Creating a Program from Start to Finish

Brandon Reyes

Colorado School of Mines

Outline

Introduction

- Objective: Create a program that computes a matrix vector product.
- To begin, we will first establish the "theory" of the problem.

$$\begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

- Where the $i - j^{th}$ entry of matrix B is 1.5*i*j and the i^{th} entry of vector A is .5*i.

Creating a Module

```
module matrix_module
  contains

  !Constructs an N by 1 vector A
  subroutine A_vec(N,A)
    .
    .
    .
  end subroutine A_vec

  !Constructs the B matrix for a given N and M
  function B_mat(M,N)
    .
    .
    .
  end function B_mat

  !computes the vector C serially
  subroutine serial_mult(A,B,C)
    .
    .
    .
  end subroutine serial_mult

  !Vectorized form of matrix vector multiplication
  subroutine vec_mult(A,B,C)
    .
    .
    .
  end subroutine vec_mult

end module matrix_module
```

Coding the A vector

```
!Constructs an N by 1 vector A
subroutine A_vec(N,A)
  Implicit none
  !Declaring Variables
  integer, parameter :: mykind = selected_real_kind(16,300)
  real(kind=mykind),allocatable, dimension(:),intent(out) :: A
  integer,intent(in) :: N
  integer :: i

  !Allocating our vector of length N
  Allocate(A(N))

  !Filling in the A matrix, where the i th entry is .5*i
  do i = 1,N
     A(i) = real(.5,mykind)*real(i,mykind)
  end do

end subroutine A_vec
```

Coding the B matrix

```
!Constructs the B matrix for a given N
function B_mat(M,N)
  implicit none
  !Declaring Variables
  integer, parameter :: mykind = selected_real_kind(16,300)
  real(kind=mykind),allocatable, dimension(:,:) :: B_mat
  integer,intent(in) :: M,N
  integer :: i,j

  !Allocating our N by N matrix
  Allocate(B_mat(M,N))

  !Filling in the B matrix, where the i-j th entry is 1.5*i*j
  do i = 1,M
     do j=1,N
        B_mat(i,j) = real(1.5,mykind)*real(i,mykind)*real(j,mykind)
     end do
  end do

end function B_mat
```

Serial Matrix-Vector Multiplication

```
!computes the vector C serially
subroutine serial_mult(A,B,C)
  implicit none
  !Declaring our Variables
  integer, parameter :: mykind = selected_real_kind(16,300)
  real(kind=mykind),allocatable, dimension(:,:),intent(in) :: B
  real(kind=mykind),allocatable, dimension(:), intent(in) :: A
  real(kind=mykind),allocatable, dimension(:), intent(out) :: C
  real(kind=mykind) :: vec_sum
  integer :: i,j,A_row,B_row,B_col

  !Obtaining the dimensions of A
  A_row = size(A)

  !Obtaining the dimensions of B
  B_row = size(B(:,1))
  B_col = size(B(1,:))

  !Allocating our C vector which is the matrix B times the vector A
  ALLOCATE(C(A_row))

  !computing the vector C in a serial manner
  do i= 1,B_row
     vec_sum = 0
     do j = 1,B_col
        vec_sum = vec_sum + B(i,j)*A(j)
     end do
     C(i) = vec_sum
  end do

end subroutine serial_mult
```

Vectorized Matrix-Vector Multiplication

```
!Vectorized form of matrix vector multiplication
subroutine vec_mult(A,B,C)
  implicit none
  !Declaring our Variables
  integer, parameter :: mykind = selected_real_kind(16,300)
  real(kind=mykind),allocatable, dimension(:,:),intent(in) :: B
  real(kind=mykind),allocatable, dimension(:), intent(in) :: A
  real(kind=mykind),allocatable, dimension(:), intent(out) :: C
  integer :: i,A_row,B_row,B_col

  !Obtaining the dimensions of A
  A_row = size(A)

  !Obtaining the dimensions of B
  B_row = size(B(:,1))
  B_col = size(B(1,:))

  !Allocating our C vector which is the matrix B times the vector A
  ALLOCATE(C(A_row))

  !computing the vectorx C using vectors
  do i = 1,B_row
     C(i) = sum(B(i,:)*A)
  end do


end subroutine vec_mult
```

Outline of the Main File

- Read in values M and N from input.txt file to specify the M by N matrix B and N by 1 vector A.
- Call and time the serial matrix vector product .
- Call and time the vectorized version of the matrix vector product.
- Print the matrix B and vector A if N is less than 5 for the serial version of the product.
- Print the vector C if N is less than 5 for both versions of the product.

Creating the main file

```
program main
  use matrix_module   !allows us to use the module we created
  implicit none

  !Declaring our variables
  integer, parameter :: mykind = selected_real_kind(16,300)
  real(kind=mykind),allocatable, dimension(:,:) :: B
  real(kind=mykind),allocatable, dimension(:) :: A,C
  integer :: N,M,i
  real :: time1,time2

  !opening our input file
  OPEN(UNIT=13,file='input.txt')

  !Reading in the number of rows of our matrix
  READ(13,*) M

  !setting the number of columns of our matrix and the number of rows of vector A
  READ(13,*) N

  !closing the file because it is not needed anymore
  close(13)

  !constructing the N by 1 A vector
  call A_vec(N,A)

  !constructing the M by N B matrix
  B = B_mat(M,N)
```

Creating the main file

```
!printing the vector A and matrix B if N < 5
if(N < 5)then
   print *, ''
   print *, 'Matrix B :'
   print *, ''
   do i = 1,M
      print *, B(i,:)
   end do
   print *, ''
   print *, 'Vector A :'
   print *, ''
   do i = 1,N
      print *, A(i)
   end do
   print *, ''
end if

!Computing C serially
call CPU_TIME(time1)
call serial_mult(A,B,C)
call CPU_TIME(time2)

!creating a space for asthetics
if(N > 4)then
   print *, ''
end if

print *, 'Elapsed time for serial version : ', time2 - time1
print *, ''
```

Creating the main file

```
    if(N < 5)then
       !printing out C from the serial matrix multiplication
       print *, 'Vector C from serial version : '
       print *, ''
       do i = 1,N
          print *, C(i)
       end do
       print *, ''
    end if

    !Deallocating C so we can use it again for vector matrix multiplication
    DEALLOCATE(C)

    !Computing C using vectors
    call CPU_TIME(time1)
    call vec_mult(A,B,C)
    call CPU_TIME(time2)
    print*, 'Elapsed time for vectorized version : ', time2 - time1
    print*, ''

    if(N < 5)then
       !printing out C from the vector matrix multiplication
       print *, 'Vector C from vector version : '
       print *, ''
       do i = 1,N
          print *, C(i)
       end do
       print *, ''
    end if

    !Freeing the memory for all allocatables
    DEALLOCATE(C,A,B)

end program main
```

Creating the makefile

```
FC90 = gfortran
vector_files = matrix_module.f90 main.f90

all:main_exe

main_exe:$(vector_files)
	$(FC90) $(vector_files) -o	$@
clean:
	rm	*_exe	*.mod *~
```

The readme.txt file

```
This program creates an M by N matrix B (where the i-j th entry is 1.5*i*j) and multiplies it by
a N by 1 vector A (where the i th entry is .5*i). The result is then stored in a vector C. Note
that this program computes this matrix vector multiplication by first doing it in a serial manner
and then in a vectorized way.

To compile the program :

type :

make main_exe

then press enter to create our executable

To run the program :

./main_exe

To run the program and save to a.out :

./main_exe > a.out
```

Results

Now that we have coded everything, let's compile and run our code.

Turning in Assignments

- Make sure you have all of your essential files
  - For our program the essential files are: input.txt , main.f90, makefile, matrix_module.f90, and readme.txt .
- Do not leave executable files (_exe) or compiled modules (.mod).
- Create a zip file of your final turn in folder
  - If all of our files are in the directory all_mult, we can create a zip file using the following command:
    tar -czvf all_mult.tar.gz all_mult
- Report
  - The report must be very well written, assume that your audience is someone who has little to no experience with coding or the assignment at hand.
  - Make sure all your code is well commented. I will go through the code, so make it readable.
  - At the end of your report, cite who helped you.
  - If you would like to see an example report come visit me during my office hours (4-5pm everyday).