

Update on Lab 7

R. Robinson 4/4/01

There are a couple of problems in the way I describe the discretization in this lab. Here are some hints about what now seems to me to be the correct approach:

We want to divide the square $[-1, 1] \times [-1, 1]$ up into a grid of discrete points. Suppose we agree to divide into $n + 1$ points in the vertical and horizontal directions and make $\Delta x = \Delta y$ the distance between the grid points. Since between $n + 1$ points there are n intervals, we have $\Delta x = \Delta y = (b - a)/n = 2/n$.

We can then take $x_1 = -1$, and $x_j = -1 + (j - 1)\Delta x = -1 + \frac{2(j-1)}{n}$, for $j = 2, \dots, n + 1$, so that $x_{n+1} = 1$. Similarly we let $y_1 = 1$, and $y_i = 1 - (i - 1)\Delta y = 1 - \frac{2(i-1)}{n}$, for $i = 2, \dots, n + 1$, so that $y_{n+1} = -1$. (The difference between x and y corresponds to the difference between how we index matrices and squares in the plane!)

Then if we denote the grid point by $v_{i,j} = (x_j, y_i)$, we have $v_{i,j} = (x_j, y_i) = (-1 + \frac{2(j-1)}{n}, 1 - \frac{2(i-1)}{n})$ where $i = 1, \dots, n - 1, j = 1, \dots, n - 1$.

Recall that the goal of the lab is to construct a "temperature" function u on $[-1, 1] \times [-1, 1]$. We want to approximate it by a function on the grid; that is an assignment of a number (temperature) to each grid point. Given any function u , its values on the grid can be represented by a matrix. Let's call this matrix $U = (u_{i,j})$. We define it as follows:

$$u_{i,j} = u(v_{i,j}) = u(-1 + \frac{2(j-1)}{n}, 1 - \frac{2(i-1)}{n}).$$

This allows us to work exclusively with matrices once we have converted the initial data to this format.

Note that the top, bottom, left and right of the plate $[-1, 1] \times [-1, 1]$ correspond to the top row, bottom row, left column and right column of the matrix.

In the initialization step we are supposed to set the matrix to zero, then put given boundary values in for these boundary rows and columns. this is what I will describe now. After that, there is the iteration step and the graphing step, both of which I will mention below.

Thus, in the initialization, we should put (in MATLAB in the case $n=6$)

```
> U=zeros(7,7)
```

Then we want to make

$$u_{1,j} = f_i(-1 + \frac{2(j-1)}{n}), j = 1, \dots, n - 1,$$
$$u_{n+1,j} = f_b(-1 + \frac{2(j-1)}{n}), j = 1, \dots, n - 1,$$

$$u_{i,1} = g_l(1 - \frac{2(j-1)}{n}), i = 1, \dots, n-1$$

and

$$u_{i,n+1} = g_r(1 - \frac{2(j-1)}{n}), i = 1, \dots, n-1,$$

where

$$f_t(x) = x^5 - 10x^3 + 5x$$

$$f_b(x) = x^5 - 10x^3 + 5x$$

$$g_l(y) = -1 + 10y^2 - 5y^4$$

$$g_r(y) = 1 - 10y^2 + 5y^4$$

As an example of how to do this in MATLAB, here is how I might fill in the top row:

```
>dx=2/n
> x=-1:dx:1
> U(1,:)=ft(x)
```

Of course, I would need an m-file with the function "ft" defined in it. I would then do the same for the other three sides, so for that I would also need m-files fb.m, gl.m and gr.m. Note that the two columns would need to column vectors, so a transpose of the y-vector is required.

In practice I would probably put this all in an m-file called "initialize.m" which would give me a matrix U of size $(n+1)^2$ by typing

```
> U=initialize(n).
```

I would also have an m-file called "relax.m" that would perform one round of relaxation. I would use it like this.

```
>U1=relax(U)
```

Finally I would have a while loop that compares U and U1 (entry by entry). If they are not close put

U(:,:)=U1(:,:) and U1=relax(U) again. Stop if they are close and output U.