Ordinary differential equations

Martin van Sint Annaland, Ivo Roghair

m.v.sintannaland@tue.nl

Chemical Process Intensification,
Eindhoven University of Technology

Part II

Implicit methods, Systems of ODEs and Boudary Value Problems

Today's outline

Problems with Euler's method: instability

Consider the ODE:

$$\frac{dy}{dx} = f(x, y(x)) \qquad \text{with} \qquad y(x=0) = y_0$$

First order approximation of derivative: $\frac{dy}{dx} = \frac{y_{i+1} - y_i}{\Delta x}$.

Where to evaluate the function f?

1. Evaluation at x_i: Explicit Euler method (forward Euler)
2. Evaluation at x_{i+1}: Implicit Euler method (backward Euler)

Problems with Euler's method: instability – forward Euler

Explicit Euler method (forward Euler):

- Use values at x_i:
  $\frac{y_{i+1}-y_i}{\Delta x} = f(x_i, y_i) \Rightarrow y_{i+1} = y_i + hf(x_i, y_i)$.
- This is an explicit equation for y_{i+1} in terms of y_i.
- It can give instabilities with large function values.

Consider the first order batch reactor:

$$\frac{dc}{dt} = -kc \Rightarrow c_{i+1} = c_i - kc_i\Delta t \Rightarrow \frac{c_{i+1}}{c_i} = 1 - k\Delta t$$

It follows that unphysical results are obtained for $k\Delta t \geq 1$!!

Stability requirement

$$k\Delta t < 1$$

(but probably accuracy requirements are more stringent here!)

Problems with Euler's method: instability – backward Euler

Implicit Euler method (backward Euler):

- Use values at x_{i+1}:
  $\frac{y_{i+1}-y_i}{\Delta x} = f(x_{i+1}, y_{i+1}) \Rightarrow y_{i+1} = y_i + hf(x_{i+1}, y_{i+1})$.
- This is an implicit equation for y_{i+1}, because it also depends on terms of y_{i+1}.

Consider the first order batch reactor:

$$\frac{dc}{dt} = -kc \Rightarrow c_{i+1} = c_i - k{\color{red}c_{i+1}}\Delta t \Rightarrow \frac{c_{i+1}}{c_i} = \frac{1}{1 + k\Delta t}$$

This equation does never give unphysical results!
The implicit Euler method is *unconditionally stable*
(but maybe not very accurate or efficient).

Semi-implicit Euler method

Usually f is a non-linear function of y, so that linearization is required (recall Newton's method).

$$\frac{dy}{dx} = f(y) \Rightarrow y_{i+1} = y_i + hf(y_{i+1}) \quad \text{using} \quad f(y_{i+1}) = f(y_i) + \left.\frac{df}{dy}\right|_i (y_{i+1} - y_i) + \ldots$$

$$\Rightarrow y_{i+1} = y_i + h\left[f(y_i) + \left.\frac{df}{dy}\right|_i (y_{i+1} - y_i)\right]$$

$$\Rightarrow \left(1 - h\left.\frac{df}{dy}\right|_i\right) y_{i+1} = \left(1 - h\left.\frac{df}{dy}\right|_i\right) y_i + hf(y_i)$$

$$\boxed{\Rightarrow \quad y_{i+1} \quad = \quad y_i \;+\; h\left(1 - h\left.\frac{df}{dy}\right|_i\right)^{-1} f(y_i)}$$

For the case that $f(x, y(x))$ we could add the variable x as an additional variable $y_{n+1} = x$. Or add one fully implicit Euler step (which avoids the computation of $\frac{\partial f}{\partial x}$):

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1}) \Rightarrow y_{i+1} = y_i + h\left(1 - h\left.\frac{df}{dy}\right|_i\right)^{-1} f(x_{i+1}, y_i)$$

Semi-implicit Euler method - example

Second order reaction in a batch reactor:
$\frac{dc}{dt} = -kc^2$ with $c_0 = 1 \text{ mol m}^{-3}$, $k = 1 \text{ m}^3 \text{ mol}^{-1} \text{ s}^{-1}$, $t_{\text{end}} = 2 \text{ s}$
Analytical solution: $c(t) = \frac{c_0}{1+kc_0 t}$

Define $f = -kc^2$, then $\frac{df}{dc} = -2kc \Rightarrow c_{i+1} = c_i - \frac{hkc_i^2}{1+2hkc_i}$.

N	ζ	$\frac{\zeta_{\text{numerical}} - \zeta_{\text{analytical}}}{\zeta_{\text{analytical}}}$	$r = \frac{\log\left(\frac{\epsilon_i}{\epsilon_{i-1}}\right)}{\log\left(\frac{N_{i-1}}{N_i}\right)}$
20	0.654066262	1.89×10^{-2}	—
40	0.660462687	9.31×10^{-3}	1.02220
80	0.663589561	4.62×10^{-3}	1.01162
160	0.665134433	2.30×10^{-3}	1.00594
320	0.665902142	1.15×10^{-3}	1.00300

Second order implicit method: Implicit midpoint method

Implicit midpoint rule (second order)	Explicit midpoint rule (modified Euler method)
$y_{i+1} = y_i + hf\left(x_i + \frac{1}{2}h, \frac{1}{2}(y_i + y_{i+1})\right)$	$y_{i+1} = y_i + hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$

in case $f(y)$ then:

$$f\left(\frac{1}{2}(y_i + y_{i+1})\right) = f_i + \left.\frac{df}{dy}\right|_i \left(\frac{1}{2}(y_i + y_{i+1}) - y_i\right) = f_i + \frac{1}{2}\left.\frac{df}{dy}\right|_i (y_{i+1} - y_i)$$

Implicit midpoint rule reduces to:

$$y_{i+1} = y_i + hf_i + \frac{h}{2}\left.\frac{df}{dy}\right|_i (y_{i+1} - y_i)$$

$$\Rightarrow \left(1 - \frac{h}{2}\left.\frac{df}{dy}\right|_i\right) y_{i+1} = \left(1 - \frac{h}{2}\left.\frac{df}{dy}\right|_i\right) y_i + hf_i$$

$$\boxed{\Rightarrow \quad y_{i+1} \quad = \quad y_i \; + \; h\left(1 - \frac{h}{2}\left.\frac{df}{dy}\right|_i\right)^{-1} f_i}$$

Implicit midpoint method — example

Second order reaction in a batch reactor:
$\frac{dc}{dt} = -kc^2$ with $c_0 = 1\ \mathrm{mol\,m^{-3}}$, $k = 1\ \mathrm{m^3\,mol^{-1}\,s^{-1}}$, $t_{\mathrm{end}} = 2\ \mathrm{s}$
(Analytical solution: $c(t) = \frac{c_0}{1+kc_0 t}$).

Define $f = -kc^2$, then $\frac{df}{dc} = -2kc$.

Substitution:

$$
\begin{aligned}
c_{i+1} &= c_i + h\left(1 - \frac{h}{2}\cdot(-2kc_i)\right)^{-1}\cdot(-kc_i^2) \\
&= c_i - \frac{hkc_i^2}{1+hkc_i} = \frac{c_i + hkc_i^2 - hkc_i^2}{1+hkc_i} \Rightarrow c_{i+1} = \frac{c_i}{1+hkc_i}
\end{aligned}
$$

You will find that this method is exact for all step sizes h because of the quadratic source term!

Implicit midpoint method — example

Second order reaction in a batch reactor:
$\frac{dc}{dt} = -kc^2$ with $c_0 = 1\ \mathrm{mol\,m^{-3}}$, $k = 1\ \mathrm{m^3\,mol^{-1}\,s^{-1}}$, $t_{\mathrm{end}} = 2\ \mathrm{s}$
Analytical solution: $c(t) = \frac{c_0}{1+kc_0t}$

$$c_{i+1} = \frac{c_i}{1 + hkc_i}$$

N	ζ	$\frac{\zeta_{\mathrm{numerical}}-\zeta_{\mathrm{analytical}}}{\zeta_{\mathrm{analytical}}}$	$r = \frac{\log\left(\frac{\epsilon_i}{\epsilon_{i-1}}\right)}{\log\left(\frac{N_{i-1}}{N_i}\right)}$
20	0.6666666667	1.665×10^{-16}	—
40	0.6666666667	0	—
80	0.6666666667	0	—
160	0.6666666667	0	—
320	0.6666666667	0	—

Implicit midpoint method — example

Third order reaction in a batch reactor: $\frac{dc}{dt} = -kc^3$
Analytical solution: $c(t) = \frac{c_0}{\sqrt{1+2kc_0^2 t}}$

$$c_{i+1} = c_i - \frac{hkc_i^3}{1 + \frac{3}{2}hkc_i^2}$$

N	ζ	$\frac{\zeta_{\text{numerical}} - \zeta_{\text{analytical}}}{\zeta_{\text{analytical}}}$	$r = \frac{\log\left(\frac{\epsilon_i}{\epsilon_{i-1}}\right)}{\log\left(\frac{N_{i-1}}{N_i}\right)}$
20	0.5526916174	1.71×10^{-4}	—
40	0.5527633731	4.17×10^{-5}	2.041
80	0.5527807304	1.03×10^{-5}	2.021
160	0.5527849965	2.55×10^{-6}	2.011
320	0.5527860538	6.34×10^{-7}	2.005

Today's outline

Systems of ODEs

A system of ODEs is specified using vector notation:

$$\frac{d\boldsymbol{y}}{dx} = \boldsymbol{f}(x, \boldsymbol{y}(x))$$

for

$$\frac{dy_1}{dx} = f_1(x, y_1(x), y_2(x)) \quad \text{or} \quad f_1(x, y_1, y_2)$$
$$\frac{dy_2}{dx} = f_2(x, y_1(x), y_2(x)) \quad \text{or} \quad f_2(x, y_1, y_2)$$

The solution techniques discussed before can also be used to solve systems of equations.

Systems of ODEs: Explicit methods

Forward Euler method

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\boldsymbol{f}(x_i, \boldsymbol{y}_i)$$

Improved Euler method (classical RK2)

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \frac{h}{2}(\boldsymbol{k}_1 + \boldsymbol{k}_2) \quad \text{using} \quad \begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(x_i, \boldsymbol{y}_i) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(x_i + h, \boldsymbol{y}_i + h\boldsymbol{k}_1) \end{aligned}$$

Modified Euler method (midpoint rule)

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\boldsymbol{k}_2 \quad \text{using} \quad \begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(x_i, \boldsymbol{y}_i) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(x_i + \frac{h}{2}, \boldsymbol{y}_i + \frac{h}{2}\boldsymbol{k}_1) \end{aligned}$$

Systems of ODEs: Explicit methods

Classical fourth order Runge-Kutta method (RK4)

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\left(\frac{\boldsymbol{k}_1}{6} + \frac{1}{3}\left(\boldsymbol{k}_2 + \boldsymbol{k}_3\right) + \frac{\boldsymbol{k}_4}{6}\right)$$

using

$$\boldsymbol{k}_1 = \boldsymbol{f}(x_i, \boldsymbol{y}_i)$$

$$\boldsymbol{k}_2 = \boldsymbol{f}(x_i + \frac{h}{2}, \boldsymbol{y}_i + \frac{h}{2}\boldsymbol{k}_1)$$

$$\boldsymbol{k}_3 = \boldsymbol{f}(x_i + \frac{h}{2}, \boldsymbol{y}_i + \frac{h}{2}\boldsymbol{k}_2)$$

$$\boldsymbol{k}_4 = \boldsymbol{f}(x_i + h, \boldsymbol{y}_i + h\boldsymbol{k}_3)$$

Solving systems of ODEs in Matlab

Solving systems of ODEs in Matlab is completely analogous to solving a single ODE:

1. Create a function that specifies the ODEs. This function returns the $\frac{d\mathbf{y}}{dx}$ vector.
2. Initialise solver variables and settings (e.g. step size, initial conditions, tolerance), in a separate script. Initial conditions and tolerances should be given per-equation, i.e. as a vector.
3. Call the ODE solver function, using a *function handle* to the ODE function described in point 1.
    - The ODE solver will return the vector for the independent variable, and a solution matrix, with a column as the solution for each equation in the system.

Solving systems of ODEs in Matlab: example

We solve the system: $\frac{dx_1}{dt} = -x_1 - x_2, \quad \frac{dx_2}{dt} = x_1 - 2x_2$

Create an ODE function

```
function [dxdt] = myODEFunction(t,x)
dxdt(1) = -x(1) - x(2);
dxdt(2) =  x(1) - 2*x(2);
dxdt=dxdt'; % Transpose to column vector
return
```

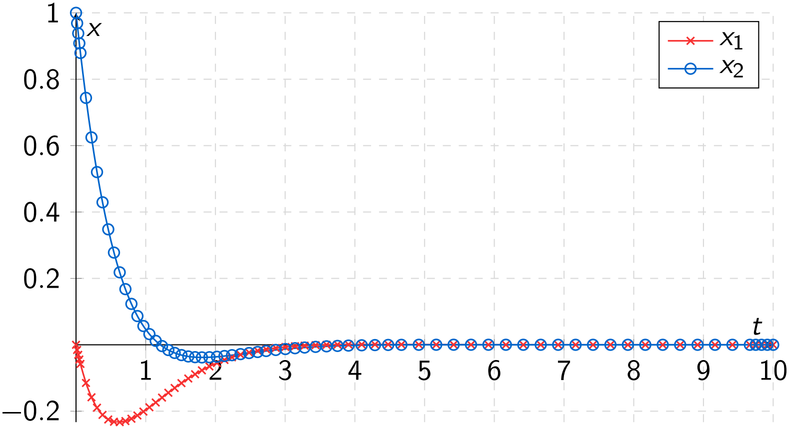
Create a solution script

```
x_init = [0 1];              % Initial conditions
tspan = [0 10];              % Time span
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4]);
[t,x] = ode45(@myODEfunction,tspan,x_init,options);
```

Solving systems of ODEs in Matlab: example

Plot the solution:

```
plot(t,x(:,1),'r-x',t,x(:,2),'b-o')
```



Solving systems of ODEs in Matlab: repeated notes

A few notes on working with `ode45` and other solvers. If we want to give additional arguments (e.g. `a`, `b` and `c`) to our ODE function, we can list them in the function line:

```
function [dxdt] = myODE(t,x,a,b,c)
```

The additional arguments can now be set in the solver script by *adding them after the options*:

```
[t,x] = ode45(@myODE,tspan,x_0,options,a,b,c);
```

- Of course, in the solver script, the variables do not need to be called `a`, `b` and `c`:

  ```
  [t,x] = ode45(@myODE,tspan,x_0,options,k1,phi,V);
  ```

- These variables may be of any type (vectors, matrix, struct). Especially a struct is useful to carry many values in 1 variable.

Systems of ODEs: Implicit methods

Backward Euler method

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h \left(1 - h \left. \frac{d\boldsymbol{f}}{d\boldsymbol{y}} \right|_i \right)^{-1} \boldsymbol{f}(\boldsymbol{y}_i)$$

Implicit midpoint method

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h \left(1 - \frac{h}{2} \left. \frac{d\boldsymbol{f}}{d\boldsymbol{y}} \right|_i \right)^{-1} \boldsymbol{f}(\boldsymbol{y}_i)$$

Stiff systems of ODEs

A system of ODEs can be stiff and require a different solution method. For example:

$$\frac{dc_1}{dt} = 998c_1 + 1998c_2 \qquad \frac{dc_2}{dt} = -999c_1 - 1999c_2$$

with boundary conditions $c_1(t=0) = 1$ and $c_2(t=0) = 0$.
The analytical solution is:

$$c_1 = 2e^{-t} - e^{-1000t} \qquad c_2 = -e^{-t} + e^{-1000t}$$

For the explicit method we require $\Delta t < 10^{-3}$ despite the fact that the term is completely negligible, but essential to keep stability.

> The "disease" of stiff equations: we need to follow the solution on the shortest length scale to maintain stability of the integration, although accuracy requirements would allow a much larger time step.

Demonstration with example

Forward Euler (explicit)

$$\frac{c_{1,i+1} - c_{1,i}}{dt} = 998c_{1,i} + 1998c_{2,i}$$
$$\frac{c_{2,i+1} - c_{2,i}}{dt} = -999c_{1,i} - 1999c_{2,i}$$

$$\Rightarrow \begin{array}{l} c_{1,i+1} = (1 + 998\Delta t)\, c_{1,i} + 1998\Delta t c_{2,i} \\ c_{2,i+1} = -999\Delta t c_{1,i} + (1 - 1999\Delta t)\, c_{2,i} \end{array}$$

Demonstration with example

Backward Euler (implicit)

$$\frac{dc_{1,i+1} - c_{1,i+1}}{dt} = 998c_{1,i+1} + 1998c_{2,i+1}$$
$$\frac{dc_{2,i+1} - c_{2,i+1}}{dt} = -999c_{1,i+1} - 1999c_{2,i+1}$$

$$\Rightarrow \begin{array}{l} (1 - 998\Delta t)\, c_{1,i+1} - 1998\Delta t c_{2,i} = c_{1,i} \\ 999\Delta t c_{1,i+1} + (1 + 999\Delta t)\, c_{2,i+1} = c_{2,i} \end{array}$$

$$A\boldsymbol{c}_{i+1} = \boldsymbol{c}_i \text{ with } A = \begin{pmatrix} 1 - 998\Delta t & -1998\Delta t \\ 999\Delta t & 1 + 1999\Delta t \end{pmatrix} \text{ and } \boldsymbol{b} = \begin{pmatrix} c_{1,i} \\ c_{2,i} \end{pmatrix}$$

Demonstration with example

Backward Euler (implicit) $A\boldsymbol{c}_{i+1} = \boldsymbol{c}_i$ with

$$A = \begin{pmatrix} 1-998\Delta t & -1998\Delta t \\ 999\Delta t & 1+1999\Delta t \end{pmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} c_{1,i} \\ c_{2,i} \end{pmatrix}$$

Cramers rule:

$$c_{1,i+1} = \frac{\begin{vmatrix} c_{1,i} & -1998\Delta t \\ c_{2,i} & 1+1999\Delta t \end{vmatrix}}{\det \begin{vmatrix} A \end{vmatrix}} = \frac{(1+1999\Delta t)c_{1,i}+1998\Delta t c_{2,i}}{(1-998\Delta t)(1+1999\Delta t)+1998\cdot 999\Delta t^2}$$

$$c_{2,i+1} = \frac{\begin{vmatrix} 1-998\Delta t & c_{1,i} \\ 999\Delta t & c_{2,i} \end{vmatrix}}{\det \begin{vmatrix} A \end{vmatrix}} = \frac{-999\Delta t c_{1,i}+(1-998\Delta t)c_{2,i}}{(1-998\Delta t)(1+1999\Delta t)+1998\cdot 999\Delta t^2}$$

Forward Euler: $\Delta t \leq 0.001$ for stability
Backward Euler: always stable, even for $\Delta t > 100$ (but then not very accurate!)

Demonstration with example

Cure for stiff problems: use implicit methods! To find out whether your system is stiff: check whether one of the eigenvalues have an imaginary part

Implicit methods in Matlab

Matlab offers a stabilized solver, `ode15s`, for stiff problems.

$$\frac{dc_1}{dt} = 998c_1 + 1998c_2 \quad \frac{dc_2}{dt} = -999c_1 - 1999c_2,\ c_1(0) = 1,\ c_2(0) = 0$$

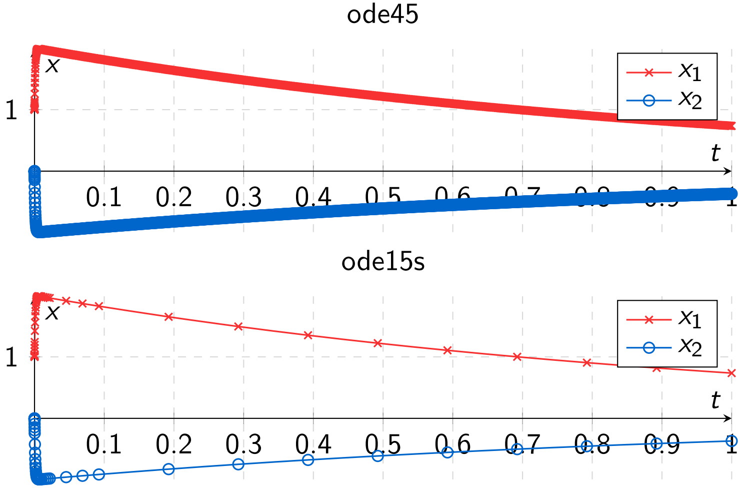
- Create the ode function

```
function [dcdt] = stiff_ode(t,c)
dcdt = zeros(2,1); % Pre-allocation
dcdt(1) =  998 * c(1) + 1998*c(2);
dcdt(2) = -999 * c(1) - 1999*c(2);
return
```

- Compare the resolution of the solutions

```
subplot(2,1,1);
ode45(@stiff_ode, [0 1], [1 0]);
subplot(2,1,2);
ode15s(@stiff_ode, [0 1], [1 0]);
```

Implicit methods in Matlab



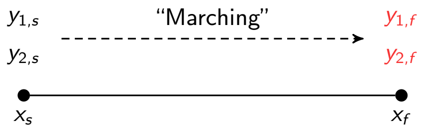
The explicit solver requires 1245 data points (default settings), the implicit solver requires just 48!

Today's outline

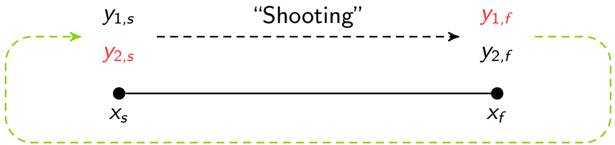
Importance of boundary conditions

The nature of boundary conditions determines the appropriate numerical method. Classification into 2 main categories:

- *Initial value problems (IVP)*
  We know the values of all y_i at some starting position x_s, and it is desired to find the values of y_i at some final point x_f.

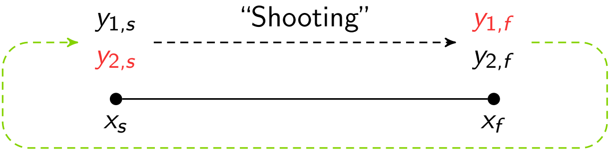


- *Boundary value problems (BVP)*
  Boundary conditions are specified at more than one x. Typically, some of the BC are specified at x_s and the remainder at x_f.



Shooting method

How to solve a BVP using the shooting method:



- Define the system of ODEs
- Provide an initial guess for the unknown boundary condition
- Solve the system and compare the resulting boundary condition to the expected value
- Adjust the guessed boundary value, and solve again. Repeat until convergence.
    - Of course, you can subtract the expected value from the computed value at the boundary, and use a non-linear root finding method

BVP: example in Excel

Consider a chemical reaction in a liquid film layer of thickness δ:

$$\mathcal{D}\frac{d^2c}{dx^2} = k_R c \text{ with } \quad \begin{aligned} c(x=0) &= C_{A,i,L} = 1 && \text{(interface concentration)} \\ c(x=\delta) &= 0 && \text{(bulk concentration)} \end{aligned}$$

Question: compute the concentration profile in the film layer.

Step 1: Define the system of ODEs

This second-order ODE can be rewritten as a system of first-order ODEs, if we define the flux q as:

$$q = -\mathcal{D}\frac{dc}{dx}$$

Now, we find:

$$\begin{aligned} \frac{dc}{dx} &= -\frac{1}{\mathcal{D}}q \\ \frac{dq}{dx} &= -k_R c \end{aligned}$$

BVP: example in Excel

Solving the two first-order ODEs in Excel. First, the cells with constants:

	A	B	C
1	CAiL	1	mol/m3
2	D	1e–8	m2/s
3	kR	10	1/s
4	delta	1e–4	m
5	N	100	
6	dx	=B4/B5	

$$\frac{dc}{dx} = -\frac{1}{\mathcal{D}}q$$

$$\frac{dq}{dx} = -k_R c$$

Now, we program the forward Euler (explicit) schemes for c and q below:

	A	B	C
10	x	c	q
11	0	=B1	10
12	=A11+$B$6	=B11+$B$6*(–1/$B$2*C11)	=C11+$B$6*(-$B$3*B11)
13	=A12+$B$6	=B12+$B$6*(–1/$B$2*C12)	=C12+$B$6*(-$B$3*B12)
...	...	...	...
111	=A110+$B$6	=B110+$B$6*(–1/$B$2*C110)	=C110+$B$6*(-$B$3*B110)

BVP: example in Excel

- We now have profiles for c and q as a function of position x.
- The concentration $c(x = \delta)$ depends (eventually) on the boundary condition at the interface $q(x = 0)$
- We can use the solver to change $q(x = 0)$ such that the concentration at the bulk meets our requirement: $c(x = \delta) = 0$

BVP: example in Matlab

We first program the system of ODEs in a separate function:

$$\frac{dc}{dx} = -\frac{1}{\mathcal{D}}q$$
$$\frac{dq}{dx} = -k_R c$$

```
function [dxdt] = BVPODE(t,x,ps)
dxdt(1)=-1/ps.D*x(2);
dxdt(2)=-ps.kR*x(1);
dxdt=dxdt';
return
```

Note that we pass a variable (type: struct) that contains required parameters: `ps`.

BVP: example in Matlab

The ODE function is solved via ode45, after setting a number of initial and boundary conditions:

```
function f = RunBVP(bcq,ps)
[x,cq] = ode45(@BVPODE,[0 ps.delta],[1 bcq], [], ps);
f = cq(end,1) - 0;
plotyy(x,cq(:,1),x,cq(:,2));
return;
```

Note the following:

- We use the interval $0 \leq x \leq \delta$
- Boundary conditions are given as: $c(x=0)=1$ and $q(x=0)=$ `bcq`, which is given as an argument to the function (i.e. changable from 'outside'!)
- The function returns `f`, the difference between the computed and desired concentration at $x=\delta$.

BVP: example in Matlab

Finally, we should solve the system so that we obtain the right boundary condition $q = \texttt{bcq}$ such that $c(x = \delta) = 0$. We can use the built-in function `fzero` to do this

```
% Parameter definition
ps.D=1e-8;
ps.kR=10;
ps.delta=1e-4;

% Solve for flux boundary condition (initial guess: 0)
opt = optimset('Display','iter');
flux = fzero(@RunBVP,0,opt,ps);
```

BVP example: analytical solution

Compare with the analytical solution:

$$q = k_L E_A C_{A,i,L} \quad \text{with}$$

$$E_A = \frac{Ha}{\tanh Ha} \qquad \text{(Enhancement factor)}$$

$$Ha = \frac{\sqrt{k_R \mathcal{D}}}{k_L} \qquad \text{(Hatta number)}$$

$$k_L = \frac{\mathcal{D}}{\delta} \qquad \text{(mass transfer coefficient)}$$

Today's outline

Other methods

Other explicit methods:

- Burlisch-Stoer method (Richardson extrapolation + modified midpoint method)

Other implicit methods:

- Rosenbrock methods (higher order implicit Runge-Kutta methods)
- Predictor-corrector methods

Summary

- Several solution methods and their derivation were discussed:
  - Explicit solution methods: Euler, Improved Euler, Midpoint method, RK45
  - Implicit methods: Implicit Euler and Implicit midpoint method
  - A few examples of their spreadsheet implementation were shown
- We have paid attention to accuracy and instability, rate of convergence and step size
- Systems of ODEs can be solved by the same algorithms. Stiff problems should be treated with care.
- An example of solving ODEs with Matlab was demonstrated.