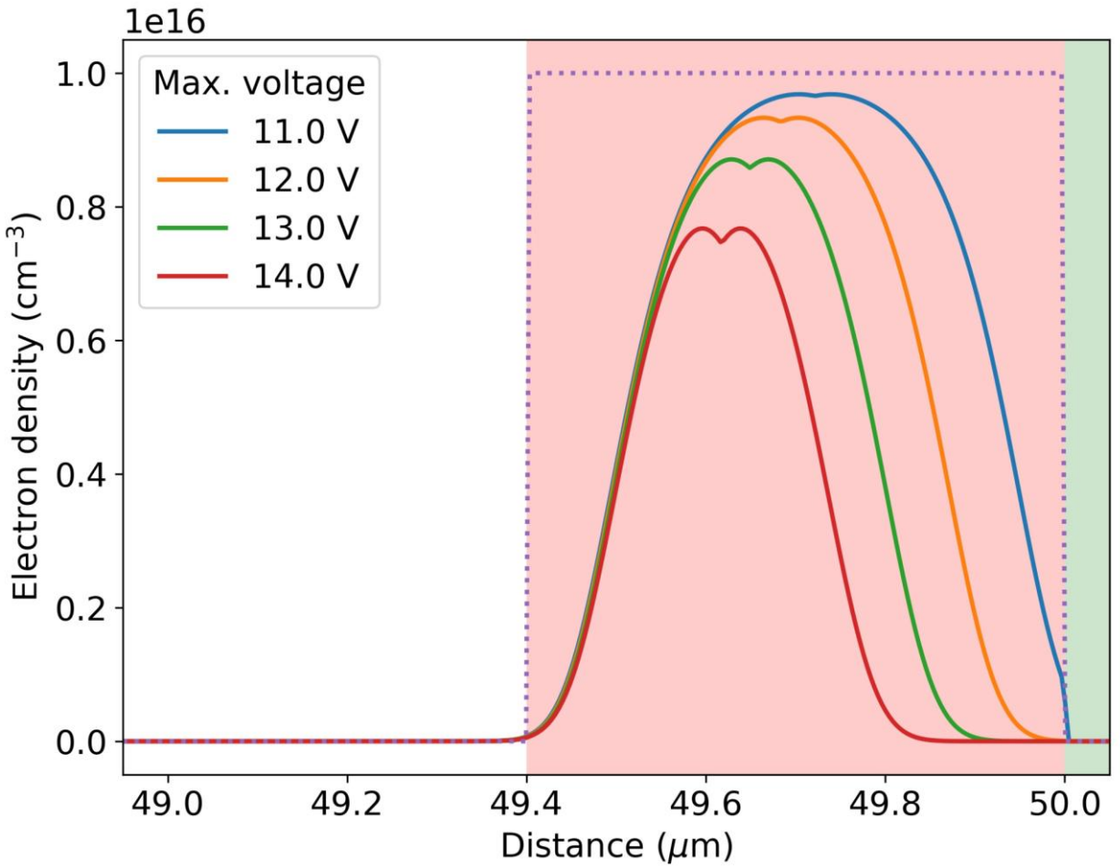




Images from Mac Zhou (2020)

Charge Packet Shapes

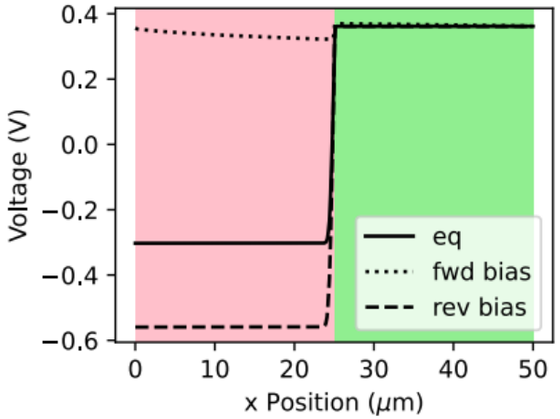
Top: A (1D) charge packet shape simulation in a CCD buried channel with an insulator on top

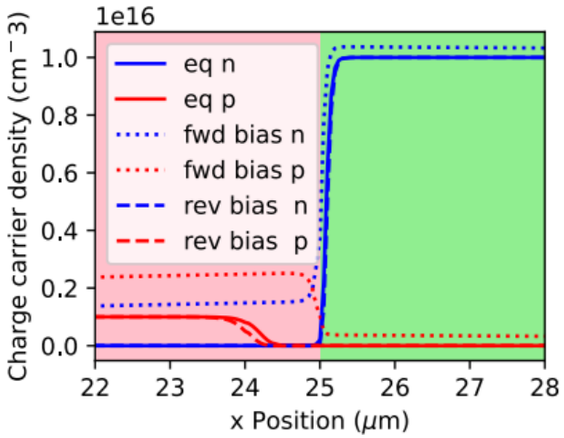


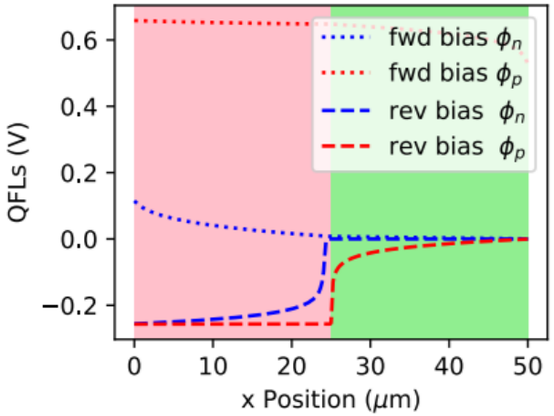
Images from Tevz Lotric (2022/23)

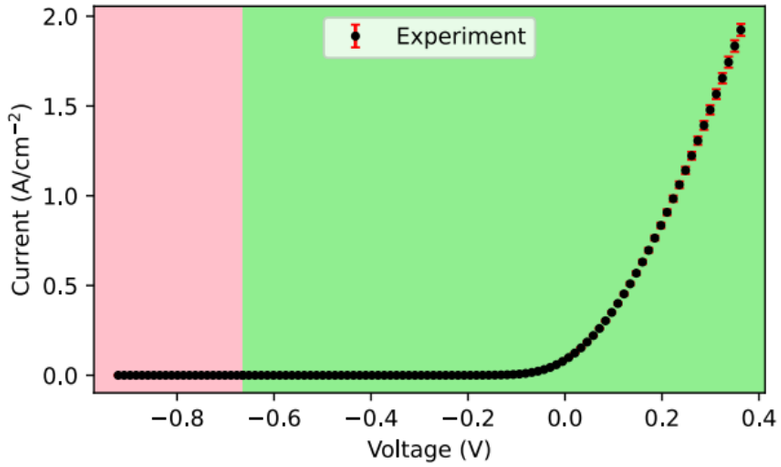
C++ simulations of diodes

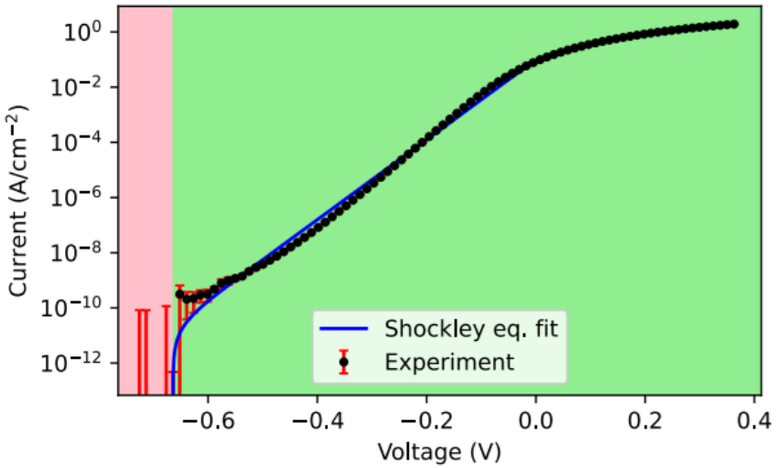
At this point, we can simulate diode structures routinely!
But this is the first time we had it working in the new C++ code!



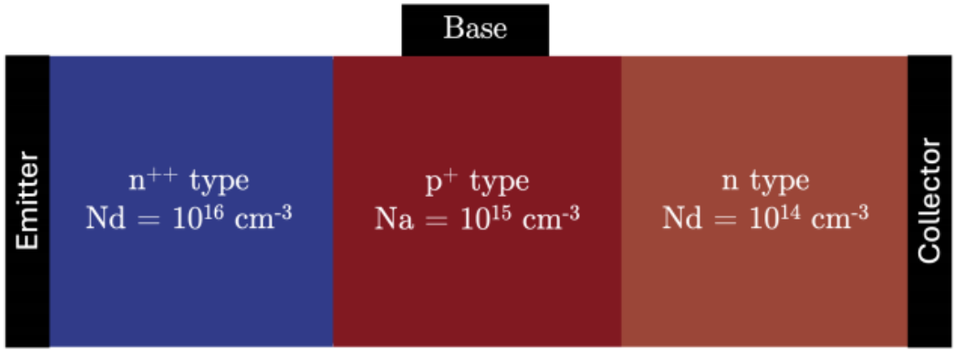




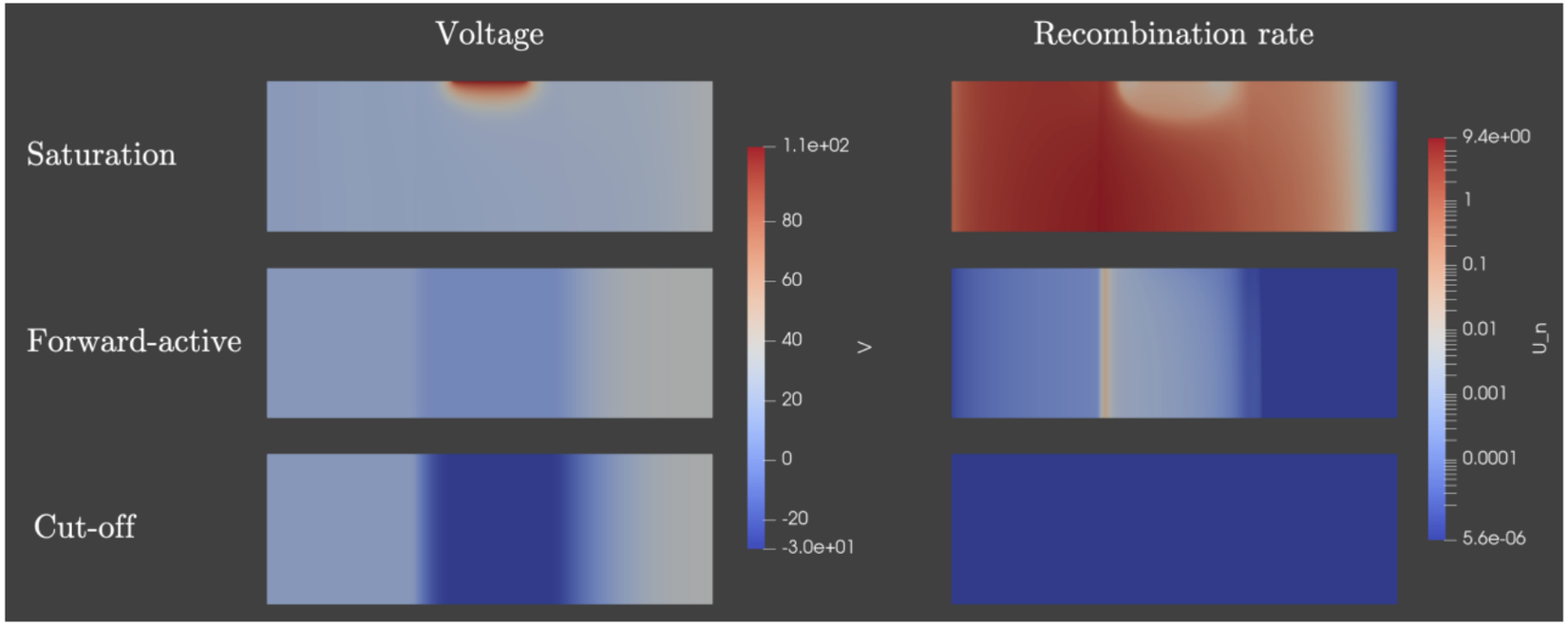




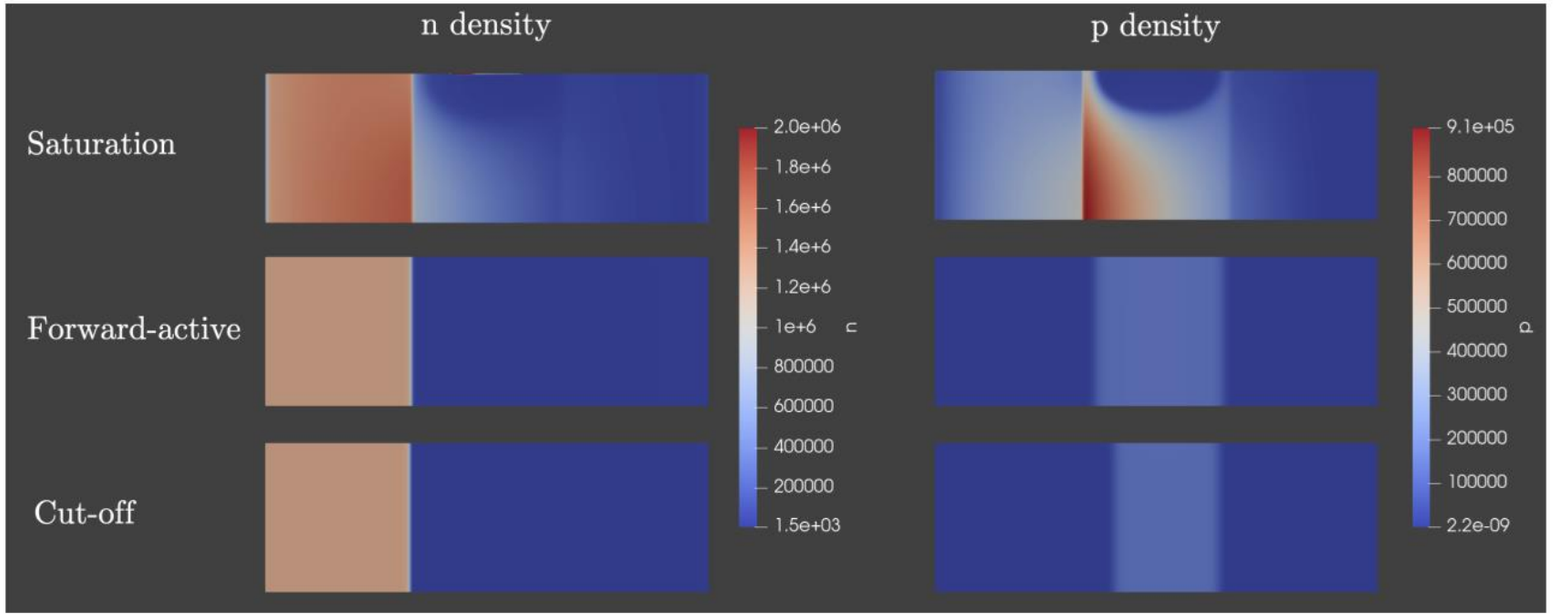
2D Device Simulation

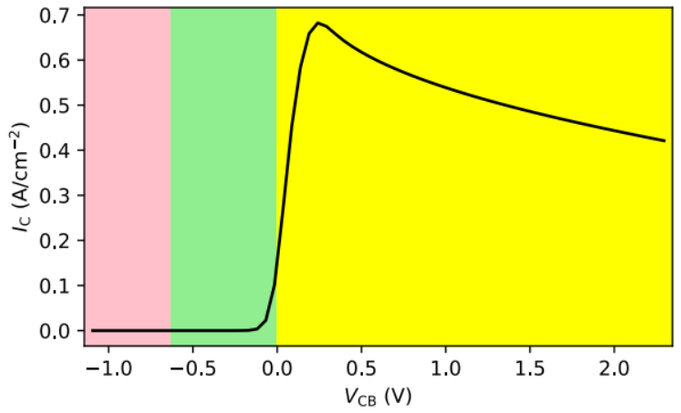


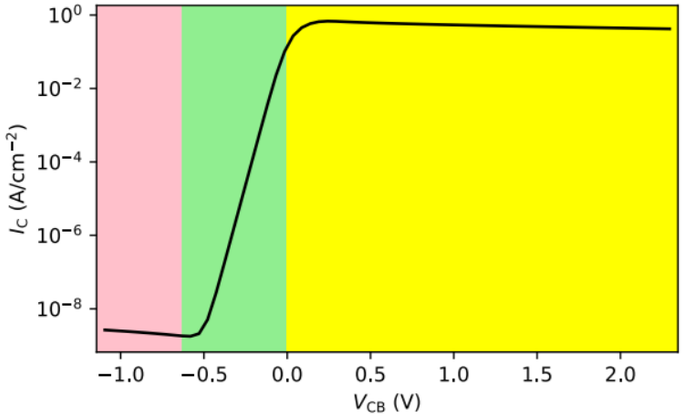
An NPN BJT cannot be properly simulated in 1D because the base contact needs to be in the 2nd dimension.



2D Device simulation #2







Extracted transistor characteristics (note we can't yet set constant currents at boundaries, so these don't look "quite" textbook

Theory #1 - Poisson's equation

- We start with Poisson's equation, where we assume that the medium has a uniform permittivity ε_r:

$$\nabla^2 V = -\frac{q}{\varepsilon_r \varepsilon_0}(p - n + N_D^+ - N_A^-)$$

p = hole density
n = electron density
N_D = donor doping
N_A = acceptor doping
V = Voltage
q = electron charge

Theory #2 – System of equations

- Poisson's equation:

$$\nabla^2 V = -\frac{q}{\varepsilon_r \varepsilon_0}(p - n + N_D^+ - N_A^-)$$

- Charge continuity equations:

$$\frac{\partial n}{\partial t} = 0 = \frac{1}{q}\nabla \cdot \mathbf{J_n} + U_n \qquad \frac{\partial p}{\partial t} = 0 = -\frac{1}{q}\nabla \cdot \mathbf{J_p} + U_p$$

- Current given by the drift diffusion equation:

$$\mathbf{J}_n = qn\mu_n \mathbf{E} + qD_n \nabla n \qquad \mathbf{J}_p = qp\mu_p \mathbf{E} - qD_p \nabla p$$

p = hole density
n = electron density
N_D = donor doping
N_A = acceptor doping
V = Voltage
q = electron charge

J = current density
U = recombination rate

μ = mobility
D = diffusivity
E = electric field

Theory #3 – Boltzmann approximation

- We can write the electron and hole number densities in terms of Quasi-Fermi levels ϕ_n and ϕ_p:

$$n = n_i \exp\left(\frac{q(V-\phi_n)}{k_B T}\right) \qquad p = n_i \exp\left(\frac{q(\phi_p - V)}{k_B T}\right)$$

- N.B. these are the Boltzmann approximated densities. You can use Fermi-Dirac instead, which changes the exponential to a more complicated integral, but conceptually similar

Theory #4 – Scale factors

- It is also useful to scale voltage quantities:

$$\bar{V} = \frac{qV}{k_B T} \qquad \bar{\phi_n} = \frac{q\phi_n}{k_B T} \qquad \bar{\phi_p} = \frac{q\phi_p}{k_B T}$$

- We also scale the lengths:

$$L_D = \sqrt{\frac{\varepsilon_r \varepsilon_0 k_B T}{q^2 N_{\max}}}$$

Theory #5 – Solving Poisson's equation

- Solving Poisson's equation:

$$\bar{\nabla}^2 \bar{V} = \frac{n_i}{N_{\max}} \left(\exp\left(\bar{V} - \bar{\phi}_n\right) - \exp\left(\bar{\phi}_p - \bar{V}\right) - \frac{C}{n_i} \right)$$

- We use "Gummel's method":

$$\bar{V}_{\text{new}} = \delta\bar{V} + V_{\text{old}}$$

Taylor expand:

$$\exp \pm\delta\bar{V} \approx 1 \pm \delta\bar{V}$$

- After substituting into Poisson's equation:

$$\frac{d^2\bar{V}_{\text{new}}}{d\tilde{x}^2} = \frac{n_i}{N_{\max}} \underbrace{\left(\exp\left(\bar{V}_{\text{old}} - \bar{\phi}_n\right) - \exp\left(\bar{\phi}_p - \bar{V}_{\text{old}}\right) - \frac{C}{n_i} \right)}_{A} + \frac{n_i}{N_{\max}} \delta\bar{V} \underbrace{\left(\exp\left(\bar{V}_{\text{old}} - \bar{\phi}_n\right) + \exp\left(\bar{\phi}_p - \bar{V}_{\text{old}}\right) \right)}_{B}$$

Theory #6 - Solving Poisson's equation

- … substituting into Poisson's equation:

$$\frac{d^2\bar{V}_{\text{new}}}{d\tilde{x}^2} = \frac{n_i}{N_{\max}} \underbrace{\left(\exp\left(\bar{V}_{\text{old}} - \bar{\phi}_n\right) - \exp\left(\bar{\phi}_p - \bar{V}_{\text{old}}\right) - \frac{C}{n_i} \right)}_{A} + \frac{n_i}{N_{\max}} \delta\bar{V} \underbrace{\left(\exp\left(\bar{V}_{\text{old}} - \bar{\phi}_n\right) + \exp\left(\bar{\phi}_p - \bar{V}_{\text{old}}\right) \right)}_{B}$$

- And after some rearrangement:

$$\left(\bar{\nabla}^2 - B\right) \bar{V}_{\text{new}} = A - \frac{C}{N_{\max}} - \bar{V}_{\text{old}} B$$

- This solves the voltage

Theory #7 – Einstein approximation

- For current we use the Einstein approximation:

$$\mu k_B T = Dq$$

- … to obtain a simplified form for current density:

$$\mathbf{J}_n = -qn\mu_n \nabla \phi_n$$

- However, we are interested in the current at the cell's boundary and since we already use a linear approximation for V, using another linear approximation to interpolate n at the cell boundaries is going to lead to a very rapid numerical instability. This is because n depends on V exponentially. The optimal solution for the interpolation problem is given by the Scharfetter-Gummel Discretization

Theory #8 - Scharfetter-Gummel Discretization

- Scharfetter-Gummel Discretization for electrons:

$$\mathbf{J}_n^{\text{mid}} = \frac{D^{\text{mid}} q}{dL_D} \left(\text{Ber}(\bar{V}^{\text{out}} - \bar{V}^{\text{in}}) n^{\text{out}} - \text{Ber}(\bar{V}^{\text{in}} - \bar{V}^{\text{out}}) n^{\text{in}} \right)$$

- Where we use the Bernoulli function:

$$\text{Ber}(x) = \frac{x}{e^x - 1}$$

Theory #9 - Finite volume method

- Using the divergence theorem:

$$\int_{\mathcal{V}} f(\mathbf{r})\, d\mathcal{V} = \int_{\mathcal{V}} \nabla \cdot \mathbf{\Phi}\, d\mathcal{V} = \int_{S} \mathbf{\Phi} \cdot \mathbf{dS} \quad \Rightarrow \quad \sum_{\text{faces}} \mathbf{\Phi} \cdot \mathbf{S} = \mathcal{V} f(\mathbf{r})$$

- We discretize the equation for voltage:

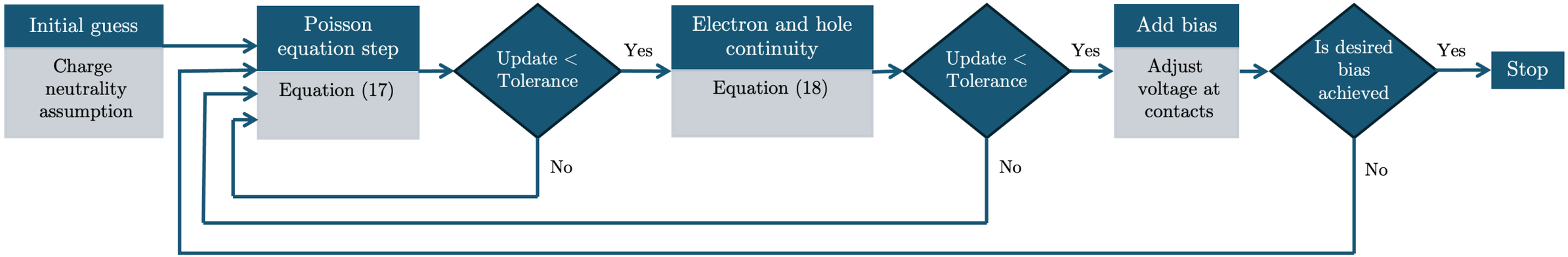
$$\sum_{\text{faces}} (V_{\text{new}}^{\text{out}} - \bar{V}_{\text{new}}^{\text{in}}) \frac{S}{d} - \mathcal{V} B \bar{V}_{\text{new}}^{\text{in}} = \mathcal{V} \left(A - \frac{C}{N_{\max}} - \bar{V}_{\text{old}} B \right)$$

- and for current continuity:

$$\sum_{\text{faces}} \mathbf{J}_n^{\text{mid}} \cdot \mathbf{S} = -U_n q \mathcal{V} \qquad \sum_{\text{faces}} \mathbf{J}_p^{\text{mid}} \cdot \mathbf{S} = U_p q \mathcal{V}$$

Implementation - Gummel Iteration scheme

- For the initial guess we assume charge neutrality, which gives an analytical solution for the voltage.



- Solution is in the form (V, n, p)
- We need to adjust biasing of the simulated device in small steps, otherwise the solution will no longer converge

Implementation – discretization 1D

- 1D – easy: $$\left(\bar{\nabla}^2 - B\right)\bar{V}_{\mathrm{new}} = A - \frac{C}{N_{\max}} - \bar{V}_{\mathrm{old}}B$$

- We calculate the second order derivative as follows:

$$f''(\mathbf{x}_i) = \frac{1}{h^2}\begin{pmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 1 & -2 & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & -2 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & -2 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix} \begin{pmatrix} \uparrow \\ x_{j-1} \\ x_j \\ x_{j+1} \\ \downarrow \end{pmatrix}$$

- The matrix only has non-zero elements next to the diagonal
- Note: we use sparse matrices

Implementation – DUNE Numerics

- DUNE, the Distributed and Unified Numerics Environment is a modular toolbox for solving partial differential equations (PDEs) with grid-based methods. It supports the easy implementation of methods like Finite Elements (FE), Finite Volumes (FV), and also Finite Differences (FD). (https://www.dune-project.org)

Distributed and Unified Numerics Environment

Implementation – discretization 3D

- Dune provides an easy way to find neighboring cells:

```
1  for(const auto& elem: elements(view))
2  {
3      for(const auto& isect: intersections(view,elem))
4      {
5          if(isect.boundary())
6          {
7              // code goes here
8          }
9          if(isect.neighbor())
10         {
11             // code goes here
12         }
13     }
14 }
```

- The index of any element can be easily retrieved using the mapper concept:

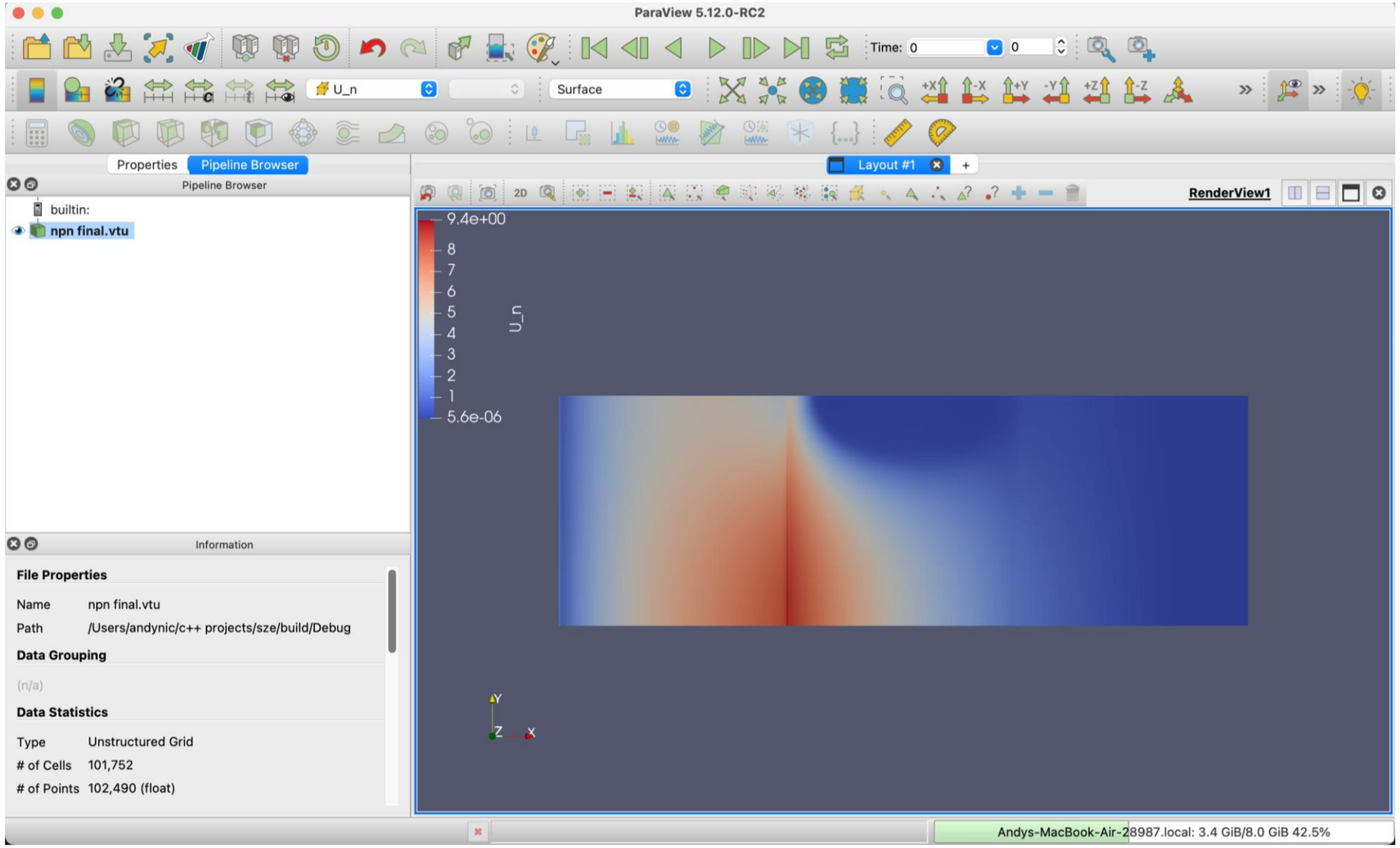
```
mapper.index(elem)
```

Implementation – benefits of using DUNE

- Implementation is the same for any number of dimensions
- Implementation is independent of grid geometry
  - this allows us to use cells of variable sizes in the future
- Data output in the format of the visualization toolkit (vtk)
- BiCGSTABSolver from the Iterative Solvers Template Library
  - Can run in parralel

ParaView

- Open source



Implementation - summary

- Simulate using the drift-diffusion model
  - good tradeoff between performance and accuracy
- Using C++
  - a lot faster since it is a compiled language
  - DUNE library
- Abstract implementation
  - we use templated classes that work with different data types
  - the implementation is independent from the grid’s geometry and the number of dimensions
  - the mobility and recombination physics models used by the simulation can be easily changed

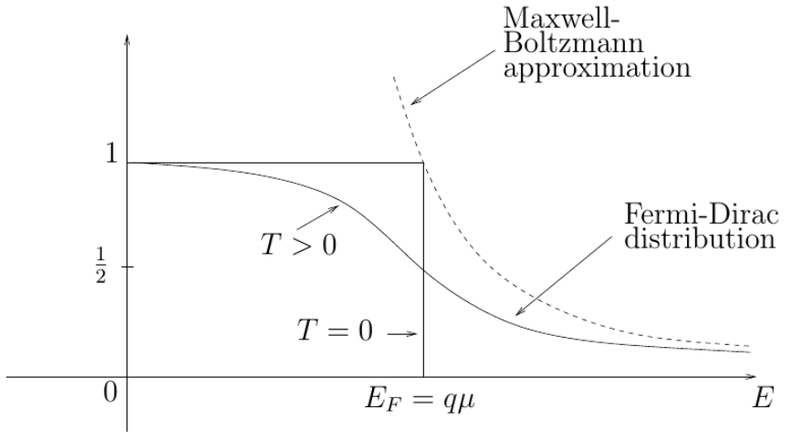
Further work

- Port mobility & recombination models over to new code
- Implement insulating boundary conditions properly in 2D
- Parallelize (this should be easy due to the DUNE architecture)
- Compare with commercial simulators
- 3D
- Implement Newton iterations, and a “smart” heuristic for when to switch implementation strategies

Further work

There are 3 bits of “physics” we might want to add into our simulation (before going to more advanced equations entirely)

Mobility models – allowing mobility to vary with field and other quantities adds a lot more realism to drift-diffusion simulations
Recombination / generation – including SRH trapping, impact ionization, zener breakdown etc.
Thermodynamics – swap Boltzmann out for Fermi-Dirac, this is a lot more accurate in some temperature regimes



$$\mu_n(E) = \mu_{n0}\left[1 + \left(\frac{\mu_{n0}E}{V_{\text{sat}}^n}\right)^{\beta_n}\right]^{-\frac{1}{\beta_n}}$$

$$\mu_p(E) = \mu_{p0}\left[1 + \left(\frac{\mu_{p0}E}{V_{\text{sat}}^p}\right)^{\beta_p}\right]^{-\frac{1}{\beta_p}}$$

Xihan Deng & Yichen “Mac” Zhou
Implemented several mobility and recombination models
For our 1D python simulation in 2020/2021

Integration with other software

- Simulator calculates the E-field structure from doping & geometry
- Can be used to provide field structure for Allpix Squared

Thank you!

Dr. Dan Weatherill

Prof. Ian Shipsey

andrei-nicolae.taropa@lincoln.ox.ac.uk