MA40171 Assessed Coursework 1 — 2022/23

Set: Thu 10th Nov 2022.
Due: Thu 24th Nov 2022, 17.00.
Submission: Please submit a **single zip file** at the Moodle submission point. The zip file should contain **a written report** and **all MATLAB files** required for running your code. Your submission needs to **adhere to the following specifications**, which you should read very carefully. Failure to adhere to the specifications will lead you to losing overall marks (the maximum marks deductable are written in red against each specification, e.g. "[−1] " for the zip file name specification.)

- The zip file should be named "`assignment1.zip`". [−1]
- The zip file should be 'flat', i.e. have no folders or subfolders inside it. For example, the MATLAB files should **not** be put inside a subfolder within the zip file. [−1]
- The report should be a **single PDF file** with the name "`report.pdf`". [−2]
- The first page of the report should be the completed coursework cover sheet (available on the Moodle page). [−1]
- The report should only contain solutions to the questions marked as **[written]** in the assignment. The answers *may* include plots if explicitly asked in the question. The report should **not** contain any solutions to questions marked as **[code]**. In particular code printouts or snippets should **not** be included in the report. [−2]
- The answer to each question (e.g. Q1, Q2, . . .) should start on a **new page**. [−1]
- Answers should be presented in the order that the questions appear in the assignment. [−1]
- Your code should generate **no printed output** on MATLAB's `Command Window`. [−2]

Because this assignment requires electronic submissions, unfortunately it cannot be marked anonymously. Therefore you should please **put your name** on the cover sheet. A blank cover sheet can be downloaded from the course Moodle page.

Marks: This assignment is worth 13% of the mark for the unit. Figures in square brackets, e.g. [2], represent the marks available for each question, out of a total of 25.

Important:

1. **The work you hand in should be done by yourself without collaboration with others. Cheating is a serious offence.**
2. The file name and function specifications in the programming questions need to be followed **very precisely**. I will run multiple tests, and if your code is slow or inefficient, or fails due to **incorrect filenames, function names, incorrect order of parameters, wrong dimensions of inputs or outputs** etc., **you will lose marks** for the failed tests.
3. If exceptional circumstances, e.g. disability or illness, prevent you from completing this assignment by the deadline, make sure you follow the procedures at

   `https://moodle.bath.ac.uk/course/view.php?id=1757&section=2#support2`

   and contact the DoS `maths-dos@bath.ac.uk` **as soon as possible**.
4. Partial answers to questions will receive partial credit, particularly if they are well justified.
5. Your codes should be written such that it works with a recent version of MATLAB (R2018, R2019 or R2020) or MATLAB Online.
6. I hope you will understand that I cannot answer individual questions from students about the assignment (e.g. by email), since that would be unfair to other students who are doing the assignment without consultation.
7. You may use any results proved in lectures. If you do, please quote them clearly.
8. When programming in MATLAB do not use the word `eps` as the name of a variable. This is because this means something specific in MATLAB.

This assignment is about the numerical solution of the system of ODEs in $\mathbb{C}^d$ (where $d \in \mathbb{N}$):

$$\frac{\mathrm{d}\mathbf{y}(t)}{\mathrm{d}t} = \mathrm{i}H\mathbf{y}(t), \quad t \in \mathbb{R} \setminus \{0\}, \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{C}^d. \tag{$\star$}$$

Here i is the complex unit ($\mathrm{i}^2 = -1$) and H is a complex matrix which has the property,

$$H^* = H, \qquad H \in \mathbb{C}^{d\times d}.$$

The *adjoint* of a matrix $A \in \mathbb{C}^{n_1\times n_2}$ is defined as the transpose of the complex conjugate, $A^* := \overline{A}^T \in \mathbb{C}^{n_2\times n_1}$ and it obeys the relation $(AB)^* = B^*A^*$ for any $A \in \mathbb{C}^{n_1\times n_2}$ and $B \in \mathbb{C}^{n_2\times n_3}$. A vector $\mathbf{a} \in \mathbb{C}^d$ can also be thought of as a column matrix $\mathbf{a} \in \mathbb{C}^{d\times 1}$, and its adjoint is a row matrix $\mathbf{a}^* \in \mathbb{C}^{1\times d}$.

The solution of ($\star$) will be denoted by $\mathbf{y}(t)$ throughout this assignment, where $\mathbf{y} : \mathbb{R} \to \mathbb{C}^d$. An approximation to the true solution $\mathbf{y}(t)$ will be computed by numerical methods over a finite interval $[0, T]$ for a specified $T \in \mathbb{R}_{>0}$, using a finite number of time steps $N \in \mathbb{N}$ such that the time-step size is $h = T/N \in \mathbb{R}_{>0}$. $\mathbf{U}_n \in \mathbb{C}^d$ will denote the computed approximation to $\mathbf{y}(t_n)$, for each $n \in \{0, 1, \ldots, N\}$, where $t_n = nh$ and $t_N = Nh = T$.

Throughout this assignment we will use the notation $\mathbb{N} = \{1, 2, 3, \ldots\}$ for the set of natural numbers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ for the set of non-negative integers, $\mathbb{R}_{>0} = \{s \in \mathbb{R} : s > 0\}$ for positive real numbers, and $\mathbb{R}_{\geq 0} = \{s \in \mathbb{R} : s \geq 0\}$ for non-negative real numbers.

1. The ℓ^2 norm of a vector $\mathbf{v} \in \mathbb{C}^d$ is defined as

$$\|\mathbf{v}\|_2 = (\mathbf{v}^*\mathbf{v})^{\frac{1}{2}}.$$

(a) [written] Prove that $Q(\mathbf{v}) := \mathbf{v}^*H\mathbf{v}$ is real-valued for any $\mathbf{v} \in \mathbb{C}^d$.

(b) [written] Show that $\mathbf{y}(t)^*$ solves the system of ODEs

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = -\mathrm{i}\mathbf{x}(t)H, \quad t \in \mathbb{R} \setminus \{0\}, \qquad \mathbf{x}(0) = \mathbf{y}_0^*.$$

(c) [written] Prove that $\|\mathbf{y}(t)\|_2 = \|\mathbf{y}(0)\|_2$, for all $t \in \mathbb{R}$.

(d) [written] Prove that $Q(\mathbf{y}(t)) = Q(\mathbf{y}(0))$, for all $t \in \mathbb{R}$, where $Q(\mathbf{v})$ was defined in Q1(a).

(e) [written] Show that solution of the system of ODEs ($\star$) is

$$\mathbf{y}(t) = \exp(\mathrm{i}tH)\mathbf{y}_0,$$

where the exponential of a square matrix $A \in \mathbb{C}^{d\times d}$, $d \in \mathbb{N}$, is defined as

$$\exp(A) := \sum_{k=0}^{\infty} \frac{A^k}{k!}.$$

2. [code] Implement a function called `LMMcoefficients` in the file `LMMcoefficients.m` to compute the coefficients $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_q)$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_q)$ that define the q-step linear multistep method (LMM) of the form

$$\frac{1}{h}\sum_{j=0}^{q} \alpha_j \mathbf{U}_{n+j} = \beta_0 \mathbf{f}(\mathbf{U}_n) + \beta_q \mathbf{f}(\mathbf{U}_{n+q}), \qquad (\clubsuit)$$

with the **highest-order**. The coefficients α and β should specify the LMM in the **standard form**. Your function should have the following syntax:

```
[alpha, beta] = LMMcoefficients(q, beta0)
```

where `q` $\in \mathbb{N}$ and `beta0` is the value of the coefficient $\beta_0 \in \mathbb{R}$. The outputs `alpha` and `beta` should specify the row vectors of coefficients $\alpha, \beta \in \mathbb{R}^{q+1}$, such that the j th components of `alpha` and `beta` store α_{j-1} and β_{j-1}, respectively, for $j = 1, \ldots, q+1$. **[3]**

3. [code] Consider an arbitrary q-step ($q \in \mathbb{N}$) linear multistep method (LMM),

$$\frac{1}{h}\sum_{j=0}^{q} \alpha_j \mathbf{U}_{n+j} = \sum_{j=0}^{q} \beta_j \mathbf{f}(\mathbf{U}_{n+j}). \qquad (\spadesuit)$$

Implement a function called `LMMsolve` in the file `LMMsolve.m` to compute the numerical solution of ($\star$) until (and including) $t_N = Nh = T$ using the above LMM. Your function should have the following syntax:

```
[Ut, normt, Qt, t] = LMMsolve(alpha, beta, H, Ustartingvalues, T, N)
```

where `alpha` and `beta` are row vectors of length $q+1$ that specify the vector of coefficients $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_q) \in \mathbb{R}^{q+1}$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_q) \in \mathbb{R}^{q+1}$ which define the LMM in ($\spadesuit$). i.e. the j th components of `alpha` and `beta` store α_{j-1} and β_{j-1}, respectively, for $j = 1, \ldots, q+1$. The input `H` is the $d \times d$ complex matrix ($d \in \mathbb{N}$) that defines the system of ODEs ($\star$), `Ustartingvalues` is a matrix of dimensions $d \times q$ whose $j+1$ th column contains the starting value $\mathbf{U}_j$ ($j = 0, \ldots, q-1$), `T` is the final time ($T \in \mathbb{R}_{>0}$) and `N` $\in \mathbb{N}$ is the number of steps (you should assume $N > q$). The time step $h \in \mathbb{R}_{>0}$ is computed as $h = T/N$.

The output `Ut` should be a matrix of dimensions $d \times (N+1)$, where the first q columns are identical to `Ustartingvalues`, and the $n+1$ th column stores the numerical solution $\mathbf{U}_n \approx \mathbf{y}(t_n)$ for $n = q, \ldots, N$, produced using the LMM in ($\spadesuit$). The outputs `normt`, `Qt` and `t` should be row vectors of length $N+1$ each (i.e. have the shape $1 \times (N+1)$). The j th entry of `normt`, `Qt` and `t` should be $\|\mathbf{U}_{j-1}\|_2$, $Q(\mathbf{U}_{j-1})$, and t_j, respectively, for $j = 1, \ldots, N+1$, where Q and the ℓ^2 norm $\|\cdot\|_2$ were defined in Q1.

Your code will be tested for efficiency in two separate regimes:

(i) when d is small to moderate (50 to 500), H is a fully dense matrix, but the number of time steps is very large $N \gg d$ and

(ii) when d is very large (> 500), H is a sparse matrix, and N is not too large ($N < d$).

Your code will also be tested separately for *explicit* LMMs and *implicit* LMMs, and it should be efficient for both. You can have different code routes within `LMMsolve` for handling these cases. For instance, if the matrix `H` is sparse your code could take a route that is efficient for case (ii). Otherwise, by default, your code should take a route efficient for case (i). You can use the MATLAB function `issparse` to check if the matrix `H` is sparse or not. **[8]**

4. Recall from Q1(e) that the true solution of $(\star)$ is given by the matrix exponential, $\mathbf{y}(t) = \exp(\mathrm{i}tH)\mathbf{y}_0$. In MATLAB this can be computed using the matrix exponential function `expm`. For example, the solution $\mathbf{y}(t)$ at time $t \in \mathbb{R}$ is obtained using the following syntax:

```
yt = expm(1i*t*H)*y0;
```

(a) **[code]** Write a function `startingvalues` in the file `startingvalues.m` with the following syntax,

```
Ustartingvalues = startingvalues(q, H, U0, h)
```

which produces exact starting values for a q-step LMM using `expm`. Here `q` $\in \mathbb{N}$ is the number of steps of the q-step LMM, `H` is the $d \times d$ complex matrix ($d \in \mathbb{N}$) that defines the system of ODEs $(\star)$, `U0` is a d length column vector containing the initial value $\mathbf{U}_0$, and `h` $\in \mathbb{R}_{>0}$ is the time step size. The output `Ustartingvalues` should be a matrix of dimensions $d \times q$ whose $j+1$ th column contains the starting value $\mathbf{U}_j := \exp(\mathrm{i}t_j H)\mathbf{U}_0$, $j = 0, \ldots, q-1$.

(b) **[code]** Solve the system of ODEs $(\star)$ for $t \in [0, T]$, where $T = 5$,

$$H = \begin{pmatrix} 1 & 4 & 5 \\ 4 & 2 & 6 \\ 5 & 6 & 3 \end{pmatrix}, \qquad \mathbf{y}_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

with $N = 500$ time-steps, using the following methods

- **FE:** *forward Euler* method, i.e. Adams–Bashforth for $q = 1$.
- **BE:** *backward Euler* method, i.e. Adams–Moulton for $q = 1$.
- **TR:** *trapezoidal rule*, i.e. Adams–Moultons for $q = 2$.
- **LMM:** the LMM of the form $(\clubsuit)$ with $q = 4$, $\beta_0 = 0.5$. You should use your code from Q2 to find the coefficients `alpha` and `beta`. You should also use the function `startingvalues` which you implemented in Q4(a).

Your code for computing the above solutions should be contained in a MATLAB *script file* `plots.m` which should generate **exactly two figures** (using MATLAB's `figure` command).

- The first figure should be titled "change in norm" (using MATLAB's `title` command) and should contain a plot for each of the four methods showing the absolute change in ℓ^2 norm, $|\|\mathbf{U}_n\|_2 - \|\mathbf{U}_0\|_2|$, against time, t_n, for $n = 0, \ldots, N$.
- The second figure should be titled "change in Q" and should contain a plot for each of the four methods showing the absolute change in Q, $|Q(\mathbf{U}_n) - Q(\mathbf{U}_0)|$, against time, t_n, for $n = 0, \ldots, N$.

Each figure should have a legend (produced using MATLAB's `legend` command) with each method's plot labeled appropriately (i.e. "FE", "BE", "TR" and "LMM"). The y-axis for both plots should be scaled logarithmically (use MATLAB's `semilogy` command instead of `plot`). The x-axis should be labeled "time" (using MATLAB's `xlabel` command).

(c) **[written]** For the example described in Q4(b), which of the four methods is most accurate in terms of $\|\mathbf{U}_N - \mathbf{y}(T)\|_2$? Which method is most accurate in terms of conserving the ℓ^2 norm $|\|\mathbf{U}_N\|_2 - \|\mathbf{U}_0\|_2|$?

(d) **[written]** Identify the method in Q4(b) that seems to conserve the value of Q (i.e. $Q(\mathbf{U}_n) = Q(\mathbf{U}_0)$ for $n = 0, \ldots, N$) up to machine precision (roughly 10^{-15}) and prove that it does conserve Q.

[**Hint:** You may use without proof that $(A^{-1})^* = (A^*)^{-1}$ and $f(A)g(A) = g(A)f(A)$ for all functions $f, g : \mathbb{C}^{d\times d} \to \mathbb{C}^{d\times d}$ of a square matrix $A \in \mathbb{C}^{d\times d}$, $d \in \mathbb{N}$. Note $A^0 = I$ is the identity matrix.]