Assignment 3

2 November, 2023
Due date: 15 November

Instructions

- Submit to Avenue a **PDF file** containing your solutions.
  A file in another format e.g. IMG, PNG, will be ignored.
- Name your PDF file **Lastname-Firstname-studentnumber.pdf**.
- Name your MATLAB files **exactly** as specified.
- Submit your MATLAB code to Avenue and also include it in your PDF.
- Include your numerical results and plots in the PDF.
- Do not submit compressed (.zip, .rar) files.
  We will **ignore any compressed file** containing your files.

On using generative AI. Your are not permitted to use generative AI for this assignment. In alignment with McMaster academic integrity policy, it "shall be an offense knowingly to ... submit academic work for assessment that was purchased or acquired from another source". This includes work created by generative AI tools. Also state in the policy is the following, "Contract Cheating is the act of "outsourcing of student work to third parties" (Lancaster & Clarke, 2016, p. 639) with or without payment." Using Generative AI tools is a form of contract cheating. Charges of academic dishonesty will be brought forward to the Office of Academic Integrity.

Problem 1 [6 points]

a. [2 points] Implement the function

```
function [q, nfun] = adsimpson(f, a, b, tol)
```

It applies adaptive Simpson to approximate $\int_a^b f(x)dx$ with tolerance `tol`. It returns the approximation in `q` and the number of function evaluations in `nfun`. Set the depth of the recursion to 1000. `f` is a function handle for evaluating $f(x)$.

b. [4 points] Using `adsimpson`, implement

```
function q = dsimpson(f, a, b, c, d, tol)
```

It computes an approximation for $\int_c^d \left(\int_a^b f(x,y)dx \right) dy$ by calling `adsimpson`. `f(x,y)` is a function handle for evaluating $f(x,y)$.

`dsimpson` must not call any Matlab functions and must not use any loops.

Then run the script

```
fun = @(x,y) 1./( sqrt(x + y) .* (1 + x + y).^2 );
a = 0.1; b = 1; c = 0.1; d = 2;
tol = 1e-10;
qs = dsimpson(fun,a,b,c,d,tol);
q = integral2(fun,a,b,c,d, 'AbsTol',1e-14);
fprintf('dsimpson %20.16e\n', qs);
fprintf('integral2  %20.16e\n', q);
fprintf('|dsimpson-integral2|=%.2e\n', abs(q-qs))
```

.

Submit the output in the PDF.

Problem 2 [4 points] The period of a simple pendulum is determined by the complete elliptic integral of the first kind

$$K(x) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - x^2 \sin^2 \theta}}.$$

Use your `adsimpson` to evaluate this integral for enough values of x to draw a smooth plot of $K(x)$ over the range $-.99 \le x \le .99$.

Write a Matlab program `pendulum.m` that plots

a. $K(x)$ versus x.

b. the number of function evaluations versus x.

Explain the behaviour of the number of function evaluations versus x.

Problem 3 [12 points] Implement the composite trapezoidal rule in file `trapezoid.m` and the composite Simpson's rule in file `simpson.m`.

a. [2 points] Using the error formula for the trapezoidal rule find the smallest number of subintervals n required to approximate

$$\int_0^{\pi/2} e^x \cos(x) dx$$

with absolute error of at most tol $= 10^{-4}$.

b. [2 points] Determine experimentally the smallest value of n for which the actual error (exact integral - approximation) is $\leq$ tol.

c. [1 point] Using `loglog`, plot on the same plot the bound for the error versus n and the actual error versus n for $n = 2 : 200$.

d. [1 point] Explain the shape of the curves in this plot.

Repeat a.-c. with the composite Simpson's rule. In c., use $n = 2 : 2 : 200$.

For b. and c., name your Matlab program `errors.m`.

Problem 4 [4 points] Consider adding two $n \times n$ matrices using the functions below.

```
function C = addR(A, B)
    [n, ~] = size(A);
    for i = 1:n
        C(i, :) = A(i, :) + B(i, :);
    end
end
function C = addC(A, B)
    [n, ~] = size(A);
    for j = 1:n
        C(:, j) = A(:, j) + B(:, j);
    end
end
```

(a) [4 points] Write a MATLAB program `timeadd.m` that times their execution for matrices of size $n = 500 : 100 : 1500$. You can use the `timef` function in `timef.m`.

Theoretically, adding two $n \times n$ matrices is $O(n^2)$. Your program should find values for c and k in cn^k using least squares and output these values.

Denote the constants you compute for `addR` by c_{row} and k_{row}, and for `addC` by c_{col} and k_{col}. Your `timeadd.m` should output these constants and also plot (on the same plot) the CPU time for `addR` and `addC` versus n using `loglog`.

Bonus In (b), my `timeadd` computes k_{row} close to 3, which does not make sense. That is an $O(n^2)$ algorithm runs in cubic time.

[6 points] If you are also getting $k_{\text{row}} \approx 3$, explain why $k_{\text{row}} \approx 3$.

Problem 5 [2 points] Find an equation of the form $y = ae^{x^2} + bx^3$ that best fits

x	-1	0	1
y	0	1	2

in the least squares sense.

Problem 6 [4 points] The function $F(l) = k(l - l_0)$ is the force required to stretch a spring l units, where the constant $l_0 = 5.3$ is the length of the unstretched spring.

a. Given the measurements

l	7	9.4	12.3
$F(l)$	2	4	6

determine k using least squares.

b. Additional measurements are made giving the data

l	8.3	11.3	14.4	15.9
$F(l)$	3	5	8	10

Determine k from these data.

Which of the fits in (a) and (b) fits best the whole data

l	7	9.4	12.3	8.3	11.3	14.4	15.9
$F(l)$	2	4	6	3	5	8	10

Give sufficient detail supporting your claim.