Runge-Kutta Methods

Shehryar Malik

Information Technology University

October 16, 2019

Notation

Notation

Function: $x(t)$

Notation

Function: $x(t)$

Denote $x_n = x(t_n)$ and $x_{n+1} = x(t_{n+1})$

Notation

Function: $x(t)$

Denote $x_n = x(t_n)$ and $x_{n+1} = x(t_{n+1})$

Let $h = t_{n+1} - t_n$ be the step size.

Notation

Function: $x(t)$

Denote $x_n = x(t_n)$ and $x_{n+1} = x(t_{n+1})$

Let $h = t_{n+1} - t_n$ be the step size.

Finally, let $x_i' = \left.\frac{dx}{dt}\right|_{t=t_i}$, $x_i'' = \left.\frac{d^2x}{dt^2}\right|_{t=t_i}$ and so on.

Forward Euler Method

Taylor's Series:

$$x_{n+1} = x_n + hx'_n + \frac{h^2}{2!}x''_n + \frac{h^3}{3!}x'''_n + \dots \tag{1}$$

Forward Euler Method

Taylor's Series:

$$x_{n+1} = x_n + hx_n' + \frac{h^2}{2!}x_n'' + \frac{h^3}{3!}x_n''' + \dots \tag{1}$$

Suppose we know the function $f(x,t) = x'(t)$. Then

$$x_{n+1} \approx x_n + hf(x_n, t_n). \tag{2}$$

Forward Euler Method

Taylor's Series:

$$x_{n+1} = x_n + hx'_n + \frac{h^2}{2!}x''_n + \frac{h^3}{3!}x'''_n + \dots \tag{1}$$

Suppose we know the function $f(x,t) = x'(t)$. Then

$$x_{n+1} \approx x_n + hf(x_n, t_n). \tag{2}$$

This is the Forward-Euler method

Forward Euler Method

Taylor's Series:

$$x_{n+1} = x_n + hx_n' + \frac{h^2}{2!}x_n'' + \frac{h^3}{3!}x_n''' + \dots \tag{1}$$

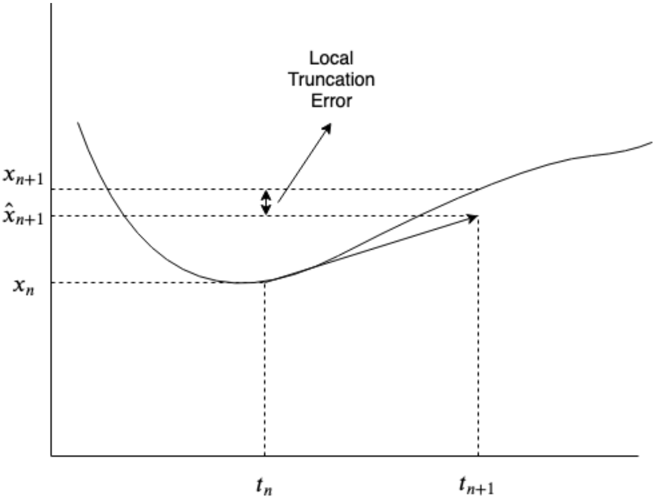
Suppose we know the function $f(x, t) = x'(t)$. Then

$$x_{n+1} \approx x_n + hf(x_n, t_n). \tag{2}$$

This is the Forward-Euler method

Local truncation error is $\mathcal{O}(h^2)$.

Visualizing the Forward Euler Method



Second Order Runge-Kutta Method

Second Order Runge-Kutta Method

Expand Taylor's Series upto 2nd derivative:

$$x_{n+1} \approx x_n + hx'_n + \frac{h^2}{2!}x''_n. \tag{3}$$

Second Order Runge-Kutta Method

Expand Taylor's Series upto 2nd derivative:

$$x_{n+1} \approx x_n + h x_n' + \frac{h^2}{2!} x_n''. \tag{3}$$

LTE is $\mathcal{O}(h^3)$

Second Order Runge-Kutta Method

Expand Taylor's Series upto 2nd derivative:

$$x_{n+1} \approx x_n + hx'_n + \frac{h^2}{2!}x''_n. \tag{3}$$

LTE is $\mathcal{O}(h^3)$

Using Chain Rule

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial f(x,t)}{\partial t} + \frac{\partial f(x,t)}{\partial x} f(x,t). \tag{4}$$

Second Order Runge-Kutta Method

Expand Taylor's Series upto 2nd derivative:

$$x_{n+1} \approx x_n + hx'_n + \frac{h^2}{2!}x''_n. \tag{3}$$

LTE is $\mathcal{O}(h^3)$

Using Chain Rule

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial f(x,t)}{\partial t} + \frac{\partial f(x,t)}{\partial x}f(x,t). \tag{4}$$

Substituting this above gives

$$x_{n+1} \approx x_n + hf(x_n,t_n) + \frac{h^2}{2!}\left[\frac{\partial f(x_n,t_n)}{\partial t} + \frac{\partial f(x_n,t_n)}{\partial x}f(x_n,t_n)\right]. \tag{5}$$

Second Order Runge-Kutta Method

Goal: Develop method with error $\mathcal{O}(h^3)$ without computing derivatives of $f(x, t)$

Second Order Runge-Kutta Method

Goal: Develop method with error $\mathcal{O}(h^3)$ without computing derivatives of $f(x, t)$

Idea: Use derivative at (x_n, t_n) and (x_k, t_k).

Second Order Runge-Kutta Method

Goal: Develop method with error $\mathcal{O}(h^3)$ without computing derivatives of $f(x,t)$

Idea: Use derivative at (x_n, t_n) and (x_k, t_k).

Define

$$t_k = t_n + \alpha h \tag{6}$$

for some α and

Second Order Runge-Kutta Method

Goal: Develop method with error $\mathcal{O}(h^3)$ without computing derivatives of $f(x, t)$

Idea: Use derivative at (x_n, t_n) and (x_k, t_k).

Define

$$t_k = t_n + \alpha h \tag{6}$$

for some α and

$$x_k = x_n + \beta h f(x_n, t_n) \tag{7}$$

for some β.

Second Order Runge-Kutta Method

Goal: Develop method with error $\mathcal{O}(h^3)$ without computing derivatives of $f(x,t)$

Idea: Use derivative at (x_n, t_n) and (x_k, t_k).

Define

$$t_k = t_n + \alpha h \tag{6}$$

for some α and

$$x_k = x_n + \beta h f(x_n, t_n) \tag{7}$$

for some β.

Also define

$$\begin{aligned} k_1 &= hf(x_n, t_n), \\ k_2 &= hf(x_k, t_k). \end{aligned} \tag{8}$$

Second Order Runge-Kutta Method

Want to use k_1 and k_2 to get x_{n+1}

$$x_{n+1} = x_n + ak_1 + bk_2 \tag{9}$$

Second Order Runge-Kutta Method

Want to use k_1 and k_2 to get x_{n+1}

$$x_{n+1} = x_n + ak_1 + bk_2 \tag{9}$$

Note that

$$\begin{aligned} k_2 &= hf(x_n + \alpha hf(x_n, t_n), t_n + \beta h) \\ &\approx h\left[f(x_n, t_n) + \left[\frac{\beta hf(x_n, t_n)}{2!}\frac{\partial f(x_n, t_n)}{\partial x} + \frac{\alpha h}{2!}\frac{\partial f(x_n, t_n)}{\partial t}\right]\right]. \end{aligned}$$

Second Order Runge-Kutta Method

Want to use k_1 and k_2 to get x_{n+1}

$$x_{n+1} = x_n + ak_1 + bk_2 \tag{9}$$

Note that

$$\begin{aligned} k_2 &= hf(x_n + \alpha hf(x_n, t_n), t_n + \beta h) \\ &\approx h\left[f(x_n, t_n) + \left[\frac{\beta hf(x_n, t_n)}{2!}\frac{\partial f(x_n, t_n)}{\partial x} + \frac{\alpha h}{2!}\frac{\partial f(x_n, t_n)}{\partial t}\right]\right]. \end{aligned}$$

Error is $\mathcal{O}(h^3)$

Second Order Runge-Kutta Method

Substitute and rearrange:

$$x_{n+1} \approx x_n + (a+b)hf(x_n, t_n) + \frac{h^2}{2!}\left[b\beta f(x_n, t_n)\frac{\partial f(x_n, t_n)}{\partial x} + b\alpha\frac{\partial f(x_n, t_n)}{\partial t}\right]$$

Second Order Runge-Kutta Method

Substitute and rearrange:

$$x_{n+1} \approx x_n + (a+b)hf(x_n, t_n) + \frac{h^2}{2!}\left[b\beta f(x_n, t_n)\frac{\partial f(x_n, t_n)}{\partial x} + b\alpha\frac{\partial f(x_n, t_n)}{\partial t}\right]$$

Also had previously

$$x_{n+1} \approx x_n + hf(x_n, t_n) + \frac{h^2}{2!}\left[\frac{\partial f(x_n, t_n)}{\partial t} + \frac{\partial f(x_n, t_n)}{\partial x}f(x_n, t_n)\right]$$

Second Order Runge-Kutta Method

Substitute and rearrange:

$$x_{n+1} \approx x_n + (a+b)hf(x_n, t_n) + \frac{h^2}{2!}\left[b\beta f(x_n, t_n)\frac{\partial f(x_n, t_n)}{\partial x} + b\alpha\frac{\partial f(x_n, t_n)}{\partial t}\right]$$

Also had previously

$$x_{n+1} \approx x_n + hf(x_n, t_n) + \frac{h^2}{2!}\left[\frac{\partial f(x_n, t_n)}{\partial t} + \frac{\partial f(x_n, t_n)}{\partial x}f(x_n, t_n)\right]$$

Compare coefficients

$$\begin{aligned} a + b &= 1, \\ b\beta &= 1, \\ b\alpha &= 1. \end{aligned}$$

Second Order Runge-Kutta Method

Substitute and rearrange:

$$x_{n+1} \approx x_n + (a+b)hf(x_n, t_n) + \frac{h^2}{2!}\left[b\beta f(x_n, t_n)\frac{\partial f(x_n, t_n)}{\partial x} + b\alpha\frac{\partial f(x_n, t_n)}{\partial t}\right]$$

Also had previously

$$x_{n+1} \approx x_n + hf(x_n, t_n) + \frac{h^2}{2!}\left[\frac{\partial f(x_n, t_n)}{\partial t} + \frac{\partial f(x_n, t_n)}{\partial x}f(x_n, t_n)\right]$$

Compare coefficients

$$\begin{aligned} a + b &= 1, \\ b\beta &= 1, \\ b\alpha &= 1. \end{aligned}$$

Infinite number of possible solutions.

Second Order Runge-Kutta Method

Classic second order Runge-Kutta chooses $\alpha = \beta = 1$ and $a = b = \frac{1}{2}$

Second Order Runge-Kutta Method

Classic second order Runge-Kutta chooses $\alpha = \beta = 1$ and $a = b = \frac{1}{2}$

$$\begin{aligned} k_1 &= hf(x_n, t_n), \\ k_2 &= hf(x_n + hk_1, t_n + h), \\ x_{n+1} &= x_n + \frac{1}{2}(k_1 + k_2). \end{aligned}$$

Fourth Order Runge-Kutta

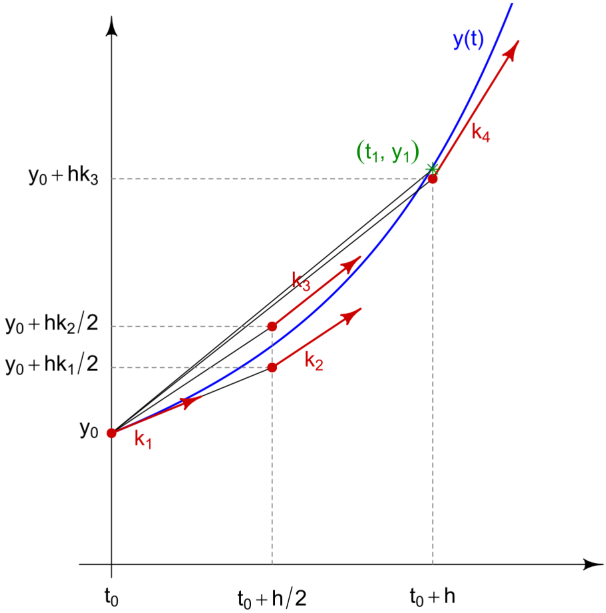
$$
\begin{aligned}
k_1 &= hf(x_n, t_n), \\
k_2 &= hf(x_n + k_1/2, t_n + h/2), \\
k_3 &= hf(x_n + k_2/2, t_n + h/2), \\
k_4 &= hf(x_n + k_3, t_n + h), \\
x_{n+1} &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

Fourth Order Runge-Kutta

$$
\begin{aligned}
k_1 &= hf(x_n, t_n), \\
k_2 &= hf(x_n + k_1/2, t_n + h/2), \\
k_3 &= hf(x_n + k_2/2, t_n + h/2), \\
k_4 &= hf(x_n + k_3, t_n + h), \\
x_{n+1} &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned}
$$

Error is $\mathcal{O}(h^5)$

Visualizing the Fourth Order Runge-Kutta Method



Implict VS Explicit Methods

Explicit Methods:

Implict VS Explicit Methods

Explicit Methods:

Can express x_{n+1} in terms other than itself.

Implict VS Explicit Methods

Explicit Methods:

Can express x_{n+1} in terms other than itself.

However, these methods are not numerically stable. Need to put certain conditions on h.

Implict VS Explicit Methods

Explicit Methods:

Can express x_{n+1} in terms other than itself.

However, these methods are not numerically stable. Need to put certain conditions on h.

Implicit Methods:

Implict VS Explicit Methods

Explicit Methods:

Can express x_{n+1} in terms other than itself.

However, these methods are not numerically stable. Need to put certain conditions on h.

Implicit Methods:

Are stable for all values of h.

Implict VS Explicit Methods

Explicit Methods:

Can express x_{n+1} in terms other than itself.

However, these methods are not numerically stable. Need to put certain conditions on h.

Implicit Methods:

Are stable for all values of h.

However, we can not express x_{n+1} in terms other than itself.

Backward Euler Method

Let us expand x_n using the Taylor's Series as follows

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{10}$$

Backward Euler Method

Let us expand x_n using the Taylor's Series as follows

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{10}$$

Considering only the first two terms on the right hand side above and rearranging gives

$$x_{n+1} \approx x_n + hf(x_{n+1}, t_{n+1}). \tag{11}$$

Backward Euler Method

Let us expand x_n using the Taylor's Series as follows

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{10}$$

Considering only the first two terms on the right hand side above and rearranging gives

$$x_{n+1} \approx x_n + hf(x_{n+1}, t_{n+1}). \tag{11}$$

Want to find roots of

$$x_{n+1} - x_n - hf(x_{n+1}, t_{n+1}) = 0. \tag{12}$$

Backward Euler Method

Let us expand x_n using the Taylor's Series as follows

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{10}$$

Considering only the first two terms on the right hand side above and rearranging gives

$$x_{n+1} \approx x_n + hf(x_{n+1}, t_{n+1}). \tag{11}$$

Want to find roots of

$$x_{n+1} - x_n - hf(x_{n+1}, t_{n+1}) = 0. \tag{12}$$

Can do so through Newton's Rapshon method.

Backward Euler Method

Let us expand x_n using the Taylor's Series as follows

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{10}$$

Considering only the first two terms on the right hand side above and rearranging gives

$$x_{n+1} \approx x_n + hf(x_{n+1}, t_{n+1}). \tag{11}$$

Want to find roots of

$$x_{n+1} - x_n - hf(x_{n+1}, t_{n+1}) = 0. \tag{12}$$

Can do so through Newton's Rapshon method.

Error however is still $\mathcal{O}(h^2)$.

General Form of Implicit Runge-Kutta Methods

General Form of Implicit Runge-Kutta Methods

For q-stage implicit Runge-Kutta

$$x_{n+1} = x_n + \sum_{i=1}^{q} b_i k_i \tag{13}$$

for some known constants b_i where

General Form of Implicit Runge-Kutta Methods

For q-stage implicit Runge-Kutta

$$x_{n+1} = x_n + \sum_{i=1}^{q} b_i k_i \tag{13}$$

for some known constants b_i where

$$\begin{aligned} k_1 &= hf(x_n + h\sum_{i=1}^{q} a_{1i}k_i, t_n + c_1 h) \\ &\vdots \\ k_q &= hf(x_n + h\sum_{i=1}^{q} a_{qi}k_i, t_n + c_q h) \end{aligned} \tag{14}$$

General Form of Implicit Runge-Kutta Methods

For q-stage implicit Runge-Kutta

$$x_{n+1} = x_n + \sum_{i=1}^{q} b_i k_i \tag{13}$$

for some known constants b_i where

$$\begin{aligned} k_1 &= hf(x_n + h\sum_{i=1}^{q} a_{1i}k_i, t_n + c_1 h) \\ &\vdots \\ k_q &= hf(x_n + h\sum_{i=1}^{q} a_{qi}k_i, t_n + c_q h) \end{aligned} \tag{14}$$

k_j is approximating $\frac{dx}{dt}$ at $t + c_j h$.

General Form of Implicit Runge-Kutta Methods

For q-stage implicit Runge-Kutta

$$x_{n+1} = x_n + \sum_{i=1}^{q} b_i k_i \tag{13}$$

for some known constants b_i where

$$\begin{aligned} k_1 &= hf(x_n + h\sum_{i=1}^{q} a_{1i}k_i, t_n + c_1 h) \\ &\vdots \\ k_q &= hf(x_n + h\sum_{i=1}^{q} a_{qi}k_i, t_n + c_q h) \end{aligned} \tag{14}$$

k_j is approximating $\frac{dx}{dt}$ at $t + c_j h$.

First argument to f is approximation of $x(t + c_j h)$ using all k's.

Alternative Formulation

Approximate x_{n+1} via

$$x_{n+1} = x_n + h \sum_{i=1}^{q} b_i f(x^{n+c_i}, t^{n+c_i}) \tag{15}$$

where $t^{n+c_i} = t_n + c_i h$ and

Alternative Formulation

Approximate x_{n+1} via

$$x_{n+1} = x_n + h \sum_{i=1}^{q} b_i f(x^{n+c_i}, t^{n+c_i}) \tag{15}$$

where $t^{n+c_i} = t_n + c_i h$ and

$$x^{n+c_i} = x_n + h \sum_{j=1}^{q} a_{ij} f(x^{n+c_j}, t^{n+c_j}) \tag{16}$$

approximates $x(t_n + c_i h)$

A Deep Learning Application

A Deep Learning Application

Rearrange previous two equations:

$$x_n = x_{n+1} - h \sum_{i=1}^{q} b_i f(x^{n+c_i}) \tag{17}$$

$$x_n = x^{n+c_i} - h \sum_{j=1}^{q} a_{ij} f(x^{n+c_j}). \tag{18}$$

A Deep Learning Application

Rearrange previous two equations:

$$x_n = x_{n+1} - h \sum_{i=1}^{q} b_i f(x^{n+c_i}) \tag{17}$$

$$x_n = x^{n+c_i} - h \sum_{j=1}^{q} a_{ij} f(x^{n+c_j}). \tag{18}$$

Ask a neural net to predict

$$[x^{n+c_1}, \ldots, x^{n+c_q}, x_{n+1}]. \tag{19}$$

given t

A Deep Learning Application

Rearrange previous two equations:

$$x_n = x_{n+1} - h \sum_{i=1}^{q} b_i f(x^{n+c_i}) \tag{17}$$

$$x_n = x^{n+c_i} - h \sum_{j=1}^{q} a_{ij} f(x^{n+c_j}). \tag{18}$$

Ask a neural net to predict

$$[x^{n+c_1}, \ldots, x^{n+c_q}, x_{n+1}]. \tag{19}$$

given t

Calculate $q+1$ estimates of x_n using these outputs.

A Deep Learning Application

Rearrange previous two equations:

$$x_n = x_{n+1} - h \sum_{i=1}^{q} b_i f(x^{n+c_i}) \tag{17}$$

$$x_n = x^{n+c_i} - h \sum_{j=1}^{q} a_{ij} f(x^{n+c_j}). \tag{18}$$

Ask a neural net to predict

$$[x^{n+c_1}, \ldots, x^{n+c_q}, x_{n+1}]. \tag{19}$$

given t

Calculate $q+1$ estimates of x_n using these outputs.

Enforce

$$\mathcal{L} = \sum_{j=1}^{q+1} ||\hat{x}_{n,j} - x_n||^2. \tag{20}$$