Runge-Kutta Methods

Shehryar Malik

October 15, 2019

These notes are meant to be a simple introduction to the explicit and implicit Runge-Kutta methods. They are by no means exhaustive, complete or rigorous. Most of the material presented in these notes is derived from [1], [2] and [3].

1 Preliminaries

We begin by revisiting the Taylor's Series expansion of a function $x(t)$. Let x_n and x_{n+1} denote the values of $x(t)$ at $t = t_n$ and $t = t_{n+1}$ respectively. Define h to be $t_{n+1} - t_n$ and let x_i' denote the first derivative, x_i'' the second derivative and so on and so forth, of x with respect to t evaluated at $t = t_i$. We have

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

Let $f(x, t)$ denote the derivative of x with respect to t at x and t. In these notes, we'll assume that f is a known function. We can approximate x_{n+1} using only the first two terms of the Taylor's Series above as

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

This is known as the Forward Euler method. Note that the error in this approximation is of the order of $\mathcal{O}(h^2)$. We call this as the local truncation error (LTE), since it is a result of the truncation of the Taylor's Series. Figure 1 graphically visualizes Equation 2. $\hat{x}_{n+1}$ is the approximated value of x_{n+1}. Finally, note that we can extend the Taylor's Series to functions of two variables. Suppose that g is a function of x and t. We have

$$\begin{aligned} g(x_n + h_x, t_n + h_t) = g(x_n, t_n) + \left[h_x \frac{\partial g(x_n, t_n)}{\partial x} + h_t \frac{\partial g(x_n, t_n)}{\partial t}\right] + \\ \left[\frac{h_x}{2!}\frac{\partial^2 g(x_n, t_n)}{\partial x^2} + \frac{h_t}{2!}\frac{\partial^2 g(x_n, t_n)}{\partial t^2}\right] + \dots \end{aligned} \tag{3}$$

for some constants h_x and h_t.

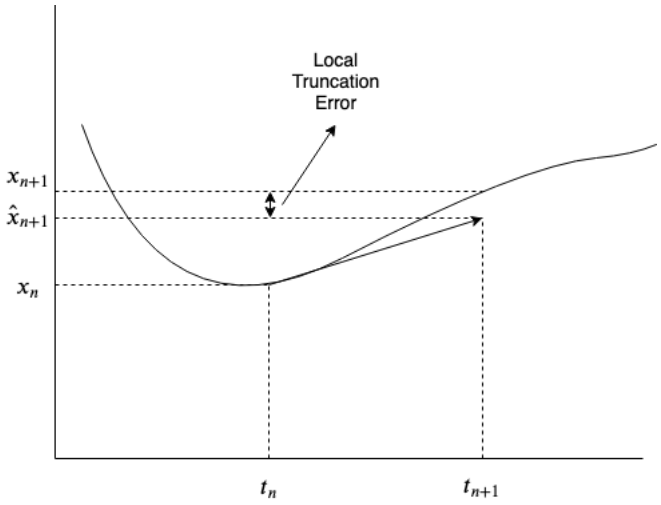

Figure 1: Visualizing the Forward Euler Method

In these notes, we will discuss the Runge-Kutta (RK) methods. In addition to using the derivative of x with respect to t at t_n to estimate x_{n+1} (as in the Forward Euler method), the Runge-Kutta methods also make use of the derivatives of x with respect to t at different points in-between t_n and t_{n+1}.

2 Explicit Methods

In this section, we will talk about the explicit Runge-Kutta methods. The reason why these methods are called explicit will become apparent in the next section when we talk about the implicit Runge-Kutta methods.

2.1 Second Order Runge-Kutta Method

Let us expand the Taylor's Series up to the second derivative.

$$x_{n+1} \approx x_n + hx'_n + \frac{h^2}{2!}x''_n. \tag{4}$$

The local truncation error in this approximation is clearly of the order of $\mathcal{O}(h^3)$. We had earlier defined $f(x,t)$ to be the derivative of x with respect t. Using the chain rule, we have

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial f(x,t)}{\partial t} + \frac{\partial f(x,t)}{\partial x} f(x,t). \tag{5}$$

Substituting this into Equation 4 gives

$$x_{n+1} \approx x_n + hf(x_n, t_n) + \frac{h^2}{2!}\left[\frac{\partial f(x_n, t_n)}{\partial t} + \frac{\partial f(x_n, t_n)}{\partial x} f(x_n, t_n)\right]. \quad (6)$$

We will make use of this result later on.

We would like to develop a method of approximating x_{n+1} which has an error of the order of $O(h^3)$ without having to compute the derivatives of f with respect to x and t. The second order Runge-Kutta method makes use of the derivative of x with respect to t at two points: t_n and t_k where

$$t_k = t_n + \alpha h \quad (7)$$

for some constant α. However, in order to calculate the derivative of x with respect to t at t_k using the function $f(x, t)$, we need to know the value of x_k. Let us express x_k as

$$x_k = x_n + \beta h f(x_n, t_n) \quad (8)$$

for some appropriate constant β. We will choose a value for β later on. We can now compute the derivative of x with respect to t at both (t_n, x_n) and (t_k, x_k) using $f(x, t)$. Define

$$\begin{aligned} k_1 &= hf(x_n, t_n), \\ k_2 &= hf(x_k, t_k). \end{aligned} \quad (9)$$

We would now like to use k_1 and k_2 to estimate x_{n+1}. Suppose we wish to use them in some linear fahsion as follows

$$x_{n+1} = x_n + ak_1 + bk_2 \quad (10)$$

for some appropriate constants a and b. To do so, let us rewrite k_2 using Equations 7 and 8 as follows

$$k_2 = hf(x_n + \alpha h f(x_n, t_n), t_n + \beta h). \quad (11)$$

Let us now approximate k_2 using the Taylor's Series from Equation 3 as follows

$$k_2 \approx h\left[f(x_n, t_n) + \left[\frac{\beta h f(x_n, t_n)}{2!}\frac{\partial f(x_n, t_n)}{\partial x} + \frac{\alpha h}{2!}\frac{\partial f(x_n, t_n)}{\partial t}\right]\right]. \quad (12)$$

Note that the error in the term within the outermost square brackets is of the order of $\mathcal{O}(h^2)$. However, the error in k_2 is of the order of $\mathcal{O}(h^3)$ as we further multiply the left hand side above with h. Let us now substitute both k_1 and k_2 (from Equations 9 and 12) into Equation 10 as follows

$$\begin{aligned} x_{n+1} \approx\ & x_n + ahf(x_n, t_n) \\ & + bh\left[f(x_n, t_n) + \left[\frac{\beta h f(x_n, t_n)}{2!}\frac{\partial f(x_n, t_n)}{\partial x} + \frac{\alpha h}{2!}\frac{\partial f(x_n, t_n)}{\partial t}\right]\right]. \end{aligned} \quad (13)$$

Rearranging this gives

$$x_{n+1} \approx x_n + (a+b)hf(x_n,t_n) + \frac{h^2}{2!}\left[b\beta f(x_n,t_n)\frac{\partial f(x_n,t_n)}{\partial x} + b\alpha\frac{\partial f(x_n,t_n)}{\partial t}\right]. \tag{14}$$

We want to choose constants a, b, α, β in the equation such that the error in x_{n+1} is of the order of $\mathcal{O}(h^3)$. Since we know that Equation 6 has an error of order $\mathcal{O}(h^3)$, we can directly compare its coefficients with the coefficients in the equation above. This yields

$$\begin{aligned} a + b &= 1, \\ b\beta &= 1, \\ b\alpha &= 1. \end{aligned} \tag{15}$$

We have an infinite number of possibilities here, since we only have three equations for four variables. The classic second order Runge Kutta method chooses $\alpha = \beta = 1$ and $a = b = \frac{1}{2}$. The second order Runge Kutta, thus, approximates x_{n+1} using the following scheme

$$\begin{aligned} k_1 &= hf(x_n, t_n), \\ k_2 &= hf(x_n + hk_1, t_n + h), \\ x_{n+1} &= x_n + \frac{1}{2}(k_1 + k_2). \end{aligned} \tag{16}$$

2.2 Fourth Order Runge-Kutta Method

In a similar fashion, we can derive the fourth order Runge-Kutta method which has an error of the order of $\mathcal{O}(h^5)$. We state the method below without proof.

$$\begin{aligned} k_1 &= hf(x_n, t_n), \\ k_2 &= hf(x_n + k_1/2, t_n + h/2), \\ k_3 &= hf(x_n + k_2/2, t_n + h/2), \\ k_4 &= hf(x_n + k_3, t_n + h), \\ x_{n+1} &= \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{17}$$

Figure 2 [taken from [4]] shows the different points whose derivatives are considered by the fourth order Runge-Kutta method (note that the figure uses the symbol y instead of x and the subscripts 0 and 1 instead of n and $n+1$ respectively).

3 Implicit Runge-Kutta Methods

Notice that in the methods discussed in the last section, x_{n+1} could be *explicity* expressed in terms other than x_{n+1}. As we shall see in this section, such an

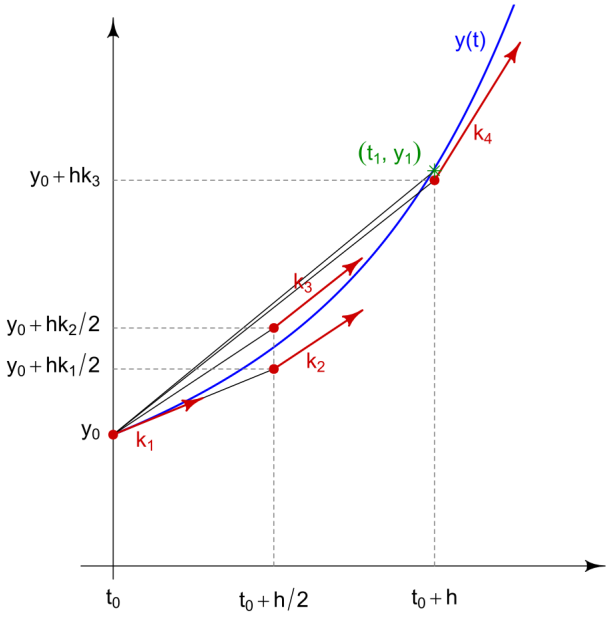

Figure 2: Visualizing the Fourth Order Runge-Kutta Method [Source: [4]]

explicit expression is not possible in the case of implicit methods. This is because in order to approximate x_{n+1} implicit methods also make use of the derivative of x with respect to t at t_{n+1}. It turns out that this allows implicit methods to be far more numerically stable than explicit methods. However, the discussion of stability of both these methods is beyond the scope of these notes.

Let us begin with the most simple implicit methods - the Backward Euler method.

3.1 Backward Euler Method

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

$$x(t_n) = x(t_{n+1} - h) = x_{n+1} - hx'_{n+1} - \frac{h^2}{2!}x'_{n+1} + \dots \tag{18}$$

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

$$x_{n+1} \approx x_n + hf(x_{n+1}, t_{n+1}). \tag{19}$$

Since x_{n+1} cannot be separated from the rest of the equation, we cannot solve for it directly. Notice that since x_n and t_n are known constants and f is a known function, we are actually interested in finding the roots of the equation

$$x_{n+1} - x_n - hf(x_{n+1}, t_{n+1}) = 0. \tag{20}$$

One way of finding these roots is through the Newton-Raphson method, the details of which we will not go into in these notes. Notice that the approximation of x_{n+1} through this method has an error of the order of $\mathcal{O}(h^2)$ which is the same as what we had in the Forward Euler method. Therefore, while implicit methods are generally far more numerically stable than their explicit counterparts, they offer no significant advantage in terms of the error. However, numerical stability in iteself is a very important quality to have in a numerical method.

3.2 General Form of Implicit Runge-Kutta Methods

We shall now present the general form of the implicit Runge-Kutta methods. The derivation of this form is well beyond the scope of these notes. The q-stage implicit Runge-Kutta method approximates x_{n+1} via

$$x_{n+1} = x_n + \sum_{i=1}^{q} b_i k_i \tag{21}$$

for some known constants b_i where

$$\begin{aligned} k_1 &= hf(x_n + h\sum_{i=1}^{q} a_{1i}k_i, t_n + c_1 h) \\ &\vdots \\ k_q &= hf(x_n + h\sum_{i=1}^{q} a_{qi}k_i, t_n + c_q h) \end{aligned} \tag{22}$$

for some known constants a_{ji} and c_j.

Notice that each k_j essentially tries to approximate the derivative of x with respect to t at $t + c_j h$ (multplied by h). Also, note that the first argument to f

$$x_n + h\sum_{i=1}^{q} a_{ji}k_i$$

for each k_j is an approximation of x at $t_n + c_j h$ using all of the k. This interpretation allows us to reformulate the general form of implicit Runge-Kutta method in another way. We are interested in approximating x_{n+1} via

$$x_{n+1} = x_n + h\sum_{i=1}^{q} b_i f(x^{n+c_i}, t^{n+c_i}) \tag{23}$$

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

$$x^{n+c_i} = x_n + h\sum_{j=1}^{q} a_{ij} f(x^{n+c_j}, t^{n+c_j}) \tag{24}$$

is an approximation of $x(t)$ at $t_n + c_i h$.

The constants a_{ji}, b_i, c_i are calculated in the same way as a, b, α and β were found for the second order explicit Runge-Kutta method above i.e. by expanding x_{n+1} completely and comparing its coefficients with those of the Taylor's series.

Finally, we must stress that the number of stages in the Runge-Kutta methods must not be confused with the order of the methods. The minimum number of stages required to achieve an error of the order of $\mathcal{O}(h^p)$ for some p is still an open problem.

4 A Deep Learning Application of the Runge-Kutta Methods

The application presented in this section is based on [5] and [6]. Consider a partial differential equation of the form

$$u_t + f(u(x,t);\lambda) = 0 \tag{25}$$

where f is some known (non-linear) function parametrized by some known constants λ. u_t denotes the partial derivative of $u(x,t)$ with respect to t. Let us apply the Runge-Kutta method with q-stages to this equation. We have

$$u_{n+1} = u_n + h\sum_{i=1}^{q} b_i f(u^{n+c_i}) \tag{26}$$

where $u_k \approx u(t_k, x)$ and

$$u^{n+c_i} = u_n + h\sum_{j=1}^{q} a_{ij} f(u^{n+c_j}) \tag{27}$$

is an approximation of $u(t,x)$ at $u(t_n + c_i\Delta t, x)$ for some known constants a, b and c. Note that x is fixed. Let us rewrite Equation 26 as

$$u_n = u_{n+1} - h\sum_{i=1}^{q} b_i f(u^{n+c_i}) \tag{28}$$

and Equation 27 as

$$u_n = u^{n+c_i} - h\sum_{j=1}^{q} a_{ij} f(u^{n+c_j}). \tag{29}$$

We can feed x to a neural network and require it to output an approximation of

$$[u^{n+c_1}, \ldots, u^{n+c_q}, u_{n+1}]. \tag{30}$$

For convenience, we shall denote these approximations as $[\hat{u}^{n+c_1}, \ldots, \hat{u}^{n+c_{q+1}}]$. We can use these approximations to calculate $q+1$ estimates of u_n using Equations 28 and 29 (q from each u^{n+c_i} and one from u_{n+1}). We shall denote the jth estimate with $\hat{u}_{n,j}$ Of course we require that all of these $\hat{u}_{n,j}$ are equal. We can enforce this constraint by minimizing the following loss function with respect to the neural net's parameters.

$$\mathcal{L} = \sum_{j=1}^{q+1} ||\hat{u}_{n,j} - u_n||^2. \tag{31}$$

Note that as the neural network does not have access to the value of t_n, we require that, when constructing the training set, u should only be evaluated at a fixed t_n. The entire system presented above, consequently, approximates the values of u at t_n and $t_n + h$ (where h is also fixed) and q distinct points in between them for all values of x.

References

[1] Implicit Runge-Kutta methods. `https://anchp.epfl.ch/wp-content/uploads/2018/05/part2-1.pdf`, Last accessed on October 14, 2019.

[2] Numerical Approximations. https://math.la.asu.edu/~dajones/class/275/ch2.pdf, Last accessed on October 14, 2019.

[3] Runge-Kutta methods. http://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html, Last accessed on October 14, 2019.

[4] Runge-Kutta methods. https://en.wikipedia.org/wiki/Runge-Kutta_methods, Last accessed on October 14, 2019.

[5] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[6] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.