Submission Date: Monday 25th November 2019, 11pm **Assessed Coursework 1**

Your results to the assessed coursework must be submitted using this template. Please cut and paste the subsequent output into the correct parts of this file. Once this template has been completed, you must then create a pdf file for submission. Under Windows 10/Windows 7 use Texmaker 5; this may be accessed as follows:

`Start > UoN Applications > (UoN) Texmaker 5`

Open this file under `File`*; to build the pdf file, click the arrow next to* `Quick Build`*; this will then generate the file course_work1_submission.pdf.*
A single zip or tar file containing your solution should be submitted *through the module webpage on Moodle.* **NOTE:** *All parameters and values (such as polynomial degrees etc.) should be set within your codes: do NOT use inputs such as obtained with* `std::cin`*.*

File checklist:
course_work2_submission.pdf
input.dat
mat.cpp
fun.cpp
q1a.cpp
q1b.cpp
q2a.cpp
q2b.cpp
q2c.cpp
q3.cpp

1(b) Enter your output here:

```
The matrix entries vector is: ( 5 3 2 3 1 2 -1 1 1 )
The column number vector is: ( 1 3 4 2 4 4 1 2 4 )
The row start vector is: ( 1 4 6 7 10 )
```

1(b) Enter your output here:

```
Ax is: ( 33 0 -3 -7 )
```

2(a) Enter your output here:

```
u(x) evaluated at uniform points: ( -1.22515e-016 -0.317971 -0.506931 -0.603024
-0.632121 -0.603024 -0.506931 -0.317971 0 0.466214 1.02811 1.51904 1.71828
1.51904 1.02811 0.466214 )

u'(x) evaluated at uniform points: ( -3.14159 -1.97956 -1.09532 -0.477259
7.07656e-017 0.477259 1.09532 1.97956 3.14159 4.25562 4.50534 3.02848
5.22891e-016 -3.02848 -4.50534 -4.25562 )
```

2(b) Enter your output here:

```
The exact derivative is: ( -1.412 -0.375054 0.375054 1.412 3.14159 4.57488
2.51287 -2.51287 -4.57488 -3.14159 )

The approximate derivative is: ( -1.53417 -0.423059 0.423059 1.53417 3.1111
3.97111 1.97111 -1.97111 -3.97111 -3.1111 )

The CPU time used is: 0ms

The CPU time is given as 0, I have implemented the clock function as described
online, I believe there could be a problem with the operating system I am
running codeblocks on. It appears the clicks aren't being counted properly.
```

2(c) Enter your output here:

```
The exact derivative is: ( -3.14159 -3.61835 -4.56038 -2.23916 4.0491 3.14159
0.777099 -0.326828 -1.47423 -2.66367 -3.14159 )

The Chebyshev differentiation matrix gives: ( -3.36358 -3.49682 -4.71528 -2.03564
3.82998 3.3139 0.659923 -0.243906 -1.53898 -2.60775 -3.24811 )

The CPU time used is: 0ms

Again the CPU time is given as 0 when using the clock function.
```

2(d) Enter your output here:

```
I was unable to find a value of n that the tolerance would be satisfied
for the finite difference calculation.

The tolerance of 10^-9 was met at n = 294693.

The CPU time used is: 0ms

The error is: 9.98896e-010, it occurs at x = 0.696647


The Chebyshev differentiation matrix gives the correct tolerance at n = 39.

The CPU time used is: 0ms

The error is: 5.35713e-011, it occurs at x = -0.354605


These results make it clear that the Chebyshev differentiation matrix is
much more effective at estimating the derivative. The Chebyshev
```

```
differentiation matrix gets to the tolerance 10^-9 at n = 36, this
n value compared to the finite difference n it is clearly much smaller.
```

2(e) Enter your output here:

```
Finite difference:
The alpha values are: ( -0.433862 1.86199 1.85714 1.98806 )

Chebyshev:
The alpha values are: ( 3.19307 6.98381 16.3665 17.3517 )

The finite difference alpha values appear to tend to 2, this matches with
the theory as the method should converge quadratically which is of order 2.

The Chebyshev alpha values appear to increase rapidly, the convergence is
exponential and so it is expected that the alpha values wouldn't
converge. The size of the alpha values do show the rapid convergence
of the Chebyshev differentiation method.
```

3 Enter your output here:

```
When m decreases the solution at the final time becomes unstable, this
suggests for each n there is a minimum m value for which the method converges.

n=10,m=2550:

The u value for x = 0 at time T = 10 is 6.17233e-012


n=10,m=2500:

The u value for x = 0 at time T = 10 is -2.85948e+010


I created a function that uses a while loop to check whether the method
is stable for the given n and m, the while loop increased by the m value
by one each time until the method converges for the given n. I then
calculated the h values. This method gives these values:

n = 4, m = 98, hmin = 0.292893
n = 5, m = 198, hmin = 0.190983
n = 6, m = 373, hmin = 0.133975
n = 7, m = 650, hmin = 0.0990311
n = 8, m = 1069, hmin = 0.0761205
n = 9, m = 1674, hmin = 0.0603074
n = 10, m = 2513, hmin = 0.0489435
n = 11, m = 3238, hmin = 0.040507
n = 12, m = 5112, hmin = 0.0340742
```

```
n = 13, m = 6994, hmin = 0.0290582
n = 14, m = 9367, hmin = 0.0250721
n = 15, m = 12991, hmin = 0.0218524
```

This gives this plot (where k = 10/m, i.e. the time step size), it is clear the relationship between k and hmin is something like k = A*hmin + B (A and B are constants) for hmin > 0.15, i.e. the relationship is linear, for hmin < 0.15 the line tends towards the origin as a curve (i.e. no longer linear).