Lecture 3

Time integration : implicit methods

D. Dickinson
d.dickinson@york.ac.uk

York Plasma Institute, School of PET, University of York

Semester 1

Overview of course

This course provides a brief overview of concepts relating to numerical methods for solving differential equations.
Topics to be covered include:

- Integrating ODEs
  - ~~Explicit techniques~~
  - Implicit techniques
  - Using Scipy to integrate ODEs
- Spatial discretisation
  - Finite differencing
  - Spectral methods
  - Finite elements
- Particle In Cell (PIC) approaches
- Continuum techniques

Notes available on the VLE.

Overview this lecture

This lecture will look at

- Limitations of explicit methods.
- Stability.
- Implicit integration techniques.

Limitations to explicit methods

Explicit methods are often a useful tool for solving a range of problems. However, to obtain an accurate solution it's important to make sure the time step is small enough. Look at exponential decay

Consider the simple problem below which has solution $y = y_0 \exp(-10t)$.

$$\frac{dy}{dt} = -10y$$

Limitations to explicit methods

Explicit methods are often a useful tool for solving a range of problems. However, to obtain an accurate solution it's important to make sure the time step is small enough.

Consider the simple problem below which has solution $y = y_0 \exp(-10t)$.

$$\frac{dy}{dt} = -10y$$

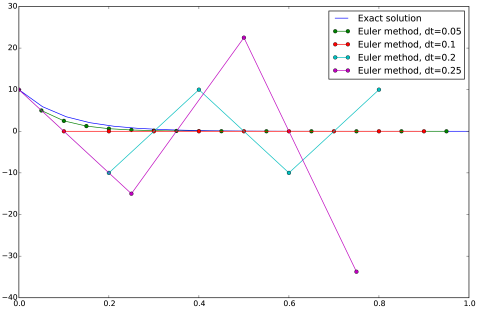

Figure: Numerical solution for various time steps, showing solution starts to oscilate and grow in magnitude for larger time steps.

- Figure shows Euler solution with different δt.
- $\delta t = 0.05$ we get decaying solution
- $\delta t = 0.1 - 0.2$ get oscillatory solution.
- $\delta t = 0.25$ solution grows

How can we work out when we'll get which kind of behaviour?

Limitations to explicit methods

Why did the solution blow up for large time step? Let's look at the discretised Euler problem:

$$\frac{y^{n+1} - y^n}{\delta t} = -10y^n \Rightarrow y^{n+1} = y^n \left(1 - 10\delta t\right)$$

So if $10\delta t > 2$ the magnitude of the solution will grow with time $\longrightarrow$ the method will be unstable, explaining the observed growth when $\delta t > 0.2$ and oscillation with $0.1 \leq \delta t \leq 0.2$ (where $1 - 10\delta t$ is -ve).

RHS=0 if dt is 0.1
This is why previous numerical solution would go straight to 0, dt was too big

If dt = 0.2, then the RHS is -ve, so we get an oscillating and growing solution

There is a limit on the time step we can take.
- But, it can be expensive to take small or wrong time step, so how can we get a better method?

Limitations to explicit methods

Why did the solution blow up for large time step? Let's look at the discretised Euler problem:

$$\frac{y^{n+1} - y^n}{\delta t} = -10y^n \Rightarrow y^{n+1} = y^n \left(1 - 10\delta t\right)$$

So if $10\delta t > 2$ the magnitude of the solution will grow with time $\longrightarrow$ the method will be unstable, explaining the observed growth when $\delta t > 0.2$ and oscillation with $0.1 \leq \delta t \leq 0.2$ (where $1 - 10\delta t$ is -ve).

Is there a way to avoid this?

- Use a different explicit method which has different stability properties.

Limitations to explicit methods

Why did the solution blow up for large time step? Let's look at the discretised Euler problem:

$$\frac{y^{n+1} - y^n}{\delta t} = -10y^n \Rightarrow y^{n+1} = y^n \left(1 - 10\delta t\right)$$

So if $10\delta t > 2$ the magnitude of the solution will grow with time $\longrightarrow$ the method will be unstable, explaining the observed growth when $\delta t > 0.2$ and oscillation with $0.1 \leq \delta t \leq 0.2$ (where $1 - 10\delta t$ is -ve).

Is there a way to avoid this?

- Use a different explicit method which has different stability properties.
  $\Rightarrow$ Still susceptible to the problem, just tweak maximum timestep.
- What if we change where we Taylor expand?

Limitations to explicit methods

Why did the solution blow up for large time step? Let's look at the discretised Euler problem:

$$\frac{y^{n+1} - y^n}{\delta t} = -10y^n \Rightarrow y^{n+1} = y^n\left(1 - 10\delta t\right)$$

So if $10\delta t > 2$ the magnitude of the solution will grow with time $\longrightarrow$ the method will be unstable, explaining the observed growth when $\delta t > 0.2$ and oscillation with $0.1 \leq \delta t \leq 0.2$ (where $1 - 10\delta t$ is -ve).

Is there a way to avoid this?

- Use a different explicit method which has different stability properties.
  $\Rightarrow$ Still susceptible to the problem, just tweak maximum timestep.
- What if we change where we Taylor expand?
  $\Rightarrow$ Let's investigate the implications of this.

Forward Taylor expansion
Backwards Taylor expansion

Implicit techniques

Express our solution at the nest time in terms of our solution at the next time. Can't write down all the terms until we have a solution.

Consider expanding around the next time point rather than the current one, the discretised problem is then

$$\frac{y^{n+1} - y^n}{\delta t} = -10y^{n+1} \Rightarrow y^{n+1} = \frac{y^n}{(1 + 10\delta t)}$$

This is known as the Backwards Euler method. It is an implicit method as the value at the next time point depends on the derivative there. Clearly for any positive non-zero value of δt we find that $\left|y^{n+1}\right| < |y^n|$ so the method is stable and as $1/\left(1 + 10\delta t\right) > 0$ the are no oscillating solutions.

Methods for which the solution of the exponentially decaying problem always tend to zero are known as A-stable.

No matter what time step we pick, the solution will always get smaller - eliminates problem of growing solution.

Doesn't always work well, if you pick a bad time step can still give bad approx. but won't blow up like explicit method

Implicit techniques

Always decays

Not always good approx.

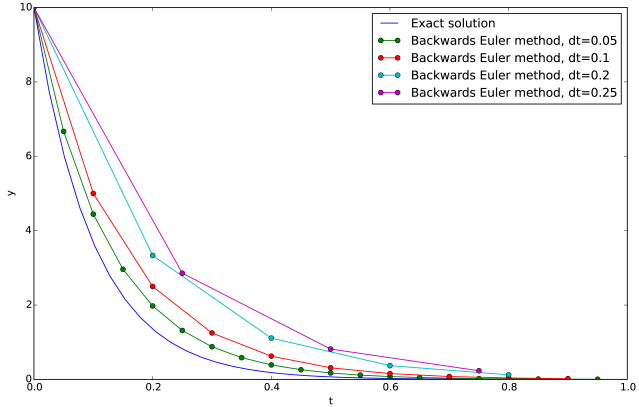

Figure: Numerical solution using implicit Euler method for various time steps. Shows no cases which oscilate or increase in magnitude.

The figure shows the solution of the exponentially decaying problem using the Backwards Euler method with different time steps. For the Python code see the VLE (or lecture).

Implicit techniques

Implicit methods are very useful for solving problems with a large range of timescales, often referred to as stiff. An example of such a problem is

in plasmas, different species react on different time scales

$$\frac{dy}{dt} = 998y + 1998g, \quad \frac{dg}{dt} = -999y - 1999g$$

which given $y(0) = 1$ and $g(0) = 0$ has solutions

$$y = 2\exp(-t) - \exp(-1000t), \quad g = -\exp(-t) + \exp(-1000t)$$

An explicit method would need a timestep small enough to resolve the rapid decay in the second term of each solution ($\sim \delta t < 1/1000$). Implicit methods would allow much larger timesteps ($\delta t \sim 1$) without losing the essential structure of the solution.

It's worth noting that many problems in plasma physics are stiff in nature.

Implicit techniques : Disadavantages

Because implicit methods have the solution at the next time point on both sides of the equation we must in general solve a system of (potentially nonlinear) equations (see the VLE notes). In practice relatively advanced techniques may be required to do this and the computation involved may outweigh that involved with evaluating the derivative function $F(\boldsymbol{y})$.

Implicit methods are more expensive

Implicit techniques : Disadavantages

Because implicit methods have the solution at the next time point on both sides of the equation we must in general solve a system of (potentially nonlinear) equations (see the VLE notes). In practice relatively advanced techniques may be required to do this and the computation involved may outweigh that involved with evaluating the derivative function $F(\boldsymbol{y})$.

Whilst implicit methods can allow a larger timestep relative to explicit schemes the cost of each timestep is larger.

Implicit techniques : Disadavantages

Because implicit methods have the solution at the next time point on both sides of the equation we must in general solve a system of (potentially nonlinear) equations (see the VLE notes). In practice relatively advanced techniques may be required to do this and the computation involved may outweigh that involved with evaluating the derivative function $F(\boldsymbol{y})$.

Whilst implicit methods can allow a larger timestep relative to explicit schemes the cost of each timestep is larger.

Whilst implicit methods are stable with large time steps we must still ensure that the time step doesn't exceed that of the timescale for the important physical processes.

Implicit techniques : Disadavantages

Because implicit methods have the solution at the next time point on both sides of the equation we must in general solve a system of (potentially nonlinear) equations (see the VLE notes). In practice relatively advanced techniques may be required to do this and the computation involved may outweigh that involved with evaluating the derivative function $F(\boldsymbol{y})$.

Whilst implicit methods can allow a larger timestep relative to explicit schemes the cost of each timestep is larger.

Whilst implicit methods are stable with large time steps we must still ensure that the time step doesn't exceed that of the timescale for the important physical processes. Stiff problems - different time scales

unphysical solutions: blowing up, changing sign

It is easy to tell with explicit methods when the time step is too large as the solution blows up, with implicit methods it is much harder.

With implicit methods, it is harder to tell when a timestep is too big because it doesn't blow up.

Summary

The choice between using explicit or implicit methods will depend on the type of problem you're solving.

- If you're interested in the fastest timescale in your system then an explicit method is likely to be the best.
- If your (discretised) equations contain rapid oscillations which don't impact on the final solution then an implicit technique may be better.

It's not always clear what timescales are involved with a given system of discretised equations; may need analysis and testing to determine what approach is the best.

Summary

The choice between using explicit or implicit methods will depend on the type of problem you're solving.

- If you're interested in the fastest timescale in your system then an explicit method is likely to be the best. Small time steps. Because explicit is less costly and can handle lots of tiny time steps easier
- If your (discretised) equations contain rapid oscillations which don't impact on the final solution then an implicit technique may be better. You don't care about the fastest time scales

It's not always clear what timescales are involved with a given system of discretised equations; may need analysis and testing to determine what approach is the best.

It would be unfortunate if you spent a long time implementing a sophisticated implicit method, only to find the time step is limited to the explicit value! Thankfully it's recommended that usually rather than implementing a method yourself you use an external library. This usually makes it very easy to test out different methods. We'll see an example of this in the next lecture when we look at SciPy's solve_ivp function. External library can decide for you which method to use, e.g. scipy