CS 205b / CME 306

Application Track

Homework 2

1. **ALE** An Eulerian formulation of conservation of mass uses control volumes that are fixed in space as material flows freely through the control volumes. A Lagrangian formulations uses control volumes that move with the material, so that material never flows into our out of the control volume. An ALE (Arbitrary Lagrangian Eulerian) formulation is somewhere between these two. Control volumes move around with a velocity $\mathbf{v}$, as material flows freely through them. The material has density ρ and velocity $\mathbf{u}$. The velocity field $\mathbf{v}$ of the observer and the velocity field $\mathbf{u}$ of the material being observed are independent and may vary in both space and time. In particular, control volumes can move around and change shape over time.

   (a) Adapt the derivation of the weak form for conservation of mass to the ALE case, where the control volume itself also moves around based on a velocity field $\mathbf{v}$.

   The amount of mass in a control volume Ω is the same as in the Eulerian case:

   $$mass = \int_\Omega \rho \, dV.$$

   As before, any change in this mass is due to material leaving and entering the control volume. Let $\mathbf{n}$ be the outward-facing (unit) normal of a small patch of the boundary $\partial\Omega$. The motion of the fluid with respect to the control volume is $\mathbf{u}-\mathbf{v}$, so $\mathbf{n}\cdot(\mathbf{u}-\mathbf{v})$ is the rate of movement of fluid across the boundary of the control volume at any point (with positive indicating movement out of the volume). If the surface patch has area A, then the rate of mass flow across the boundary patch is $A(\mathbf{n}\cdot(\mathbf{u}-\mathbf{v}))\rho = \rho(\mathbf{u}-\mathbf{v})\cdot dS$, where dS is the surface element. This leads to the statement of mass conservation

   $$\frac{\partial}{\partial t}\int_\Omega \rho \, dV = -\int_{\partial\Omega} \rho(\mathbf{u}-\mathbf{v})\cdot dS.$$

   (b) Show that in the special case that $\mathbf{v}=\mathbf{0}$, the weak form derived for the Eulerian case is recovered.

   With $\mathbf{v}=\mathbf{0}$, we get

   $$\frac{\partial}{\partial t}\int_\Omega \rho \, dV = -\int_{\partial\Omega} \rho\mathbf{u}\cdot dS,$$

   which is what was obtained in the Eulerian case.

(c) Show what equation is obtained in the special case that $\mathbf{v} = \mathbf{u}$. Give a physical explanation for why this equation corresponds to a Lagrangian formulation of conservation of mass in weak form.

If $\mathbf{v} = \mathbf{u}$, then the ALE weak form simplifies to

$$\frac{\partial}{\partial t}\int_\Omega \rho\, dV = 0,$$

which can be integrated to

$$\int_\Omega \rho\, dV = constant.$$

This just states that the mass in a control volume does not change (is a constant). Since the control volume moves with the mass in the Lagrangian formulation, the amount of mass in the control volume does not change, which is consistent with the equation.

(d) Convert the ALE weak form of conservation of mass into strong form. Be careful when moving the time derivative inside the integration. Show that this matches what was obtained using the Eulerian formulation.

$$\begin{aligned}
0 &= \frac{\partial}{\partial t}\int_\Omega \rho\, dV + \int_{\partial\Omega} \rho(\mathbf{u}-\mathbf{v})\cdot dS \\
&= \left(\int_\Omega \frac{\partial \rho}{\partial t}\, dV + \int_{\partial\Omega} \rho\mathbf{v}\cdot dS\right) + \left(\int_{\partial\Omega} \rho\mathbf{u}\cdot dS - \int_{\partial\Omega} \rho\mathbf{v}\cdot dS\right) \\
&= \int_\Omega \frac{\partial \rho}{\partial t}\, dV + \int_{\partial\Omega} \rho\mathbf{u}\cdot dS \\
&= \int_\Omega \frac{\partial \rho}{\partial t}\, dV + \int_\Omega \nabla\cdot(\rho\mathbf{u})\, dV \\
&= \int_\Omega \frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u})\, dV \\
0 &= \rho_t + \nabla\cdot(\rho\mathbf{u})
\end{aligned}$$

2. **Duhamel's Principle** Consider the two ordinary differential equations $x' = \lambda x$ and $y' = \lambda y + \gamma$, where x, y, λ, and γ are all complex numbers.

(a) Find analytic solutions to the recurrences $r_{n+1} = \alpha r_n$ and $s_{n+1} = \alpha s_n + \beta$. Be careful of special cases.

The first recurrence is readily apparent, $r_n = \alpha^n r_0$. In the case of s, make the substitution $s_n = u_n + \kappa$. Then we have $u_{n+1} + \kappa = \alpha(u_n + \kappa) + \beta$. or $u_{n+1} = \alpha u_n + (\alpha\kappa + \beta - \kappa)$.

If $\alpha \neq 1$, we can choose $\kappa = \frac{\beta}{1-\alpha}$, resulting in $u_{n+1} = \alpha u_n$. The solution to this is $u_n = \alpha^n u_0$. Then, $s_n - \kappa = \alpha^n(s_0 - \kappa)$, so that we have the analytic solution $s_n = \alpha^n(s_0 - \frac{\beta}{1-\alpha}) + \frac{\beta}{1-\alpha}$. If $\alpha = 1$, then the recurrence is $s_{n+1} = s_n + \beta$, from which the solution is just $s_n = s_0 + n\beta$.

(b) For which α and β is r_n bounded but s_n unbounded?

If $\alpha \neq 1$, both sequences are bounded precisely when α^n is bounded. When $\alpha = 1$, $r_n = r_n$ is bounded, but $s_n = s_0 + n\beta$ is bounded only if $\beta = 0$. Thus, the requested situation only occurs when $\alpha = 1$ and $\beta \neq 0$.

(c) For which α and β is s_n bounded but r_n unbounded?

This situation never occurs.

(d) Show that for trapezoid rule, the update rule for x_n has the same form as the recurrence r_n, and the update rule for y_n has the same form as recurrence s_n. Also show that the expressions for α are the same and do not depend on γ.

Trapezoid rule for x_n and y_n give

$$\begin{aligned}
x_{n+1} &= x_n + \frac{\Delta t}{2}(\lambda x_n + \lambda x_{n+1}) \\
\left(1 - \frac{\Delta t}{2}\lambda\right) x_{n+1} &= \left(1 + \frac{\Delta t}{2}\lambda\right) x_n \\
x_{n+1} &= \left(\frac{1 + \frac{\Delta t}{2}\lambda}{1 - \frac{\Delta t}{2}\lambda}\right) x_n \\
y_{n+1} &= y_n + \frac{\Delta t}{2}((\lambda y_n + \gamma) + (\lambda y_{n+1} + \gamma)) \\
\left(1 - \frac{\Delta t}{2}\lambda\right) y_{n+1} &= \left(1 + \frac{\Delta t}{2}\lambda\right) y_n + \Delta t \gamma \\
y_{n+1} &= \left(\frac{1 + \frac{\Delta t}{2}\lambda}{1 - \frac{\Delta t}{2}\lambda}\right) y_n + \Delta t \gamma
\end{aligned}$$

The two recurrences have the desired form, with

$$\alpha = \frac{1 + \frac{\Delta t}{2}\lambda}{1 - \frac{\Delta t}{2}\lambda} \qquad \beta = \Delta t \gamma,$$

which satisfies what was to be shown.

(e) What do you conclude about the dependence of the stability of trapezoid rule on the inhomogeneous term γ? Would the conclusion change much if backward Euler or forward Euler were being studied instead?

The stability only depends on γ when $\alpha = 1$, which in the case of trapezoid rule only occurs if $\Delta t\lambda = 0$. $\Delta t = 0$ is pointless and also results in $\beta = 0$, so stability differs only if lambda $= 0$. The stability of trapezoid rule does not depend on the inhomogeneous term as long as the homogeneous term is nonzero. Considering the different values of α that would have been obtained for forward and backward Euler, the same conclusion would be obtained for those as well.

3. **Plotting Stability** Note this problem contains a small programming component. The lecture notes show stability plots for forward Euler (FE), backward Euler (BE), trapezoid rule (TR), second order Runge-Kutta (RK2), third order Runge-Kutta (RK3), and fourth order Runge-Kutta (RK4). These plots were obtained by considering the equation $y' = \lambda y$, where λ is complex.

   (a) State the update rules for FE, BE, TR, and RK2 when applied to $y' = f(y)$.

$$\begin{aligned} y_{n+1} &= y_n + \Delta t f(y_n) && \textit{forward Euler} \\ y_{n+1} &= y_n + \Delta t f(y_{n+1}) && \textit{backward Euler} \\ y_{n+1} &= y_n + \frac{\Delta t}{2}(f(y_n) + f(y_{n+1})) && \textit{trapezoid rule} \\ y_{n+1} &= y_n + \Delta t f(y_n + \frac{1}{2}\Delta t f(y_n)) && \textit{second order Runge-Kutta} \end{aligned}$$

   (b) Let $f(y) = \lambda y$, so that the differential equation is $y' = \lambda y$. When solved for y_{n+1} in terms of y_n, the update rule should have the form $y_{n+1} = C y_n$, where C is a complex number that depends only on $\lambda \Delta t$. Find C for FE, BE, TR, and RK2.

   We already have effectively done this for trapezoid rule, and the process is similar for the others.

$$\begin{aligned} C &= 1 + \Delta t\lambda && \textit{forward Euler} \\ C &= \frac{1}{1 - \Delta t\lambda} && \textit{backward Euler} \\ C &= \frac{1 + \frac{1}{2}\Delta t\lambda}{1 - \frac{1}{2}\Delta t\lambda} && \textit{trapezoid rule} \\ C &= 1 + \Delta t\lambda + \frac{1}{2}(\Delta t\lambda)^2 && \textit{second order Runge-Kutta} \end{aligned}$$

   (c) What must be true of C for a method to be stable for a given choice of Δt and λ?

   We should have $|C| < 1$.

(d) Let $\Delta t = 1$ and sample the complex plain in the region $-3 \leq \mathrm{Re}(\lambda) \leq 3$ and $-3 \leq \mathrm{Im}(\lambda) \leq 3$, determining for each value of λ whether the scheme will be stable. Use white to indicate unstable and a distinct color (not black) to indicate stable. Add black axes to the images (two lines is fine), making sure that the axes are on top of everything else. It is recommended that you use matlab or octave for this assignment, though a solution using C++ and ImageMagick is also acceptable. For each of FE, BE, TR, and RK2, you should submit (on paper) the image obtained and the source code used to construct it (even if the four programs are nearly identical). The images should closely match the ones in the lecture notes.

Forward Euler

```
# Generate a grid of complex numbers L in the desired range
[x y] = meshgrid(-3:1/64:3);
L = x - y * I;

# Compute the value C (forward Euler)
C = 1 + L;

# white (1) = unstable (|C| >= 1), gray (1/2) = stable (|C| < 1)
M = 1-(abs(C) < 1)/2;

# Add black axes
N = max(0,M - (real(L)==0) - (real(I*L)==0));

# Display image
imshow(N);
```

Backward Euler

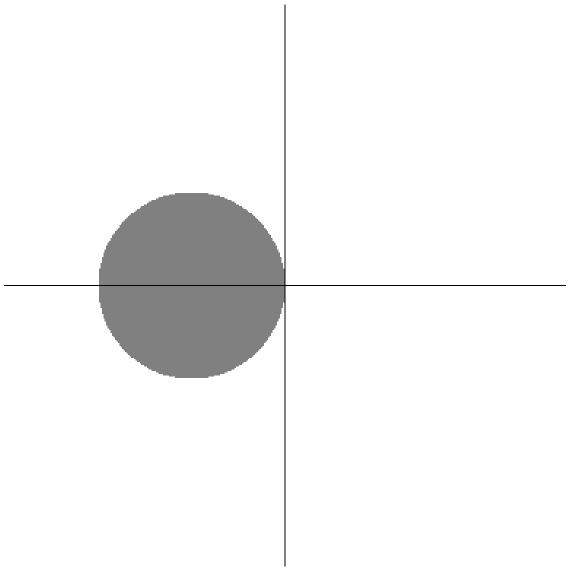
```
# Generate a grid of complex numbers L in the desired range
[x y] = meshgrid(-3:1/64:3);
L = x - y * I;

# Compute the value C (backward Euler)
C = 1 ./ (1 - L);

# white (1) = unstable (|C| >= 1), gray (1/2) = stable (|C| < 1)
M = 1-(abs(C) < 1)/2;

# Add black axes
N = max(0,M - (real(L)==0) - (real(I*L)==0));

# Display image
imshow(N);
```

Trapezoid Rule

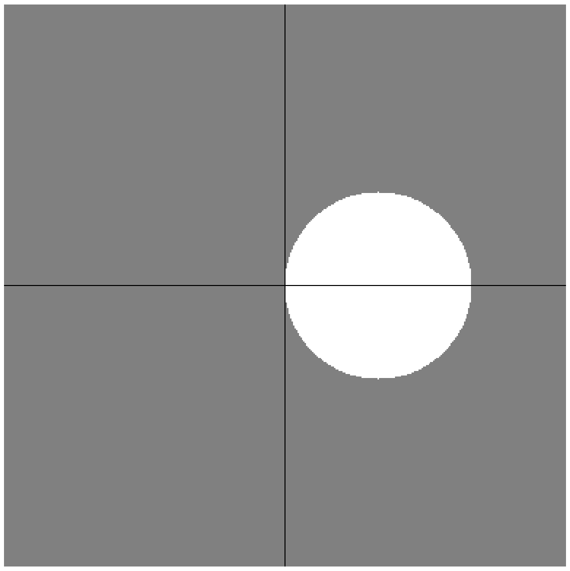
```
# Generate a grid of complex numbers L in the desired range
[x y] = meshgrid(-3:1/64:3);
L = x - y * I;

# Compute the value C (trapezoid rule)
C = (1 + L/2) ./ (1 - L/2);

# white (1) = unstable (|C| >= 1), gray (1/2) = stable (|C| < 1)
M = 1-(abs(C) < 1)/2;

# Add black axes
N = max(0,M - (real(L)==0) - (real(I*L)==0));

# Display image
imshow(N);
```

Second Order Runge-Kutta

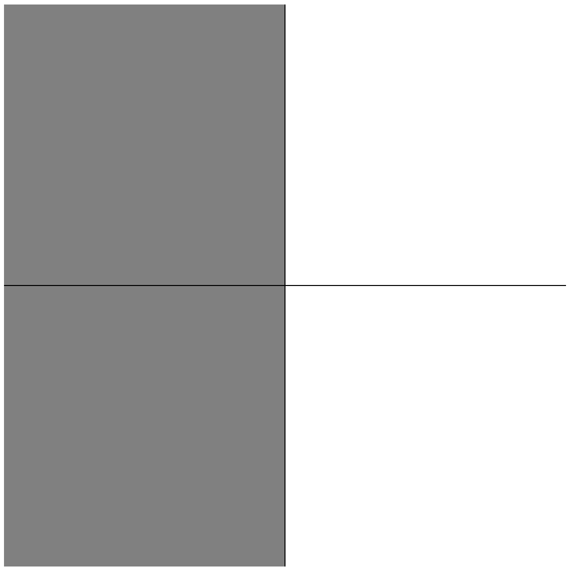
```
# Generate a grid of complex numbers L in the desired range
[x y] = meshgrid(-3:1/64:3);
L = x - y * I;

# Compute the value C (second order Runge-Kutta)
C = 1 + L + L.*L/2;

# white (1) = unstable (|C| >= 1), gray (1/2) = stable (|C| < 1)
M = 1-(abs(C) < 1)/2;

# Add black axes
N = max(0,M - (real(L)==0) - (real(I*L)==0));

# Display image
imshow(N);
```



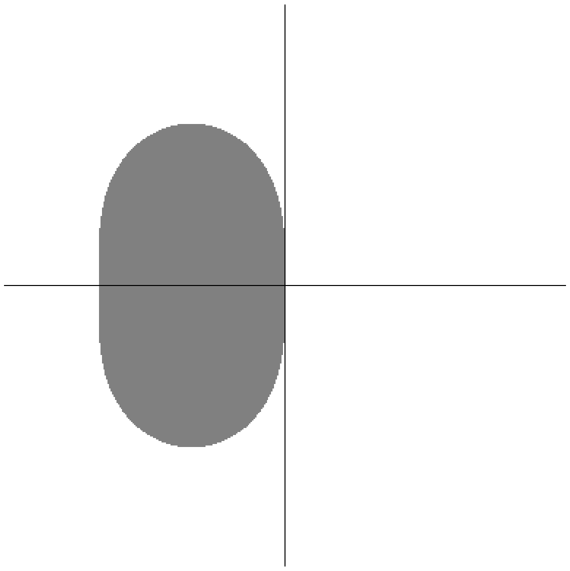
(a) Forward Euler



(b) Backward Euler



(c) Trapezoid Rule



(d) Second Order Runge-Kutta

Figure 1: Output of octave scripts.