AM 129 HW2 Deliverable

Sarosh Sopariwalla
October 27, 2020

Abstract

Our goal for this project was to solve a common type of ordinary differential equation called a boundary value problem using numerical methods. The structure of our final code was given to us but we were responsible for writing functions and subroutines as well as implementing the final overall steps.

Usage of real(dp)

In both of our `dft.90` and `dftmod.f90`, we tell Fortran to refer to certain conventions as defined in our utility file. We did so by writing `use utility, only : dp, pi`. Thus, whenever we use `dp`, Fortran understands that it should refer to the definition of dp found in the utility file: `integer, parameter :: dp = kind(0.d0)`.

Why not Combine dft.f90 and dftmod.f90?

One could argue that since our program is so simplistic, it makes sense to just define one file `dft.f90` and put all of our subroutines, functions, and main code in there. While this is doable, it would just make our life more difficult and make the `dft.f90` harder to read and debug.

Functions at the Bottom of dft.90

We can include internal procedures(functions/subroutines) to our code by adding the `contains` argument at the end of our file. Thus, all the functions we wrote after `contains` did not need to be declared `external`. The keyword `elemental` tells our procedure to work on each element of the input array individually. So if our procedure computed the sine of our input, we could feed in an array as our input and the `elemental` keyword would tell our procedure to compute the sine of each element. (Rather than attempting to compute the sine of an array which wouldn't make sense)

Maximum Error as N Varies

Looking at the table below, it is clear that as N increases, the maximum error decreases. Looking at the limited data below, we see that the maximum error does eventually stop decreasing. Specifically the maximum error for $N = 80$ is $3.11 * 10^{-15}$ whereas the maximum error for $N = 100$ is $3.33 * 10^{-15}$. (Note $3.33 * 10^{-15} > 3.11 * 10^{-15}$). If we plug in large N (like $N = 400$) we see a maximum error of $5.32 * 10^{-15}$. Thus, Maximum error is minimized somewhere between $N = 80$ and $N = 100$. However, even though the minimum occurs between these values of N, this is not very important because all the errors on the scale of 10^{15}. (On the flip side, this is good to know because we save processing power by using smaller $N \approx 80$.) The fact that the error stops decreasing indicates that for large values of N, our precision level is too low. A table displaying Maximum Error as N varies can be found below:

N	Maximum Error
20	$1.45 * 10^{-2}$
40	$9.96 * 10^{-7}$
60	$5.64 * 10^{-10}$
80	$3.11 * 10^{-15}$
100	$3.33 * 10^{-15}$

Change dp Definition and Study Maximum Error for Varying N

We still notice that the error stops decreasing at a certain point; in particular, the maximum error stops decreasing for some N between 80 and 100. Furthermore, the Maximum error tends to be higher in this case where we have `dp=kind(0.e0)`. Notice, the minimum of the maximum error earlier was on the scale of 10^{-15}, but the minimum of the maximum error here is closer to 10^{-6}.

N	Maximum Error
20	$1.45 * 10^{-2}$
40	$3.70 * 10^{-6}$
60	$2.15 * 10^{-6}$
80	$1.19 * 10^{-6}$
100	$2.38 * 10^{-6}$

Understanding the makefile Flags

There are several flags in our makefile that we need to interpret. These are labeled as FFLAGS and we explain each one here below. All this information can be found on the gfortran manual at https://linux.die.net/man/1/gfortran.

- `-Wall` is a flag that tells the compiler to use several other flags that give warning options about syntax and usage. Calling `-Wall` is the same as calling `-Waliasing, -Wampersand, -Wsurprising, -Wintrinsics-std, -Wno-tabs, -Wintrinsic-shadow,` and `-Wline-truncation`. We will go over a few of these in the following lines but, the rest can be found in the gfortran manual.
- `-Wextra` essentially enables extra warning flags that are not enabled by `-Wall`. It also warns about unused parameters in your code.
- `-Wimplicit-interface` warns if a procedure is called without an explicit interface our environment.
- `-Wno-surprising` warns about any suspicious code (for example, if you try to do an integer select between two bounds but the upper bound is less than the lower bound)
- `-fPIC` determines whether or not the compiler generates ”position-independent code”. Essentially, this checks that we have a certain section of memory dedicated to the outputs of our code.
- `-fmax-errors=1` limits the number of error messages. (In this case, we will only see a maximum of one error message.)
- `-g` tells the compiler to compile using the dbx debugger. dbx debugger is used to debug various languages like C, Fortran, and Pascal. It is generally used on Linux systems rather than Macs or Windows
- `-fcheck=all` enables all run time checks. If the code runs too long (or infinitely) it will automatically get cut off at a certain point.
- `-fbacktrace` If there is a runtime error Fortran will go back to find where the problem occured. So if one of our functions had an error. Fortran would not just say there was an error at MatVecProd. It would go into MatVecProd and point out the error specifically.

Conclusion

Our goal was to solve the boundary value problem $(1 - \frac{d^2}{dx^2})u = f, x \in (0, 2\pi)$ where $u(0) = u(2\pi)$ and $f(x) = e^{\sin(x)}(1 + \sin(x) - \cos^2(x)) + e^{\cos(x)}(9\sin^2(3x) - 9\cos(3x) - 1)$. Clearly this problem would be hard to solve by hand, so we solved it numerically using Fortran 90. As students, we had three main tasks. The first two tasks involved filling out the `dftmod.f90` file (that was later used by the main `dft.90` file.) Firstly, we wrote our own function called `matvecprod` that finds the product of a matrix and vector without using MATMUL. Then, we filled out a subroutine `dft_TransMat` that created a transformation matrix. Next, we wrote our own subroutine `dft_InvTransMat` that created the inverse transformation matrix. Finally, we filled out some of the details in `dft.90` and implemented our functions and subroutines. Our algorithm relied upon a parameter N and through some experimentation, we found that our error (between the numerical and analytic solution) was minimized for N close to 80.