Submissions: For this homework, you should submit Matlab M-files and a write-up. Your submissions should include
(1) A description of how you solved the problem. This description should be in your write-up.
(2) The specific output requested for each problem in the write-up and
(3) The Matlab M-files that you wrote. (I run your codes.) If your code has “strange”
behavior such as crashing, infinite loops, or excessively long run times (more than a few
seconds), let me know in your submissions.

I strongly recommend that you start early on these problems and send questions or come to see me as soon as you encounter problems. You can email tah@mit.edu to set a time for an appointment. Office hours are Tuesday 1:00-2:00 pm.

Question (1): (25-points) Write a program that reads a file containing text, counts the number of characters (letters a-z) and words in the text, and output the text with capital letters at the beginning of each sentence. The text below is contained in the file Q1_text.txt.

we take as a self evident foundational principle that the set of effects to be considered as contributing to local station displacements and the conventional models to be applied for their compensation should be guided by rational and well considered bases, and should not be developed haphazardly or randomly. For historical reasons and general consistency, it might be prudent to retain some past practices even if they are not fully consistent with the adopted principles; but future expansions should be determined by specified rules. this position paper proposes such a set of guidelines and rationales for IERS conventions updates.

Hints:
(1) Look at the ASCII table and check the relationship between upper and lower case letters. In C/C++ strings are just integer arrays (declared as char) and the entries in these arrays can be changed with integer arithmetic.
(2) Remember if reads are coded with scanf or getchar (both of which read from stdiin) then the file Q1_text.txt can be re-directed into the program using:
Q1C < Q1text.txt where Q1C is the name of the program (you can call the program any name you like).

Question (1) Solution
This solution is very similar to the C solution in the algorithm used because the characters are scanned individually and the end-of-line (\n) characters are still in the string. In the solution code, we read the file fileread which reads the whole file into a string (with \n still included). We could have read fgetl to read each line in which case the fortran code would have been a better starting point. The other difference from the C-code is the \n’ is not interpreted as a newline character. We need to make a string with that character using char(10) (10 is the ASCII code for \n).
Input of the file name in the solution is done using iugetfile. While using this routine it is important to return both the directory and filename. In case the file is not in the current directory.

**The Matlab code I used is linked to HW05Q01.m**

```matlab
>> HW05Q01
12.010 HW05 Question 01
input file /Users/tah/TAH_docs/12.010/Q1_text.txt
There are 533 characters and 99 words in the file
Capitalized text
We take as a self evident foundational principle that the set of effects
to be considered as contributing to local station displacements and the
conventional models to be applied for their compensation should be
guided by rational and well considered bases, and should not be developed
haphazardly or randomly. For historical reasons and general consistency,
it might be prudent to retain some past practices even if they are not
fully consistent with the adopted principles; but future expansions
should be determined by specified rules. This position paper proposes
such a set of guidelines and rationales for IERS conventions updates.
>>

For the full test case:
```matlab
>> HW05Q01
12.010 HW05 Question 01
input file /Users/tah/TAH_docs/12.010/HW_2010/HW05_soln/Q1_full.txt
There are 341 characters and 83 words in the file
Capitalized text
This file tests different aspects of 12.010 Question 1 counting and
capitalization. There can be more and less than 2 spaces at the
end of a sentence! In this case there is just one and here we have a question?
Will this is capitalized. I may also include the text with a quote.

"Will this be counted and capitalized."

Will the code count a hyphenated word as one or two words?
The results should be:

File Q1_full.Txt has 341 characters and 83 words
Question (2): **(25-points)** Write a program to accurately sum a large number of floating point numbers. The floating point numbers to be summed should be constructed as random numbers of the form

\[
\text{Value} = \text{mantissa} \times 10^{\text{exponent}}
\]

Where mantissa and exponent are generated as random numbers between 0-1 and exponent as random number between 0 and max_exponent. These numbers should be generated with 7 significant digits of accuracy.

max_exponent should be a user specified value (typical value would be 10) and the number of values to be summed should also be user specified (typical value 10000). The sum should be made using an accuracy of 7-significant digits and your results can be compared to a summation performed with high-accuracy to see how accurate your result is. The program should try to achieve the maximum summation accuracy and output the accuracy of the low accuracy sum. Your submission should explain how your algorithm achieves its accuracy using only low precision variables and the different methods you considered. Can a formula be found that allows a prediction of the accuracy of the algorithm?

**Question 2 Solution**

This solution is effectively identical to the Fortran and C solutions except that the matlab sort function is used to sort the data. The rand function here operates by default the same as in Fortran, generating uniform numbers between 0 and 1. Also as an option to rand, you can specify the precision of the numbers. Matlab has both sort and sum routines and for the sum routine you specify if ‘native’ or ‘double’ precision. By making the random number single and summing with ‘native’ precision, we can sum in single precision without using a for loop.

For this solution we also implemented the Kahan summation algorithm which seeks to minimize errors in summation by keeping track of the estimated errors. A description can be found at:


This algorithm applied to the non-sorted data works very well.

**Results:**

**The Matlab code is linked to [HW05Q02.m](hw05q02.m)**

**Example run:**

```matlab
>> HW05Q02
12.010 HW05 Q 02: Test summation methods
Maximum exponent value <38 10
Number of values to sum 10000

<table>
<thead>
<tr>
<th>Method</th>
<th>Real*4 Sum</th>
<th>Real*8 Sum</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Sum (sums)</td>
<td>3.105052426e+12</td>
<td>3.1050785615458271e+12</td>
<td>-8.417e-06</td>
</tr>
<tr>
<td>Ascending Sum (suma)</td>
<td>3.105078641e+12</td>
<td>3.1050785615458271e+12</td>
<td>2.547e-08</td>
</tr>
<tr>
<td>Descending Sum (sumd)</td>
<td>3.105020707e+12</td>
<td>3.1050785615458271e+12</td>
<td>-1.863e-05</td>
</tr>
<tr>
<td>Binned Sum (sumb)</td>
<td>3.105078378e+12</td>
<td>3.1050785615458271e+12</td>
<td>-5.895e-08</td>
</tr>
</tbody>
</table>
```

NB 16
**Question 3: (50-points)** Solution to the Lorenz Strange Attractor problem. Implement a solution to the following differential equations that define the Lorenz Strange Attractor problem:

\[\begin{align*}
x' &= \sigma (y - x) \\
y' &= -xz + Rx - y \\
z' &= xy - Bz
\end{align*}\]

where \(x', y'\) and \(z'\) are \(dx/dt; dy/dt\) and \(dz/dt\) where \(t\) is time; \(\sigma\) is Prandtl numbers, \(R\) is the ratio of Rayleigh numbers and \(B\) is geometric factor.

Your program should take as input values for \(\sigma, R\) and \(B\); the initial values of \(x, y,\) and \(z;\) the length of time for the solution and the accuracy of the solution. The accuracy can be specified as relative or absolute error. (Absolute error is the absolute difference between your solution and the real solution; relative error is the absolute error divided by the typical size of the variable. The relative error is not dependent on the units of \(x,y,z).\)

Your solution should explain how you solved the differential equations and the method to used to achieve the desired accuracy of the solution.

(a) To test your program: Output at 1 second intervals, the solution using:
- \(\sigma = 10;\) \(R= 40;\) and \(B=8/3;\)
- initial values of \(x=0,\) \(y=-10,\) and \(z=0;\)
- Integration should be for 20-seconds. The accuracy should be errors of order \(10^{-4}.\) Take advantage of Matlab’s ordinary differential equation solvers for this problem.

(b) Generate a 3-D plot of your results, and time series plots of \(x,\) \(y\) and \(z.\)

**Question 3 Solution**

This solution is based on the ODE45 solver, which seems to generate the most accurate results in the least amount time. The matlab solution is much slower than the Fortran variable step size run and seems to generate a similar quality result.

An inputdlg call is used to input values and default values can is easily included.
The accuracy is achieved using a combination of RelTol and AbsTol in the odeset function. The RelTol is limited a 2.210-14 and once this level is reached the value is not decreased below this level. During each iteration a figure is updated showing the differences between the solutions on a log plot. The one for the last iteration is shown below.

Graphics, included an animated sequence is output by the program.

Matlab code is given in HW05Q03.m and HW05Q3_deriv.m
Where the second m-file is the one that computes the derivatives and whose name is passed through the ODE45 call.

>> HW05Q03
Results for the fortran solutions are given below. The solutions differ slightly but are very close in general.

% time ..//HW02_soln/HW02Q03b 0 -10 0 20 1 "" "" R 1e-4
+ 12.010 HW 02 Q 03: Initial Conditions at time 0.00000 seconds
* XYZ values are 0.000000 -10.000000 0.000000
* Parameters: Prandtl # 10.000 Rayleigh Ratio 40.000 Geometry
2.666667
* Error Tolerance 0.100E-03
There are 21 saved values
.
<table>
<thead>
<tr>
<th>Time</th>
<th>X Value</th>
<th>Y Value</th>
<th>Z Value</th>
<th>Time Increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>1.137610</td>
</tr>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>2.275110</td>
</tr>
<tr>
<td>2</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.0000</td>
<td>3.165275</td>
</tr>
<tr>
<td>3</td>
<td>3.0000</td>
<td>3.0000</td>
<td>3.0000</td>
<td>4.092097</td>
</tr>
<tr>
<td>4</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.0000</td>
<td>4.999984</td>
</tr>
<tr>
<td>5</td>
<td>5.0000</td>
<td>5.0000</td>
<td>5.0000</td>
<td>5.894895</td>
</tr>
<tr>
<td>6</td>
<td>6.0000</td>
<td>6.0000</td>
<td>6.0000</td>
<td>6.785196</td>
</tr>
<tr>
<td>7</td>
<td>7.0000</td>
<td>7.0000</td>
<td>7.0000</td>
<td>7.675501</td>
</tr>
<tr>
<td>8</td>
<td>8.0000</td>
<td>8.0000</td>
<td>8.0000</td>
<td>8.565807</td>
</tr>
<tr>
<td>9</td>
<td>9.0000</td>
<td>9.0000</td>
<td>9.0000</td>
<td>9.456113</td>
</tr>
<tr>
<td>10</td>
<td>10.0000</td>
<td>10.0000</td>
<td>10.0000</td>
<td>10.346419</td>
</tr>
<tr>
<td>11</td>
<td>11.0000</td>
<td>11.0000</td>
<td>11.0000</td>
<td>11.236725</td>
</tr>
<tr>
<td>12</td>
<td>12.0000</td>
<td>12.0000</td>
<td>12.0000</td>
<td>12.127031</td>
</tr>
<tr>
<td>13</td>
<td>13.0000</td>
<td>13.0000</td>
<td>13.0000</td>
<td>13.017337</td>
</tr>
<tr>
<td>14</td>
<td>14.0000</td>
<td>14.0000</td>
<td>14.0000</td>
<td>13.907643</td>
</tr>
<tr>
<td>15</td>
<td>15.0000</td>
<td>15.0000</td>
<td>15.0000</td>
<td>14.797949</td>
</tr>
<tr>
<td>16</td>
<td>16.0000</td>
<td>16.0000</td>
<td>16.0000</td>
<td>15.688255</td>
</tr>
<tr>
<td>17</td>
<td>17.0000</td>
<td>17.0000</td>
<td>17.0000</td>
<td>16.578561</td>
</tr>
<tr>
<td>18</td>
<td>18.0000</td>
<td>18.0000</td>
<td>18.0000</td>
<td>17.468867</td>
</tr>
<tr>
<td>19</td>
<td>19.0000</td>
<td>19.0000</td>
<td>19.0000</td>
<td>18.359173</td>
</tr>
<tr>
<td>20</td>
<td>20.0000</td>
<td>20.0000</td>
<td>20.0000</td>
<td>19.249479</td>
</tr>
</tbody>
</table>

Graphics output.

Difference between last two iterations.

![Graph](image-url)
Solution as a time series. (NOTE this figure was saved as .eps from matlab and then converted to PDF with preview and inserted as figure. Iter 11 RelTol 2.22e−14 AbsTol 1.00e−16 RMS difference 1.6574e−04

The version below is the .eps file directly inserted into the document.
Solution in 3D

Iter 11 RelTol  2.22e−14 AbsTol  1.00e−16 RMS difference  1.6574e−04
Iter 11 RelTol 2.22e-14 AbsTol 1.00e-16 RMS difference 1.6574e-04