Mathematics 4670 Homework Three

1. In the handout describing the simple method for finding interpolating polynomials I had a subroutine named createa with header

subroutine createa(n, xdata, ydata, a)

to find the coefficients using the recursive divided difference idea. Inside the subroutine create the two index table as we discussed in class and just before you return from the subroutine, save your results into the one dimensional array *a* (indexed zero to *n*) and deallocate the two dimensional array. I want you to explain in full detail your thoughts, as detailed in the handout I gave you. If you present some source code with little or no discussion I'll award zero points.

CODE:

! Mikayla Webber

! 4670 Numerical Analysis

! Homework Three Due 10/18/17

module secret ! creates secret module

integer :: fcounter ! declares counter that could be put into

! a do loop to count iterations

end module secret

program WebberHomework3Question1 ! main program

use secret ! use secret module

implicit none

integer :: i

integer :: n

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

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

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

n = 4 ! amount of xdata/ydata values - 1

allocate(xdata(0:n), ydata(0:n), a(0:n))

xdata = (/ 2.0d0, 4.0d0, 6.0d0, 5.0d0, 9.0d0 /) ! hardcoded test data

ydata = (/ 12.0d0, 19.0d0, 14.0d0, 17.0d0, 10.0d0 /) ! hardcoded test data

print\*, 'Newtons Divided Difference Coefficients: '

call divdiff(n, xdata, ydata, a) ! calls the subroutine function

do i = 0, n

print\*, a(i) ! do loop to print data

end do

deallocate(xdata, ydata, a)

stop

end program WebberHomework3Question1 ! ends main program

subroutine divdiff(n, xdata, ydata, a) ! beginning of subroutine divdiff

! with the variables n, xdata, ydata, a

integer :: i

integer :: j

integer :: n

double precision :: a(0:n)

double precision :: xdata(0:n)

double precision :: ydata(0:n)

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

allocate (T(0:n,0:n))

do i = 0, n

T(i,0) = ydata(i)

end do

a(0) = ydata(0)

do i = 1, n

do j = 1, n

T(i,j) = ((T(i,(j - 1)) - T((i - 1),(j - 1))) / (xdata(i) - xdata(i - j)))

if (i == j) then

a(i) = T(i,j)

end if

end do

end do

deallocate(T)

end subroutine divdiff ! ends the subroutine

CODE RUN RESULTS (for hardcoded test data):

Newtons Divided Difference Coefficients:

12.0000000000

3.50000000000

-1.5000000000

0.33333333333

-2.142857142857E-02

CODE ALTERNATIVE (This one has two files and allows you to change the values in the readTable.txt and output the results to resultsTable.txt)

! Mikayla Webber

! 4670 Numerical Analysis

! Homework Three Due 10/18/17

module secret ! creates secret module

integer :: fcounter ! creates counter which could track

! iterations through a do loop

end module secret

program WebberHomework3Question1 ! main program

use secret ! uses secret module

implicit none

integer :: i

integer :: n

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

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

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

n = 10

open(unit=8, file='readTable.txt', status='old') ! opens a text file to read data from

open(unit=9, file='resultTable.txt', status='replace') ! creates/rewrites text file with the

! results

read(8,\*) n

write(9,\*) n

allocate(xdata(0:n), ydata(0:n), a(0:n))

do i = 0, n

read(8,\*) xdata(i), ydata(i) ! reads data from file

write(9,\*) xdata(i), ydata(i) ! writes data to file

fcounter = (fcounter + 1) ! increase the counter

end do

print\*,'This program was ran this many times: ', fcounter

print\*, 'Newtons Divided Difference Coefficients: '

call divdiff(n, xdata, ydata, a)

do i = 0, n

print\*, a(i)

end do

deallocate(xdata, ydata, a)

close(8) ! closes read file

close(9) ! closes write file

stop

end program WebberHomework3Question1

subroutine divdiff(n, xdata, ydata, a) ! creates subroutine function with

! variables n, xdata, ydata, and a

integer :: i

integer :: j

integer :: n

double precision :: a(0:n)

double precision :: xdata(0:n)

double precision :: ydata(0:n)

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

allocate (T(0:n,0:n))

do i = 0, n

T(i,0) = ydata(i)

end do

a(0) = ydata(0)

do i = 1, n

do j = 1, n

T(i,j) = ((T(i,(j - 1)) - T((i - 1),(j - 1))) / (xdata(i) - xdata(i - j)))

if (i == j) then

a(i) = T(i,j)

end if

end do

end do

deallocate(T)

end subroutine divdiff

DISCUSSION:

When I originally read the problem I had the idea of opening a file to input the data and then creating another file to output the results into. That way if the data needs to be changed it will be easier to do and there would be less of a chance of the code that is doing the calculations being disturbed. This is what the "Code Alternative" above includes. This alternative code is the same in every aspect to the original code posted above except for a few added changes. In the alternative code there is a counter incrementing in one of the do loops to keep track of iterations and also prints the number of iterations to the user. It also opens the readTable.txt file for data input and creates (or writes over) the resultsTable.txt with the results. After it opens the files there is also a loop which goes through the readTable file to get the data and writes the results to the resultsTable.txt file. There are also two additional lines closing the files.

After creating the version with the files, I re-read the instructions and decided to make the code above the alternative since it more closely reflects the instructions. Here is where I will describe the steps that the program is going through.

The program begins by creating a secret module, like the one discussed during class, which helps encapsulate the definitions and it creates the integer fcounter which will count the iterations in the main program. After the secret module the main program begins (called WebberHomework3Question1) and it uses the module secret. In the main there are two integers declared and three allocatable dimensions (of the type double precision). One of the integers, *n*, will be set to the amount of *xdata* and *ydata* values minus one. For example, if you had five *xdata* values and 5 *ydata* values then you would set *n* to 4. The next line in the program allocates memory for the *xdata*, *ydata*, and *a* all with the bounds of zero to *n*. The next two lines are the hardcoded data for the *xdata* and *ydata* where you would insert the data (for example: *x* for *xdata* and *f(x)* for *ydata*). The ".0d0" endings are not necessary but provide extra reassurance that the data will be used correctly. The calculations for the divided difference are done in the divdiff subroutine which is defined after the main program so to use it we need to have a call statement which is the "*call divdiff(n, xdata, ydata, a)*" and within the parenthesis is the data which is being passed to the subroutine function. Next is a do loop which will output the results for 0 to *n*. Afterwards, we deallocated the memory we allocated in the beginning and stop the program.

The subroutine is defined with the variables being passed it within the parenthesis. Within the subroutine there are three integers declared and three double precision variables within the bounds of 0 to *n*. There is also an allocatable two dimension variable T created. After declaring what will be used in the subroutine memory is allocated to the two dimension array T with the bounds of 0 to *n*. Next is a do loop that sets the *ydata(i)* to the first position of the T array. After the loop *ydata(i)* is set to *a.* The next do loop has a nested do loop within it. Both loops go from one to *n*. The inside loop formula is: *T( i, j which is decremented with each run through the second loop) - T (i which is decremented with each run through the first loop, j which is decremented with each run through the second loop) divided by the i in xdata minus (i-j) which is held in the xdata.* However, if the *i* value is the same as the *j* value then the *i* value in *a* is set as both the *i* and *j* value held in *T*. The two dimensional array T is then deallocated and the subroutine is ended.

2. Write a program using the subroutine from problem one to reproduce table 3.9 from the text book. Provide your driver program and the output. You've already fully described subroutine divdiff so you don't need to show it again.

CODE:

! Mikayla Webber

! 4670 Numerical Analysis

! Homework Three Due 10/18/17

module secret ! creates secret module

integer :: fcounter ! declares counter that could be put into

! a do loop to count iterations

end module secret

program WebberHomework3Question2 ! main program

use secret ! use secret module

implicit none

integer :: i

integer :: n

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

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

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

n = 4 ! amount of xdata/ydata values - 1

allocate(xdata(0:n), ydata(0:n), a(0:n))

xdata = (/ 1.0d0, 1.3d0, 1.6d0, 1.9d0, 2.2d0 /) ! hardcoded test data

ydata = (/ 0.7651977d0, 0.6200860d0, 0.4554022d0, 0.2818186d0, 0.1103623d0 /)

! hardcoded test data

print\*, 'Newtons Divided Difference Coefficients: '

call divdiff(n, xdata, ydata, a) ! calls the subroutine function

do i = 0, n

print\*, a(i) ! do loop to print data

end do

deallocate(xdata, ydata, a)

stop

end program WebberHomework3Question2 ! ends main program

subroutine divdiff(n, xdata, ydata, a) ! beginning of subroutine divdiff

! with the variables n, xdata, ydata, a

integer :: i

integer :: j

integer :: n

double precision :: a(0:n)

double precision :: xdata(0:n)

double precision :: ydata(0:n)

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

allocate (T(0:n,0:n))

do i = 0, n

T(i,0) = ydata(i)

end do

a(0) = ydata(0)

do i = 1, n

do j = 1, n

T(i,j) = ((T(i,(j - 1)) - T((i - 1),(j - 1))) / (xdata(i) - xdata(i - j)))

if (i == j) then

a(i) = T(i,j)

end if

end do

end do

deallocate(T)

end subroutine divdiff ! ends the subroutine

CODE RUN RESULTS (for table 3.9 test data that was hardcoded):

Newtons Divided Difference Coefficients:

0.765197700000

-0.483705666667

-0.108733888889

6.587839506173E-02

1.825102880660E-03

3. Use your subroutine to find the coefficients of the polynomial of smallest degree that interpolates the following data.

x y

1 -4

2 -11

3 -14

4 -7

5 16

CODE:

! Mikayla Webber

! 4670 Numerical Analysis

! Homework Three Due 10/18/17

module secret ! creates secret module

integer :: fcounter ! declares counter that could be put into

! a do loop to count iterations

end module secret

program WebberHomework3Question3 ! main program

use secret ! use secret module

implicit none

integer :: i

integer :: n

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

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

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

n = 4 ! amount of xdata/ydata values - 1

allocate(xdata(0:n), ydata(0:n), a(0:n))

xdata = (/ 1.0d0, 2.0d0, 3.0d0, 4.0d0, 5.0d0 /) ! hardcoded test data

ydata = (/ -4.0d0, -11.0d0, -14.0d0, -7.0d0, 16.0d0 /) ! hardcoded test data

print\*, 'Newtons Divided Difference Coefficients: '

call divdiff(n, xdata, ydata, a) ! calls the subroutine function

do i = 0, n

print\*, a(i) ! do loop to print data

end do

deallocate(xdata, ydata, a)

stop

end program WebberHomework3Question3 ! ends main program

subroutine divdiff(n, xdata, ydata, a) ! beginning of subroutine divdiff

! with the variables n, xdata, ydata, a

integer :: i

integer :: j

integer :: n

double precision :: a(0:n)

double precision :: xdata(0:n)

double precision :: ydata(0:n)

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

allocate (T(0:n,0:n))

do i = 0, n

T(i,0) = ydata(i)

end do

a(0) = ydata(0)

do i = 1, n

do j = 1, n

T(i,j) = ((T(i,(j - 1)) - T((i - 1),(j - 1))) / (xdata(i) - xdata(i - j)))

if (i == j) then

a(i) = T(i,j)

end if

end do

end do

deallocate(T)

end subroutine divdiff ! ends the subroutine

CODE RUN RESULTS:

Newtons Divided Difference Coefficients:

-4.00000000000

-7.00000000000

2.00000000000

1.00000000000

0.00000000000

4. Suppose that you were calculaing divided differences as we have indicated, and all the divided differences in column number *k* were zero. What does that say about divided differences in later columns? What does that say about your data set?

If you were calculating the divided differences and all of the divided differences in column *k* were zero this says that no matter what the *f(x)* value is the divided differences in the later columns would also be zero. This says that the data set is also equivalent to zero.