Ordinary differential equation II

1 Some Basic Methods

1.1 Backward Euler method (implicit method)

The algorithm looks like this:

$$y_n = y_{n-1} + hf_n \tag{1}$$

In contrast to the explicit forward Euler method we need to solve a non-linear equation at each time step. Backward Euler is also of order 1 and O(h) convergent like forward Euler, but it is more stable than forward Euler.
Example: Logistic population growth function

$$N'(t) = rN(t)(1 - \frac{N(t)}{K}) \tag{2}$$

where r is the intrinsic growth rate, and K is the carrying capacity (maximum of individuals the environment will be able to support).

```
backeuler[a_, b_, h_, n0_] := Module[{i, K, r, xx = {}, res, t},
  res =  n0;
  K = 200;
  r = 0.1;
  t = (b - a)/h;
  For[i = 0, i < t , i++,
   (*Solve[rn ==  res + h  (r rn (1-rn/K)),rn]*)
   res = (-K + h K r + Sqrt[(-K + h K r)^2 + 4 h K r res])/(2 h r);
   AppendTo[xx, {h i, res}]
   ];
  xx
  ]

euler[a_, b_, h_, n0_] := Module[{K, r, i, xx = {}, res, t},
  res =  n0;
  K = 200;
  r = 0.1;
  t = (b - a)/h;
  For[i = 0, i < t , i++,
   res +=  h  (r res (1 - res/K));
   AppendTo[xx, {h i, res}]
   ];
  xx
  ]
```

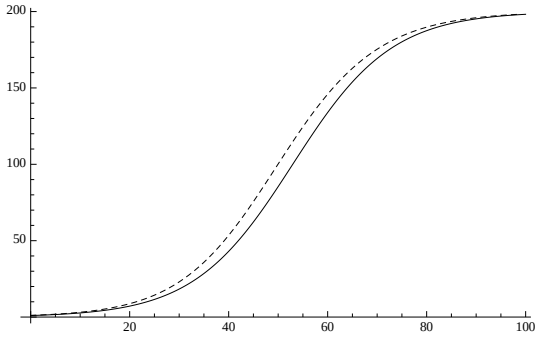

Figure 1: Comparison of solutions of the logistic population growth: dashed line is Backward Euler with $h = 0.1$ and also the analytical solution, solid line is the forward Euler with stepsize $h = 0.01$.

1.2 Midpoint method (explicit multistage method)

The midpoint method is sometimes called *modified Euler* method and is also an example of a Runge-Kutta method, one needs to do more work but is considerably more accurate than the Euler methods. It is also explicit and can be built up easily.

$$y_{n+1/2} = y_n + \frac{h}{2} f(t_n, y_n) \tag{3}$$

$$y_{n+1} = y_n + h f(t_{n+1/2}, y_{n+1/2}) \tag{4}$$

The midpoint method is a second order method: $O(h^2)$

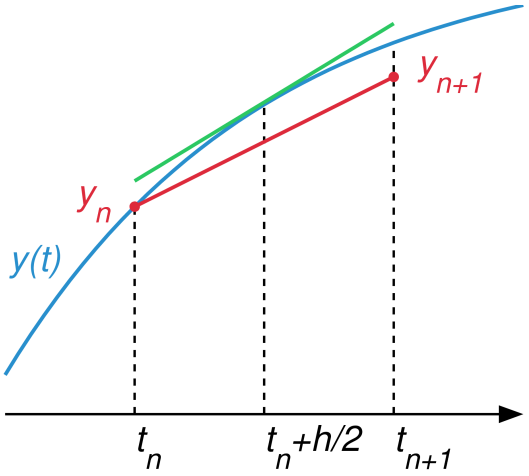

Figure 2: Illustration of the midpoint method (Wikipedia)

1.3 Heun's method (predictor-corrector)

Heun's method has several other names: modified Euler's method or the explicit trapezoidal rule. It can be seen as an extension of Euler's method into a two-stage second order Runge-Kutta method. It proceeds first to calculate an approximate $y_n + 1$ using forward Euler and then refines that estimate:

$$\tilde{y}_{n+1} = y_n + h f(t_n, y_n) \tag{5}$$

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1}) \tag{6}$$

Heun's method is a predictor-corrector method using the forward Euler's method as predictor and the trapezoidal method as the corrector.

2 Evaluation of solutions

We saw with the Euler method that we need to check our results. Without analytical results this is rather difficult and we need to develop a language to discuss such issues. We need to be able to explore the local truncation or discretization error (e.g. $h = 0.01$ and $h = 0.1$ in the Euler method).

We need to worry about the global error because we sum over larger numbers of small steps that propagate the local error.

- **Consistency**: Does the discrete approximation approach the exact solution as h goes to zero? (local discretization error)
- **Convergence**: Given consistency, how quickly does the discrete approximation approach the exact solution.
- **Stability**: How sensitive is the solution to perturbations in the initial conditions or the function?
- **Stiffness**

2.1 Consistency

Let $y(t_i)$ represent the **exact** solution to the discrete equations.
Let $Y(t_i)$ represent the exact solution to the IVP. Now ask whether $y(t_i)$ approach $Y(t_i)$ as h is reduced. We define a difference operator for a numerical method. For example for the Euler method we have

$$N_h(y_n) = \frac{y(t_n) - y(t_{n-1})}{h} - f(t_{n-1}, y_{n-1}) \tag{7}$$

Let $\mathbf{y} = \{y_0, y_1, ..., y_N\}$ and $\mathbf{Y} = \{Y_0, Y_1, ..., Y_N\}$ then

$$N_h(\mathbf{y}) = 0 \tag{8}$$
$$N_h(\mathbf{Y}) = d \neq 0 \tag{9}$$

For the forward Euler method we then can rephrase

$$N_h(y) = \frac{y_n - y_{n-1}}{h} - f_{n-1} \tag{10}$$

the local truncation error d is the residual, when the difference operator is applied to the exact solution

$$N_h(Y) = \frac{Y_n - Y_{n-1}}{h} - f_{n-1} = d \tag{11}$$

where d is the local truncation error. For a consistent method d goes to zero as h goes to zero and the discrete solution approaches the continuous solution.

2.2 Convergence

Example for consistency and convergence (local error): improvement of d using (11) on a very simply ODE. $y'(t) = \sin t$ with $y(0) = \frac{1}{2}$, its analytical is solution is $y(t) = 1\frac{1}{2} - \cos t$ Replacing the Y_n and Y_{n-1} with $y(t_n)$ and $y(t-h), respectively$and the f_{n-1} with $y'(t-h)$ we can look ad the effect of different h on d.
For an ODE over the interval $[a, b]$ when we reduce h the local error decreases, but the number of steps increases $N = (b-a)/h$, but for convergent methods the local error decreases at a faster rate than the global error increases, if convergent then

$$\lim_{h \to 0} \mathbf{y} = \mathbf{Y} \qquad Nh = (b - a) \tag{12}$$

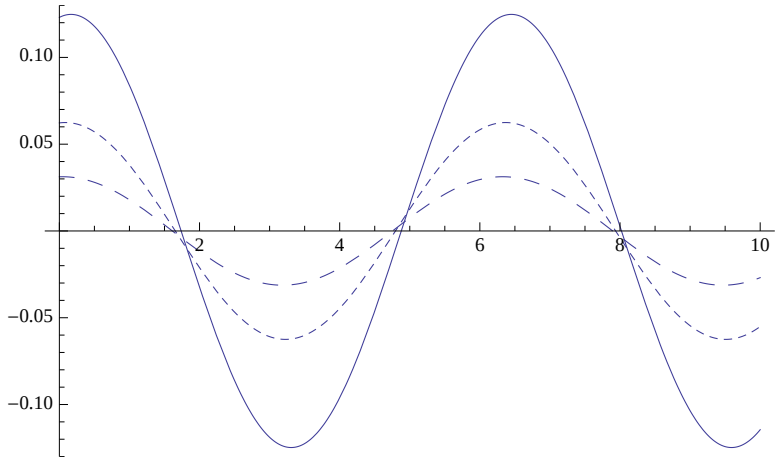

Figure 3: the difference between the true results and the Euler method for different stepsizes $h = \{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ (solid line, narrow dashing, wide dashing).)

Convergence describes the rate at which the discrete solutions approach the continuous (true) solution. The global error is

$$\epsilon_n = |y_n - Y(t_n)| \qquad \text{with} \qquad e_0 = 0 \tag{13}$$

a method N_h converges with order p, if for all n

$$\epsilon_n = O(h^p) \tag{14}$$

For the forward Euler method we can get the exact solution using a Taylor expansion around t_{n-1}

$$Y_n = Y_{n-1} + hY'(t_{n-1}) + \frac{h^2}{2!}Y''(\xi) \tag{15}$$

$$N_h(Y(t_n) = \frac{Y_n - Y_{n-1}}{h} - f_{n-1} = O(h) \tag{16}$$

where N_h is the local truncation error and the Euler method shows first order convergence

$$\mathbf{d} = O(h) \tag{17}$$

Consider a general linear problem over $t = [a, b]$

$$y'(t) = \lambda y(t) + g(t) \tag{18}$$

Applying Euler leads to

$$y_n = (1 + h\lambda)y_{n-1} + hg(t_{n-1}) \tag{19}$$

using the Taylor expansion (15) we get for this particular problem

$$Y_n = (1 + h\lambda)Y_{n-1} + hg(t_{n-1}) + \frac{h^2}{2!}Y''(\xi) \tag{20}$$

the last term is the local error and is equivalent to

$$hd_{n-1} \tag{21}$$

To calculate the global error we fill in

$$e_n = |Y_n - y_N| \tag{22}$$
$$= |(1 + h\lambda)e_{n-1} - hd_{n-1}| \tag{23}$$

This difference equation can be "solved"

$$e_n \leq (b - a)e^{|\lambda|(b-a)}O(h) \tag{24}$$

and for general non-linear problem, it can be shown that it is

$$e_n \leq Te^{LT}O(h) \tag{25}$$

For our simply problem $y'(t) = sin(t)$ we calculate

$$e_N = |e_{N-1} + hd_{n-1}| = |e_{N-1} + h\left[(Y(t) - Y(t-h))/h - \sin(t-h)\right]| \tag{26}$$

For $h = 1/4 : e_N = 0.81655$, $h = 1/8 : e_N = 0.409205$, $h = 1/16 : e_N = 0.204526$.

2.2.1 Summary

- Consistency relates to the size of the local error (hd_n)
- Convergence relates to the size of the global error which depends on the how local errors propagate
- All methods in the real-world are consistent and convergent: if you make h small enough you can solve the problem.

2.3 Absolute Stability

is based on a test equation with complex λ and real y_0

$$y' = \lambda y, \qquad y_0 \tag{27}$$

with the exact solution

$$y(t) = y_0 e^{\lambda t} \tag{28}$$

The behavior depends in values of λ and we can test how the numerical methods responds to this equation. The characteristics of the **exact** solution are:

$$Re(\lambda) < 0 \Rightarrow \lim_{t\to\infty} |y(t)| = 0 \qquad \text{(damped)} \tag{29}$$
$$Re(\lambda) = 0 \Rightarrow \lim_{t\to\infty} |y(t)| = y_0 \qquad \text{(oscillatory)} \tag{30}$$
$$Re(\lambda) > 0 \Rightarrow \lim_{t\to\infty} |y(t)| = \infty \qquad \text{(unbounded)} \tag{31}$$
$$\tag{32}$$

We would want that the numerical solution has the same characteristics, particularly for $\lambda \leq 0$. The region of absolute stability of a numerical solution is defined as the region on the complex plane defined by $h\lambda$, for which the solution the numerical scheme remains bounded. If it is bounded in the left-hand plane $Re(h\lambda) < 0$ the the methods is called **A-stable**. The right-hand plane behavior depends on the application: stable, unstable does not matter.

2.3.1 Stability of the presented methods

Forward Euler:
Testing the absolute stability with the test equation, forward Euler method is

$$y_n = y_{n-1} + h\lambda y_{n-1} \tag{33}$$

$$\frac{y_n}{y_{n-1}} = 1 + h\lambda \tag{34}$$

it is stable if

$$|1 + h\lambda| \leq 1, \tag{35}$$

this is a disk with unit radius centered at $(-1, 0)$, therefore the Forward Euler method **is not A-stable**.
Backward Euler:
Testing the absolute stability with the test equation, backward Euler method is

$$y_n = y_{n-1} + h\lambda y_n \tag{36}$$

$$\frac{y_n}{y_{n-1}} = \frac{1}{1 - h\lambda} \tag{37}$$

it is stable if

$$|1 + h\lambda| \geq 1, \tag{38}$$

this is the area outside a disk with unit radius centered at $(1, 0)$, therefore the Backward Euler method **is A-stable**

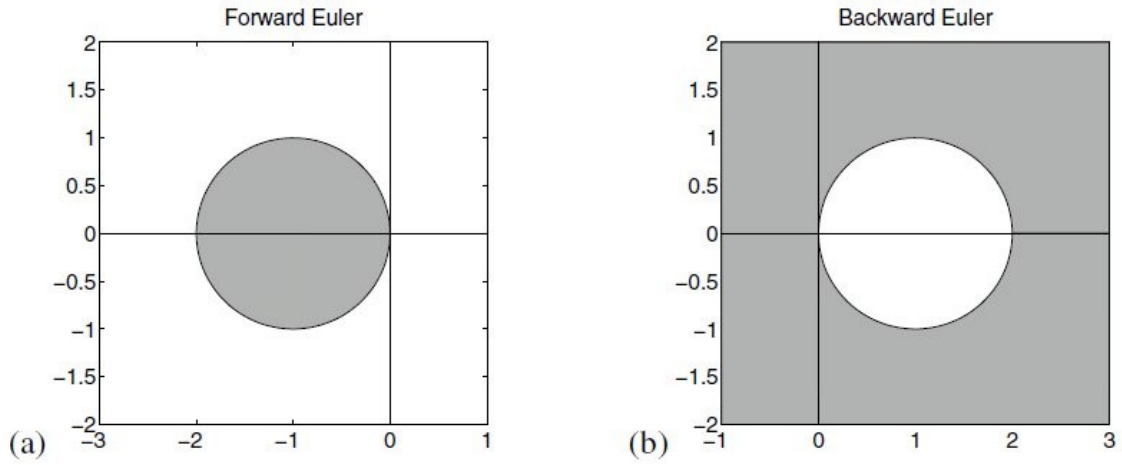

Figure 4: Stability regions for Forward and Backward Euler methods

2.4 Stiffness

Normally the step-size h required for accurate estimation is smaller than the step-size required for stability.

- accuracy: discrete solution approaches *true* continuous solution
- stability: discrete solution doesn't blow up

$$h_{\text{accuracy}} \ll h_{\text{stability}}$$

For stiff problems, the opposite is true. Stability is a bigger concern!

$$h_{\text{accuracy}} \gg h_{\text{stability}}$$

"A stiff equation is a differential equation for which certain numerical methods for solving the equation are numerically unstable, unless the step size is taken to be extremely small. It has proven difficult to formulate a precise definition of stiffness, but the main idea is that the equation includes some terms that can lead to rapid variation in the solution." (http://en.wikipedia.org/wiki/Stiff_equation)

Stiffness depends on:

- the initial value problem
- interval of integration – size and location
- accuracy requirements
- the region of absolute stability of the method

There is a competition between accuracy and $h_{\text{stability}}$. Usually, we are after a smooth and slowly varying solution, when perturbations of the data have rapidly varying solutions.
A simple example:

$$y'(t) = -\sin(t); \qquad y(0) = 1 \tag{39}$$

with solution

$$y(t) = \cos(t) \tag{40}$$

adding the solution back into the problem

$$y'(t) = \lambda[y - \cos(t)] - \sin(t); \qquad y(t_0) = y_0 \tag{41}$$

has the solution

$$y(t) = e^{\lambda(t-t_0)}[y_0 - \cos(t_0)] + \cos(t) \tag{42}$$

The term $e^{\lambda(t-t_0)}[y_0 - \cos(t_0)]$ is transient because it decays for $Re(\lambda) < 0$.
Perturbations occur naturally because of truncation errors in numerical solutions, Figure 5 shows such perturbations of the exact function. If we use that particular problem

$$y'(t) = \lambda[y - \cos(t)] - \sin(t); \qquad y(0) = 1; \quad \lambda = -10 \tag{43}$$

Solving this using the forward and backward Euler method using a $\lambda = -10$ and changing the stepsize h. I used these mathematica scripts to generate the plots in figure 6

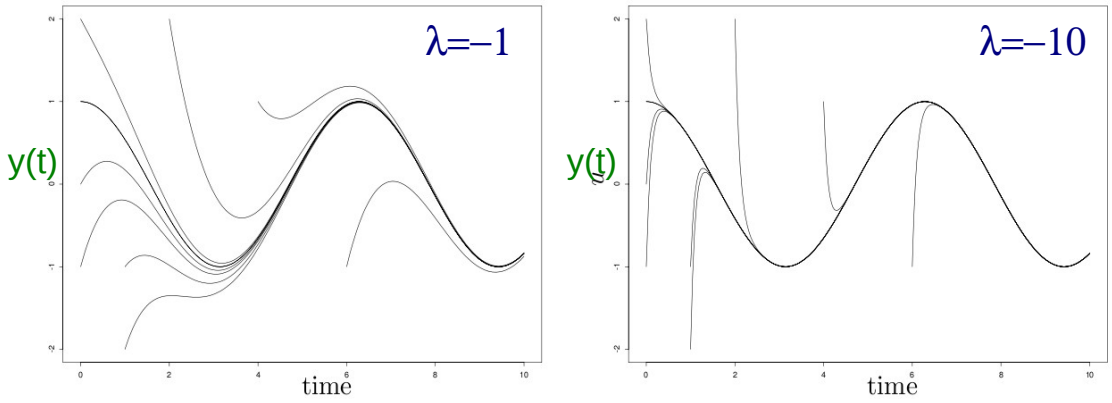

Figure 5: Initial value perturbations by different values of t_0 at different values for λ

```
euler[a_, b_, h_, \[Lambda]_] := Module[{i, xx = {}, res, t},
  res =  0.5;
  t = (b - a)/h;
  For[i = 0, i < t , i++,
   res +=  h (\[Lambda] (res \[Minus] Cos[h i]) - Sin[h  i]);
   AppendTo[xx, {h i, res}]
   ];
  xx
  ]

  backeuler[a_, b_, h_, \[Lambda]_] := Module[{i, xx = {}, res, t},
  res =  0.5;
  t = (b - a)/h;
  For[i = 0, i < t , i++,
   (*Solve[rn == res + h(l(rn\[Minus]Cos[i h + h])-Sin[i h + h]),rn]*)

      res = (-res + h \[Lambda] Cos[h + h i] + h Sin[h + h i])/(-1 +
     h \[Lambda]);
   AppendTo[xx, {h i, res}]
   ];
  xx
  ]
```

2.5 L-Stability

Same test problem as A-stability

$$y' = \lambda y, \qquad y(0) = y_0 \tag{44}$$

but the test changes to

$$\lim_{Re(h\lambda)\to-\infty} \frac{y_n}{y_{n-1}} = 0. \tag{45}$$

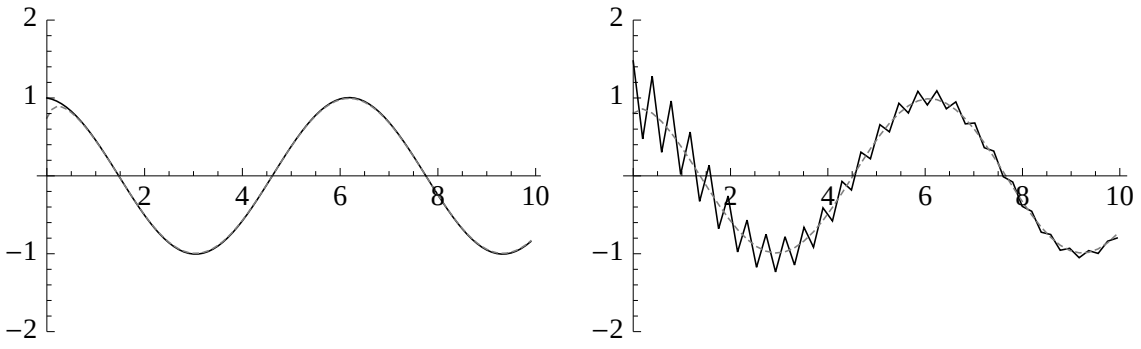

Figure 6: left graph uses $h = 0.1$, right graph uses $h = 0.195$, the dashed line is the backward Euler method, the solid line is the forward Euler.

If the fraction approaches this limit then the solution **is L-stable**.
Forward Euler:

$$\lim_{Re(h\lambda)\to-\infty} \frac{y_n}{y_{n-1}} = 1 + h\lambda \to -\infty. \tag{46}$$

Backward Euler:

$$\lim_{Re(h\lambda)\to-\infty} \frac{y_n}{y_{n-1}} = \frac{1}{1-h\lambda} \to 0. \tag{47}$$

Therefore the Backward Euler method is L-stable, but Forward-Euler is not.
Stiffness is quite common when multiple "time scales" are present in the problem, and this is a frequent problem in systems of ODEs.