Differential equations for chemical kinetics

Atmospheric Modelling course

Sampo Smolander
sampo.smolander@helsinki.fi

15.11.2010

Chemical equation

$$\textit{reactants} \rightarrow \textit{products} : \textit{rate coefficient}$$

$$\mathrm{SO_3} + \mathrm{H_2O} \rightarrow \mathrm{H_2SO_4} : 5.0^{-15}$$

$$\frac{\partial[\mathrm{H_2SO_4}]}{\partial t} = k_1[\mathrm{SO_3}][\mathrm{H_2O}]$$

$$\frac{\partial[\mathrm{SO_3}]}{\partial t} = -k_1[\mathrm{SO_3}][\mathrm{H_2O}]$$

$$\frac{\partial[\mathrm{H_2O}]}{\partial t} = -k_1[\mathrm{SO_3}][\mathrm{H_2O}]$$

$$k_1 = 5.0^{-15}\,\mathrm{mol/s}$$

$$\mathrm{CH_4} + 2\,\mathrm{O_2} \rightarrow \mathrm{CO_2} + 2\,\mathrm{H_2O} \quad : \quad k_2$$

$$\begin{aligned}
\frac{\partial[\mathrm{CH_4}]}{\partial t} &= -k_2[\mathrm{CH_4}][\mathrm{O_2}]^2 \\
\frac{\partial[\mathrm{O_2}]}{\partial t} &= -2\,k_2[\mathrm{CH_4}][\mathrm{O_2}]^2 \\
\frac{\partial[\mathrm{CO_2}]}{\partial t} &= k_2[\mathrm{CH_4}][\mathrm{O_2}]^2 \\
\frac{\partial[\mathrm{H_2O}]}{\partial t} &= 2\,k_2[\mathrm{CH_4}][\mathrm{O_2}]^2
\end{aligned}$$

$$\mathrm{CH_4} + 2\,\mathrm{O_2} \rightarrow \mathrm{CO_2} + 2\,\mathrm{H_2O} \quad : \quad k_2$$

There is ca. 21% = 210 000 ppm oxygen and 1.8 ppm methane in atmosphere, so we don't really need to monitor the *tiny* change in the amount of oxygen

$$\mathrm{CH_4} \rightarrow \mathrm{CO_2} + 2\,\mathrm{H_2O} \quad : \quad k_2 \cdot [\mathrm{O_2}]^2$$

or actually not water either

$$\mathrm{CH_4} \rightarrow \mathrm{CO_2} \quad : \quad k_2 \cdot [\mathrm{O_2}]^2$$

but the reaction rate still depends on the oxygen concentration (Writing like this ignores the conservation of atoms, but we don't care)

We can have:

Variable reactants
we monitor the change

Constant reactants
we need to know the concentration, to calculate reaction rates, but we can assume constant concentration

Variable products
we monitor the change

Uninteresting products
we’re not interested at all

Brusselator (theoretical example)

$$
\begin{aligned}
\mathrm{A} &\rightarrow \mathrm{X} && : k_1 \\
2\,\mathrm{X} + \mathrm{Y} &\rightarrow 3\,\mathrm{X} && : k_2 \\
\mathrm{B} + \mathrm{X} &\rightarrow \mathrm{Y} + \mathrm{C} && : k_3 \\
\mathrm{X} &\rightarrow \mathrm{D} && : k_4
\end{aligned}
$$

Constant: A, B

Variable: X, Y

Uninteresting: C, D

http://en.wikipedia.org/wiki/Brusselator
http://www.idea.wsu.edu/OscilChem/#Brusselator%20Model

$$\mathrm{A} \rightarrow \mathrm{X} \qquad \frac{\partial[\mathrm{X}]}{\partial t} = -k_1[\mathrm{A}]$$

$$2\,\mathrm{X} + \mathrm{Y} \rightarrow 3\,\mathrm{X} \qquad \begin{aligned} \frac{\partial[\mathrm{X}]}{\partial t} &= k_2[\mathrm{X}]^2[\mathrm{Y}] \\ \frac{\partial[\mathrm{Y}]}{\partial t} &= -k_2[\mathrm{X}]^2[\mathrm{Y}] \end{aligned}$$

$$\mathrm{B} + \mathrm{X} \rightarrow \mathrm{Y} + \mathrm{C} \qquad \begin{aligned} \frac{\partial[\mathrm{X}]}{\partial t} &= -k_3[\mathrm{X}][\mathrm{B}] \\ \frac{\partial[\mathrm{Y}]}{\partial t} &= k_3[\mathrm{X}][\mathrm{B}] \end{aligned}$$

$$\mathrm{X} \rightarrow \mathrm{D} \qquad \frac{\partial[\mathrm{X}]}{\partial t} = -k_4[\mathrm{X}]$$

We don’t need to write equations for A, B because they stay constant

And C, D we are not interested in

Combine these

$$\frac{\partial[\mathrm{X}]}{\partial t} = -k_1[\mathrm{A}]\,,\ \frac{\partial[\mathrm{X}]}{\partial t} = k_2[\mathrm{X}]^2[\mathrm{Y}]\,,\ \frac{\partial[\mathrm{X}]}{\partial t} = -k_3[\mathrm{X}][\mathrm{B}]\,,\ \frac{\partial[\mathrm{X}]}{\partial t} = -k_4[\mathrm{X}]$$

into

$$\frac{\partial[\mathrm{X}]}{\partial t} = -k_1[\mathrm{A}] + k_2[\mathrm{X}]^2[\mathrm{Y}] - k_3[\mathrm{X}][\mathrm{B}] - k_4[\mathrm{X}]$$

Do same for [Y], get...

$$\frac{\partial[\mathrm{X}]}{\partial t} = -k_1[\mathrm{A}] + k_2[\mathrm{X}]^2[\mathrm{Y}] - k_3[\mathrm{X}][\mathrm{B}] - k_4[\mathrm{X}]$$

$$\frac{\partial[\mathrm{Y}]}{\partial t} = -k_2[\mathrm{X}]^2[\mathrm{Y}] + k_3[\mathrm{X}][\mathrm{B}]$$

Or with more chemicals and reactions, you'll just get a larger system of coupled differential equations. But the principle is easy?

Next: numerics

$$\frac{\partial[\mathrm{X}]}{\partial t} = -k_1[\mathrm{A}] + k_2[\mathrm{X}]^2[\mathrm{Y}] - k_3[\mathrm{X}][\mathrm{B}] - k_4[\mathrm{X}]$$

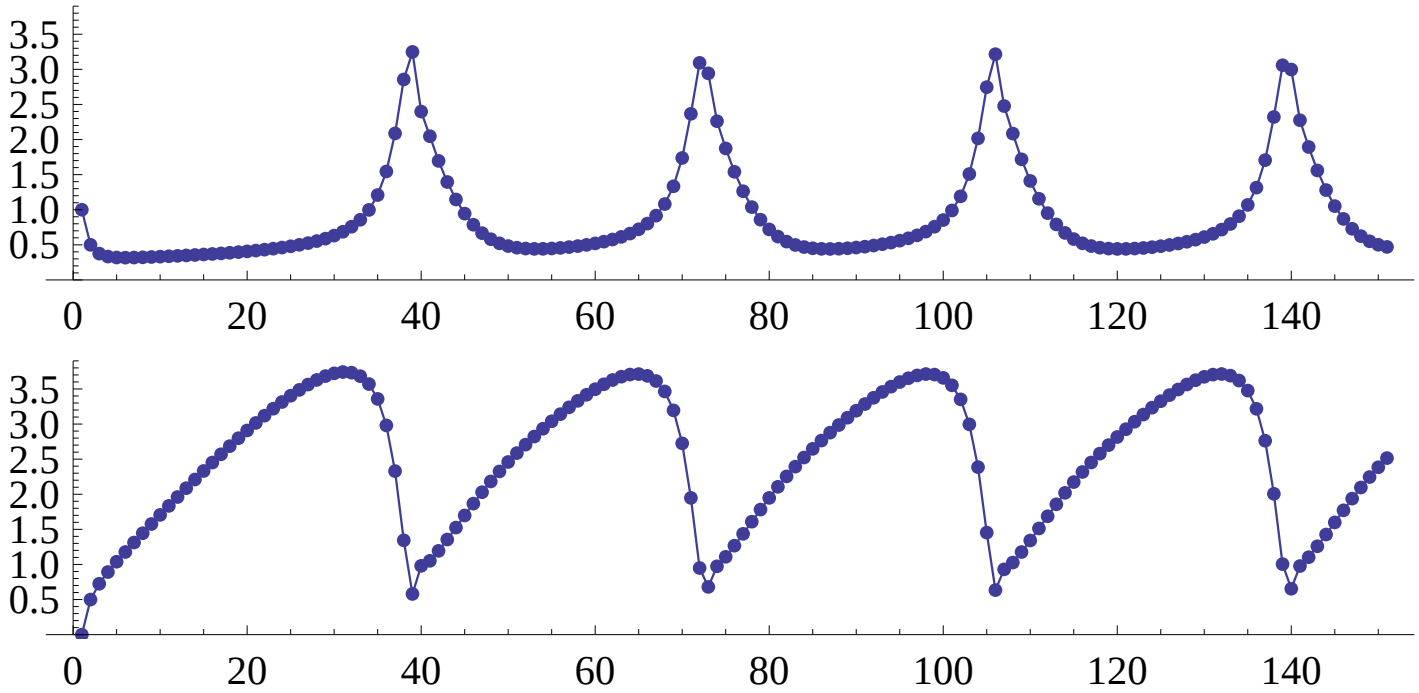
$$\frac{\partial[\mathrm{Y}]}{\partial t} = -k_2[\mathrm{X}]^2[\mathrm{Y}] + k_3[\mathrm{X}][\mathrm{B}]$$

```
! c(1)=X, c(2)=Y; cc(1)=A, cc(2)=B
SUBROUTINE deriv(dc,c,cc,k)
  REAL(dp), INTENT(in)  :: c(dim_c),cc(dim_cc),k(dim_k)
  REAL(dp), INTENT(out) :: dc(dim_c)
  dc(1) = k(1)*cc(1) + k(2)*c(1)*c(1)*c(2) &
          - k(3)*cc(2)*c(1) - k(4)*c(1)
  dc(2) = k(3)*cc(2)*c(1) - k(2)*c(1)*c(1)*c(2)
END SUBROUTINE deriv
```

```
PROGRAM brusselator
   ! http://en.wikipedia.org/wiki/Brusselator
   IMPLICIT NONE
   INTEGER, PARAMETER :: dp = SELECTED_REAL_KIND(15)
      ! so that things will be in double precision
      ! http://en.wikipedia.org/wiki/Double_precision_floating-point_format
   INTEGER, PARAMETER :: dim_c  = 2, dim_cc = 2, dim_k  = 4
   REAL(dp) :: dt,t,t2,k(dim_k),c(dim_c),dc(dim_c),cc(dim_cc)

   dt = 0.2                          ! time step
   k = (/ 1.0, 1.0, 1.0, 1.0 /)  ! rate coeffs
   ! initial conditions
   cc(1) = 1.0 ; cc(2) = 2.5 ; c(1) = 1.0 ; c(2) = 0.0

   t = 0.0 ; t2 = 10*60.0
   WRITE(*,*) c
   DO WHILE (t < t2)
      CALL deriv(dc,c,cc,k)
      c = c + dc*dt
      WRITE(*,*) c
      t = t+dt
   END DO

CONTAINS

   SUBROUTINE deriv(dc,c,cc,k)
      REAL(dp), INTENT(in)  :: c(dim_c),cc(dim_cc),k(dim_k)
      REAL(dp), INTENT(out) :: dc(dim_c)
      dc(1) = k(1)*cc(1) + k(2)*c(1)*c(1)*c(2) - k(3)*cc(2)*c(1) - k(4)*c(1)
      dc(2) = k(3)*cc(2)*c(1) - k(2)*c(1)*c(1)*c(2)
   END SUBROUTINE deriv

END PROGRAM brusselator
```

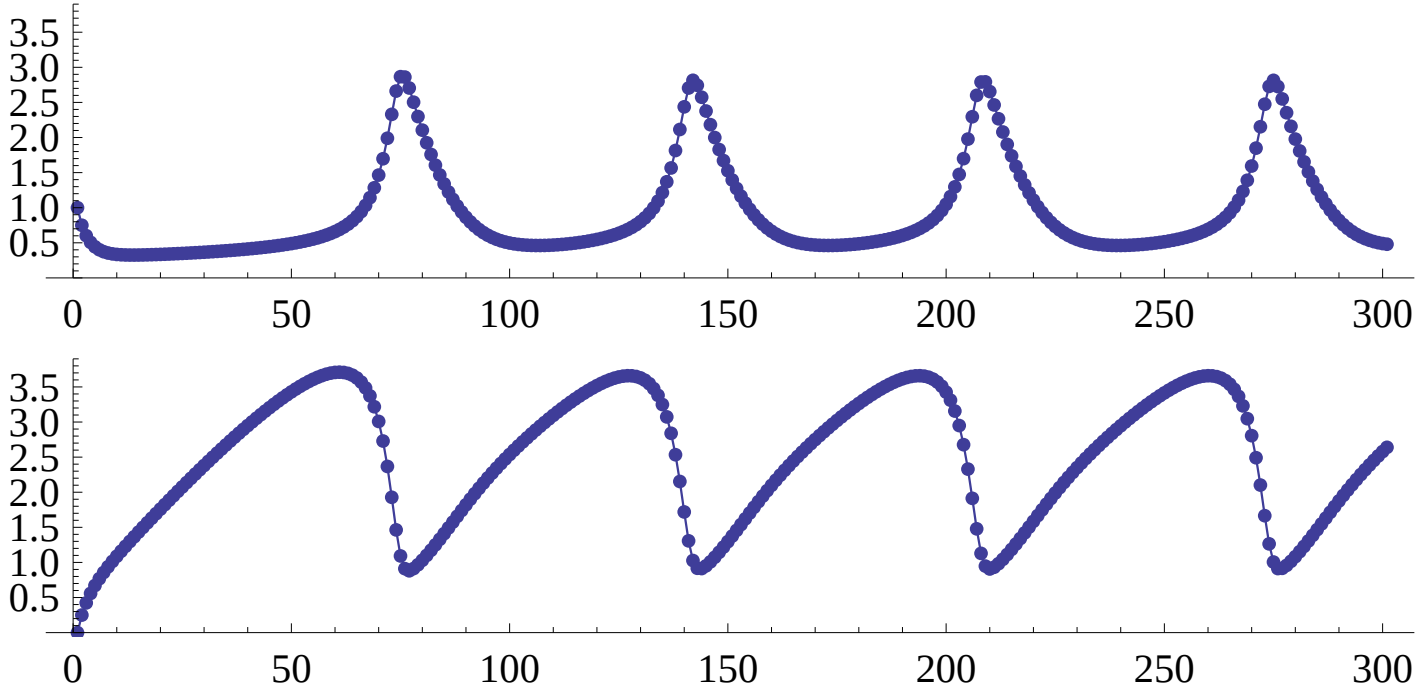
Run the program

```
./a.out > xy.dat
```



Well, maybe the dt = 0.2 time step is a little large, try 0.1

Gives a bit smoother curves



(Not that this is very important in this course)

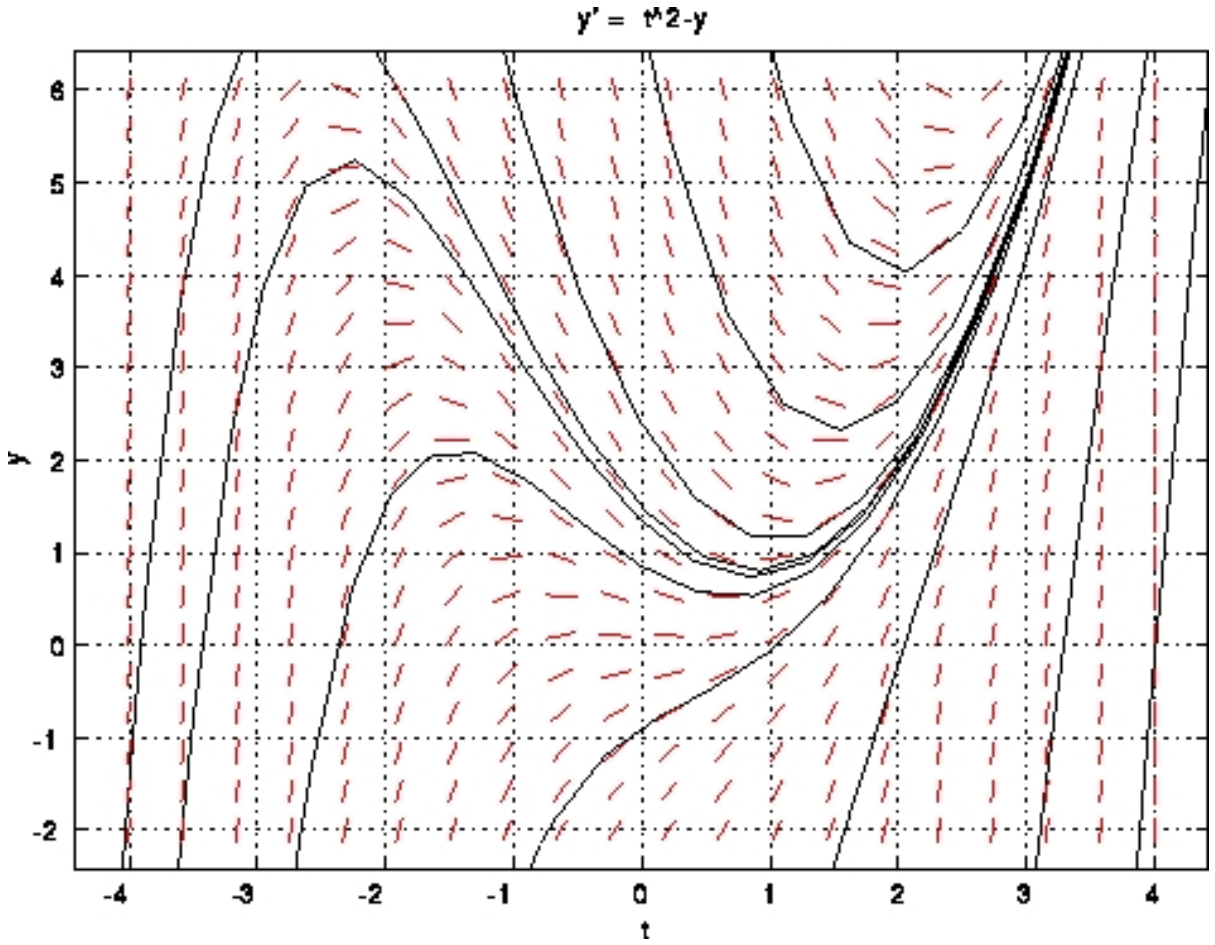
A system of differential equations can be expressed as one vector-valued equation

$$\frac{\partial \mathbf{C}}{\partial t} = \mathbf{f}(\mathbf{C}, t)$$

$\mathbf{C} = (c_1, c_2, \ldots)$ are the chemical concentrations and $\mathbf{f}$ gives the derivatives $\left(\frac{\partial c_1}{\partial t}, \frac{\partial c_2}{\partial t}, \ldots\right)$

$\mathbf{f}$ may or may not dependt on t

Equation like $\frac{\partial \mathbf{C}}{\partial t} = \mathbf{f}(\mathbf{C}, t)$ defines a *vector field* in every point in space



Then we start from an *initial point* and try to follow a *trajectory*

In the previous code, we are at some state $\mathbf{C}(t)$, then we evaluate the derivative at this state, and take a step to that direction

This is called the *Euler method*
http://en.wikipedia.org/wiki/Euler_method

It is the simplest, but also most imprecise method for solving differential equations numerically

It is good enough for this course, but I just want to mention one another method

Maybe the derivative changes during the step?

With stepsize $dt = h$ we step from $\mathbf{C}(t)$ to $\mathbf{C}(t+h)$ just following the direction $\frac{\partial \mathbf{C}(t)}{\partial t}$ but at $\mathbf{C}(t+h)$ we should already be going to direction $\frac{\partial \mathbf{C}(t+h)}{\partial t}$

Next: Runge-Kutta method

$$\frac{\partial \mathbf{C}(t)}{\partial t} = \mathbf{f}(\mathbf{C}(t), t)$$

At the initial point $\mathbf{C}_t$ the direction is

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{C}_t, t)$$

take a half-step $h/2$ to that direction, and see what's the new direction there

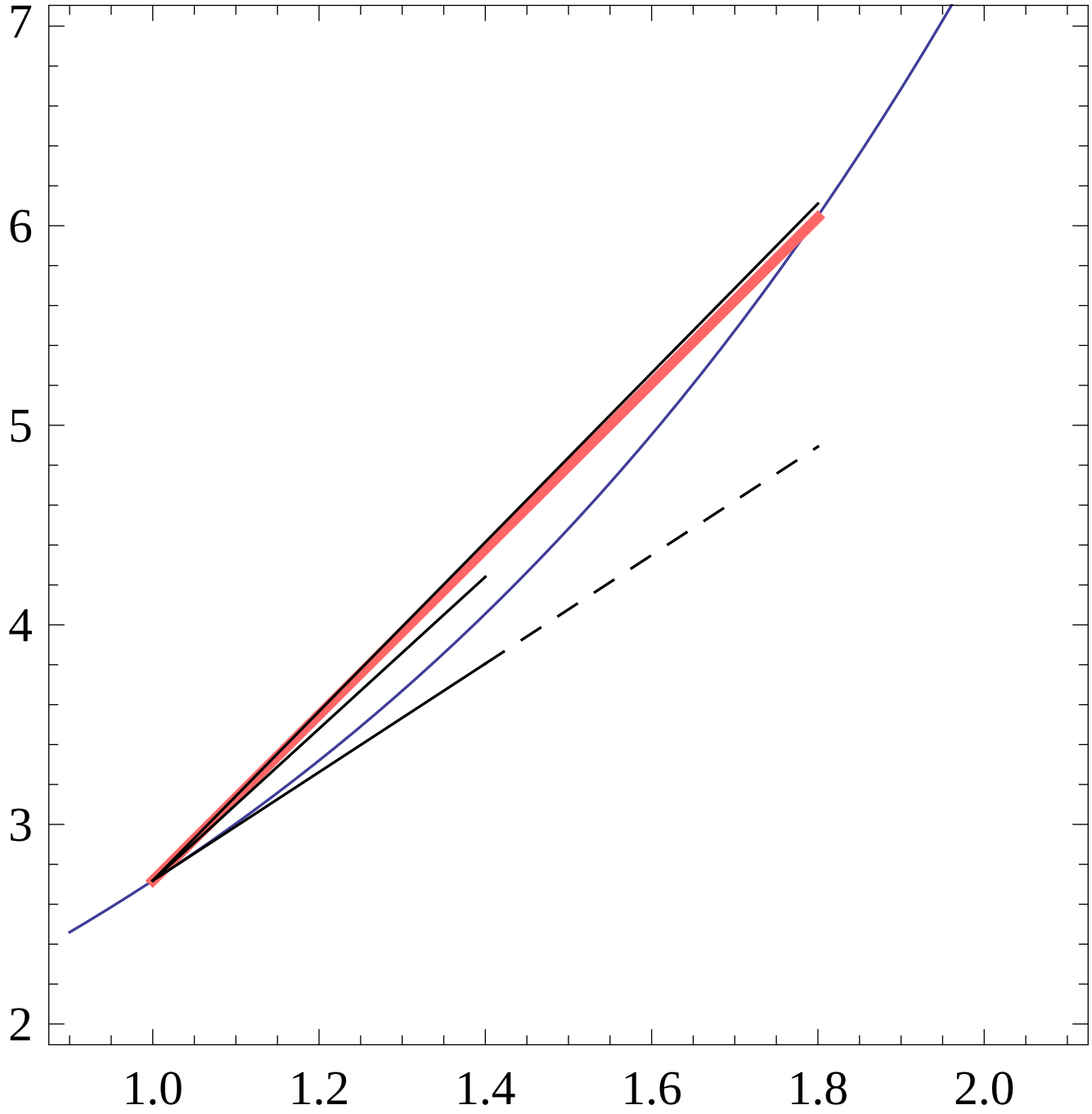
$$\mathbf{k}_2 = \mathbf{f}(\mathbf{C}_t + \mathbf{k}_1 \cdot h/2, t + h/2)$$

go back to initial point, and take a half-step to direction $\mathbf{k}_2$ and see the new direction there

$$\mathbf{k}_3 = \mathbf{f}(\mathbf{C}_t + \mathbf{k}_2 \cdot h/2, t + h/2)$$

go back, take a full step to direction $\mathbf{k}_3$

$$\mathbf{k}_4 = \mathbf{f}(\mathbf{C}_t + \mathbf{k}_3 \cdot h, t + h)$$


7
6
5
4
3
2
1.0
1.2
1.4
1.6
1.8
2.0

Now, $\mathbf{C}_t + \mathbf{k}_3 \cdot h$ would already be a good step, but we're not done yet. We have 4 different candidates for the "right" direction to go

$$\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$$

Take a *weighted average*

$$\mathbf{k} = 1/6\,(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

and go *there*

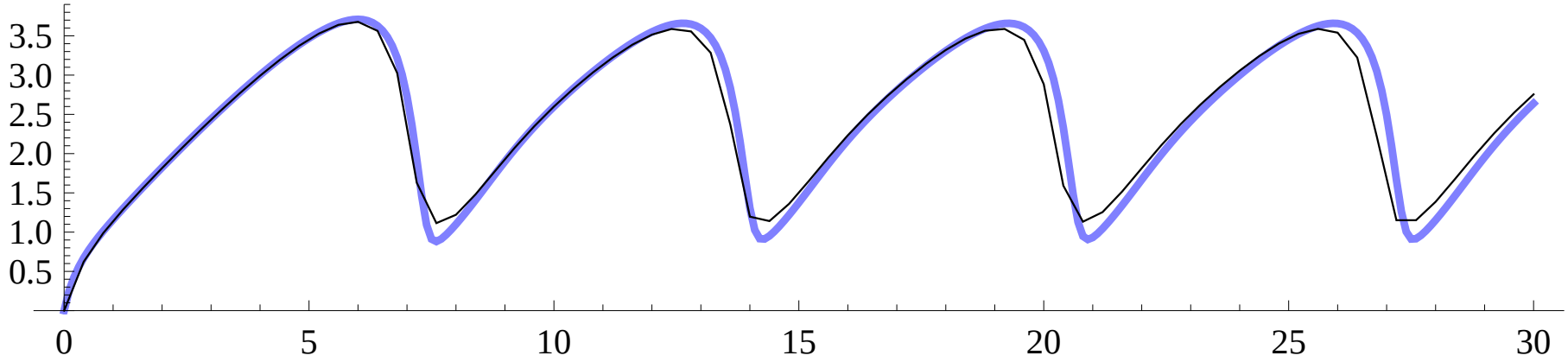
$$\mathbf{C}_{t+1h} = \mathbf{C}_t + \mathbf{k} \cdot h$$

This is the Runge-Kutta method

http://en.wikipedia.org/wiki/Runge-Kutta_methods

Not much more difficult to code

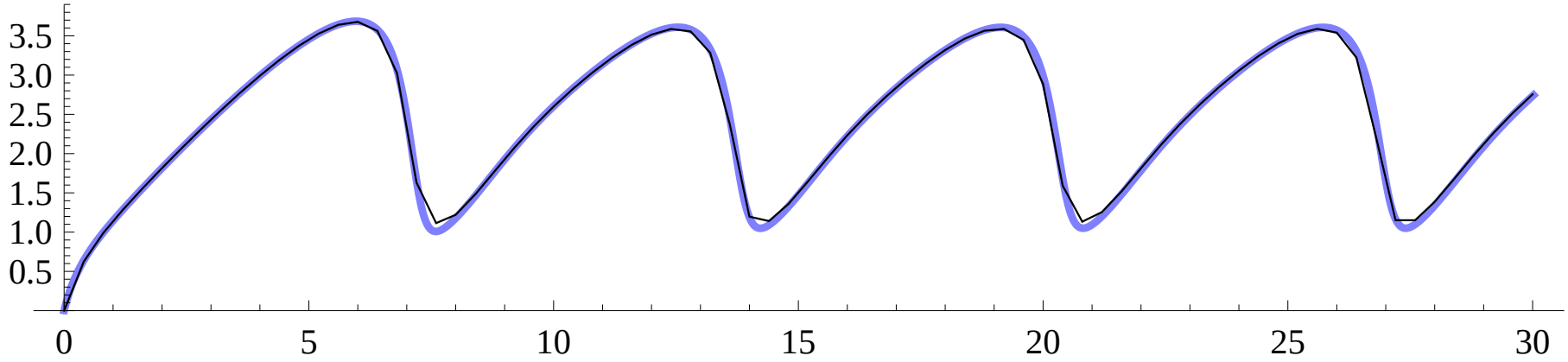
```
 ...
REAL(dp) :: k1(dim_c),k2(dim_c), &
            k3(dim_c),k4(dim_c)
 ...
DO WHILE (t < t2)
   CALL deriv(k1,c,cc,k)
   CALL deriv(k2,c+dt/2*k1,cc,k)
   CALL deriv(k3,c+dt/2*k2,cc,k)
   CALL deriv(k4,c+dt*k3,cc,k)
   dc=(k1 + 2*k2 + 2*k3 + k4)/6
   c = c + dc*dt
   WRITE(*,*) c
   t = t+dt
END DO
```



Blue: Euler, dt=0.1
Black: Runge-Kutta dt=0.4

(So that they should have about the same computational cost)

Blue looks smoother, because time step is smaller, but which is more accurate?



Blue: Euler, dt=0.02
Black: Runge-Kutta dt=0.4 (same as previously)

So taking a very small time step in Euler method, it actually converges to the Runge-Kutta solution with 20× larger step size

Now you should be able to:

– read chemical equations

– write the corresponding system of differential equations

– code numerical solution to the DE’s

Runge-Kutta is by no means very advanced algorithm either. In real work, we use algorithms with adaptive step size, and we use existing libraries and don’t code the solvers ourselves

More information: “numerical analysis” / “numerical methods” / “differential equations” textbooks.
Google: ’ODE solvers’, ’Stiff equations’