$\vec{E}$mission

A program with great potential.

Tomas Kozlej

Siddhartha Menon

Odysseas Sierepeklis

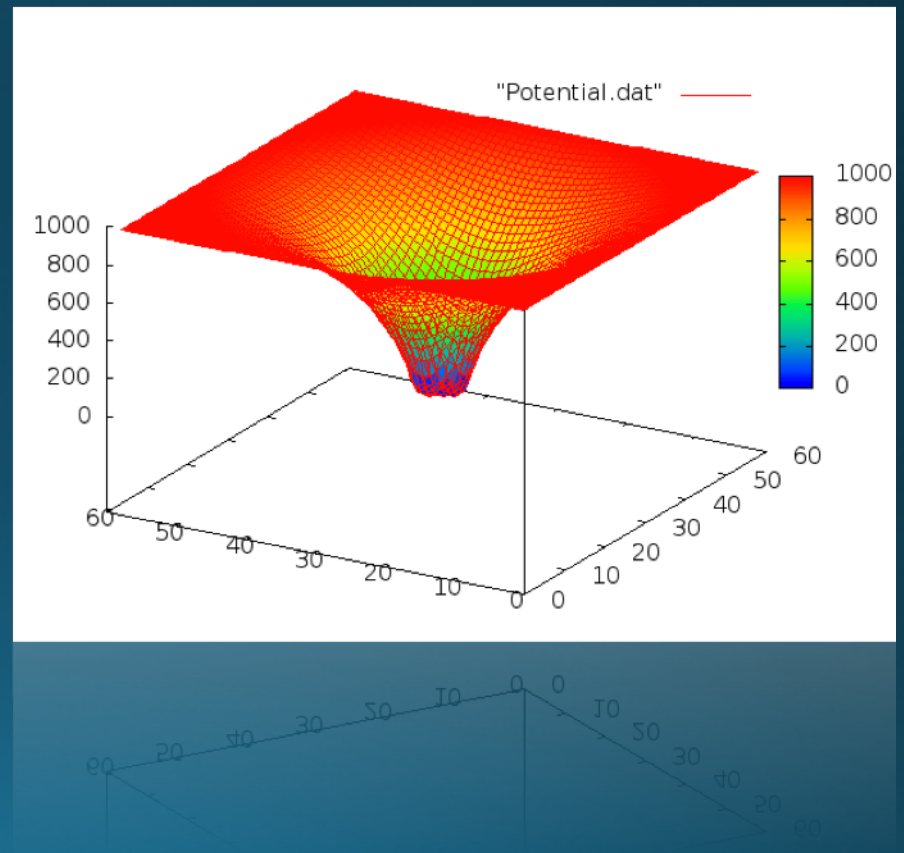
Nikolas Demetriou

Antonio Smecca

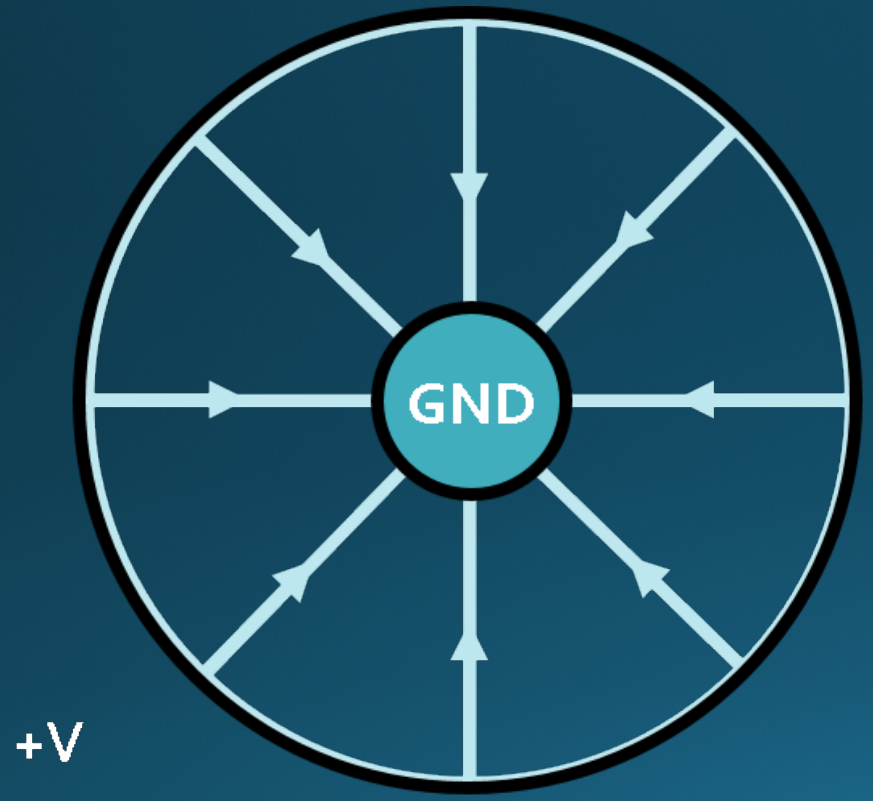
Motivation

Aims

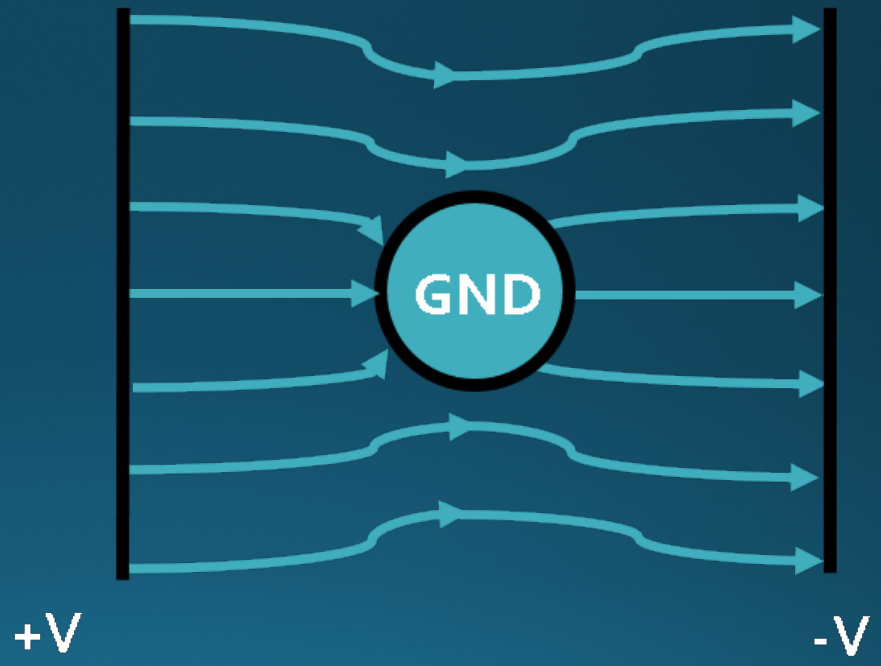
- Application simulating electric potentials.
- Quick and intuitive numerical solver for general charged geometries.
- User freedom.



Analytical Problems $\nabla^2 V = 0$ Laplace's equation

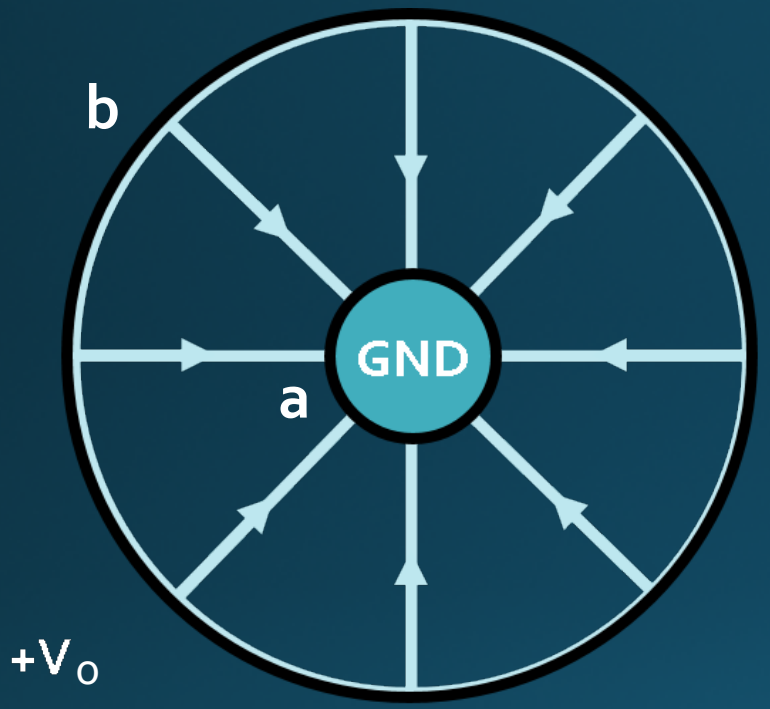


Problem 0



Problem 1

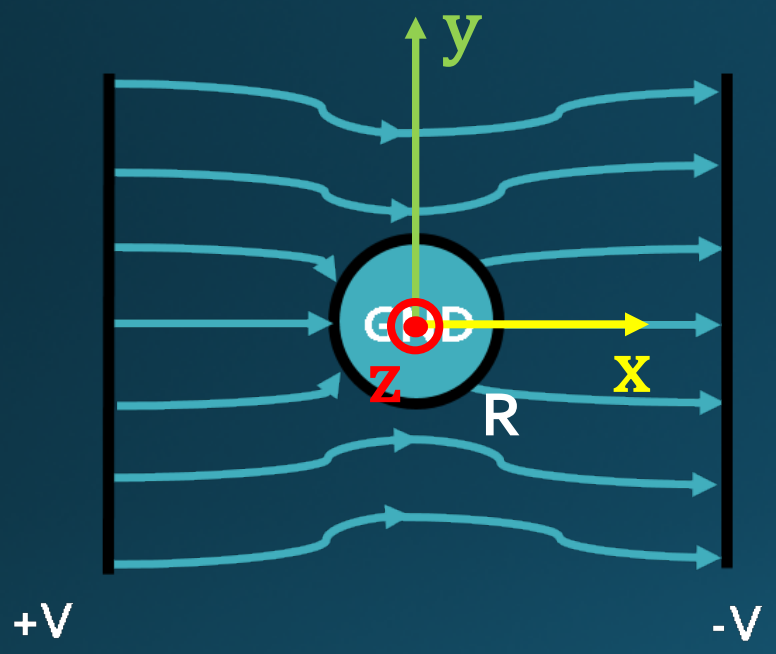
Problem 0



1. Cylindrical form of the Laplace equation
2. Omit the angular coordinate θ and height z
3. Integrate twice
4. Use boundary conditions on surface of conductor and shell

$$V(r) = \frac{V_0}{k}\ln\left(\frac{r}{a}\right) \; where \; k = \ln\left(\frac{b}{a}\right)$$

Problem 1



Maths:

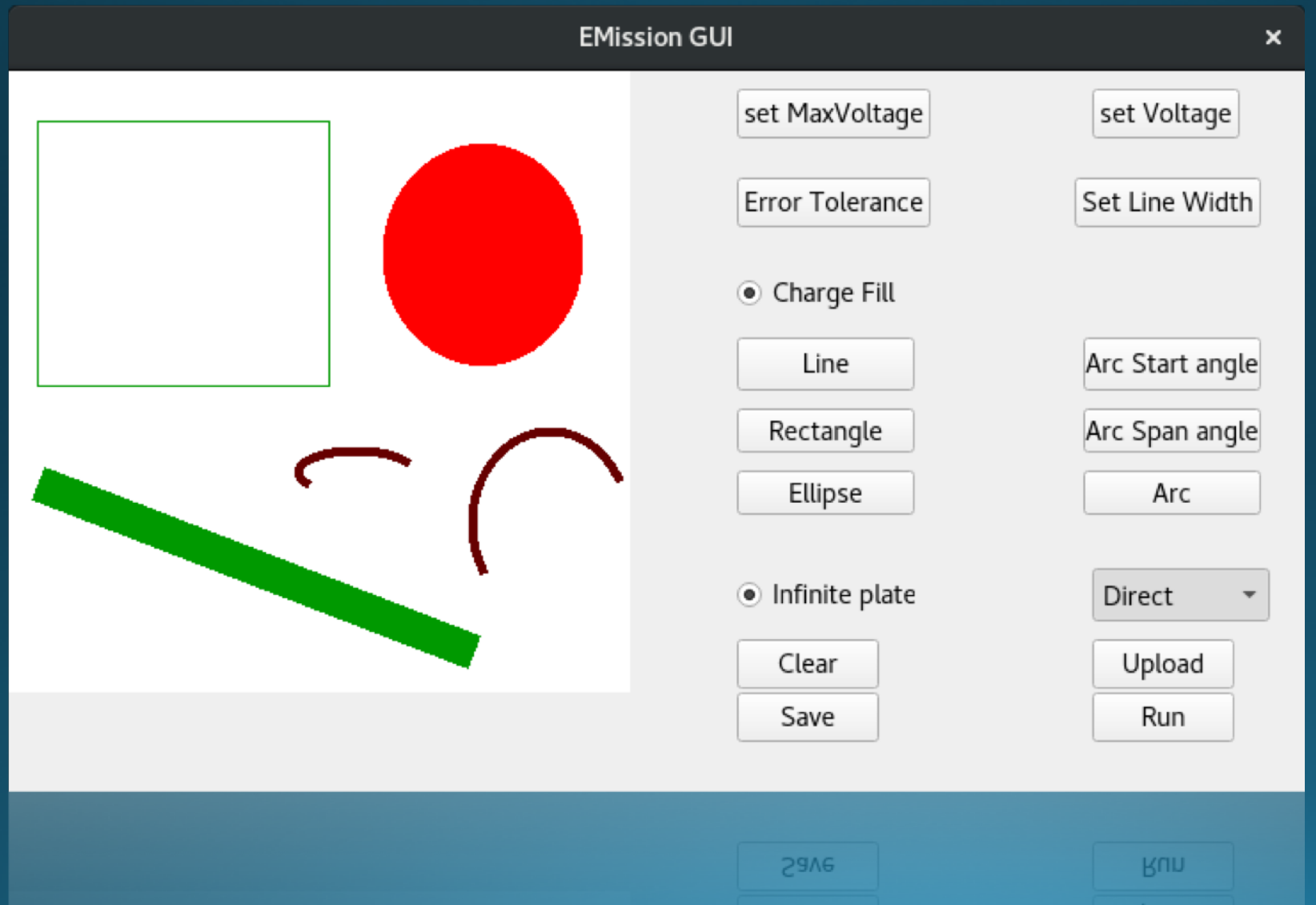
1. Cylindrical form of the Laplace equation
2. Omit the height z
3. Get general solution through separation of variables

Physics:

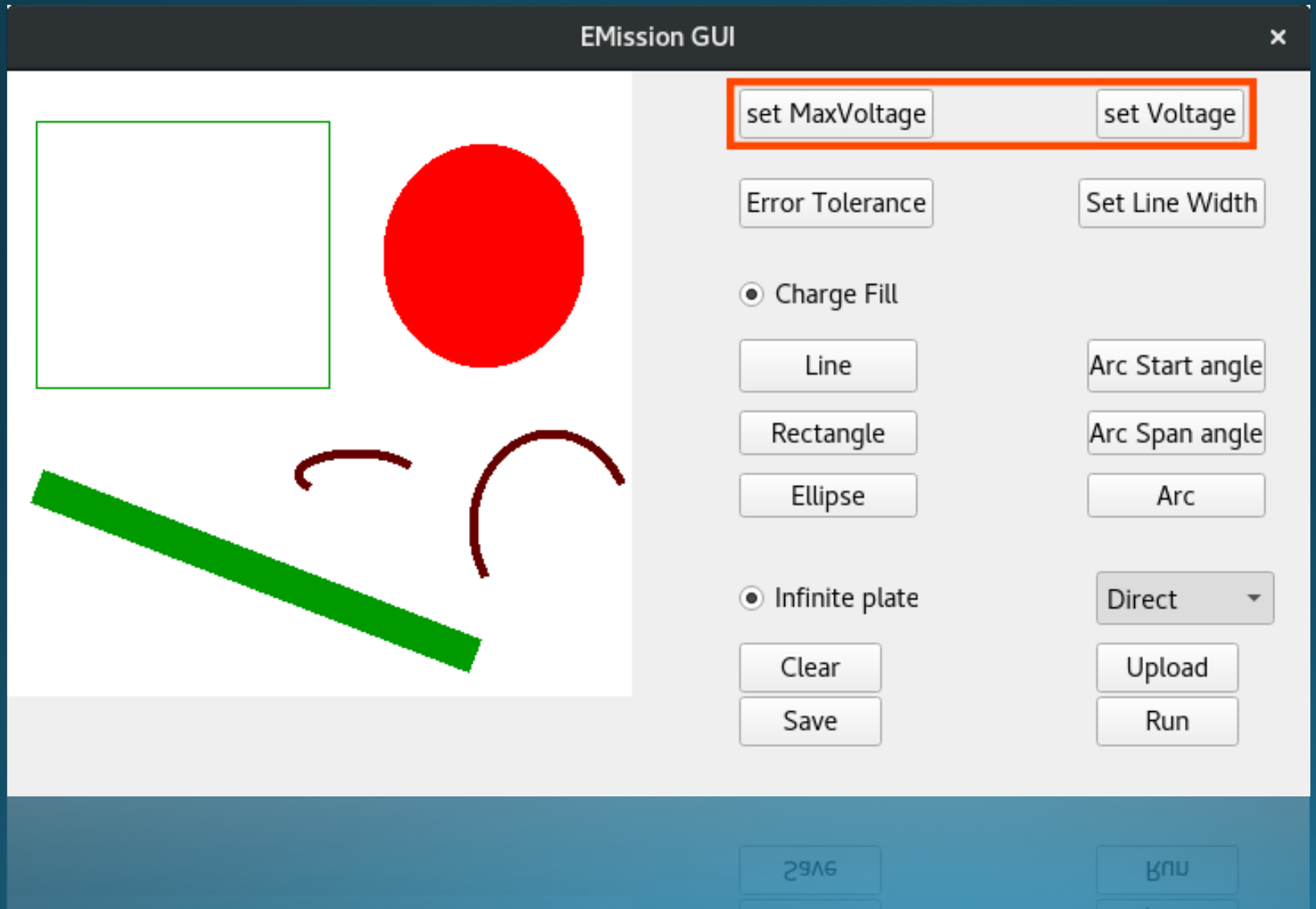
1. Potential at r >> R unaffected by conductor, $V = -E_0 x + K$
2. No charge on conductor's surface
3. Compare with general solution

$$V(r,\theta) = -E_0\left(r - \frac{R^2}{r}\right)\cos(\theta)$$

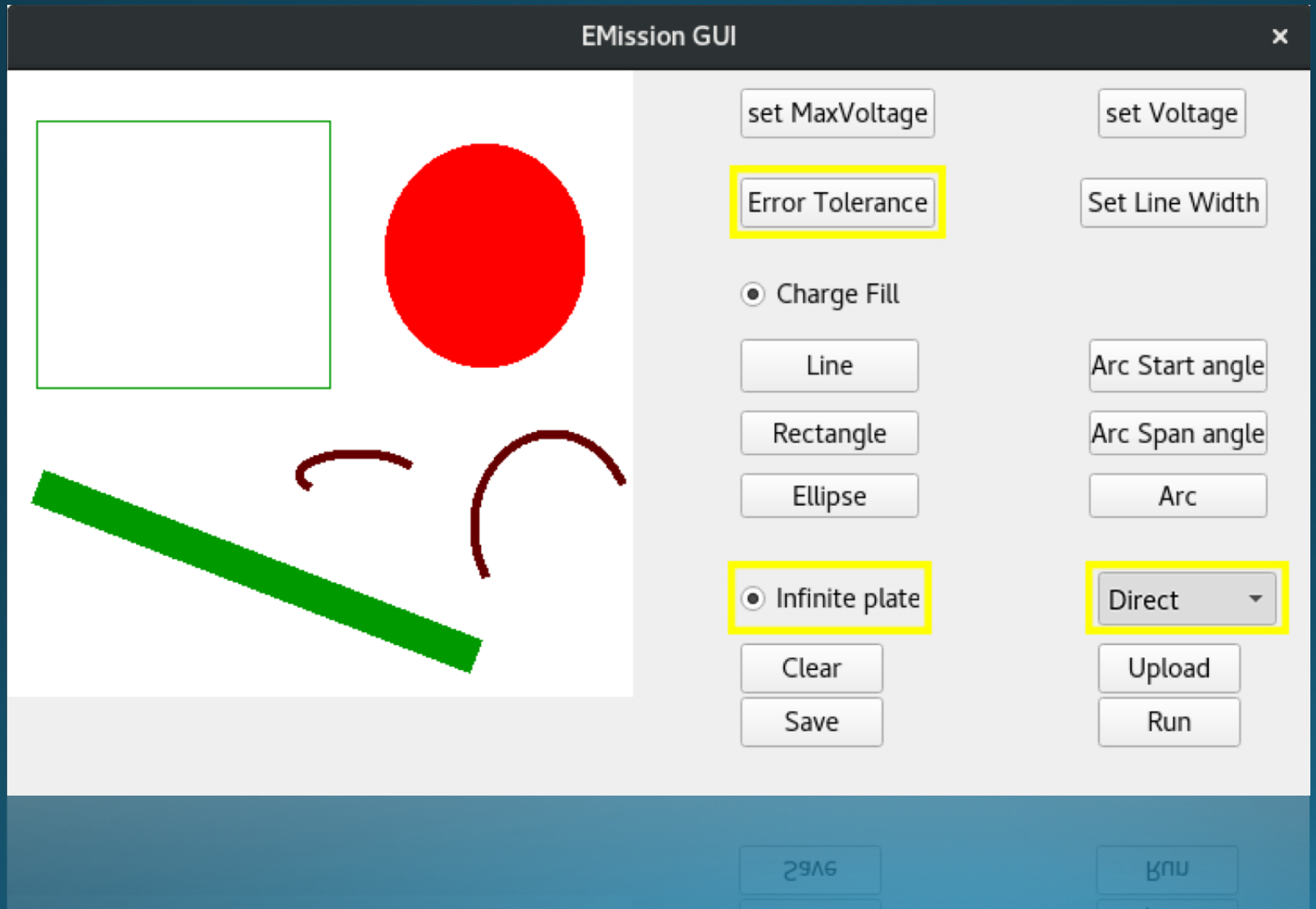
Emission GUI



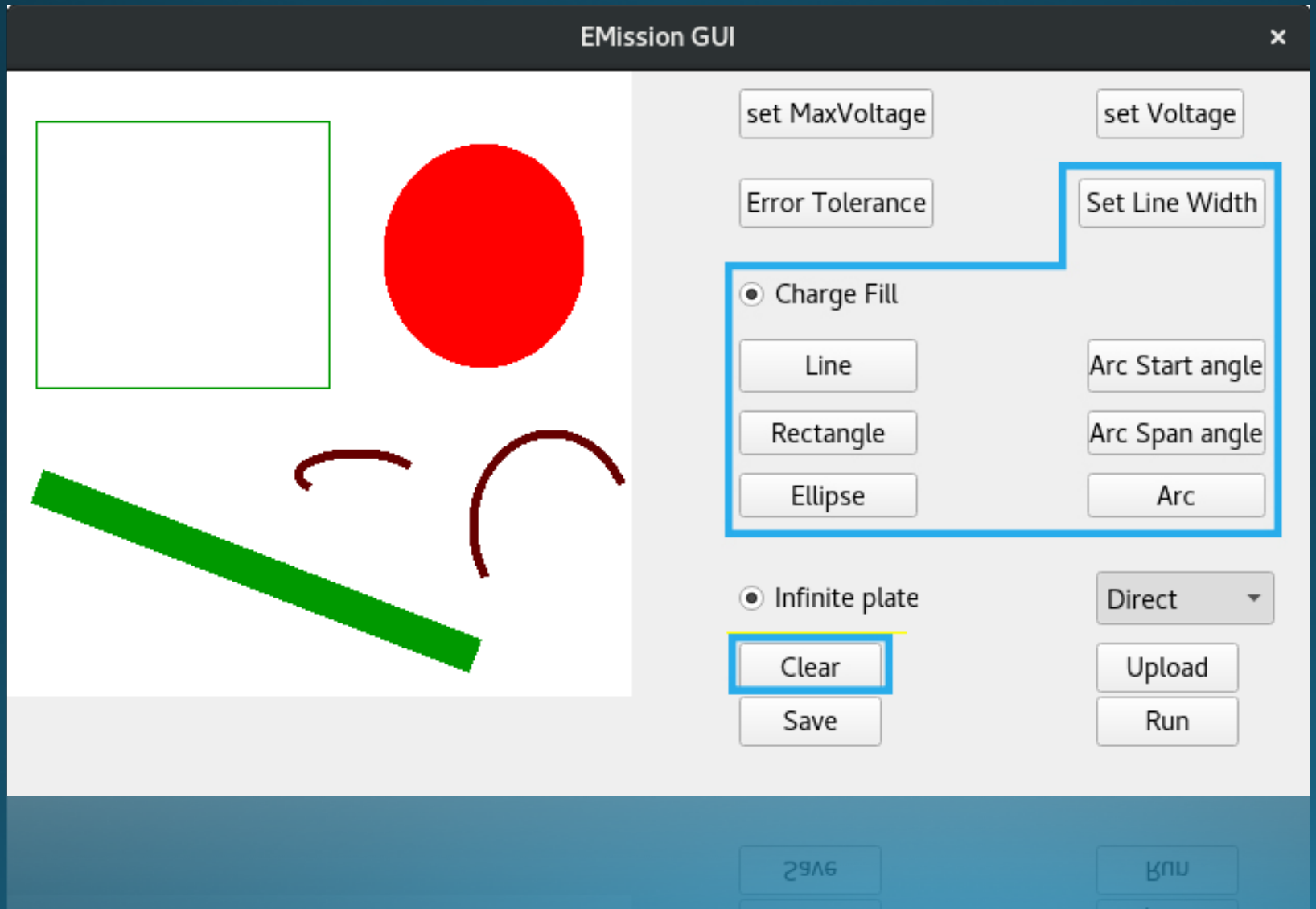
Emission GUI



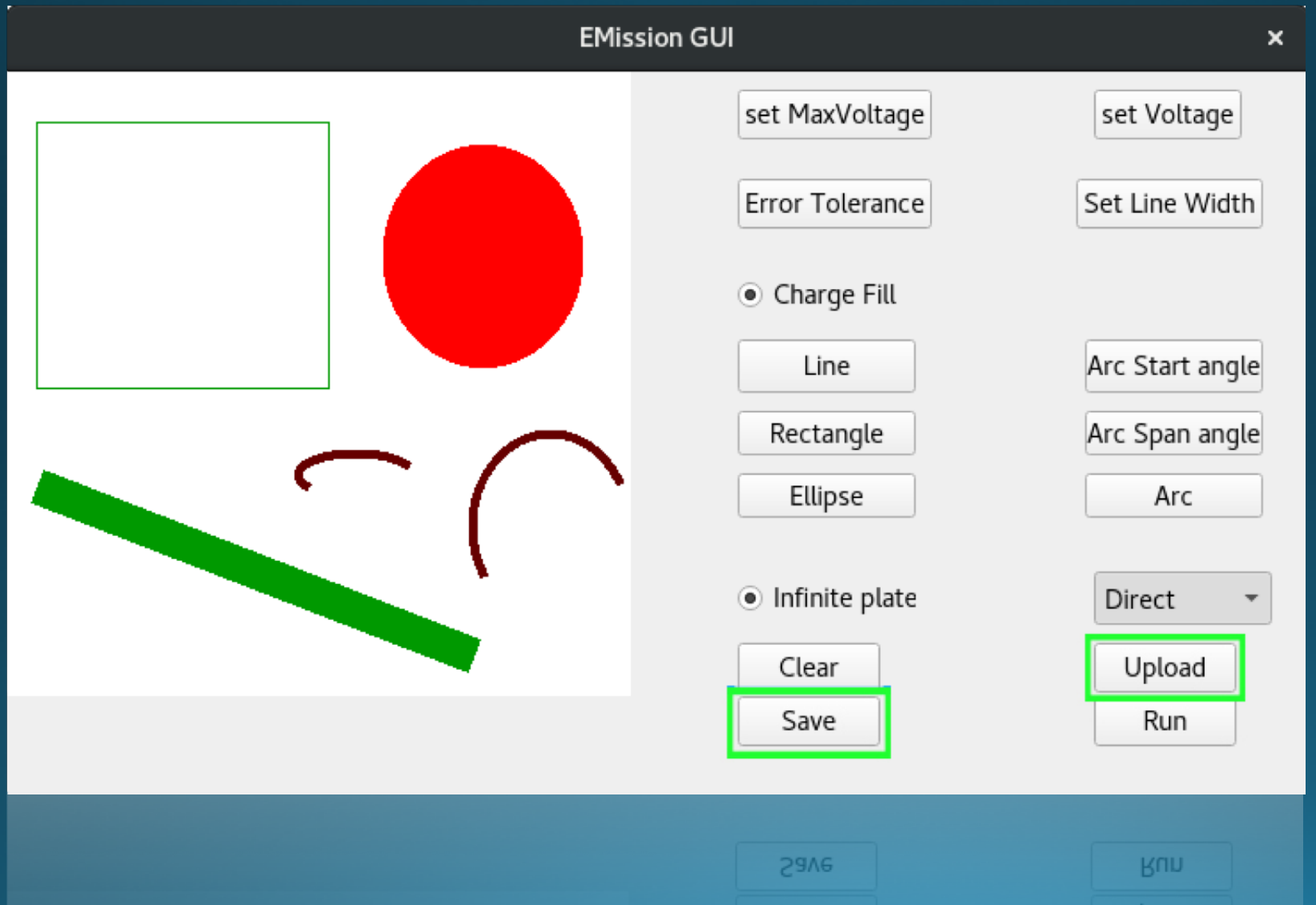
Emission GUI



Emission GUI

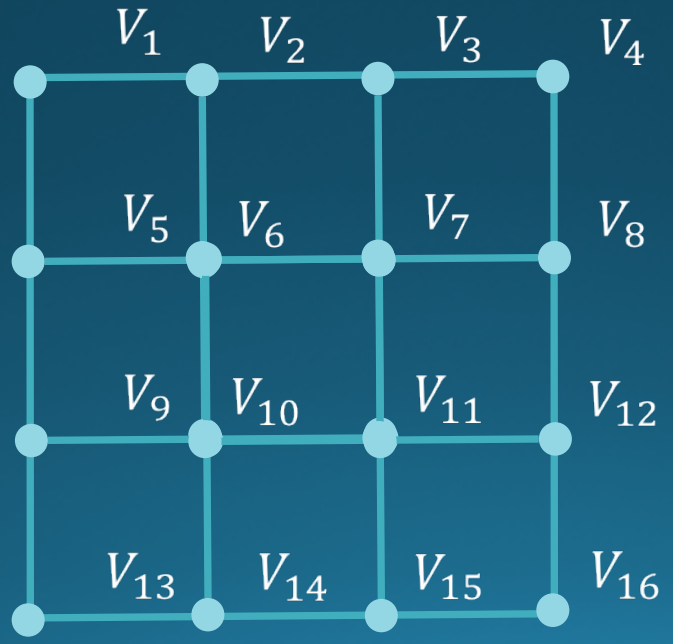


Emission GUI



Outline of the Direct Method:

1. For every point there is an associated equation derived from FPS
2. N x N grid ⇒ N^2 simultaneous equations
3. Putting this in a matrix of form $Ax = b$



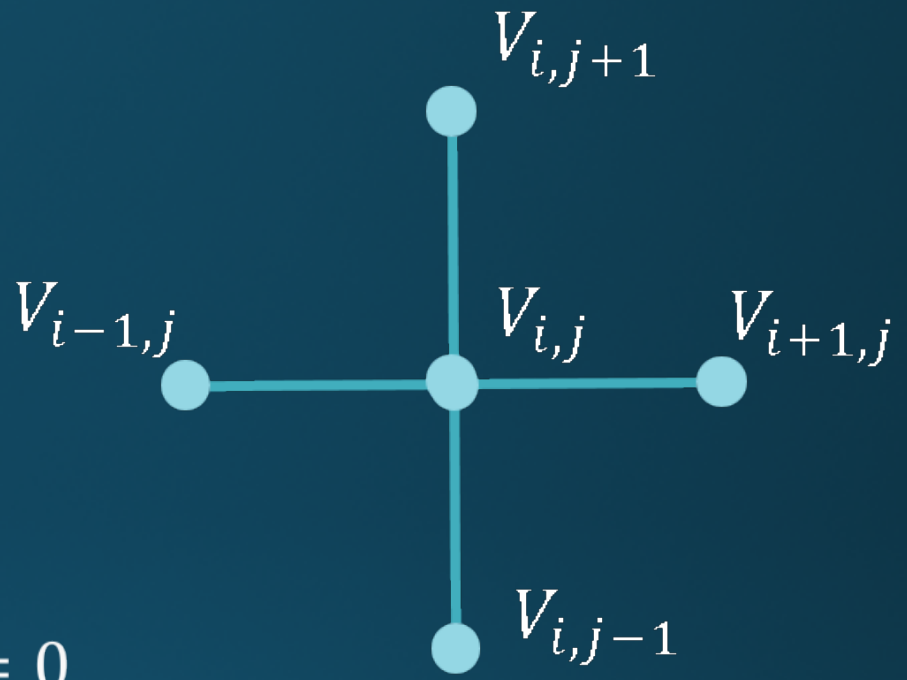
Numerical Approach

$$\nabla^2 \mathrm{V} = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

$$\approx \frac{V_{i+1,j} + 2V_{i,j} + V_{i-1,j}}{\delta x^2} + \frac{V_{i,j+1} + 2V_{i,j} + V_{i,j-1}}{\delta y^2} = 0$$

$$\delta x = \delta y \Rightarrow$$

$$V_{i,j} = \frac{1}{4}\left(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}\right)$$



Five Point Stencil

Equations for the Grid

- The equations for the whole grid:
- $-4V_1 + V_2 + 0V_3 + 0V_4 + V_5 + \cdots + 0V_{16} = 0$
- $V_1 - 4V_2 + V_3 + 0V_4 + 0V_5 + V_6 + \cdots + 0V_{16} = 0$
- ⋮ *13 more equations*
- $0V_1 + \cdots + V_{12} + 0V_{13} + 0V_{14} + V_{15} - 4V_{16} = 0$

- How can we express this as a matrix?

$$\begin{pmatrix}
-4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4
\end{pmatrix}$$

Matrix Formulation

$$\begin{pmatrix} -4 & 1 & 0 & & 0 & 0 & 0 \\ 1 & -4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -4 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & -4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & -4 & 1 \\ 0 & 0 & 0 & & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_{14} \\ V_{15} \\ V_{16} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

This matrix equation is now of the form $Ax = b$ but is not quite complete. We need to tailor it to our specific geometry.

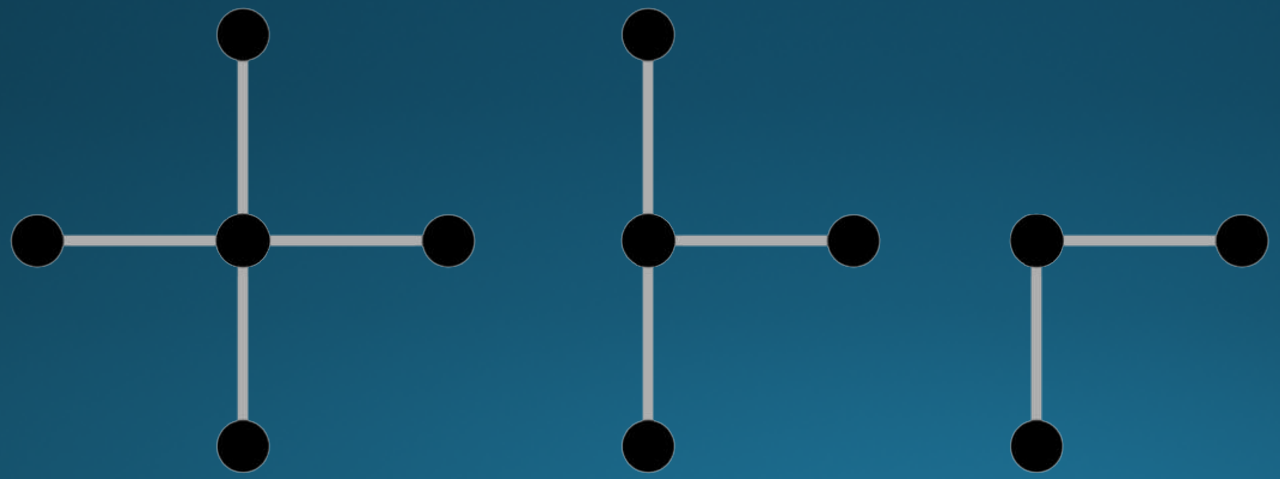
Tailoring the Matrix

Say V_2 is known to be 25V. We can then edit our system to reflect this:

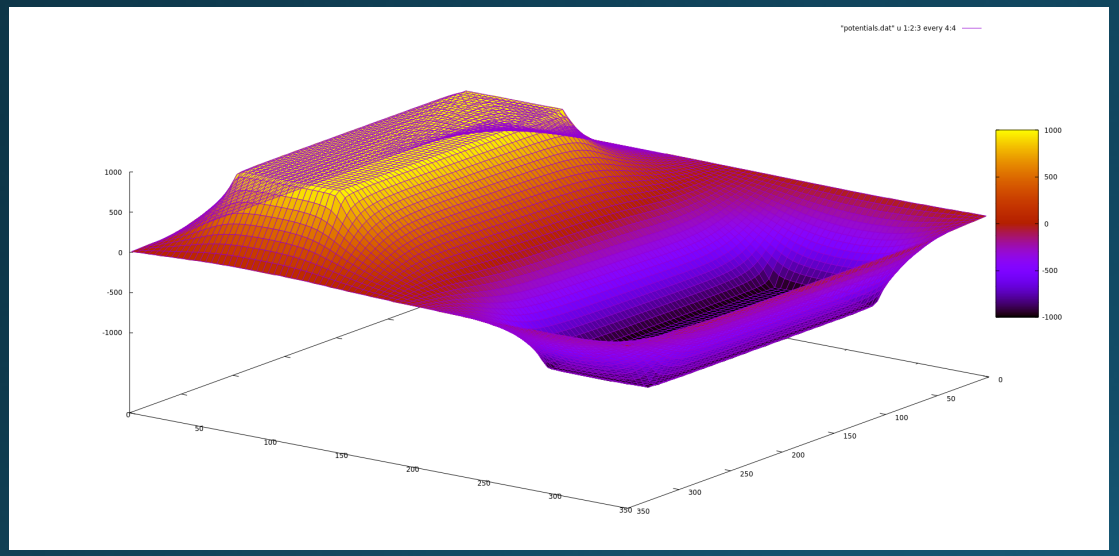
$$\begin{pmatrix} -4 & 1 & 0 & & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -4 & & 0 & 0 & 0 \\ & \vdots & & \ddots & \vdots & & \\ 0 & 0 & 0 & & -4 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & -4 & 1 \\ 0 & 0 & 0 & & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_{14} \\ V_{15} \\ V_{16} \end{pmatrix} = \begin{pmatrix} 0 \\ 25 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Adaptive Stencilling

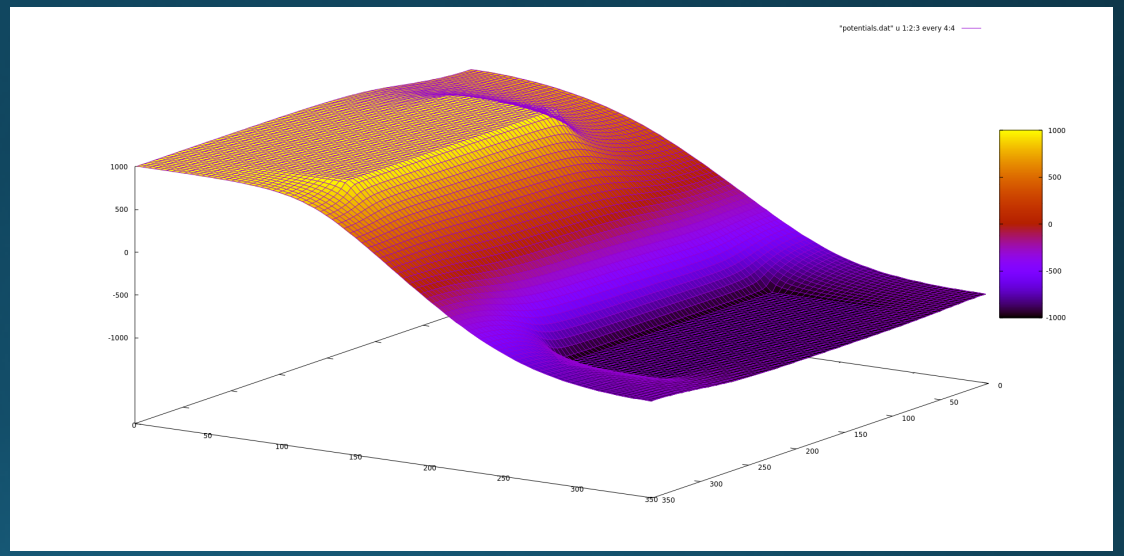
- Adaptive Stencilling is a method where in the number of *legs* in the stencil is modified depending on where we are on the grid.
- This avoids the problem of artificially depressing the boundaries.



Adaptive Stencilling



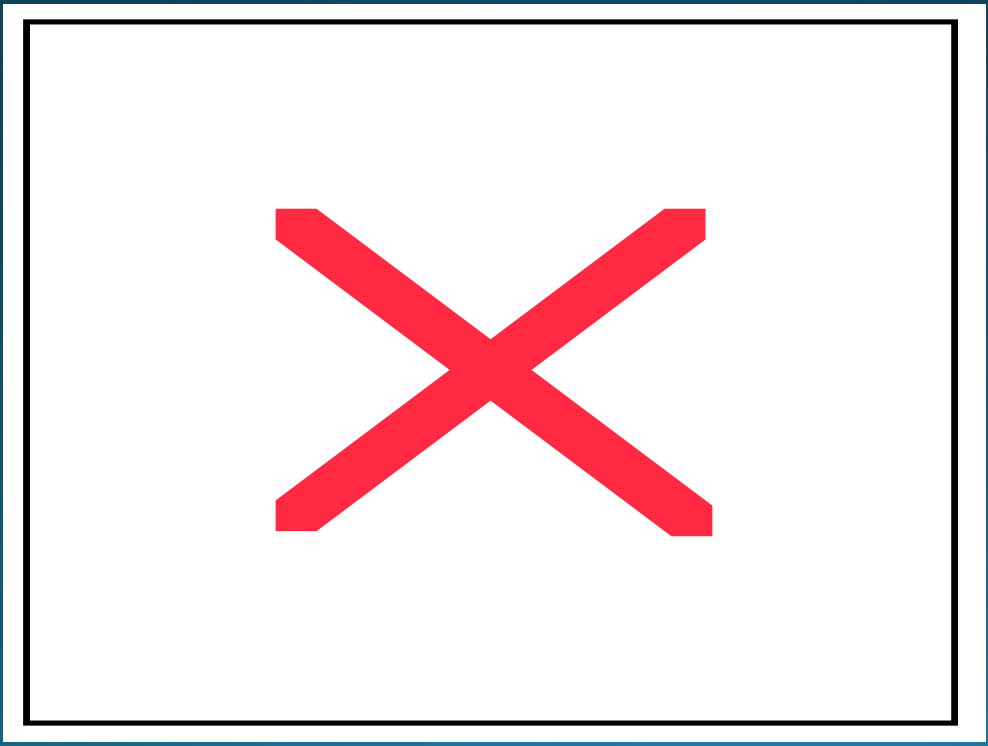
Before



After

Sparse Storage Systems

Assuming an integer is stored in 4 bytes this would imply the matrix A associated with a 400 x 400 problem would require around 95 GB to store!



Sparse Triplet Format

1	1	-4
1	2	1
1	5	1
⋮	⋮	⋮

Sparse triplet storage cuts the storage size of A from $O(N^4)$ to $O(N^2)$.

Compressed Column Storage

These column offsets tell us where the next column starts in the row array.

nz values	-4	1	1	1	-4	1	…
row index	1	2	4	1	2	3	…
col shift	1	4	…				

Solving the System

- A can never be stored in all its uncompressed glory.
- For maximum efficiency we use the SuperLU package linked with the ATLAS library.
- SuperLU accepts matrices in CCS but it is easier to generate them in sparse triplet so we use the CSPARSE package to handle the conversion.

LU Decomposition

- The basic idea is to factorise the matrix A into lower and upper triangular matrices L and U.
- The system $Ax = b$ then becomes $LUx = b$
- Letting $Ux = y$ we get the system

$$Ly = b$$
$$Ux = y$$

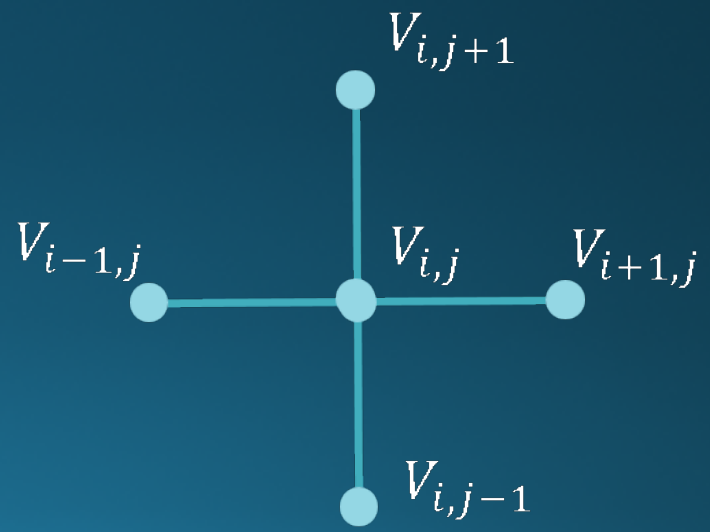
- These equations can be solved by forward, and back substitution respectively and thus no matrix inversion is necessary.

Iterative Method

- Every grid point estimated by the diffusion equation:

$$0 = \nabla^2 V \Rightarrow \frac{\partial V}{\partial t} = \nabla^2 V$$

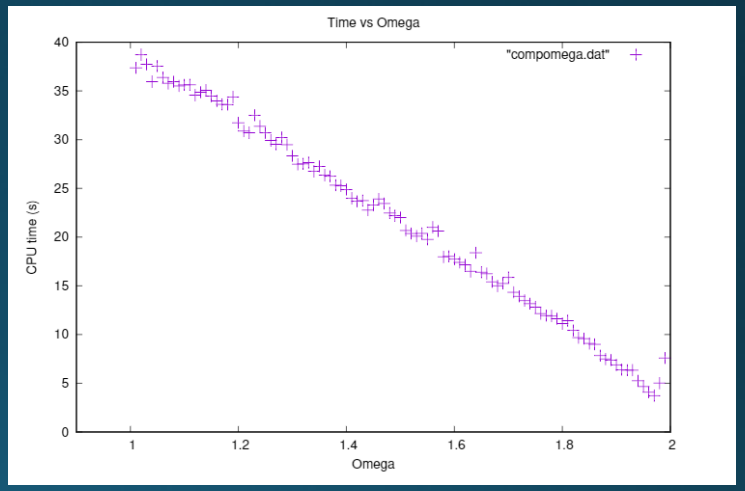
- t is a dummy index of estimations.
- First guess is poor.
- Eventual Convergence to Laplace
- Can we do better?



Iterative Optimization

- We introduce Successive Over-relaxation (SOR)
- 'Steal' future information at rate ω.
- ω is called the relaxation factor.

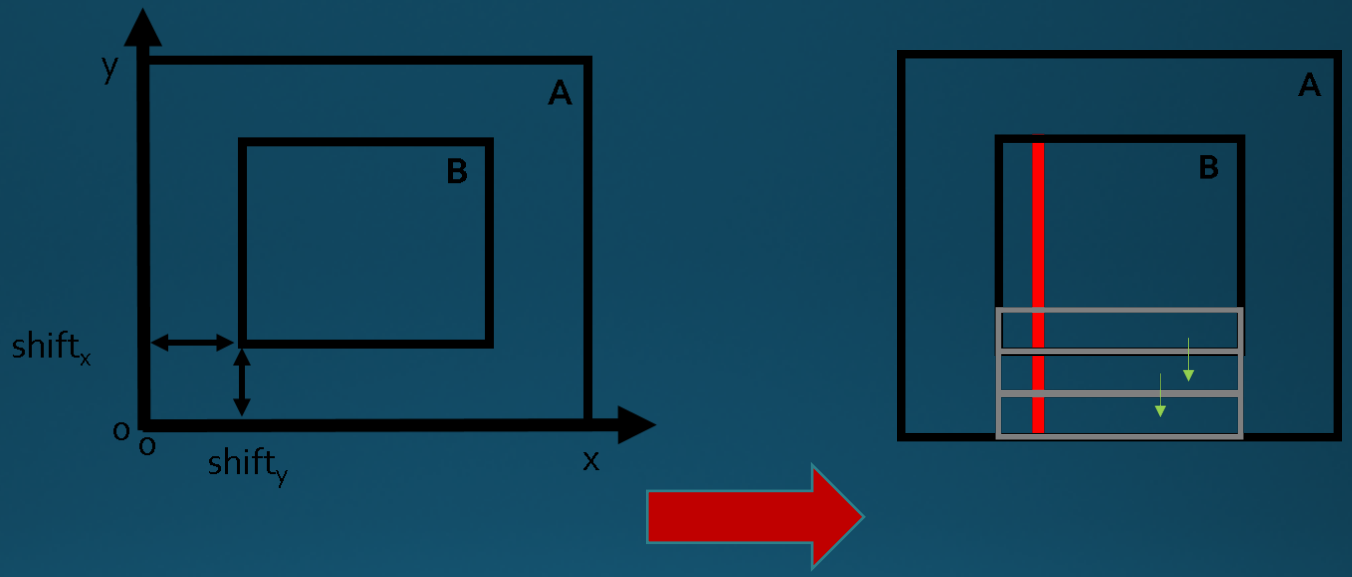
$$\omega_{optimum} \approx \frac{2}{1+\sin(\frac{\pi}{N})}$$

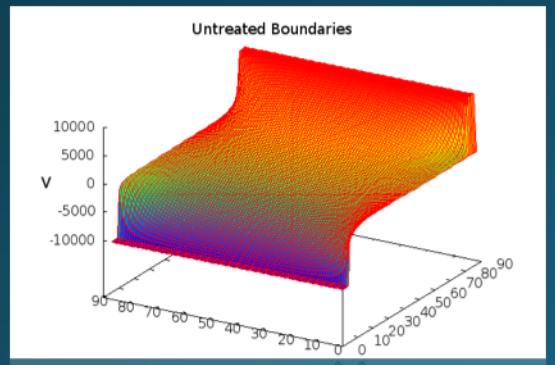


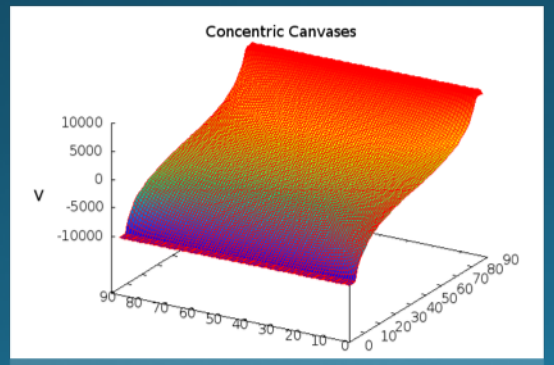
$$V_{i,j}^{n+1} = V_{i,j}^{n} + \frac{\Delta t}{\Delta^2}\omega\left(V_{i+1,j}^{n} + V_{i-1,j}^{n+1} + V_{i,j+1}^{n} + V_{i,j-1}^{n+1} - 4V_{i,j}^{n}\right)$$

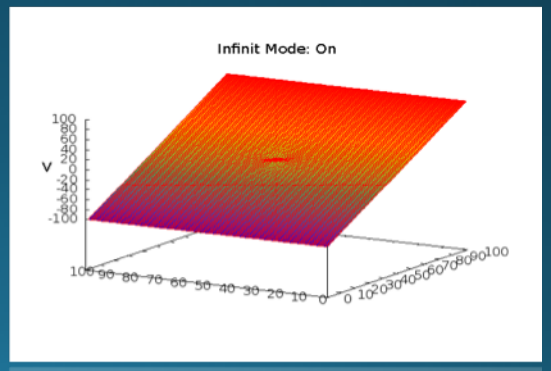
Treating boundaries: Concentric Canvases

No treatment



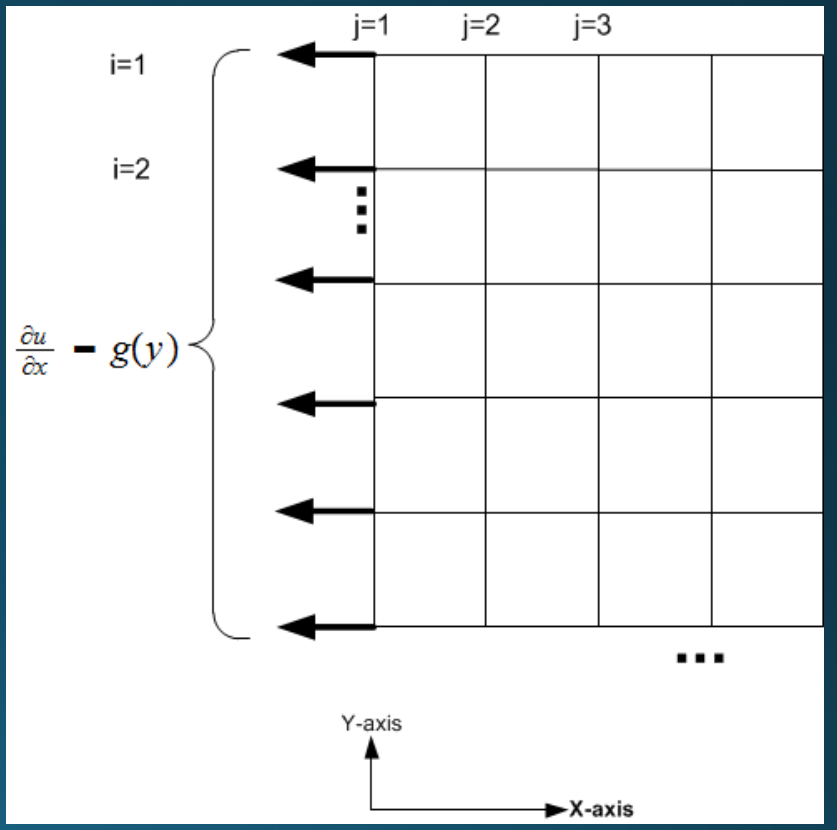






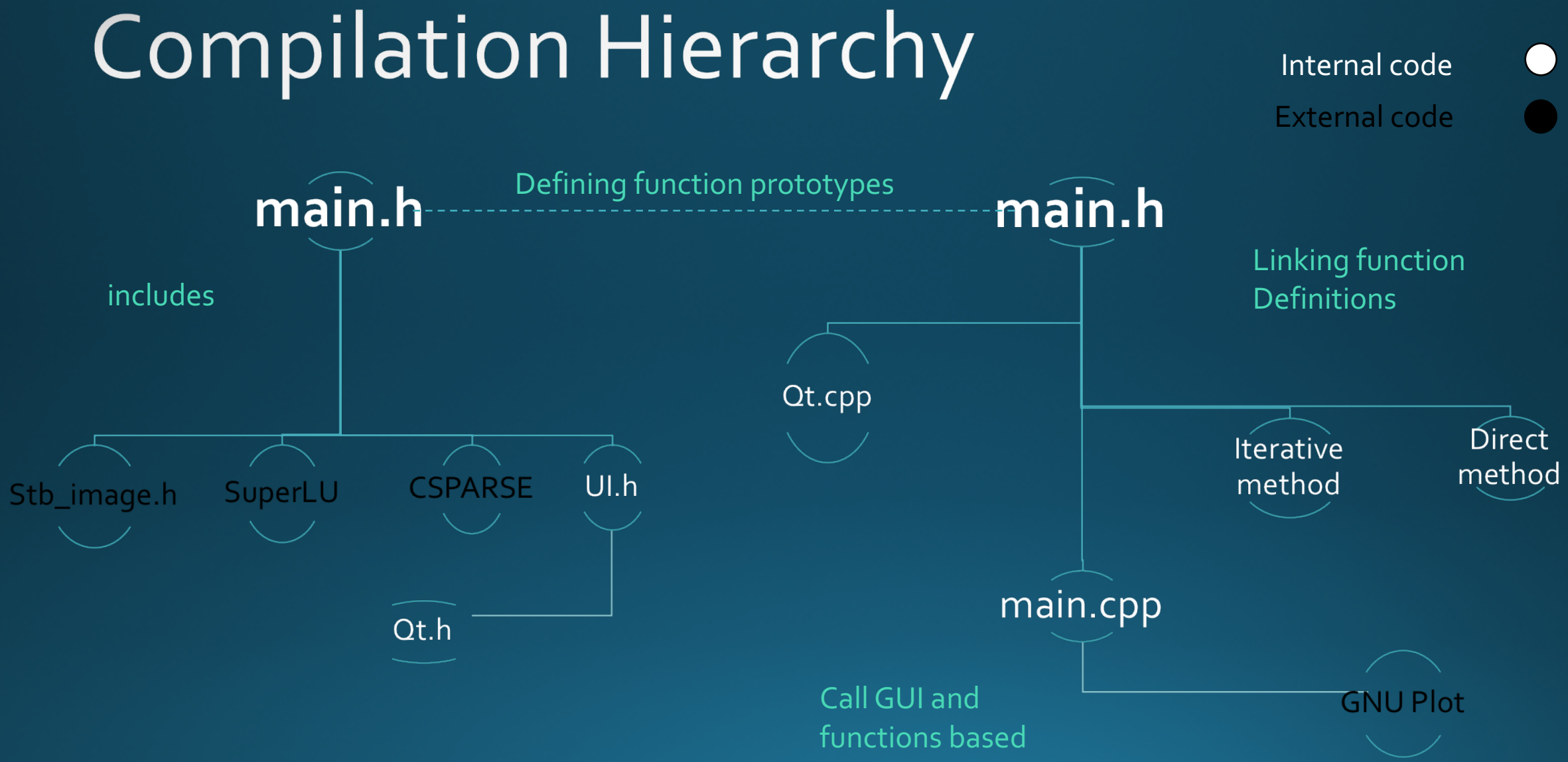
Von Neumann Boundary condition

- Assume slopes are, with good approximation, equal to each other
- $\frac{V_{2,i}-V_{0,i}}{2h} = g(y) = \frac{V_{3,i}-V_{1,i}}{2h}$
- $V_{0,i} = V_{2,i} - 2hg(y)$



https://www.12000.org/my_courses/UC_davis/fall_2010/math_228a/HWs/HW3/Neumman_BC/Neumman_BC.htm

Compilation Hierarchy

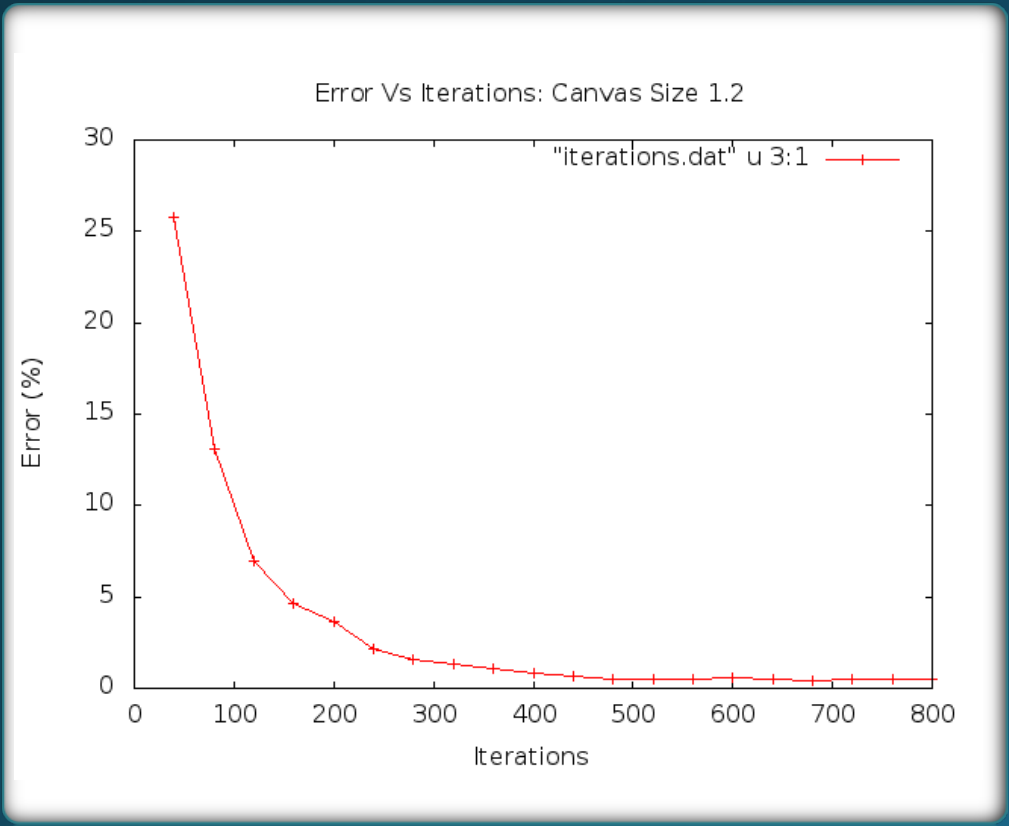


Performance of the Direct Method

Measurement	Problem 0	Problem 1	Nobel Problem
Time	8.684 s	15.130 s	4.857 s
Memory	705.830 MB	1059.962 MB	502.983 MB
Error	0.312%	0.869%	-N/A-

Tests conducted with a 350 x 350 grid.

Controlling Iterative Errors



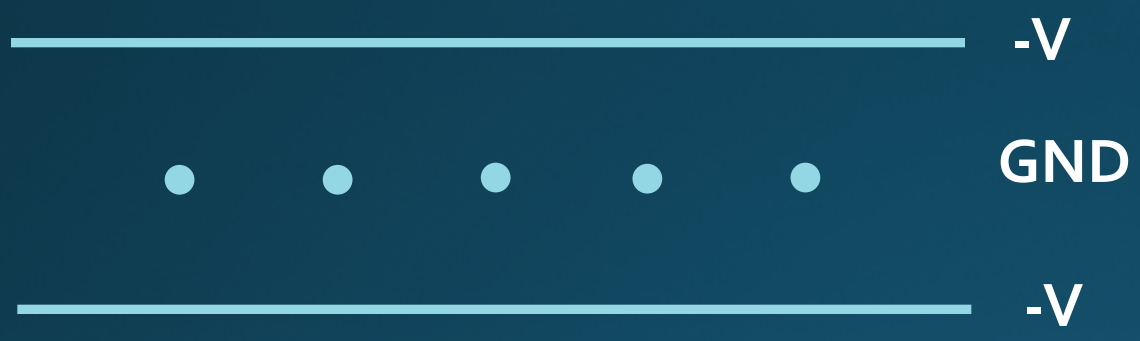
- The error of the iterative method can be reduced by adjusting the tolerance.
- The direct and iterative methods agree at larger iterations.
- We were able to achieve relative errors as low as 0.487% for Problem 0 and 0.740% for Problem 1

Comparison of the Solvers

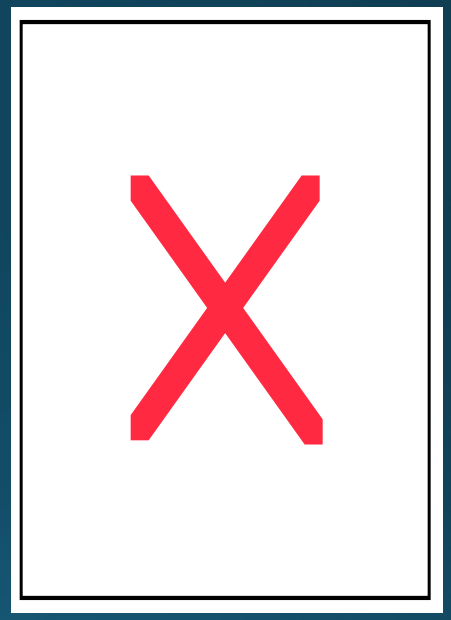
- The direct method was taken as the reference point since no analytical solution was available.
- The iterative method converged to a relative error of 0.075% after 460 iterations (4.48s).

Multiwire Proportional Chamber

Nobel Prize, Georges Charpak (1992)

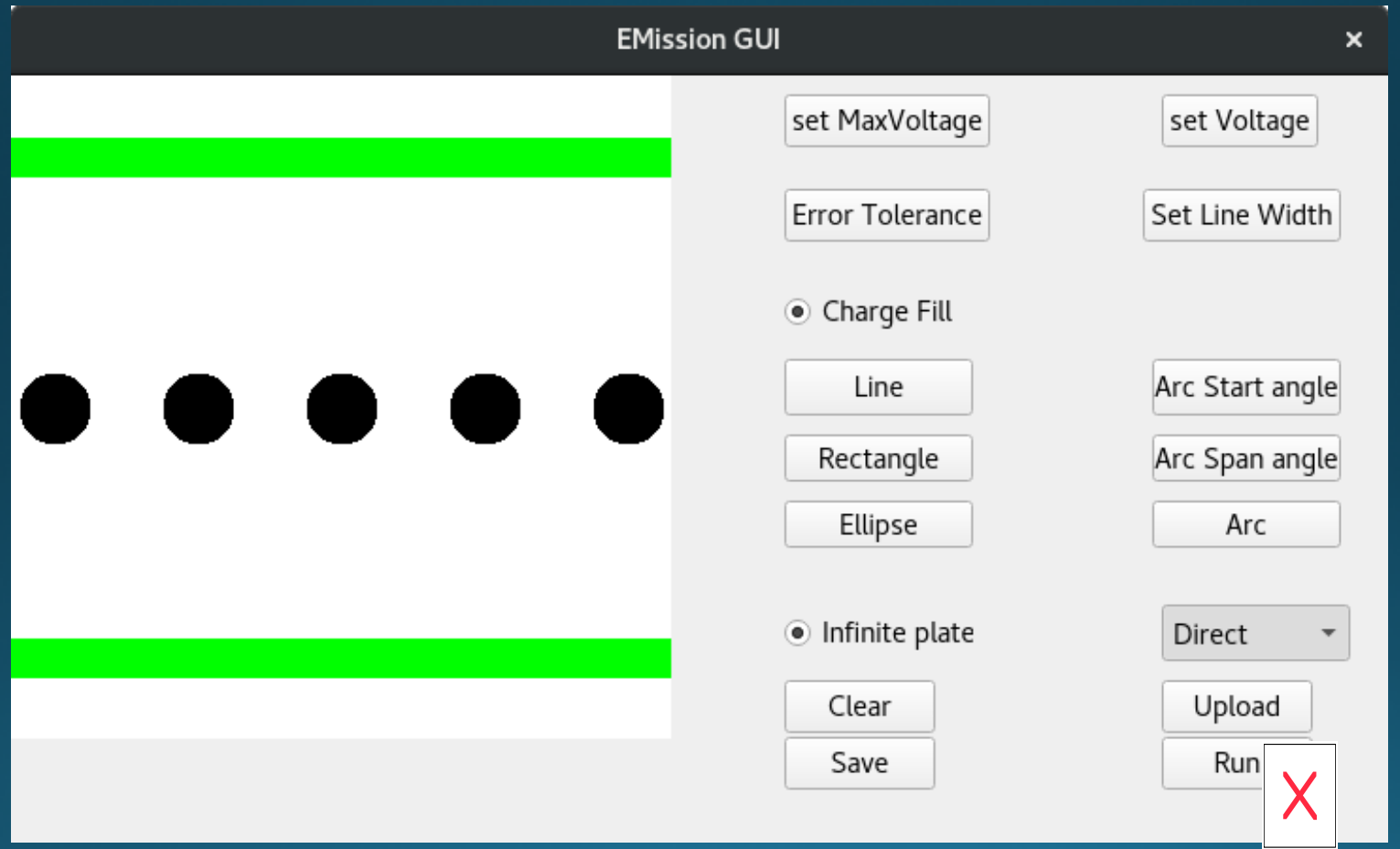


Multi wire proportional chamber

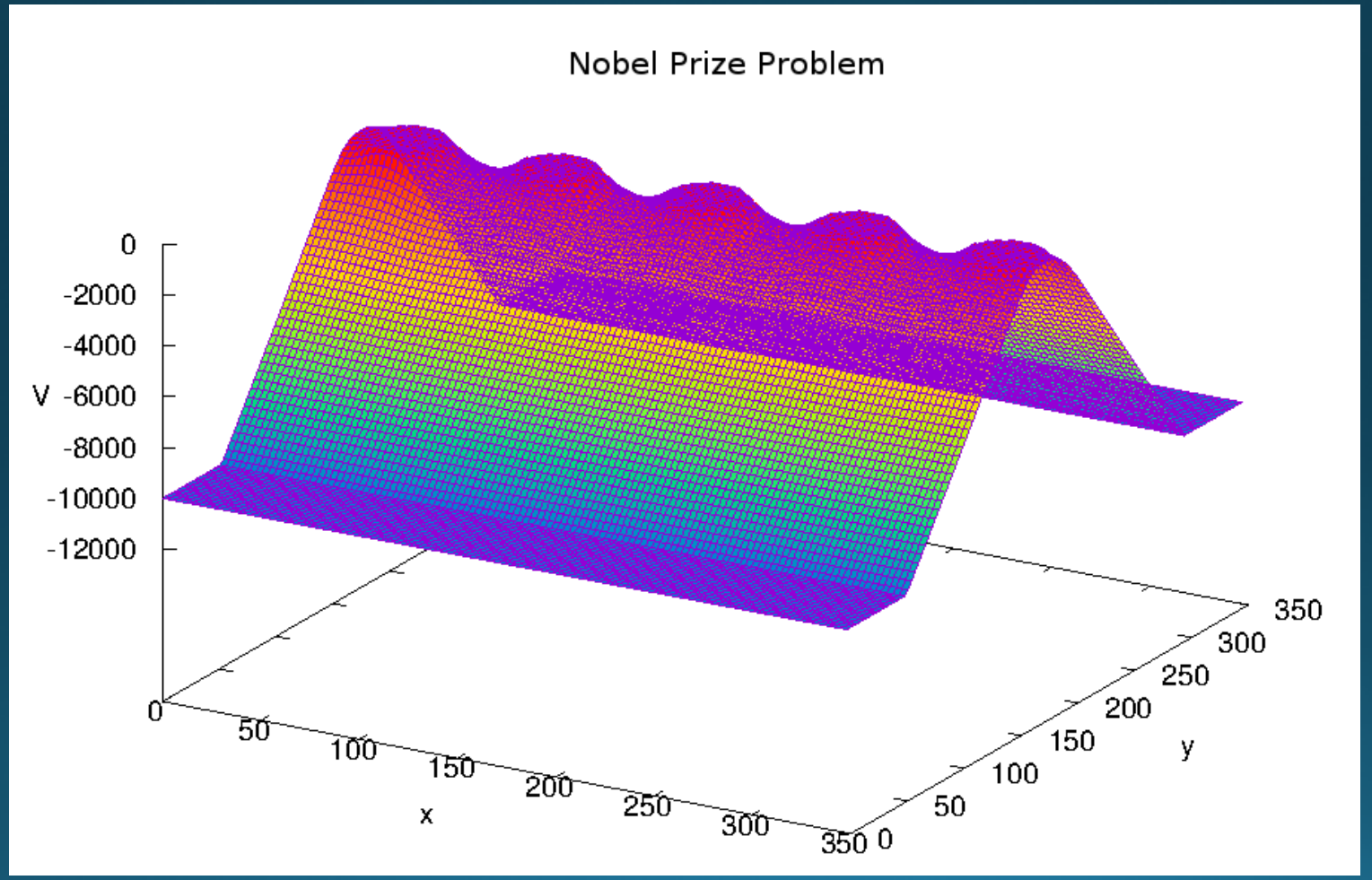


1924-2010

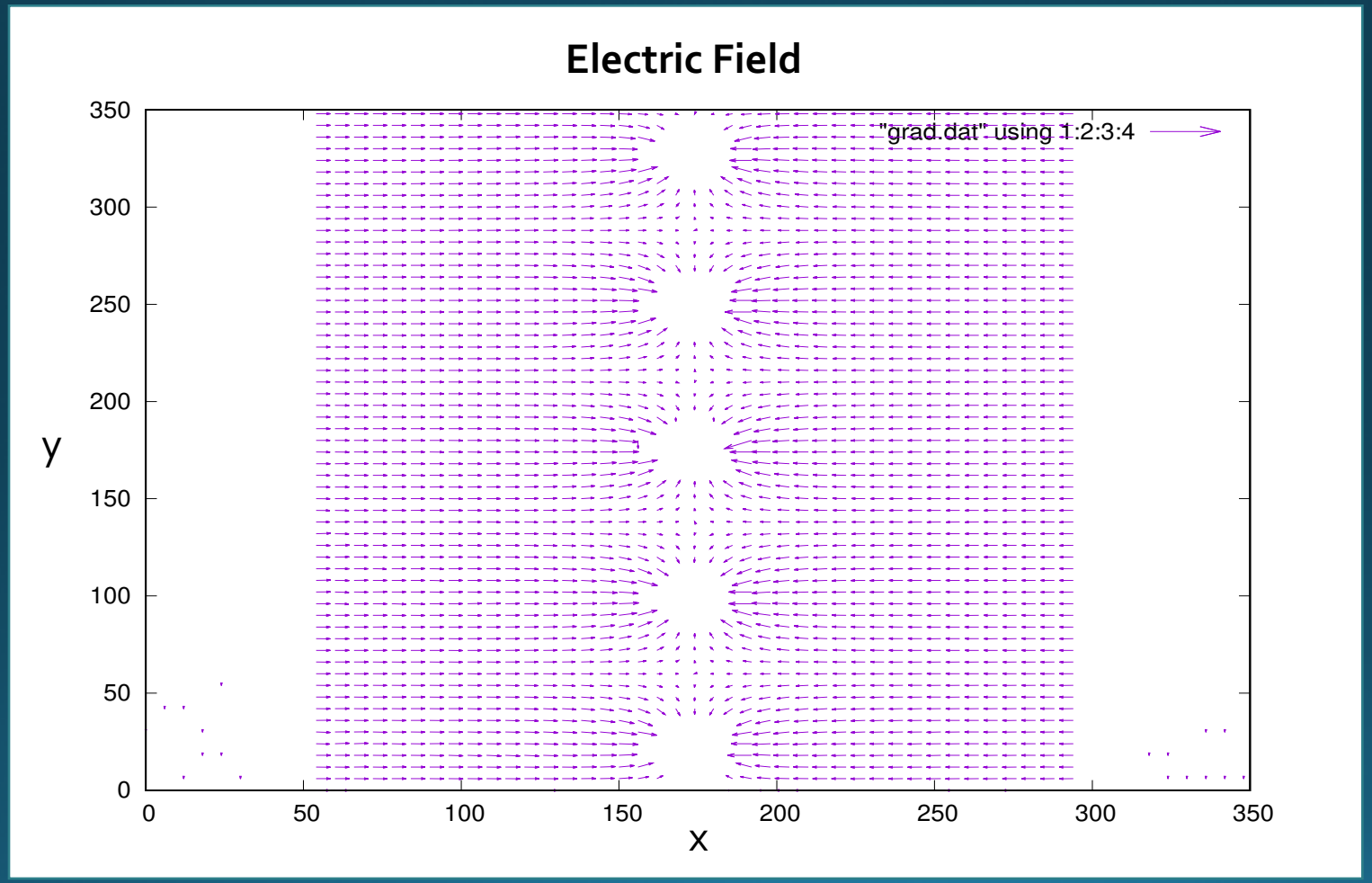
Example Run


EMission GUI
set MaxVoltage
set Voltage
Error Tolerance
Set Line Width
Charge Fill
Line
Arc Start angle
Rectangle
Arc Span angle
Ellipse
Arc
Infinite plate
Direct
Clear
Upload
Save
Run

Results:


Nobel Prize Problem
V
0
-2000
-4000
-6000
-8000
-10000
-12000
0
50
100
150
200
250
300
350
x
0
50
100
150
200
250
300
350
y

Results:


Electric Field
"grad.dat" using 1:2:3:4
y
x
0
50
100
150
200
250
300
350

Summary

- Created a custom GUI to make problem specification general and easy.
- Developed two numerical solvers that accomplish the goals of th projects through the iterative and direct paradigm.
- Coordinated the entire project through a publicly accessible repository.

Possible Improvements

- Support for dielectrics.
- Create a statically linked version to allow for easy distribution of the code.
- Implementation of multigrid methods.

On behalf of the Emission group, thank you for your attention.