Fundamentals of Computational Neuroscience
Project 2: Numerical Integration Methods

Matthew D. Boardman, #B00068260
Faculty of Computer Science, Dalhousie University

October 5, 2004

Among all the mathematical disciplines the theory of differential equations is the most important. It furnishes the explanation of all those elementary manifestations of nature which involve time.

—Marius Sophus Lie, 1842–1899 [1], [2].

1 Introduction

In this paper, given the differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -x + \mathrm{e}^{-t} \tag{1}$$

we are to solve for $x(t)$ numerically, using the Euler method and a higher order method, and compare the accuracy of these numerical approximations to the true results obtained through analytical means.

2 Analytical Solution

This equation is not separable[3] as with the examples given in class and in Appendix C of [4], but rather is a first order linear differential equation[2], of the form

$$\frac{\mathrm{d}x}{\mathrm{d}t} + p(t)x = q(t) \tag{2}$$

where, in this case:

$$p(t) = 1 \tag{3}$$
$$q(t) = \mathrm{e}^{-t} \tag{4}$$

A generalized solution to such a first order linear differential equation, restated from [5] to use our variables x and t, is given by

$$x(t) = \frac{\int \mu(t)q(t)\ \mathrm{d}t + C_1}{\mu(t)} \tag{5}$$

where $C_1 \dots C_n$ are arbitrary constants, and

$$\begin{aligned} \mu(t) &= \mathrm{e}^{\int p(t)\ \mathrm{d}t} \qquad (6) \\ &= \mathrm{e}^{\int\ \mathrm{d}t} \\ &= \mathrm{e}^{t+C_2} \\ &= C_3\ \mathrm{e}^{t} \qquad (7) \end{aligned}$$

Substituting $p(t)$, $q(t)$ and $\mu(t)$ into equation (5), we have

$$x(t) = \frac{\int \left(C_3\ \mathrm{e}^{t}\right)\left(\mathrm{e}^{-t}\right)\ \mathrm{d}t + C_1}{C_3\ \mathrm{e}^{t}} \qquad (8)$$

Solving for $x(t)$, we have

$$\begin{aligned} x(t) &= \frac{C_3 \int \mathrm{d}t + C_1}{C_3\ \mathrm{e}^{t}} \\ &= \frac{\int \mathrm{d}t + \dfrac{C_1}{C_3}}{\mathrm{e}^{t}} \\ &= \mathrm{e}^{-t}\ \left(t + C_4 + \frac{C_1}{C_3}\right) \\ x(t) &= \mathrm{e}^{-t}\ (t + C_5) \qquad (9) \end{aligned}$$

These results were then verified using Maple, which gave the output in Listing 1. This matches with the analytical result in equation 9. Note that the constant `_C1` is the notation used by Maple when adding an arbitrary constant.

```
> DE1 := diff(x(t),t) = -x(t)+exp(-t);

                                        d
                          DE1 := ------ x(t) = -x(t) + exp(-t)
                                        dt

> dsolve(DE1,x(t));

                              x(t) = (t + _C1) exp(-t)
```

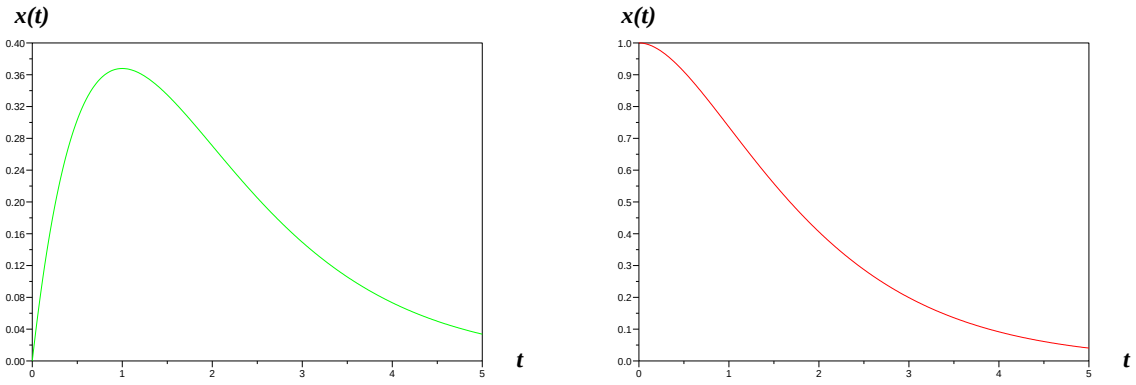
Listing 1: Output from Maple to check the results of the analytical solution.

We then need to solve for the constant. Since this is an initial value problem which is meant to emulate part of the behaviour of a spiking neuron, let us assume the initial value $x_{t=0} = 0$. The function then becomes simply

$$x(t) = t\ \mathrm{e}^{-t} \qquad (10)$$

The graph of this function is shown in Figure 1(a). Alternatively, if we chose the initial value $x_{t=0} = 1$, the curve shifts to the left by one time unit t into the negative time scale, and the x values are slightly amplified (see Figure 1(b)). However, since t is only valid for $t > 0$ in our case, the curve with $x_{t=0} = 0$ looks much more interesting

since it both rises and falls in our time scale $t > 0$, whereas the curve with $x_{t=0} = 1$ only decreases from it's initial value over the same time scale.



(a) $x(t) = t\ \mathrm{e}^{-t}$ (b) $x(t) = (t+1)\ \mathrm{e}^{-t}$

Figure 1: Comparison of the curves for the analytical result $x(t) = (t + C)\ \mathrm{e}^{-t}$ where the initial value is a) $x_{t=0} = 0$ and b) $x_{t=0} = 1$, making the constant $C = 0$ and $C = 1$ respectively. In this paper, we shall assume $x_{t=0} = 0$.

3 Euler Method

The Euler method for numerical approximation of a differential equation uses an iterative technique starting from an initial value[4]. At each time step $t = t' + \Delta t$, the successive values of $x(t' + \Delta t)$ are calculated using the previous values of t' and $x(t')$. In our case, we use the following formula:

$$x(t' + \Delta t) \quad = \quad -x(t') + \mathrm{e}^{-(t'+\Delta t)} \tag{11}$$

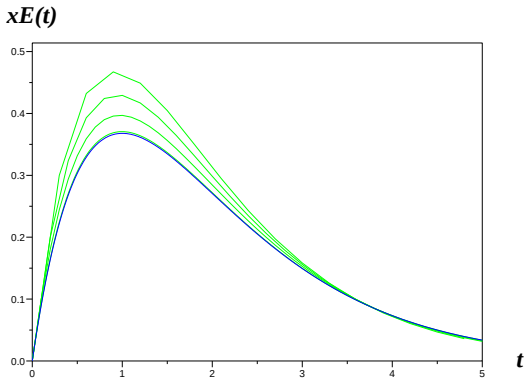
In the Euler method, as the time step between each successive iteration decreases, the accuracy of the approximation generally increases, which is the behaviour we see in Figure 2(b) and Figure 2(c); as we decrease the time step Δt from 0.3 to 0.01, we see closer and closer approximations of the actual curve. When $\Delta t = 0.01$, we have a very close approximation of the true curve.

To further investigate the sensitivity of the resulting curve to the time step Δt, we choose a point where the error in all curves is at its maximum value, when $t = 1$, and plot the value the error reaches for various values of Δt. The resulting plot is shown in Figure 2(c). We can see that the maximum error increases steadily from small values of Δt, then remains constant when $\Delta t > 1$, at which Δt equals the maximum value achieved by $x(t) = t\ \mathrm{e}^{-t}$.

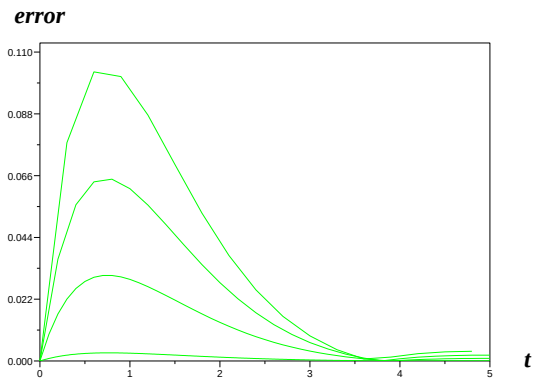
The SciLab code `euler.sci` used to generate these curves is attached.

4 Runge-Kutta Method

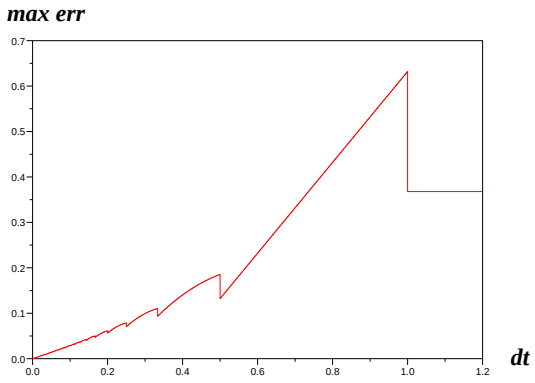
The fourth order Runge-Kutta method is a higher order method for numerical integration that can be more accurate than the Euler method by several orders of magnitude for equivalent step sizes[4]. The Runge-Kutta method uses the first order approximation in a similar fashion to the Euler method, but also takes into account the slope of the curve at each iteration step, ie. the derivative of the curve. The fourth-order Runge-Kutta method takes the second and third



(a) Curves resulting from Euler method.



(b) Error curve for several step values Δt.



(c) Maximum error at $t = 1$ for various step values Δt.

Figure 2: Approximation of $\frac{\mathrm{d}x}{\mathrm{d}t} = -x + \mathrm{e}^{-t}$ using the Euler method. In a), the curve resulting from several values of the time step Δt are shown in green (from the upper curve, $\Delta t = 0.3, 0.2, 0.1, 0.01$ respectively), compared to the actual curve in blue. In b), we show the change in absolute error over time t for these same four values of the step value Δt (from the upper curve, $\Delta t = 0.3, 0.2, 0.1, 0.01$ respectively). In c), we show the effect on the maximum absolute error of varying the time step Δt.

derivatives of the curve into account as well, using the following equations to approximate $x(t)$, restated from [4]:

$$\text{Let } f(x,t) = \frac{\mathrm{d}x}{\mathrm{d}t} \tag{12}$$
$$k_1 = \Delta t\ f(x(t),t) \tag{13}$$
$$k_2 = \Delta t\ f(x(t)+\frac{k_1}{2}, t+\frac{\Delta t}{2}) \tag{14}$$
$$k_3 = \Delta t\ f(x(t)+\frac{k_2}{2}, t+\frac{\Delta t}{2}) \tag{15}$$
$$k_4 = \Delta t\ f(x(t)+k_3, t+\Delta t) \tag{16}$$
$$x(t+\Delta t) = x(t)+\frac{k_1}{6}+\frac{k_2}{3}+\frac{k_3}{3}+\frac{k_4}{6} \tag{17}$$

(Note: there is a small misprint in the equation for k_2 in [4]).

In the last function of `euler.sci` attached, there is a SciLab algorithm for calculating the numerical approximation of our function given in equation 1. SciLab also has this method built-in using the `ode("rk",...)` function, which is equivalent to the MatLab function `ode45(...)`, along with several other approximation methods.

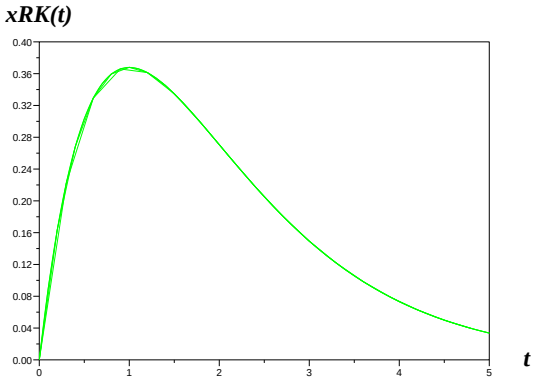
In a similar fashion to the previous section, the curve for $x(t)$ and the error curves are plotted in Figure 3. Since in Figure 3(b) we can see that the maximum error values occur when $t = 0.8$, this value is used for determining the sensitivity to the step value Δt in Figure 3(c). We can see in this figure that the maximum error remains an order of magnitude smaller in comparison to Euler, even when the step values are quite high such as at $\Delta t = 0.6$. The repeating saw-like oscillations in this curve, as with those using the Euler method in the previous section, from approximately $0.01 < \Delta t < 0.8$ are symptomatic of such numerical approximations and are not related to the curve $x(t)$ itself: indeed, performing this same analysis with trigonometric functions such as $\frac{\mathrm{d}x}{\mathrm{d}t} = sin(t)$, which yeilds a translated sine curve for $x(t)$, or simply $\frac{\mathrm{d}x}{\mathrm{d}t} = 3$, which yeilds an increasing straight line for $x(t)$, shows similar behaviour, as the error spikes and then is slowly brought down with slowly increasing step size. As with the Euler approximation of $x(t)$, the errors do not accumulate but rather reach a maximum value of approximately 0.37 when $\Delta t > 1$.

5 Comparison of Results

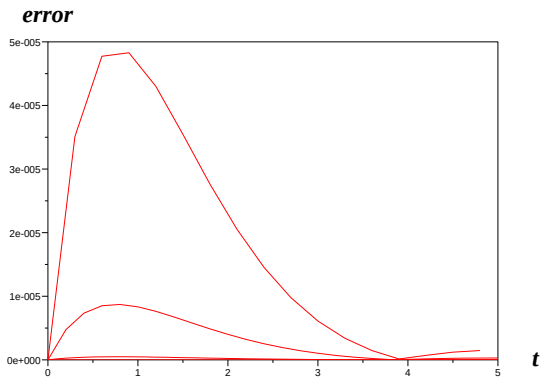
We can see from the error curves in Figure 2(b) and Figure 3(b) that the fourth-order Runge-Kutta method yields significantly better results then the Euler method, by several orders of magnitude, even when fairly large values of the step value Δt are used.

In Figure 4, we show a magnification of the resulting curves at the points of maximum error for both approximations, in comparison to the actual curve attained through analytical means in the first section, with the step size for the Euler method set to three orders of magnitude smaller than the Runge-Kutta approximation. As can be seen, the higher order approximation yields very accurate results, even given this large difference in the step size.

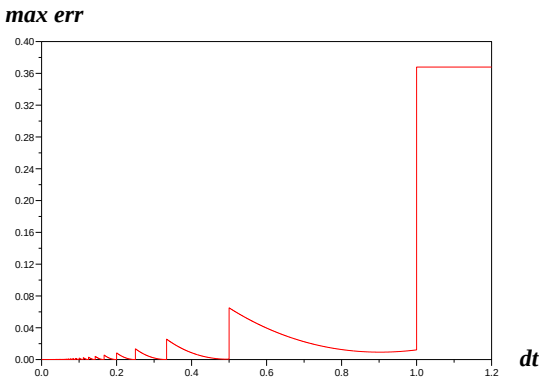
Is there a limit to the accuracy of the fourth-order Runge-Kutta approximation? In Figure 4(a), we show the error curves for the approximation when Δt is very small. Even though these step values are different by an order of magnitude, the maximum error in both cases is approximately the same (albeit extremely minute). These fluctuations may actually be caused by limitations in the precision of the floating point units in SciLab rather than a limitation of the algorithm; further investigation would be required to confirm this.



(a) Curves resulting from fourth-order Runge-Kutta method.

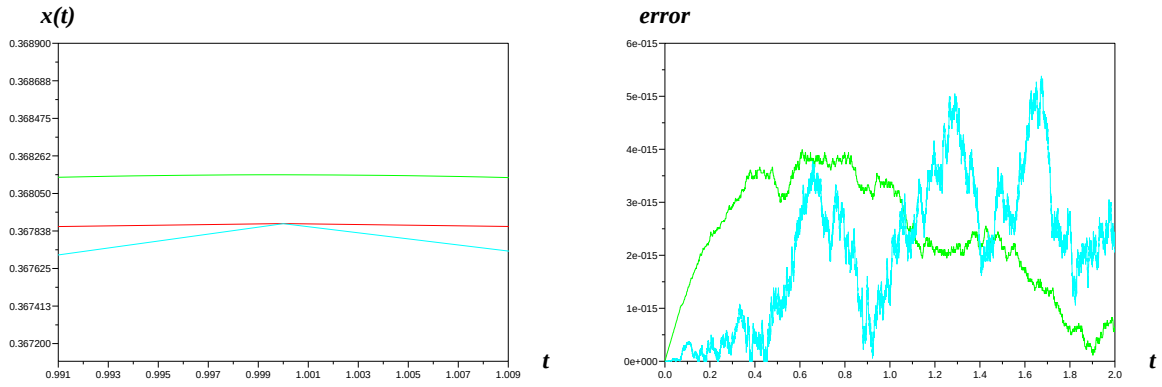


(b) Error curve for several step values Δt.



(c) Maximum error at $t = 0.8$ for various step values Δt.

Figure 3: Approximation of $\frac{\mathrm{d}x}{\mathrm{d}t} = -x + \mathrm{e}^{-t}$ using the fourth-order Runge-Kutta method. In a), the curve resulting from several values of the time step $\Delta t = 0.3, 0.2, 0.1, 0.01$ are shown: to the eye, the curves are virtually identical. In b), we show the change in absolute error over time t for the first three values of the step value $\Delta t = 0.3, 0.2, 0.1$ ($\Delta t = 0.01$ is not plotted, but the maximum error value reached was approximately 4.5 x 10^{-11}). In c), we show the effect on the maximum absolute error of varying the time step Δt, which reaches a maximum value of approximately 0.37 when $\Delta t \geq 1$.



(a) Curves of $x(t)$ resulting from both approximation methods.

(b) Error curves from Runge-Kutta with small step values.

Figure 4: In a) we show a magnification of the actual curve $x(t) = t\ \mathrm{e}^{-t}$ (red), Euler approximation with $\Delta t = 0.001$ (green) and fourth-order Runge-Kutta approximation $\Delta t = 0.1$ (cyan). In b), a comparison is shown for the Runge-Kutta approximation with $\Delta t = 0.0001$ (cyan) and $\Delta t = 0.001$ (green).

References

[1] J. J. O'Connor and E. F. Robertson, "The MacTutor History of Mathematics Archive: Marius Sophus Lie," *http://www-history.mcs.st-andrews.ac.uk/history/Mathematicians/Lie.html*, School of Mathematics and Statistics, University of St Andrews, February 2000.

[2] James Stewart, "Multivariable Calculus, Second Edition," Brooks/Cole Publishing Company, 1987–1991, pp. 926–945.

[3] James Stewart, "Single Variable Calculus," Brooks/Cole Publishing Company, 1987, pp. 508–15.

[4] Thomas P. Trappenberg, "Fundamentals of Computational Neuroscience," Oxford University Press, 2002, pp. 327–32.

[5] Eric W. Weisstein, "First-Order Ordinary Differential Equation," MathWorld–A Wolfram Web Resource, *http://mathworld.wolfram.com/First-OrderOrdinaryDifferentialEquation.html*, 1999-2004.

This document was created using the LaTeX document preparation system.