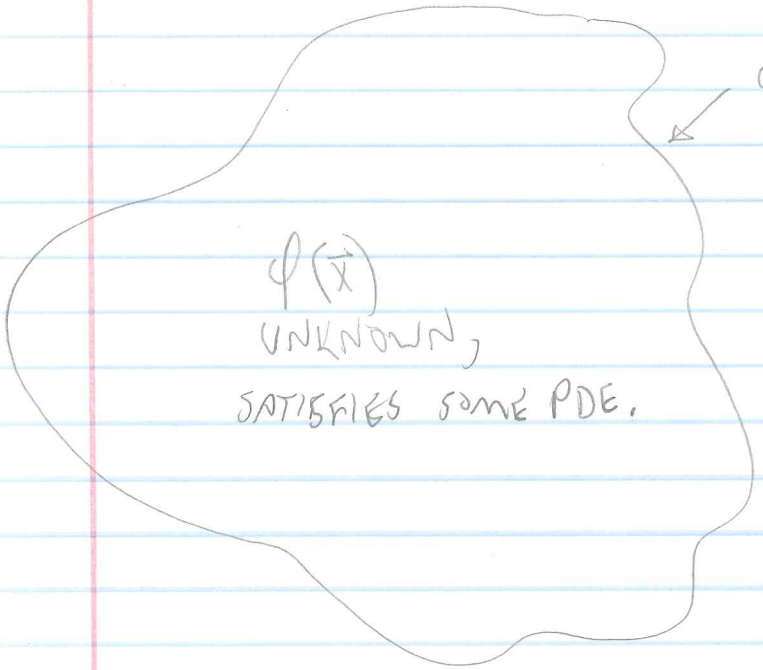
P.D.E.s

BOUNDARY VALUE PROBLEMS



CLASSIC EXAMPLE IS POISSON'S EQUATION:

$$\nabla^2 \varphi = \rho(\vec{x}) \quad , \text{ i.e. } \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\varphi = \rho$$

ELLIPTIC EQN.

THESE PROBLEMS ARE TYPICALLY SOLVED ON A GRID, E.G.

$$X_i = i\Delta \quad , \quad Y_i = i\Delta \quad , \quad Z_i = i\Delta$$

~~$$\frac{\partial \varphi}{\partial x} = \frac{\varphi(x+\delta) - \varphi(x-\delta)}{2\delta}$$~~

~~$$\frac{\partial}{\partial x}\left(\frac{\partial \varphi}{\partial x}\right) = \frac{\varphi(x+2\delta) - 2\varphi(x) + \varphi(x-2\delta)}{(2\delta)^2}$$~~

~~LET $\delta = \Delta/2$~~

$$\left.\frac{\partial^2 \varphi}{\partial x^2}\right|_{X_i} = \frac{\varphi(X_{i+1}) - 2\varphi(X_i) + \varphi(X_{i-1})}{\Delta^2}$$

SKIP?

TAYLOR SERIES:

$$\varphi(x_{i\pm1}) = \varphi(x_i) \pm \Delta \left.\frac{\partial \varphi}{\partial x}\right|_i + \frac{1}{2}\Delta^2 \left.\frac{\partial^2 \varphi}{\partial x^2}\right|_i \pm \frac{1}{6}\Delta^3 \left.\frac{\partial^3 \varphi}{\partial x^3}\right|_i + \frac{1}{24}\Delta^4 \left.\frac{\partial^4 \varphi}{\partial x^4}\right|_i \pm \cdots$$

$$\left.\frac{\partial^2 \varphi}{\partial x^2}\right|_i = \frac{\varphi(x_{i+1}) - 2\varphi(x_i) + \varphi(x_{i-1})}{\Delta^2} + \underbrace{\frac{1}{12}\Delta^2 \frac{\partial^4 \varphi}{\partial x^4} + \cdots}_{\text{TRUNCATION ERROR}}$$

SUPPOSE WE HAVE POISSON'S EQN.,

$$\nabla^2 \varphi(x,y) = \rho(x,y),$$

$$\nabla^2 \varphi_{ij} = \rho_{ij}$$

$$\frac{\left(4\varphi_{ij} - \varphi_{i+1,j} - \varphi_{i-1,j} - \varphi_{i,j+1} - \varphi_{i,j-1}\right)}{\Delta^2} = \rho_{ij}$$

$$\varphi_{ij} = \left(\frac{\varphi_{i+1,j} + \varphi_{i-1,j} + \varphi_{i,j+1} + \varphi_{i,j-1}}{4}\right) + \frac{\Delta^2}{4}\rho_{ij}$$

"ALL" WE NEED TO DO IS MAKE THIS TRUE FOR EVERY i,j INSIDE THE BOUNDARY!

HOW?

FOR AN ODE, E.G.

$$\frac{d^2\varphi}{dx^2} = \rho(x)$$

WE MIGHT SOLVE THE BOUNDARY VALUE PROBLEM BY SHOOTING.

BUT FOR PDEs, SHOOTING IS NOT AN OPTION BECAUSE THERE ARE 2 OR MORE INDEPENDENT VARIABLES (E.G. x, y, z).

WE THEREFORE SOLVE THE BOUNDARY VALUE PROBLEM BY RELAXATION.

E.G. $\nabla^2\psi = 0$, START WITH INITIAL GUESS ψ°;

CONSTRUCT AN INITIAL VALUE PROBLEM:

$$\frac{\partial\psi}{\partial t} = K\nabla^2\psi$$

TAKE LIMIT $t \to \infty$

$$\frac{\partial\psi}{\partial t} \to 0$$

$$K\nabla^2\psi \to 0$$

FTCS:

$$\psi^{n+1} = \psi^n + \delta K \nabla^2 \psi^n$$

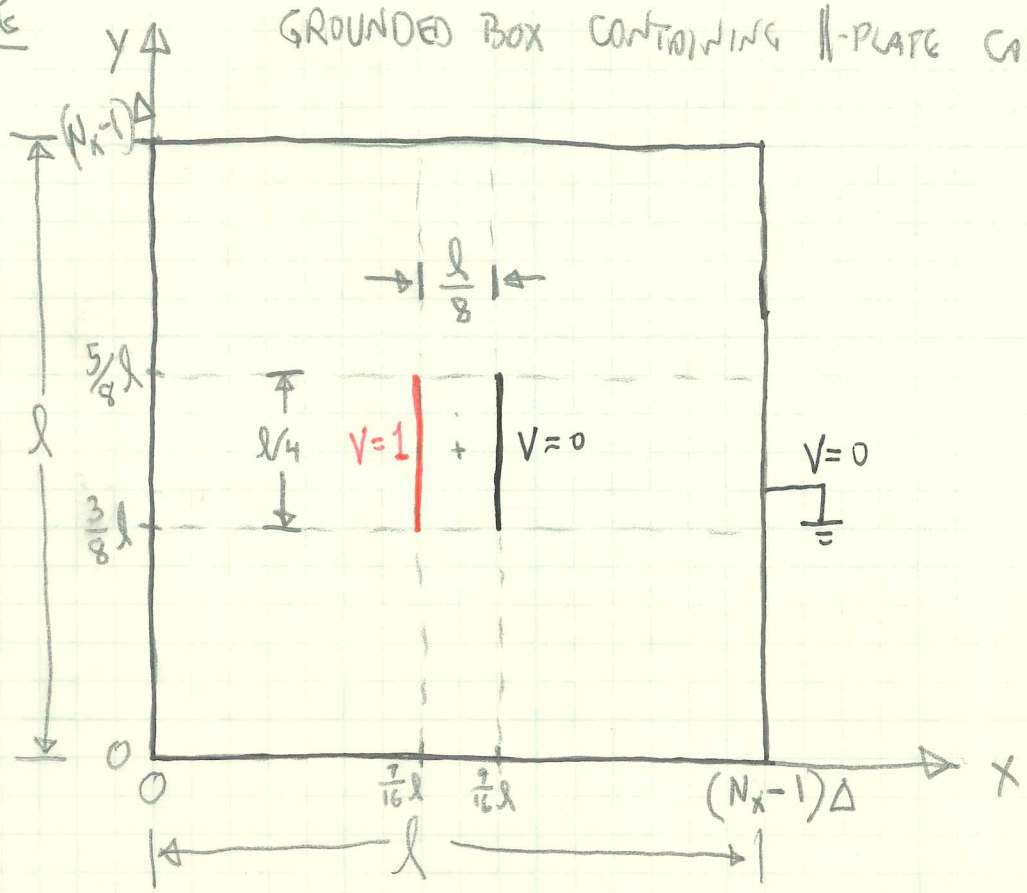
$$\psi_{ij}^{n+1} = \psi_{ij}^n + \frac{\delta K}{\Delta^2}\left(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - 4\psi_{ij}\right)$$

E.G. CHOOSE $\frac{\delta K}{\Delta^2} = \frac{1}{4}$

"POINT RELAXATION"

EXAMPLE

GROUNDED BOX CONTAINING ||-PLATE CAP.



GRID SPACING Δ

$\ell = (N_x - 1)\Delta$

CAPACITOR TOP $= \frac{5}{8}\ell = \frac{5}{8}(N_x-1)\Delta$

CAPACITOR BOTTOM $= \frac{3}{8}\ell = \frac{3}{8}(N_x-1)\Delta$

" LEFT $= \frac{7}{16}\ell = \frac{7}{16}(N_x-1)\Delta$

RIGHT $= \frac{9}{16}\ell = \frac{9}{16}(N_x-1)\Delta$

IN OUR EXAMPLE, I LET $N_x = 65$
(SO $N_x - 1$ DIVISIBLE BY 16).

SOLUTION STRATEGY:

- ELLIPTICAL BV PROB ⟶ PARABOLIC INIT VAL PROBLEM

- TIMESTEP: LOBOTOMIZED VERSION OF ALTERNATING DIRECTION IMPLICIT.

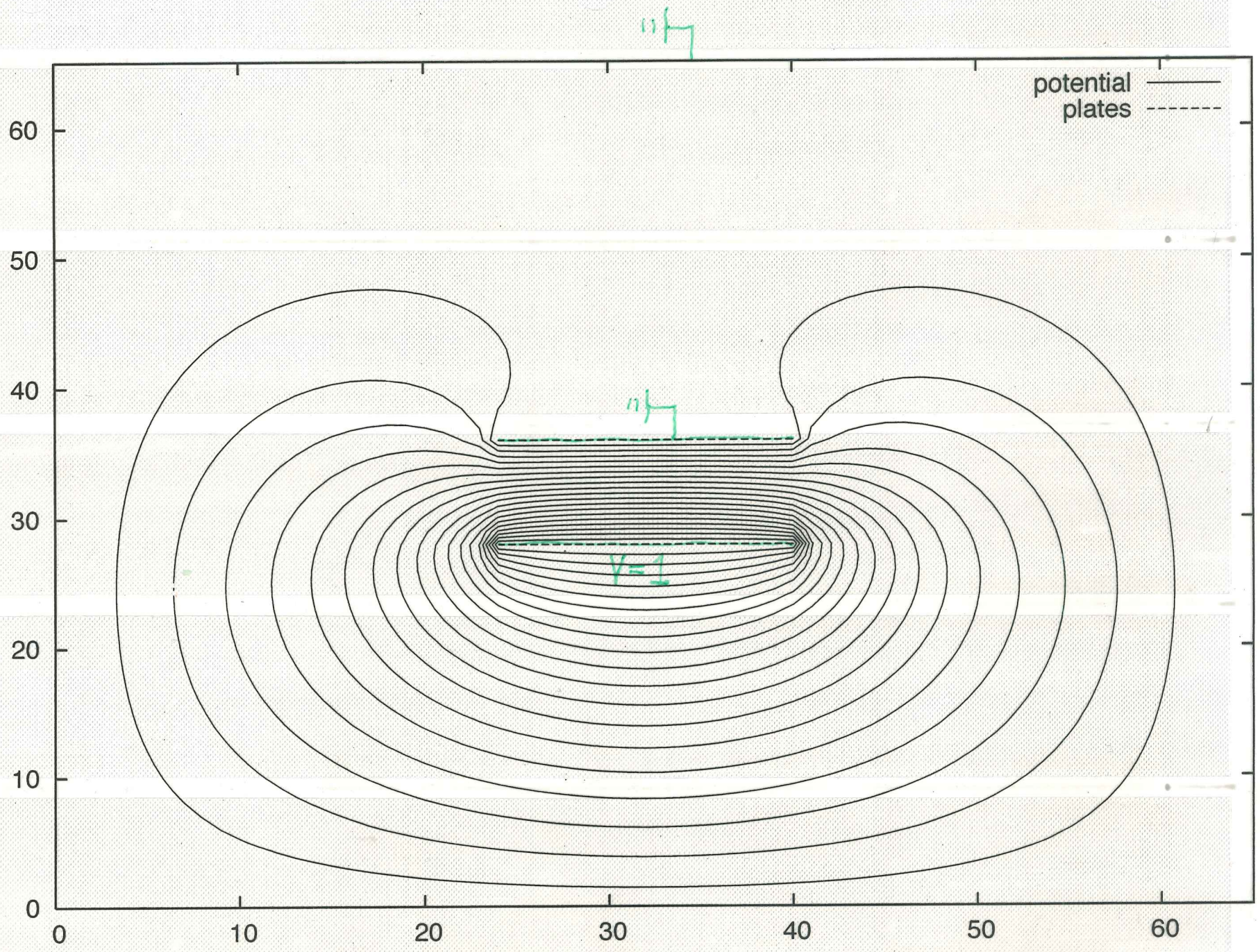
- RAMP TIMESTEP LARGE ⟶ SMALL

```
#include <stdio.h>
#include <math.h>

#include "/usr/local/src/recipes_c-ansi/include/nr.h"
#include "/usr/local/src/recipes_c-ansi/include/nrutil.h"
//#include "/usr/local/src/recipes_c-ansi/recipes/tridag.c"

void implicit(float t, float q[], short f[], long nq, float delta_t);
void A_1(float t, float delta, float q[], short f[], float a[], float b[], float
 c[]);
void bc(float psi[][], short flag[][]);

#define KAPPA 1.0
#define NX 65  //NX-1 should be divisible by 16.
#define DX 1.0
#define NT 2000 //number of relaxation time steps
#define DT 1000.0 //this is the initial time interval.

int main(void)
{
   float psi[NX][NX], q[NX], delta_t;
   short flag[NX][NX], f[NX];
   long i, j, k;
   FILE *outfile;

   // Initialize/set boundary conditions on psi[][] and
   // mark boundary locations in flag[][]
   bc(psi, flag);

   // Relax NT "time" steps
   for (k=1; k<=NT; k++) {

      //Time step delta_t ramps down from DT to 1.0;
      delta_t = DT-(float)k;
      if (delta_t < 1.0) delta_t=1.0;

      // do the rows
      for (i=1; i<NX-1; i++) {
         implicit(0.0, psi[i], flag[i], NX, delta_t);
      }

      // do the columns
      for (i=1; i<NX-1; i++) {
         for (j=1; j<NX-1; j++) {
            q[j] = psi[j][i];
            f[j] = flag[j][i];
         }
         implicit(0.0, q, f, NX, delta_t);
         for (j=1; j<NX-1; j++) {
            psi[j][i] = q[j];
         }
      }
   }

   // Save results
   outfile=fopen("data.txt","w");
   for(i=0;i<NX;i++) {
      for(j=0;j<NX;j++) fprintf(outfile,"%f ",psi[i][j]);
      fprintf(outfile,"\n");
   }
}


void implicit(float t, float q[], short flag[], long nq, float delta_t)
{
```

```
   float a[NX], b[NX], c[NX], q1[NX];
   long i;

   A_1(t, delta_t, q, flag, a, b, c); //get the tridiagonal propagator matrix
   tridag(a-1, b-1, c-1, q-1, q1-1, NX);
   for(i=0; i<NX; i++) q[i] = q1[i];
}

// Propagator for the parabolic initial value problem, FTCS scheme.
// This is used to implement the implicit time step.
void A_1(float t, float delta, float q[], short flag[], float a[], float b[], fl
oat c[])
{
   long i;
   float foo;

   foo = delta*KAPPA/(DX*DX);

   for (i=0; i<NX; i++) {
      if (flag[i] == 1) { // this cell is a fixed boundary value
         a[i] = 0.0; //lower diagonal
         b[i] = 1.0; //main diagonal
         c[i] = 0.0; //upper diagonal
      } else {
         a[i]= -foo;          //lower diagonal
         b[i]= (1.+ 2.*foo); //main diagonal
         c[i]= -foo;          //upper diagonal
      }
   }

}

void bc(float psi[NX][NX], short flag[NX][NX])
{
   long i,j,top,bottom,left,right;
   FILE *dfile;

   //first, zero the interior
   for (i=0; i<NX; i++) {
      for (j=0; j<NX; j++) {
         psi[i][j]=0.0; //zero computational domain interior
         flag[i][j]=0;  //clear boundary flag
      }
   }

   // NOW IMPLEMENT VALUES FOR DIRICHLET (FIXED) BOUNDARIES
   //edges
   for (i=0; i<NX; i++) {
      psi[0][i]=0.0;
      psi[i][0]=0.0;
      psi[NX-1][i]=0.0;
      psi[i][NX-1]=0.0;

      flag[0][i]=1;
      flag[i][0]=1;
      flag[NX-1][i]=1;
      flag[i][NX-1]=1;

   }

   //capacitor plates
   top    = (NX-1)*5/8;
   bottom = (NX-1)*3/8;
   left   = (NX-1)*7/16;
   right  = (NX-1)*9/16;
```

BC

```
    //Save a little data file with plate dimensions:
    dfile=fopen("plates.txt","w");
    fprintf(dfile,"%d %d\n",top,left);
    fprintf(dfile,"%d %d\n",bottom,left);
    fprintf(dfile,"\n");
    fprintf(dfile,"%d %d\n",top,right);
    fprintf(dfile,"%d %d\n",bottom,right);
    fprintf(dfile,"\n");
    fclose(dfile);

    for (i=bottom; i<=top; i++) {
        psi[left ][i] = 1.0;
        psi[right][i] = 0.0;

        flag[left ][i] = 1;
        flag[right][i] = 1;
    }
}
```


potential
plates
V=1
0
10
20
30
40
50
60