Polytechnique Montréal
Department of Mechanical Engineering
Course Number : **MEC6215**
Course Name : **Méthodes Numériques en Ingénierie**
Professor : **Sébastien Leclaire**
Homework : **Ordinary Differential Equations (20%)**
Date of Delivery : **October 6th 2021 23:55**

Pedagogical Guidelines

- Submit a concise report of 10 pages maximum on Moodle (excluding first page, table of contents, and references), letter format, no narrow fonts, minimum 12pt, and margin 1.9cm;
- Submit your code that can reproduce your numerical results;
- This work must be done in a team of two people;
- All data (report and code) must be submitted in a single zip file "YOURMATRICULE1-YOURMATRICULE2.zip" (less than 100MB);
- Your report must be perfectly understandable when printed in black and white.
- Prepare a maximum of 6 "PowerPoint" slides (required for the final presentation).

Objective

- Learn the basics of programming solvers adapted to ODEs with initial and boundary values.

1 Lobatto quadrature

The integration points of the Lobatto quadrature formulas are given by the solution of the equation:

$$\frac{d^{s-2}}{dx^{s-2}}\left(x^{s-1}(x-1)^{s-1}\right) = 0$$

where s indicates the number of integration points. These methods are of order $2s-2$.

Use this information to construct a Runge-Kutta formula of order 4. Represent the results as a Butcher table.

2 Computation of three-dimensional streamlines

Let's consider the following three-dimensional system of ODEs:

$$\frac{D\mathbf{x}_i}{Dt} = \mathbf{u}(\mathbf{x}_i, t) \tag{1}$$

$$\mathbf{x}_i(t=0) = \mathbf{x}_i^0 \tag{2}$$

where "i" is the index of a massless particle, $\mathbf{x}_i$ and $\mathbf{u}(\mathbf{x}_i, t)$ are respectively its instantaneous position and velocity in the three-dimensional space $\mathbf{x} = [x, y, z] \in \mathbb{R}^3$ and time $t \in \mathbb{R}^+$. Also, $\mathbf{x}_i^0$ is the initial position of the particle at time $t = 0$. This system of ODEs is useful when it is needed to evaluate the streamlines of a three-dimensional flow with velocity $\mathbf{u}(\mathbf{x}, t)$. Beside, if more equations are added to this system in order to model the acceleration of the particles, it could model the transport of particles in a flow under certain conditions. So, this system of ODEs is a base system to many models existing in engineering.

This exercise consists in programming different ODE solvers and verifying their implementations. For simplicity, the analytical velocity field, $\mathbf{u}(\mathbf{x}, t) = [u_x(\mathbf{x}, t), u_y(\mathbf{x}, t), u_z(\mathbf{x}, t)]$, is taken as follows:

$$u_x(\mathbf{x}, t) = x \sin(\omega_x t) \exp(-t) \tag{3}$$

$$u_y(\mathbf{x}, t) = y \sin(\omega_y t) \exp(-t) \tag{4}$$

$$u_z(\mathbf{x}, t) = z \sin(\omega_z t) \exp(-t) \tag{5}$$

with $\omega_x = \pi/3$, $\omega_y = 2\pi/3$ and $\omega_z = \pi$.

Use the MS Visual Studio solution **"MAIN_STREAMLINE3D.sln"** already provided in the homework statement. The code in the homework statement provides functions to compute the analytical solution of this system of ODEs and proceeds as follows:

- First, the parameter **nParticle**=100000 is used in order to write a program that could support the resolution of many particles at once;
- The initial position of the particles are randomly initialize in a unit box, i.e. $\mathbf{x}_i^0 \in [0, 1] \times [0, 1] \times [0, 1]$, using a predetermined seed.
- The constant timestep **dt** of the solver is chosen by the user.
- It should be notice that the steady state velocity field is $\mathbf{u}(t \to \infty) = \mathbf{0}$. Because of that, each particles converge to a final steady state position. The final analytical position of the particles will serve as a reference to verify the implementation of the solvers.
- In the main time marching loop, the analytical positions are computed at each time step using an analytical solver, until the position of all particles does not change to near machine accuracy. The stopping criterion use the L1 norm difference between the matrix of positions $\mathbf{x}^{n+1}$ and the same matrix $\mathbf{x}^n$ but from the last time step:

  $$||\mathbf{x}^{n+1} - \mathbf{x}^n||_{\text{L1}} \equiv \sum_i \left(|x_i^{n+1} - x_i^n| + |y_i^{n+1} - y_i^n| + |z_i^{n+1} - z_i^n|\right) \tag{6}$$

  This condition is evaluated every 50 time steps to reduce its computational cost. Indeed, computing this criterion required sending data to the host when using a GPU and this is a blocking operation with ArrayFire.
- The steady state numerical positions $\mathbf{x}^{\text{sn}}$ that were computed in the main time marching loop are then compared to the steady state analytical positions $\mathbf{x}^{\text{sa}}$. The L2 norm is used to evaluate the error:

  $$||\mathbf{x}^{\text{sn}} - \mathbf{x}^{\text{sa}}||_{\text{L2}} \equiv \left(\sum_i \left((x_i^{\text{sn}} - x_i^{\text{sa}})^2 + (y_i^{\text{sn}} - y_i^{\text{sa}})^2 + (z_i^{\text{sn}} - z_i^{\text{sa}})^2\right)\right)^{1/2} \tag{7}$$

  Obviously, the error obtained is close to machine zero when using the default analytical solver in the main time marching loop.

Instead of using the default analytical solver present in the main time marching loop, program and use the following numerical solvers:

1. Forward Euler scheme;
2. Second Order Runge-Kutta scheme (RK2 aka midpoint);
3. Fourth Order Runge-Kutta scheme (RK4); and
4. Two-step Adams-Bashforth scheme with an initial half time step and a forward Euler's predictor.

Complete the Excel file "streamline3D.xls" to obtain the order of accuracy of each schemes. Insert those Excel tables into your report document. Compare and discuss your results against the analytical solution. Also, compare and discuss the performance and the accuracy of each of the implementations. In particular, if you have access to an OpenCL or CUDA backend, please use it.

3 Shooting Method

Use the MS Visual Studio solution **"MAIN_SHOOTING_METHOD.sln"** already provided in the homework statement. Solve the following problem with the help of the *shooting method* for which the analytical solution is $y(x) = x^2 + 16/x$. Use the $RK4$ method. Report the error and comment on your results.

$$y'' = \frac{1}{8}(32 + 2x^3 - yy')$$
$$y(1) = 17, \quad y(3) = 43/3$$