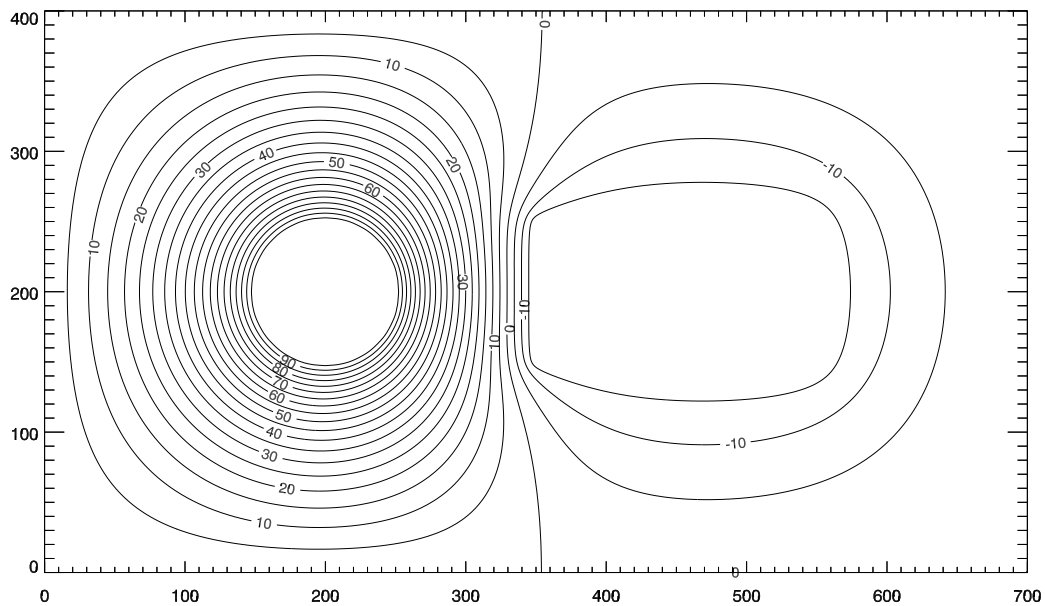


EE 434 Electricity and Magnetism, Spring 2009

Homework #3 Solution

I wrote two programs to solve this program. The first program, `sim1` generates the pattern to be processed, and the second program, `relax` performs the relaxation calculation. I then load that array into a third program, written in IDL, which creates the plot. Here is my final contour plot, and on the following pages are my program listings.



```

/*=====
Simulate a circle and a rectangle inside a grounded box.
=====*/

#include <stdlib.h>
#include <stdio.h>
#include <amjMemory.h>
#include <zlib.h>

int nX=701,nY=401;
char oFile[]="../dat/sim1.dat";
char **m;
double **phi;

int main(int argc, char *argv[]){

    int i,j;
    double r;
    gzFile *fp;

    m=amjMalloc2dChar(nX,nY,"main:m");
    phi=amjMalloc2dDouble(nX,nY,"main:phi");

    for(i=0;i<nX;i++)
        for(j=0;j<nY;j++){
            m[i][j]=0;
            phi[i][j]=0;
            r=sqrt((i-200)*(i-200)+(j-200)*(j-200));
            if(r<=50){
                m[i][j]=1;
                phi[i][j]=100;
            }
            if(i>=350&& i<=550&& j>=150&& j<=250){
                m[i][j]=1;
                phi[i][j]=-20;
            }
            if(i==0||i==nX-1||j==0||j==nY-1){
                m[i][j]=1;
                phi[i][j]=0;
            }
        }

    fp=gzopen(oFile,"w9");

    gzwrite(fp,&nX,sizeof(int));

```

```
gzwrite(fp,&nY,sizeof(int));

gzwrite(fp,m[0],sizeof(char)*nX*nY);
gzwrite(fp,phi[0],sizeof(double)*nX*nY);

gzclose(fp);

return 0;
}
```

```

/*=====
relax - performs relaxation on a potential function with given
boundary conditions.

calling sequence:

relax [-it <integer>] [-tol <double>] <input filename> [<output filename>]

-it Maximum number of iterations. Default is 1000.
-tol Maximum tolerance. Default is 10^-6.

<input filename> Initial guess.
<output filename> where to write the result. If not specified. It is
generated from the input filename.
=====*/

#include <stdlib.h>
#include <stdio.h>
#include <libgen.h>
#include <string.h>
#include <zlib.h>
#include <amjMemory.h>

void parseArgs(int argc, char *argv[]);
void loadFile();
void saveFile();
void relax();

char *programname;
int maxIt=1000;
double maxTol=1e-6;
char *iFile=NULL;
char *oFile=NULL;

int nX,nY;
char **m; /* [nX] [nY] */
double **phi; /* [nX] [nY] */

int main(int argc, char *argv[]){
    programname=strdup(basename(argv[0]));
    parseArgs(argc,argv);

    loadFile();

```

```

    relax();

    saveFile();

    exit(0);
}

void parseArgs(int argc, char *argv[]){
    int error=0,i;

    for(i=1;i<argc;i++)
        if(strcmp(argv[i],"-h")==0){
            printf("  relax [-it <integer>] [-tol <double>] <input filename> "
                "[<output filename>]\n");
            printf("\n");
            printf("  -it Maximum number of iterations. Default is 1000.\n");
            printf("  -tol Maximum tolerance. Default is 10-6.\n");
            printf("\n");
            printf("  <input filename> Initial guess.  \n");
            printf("  <output filename> where to write the result. "
                "If not specified. It is\n");
            printf("  generated from the input filename.\n");
            exit(0);
        }

    for(i=1;i<argc;i++){
        if(strcmp(argv[i],"-it")==0){
            i++;
            maxIt=atoi(argv[i]);
        }
        else if(strcmp(argv[i],"-tol")==0){
            i++;
            maxTol=atof(argv[i]);
        }
        else if(argv[i][0]=='-'){
            fprintf(stderr,"%s: error: unknown option: %s.\n",programname,argv[i]);
            exit(1);
        }
        else if(iFile==NULL){
            iFile=argv[i];
        }
        else if(oFile==NULL){
            oFile=argv[i];
        }
    }
}

```

```

    else{
        fprintf(stderr,"%s: error: too many parameters: %s. Exiting\n",
                programname,argv[i]);
        exit(1);
    }
}
}

```

```

void loadFile(){
    gzFile *fp;

    fp=gzopen(iFile,"r");

    gzread(fp,&nX,sizeof(int));
    gzread(fp,&nY,sizeof(int));

    m=amjMalloc2dChar(nX,nY,"loadFile:m");
    phi=amjMalloc2dDouble(nX,nY,"loadFile:phi");

    gzread(fp,m[0],sizeof(char)*nX*nY);
    gzread(fp,phi[0],sizeof(double)*nX*nY);

    gzclose(fp);
}

```

```

void saveFile(){
    gzFile *fp;

    fp=gzopen(oFile,"w9");

    gzwrite(fp,&nX,sizeof(int));
    gzwrite(fp,&nY,sizeof(int));
    gzwrite(fp,m[0],sizeof(char)*nX*nY);
    gzwrite(fp,phi[0],sizeof(double)*nX*nY);

    gzclose(fp);
}

```

```

void relax(){

    int i,j,k;
    double max,tmp;

```

```

for(k=1;k<=maxIt;k++){
    max=0;
    for(i=0;i<nX;i++)
        for(j=0;j<nY;j++)
            if(m[i][j]==0){
                tmp=phi[i][j];
                phi[i][j]=0.25*(phi[i-1][j]+phi[i+1][j]+phi[i][j-1]+phi[i][j+1]);
                tmp=fabs(tmp-phi[i][j]);
                if(tmp>max)
                    max=tmp;
            }
    printf("%d %lf\n",k,max);

    if(max<maxTol)
        break;
}
}

```

```

function loadPot,filename

;+
; NAME:
;   loadPot
; PURPOSE:
;   Load a potential file.
; CALLING SEQUENCE:
;   d=loadPot(filename)
; INPUTS:
;   FILENAME=STRING. Name of file to load.
; OUTPUTS:
;   D={nX=LONG,
;     nY=LONG,
;     m=bytarr(nY,nX),
;     phi=dblarr(nY,nX)
;   }
; MODIFICATION HISTORY:
;   $Log$
;-

  openr,un,filename,/get_lun,/compress

  nX=01 & nY=01
  readu,un,nX,nY
  m=bytarr(nY,nX)
  phi=dblarr(nY,nX)
  readu,un,m,phi
  free_lun,un

  return,{nX:nX,nY:nY,m:m,phi:phi}
end

```

```
@loadPot.pro

d=loadPot("../dat/relax1.dat")

set_plot,'ps'
device,filename='../afigures/relax1.eps',xsize=8,ysize=5,/inches
!p.font=1

pos=[1./16,.1,15./16,.9]
plot,[0],/nodata,pos=pos,xrange=[0,700],yrange=[0,400],$
    /xstyle,/ystyle
labels=lonarr(23)
for i=1,22,2 do labels[i]=1
contour,transpose(d.phi),pos=pos,/noerase,nlevels=23,c_labels=labels,$
    xrange=[0,700],yrange=[0,400],/xstyle,/ystyle

device,/close

end
```