Ordinary Differential Equations
Part 1

Ananda Dasgupta

PH3105, Autumn Semester 2017

ODEs in science

ODEs in science

- ODEs are among the most important mathematical structures in science.

ODEs in science

- ODEs are among the most important mathematical structures in science.
- Dynamical systems in physics are described by second order equations of motion

$$m\frac{d^2x}{dt^2} = F\left(x, \frac{dx}{dt}, t\right)$$

ODEs in science

- ODEs are among the most important mathematical structures in science.
- Dynamical systems in physics are described by second order equations of motion

$$m\frac{d^2x}{dt^2} = F\left(x, \frac{dx}{dt}, t\right)$$

- This can be recast as two coupled first order equations

$$\begin{aligned} \frac{dx}{dt} &= \frac{p}{m} \\ \frac{dp}{dt} &= F\left(x, \frac{p}{m}, t\right) \end{aligned}$$

ODEs in science

- ODEs are among the most important mathematical structures in science.
- Dynamical systems in physics are described by second order equations of motion

$$m\frac{d^2x}{dt^2} = F\left(x, \frac{dx}{dt}, t\right)$$

- This can be recast as two coupled first order equations

$$\begin{aligned}\frac{dx}{dt} &= \frac{p}{m}\\ \frac{dp}{dt} &= F\left(x, \frac{p}{m}, t\right)\end{aligned}$$

- More generally

$$\begin{aligned}\dot{q}_i &= \frac{\partial H}{\partial p_i}\\ \dot{p}_i &= -\frac{\partial H}{\partial q_i}\end{aligned}$$

ODEs in science

- ODEs abound in other scientific disciplines as well!

ODEs in science

- ODEs abound in other scientific disciplines as well!
- In biology, for example, we have the famous predator-prey model

ODEs in science

- ODEs abound in other scientific disciplines as well!
- In biology, for example, we have the famous predator-prey model
- also known as the Lotka-Volterra model

ODEs in science

- ODEs abound in other scientific disciplines as well!
- In biology, for example, we have the famous predator-prey model
- also known as the Lotka-Volterra model
- In this prey population x and predator population y obeys

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$

ODEs in science

- ODEs abound in other scientific disciplines as well!
- In biology, for example, we have the famous predator-prey model
- also known as the Lotka-Volterra model
- In this prey population x and predator population y obeys

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$

- Reaction rates in chemistry : for example the rate of the reaction $2H_2 + 2NO \rightarrow N_2 + 2H_2O$
- $$\frac{d}{dt}[H_2] = -k\,[H_2]\,[NO]^2 = \frac{d}{dt}[NO]$$

The initial value problem

First order differential equations

First order differential equations

- First order differential equation (Initial Value Problem):

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

First order differential equations

- First order differential equation (Initial Value Problem):

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

- The Euler algorithm :

$$y(t+h) \approx y(t) + hf(y(t), t)$$

First order differential equations

- First order differential equation (Initial Value Problem):

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

- The Euler algorithm :

$$y(t + h) \approx y(t) + hf(y(t), t)$$

- or more concisely

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

where $t_n \equiv t_0 + nh$, $y_n \equiv y(t_n)$.

First order differential equations

- First order differential equation (Initial Value Problem):

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

- The Euler algorithm :

$$y(t+h) \approx y(t) + hf(y(t), t)$$

- or more concisely

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

  where $t_n \equiv t_0 + nh$, $y_n \equiv y(t_n)$.
- Local error (error per step) is $\mathcal{O}(h^2)$.

First order differential equations

- First order differential equation (Initial Value Problem):

$$\frac{dy}{dt} = f\left(y, t\right) \qquad \text{subject to } y\left(t_0\right) = y_0$$

- The Euler algorithm :

$$y\left(t+h\right) \approx y\left(t\right) + hf\left(y\left(t\right), t\right)$$

- or more concisely

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

  where $t_n \equiv t_0 + nh$, $y_n \equiv y\left(t_n\right)$.
- Local error (error per step) is $\mathcal{O}\left(h^2\right)$.
- Global error in this method is $\mathcal{O}\left(h\right)$.

First order differential equations

- First order differential equation (Initial Value Problem):

$$\frac{dy}{dt} = f\left(y, t\right) \qquad \text{subject to } y\left(t_0\right) = y_0$$

- The Euler algorithm :

$$y\left(t+h\right) \approx y\left(t\right) + hf\left(y\left(t\right), t\right)$$

- or more concisely

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

  where $t_n \equiv t_0 + nh$, $y_n \equiv y\left(t_n\right)$.
- Local error (error per step) is $\mathcal{O}\left(h^2\right)$.
- Global error in this method is $\mathcal{O}\left(h\right)$.
- For a system of first order ODEs

$$\frac{dy_i}{dt} = f_i\left(y_1, \ldots, y_n; t\right), \qquad i = 1, 2, \ldots n$$

  we have

$$y_i\left(t+h\right) \approx y_i\left(t\right) + hf_i\left(y_1\left(t\right), \ldots, y_n\left(t\right); t\right)$$

Improved integration methods

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y, t) = f(t)$ depends only on t.

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y, t) = f(t)$ depends only on t.
- Then

$$y_{n+1} \equiv y(t_n + h) \approx y(t_n) + hf\left(t_n + \frac{h}{2}\right)$$

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y, t) = f(t)$ depends only on t.
- Then

$$y_{n+1} \equiv y(t_n + h) \approx y(t_n) + hf\left(t_n + \frac{h}{2}\right)$$

- This is better:

$$\begin{aligned}
\text{RHS} &= y(t_n) + h\left(f(t_n) + \frac{h}{2}\dot{f}(t_n) + \mathcal{O}\left(h^2\right)\right) \\
&= y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \mathcal{O}\left(h^3\right) \\
&= y(t_n + h) + \mathcal{O}\left(h^3\right)
\end{aligned}$$

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y, t) = f(t)$ depends only on t.
- Then

$$y_{n+1} \equiv y(t_n + h) \approx y(t_n) + hf\left(t_n + \frac{h}{2}\right)$$

- This is better:

$$\begin{aligned} \mathrm{RHS} &= y(t_n) + h\left(f(t_n) + \frac{h}{2}\dot{f}(t_n) + \mathcal{O}(h^2)\right) \\ &= y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \mathcal{O}(h^3) \\ &= y(t_n + h) + \mathcal{O}(h^3) \end{aligned}$$

- Local error $\mathcal{O}(h^3)$,

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y,t) = f(t)$ depends only on t.
- Then

$$y_{n+1} \equiv y(t_n + h) \approx y(t_n) + hf\left(t_n + \frac{h}{2}\right)$$

- This is better:

$$\begin{aligned}
\text{RHS} &= y(t_n) + h\left(f(t_n) + \frac{h}{2}\dot{f}(t_n) + \mathcal{O}\left(h^2\right)\right) \\
&= y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \mathcal{O}\left(h^3\right) \\
&= y(t_n + h) + \mathcal{O}\left(h^3\right)
\end{aligned}$$

- Local error $\mathcal{O}\left(h^3\right)$, global error $\mathcal{O}\left(h^2\right)$.

Improved integration methods

- The Euler algorithm uses the derivative at time t_n as an approximation for the rate of change over the entire interval from t_n to $t_n + h$.
- Wouldn't it be better if we used the derivative at the midpoint $t_n + h/2$?
- Easy to do if the derivative $f(y, t) = f(t)$ depends only on t.
- Then

$$y_{n+1} \equiv y(t_n + h) \approx y(t_n) + hf\left(t_n + \frac{h}{2}\right)$$

- This is better:

$$\begin{aligned} \mathrm{RHS} &= y(t_n) + h\left(f(t_n) + \frac{h}{2}\dot{f}(t_n) + \mathcal{O}\left(h^2\right)\right) \\ &= y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \mathcal{O}\left(h^3\right) \\ &= y(t_n + h) + \mathcal{O}\left(h^3\right) \end{aligned}$$

- Local error $\mathcal{O}\left(h^3\right)$, global error $\mathcal{O}\left(h^2\right)$.
- What if $f(y, t)$ depends explicitly on y (as it usually would)?

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2}p_n,\ t_n + \frac{h}{2}\right)$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2} p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = f + \frac{h}{2}\left(\frac{\partial f}{\partial y}\, p_n + \frac{\partial f}{\partial t}\right) + \frac{h^2}{8}\left(\frac{\partial^2 f}{\partial y^2}\, p_n^2 + 2\frac{\partial^2 f}{\partial y \partial t}\, p_n + \frac{\partial^2 f}{\partial t^2}\right)\Bigg|_{(y_n, t_n)} + O\left(h^3\right)$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2} p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = f + \frac{h}{2}\left(\frac{\partial f}{\partial y}\, p_n + \frac{\partial f}{\partial t}\right) + \frac{h^2}{8}\left(\frac{\partial^2 f}{\partial y^2}\, p_n^2 + 2\frac{\partial^2 f}{\partial y \partial t}\, p_n + \frac{\partial^2 f}{\partial t^2}\right)\Bigg|_{(y_n, t_n)} + O\left(h^3\right)$$

But

$$\dot{y}\left(t\right) = f\left(y, t\right)$$
$$\ddot{y}\left(t\right) = \frac{\partial f}{\partial y}\dot{y} + \frac{\partial f}{\partial t}$$
$$\dddot{y}\left(t\right) = \frac{\partial^2 f}{\partial y^2}\dot{y}^2 + \frac{\partial^2 f}{\partial t \partial y}\dot{y} + \frac{\partial f}{\partial y}\ddot{y} + \frac{\partial^2 f}{\partial y \partial t}\dot{y} + \frac{\partial^2 f}{\partial t^2}$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f(y_n, t_n)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2} p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = f + \frac{h}{2}\left(\frac{\partial f}{\partial y} p_n + \frac{\partial f}{\partial t}\right) + \frac{h^2}{8}\left(\frac{\partial^2 f}{\partial y^2} p_n^2 + 2\frac{\partial^2 f}{\partial y \partial t} p_n + \frac{\partial^2 f}{\partial t^2}\right)\Bigg|_{(y_n, t_n)} + O\left(h^3\right)$$

But

$$\begin{aligned}
\dot{y}_n &= f(y_n, t_n) = p_n \\
\ddot{y}_n &= \frac{\partial f}{\partial y}\dot{y} + \frac{\partial f}{\partial t}\Bigg|_{(y_n, t_n)} = \frac{\partial f}{\partial y} p_n + \frac{\partial f}{\partial t}\Bigg|_{(y_n, t_n)} \\
\dddot{y}_n &= \frac{\partial^2 f}{\partial y^2} p_n^2 + 2\frac{\partial^2 f}{\partial y \partial t} p_n + \frac{\partial f}{\partial y}\ddot{y}_n + \frac{\partial^2 f}{\partial t^2}\Bigg|_{(y_n, t_n)}
\end{aligned}$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2}p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = \dot{y}_n + \frac{h}{2}\ddot{y}_n + \frac{h^2}{8}\left(\dddot{y}_n - \frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n, t_n)} + \mathcal{O}\left(h^3\right)$$

But

$$\begin{aligned}
\dot{y}_n &= f\left(y_n, t_n\right) = p_n \\
\ddot{y}_n &= \frac{\partial f}{\partial y}\dot{y} + \left.\frac{\partial f}{\partial t}\right|_{(y_n, t_n)} = \frac{\partial f}{\partial y}p_n + \left.\frac{\partial f}{\partial t}\right|_{(y_n, t_n)} \\
\dddot{y}_n &= \frac{\partial^2 f}{\partial y^2}p_n^2 + 2\frac{\partial^2 f}{\partial y \partial t}p_n + \frac{\partial f}{\partial y}\ddot{y}_n + \left.\frac{\partial^2 f}{\partial t^2}\right|_{(y_n, t_n)}
\end{aligned}$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2}p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = \dot{y}_n + \frac{h}{2}\ddot{y}_n + \frac{h^2}{8}\left(\dddot{y}_n - \frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n, t_n)} + \mathcal{O}\left(h^3\right)$$

- On the other hand,

$$y_{n+1} \equiv y\left(t_n + h\right) \quad = \quad y_n + h\dot{y}_n + \frac{h^2}{2!}\ddot{y}_n + \frac{h^3}{3!}\dddot{y}_n + \mathcal{O}\left(h^4\right)$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f\left(y_n, t_n\right)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2}p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = \dot{y}_n + \frac{h}{2}\ddot{y}_n + \frac{h^2}{8}\left(\dddot{y}_n - \frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n,t_n)} + \mathcal{O}\left(h^3\right)$$

- On the other hand,

$$\begin{aligned} y_{n+1} \equiv y\left(t_n + h\right) &= y_n + h\dot{y}_n + \frac{h^2}{2!}\ddot{y}_n + \frac{h^3}{3!}\dddot{y}_n + \mathcal{O}\left(h^4\right) \\ &= y_n + hq_n \\ &\quad + \frac{h^3}{24}\left(\dddot{y}_n + 3\frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n,t_n)} + \mathcal{O}\left(h^4\right) \end{aligned}$$

The Midpoint Euler method (explicit version)

- Slope at beginning : $p_n = f(y_n, t_n)$
- Approximate $f\left(y\left(t_n + \frac{h}{2}\right), t_n + \frac{h}{2}\right)$ by

$$q_n = f\left(y_n + \frac{h}{2}p_n,\ t_n + \frac{h}{2}\right)$$

- Taylor series in several variables says that

$$q_n = \dot{y}_n + \frac{h}{2}\ddot{y}_n + \frac{h^2}{8}\left(\dddot{y}_n - \frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n,t_n)} + \mathcal{O}\left(h^3\right)$$

- On the other hand,

$$\begin{aligned} y_{n+1} \equiv y\left(t_n + h\right) &= y_n + h\dot{y}_n + \frac{h^2}{2!}\ddot{y}_n + \frac{h^3}{3!}\dddot{y}_n + \mathcal{O}\left(h^4\right) \\ &= y_n + hq_n \\ &\quad + \frac{h^3}{24}\left(\dddot{y}_n + 3\frac{\partial f}{\partial y}\ddot{y}_n\right)_{(y_n,t_n)} + \mathcal{O}\left(h^4\right) \end{aligned}$$

- The midpoint method has a local error of $O\left(h^3\right)$ - global error $O\left(h^2\right)$.

Generalization : the Runge-Kutta method

Generalization : the Runge-Kutta method

- Instead of the derivative at midpoint, we could use

$$\frac{p_n + q_n}{2} = \frac{1}{2} f\left(y_n, t_n\right) + \frac{1}{2} f\left(y_n + h p_n, t_n + h\right)$$

- aka the modified Euler method.

Generalization : the Runge-Kutta method

- Instead of the derivative at midpoint, we could use

$$\frac{p_n + q_n}{2} = \frac{1}{2} f\left(y_n, t_n\right) + \frac{1}{2} f\left(y_n + hp_n, t_n + h\right)$$

  - aka the modified Euler method.
- A more general approximation

$$\beta_1 p_n + \beta_2 q_n$$

  where $p_n = f\left(y_n, t_n\right) = \dot{y}_n$ and $q = f\left(y_n + \alpha h p_n, t_n + \alpha h\right)$.

Generalization : the Runge-Kutta method

- Instead of the derivative at midpoint, we could use

$$\frac{p_n + q_n}{2} = \frac{1}{2} f\left(y_n, t_n\right) + \frac{1}{2} f\left(y_n + hp_n, t_n + h\right)$$

  - aka the modified Euler method.

- A more general approximation

$$\beta_1 p_n + \beta_2 q_n$$

  where $p_n = f\left(y_n, t_n\right) = \dot{y}_n$ and $q = f\left(y_n + \alpha h p_n, t_n + \alpha h\right)$.

- Taylor series says

$$q_n = p_n + \alpha h \left(f_y \dot{y}_n + f_t\right)_{(y_n, t_n)} + \frac{\alpha^2 h^2}{2} \left(f_{yy} p_n^2 + 2 f_{yt} p_n + f_{tt}\right)_{(y_n, t_n)}$$

Generalization : the Runge-Kutta method

- Instead of the derivative at midpoint, we could use

$$\frac{p_n + q_n}{2} = \frac{1}{2} f\left(y_n, t_n\right) + \frac{1}{2} f\left(y_n + h p_n, t_n + h\right)$$

- aka the modified Euler method.

- A more general approximation

$$\beta_1 p_n + \beta_2 q_n$$

where $p_n = f\left(y_n, t_n\right) = \dot{y}_n$ and $q = f\left(y_n + \alpha h p_n, t_n + \alpha h\right)$.

- Taylor series says

$$\begin{aligned} q_n &= p_n + \alpha h \left(f_y \dot{y}_n + f_t\right)_{(y_n, t_n)} + \frac{\alpha^2 h^2}{2}\left(f_{yy} p_n^2 + 2 f_{yt} p_n + f_{tt}\right)_{(y_n, t_n)} \\ &= \dot{y}_n + \alpha h \ddot{y}_n + \frac{\alpha^2 h^2}{2}\left[\dddot{y}_n - f_y \ddot{y}_n\right]_{(y_n, t_n)} + \mathcal{O}\left(h^3\right) \end{aligned}$$

Generalization : the Runge-Kutta method

Generalization : the Runge-Kutta method

- The approximation

$$y(t_n + h) = y_n + h(\beta_1 p_n + \beta_2 q_n)$$

Generalization : the Runge-Kutta method

- The approximation

$$
\begin{aligned}
y\left(t_n+h\right) &= y_n + h\left(\beta_1 p_n + \beta_2 q_n\right) \\
&+ \left(1-\beta_1-\beta_2\right) h\dot{y}_n + \left(1-2\alpha\beta_2\right)\frac{h^2}{2}\ddot{y}_n \\
&+ \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y\ddot{y}_n\right)_{(y_n,t_n)}\right] + \mathcal{O}\left(h^4\right)
\end{aligned}
$$

Generalization : the Runge-Kutta method

- The approximation

$$\begin{aligned} y\left(t_n + h\right) &= y_n + h\left(\beta_1 p_n + \beta_2 q_n\right) \\ &\quad + \left(1 - \beta_1 - \beta_2\right) h\dot{y}_n + \left(1 - 2\alpha\beta_2\right) \frac{h^2}{2} \ddot{y}_n \\ &\quad + \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y \ddot{y}_n\right)_{(y_n, t_n)}\right] + \mathcal{O}\left(h^4\right) \end{aligned}$$

- Minimizing error requires

$$\begin{aligned} \beta_1 + \beta_2 &= 1 \\ 2\alpha\beta_2 &= 1 \end{aligned}$$

giving a local error of $\mathcal{O}\left(h^3\right)$.

Generalization : the Runge-Kutta method

- The approximation

$$
\begin{aligned}
y(t_n + h) &= y_n + h(\beta_1 p_n + \beta_2 q_n) \\
&\quad + (1 - \beta_1 - \beta_2) h\dot{y}_n + (1 - 2\alpha\beta_2)\frac{h^2}{2}\ddot{y}_n \\
&\quad + \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y\ddot{y}_n\right)_{(y_n, t_n)}\right] + \mathcal{O}\left(h^4\right)
\end{aligned}
$$

- Minimizing error requires

$$
\begin{aligned}
\beta_1 + \beta_2 &= 1 \\
2\alpha\beta_2 &= 1
\end{aligned}
$$

giving a local error of $\mathcal{O}\left(h^3\right)$.

- Possibilities :

Generalization : the Runge-Kutta method

- The approximation

$$\begin{aligned} y(t_n+h) &= y_n + h(\beta_1 p_n + \beta_2 q_n) \\ &\quad + (1-\beta_1-\beta_2)\, h\dot{y}_n + (1-2\alpha\beta_2)\frac{h^2}{2}\ddot{y}_n \\ &\quad + \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y\ddot{y}_n\right)_{(y_n,t_n)}\right] + \mathcal{O}\left(h^4\right) \end{aligned}$$

- Minimizing error requires

$$\begin{aligned} \beta_1+\beta_2 &= 1 \\ 2\alpha\beta_2 &= 1 \end{aligned}$$

giving a local error of $\mathcal{O}\left(h^3\right)$.

- Possibilities :
  - $\alpha = \frac{1}{2}, \beta_1 = 0, \beta_2 = 1$ - the midpoint formula.

Generalization : the Runge-Kutta method

- The approximation

$$
\begin{aligned}
y\left(t_n+h\right) &= y_n + h\left(\beta_1 p_n + \beta_2 q_n\right) \\
&\quad + \left(1-\beta_1-\beta_2\right) h\dot{y}_n + \left(1-2\alpha\beta_2\right)\frac{h^2}{2}\ddot{y}_n \\
&\quad + \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y\ddot{y}_n\right)_{(y_n,t_n)}\right] + \mathcal{O}\left(h^4\right)
\end{aligned}
$$

- Minimizing error requires

$$
\begin{aligned}
\beta_1+\beta_2 &= 1 \\
2\alpha\beta_2 &= 1
\end{aligned}
$$

giving a local error of $\mathcal{O}\left(h^3\right)$.

- Possibilities :
  - $\alpha=\frac{1}{2}, \beta_1=0, \beta_2=1$ - the midpoint formula.
  - $\alpha=1, \beta_1=\beta_2=\frac{1}{2}$ - the corrected Euler method, aka Heun's method.

Generalization : the Runge-Kutta method

- The approximation

$$
\begin{aligned}
y(t_n + h) &= y_n + h(\beta_1 p_n + \beta_2 q_n) \\
&\quad + (1 - \beta_1 - \beta_2) h\dot{y}_n + (1 - 2\alpha\beta_2)\frac{h^2}{2}\ddot{y}_n \\
&\quad + \frac{h^3}{6}\left[\dddot{y} - 3\alpha^2\beta_2\left(\dddot{y}_n - f_y\ddot{y}_n\right)_{(y_n,t_n)}\right] + \mathcal{O}\left(h^4\right)
\end{aligned}
$$

- Minimizing error requires

$$
\begin{aligned}
\beta_1 + \beta_2 &= 1 \\
2\alpha\beta_2 &= 1
\end{aligned}
$$

giving a local error of $\mathcal{O}\left(h^3\right)$.

- Possibilities :
  - $\alpha = \frac{1}{2}, \beta_1 = 0, \beta_2 = 1$ - the midpoint formula.
  - $\alpha = 1, \beta_1 = \beta_2 = \frac{1}{2}$ - the corrected Euler method, aka Heun's method.
  - $\alpha = \frac{2}{3}, \beta_1 = \frac{1}{4}, \beta_2 = \frac{3}{4}$ - Ralstone's method.

The 4th order Runge-Kutta algorithm

The 4th order Runge-Kutta algorithm

- Approximate the derivative in the interval $(t_n, t_n + h)$ by the weighted average

$$\frac{1}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$

The 4th order Runge-Kutta algorithm

- Approximate the derivative in the interval $(t_n, t_n + h)$ by the weighted average
$$\frac{1}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$
- The approximants are
$$\begin{aligned}
p_n &= f\left(y_n, t_n\right)\\
q_n &= f\left(y_n + \frac{h}{2}p_n, t_n + \frac{h}{2}\right)\\
r_n &= f\left(y_n + \frac{h}{2}q_n, t_n + \frac{h}{2}\right)\\
s_n &= f\left(y_n + hr, t_n + h\right)
\end{aligned}$$

The 4th order Runge-Kutta algorithm

- Approximate the derivative in the interval $(t_n, t_n + h)$ by the weighted average
$$\frac{1}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$
- The approximants are
$$\begin{aligned}
p_n &= f\left(y_n, t_n\right) \\
q_n &= f\left(y_n + \frac{h}{2}p_n, t_n + \frac{h}{2}\right) \\
r_n &= f\left(y_n + \frac{h}{2}q_n, t_n + \frac{h}{2}\right) \\
s_n &= f\left(y_n + hr, t_n + h\right)
\end{aligned}$$
- Thus
$$y_{n+1} \approx y_n + \frac{h}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$

The 4th order Runge-Kutta algorithm

- Approximate the derivative in the interval $(t_n, t_n + h)$ by the weighted average

$$\frac{1}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$

- The approximants are

$$\begin{aligned}
p_n &= f\left(y_n, t_n\right) \\
q_n &= f\left(y_n + \frac{h}{2}p_n, t_n + \frac{h}{2}\right) \\
r_n &= f\left(y_n + \frac{h}{2}q_n, t_n + \frac{h}{2}\right) \\
s_n &= f\left(y_n + hr, t_n + h\right)
\end{aligned}$$

- Thus

$$y_{n+1} \approx y_n + \frac{h}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$

- This has a local error of $\mathcal{O}\left(h^5\right)$ and global error of $\mathcal{O}\left(h^4\right)$.

The 4th order Runge-Kutta algorithm

- Approximate the derivative in the interval $(t_n, t_n + h)$ by the weighted average
$$\frac{1}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$
- The approximants are
$$\begin{aligned} p_n &= f\left(y_n, t_n\right) \\ q_n &= f\left(y_n + \frac{h}{2}p_n, t_n + \frac{h}{2}\right) \\ r_n &= f\left(y_n + \frac{h}{2}q_n, t_n + \frac{h}{2}\right) \\ s_n &= f\left(y_n + hr, t_n + h\right) \end{aligned}$$
- Thus
$$y_{n+1} \approx y_n + \frac{h}{6}\left(p_n + 2q_n + 2r_n + s_n\right)$$
- This has a local error of $\mathcal{O}\left(h^5\right)$ and global error of $\mathcal{O}\left(h^4\right)$.
- Both simple and accurate - arguably the most widely used method.

How big should the step size be?

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?
- One approach - the step size can be changed depending on the error at each step.

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?
- One approach - the step size can be changed depending on the error at each step.
- But ... how to estimate the error?

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?
- One approach - the step size can be changed depending on the error at each step.
- But ... how to estimate the error?
- Use one RK4 step with step-size h and two RK4 steps with step-size $\frac{h}{2}$ to get two estimates for $y\left(t_n + h\right)$ - $y_{(1)}$ and $y_{(2)}$, respectively.

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?
- One approach - the step size can be changed depending on the error at each step.
- But ... how to estimate the error?
- Use one RK4 step with step-size h and two RK4 steps with step-size $\frac{h}{2}$ to get two estimates for $y\left(t_n + h\right)$ - $y_{(1)}$ and $y_{(2)}$, respectively.
- The difference $\Delta = y_{(2)} - y_{(1)}$ is an estimate for the error.

How big should the step size be?

- The local error in RK4 is $\mathcal{O}\left(h^5\right)$.
- How big must h be for a desired accuracy ϵ?
- One approach - the step size can be changed depending on the error at each step.
- But ... how to estimate the error?
- Use one RK4 step with step-size h and two RK4 steps with step-size $\frac{h}{2}$ to get two estimates for $y\left(t_n + h\right)$ - $y_{(1)}$ and $y_{(2)}$, respectively.
- The difference $\Delta = y_{(2)} - y_{(1)}$ is an estimate for the error.
- In fact, this also yields a better estimate for $y\left(t_n + h\right)$, namely $y_{(2)} + \frac{\Delta}{15}$!

How big should the step size be?

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to

$$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$

and repeat.

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to

$$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$

  and repeat.
- Warning : guard against the step-size becoming too small!

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to

  $$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$

  and repeat.
- Warning : guard against the step-size becoming too small!
- Once the step size decreases, it stays small - may make the method very slow.

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to

  $$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$

  and repeat.
- Warning : guard against the step-size becoming too small!
- Once the step size decreases, it stays small - may make the method very slow.
- A modification by Press *et al* - at each step,

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to

$$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$

  and repeat.
- Warning : guard against the step-size becoming too small!
- Once the step size decreases, it stays small - may make the method very slow.
- A modification by Press *et al* - at each step,
  - accept the result if $|\Delta| < \epsilon$.

How big should the step size be?

- If $|\Delta| < \epsilon$, accept the step.
- If $|\Delta| > \epsilon$, change the step-size to
$$h\left(\frac{\epsilon}{|\Delta|}\right)^{1/5}$$
and repeat.
- Warning : guard against the step-size becoming too small!
- Once the step size decreases, it stays small - may make the method very slow.
- A modification by Press *et al* - at each step,
  - accept the result if $|\Delta| < \epsilon$.
  - In each step , define the new step-size as
$$Rh\left(\frac{\epsilon}{|\Delta|}\right)^{\eta}$$
where $R \sim 0.9$, and
$$\eta = \begin{cases} 0.25 & \text{where } \Delta > \epsilon \\ 0.20 & \text{where } \Delta < \epsilon \end{cases}$$

Second order ODEs : The Runge-Kutta-Nystrom method

Second order ODEs : The Runge-Kutta-Nystrom method

For the second order equation

$$\ddot{y} = f\left(y, \dot{y}, t\right)$$

Second order ODEs : The Runge-Kutta-Nystrom method

For the second order equation

$$\ddot{y} = f\left(y, \dot{y}, t\right)$$

we can use

$$\begin{aligned} y_{n+1} &= y_n + h\left(\dot{y}_n + \frac{h}{3}\left[A_n + B_n + C_n\right]\right) \\ \dot{y}_{n+1} &= \dot{y}_n + \frac{1}{3}\left[A_n + 2B_n + 2C_n + D_n\right] \end{aligned}$$

Second order ODEs : The Runge-Kutta-Nystrom method

For the second order equation

$$\ddot{y} = f\left(y, \dot{y}, t\right)$$

we can use

$$\begin{aligned} y_{n+1} &= y_n + h\left(\dot{y}_n + \frac{h}{3}\left[A_n + B_n + C_n\right]\right) \\ \dot{y}_{n+1} &= \dot{y}_n + \frac{1}{3}\left[A_n + 2B_n + 2C_n + D_n\right] \end{aligned}$$

where

$$\begin{aligned} A_n &= \frac{h}{2} f\left(y_n, \dot{y}_n, t_n\right) \\ B_n &= \frac{h}{2} f\left(y_n + \beta_n, \dot{y}_n + A_n, t_n + \frac{h}{2}\right), \text{ where } \beta_n = \frac{h}{2}\left(\dot{y}_n + \frac{A_n}{2}\right) \\ C_n &= \frac{h}{2} f\left(y_n + \beta_n, \dot{y}_n + B_n, t_n + \frac{h}{2}\right) \\ D_n &= \frac{h}{2} f\left(y_n + \delta_n, \dot{y}_n + 2C_n, t_n + h\right), \text{ where } \delta_n = h\left(\dot{y}_n + C_n\right) \end{aligned}$$

Second order ODEs : The Runge-Kutta-Nystrom method

For the second order equation

$$\ddot{y} = f\left(y, \dot{y}, t\right)$$

we can use

$$\begin{aligned}
y_{n+1} &= y_n + h\left(\dot{y}_n + \frac{h}{3}\left[A_n + B_n + C_n\right]\right) \\
\dot{y}_{n+1} &= \dot{y}_n + \frac{1}{3}\left[A_n + 2B_n + 2C_n + D_n\right]
\end{aligned}$$

where

$$\begin{aligned}
A_n &= \frac{h}{2} f\left(y_n, \dot{y}_n, t_n\right) \\
B_n &= \frac{h}{2} f\left(y_n + \beta_n, \dot{y}_n + A_n, t_n + \frac{h}{2}\right), \text{ where } \beta_n = \frac{h}{2}\left(\dot{y}_n + \frac{A_n}{2}\right) \\
C_n &= \frac{h}{2} f\left(y_n + \beta_n, \dot{y}_n + B_n, t_n + \frac{h}{2}\right) \\
D_n &= \frac{h}{2} f\left(y_n + \delta_n, \dot{y}_n + 2C_n, t_n + h\right), \text{ where } \delta_n = h\left(\dot{y}_n + C_n\right)
\end{aligned}$$

System of equations

- A system of differential equations

$$\dot{y}_i\left(t\right) = f_i\left(y_1, \ldots y_n; t\right), \qquad i = 1, 2, \ldots, n$$

  can be handled by the RK4
- However care has to be used to evaluate all the derivatives together.
- The n initial slopes $p_1, \ldots p_n$must be evaluated at the values of $y_1, \ldots, y_n$at the beginning of the interval
- These values must be used to *predict* the values of all the y_i at the middle of the interval to get the $q_1, \ldots, q_n$
- and so on ...
- Using numpy helps greatly here!