TFY4235/FY8904: Computational Physics, spring 2013

Solution set 3

Problem 1.

The problem here is to translate the physical problem — finding the potentials in a chain of 1000 random resistors in series — into a matrix problem on the form $Ax = b$. What is A and b? v is evidently a vector that contains all potentials. We enumerate the nodes of the chain from 0 to 1000. The restors are numbered 0 to 999. Node 0 has a potential of $v_0 = 0$ and node 1000 has a potential of $v_{1000} = 1$. The other potentials, v_i, where $i = 1, 999$, are unknown. The Kirchhoff equations give for each of the nodes $i = 1, 999$

$$-c_{i-1}v_{i-1} + (c_{i-1} + c_i)v_i - c_i v_{i+1} = 0 \; . \tag{1}$$

Please note in this expression that we are using *conductance* c rather than *resistance* r as was done in the lectures. The relation between the two quantities is $c = 1/r$.[1] The equations for $i = 1$ og $i = 999$ may be rewritten

$$(c_0 + c_1)v_1 - c_1 v_2 = c_0 v_0 \tag{2}$$

and

$$-c_{998}v_{998} + (c_{998} + c_{999})v_{999} = c_{999}v_{1000} \; , \tag{3}$$

where $v_0 = 0$ og $v_{1000} = 1$. From Eqs. (1), (2) and (3) we see that the matrix A is

$$A = \begin{pmatrix} (c_0 + c_1) & -c_1 & 0 & 0 & \dots & 0 & 0 \\ -c_1 & (c_1 + c_2) & -c_2 & 0 & \dots & 0 & 0 \\ 0 & -c_2 & (c_2 + c_3) & -c_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & (c_{997} + c_{998}) & -c_{998} \\ 0 & 0 & 0 & 0 & \dots & -c_{998} & (c_{998} + c_{999}) \end{pmatrix} \tag{4}$$

and the vector b is

$$b = \begin{pmatrix} c_0 v_0 \\ 0 \\ \vdots \\ 0 \\ c_{999}v_{1000} \end{pmatrix} \; . \tag{6}$$

[1] Why do I do it differently here from the lectures? This is a nice illustration in my mind of the fact that formulating a problem is not a unique process. Sometimes, a reformulation may simplify the problem.

The vector v is

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_{998} \\ v_{999} \end{pmatrix} . \tag{7}$$

The Jacobi iteration is

$$Dv^{n+1} = b - (L + U)v^n \tag{8}$$

where D is the diagonal of A. If we substitute for A and b, we get the matrix equation

$$\begin{pmatrix} (c_0 + c_1)v_1^{n+1} \\ (c_1 + c_2)v_2^{n+1} \\ \vdots \\ (c_{998} + c_{999})v_{999}^{n+1} \end{pmatrix} = \begin{pmatrix} c_0 v_0^n + c_1 v_2^n \\ c_1 v_1^n + c_2 v_3^n \\ \vdots \\ c_{998} v_{998}^n + c_{999} v_{1000}^n \end{pmatrix} . \tag{9}$$

Here is a Fortran program that sets up the resistor chain and then finds the potentials by using Eq. (9):

```
program jacobi
c Jacobi relaxation
dimension volt(0:1000),vnew(999),cond(0:999)
dimension resi(0:999)
c Initializing random number generator
rinv=0.5/2147483647.
rold=0.
ibm=4711
do i=1,1000
ibm=ibm*16807
enddo
c Generating the 1000 conductances
do i=0,999
ibm=ibm*16807
cond(i)=ibm*rinv+0.5
enddo
do i=1,999
resi(i)=1./(cond(i-1)+cond(i))
enddo
c Stopping criterion
pres=2.e-4
c Initializing voltages
do i=0,1000
volt(i)=float(i)/1000
enddo
```

```
c Iteration
do ite=1,1000000
do i=1,999
vnew(i)=(cond(i-1)*volt(i-1)+cond(i)*volt(i+1))*resi(i)
enddo
eps=0.
do i=1,999
volt(i)=vnew(i)
enddo
do i=1,999
eps=eps+(cond(i-1)*volt(i-1)+cond(i)*volt(i+1)
-(cond(i-1)+cond(i))*volt(i))**2
enddo
eps=sqrt(eps)
write(*,*) ite,eps
if(eps.lt.pres) goto 200
enddo
c Done
200 continue
end
```

The Gauss-Seidel iteration,

$$(L + D)v^{n+1} = b - Uv^n \ , \tag{10}$$

is very easy to implement. Rewrite (10) to

$$Dv^{n+1} = b - Uv^n - Lv^{n+1} \tag{11}$$

and compare with the Jacobi relaxation (8). Since we read the indices from $i = 1$ to $i = 999$, we implement the Gauss-Seidel iteration in the previous program by simply removing the work vector `vnew(i)`. Here is the Gauss-Seidel program:

```
program gauss
c Gauss-Seidel relaxation
dimension volt(0:1000),cond(0:999)
dimension resi(0:999)
c Initializing the random number generator
rinv=0.5/2147483647.
rold=0.
ibm=4711
do i=1,1000
ibm=ibm*16807
enddo
c Generating the 1000 conductances
do i=0,999
ibm=ibm*16807
```

```
cond(i)=ibm*rinv+0.5
enddo
do i=1,999
resi(i)=1./(cond(i-1)+cond(i))
enddo
c Stopping criterion
pres=2.e-4
c Initializing voltages
do i=0,1000
volt(i)=float(i)/1000
enddo
c Iteration
do ite=1,1000000
do i=1,999
volt(i)=(cond(i-1)*volt(i-1)+cond(i)*volt(i+1))*resi(i)
enddo
eps=0.
do i=1,999
eps=eps+(cond(i-1)*volt(i-1)+cond(i)*volt(i+1)
x -(cond(i-1)+cond(i))*volt(i))**2
enddo
eps=sqrt(eps)
write(*,*) ite,eps
if(eps.lt.pres) goto 200
enddo
c Done
200 continue
end
```

Now follows a direct implementation of the SOR algorithm,

$$(D + L)v^{n+1} = \omega v_{GS}^{n+1} + (1 - \omega)v^n \tag{12}$$

where v_{GS}^{n+1} is the result of the Gauss-Seidel iteration (10). The actual Fortran program looks like this:

```
program sor
c sor relaxation
dimension volt(0:1000),vnew(999),cond(0:999)
dimension resi(0:999)
c Reading parameter omega
write(*,1)
1 format(1x,' Omega = ')
read(*,*) om
c Random number initialization
```

```
rinv=0.5/2147483647.
rold=0.
ibm=1927
do i=1,1000
ibm=ibm*16807
enddo
c Generating the 1000 conductances
do i=0,999
ibm=ibm*16807
cond(i)=ibm*rinv+0.5
enddo
do i=1,999
resi(i)=1./(cond(i-1)+cond(i))
enddo
c Stopping criterion
pres=2.e-4
c Initializing voltages
do i=0,1000
volt(i)=float(i)/1000
enddo
c Iteration
do ite=1,1000000
do i=1,999
vnew(i)=volt(i)
enddo
do i=1,999
volt(i)=(cond(i-1)*volt(i-1)+cond(i)*volt(i+1))*resi(i)
enddo
do i=1,999
volt(i)=om*volt(i)+(1.-om)*vnew(i)
enddo
eps=0.
do i=1,999
eps=eps+(cond(i-1)*volt(i-1)+cond(i)*volt(i+1)
-(cond(i-1)+cond(i))*volt(i))**2
enddo
eps=sqrt(eps)
write(*,*) ite,eps
if(eps.lt.pres) goto 200
enddo
c Done
200 continue
end
```

With a stop criterion set to 2×10^{-4}, the Jacobi program used 115973 iterations, the Gauss-

Seidel program used 112 iterations. the SOR program used 75 iterations when $\omega = 2.5$. The Jabobi method performs a lot worse than theory would suggest.

The number of iterations goes *down* when increasing the number of iterations. One would naively have expected the opposite. The reason is that the current flowing through the chain is inversely proportional to the number of resistors. The stopping criterion is proportional to the current flowing. Hence, we need to adjust it down when adjusting the number of resistors up.