Runge Kutta Methods

Bernd Schröder

Errors in Euler's Method

Errors in Euler's Method

1. **Taylor's Formula**. If the function y is $n+1$ times differentiable, then for any h there is a c between x and $x+h$ so that

$$y(x+h)=y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\cdots+\frac{y^{(n)}(x)}{n!}h^n+\frac{y^{(n+1)}(c)}{(n+1)!}h^{n+1}.$$

Errors in Euler's Method

1. **Taylor's Formula**. If the function y is $n+1$ times differentiable, then for any h there is a c between x and $x+h$ so that

$$y(x+h)=y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\cdots+\frac{y^{(n)}(x)}{n!}h^n+\frac{y^{(n+1)}(c)}{(n+1)!}h^{n+1}.$$

2. **Euler's method**. For $y'=F(x,y)$, $y(x)=y_0$ we use that

$$y(x+\Delta x) \approx y(x)+y'(x)\Delta x$$

Errors in Euler's Method

1. **Taylor's Formula**. If the function y is $n+1$ times differentiable, then for any h there is a c between x and $x+h$ so that

$$y(x+h)=y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\cdots+\frac{y^{(n)}(x)}{n!}h^n+\frac{y^{(n+1)}(c)}{(n+1)!}h^{n+1}.$$

2. **Euler's method**. For $y'=F(x,y)$, $y(x)=y_0$ we use that

$$\begin{aligned} y(x+\Delta x) &\approx y(x)+y'(x)\Delta x \\ &= y_0 \end{aligned}$$

Errors in Euler's Method

1. **Taylor's Formula**. If the function y is $n+1$ times differentiable, then for any h there is a c between x and $x+h$ so that

$$y(x+h)=y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\cdots+\frac{y^{(n)}(x)}{n!}h^n+\frac{y^{(n+1)}(c)}{(n+1)!}h^{n+1}.$$

2. **Euler's method**. For $y'=F(x,y)$, $y(x)=y_0$ we use that

$$\begin{aligned} y(x+\Delta x) &\approx y(x)+y'(x)\Delta x \\ &= y_0+F(x,y_0)\Delta x \end{aligned}$$

Errors in Euler's Method

1. **Taylor's Formula**. If the function y is $n+1$ times differentiable, then for any h there is a c between x and $x+h$ so that

$$y(x+h)=y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\cdots+\frac{y^{(n)}(x)}{n!}h^n+\frac{y^{(n+1)}(c)}{(n+1)!}h^{n+1}.$$

2. **Euler's method**. For $y'=F(x,y)$, $y(x)=y_0$ we use that

$$\begin{aligned} y(x+\Delta x) &\approx y(x)+y'(x)\Delta x \\ &= y_0+F(x,y_0)\Delta x \\ &=: y_{\text{Euler}}(x+\Delta x) \end{aligned}$$

Errors in Euler's Method

Errors in Euler's Method

3. But we know that

Errors in Euler's Method

3. But we <u>know</u> that

$$y(x+\Delta x) \quad =$$

Errors in Euler's Method

3. But we know that

$$y(x+\Delta x) = y(x)$$

Errors in Euler's Method

3. But we know that

$$y(x+\Delta x) = y(x) + y'(x)\Delta x$$

Errors in Euler's Method

3. But we know that

$$y(x+\Delta x) = y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2$$

Errors in Euler's Method

3. But we know that

$$\begin{aligned} y(x+\Delta x) &= y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2 \\ &= y_{\text{Euler}}(x+\Delta x) \end{aligned}$$

Errors in Euler's Method

3. But we know that

$$\begin{aligned} y(x+\Delta x) &= y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2 \\ &= y_{\text{Euler}}(x+\Delta x)+\frac{y''(c)}{2!}(\Delta x)^2 \end{aligned}$$

Errors in Euler's Method

3. But we know that

$$
\begin{aligned}
y(x+\Delta x) &= y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2 \\
&= y_{\text{Euler}}(x+\Delta x)+\frac{y''(c)}{2!}(\Delta x)^2
\end{aligned}
$$

4. So the error in each step is proportional to $(\Delta x)^2$.

Errors in Euler's Method

3. But we know that

$$\begin{aligned} y(x+\Delta x) &= y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2 \\ &= y_{\text{Euler}}(x+\Delta x)+\frac{y''(c)}{2!}(\Delta x)^2 \end{aligned}$$

4. So the error in each step is proportional to $(\Delta x)^2$.
5. Summing the errors for $\dfrac{b-a}{\Delta x}$ steps gives an overall error proportional to Δx.

Errors in Euler's Method

3. But we know that

$$\begin{aligned} y(x+\Delta x) &= y(x)+y'(x)\Delta x+\frac{y''(c)}{2!}(\Delta x)^2 \\ &= y_{\text{Euler}}(x+\Delta x)+\frac{y''(c)}{2!}(\Delta x)^2 \end{aligned}$$

4. So the error in each step is proportional to $(\Delta x)^2$.
5. Summing the errors for $\frac{b-a}{\Delta x}$ steps gives an overall error proportional to Δx. (Details are more subtle than it looks.)

How can we Shrink the Error?

How can we Shrink the Error?

1. Shrinking Δx is costly.

How can we Shrink the Error?

1. Shrinking Δx is costly.
2. So a formula with a smaller error would be nice.

How can we Shrink the Error?

1. Shrinking Δx is costly.
2. So a formula with a smaller error would be nice.
3. The global error's proportionality to Δx in Euler's method came from the fact that Euler's method uses the first two terms of the Taylor expansion.

How can we Shrink the Error?

1. Shrinking Δx is costly.
2. So a formula with a smaller error would be nice.
3. The global error's proportionality to Δx in Euler's method came from the fact that Euler's method uses the first two terms of the Taylor expansion.
4. If we can capture more than the first two terms of the Taylor expansion, we could get a global error proportional to $(\Delta x)^n$.

How can we Shrink the Error?

1. Shrinking Δx is costly.
2. So a formula with a smaller error would be nice.
3. The global error's proportionality to Δx in Euler's method came from the fact that Euler's method uses the first two terms of the Taylor expansion.
4. If we can capture more than the first two terms of the Taylor expansion, we could get a global error proportional to $(\Delta x)^n$. This would be good, because Δx is usually small, so a higher power of Δx would be even smaller.

Improving Euler's Method

Improving Euler's Method

We'll get y'' from the expansion for y'.

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$y'(x+h) = y'(x) + y''(x)h + \frac{y'''(\tilde{c})}{2!}h^2$$

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$
\begin{aligned}
y'(x+h) &= y'(x)+y''(x)h+\frac{y'''(\tilde{c})}{2!}h^2 \\
y''(x) &= \frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h
\end{aligned}
$$

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$\begin{aligned} y'(x+h) &= y'(x)+y''(x)h+\frac{y'''(\tilde{c})}{2!}h^2 \\ y''(x) &= \frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h \\ y(x+h) &= y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\frac{y'''(c)}{3!}h^3 \end{aligned}$$

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$
\begin{aligned}
y'(x+h) &= y'(x)+y''(x)h+\frac{y'''(\tilde{c})}{2!}h^2 \\
y''(x) &= \frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h \\
y(x+h) &= y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+y'(x)h+\frac{1}{2}h^2\left(\frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h\right)+\frac{y'''(c)}{3!}h^3
\end{aligned}
$$

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$
\begin{aligned}
y'(x+h) &= y'(x)+y''(x)h+\frac{y'''(\tilde{c})}{2!}h^2 \\
y''(x) &= \frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h \\
y(x+h) &= y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+y'(x)h+\frac{1}{2}h^2\left(\frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h\right)+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+y'(x)h+\frac{1}{2}y'(x+h)h-\frac{1}{2}y'(x)h-\frac{1}{2}\frac{y'''(\tilde{c})}{2!}h^3+\frac{y'''(c)}{3!}h^3
\end{aligned}
$$

Improving Euler's Method

We'll get y'' from the expansion for y'.

$$
\begin{aligned}
y'(x+h) &= y'(x)+y''(x)h+\frac{y'''(\tilde{c})}{2!}h^2 \\
y''(x) &= \frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h \\
y(x+h) &= y(x)+y'(x)h+\frac{y''(x)}{2!}h^2+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+y'(x)h+\frac{1}{2}h^2\left(\frac{y'(x+h)-y'(x)}{h}-\frac{y'''(\tilde{c})}{2!}h\right)+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+y'(x)h+\frac{1}{2}y'(x+h)h-\frac{1}{2}y'(x)h-\frac{1}{2}\frac{y'''(\tilde{c})}{2!}h^3+\frac{y'''(c)}{3!}h^3 \\
&= y(x)+\left(\frac{1}{2}y'(x)+\frac{1}{2}y'(x+h)\right)h+\left(\frac{y'''(c)}{3!}-\frac{y'''(\tilde{c})}{4}\right)h^3
\end{aligned}
$$

Improved Euler Method

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$x_{n+1} := x_n + \Delta x$$

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

The value y_n will be an approximation for the value of the solution y at x_n.

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

The value y_n will be an approximation for the value of the solution y at x_n. (Formulated to make the transition to Runge-Kutta methods easier.)

Improved Euler Method

For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

The value y_n will be an approximation for the value of the solution y at x_n. (Formulated to make the transition to Runge-Kutta methods easier.) The one step error is proportional to $(\Delta x)^3$ and the global error is proportional to $(\Delta x)^2$.

Improved Euler Method

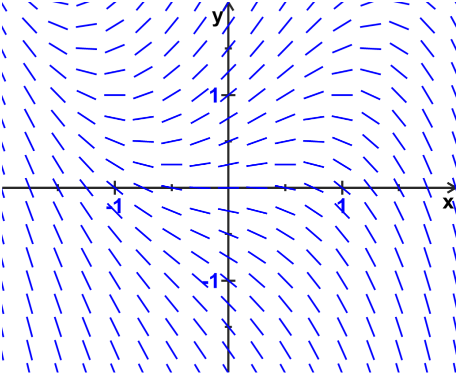
For a given step length Δx, with initial values (x_0, y_0), we compute the two recursively defined sequences $\{x_n\}_{n=0}^{\infty}$ and $\{y_n\}_{n=0}^{\infty}$ via

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

The value y_n will be an approximation for the value of the solution y at x_n. (Formulated to make the transition to Runge-Kutta methods easier.) The one step error is proportional to $(\Delta x)^3$ and the global error is proportional to $(\Delta x)^2$. (Again we omit the considerable details.)

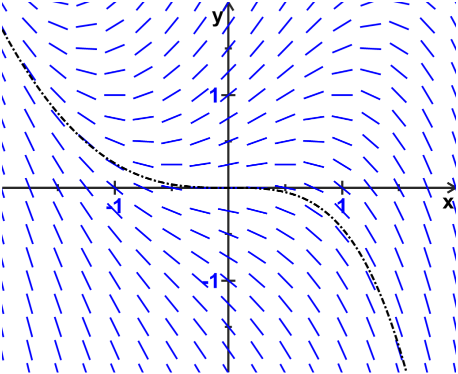
Improved Euler Method versus Euler's Method

Improved Euler Method versus Euler's Method



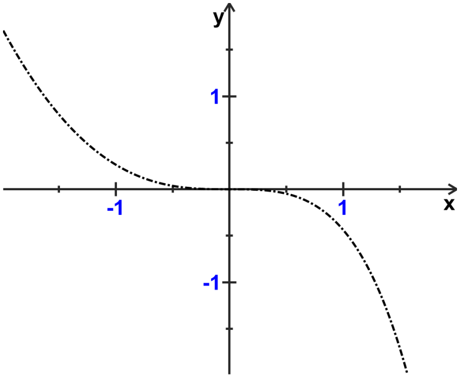
$$y' = y - x^2,\ y(0) = 0$$

Improved Euler Method versus Euler's Method



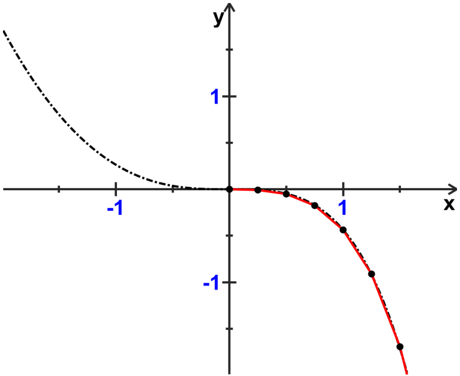
$$y' = y - x^2,\ y(0) = 0$$

Improved Euler Method versus Euler's Method



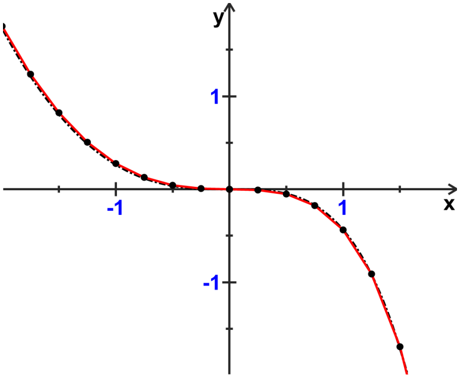
$$y' = y - x^2, \; y(0) = 0$$

Improved Euler Method versus Euler’s Method



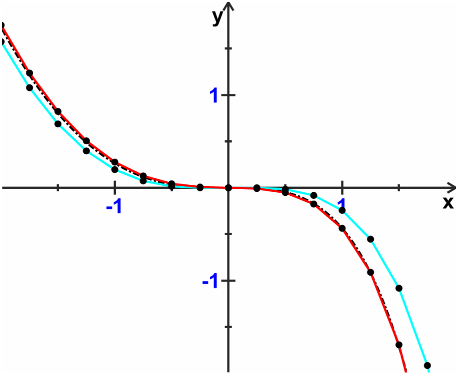
$$y' = y - x^2, \; y(0) = 0$$

Improved Euler Method versus Euler's Method



$$y' = y - x^2, \; y(0) = 0$$

Improved Euler Method versus Euler's Method



$$y' = y - x^2,\ y(0) = 0$$

Reminder for Spreadsheet Implementation

Reminder for Spreadsheet Implementation

Step length: Δx. Initial values: (x_0, y_0).

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{2}k_1 + \frac{1}{2}k_2, \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F(x_n + \Delta x, y_n + k_1)\Delta x. \end{aligned}$$

The value y_n will be an approximation for the value of the solution y at x_n.

Further Improvements

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\mathrm{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\text{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule.

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\mathrm{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule. Its increment is $f(x_i)\frac{\Delta x}{6}+4f\left(x_{i+\frac{1}{2}}\right)\frac{\Delta x}{6}+f(x_{i+1})\frac{\Delta x}{6}$.

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\text{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule. Its increment is $f(x_i)\frac{\Delta x}{6}+4f\left(x_{i+\frac{1}{2}}\right)\frac{\Delta x}{6}+f(x_{i+1})\frac{\Delta x}{6}$.
3. So how do we translate this into a formula for differential equations that has high accuracy?

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\text{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule. Its increment is $f(x_i)\frac{\Delta x}{6}+4f\left(x_{i+\frac{1}{2}}\right)\frac{\Delta x}{6}+f(x_{i+1})\frac{\Delta x}{6}$.
3. So how do we translate this into a formula for differential equations that has high accuracy?
4. Match as many terms of Taylor's formula as possible. The remainder term gives the order of the error.

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\big(x,y(x)\big)\Delta x+\frac{1}{2}F\big(x+\Delta x,y_{\text{Euler}}(x+\Delta x)\big)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\big(f(x_i)+f(x_{i+1})\big)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule. Its increment is $f(x_i)\frac{\Delta x}{6}+4f\left(x_{i+\frac{1}{2}}\right)\frac{\Delta x}{6}+f(x_{i+1})\frac{\Delta x}{6}$.
3. So how do we translate this into a formula for differential equations that has high accuracy?
4. Match as many terms of Taylor's formula as possible. The remainder term gives the order of the error.
5. The global error in Simpson's rule is $\sim(\Delta x)^4$

Further Improvements

1. The increment in the improved Euler method $\frac{1}{2}F\left(x,y(x)\right)\Delta x+\frac{1}{2}F\left(x+\Delta x,y_{\text{Euler}}(x+\Delta x)\right)\Delta x$ looks like the increment in the trapezoidal rule in numerical integration: $\frac{1}{2}\left(f(x_i)+f(x_{i+1})\right)\Delta x$
2. Simpson's rule is more accurate than the trapezoidal rule. Its increment is $f(x_i)\frac{\Delta x}{6}+4f\left(x_{i+\frac{1}{2}}\right)\frac{\Delta x}{6}+f(x_{i+1})\frac{\Delta x}{6}$.
3. So how do we translate this into a formula for differential equations that has high accuracy?
4. Match as many terms of Taylor's formula as possible. The remainder term gives the order of the error.
5. The global error in Simpson's rule is $\sim(\Delta x)^4$, so we must match the first four terms of the Taylor expansion.

The most common Runge-Kutta Method

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$x_{n+1} := x_n + \Delta x$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \end{aligned}$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \end{aligned}$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$
\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x,
\end{aligned}
$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$
\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\
k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x,
\end{aligned}
$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\
k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x, \\
k_4 &= F\left(x_n + \Delta x, y_n + k_3\right)\Delta x,
\end{aligned}$$

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\
k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x, \\
k_4 &= F\left(x_n + \Delta x, y_n + k_3\right)\Delta x,
\end{aligned}$$

y_n approximates the value of the solution y at x_n.

The most common Runge-Kutta Method

Step length Δx, initial values (x_0, y_0),

$$\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\
k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x, \\
k_4 &= F\left(x_n + \Delta x, y_n + k_3\right)\Delta x,
\end{aligned}$$

y_n approximates the value of the solution y at x_n. One step error $\sim (\Delta x)^5$, global error $\sim (\Delta x)^4$.

The most common Runge-Kutta Method

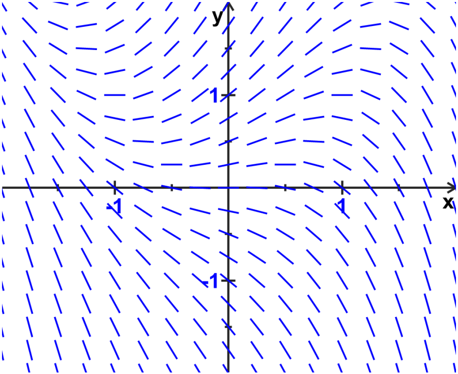
Step length Δx, initial values (x_0, y_0),

$$\begin{aligned} x_{n+1} &:= x_n + \Delta x \\ y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\ k_1 &= F(x_n, y_n)\Delta x, \\ k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\ k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x, \\ k_4 &= F\left(x_n + \Delta x, y_n + k_3\right)\Delta x, \end{aligned}$$

y_n approximates the value of the solution y at x_n. One step error $\sim (\Delta x)^5$, global error $\sim (\Delta x)^4$. (Considerable details omitted.)

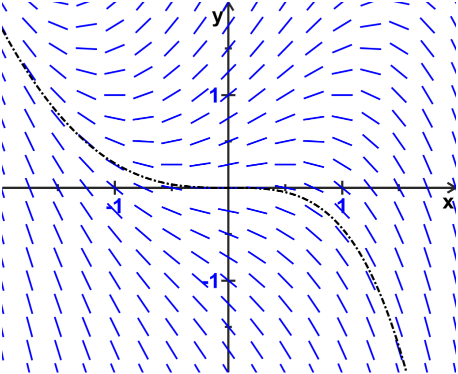
Fourth Order Runge-Kutta Method

Fourth Order Runge-Kutta Method



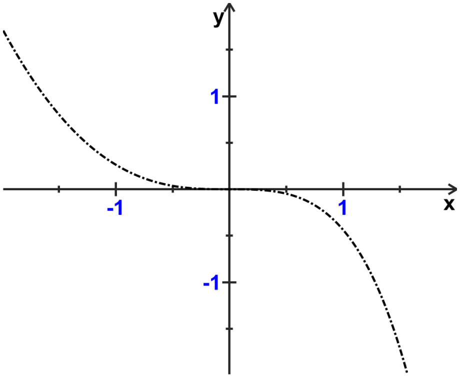
$$y' = y - x^2, \; y(0) = 0$$

Fourth Order Runge-Kutta Method



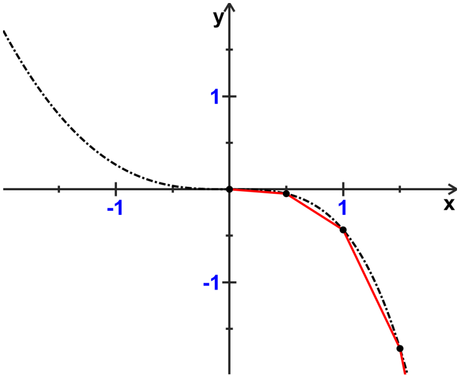
$$y' = y - x^2, \; y(0) = 0$$

Fourth Order Runge-Kutta Method



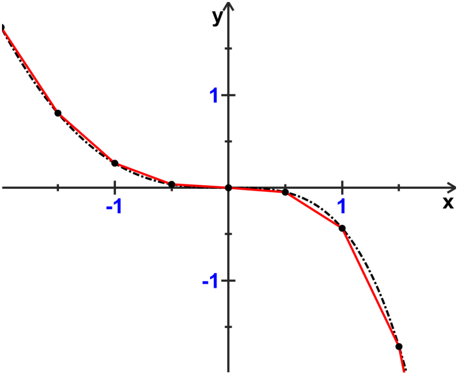
$$y' = y - x^2, \; y(0) = 0$$

Fourth Order Runge-Kutta Method



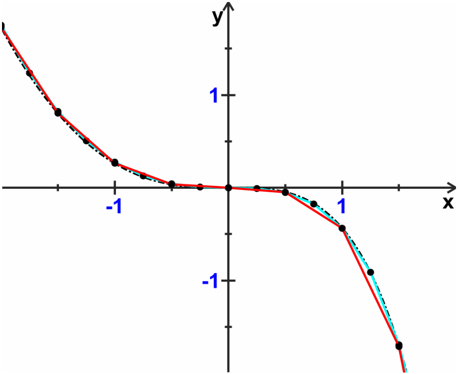
$$y' = y - x^2, \; y(0) = 0$$

Fourth Order Runge-Kutta Method



$$y' = y - x^2, \; y(0) = 0$$

Fourth Order Runge-Kutta Method



$$y' = y - x^2, \; y(0) = 0$$

Reminder for Spreadsheet Implementation

Step length: Δx. Initial values: (x_0, y_0).

$$
\begin{aligned}
x_{n+1} &:= x_n + \Delta x \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \qquad \text{where} \\
k_1 &= F(x_n, y_n)\Delta x, \\
k_2 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\right)\Delta x, \\
k_3 &= F\left(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\right)\Delta x, \\
k_4 &= F\left(x_n + \Delta x, y_n + k_3\right)\Delta x,
\end{aligned}
$$

y_n will be an approximation for the value of the solution y at x_n.

General Runge-Kutta Methods

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1k_1 + a_2k_2 + \cdots + a_mk_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1k_1 + a_2k_2 + \cdots + a_mk_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1 k_1 + a_2 k_2 + \cdots + a_m k_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1k_1 + a_2k_2 + \cdots + a_mk_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

Have seen a fourth order Runge-Kutta procedure.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1k_1 + a_2k_2 + \cdots + a_mk_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

Have seen a fourth order Runge-Kutta procedure.

Setting up a formula for increments to match a Taylor polynomial actually leads to a system of equations for the parameters in the setup of the approximation formulas.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x, y)$ by computing a sequence of values $y_{n+1} := y_n + a_1 k_1 + a_2 k_2 + \cdots + a_m k_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

Have seen a fourth order Runge-Kutta procedure.

Setting up a formula for increments to match a Taylor polynomial actually leads to a system of equations for the parameters in the setup of the approximation formulas. So there is more than one second order Runge-Kutta method.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1k_1 + a_2k_2 + \cdots + a_mk_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

Have seen a fourth order Runge-Kutta procedure.

Setting up a formula for increments to match a Taylor polynomial actually leads to a system of equations for the parameters in the setup of the approximation formulas. So there is more than one second order Runge-Kutta method. Same goes for higher orders.

General Runge-Kutta Methods

A j^{th} order **Runge-Kutta procedure** computes an approximate solution to the differential equation $y' = F(x,y)$ by computing a sequence of values $y_{n+1} := y_n + a_1 k_1 + a_2 k_2 + \cdots + a_m k_m$ so that y_{n+1} agrees with the j^{th} order Taylor polynomial of y at the previous evaluation point.

The Euler method is a first order Runge-Kutta procedure.

Improved Euler method: second order Runge-Kutta.

Have seen a fourth order Runge-Kutta procedure.

Setting up a formula for increments to match a Taylor polynomial actually leads to a system of equations for the parameters in the setup of the approximation formulas. So there is more than one second order Runge-Kutta method. Same goes for higher orders. Some of this freedom can be used to improve performance for certain types of equations.