Mathematics 4670/5670 Last Assignment

December 6, 2017

The premise is that we have a system of *n* linear equations in *n* variables *x1, x2,…,xn* which we will write as

...

Some aspects of my plan are obvious I’ll have an array with two indexes each running from 1 to *n*. I can call this two dimensional array whatever I like, for example ‘linsysmatocoefficients’. Another choice would be ‘a’. I’m going with the second one. The number *a(i, j)* will be the coefficient of variable *xj* in equation number *i*. I’ll also have a one dimensional array *x* which eventually will contain the solution to the system, and also a one dimensional array *b* which initially holds the right sides of the equations. The actual calculation follows the pseudocode:

for each column index k, k = 1, 2, 3,...,n-1

for each row index i = k + 1,...,n

let m = a(i, k) / a(k, k)

replace equation number i with what it used to be

minus m times equation number k.

end

end

By ‘equation number i’, I mean the numbers in row *i* of *a* together with *bi.* Assuming that this process runs to completion, the linear system has been reduced to an upper triangular one:

…

To finally solve the system observe that the last equation says that *xn* = *bn / a(n, n)* and that backing up through the system one equation at a time results in an equation with only one unknown. This suggests the following end game.

x(n) = b(n) / a(n, n)

for each index I = n – 1, n – 2,..., 1

x(i) = (b(i) – (sum of a(i,j)\*x(j),j = i + 1 to n))/ a(i,i)

end

For clarity the statement in the loop is, in mathematical language

Problem 1. Implement the above code in Fortran and test it on an example you select with *n* = *5*.

Below is the implementation of the above pseudocode and I have also supplied the testing data used for the five by five matrix.

! Mikayla Webber

! 4670 Numerical Analysis

! Last Assignment Due 12.13.17

program linsysmatocoefficients

implicit none

integer :: i, j, k, n

double precision :: m, sum

double precision, allocatable, dimension(:,:) :: a

double precision, allocatable, dimension(:) :: b, x

n = 5

allocate(a(1:n,1:n),x(1:n), b(1:n))

! testing data when n = 5 (5 x 5 matrix)

a(1,:) = (/ 5.0d0, 4.0d0, 6.0d0, 8.0d0, 1.0d0 /)

a(2,:) = (/ 0.0d0, 3.0d0, 2.0d0, 3.0d0, 5.0d0 /)

a(3,:) = (/ 0.0d0, 0.0d0, -7.0d0, 1.0d0, 9.0d0 /)

a(4,:) = (/ 0.0d0, 0.0d0, 2.0d0, 1.0d0, 2.0d0 /)

a(5,:) = (/ 0.0d0, 0.0d0, 0.0d0, 4.0d0, 5.0d0 /)

b = (/ 7.0d0, -1.0d0, 1.0d0, 5.0d0, 4.0d0 /)

do k = 1, (n - 1)

do i = (k + 1), n

m = a(i,k) / a(k,k)

do j = k, n

a(i,j) = a(i,j) - m \* a(k,j)

end do

b(i) = b(i) - m \* b(k)

end do

end do

x(n) = b(n) / a(n,n)

do i = (n - 1), 1, -1

sum = 0.0d0

do j = n, (i + 1), -1

sum = (sum + (a(i,j) \* x(j))) !top half

end do

x(i) = ((b(i) - sum) / a(i,i))

end do

print\*, 'creating output file for Gaussian loop'

open(unit=8, file='out1', status='replace')

do i = 1, n

write(8,\*) i, ': ', x(i)

end do

close(8)

deallocate(a, b, x)

stop

end program linsysmatocoefficients

Problem 2. In the previous discussion there is one thing that could go horribly wrong. If the matrix ends up triangular with a zero on the diagonal this is the indicator that one doesn’t have a unique solution; either there is no solution at all, or there are infinitely many solutions.

On the other hand though, if during the reduction process a zero appears as a diagonal entry in A it is not necessarily true that doom ensures. Suppose that you were working such a problem and encountered the following partially reduced matrix and right side vector

5 4 6 8 1 7

3 2 3 5 -1

0 4 5 4

2 1 2 5

-7 1 9 1

partially reduced right

coefficient matrix sides

Since a33 = 0 there’s no way to subtract a multiple of row three from row four in order to turn the two into a zero. Likewise we see, to be stuck with the minus seven. This is of course somewhat silly.I can take the last three equations and write them in a different order. The most common strategy is to put into position (3,3) the number with the largest possible absolute value, taken from . This is based on the idea that if zero is bad, far from zero must be good. Exchanging equations three and five produces

5 4 6 8 1 7

3 2 3 5 -1

-7 1 9 1

2 1 2 5

0 4 5 4

partially reduced right

coefficient matrix sides

And now the trouble has gone away. This is called maximal row pivoting. The only way this can fail is if and all entries directly below that entry are also zero. This would flag the non unique solution situation. This exchange idea can be incorporated in the prior reduction scheme with little pain:

for each column index k, k = 1, 2, 3,..., n – 1

determine an index m in {k, k + 1,...,n} so that the absolute value of a(m, k) is as large as possible,

if m > k interchange equations m and equations k not forgetting to interchange b(k) and b(m)

for each row index i = k + 1,..., n

if | a(k, k) | is small by some yard stick warn the user and possibly quit.

Assuming we’re going onwards

let m = a(i, k)/a(k, k)

replace equation number i with what it used to be minus m times equation number k.

end

end

Implement this and test it on an example with *n* = *10*.

Below is the implementation of the maximal row pivoting with the main driver program from problem one. I have also supplied the testing data for the 10 by 10 matrix which I used to test the program.

! Mikayla Webber

! 4670 Numerical Analysis

! Last Assignment Due 12.13.17

program linsysmatocoefficients

implicit none

integer :: i, j, k, n

double precision :: m, sum

double precision, allocatable, dimension(:,:) :: a

double precision, allocatable, dimension(:) :: b, x

n = 10

allocate(a(1:n,1:n),x(1:n), b(1:n))

! testing data when n = 10 (10 x 10 matrix)

a(1,:) = (/ 2.0d0,1.0d0,2.0d0,3.0d0,1.0d0,1.0d0,1.0d0,1.0d0,2.0d0,6.0d0 /)

a(2,:) = (/ 4.0d0,2.0d0,6.0d0,6.0d0,1.0d0,3.0d0,1.0d0,3.0d0,6.0d0,4.0d0 /)

a(3,:) = (/ 6.0d0,3.0d0,4.0d0,9.0d0,1.0d0,2.0d0,1.0d0,2.0d0,7.0d0,5.0d0 /)

a(4,:) = (/ 8.0d0,4.0d0,8.0d0,12.0d0,1.0d0,3.0d0,2.0d0,3.0d0,5.0d0,9.0d0 /)

a(5,:) = (/ 10.0d0,5.0d0,4.0d0,15.0d0,4.0d0,5.0d0,1.0d0,5.0d0,4.0d0,1.0d0 /)

a(6,:) = (/ 12.0d0,6.0d0,5.0d0,2.0d0,5.0d0,1.0d0,2.0d0,1.0d0,9.0d0,3.0d0 /)

a(7,:) = (/ 14.0d0,7.0d0,9.0d0,4.0d0,2.0d0,3.0d0,6.0d0,3.0d0,3.0d0,5.0d0 /)

a(8,:) = (/ 16.0d0,8.0d0,1.0d0,6.0d0,1.0d0,2.0d0,8.0d0,2.0d0,1.0d0,7.0d0 /)

a(9,:) = (/ 18.0d0,9.0d0,2.0d0,8.0d0,2.0d0,3.0d0,1.0d0,3.0d0,5.0d0,8.0d0 /)

a(10,:) = (/ 20.0d0,10.0d0,5.0d0,10.0d0,6.0d0,5.0d0,6.0d0,5.0d0,2.0d0,2.0d0 /)

b = (/ 20.0d0,12.0d0,10.0d0,25.0d0,8.0d0,6.0d0,5.0d0,18.0d0,8.0d0,2.0d0 /)

do k = 1, (n - 1)

call interchange(a, n, k, b)

do i = (k + 1), n

m = a(i,k) / a(k,k)

do j = k, n

a(i,j) = a(i,j) - m \* a(k,j)

end do

b(i) = b(i) - m \* b(k)

end do

end do

x(n) = b(n) / a(n,n)

do k = (n - 1), 1, -1

sum = 1.0d0

do i = n, (k + 1), -1

sum = (sum + (a(k,i) \* x(i))) !top half

end do

x(k) = ((b(k) - sum) / a(k,k))

end do

print\*, 'creating output file for Gaussian loop'

open(unit=8, file='out2', status='replace')

do i = 1, n

write(8,\*) i, ': ', x(i)

end do

close(8)

deallocate(a, b, x)

stop

end program linsysmatocoefficients

Below is the subroutine interchange which is used by the main driver program for problem two which involves maximal row pivoting.

subroutine interchange(a, n, k, b)

implicit none

integer :: i, j, k, n, max

double precision, allocatable, dimension(:) :: aMax

double precision :: a(n,n), b(n), bMax, vMax

allocate(aMax(n))

do i = k, n

vMax = a(i,k)

max = i

do j = (i + 1), n

if (abs(a(j,k)) > vMax) then

vMax = a(j,k)

max = j

end if

end do

aMax = a(i,:)

a(i,:) = a(max,:)

a(max,:) = aMax

bMax = b(i)

b(i) = b(max)

b(max) = bMax

end do

deallocate(aMax)

end