Week 10: aim to cover

- Euler's method: derivation
- Euler's method: performance and error analysis
- Runge-Kutta methods

Solving differential equations

One of the most useful numerical techniques:
numerical solution of differential equations .
We do only Ordinary Differential Equations (ODEs)
$\Longrightarrow$ unknown functions $y_i(t)$ depend only on 1 dependent variable t.
Recall that any n^{th} order ODE can be written as a system of 1st order ODEs
$\Longrightarrow$ any system of ODEs can be written as a system of 1st order ODEs.

$$\frac{dy_i}{dt} = f_i(t, y_1, \cdots y_n) \quad i = 1..n$$

or in vector form

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y})$$

◁ **Example:**

Special cases

- If $\mathbf{f} = \mathbf{f}(\mathbf{y})$ only, system is **autonomous** (else **nonautonomous**)
- If $\mathbf{f} = \mathbf{A}(t)\mathbf{y} + \mathbf{b}(t)$, system is **linear**
- If $\mathbf{b} = \mathbf{0}$, linear system is **homogeneous**
- If $\mathbf{A}$ is constant, linear system is **constant-coefficient system**
- If only 1 ODE, equation is **scalar**

Initial Value problems

If the **initial conditions**

$$y_i(t_0) = \alpha_i$$

are given at a single value of t (e.g. $t_0 = 0$), we have an **Initial Value Problem**. Otherwise, we have a **Boundary Value Problem**.

We only cover Initial Value Problems (IVPs).

The methods used for a system are basically the same as those for a single 1st order ODE
$\Longrightarrow$ we lose little by discussing a **scalar ODE**

$$\frac{dy}{dt} = f(t, y) \text{ where } y(t_0) = y_0$$

Finding a numerical solution means :

find a set of values $\{y_k\}$ at some set of output points $\{t_k\}$.

We need to distinguish between the true value: $y(t_k)$
and the numerical approximation : y_k

We need numerical methods since most ODEs can't be solved analytically

Example

$y' = y^3 + t^2$ can't be done by Maple.

Local Existence and uniqueness

Theorem

If $f(t, y)$ is cts in $t \in [t_0, t_f]$ and $y \in \mathbb{R}$ **and** *satisfies a* **Lipschitz condition**

$$|\, f(t, y) - f(t, \hat{y}) \,| \leq L \,|\, y - \hat{y} \,|$$

$\implies$ *the IVP has a* **unique solution** *for $t \in [t_0, t_1]$, $t_1 \leq t_f$.*

If $\frac{\partial f}{\partial y}$ continuous $\implies$ satisfies Lipschitz condition, with $L = \sup |\frac{\partial f}{\partial y}|$

The Lipschitz property is stronger than continuity of $f(t, y)$ but weaker than the continuity of $\frac{\partial f}{\partial y}$

This theorem guarantees a unique solution, at least for a while after t_0.

We will assume f satisfies a Lifschitz condition.

Sensitivity of IVP

Suppose we change IC from y_0 to $y_0 + \epsilon$? What is change in solution, measured with the function ∞ norm?

$$||y_\epsilon(t) - y(t)||_\infty = \max |y_\epsilon(t) - y(t)|$$

Under the same assumptions as above, can show that

$$||y_\epsilon(t) - y(t)||_\infty \leq c\epsilon$$

where c is independent of ϵ.
If c is not too large, the IVP is well-conditioned; if $c \gg 1$ it is ill-conditioned.

Roughly, when nearby solutions are diverging rapidly from each other,

$$\frac{\partial f}{\partial y} \gg 1$$

the IVP is ill-conditioned: no numerical method will give accurate answers.
When nearby solutions approach each other,

$$\frac{\partial f}{\partial y} < 0$$

the IVP is well-conditioned: we aim for accurate answers in this case.

Example

for autonomous linear system
solutions approach each other if $\mathrm{Re}(\lambda_{max}(A)) < 0$
all eigenvalues lie in left half of complex plane

Warning: doesn't generalize immediately to nonautonomous or nonlinear cases!

Time-stepping

All methods start at $t = t_0$ using the Initial Condition $y = y_0$
then march a distance h in t
$t_0 \to t_0 + h \equiv t_1$
$y_0 \to y_1 \approx y(t_1)$
At $t = t_1$ we have another IVP,

$$\frac{dy}{dt} = f(t, y)$$
$$y(t_1) = y_1$$

so just repeat this procedure until $t = t_f$.
Called **time-stepping**.

Types of time-stepping

- If h is constant $\rightarrow$ **fixed step method**
- If h is changed from step to step $\rightarrow$ **variable step method**
- If, to get from t_n to t_{n+1} we only use y_n $\rightarrow$ **1 step method**
- If, to get from t_n to t_{n+1} we use previous values $y_{n-1}, y_{n-2}..$ $\rightarrow$ **multistep method**

We start with fixed step methods but modern codes are usually variable step. What method to choose depends on:

- discretization error
- stability properties of method
- efficiency e.g. number of function evaluationss
- ease of use

In MATLAB

it's very easy to solve an IVP

1. define the ODE $y' = f(t,y)$
   e.g. `myde = @(t,y) y.^2 + t.^3`   must be in order (t,y)!!
2. define the Initial Condition
   e.g. `y0 = 0.5`
3. then solve with one of MATLAB's solvers
   e.g. `[t,y] = ode45(myde,[0,1],y0);`
   MAGIC

plot the result

`plot(t,y)`

We now explore how this magic is performed ...

The simplest method: Euler's Method

Start with y_0. We know $y'(t_0) = f(t_0, y_0)$ so step a distance h with that slope:

$$y_1 = y_0 + hf(t_0, y_0)$$

Now repeat, using in general

Euler's Method

$$y_{n+1} = y_n + hf(t_n, y_n)$$

We are approximating the ODE by solving a **difference equation**
$\rightarrow$ **discretization error** is produced

Derivation using Taylor series

Taylor series $\rightarrow$ **local error** $\sim h^2$

this idea of matching with Taylor series leads to Runge-Kutta methods (1-step methods)

Derivation by approximating y'

Use Forward Difference to approximate y'
$\rightarrow$ **local error** $\sim h^2$ (as before)
This idea of approximating y' leads to BDF methods (multistep methods)

Derivation by quadrature

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(\tau, y(\tau))\, d\tau$$

use Left Hand rectangle rule! $\rightarrow$ **local error** $\sim h^2$

Approximating this integral leads to Adams methods (multistep methods)

The local error is a truncation error or discretization error

How does it perform?

- **Demo**

Here we know the exact answers so we plot the global error as a function of time (on a semilog plot) for three different problems, for 3 choices of h.

Some observations from the results...

1. sometimes the error grows with t, sometimes not

2. the **global error** $y(t_k) - y_k$ appears to be $\propto h$ (if the method works at all)

3. sometimes it blows up, if stepsize is too large

To understand these observations, we need some error analysis.

End of Lecture 19