Lecture 2

Time integration : explicit 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

- Generic ODE form.
- Taylor expansion as a starting point.
- The Euler method
- Practical implementation

The generic ODE form

This lecture course will deal pretty much exclusively with solving equations of the form

$$\frac{\partial y}{\partial t} = F\left(y, t\right) \tag{1}$$

which is a first order ODE.

You might think this is of limited use, what if you're not dealing with a first order ODE?

It's possible to write most equations you're likely to need to integrate in this form.

This includes both Nth order ODEs as well as PDEs (which we'll see later).

The generic ODE form

This lecture course will deal pretty much exclusively with solving equations of the form

$$\frac{\partial y}{\partial t} = F\left(y, t\right) \tag{1}$$

which is a first order ODE.

You might think this is of limited use, what if you're not dealing with a first order ODE?

It's possible to write most equations you're likely to need to integrate in this form.

This includes both Nth order ODEs as well as PDEs (which we'll see later).

So how do we do this?

$\Rightarrow$ Write y as a (n-dimensional) vector $\boldsymbol{y}$.

Because it isn't always just "y"

The generic ODE form

Replacing y with $\boldsymbol{y}$, we have

$$\frac{\partial \boldsymbol{y}}{\partial t} = F\left(\boldsymbol{y}, t\right) \qquad (2)$$

Now consider a 2nd order ODE: Simplifies 2nd order ODE by writing it as 1st order ODE

$$\frac{\partial^2 \alpha}{\partial t^2} = g\left(\alpha\right)$$

We can introduce an intermediate variable, β, such that

$$\frac{\partial \beta}{\partial t} = g\left(\alpha\right), \quad \frac{\partial \alpha}{\partial t} = \beta$$

a' = B
B' = g(a)

If we now define our vector, $\boldsymbol{y}$, as:

$$\boldsymbol{y} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \Rightarrow \frac{\partial \boldsymbol{y}}{\partial t} = \begin{pmatrix} \beta \\ g\left(\alpha\right) \end{pmatrix}$$

state vector

e.g.
F = ma = m d^2x/dt^2

The generic ODE form

Replacing y with $\boldsymbol{y}$, we have

$$\frac{\partial \boldsymbol{y}}{\partial t} = F\left(\boldsymbol{y}, t\right) \tag{2}$$

Now consider a 2nd order ODE:

$$\frac{\partial^2 \alpha}{\partial t^2} = g\left(\alpha\right)$$

We can introduce an intermediate variable, β, such that

$$\frac{\partial \beta}{\partial t} = g\left(\alpha\right), \quad \frac{\partial \alpha}{\partial t} = \beta$$

If we now define our vector, $\boldsymbol{y}$, as:

$$\boldsymbol{y} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \Rightarrow \frac{\partial \boldsymbol{y}}{\partial t} = \begin{pmatrix} \beta \\ g\left(\alpha\right) \end{pmatrix}$$

We've replaced our 2nd order ODE with two 1st order ODEs (written as a single vector problem). The same technique can be applied to Nth order ODEs to replace the problem with N 1st order ODEs.

Taylor expansion

Euler method - do not use! :D

Initial value problems

So now we know we want to solve for $\dot{\boldsymbol{y}}\left(t\right) = F\left(\boldsymbol{y}\right)$ how do we make progress? $\longrightarrow$ Consider evolving $\boldsymbol{y}$ from some initial state at $t = t_0$ to a small time later, $t = t_0 + \delta t \rightarrow$ Taylor expand[1]:

$$\boldsymbol{y}\left(t = t_0 + \delta t\right) = \boldsymbol{y}\left(t_0\right) + \delta t \left.\frac{\partial \boldsymbol{y}}{\partial t}\right|_{t_0} + \frac{\delta t^2}{2} \left.\frac{\partial^2 \boldsymbol{y}}{\partial t^2}\right|_{t_0} + \cdots$$

[1]By Taylor expanding we're assuming the function is continuously differentiable etc. This is not always true (e.g. shocks) and different techniques may be required.

Taylor expansion

So now we know we want to solve for $\dot{\boldsymbol{y}}(t) = F(\boldsymbol{y})$ how do we make progress? $\longrightarrow$ Consider evolving $\boldsymbol{y}$ from some initial state at $t = t_0$ to a small time later, $t = t_0 + \delta t \rightarrow$ Taylor expand[1]:

$$\boldsymbol{y}(t = t_0 + \delta t) = \boldsymbol{y}(t_0) + \delta t \left.\frac{\partial \boldsymbol{y}}{\partial t}\right|_{t_0} + \frac{\delta t^2}{2} \left.\frac{\partial^2 \boldsymbol{y}}{\partial t^2}\right|_{t_0} + \cdots$$

Taking just the first two terms we're left with

$$\boldsymbol{y}_1 = \boldsymbol{y}_0 + \delta t F(\boldsymbol{y}_0)$$

which gives us an expression for $\boldsymbol{y}$ at the next time point in terms of the value and its derivative, both evaluated at the current moment in time.

$\longrightarrow$ Can repeat this process to find $\boldsymbol{y}$ at all later times.

[1]By Taylor expanding we're assuming the function is continuously differentiable etc. This is not always true (e.g. shocks) and different techniques may be required.

Taylor expansion : Euler

Taking just these first two terms gives us what's known as Euler's method. It can be summarised as

- Find gradient at current time.

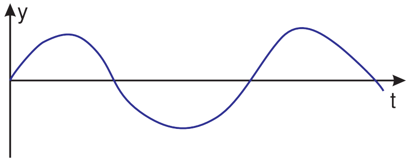

Figure: Sketch of some arbitrary function

Taylor expansion : Euler

Taking just these first two terms gives us what's known as Euler's method. It can be summarised as

- Find gradient at current time.
- Use gradient to linearly extrapolate to new time.

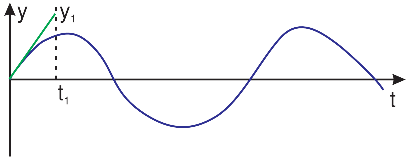

Figure: Sketch of some arbitrary function along with the gradient calculated at some point and extrapolated forwards by some step.

Taylor expansion : Euler

Taking just these first two terms gives us what's known as Euler's method. It can be summarised as

- Find gradient at current time.
- Use gradient to linearly extrapolate to new time.
- Repeat until $t \geq t_f$.

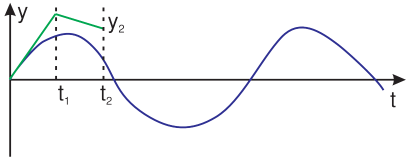

Figure: Sketch of some arbitrary function along with the gradient calculated at some later point and extrapolated forwards by another step.

It is very simple to understand and will be used in several demonstrations.

Taylor expansion : Euler

Taking just these first two terms gives us what's known as Euler's method. It can be summarised as

- Find gradient at current time.
- Use gradient to linearly extrapolate to new time.
- Repeat until $t \geq t_f$.

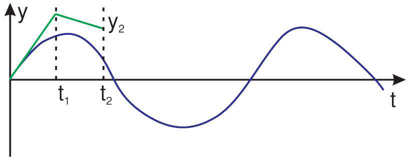

Figure: Sketch of some arbitrary function along with the gradient calculated at some later point and extrapolated forwards by another step.

It is very simple to understand and will be used in several demonstrations.

Unfortunately it's not very good in practice as it's neither very accurate or stable
$\rightarrow$ don't use it for anything serious!

Taylor expansion : Error/Order

The Euler method isn't very accurate, but how can we quantify the error? $\rightarrow$ Look at the terms we dropped in the Taylor expansion.

Small steps and small dt or error too large to bother

Assuming we're taking small steps, such that δt is small, we can see that the dominant error in our Taylor expansion comes from dropping the second derivative term, so we can estimate the error as

$$\epsilon = \frac{\delta t^2}{2} \left.\frac{\partial^2 y}{\partial t^2}\right|_{t_0} \propto \delta t^2$$

Approximate error by biggest piece (next term we dropped)

This doesn't look too bad $\rightarrow$ halve δt and we reduce ϵ by a factor four.

[2] Known as the local truncation error.

Taylor expansion : Error/Order

The Euler method isn't very accurate, but how can we quantify the error? $\rightarrow$ Look at the terms we dropped in the Taylor expansion.

Assuming we're taking small steps, such that δt is small, we can see that the dominant error in our Taylor expansion comes from dropping the second derivative term, so we can estimate the error as

$$\epsilon = \frac{\delta t^2}{2} \left.\frac{\partial^2 y}{\partial t^2}\right|_{t_0} \propto \delta t^2$$

This doesn't look too bad $\rightarrow$ halve δt and we reduce ϵ by a factor four. But this is only the error on one step[2], as we've reduced the time step we need to increase the number of steps so the total error is given by

total time / step size

$$\epsilon|_{t_f} \propto \frac{t_f}{\delta t}\delta t^2 = t_f \delta t$$

which scales linearly with δt, hence the Euler method is 1st order accurate.

So if you drop by factor of 2, error only drops by factor of 2 - error drops linearly with time step

[2] Known as the local truncation error.

Alternative schemes

Don't use Euler method :D

There are many alternatives to Euler which have better stability and accuracy properties. Some of these are described in the online notes.

A popular family of methods are known as Runge-Kutta methods, with the most popular of these known as the fourth order Runge-Kutta scheme or simply RK4. These are single step multi-stage methods[3].

Can take bigger steps while still having alright error.
Uses more computer memory and time than Euler

[3] The derivation of the RK schemes will not be covered directly here, but is based along the same lines as the Euler approach.There are many resources online which discuss this for the interested, such as this website describing the derivation of RK methods.

Alternative schemes

There are many alternatives to Euler which have better stability and accuracy properties. Some of these are described in the online notes.

A popular family of methods are known as Runge-Kutta methods, with the most popular of these known as the fourth order Runge-Kutta scheme or simply RK4. These are single step multi-stage methods[3].

Another class of techniques are known as multistep methods. These schemes use values at previous steps as well as the current one. This allows higher order terms in the Taylor expansion to be eliminated and hence lower error to be achieved.

Higher memory costs than RK4
Not self-starting -start of simulation has more error
Error can propagate

[3] The derivation of the RK schemes will not be covered directly here, but is based along the same lines as the Euler approach.There are many resources online which discuss this for the interested, such as this website describing the derivation of RK methods.

Alternative schemes

There are many alternatives to Euler which have better stability and accuracy properties. Some of these are described in the online notes.

A popular family of methods are known as Runge-Kutta methods, with the most popular of these known as the fourth order Runge-Kutta scheme or simply RK4. These are single step multi-stage methods[3].

Another class of techniques are known as multistep methods. These schemes use values at previous steps as well as the current one. This allows higher order terms in the Taylor expansion to be eliminated and hence lower error to be achieved.

In the next lecture we'll see another class of methods, known as implicit methods, which have quite different properties. But first let's look at some practical examples (see VLE for python files).

Taylor expands about the next step and looks backwards

[3]The derivation of the RK schemes will not be covered directly here, but is based along the same lines as the Euler approach.There are many resources online which discuss this for the interested, such as this website describing the derivation of RK methods.