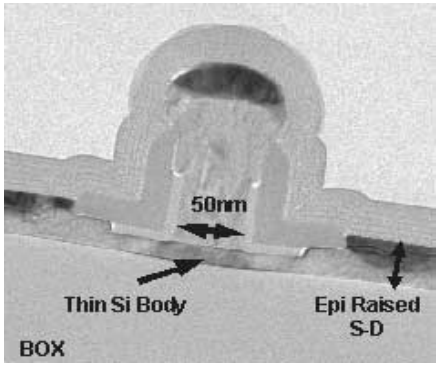
Nano-Scale Device Simulations Using PROPHET
Part I: Basics

Yang Liu

TCAD Group, Stanford University

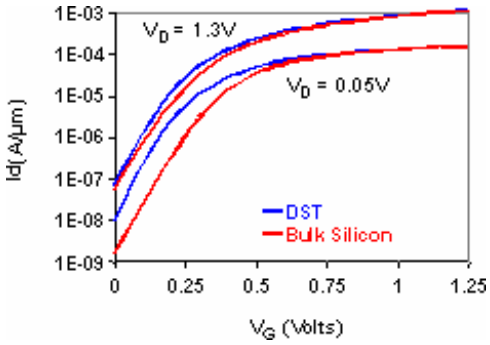
Dec. 2005

Nano-scale devices & modeling

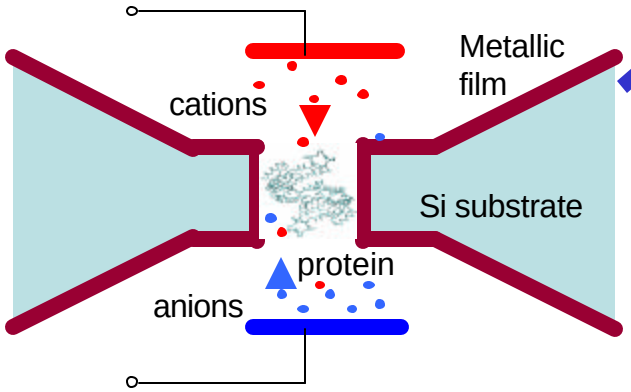


(Intel, 2002)

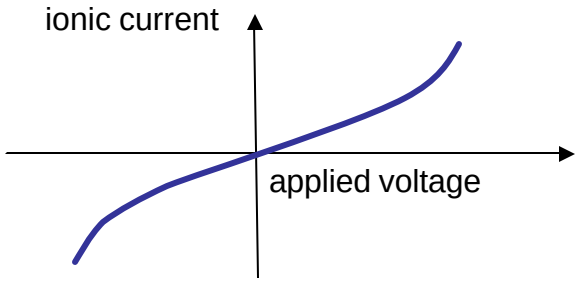
Semiconductor devices



Device modeling & simulation



Nano-bio devices

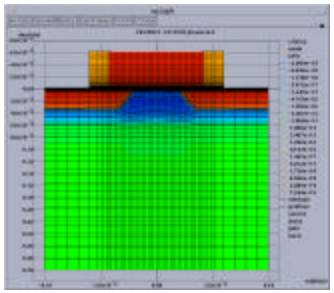


Simulated output to be compared with measurements

Modeling & Simulation Approach

- Establish device equations (e.g. continuum-based partial differential equations)
- Discretize PDEs into difference equations
- Solve difference equations (usually nonlinear)
- Post-process; interpret simulation results

Example: simulation of a transistor



Device structure

$$\frac{\partial n(\vec{r})}{\partial t} = \frac{1}{q}\nabla \bullet \vec{J}_n(\vec{r}) - U_n(\vec{r})$$

$$\frac{\partial p(\vec{r})}{\partial t} = -\frac{1}{q}\nabla \bullet \vec{J}_p(\vec{r}) - U_p(\vec{r})$$

$$\nabla \bullet \varepsilon(\vec{r})\nabla\psi(\vec{r}) + q(p(\vec{r}) - n(\vec{r}) + N_d(\vec{r}) - N_a(\vec{r})) = 0$$

Shockley Eqns.

MODEL EQNS.

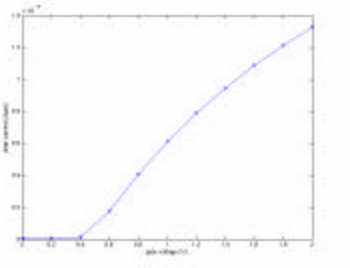
$$\begin{pmatrix} \vec{F}_1(\vec{y},\vec{n},\vec{p}) \\ \vec{F}_2(\vec{y},\vec{n},\vec{p}) \\ \vec{F}_3(\vec{y},\vec{n},\vec{p}) \end{pmatrix} = 0$$

Nonlinear difference eqns.

$$\begin{pmatrix} \frac{\partial \vec{F}_1}{\partial \vec{\psi}} & \frac{\partial \vec{F}_1}{\partial \vec{n}} & \frac{\partial \vec{F}_1}{\partial \vec{p}} \\ \frac{\partial \vec{F}_2}{\partial \vec{\psi}} & \frac{\partial \vec{F}_2}{\partial \vec{n}} & \frac{\partial \vec{F}_2}{\partial \vec{p}} \\ \frac{\partial \vec{F}_3}{\partial \vec{\psi}} & \frac{\partial \vec{F}_3}{\partial \vec{n}} & \frac{\partial \vec{F}_3}{\partial \vec{p}} \end{pmatrix}_{k-1} \cdot \begin{pmatrix} \vec{\psi}_k - \vec{\psi}_{k-1} \\ \vec{n}_k - \vec{n}_{k-1} \\ \vec{p}_k - \vec{p}_{k-1} \end{pmatrix} = -\begin{pmatrix} \vec{F}_1(\vec{\psi}_{k-1},\vec{n}_{k-1},\vec{p}_{k-1}) \\ \vec{F}_2(\vec{\psi}_{k-1},\vec{n}_{k-1},\vec{p}_{k-1}) \\ \vec{F}_3(\vec{\psi}_{k-1},\vec{n}_{k-1},\vec{p}_{k-1}) \end{pmatrix}$$

Newton method

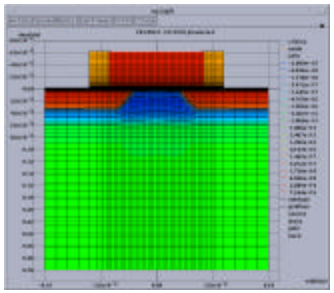
DISCRETIZATION/ NUMERICS



Id-Vg output

OUTPUT

Example: simulation of a transistor



Device structure

$$\frac{\partial n(\vec{r})}{\partial t} = \frac{1}{q}\nabla \bullet \vec{J}_n(\vec{r}) - U_n(\vec{r})$$

$$\frac{\partial p(\vec{r})}{\partial t} = -\frac{1}{q}\nabla \bullet \vec{J}_p(\vec{r}) - U_p(\vec{r})$$

$$\nabla \bullet \varepsilon(\vec{r})\nabla\psi(\vec{r}) + q(p(\vec{r}) - n(\vec{r}) + N_d(\vec{r}) - N_a(\vec{r})) = 0$$

Shockley Eqns.

MODEL EQNS.

$$\begin{pmatrix} \vec{F}_1(\vec{y},\vec{n},\vec{p}) \\ \vec{F}_2(\vec{y},\vec{n},\vec{p}) \\ \vec{F}_3(\vec{y},\vec{n},\vec{p}) \end{pmatrix} = 0$$

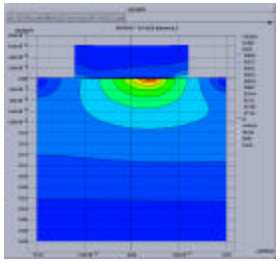
Nonlinear difference eqns.

$$\begin{pmatrix} \frac{\partial \vec{F}_1}{\partial \bar{\psi}} & \frac{\partial \vec{F}_1}{\partial \bar{n}} & \frac{\partial \vec{F}_1}{\partial \bar{p}} \\ \frac{\partial \vec{F}_2}{\partial \bar{\psi}} & \frac{\partial \vec{F}_2}{\partial \bar{n}} & \frac{\partial \vec{F}_2}{\partial \bar{p}} \\ \frac{\partial \vec{F}_3}{\partial \bar{\psi}} & \frac{\partial \vec{F}_3}{\partial \bar{n}} & \frac{\partial \vec{F}_3}{\partial \bar{p}} \end{pmatrix}_{k-1} \cdot \begin{pmatrix} \bar{\psi}_k - \bar{\psi}_{k-1} \\ \bar{n}_k - \bar{n}_{k-1} \\ \bar{p}_k - \bar{p}_{k-1} \end{pmatrix} = -\begin{pmatrix} \vec{F}_1(\bar{\psi}_{k-1}, \bar{n}_{k-1}, \bar{p}_{k-1}) \\ \vec{F}_2(\bar{\psi}_{k-1}, \bar{n}_{k-1}, \bar{p}_{k-1}) \\ \vec{F}_3(\bar{\psi}_{k-1}, \bar{n}_{k-1}, \bar{p}_{k-1}) \end{pmatrix}$$

Newton method

DISCRETIZATION/ NUMERICS

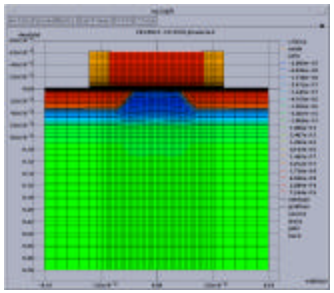
What if model improvement is needed?



temp. distribution

OUTPUT

Example: simulation of a transistor



Device structure

$$\frac{\partial n(\vec{r})}{\partial t} = \frac{1}{q}\nabla \bullet \vec{J}_n(\vec{r}) - U_n(\vec{r})$$

$$\frac{\partial p(\vec{r})}{\partial t} = -\frac{1}{q}\nabla \bullet \vec{J}_p(\vec{r}) - U_p(\vec{r})$$

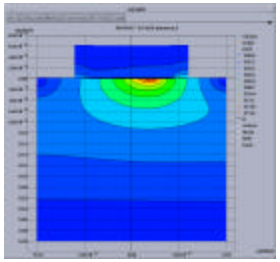
$$\nabla \bullet \varepsilon(\vec{r})\nabla\psi(\vec{r}) + q(p(\vec{r}) - n(\vec{r}) + N_d(\vec{r}) - N_a(\vec{r})) = 0$$

Shockley Eqns.

MODEL EQNS.

PROPHET: DISCRETIZATION/ NUMERICS MADE TRANSPARENT!

What if model improvement is needed?



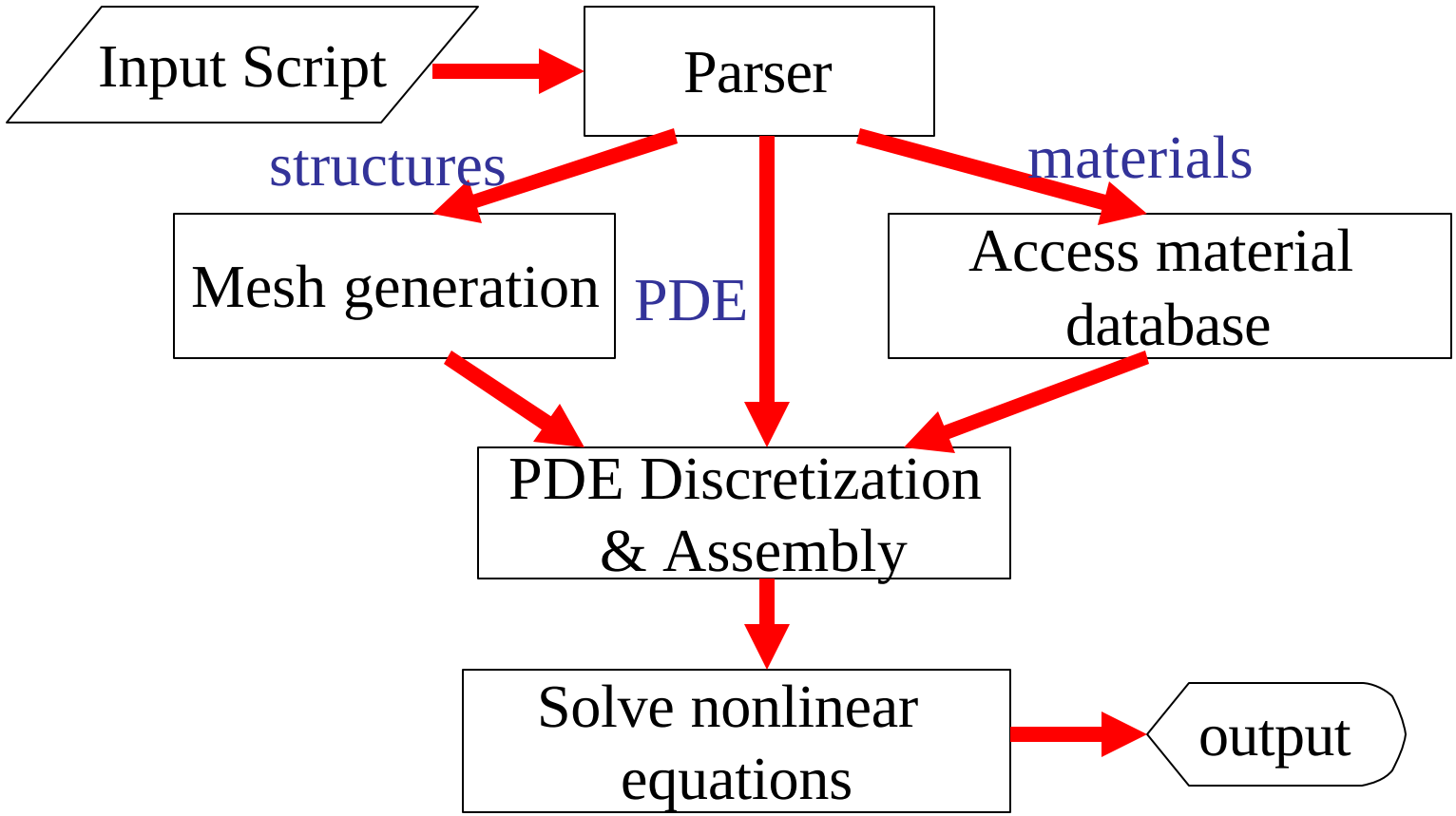
temp. distribution

OUTPUT

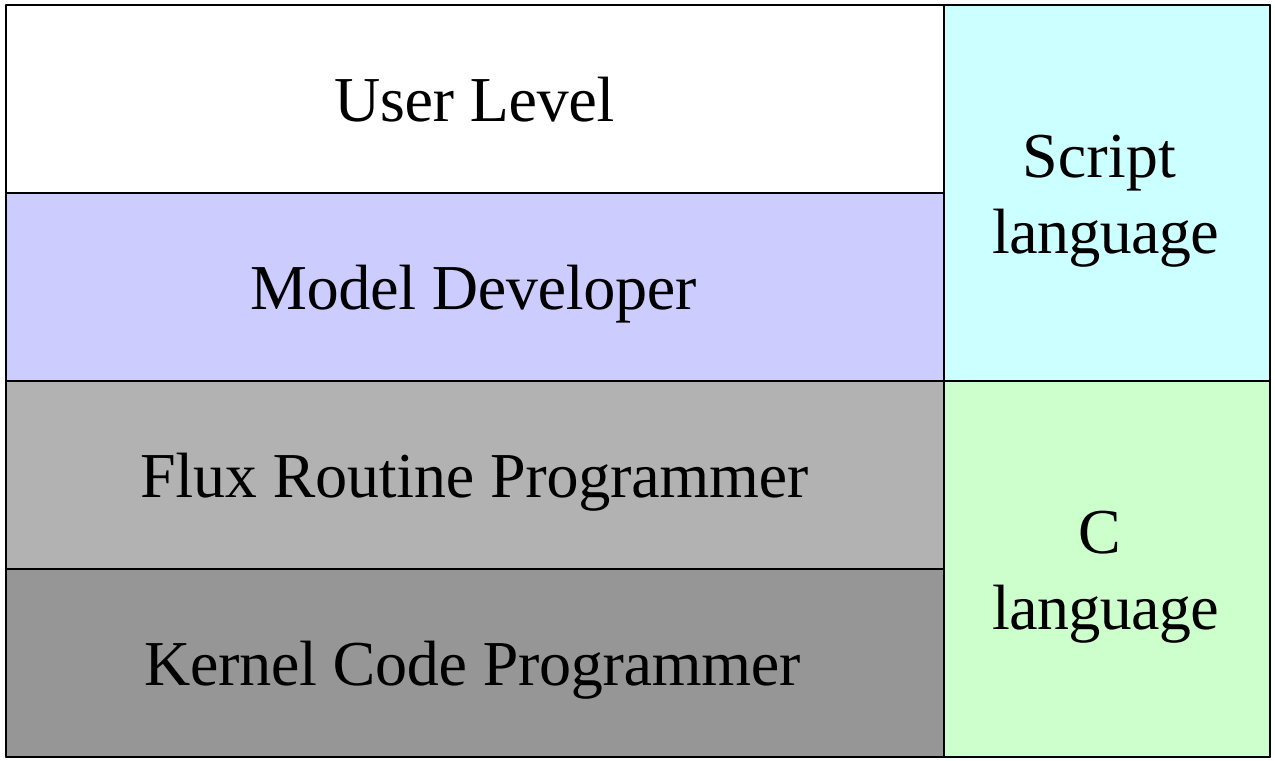
Introduction to PROPHET

- PROPHET: a general PDE solver; suitable for continuum-based simulations of semiconductor and nano-bio devices
- Comparison with other numerical tools
  - Specialized semiconductor simulators
  - Matlab, Mathematica
- Link: http://www.nanohub.org/resource_files/tools/prophet/doc/guide.html

PROPHET simulation flowchart



PROPHET Hierarchical Programming



Introduction to PDEs

- Elliptic
  - Poisson eqn. $e\nabla^2 u = e\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)=C(x,y,z)$
- Parabolic
  - Diffusion eqn. $\frac{\partial u}{\partial t}=D\nabla^2 u=D\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$
- Hyperbolic
  - Drift eqn. $\frac{\partial u}{\partial t}=\vec{v}\cdot\nabla u=v_x\frac{\partial u}{\partial x}+v_y\frac{\partial u}{\partial y}+v_z\frac{\partial u}{\partial z}$
  - Wave eqn. $\frac{\partial^2 u}{\partial t^2}=c^2\nabla^2 u=c^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)$

Example: 1-D Poisson Equation

- Solve potential u(x) for known charge distribution C(x)

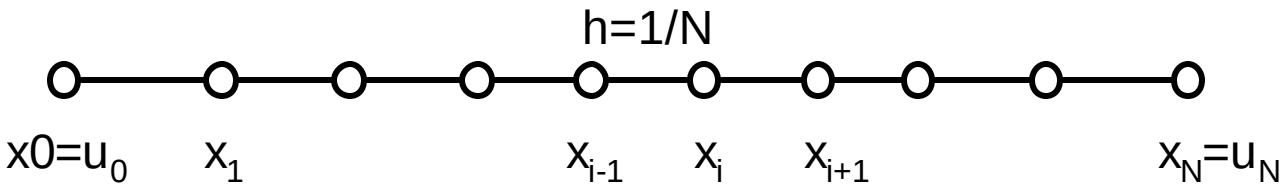
$$\partial^2 u(x) / \partial x^2 = C(x)$$

- Dirichlet boundary condition: $u|_{x=0}=u_0$; $u|_{x=1}=u_N$
- Analytical solution for constant $C(x)=C_0$

$$u(x)=\frac{C_0}{2}x^2+\left(u_N-u_0-\frac{C_0}{2}\right)x+u_0$$

Numerically solve 1-D Poisson Eqn.

- Partition domain x=[0,1] into N sub-domains:



- Discretize Poisson eq. into difference eqs.:

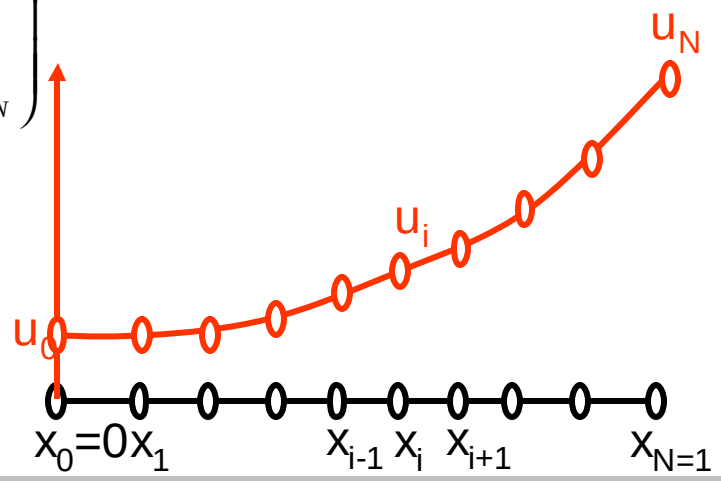
$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x=x_i} = \frac{u_{i-1}+u_{i+1}-2u_i}{h^2} + O(h^2) \approx \frac{u_{i-1}+u_{i+1}-2u_i}{h^2}$$

Numerically solve 1-D Poisson Eq.

- Assemble obtained N-1 difference eqs.

$$\begin{pmatrix} -2 & 1 & & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \bullet & \bullet & \bullet & & \\ & & & \bullet & \bullet & \bullet & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \bullet \\ \bullet \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} h^2 C_1 - u_0 \\ h^2 C_2 \\ h^2 C_3 \\ \bullet \\ \bullet \\ h^2 C_{N-2} \\ h^2 C_{N-1} - u_N \end{pmatrix}$$

- Solve linear system



PROPHET for 1-D Poisson Eq.

Part 1 of PROPHET script “poisson1d.pf”

```
#create dbase entry for sysvar u
dbase createlist name=library/physics/silicon/u
dbase prefix=library/physics/silicon/u
dbase create name=dirichlet.contact0 rval=0.0
dbase create name=dirichlet.contact1 rval=1.0
dbase create name=scale rval=1.0
dbase create name=Dix rval=1
dbase prefix=""

#define system poisson
system name=poisson
+  sysvars=u
+  term0=box_div.lapflux(u|u)@{silicon}
+  term1=nodal.copy(c|u)@{silicon}
+  term2=dirichlet.default_dirichlet(0|u)@{silicon/contact0,silicon/contact1}
```

PROPHET for 1-D Poisson Eq.

Part 2 of PROPHET script “poisson1d.pf”

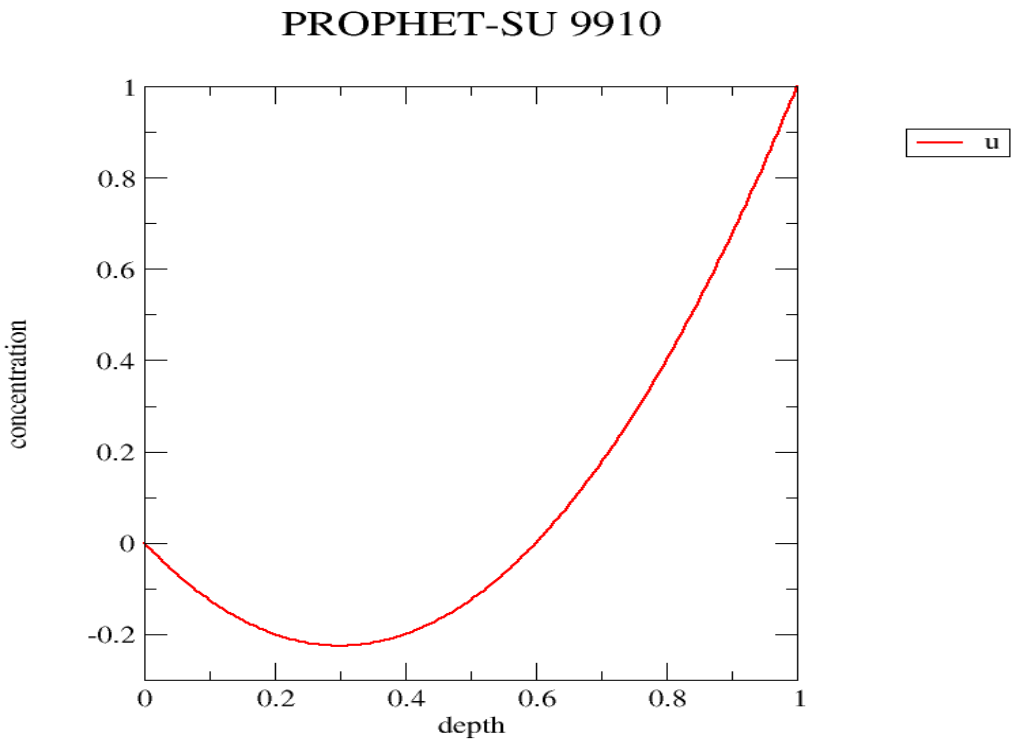
```
#define simulation mesh and boundaries
grid xloc=0,1 xdel=0.005 dim=1
boundary xmin=0 xmax=0 name=contact0
boundary xmin=1 xmax=1 name=contact1

#define fixed charge
field set=c val=5
#initialize sysvar u
field set=u val=0

#solve and plot
solve system=poisson
graph quantity=u
exit
```

PROPHET for 1-D Poisson Eq.

Output of PROPHET script “poisson1d.pf”

```
  8  system      cpu:      0.25 sec  0r  0i       0n       0e  [2704k]
-------------------------------------------------------------------------
Tensor_mesh: dim = 1
Creating tensor mesh: 201  = 201 nodes
  9  grid        cpu:      0.26 sec  1r  3i     201n     200e  [2744k]
-------------------------------------------------------------------------
contact0: 1 elements matched by boundary specification
 10  boundary    cpu:      0.26 sec  1r  4i     201n     200e  [2752k]
-------------------------------------------------------------------------
contact1: 1 elements matched by boundary specification
 11  boundary    cpu:      0.26 sec  1r  5i     201n     200e  [2760k]
-------------------------------------------------------------------------
 12  field       cpu:      0.26 sec  1r  5i     201n     200e  [2768k]
-------------------------------------------------------------------------
 13  field       cpu:      0.26 sec  1r  5i     201n     200e  [2776k]
-------------------------------------------------------------------------
Starting Newton iterations:
  solving for: u
  (linear eq. solver: direct, package: petsc)
  Newton iter 0                0.983u      0.5n     0.005lg     0.005lr
  Newton iter 1             2.03e-14u 4.85e-16n 4.85e-18lg 4.85e-18lr
  Newton converged:      residual norm = 3.24e-16n, reduction = 6.49e-16
                    cpu = 0.02 sec
 14  solve       cpu:      0.30 sec  1r  5i     201n     200e  [2912k]
```

PROPHET for 1-D Poisson Eq.

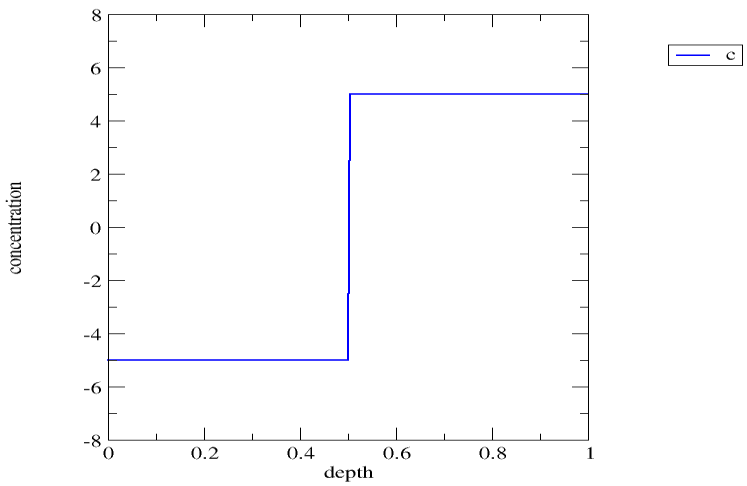


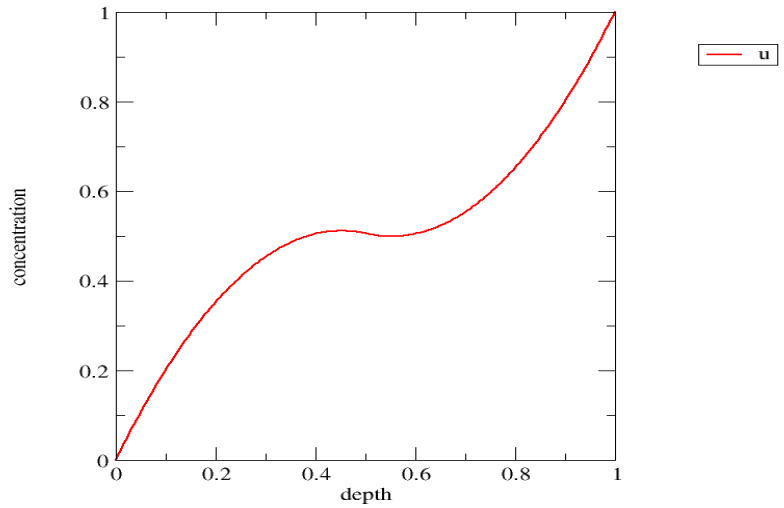
Simulated u(x) from PROPHET script “poisson1d.pf”

PROPHET for 1-D Poisson Eq.

Simulate for different C(x). Only need to change:

```
#define fixed charge
field set=c1 val=-5 xrange=[0:0.5]
field set=c2 val=5 xrange=(0.5:1]
field set=c val=c1+c2
```





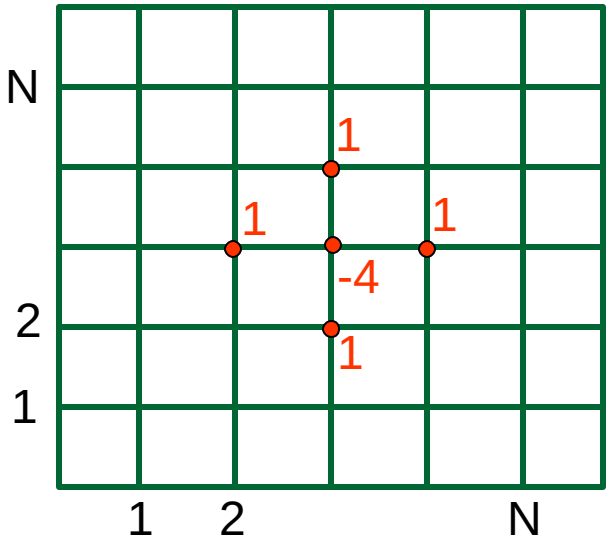
Numerically Solve 2-D Poisson Eq.

- 2-D Poisson eq:

$$\partial^2 u(x,y)/\partial x^2 + \partial^2 u(x,y)/\partial y^2 = C(x,y)$$

- 5-point discretization method

$$\nabla^2 u \approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}}{h^2}$$

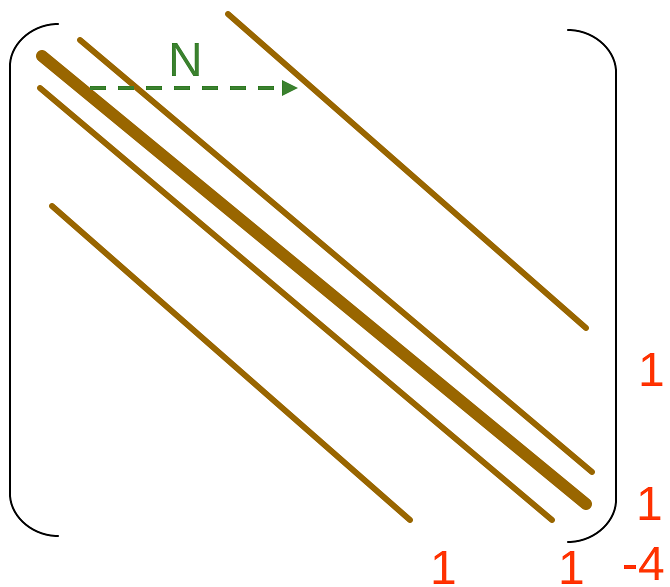


Numerically Solve 2-D Poisson Eq.

- Assemble difference eqns:

$$A \cdot \left(u_{i,j}\right) = \left(h^2 C_{i,j}\right)$$

$$A =$$

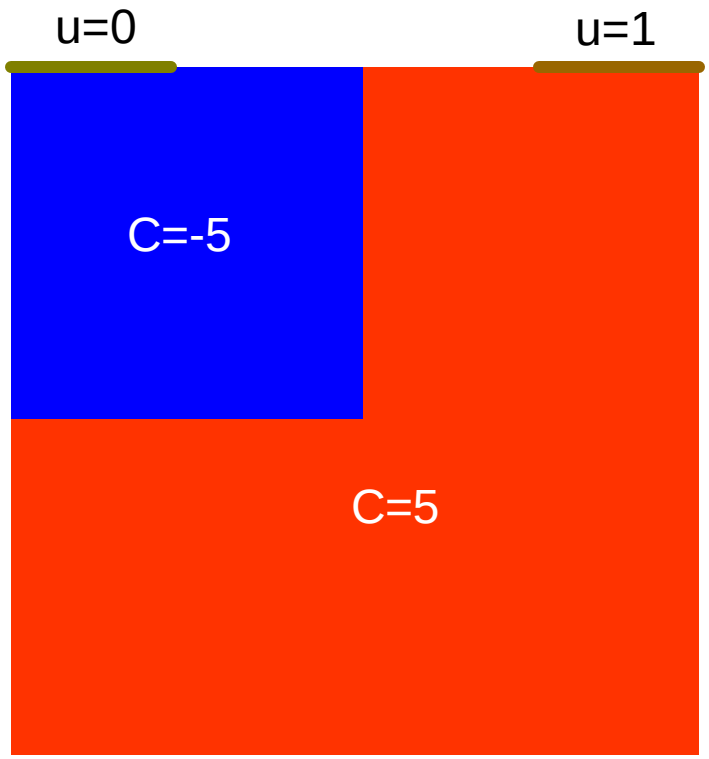


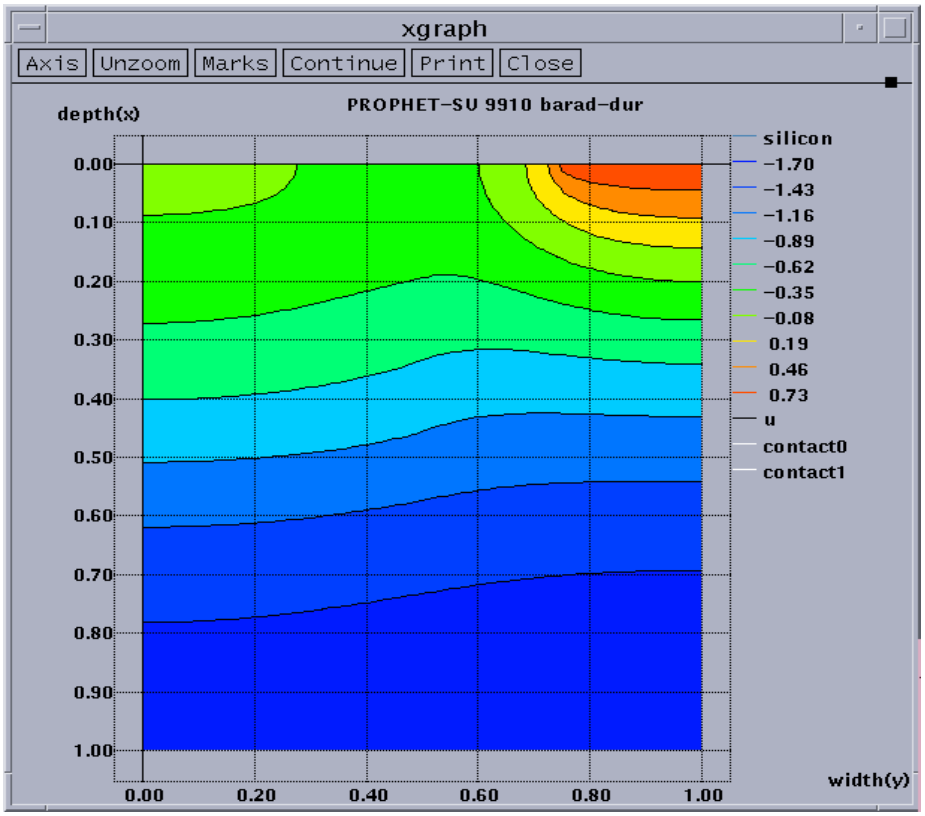
PROPHET for 2-D Poisson Eq.

Part 2 of PROPHET script “poisson2d.pf”

```
#define simulation mesh and boundaries
grid xloc=0,1 xdel=0.02 yloc=0,1 ydel=0.02 dim=2
boundary xmin=0 xmax=0 ymin=0 ymax=0.25 name=contact0
boundary xmin=0 xmax=0 ymin=0.75 ymax=1 name=contact1

#define fixed charge
field set=c1 val=5
field set=c2 val=-10 xrange=[0:0.5] yrange=[0:0.5]
field set=c val=c1+c2
#initialize sysvar u
field set=u val=0

#solve and plot
solve system=poisson
graph quantity=u contour color
```

PROPHET for 2-D Poisson Eq.





u(x) simulated with PROPHET

PROPHET command syntax

- General command format

```
Command_name keyword=value keyword=value …
Keyword value types: logical, real, integer, string …
```

- Examples:

```
bias system=silicon_dd nstep=5 vstep=0.1 elec=anode
field set=c val=5 xrange=[0:0.1]
dbase create name=library/physics/silicon/psi/Dix rval=8.85e-14
```

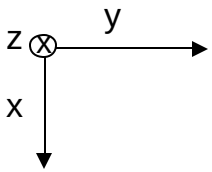
- Notice that:
  - Lines that start with “#” are comments
  - Long lines can be broken using “+” as connectors

PROPHET command syntax

- Keyword value types:
  - logical: true (“t”, “1” or by default)/false (“f” or “0”)
  - real: floating point value (140.0, 140 or 1.4e2; expressions are allowed)
  - real array: separated by commas (“0,0.01,0.1,1”)
  - integer and integer array
  - string: string of characters (system=silicon_dd)
  - string array: array of strings (elec=gate,drain)

PROPHET command: GRID

- GRID: generates a 1D/2D/3D rectlinear grid
  - xloc/yloc: real arrays that define grid segments
  - xdel/ydel: grid size for each segment
  - dim: grid dimension
  - material: region material



- Examples

```
grid dim=1 xloc=0,0.05,1 xdel=0.005,0.01,0.1 mat=silicon
```

```
grid dim=2 xloc=0,0.05,1 xdel=0.005,0.01,0.1
+    yloc=0,0.5,1 ydel=0.05,0.05,0.1 mat=silicon
```

PROPHET command: DEPOSIT

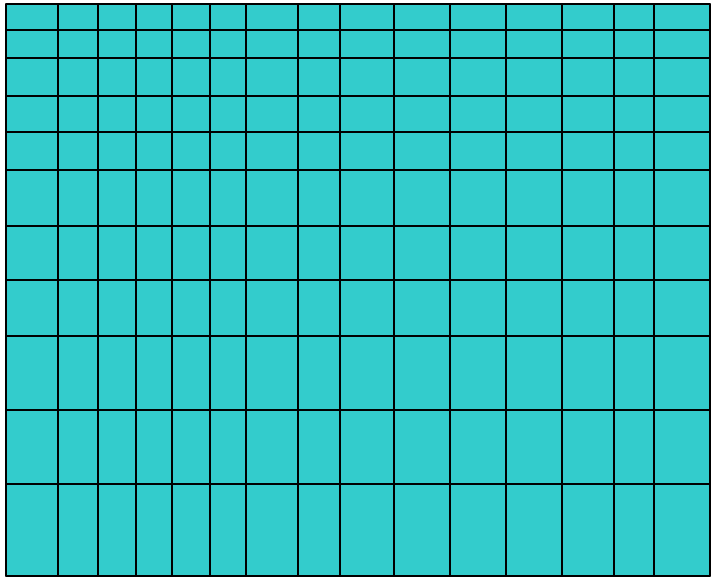
- Add a material layer onto existing structure
  - thickness: thickness of deposited layer
  - xdel: grid size of deposited layer
  - start/end: define area of selective deposition
  - material:
- Examples:

```
deposit mat=oxide thick=0.002 xdel=0.002 #Gate oxide
deposit mat=poly start=-0.03 end=0.03 thick=0.0,0.1 xdel=0.002,0.05 #poly gate
deposit mat=oxide start=-0.05 end=-0.03 thick=0.0,0.1 xdel=0.002,0.05 #spacer
deposit mat=oxide start=0.03 end=0.05 thick=0.0,0.1 xdel=0.002,0.05 zipper
```

Example: GRID/DEPOSIT

PROPHET script “deposit.pf”

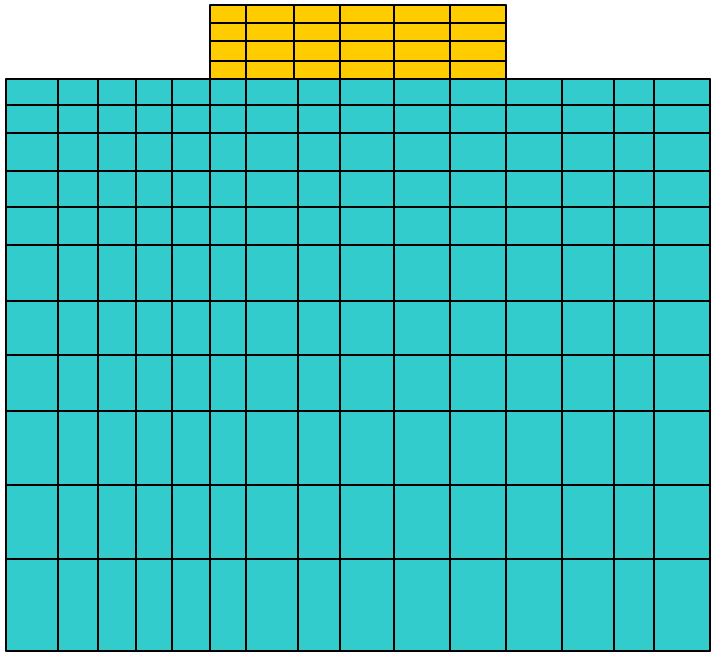
```
#substrate
grid dim=2 xloc=0,0.1,1 xdel=0.01,0.02,0.1
+    yloc=0,1 ydel=0.1 mat=silicon
```



Example: GRID/DEPOSIT

PROPHET script “deposit.pf”

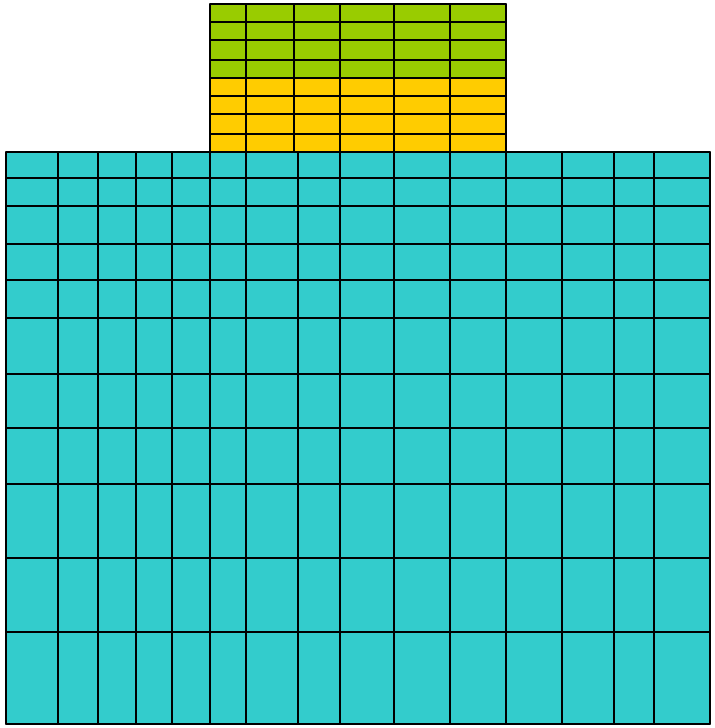
```
#substrate
grid dim=2 xloc=0,0.1,1 xdel=0.01,0.02,0.1
+    yloc=0,1 ydel=0.1 mat=silicon
#gate oxide
deposit thick=0.1 xdel=0.01
+     mat=oxide start=0.3 end=0.7
```



Example: GRID/DEPOSIT

PROPHET script “deposit.pf”

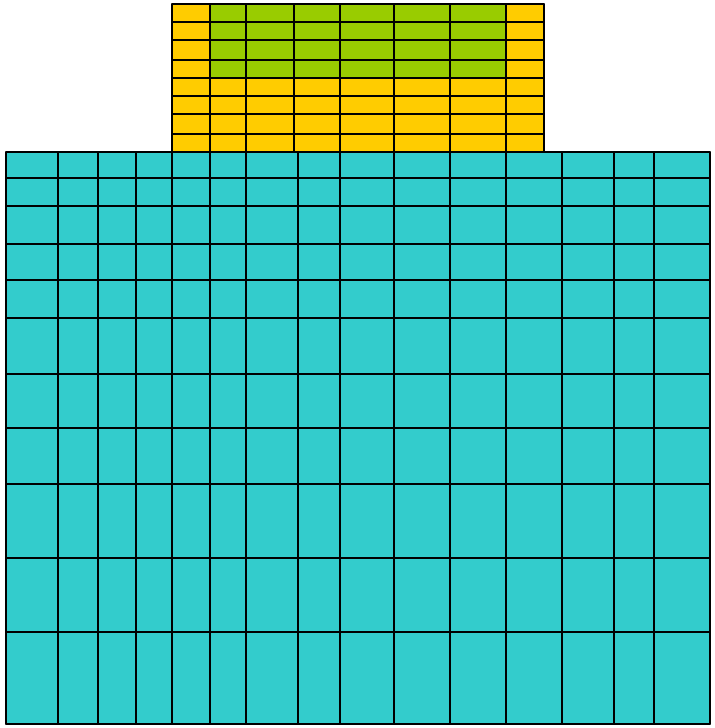
```
#substrate
grid dim=2 xloc=0,0.1,1 xdel=0.01,0.02,0.1
+    yloc=0,1 ydel=0.1 mat=silicon
#gate oxide
deposit thick=0.1 xdel=0.01
+     mat=oxide start=0.3 end=0.7
#poly gate
deposit thick=0.1 xdel=0.01
+     mat=poly start=0.3 end=0.7
```



Example: GRID/DEPOSIT

PROPHET script “deposit.pf”

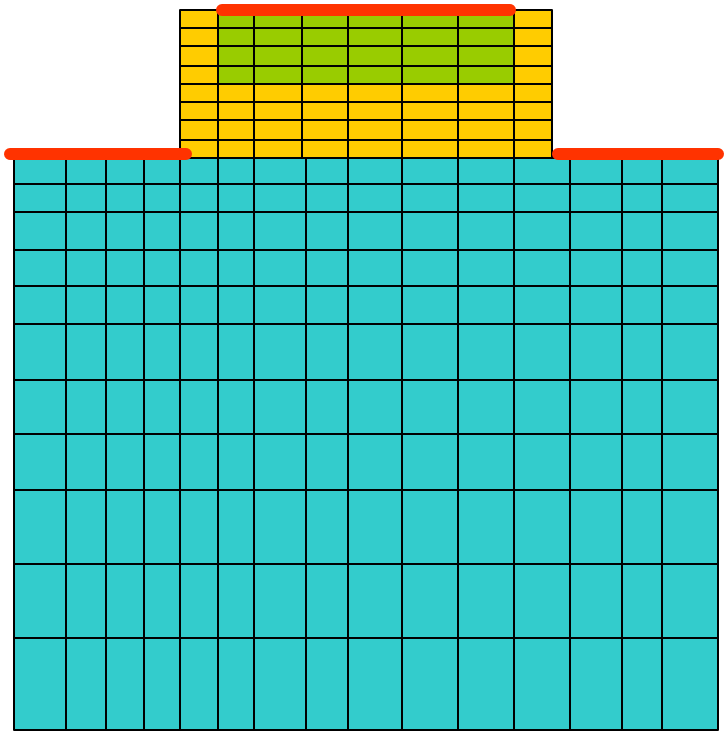
```
#substrate
grid dim=2 xloc=0,0.1,1 xdel=0.01,0.02,0.1
+    yloc=0,1 ydel=0.1 mat=silicon
#gate oxide
deposit thick=0.1 xdel=0.01
+     mat=oxide start=0.3 end=0.7
#poly gate
deposit thick=0.1 xdel=0.01
+     mat=poly start=0.3 end=0.7
#spacer oxide
deposit thick=0.2 xdel=0.01
+     mat=oxide start=0.25 end=0.3
deposit thick=0.2 xdel=0.01 mat=oxide
+     start=0.7 end=0.75 zipper
```



PROPHET command: BOUNDARY

- Define device contacts
  - xmin/ymin,xmax/ymax
  - name, material, …
- Example

```
#define source/drain contacts
boundary xmin=0 xmax=0 ymin=0.0
+  ymax=0.25 name=source
boundary xmin=0 xmax=0 ymin=0.75
+  ymax=1.0 name=drain
#define gate contact
boundary xmin=-0.2 xmax=-0.2 ymin=0.3
+  ymax=0.7 name=gate
```



PROPHET command: FIELD

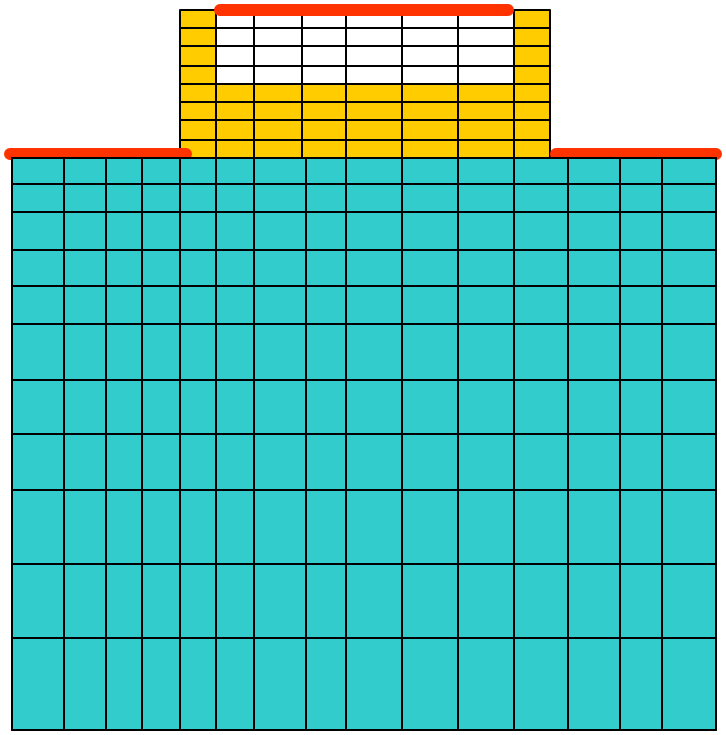
- Defines various fields over existing grids
  - set: name of the field
  - material: limit field definition to certain materials
  - value: an expression evaluated across grids; can include constants, other fields, coordinates (X,Y,Z upper case). Example:

```
field set=c val=5*gbox(X,0,1,0.01)*gbox(Y,1,2,0.01)
```

specifies a flat profile of 5 inside the interval (X,Y)=(0~1,1~2) with Gaussian decay of characteristic length 0.01 outside

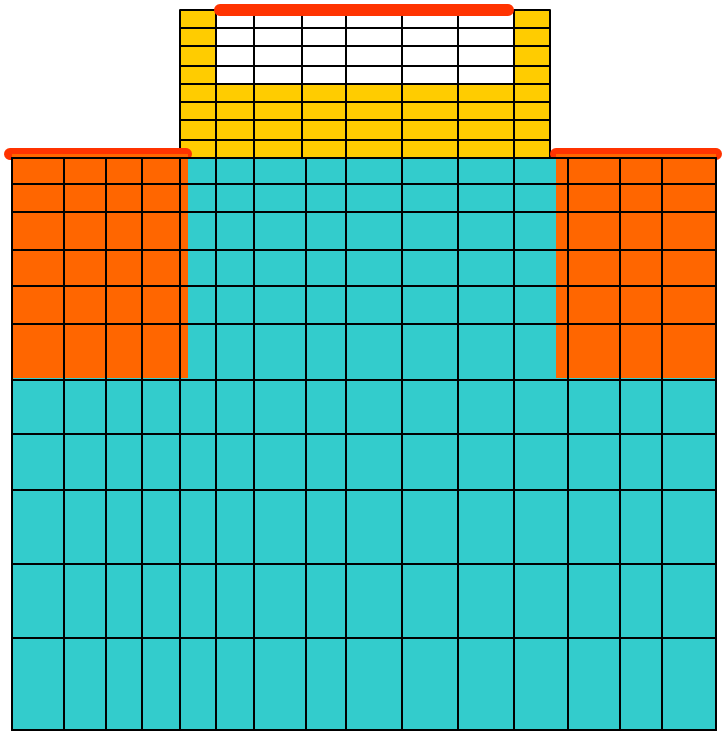
Example: FIELD

```
#p-substrate
field set=psub val=-1 mat=silicon
```



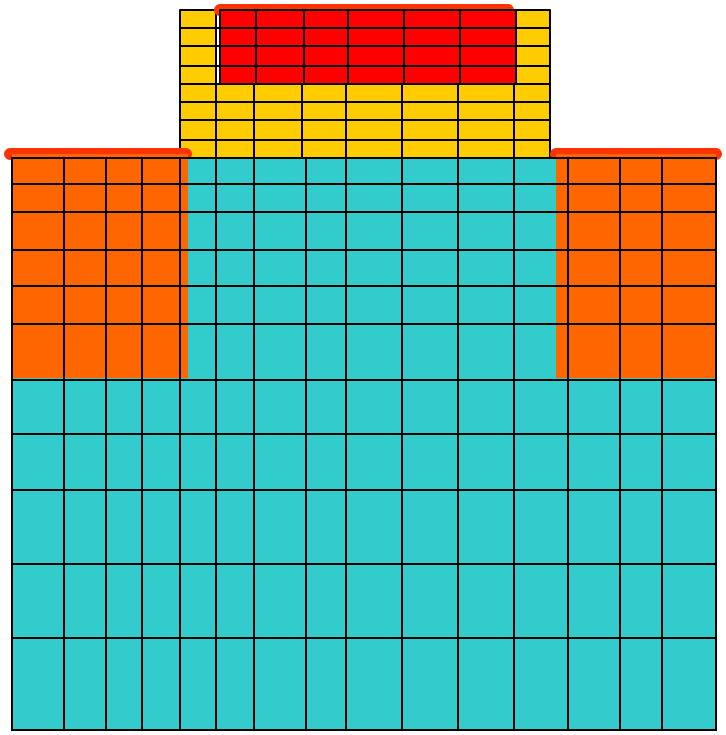
Example: FIELD

```
#p-substrate
field set=psub val=-1 mat=silicon
#n-diffusion at source/drain
field set=ns mat=silicon
+  val=6*gbox(X,0,0.3,0.01)*gbox(Y,0,0.25,0.01)
field set=nd mat=silicon
+  val=6*gbox(X,0,0.3,0.01)*gbox(Y,0.75,1,0.01)
```



Example: FIELD

```
#p-substrate
field set=psub val=-1 mat=silicon
#n-diffusion at source/drain
field set=ns mat=silicon
+  val=6*gbox(X,0,0.3,0.01)*gbox(Y,0,0.25,0.01)
field set=nd mat=silicon
+  val=6*gbox(X,0,0.3,0.01)*gbox(Y,0.75,1,0.01)
#n-poly
field set=npoly val=11 mat=poly
#total
field set=c val=psub+ns+nd+poly
```



PROPHET command: SOLVE/BIAS

- SOLVE: solves steady-state/transient PDEs
  - system, seconds, …

```
solve system=my_diffusion seconds=0.01
```

- BIAS: more features for pre-defined PDEs
  - system, init, electrode, nstep, voltage, vstep, time

```
bias system=silicon_poisson init
bias system=silicon_dd elec=anode voltage=0.5
bias system=silicon_dd elec=gate vstep=0.1 nstep=20
bias system=silicon_dd elec=gate voltage=1.0 time=1e-6
```

PROPHET command SAVE/LOAD/GRAPH

- SAVE/LOAD in PROPHET native format

```
save pas=my_res1.pas
load pas=my_res1.pas
```

- GRAPH: visualize simulation results
  - quantity, xpos, ypos, xmin, xmax, region, contour, color, gridline, iv, print, outfile

```
graph quantity=psi ypos=0.0 xmin=-1.0 xmax=1
graph quantity=netdope contour color
graph quantity=gridline
graph quantity=electrons xpos=0.5 print outfile=electrons.txt
```

Conclusion

- General introduction to device simulations and PROPHET
- Getting started with PROPHET: examples of Poisson equation
- PROPHET command syntax: grid, deposit, field, boundary, solve, bias, save, load, graph

PROPHET Roadmap

- Lect. #1: basics
  - Problem set
    - Description in Breeze Lab #1
    - Access through nanoHUB
- Lect. #2: PDE def.
- Advanced topics
  - Nano-bio devices
  - Nano-wire devices
  - Organic semiconductor nano-devices
  - http://www.nanohub.org