02685

Scientific Computing for Differential Equations

Assignment 1

Phillip Brinck Vetter (s144097)
Steffen Sloth (s144101)
Raja Shan Zaker Mahmood (s144102)

Group 1
February 18, 2018
Technical University of Denmark

Introduction

This assignment presents answers to assignment 1 of the first weeks during the course. When references are made to the "book", we are referring to the textbook by Leveque [LeVeque, 2007], used in the course.

Exercise 1

a)

We want to determine two finite-difference method (FDM) stencils for approximation of the second derivative $u^{(2)}(x)$. We first find an off-centered stencil, with $(\alpha, \beta) = (0, 4)$. Calling the FDM operator D^2_{04}, we can expand the second order derivative at the point $\bar{x}$ as a weighted sum of the function values at the stencil points.

$$D^2_{04}u(x) = \alpha_1 u(\bar{x}) + \alpha_2 u(\bar{x}+h) + \alpha_3 u(\bar{x}+2h) + \alpha_4 u(\bar{x}+3h) + \alpha_5 u(\bar{x}+4h) \tag{1}$$

The function $u(t)$ is assumed to be sufficiently smooth to have a Taylor series approximation in the neighbourhood considered. The general form of this series for each point $\bar{x}$ in the stencil takes the form

$$\begin{aligned} D^2_{04} &= \alpha_1 u(\bar{x}) + \alpha_2 \sum_{n=0}^{\infty} \frac{h^n}{n!} u^{(n)}(\bar{x}) + \alpha_3 \sum_{n=0}^{\infty} \frac{(2h)^n}{n!} u^{(n)}(\bar{x}) + \alpha_4 \sum_{n=0}^{\infty} \frac{(3h)^n}{n!} u^{(n)}(\bar{x}) + \alpha_5 \sum_{n=0}^{\infty} \frac{(4h)^n}{n!} u^{(n)}(\bar{x}) \\ &= \alpha_1 u(\bar{x}) + \sum_{n=0}^{\infty} \frac{h^n}{n!} \left(\alpha_2 + 2^n \alpha_3 + 3^n \alpha_4 + 4^n \alpha_5\right) u^{(n)}(\bar{x}) \end{aligned}$$

Evaluating the fifth partial sum yields a linear system which we seek to solve such that all but the coefficients of the second derivative vanish while the second derivative coefficient assumes unity. The system takes the form

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 2^2 & 3^2 & 4^2 \\ 0 & 1 & 2^3 & 3^3 & 4^3 \\ 0 & 1 & 2^4 & 3^4 & 4^4 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2/h^2 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

Solving the system we obtain the following values of the coefficients.

$$\alpha_1 = \frac{35}{12h^2}, \alpha_2 = -\frac{26}{3h^2}, \alpha_3 = \frac{19}{2h^2}, \alpha_4 = -\frac{14}{3h^2}, \alpha_5 = \frac{11}{12h^2}$$

The same result can be found by applying the function *fdcoeffV.m*, as exemplified in [LeVeque, 2007], page 11. Using the found coefficients we can write the second order derivative approximation operator as

$$D_{04}u(\bar{x}) = \frac{\frac{35}{12}u(\bar{x}) - \frac{26}{3}u(\bar{x}+h) + \frac{19}{2}u(\bar{x}+2h) - \frac{14}{3}u(\bar{x}+3h) + \frac{11}{12}u(\bar{x}+4h)}{h^2}$$

The values can be obtained in a similar fashion for the stencil with $(\alpha, \beta) = (-2, 2)$. Here the resulting linear system takes the form

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & -2 & -1 & 1 & 2 \\ 0 & 4 & 1 & 1 & 4 \\ 0 & -8 & -1 & 1 & 8 \\ 0 & 16 & 1 & 1 & 16 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_{-2} \\ \alpha_{-1} \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2/h^2 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

with solution

$$\alpha_0 = -\frac{5}{2h^2}, \alpha_{-2} = -\frac{1}{12h^2}, \alpha_{-1} = \frac{4}{3h^2}, \alpha_1 = \frac{4}{3h^2}, \alpha_2 = -\frac{1}{12h^2}$$

and associated second order derivative approximation operator

$$D_{-22}u(\bar{x}) = \frac{\frac{5}{2}u(\bar{x}) - \frac{1}{12}u(\bar{x}+h) + \frac{4}{3}u(\bar{x}+2h) + \frac{4}{3}u(\bar{x}+3h) - \frac{1}{12}u(\bar{x}+4h)}{h^2}$$

The provided MATLAB-program *fdcoeffV.m* is used in the implementation to obtain these coefficients for any given stencil by the same method as have been covered here.

b)

The order of accuracy is found by computing the leading order coefficient of the Taylor expansion for the local truncation error (LTE). Considering the first stencil we find

$$\begin{aligned} \tau(\bar{x}) = D^2_{04}u(\bar{x}) - u^{(2)}(\bar{x}) &= \frac{1}{h^2}\left[\alpha_1 u(\bar{x}) + \sum_{n=0}^{\infty} \frac{h^n}{n!}\left(\alpha_2 + 2^n\alpha_3 + 3^n\alpha_4 + 4^n\alpha_5\right)u^{(n)}(\bar{x})\right] - u^{(2)}(\bar{x}) \\ &= \frac{1}{h^2}\left[\sum_{n=5}^{\infty} \frac{h^n}{n!}\left(\alpha_2 + 2^n\alpha_3 + 3^n\alpha_4 + 4^n\alpha_5\right)u^{(n)}(\bar{x})\right] \\ &= \frac{h^3}{5!}(\alpha_1 + 2^5\alpha_2 + 3^5\alpha_3 + 4^5\alpha_4 u^{(5)}(\bar{x}) + O(h^4) = \underline{\underline{\frac{5}{6}h^3u^{(5)}(\bar{x}) + O(h^4)}} \end{aligned}$$

Applying the same principle we find the leading order of the second stencil to be

$$\tau(\bar{x}) = D^2_{-22}u(\bar{x}) - u^{(2)}(\bar{x}) = \underline{\underline{-\frac{1}{90}h^4u^{(6)}(\bar{x}) + O(h^5)}}$$

It is clear that the accuracy for the D^2_{-22} stencil is highest. The possible explanation why could be due to its symmetric properties of the stencil around the expansion point. This is in analogy to how the centered finite difference scheme has a higher order of accuracy than the forward and backwards finite difference schemes.

c)

Firstly we note that the function of interest ($u(x) = \exp(\sin(x))$) has a analytic solution of the form:

$$\frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = \exp(\sin(x))\cos(x)^2 - \exp(\sin(x))\sin(x) \tag{4}$$

which can be used to visualize if the approximation converges toward the correct solution, and later for estimating the local truncation error of the approximation.

We use the hint and implement the Fornberg method using the *fdcoeffF.m* script from the books web-page. We use a centred stencil with two points in each direction (a total of five points) to approximate the second order derivative. The function *fdcoeffF.m* takes three inputs: *1)* The order of the method, here second order ($k = 2$). *2)* A point $\overline{x}$ (call *xbar* in the code) here $\overline{x} = \pi/2$. *3)* A vector x with the desired stencil points. We calculate five coefficients (denoted c_i) around the point $\overline{x}$, and use them to calculated the approximation of the second order derivative in the point $\overline{x}$ by summing the product of the coefficients and the function values in the stencil points (the five points):

$$u''(\overline{x}) \approx c_1 u(\overline{x} - 2h) + c_2 u(\overline{x} - h) + c_3 u(\overline{x}) + c_4 u(\overline{x} + h) + c_5 u(\overline{x} + 2h) \tag{5}$$

where h is the step size.
The implemented code actually calculates the second order derivative for every point in the x-interval using coefficients evaluated around the point $\overline{x}$. This makes it possible to plot the approximation compared to the true solution to the second order derivative such that it can be visualized if the code works correctly. The result is shown in Figure 1. The approximation is seen to fit

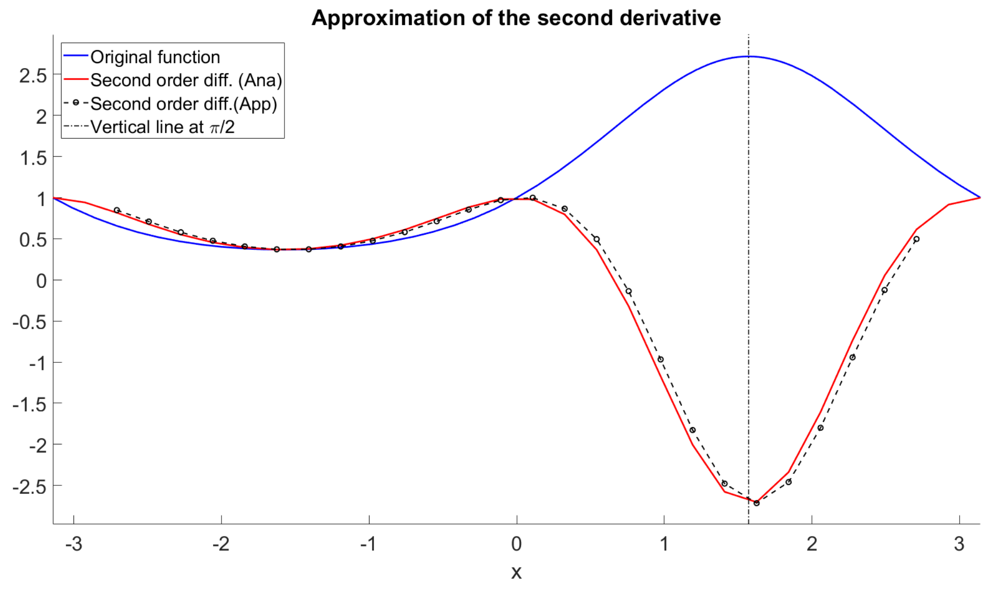


Figure 1: Plot of the function of interest ($u(x)$), its true derivative, and an approximation to its derivative using the Fornberg method with a centred stencil of 5 points evaluated around the point $\overline{x} = \pi/2$. All on a grid in the interval $\pm\pi$ with a grid step size of $\pi/15$.

the true solution very nicely inside the interval as wanted. Looking at the function values at the point $\overline{x} = \pi/2$ one finds that the true solution yields $u^{(2)}_{true}(\pi/2) = \exp(\sin(\pi/2))\cos(\pi/2)^2 - \exp(\sin(\pi/2))\sin(\pi/2) = -\exp(1) \approx -2.7024$ while the approximation yields $u^{(2)}_{app.}(\pi/2) = -2.7146$.

The variation from the true result is $\Delta u = 0.0122$. In conclusion the approximation is accurate up to two significant figures for the provided $u(x)$ with the chosen parameters and methods so the code appears to behave as intended.

d)

The convergence test is carried out using a slightly modified code (compared to the one used in part 1.c). The modified version uses an x of just the five points needed for the stencil i.e computations are only carried out in the wanted neighbourhood of $\bar{x}$. This greatly reduces the required computations. The estimation of the local truncation error in the point $\bar{x}$ is found using the method discussed in section 1.1 in the book. The approximated value is subtracted from the true second order derivative as demonstrated below.

$$\tau(\bar{x}) = D_f u(\bar{x}) - u^{(2)}(\bar{x}) - = Ch^4 u^{(5)}(\bar{x}) + O(h^5) \tag{6}$$

Here D_f is the operator associated with the Fornberg method. Its clear from equation (6) that the expected error is of fourth order in h. The associated graph would then be a straight line under a log-log transformation of the form

$$\log(\tau(\log(\overline{x}))) \approx \log(Cu^5(\bar{x})) + 4\log(h) \tag{7}$$

The scheme runs in a for-loop in which the step size h takes the values $h = 1/2^s$ for $s = [2; 8]$. The resulting plot of the local truncation error vs. step size together with the curve h^4 in a double-log plot is seen in Figure 2. The local truncation error exhibits the expected behavior with a rate of

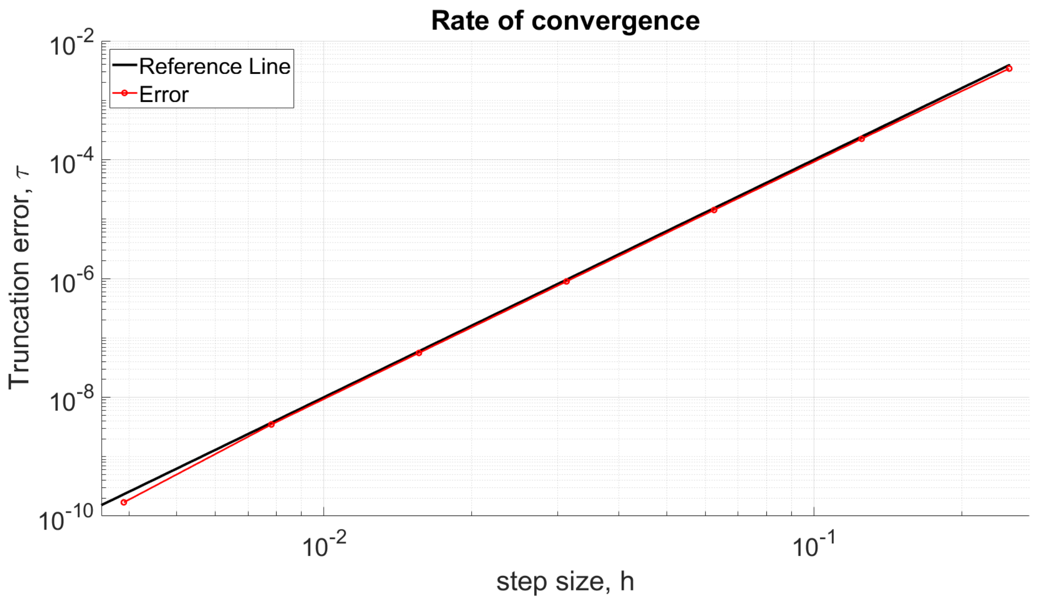


Figure 2: A double-log plot of the rate of convergence. The red line is the local truncation error for different step seizes ($h = 1/2^s$ for $s = [2; 8]$), and the black line is a reference line of the expected fourth order in h dependence (h^4). The rate of convergence can be seen to fit the expectation nicely.

convergence that lies almost on top of the reference line thus indeed seems to be of fourth order in h.

Exercise 2

a)

We consider the following ordinary differential equation (ODE) with specified initial conditions (IC) is given by the following.

$$y'(t) = 4t\sqrt{y(t)} \qquad\qquad y(0) = 1 \tag{8}$$

The first relevant question is of course if there exist a unique solution to such an initial value problem (IVP). This can be guaranteed if we can bound the function $f(y(t), t) = 4t\sqrt{y(t)}$ with respect to the variable $y(t)$ on the interval considered. Consider the following bound on the function values with respect to y. The dependence on t can be ignored in this case.

$$\begin{aligned} |f(y,t) - f(\tilde{y},t)| &= \left|4t\sqrt{y} - 4t\sqrt{\tilde{y}}\right| \\ &= \left|\frac{4t}{\sqrt{y} + \sqrt{\tilde{y}}}\right| |y - \tilde{y}| \end{aligned}$$

Thus we have found a Lipschitz constant $L \geq \left|\frac{1}{2}\right|$ suitable for the problem. The value of $y(0)$ is crucial here as the square root function ceases to be Lipschitz continuous at $y(t) = 0$. Having found L the Picard-Lindelöf theorem guarantees existence and uniqueness for the solution to the IVP on an interval with $y(0) \geq 1$. We proceed to find the unique solution to the IVP by separation of the variables.

$$\begin{aligned} \frac{dy}{dt} &= 4t\sqrt{y} \\ &\rightarrow \\ \frac{dy}{\sqrt{y}} &= 4tdt \\ &\rightarrow \\ \int_1^y \frac{1}{\sqrt{y^*}} dy^* &= \int_0^t 4t^* dt^* \\ &\rightarrow \\ \left[2\sqrt{y^*}\right]_1^y &= \left[2t^{*^2}\right]_0^t \\ &\rightarrow \\ 2\sqrt{y} - 2 &= 2t^2 \\ &\rightarrow \\ y = t^4 + 2t^2 + 1 &= \left(t^2 + 1\right)^2 \end{aligned} \tag{9}$$

b)

To demand that a numerical scheme solves an IVP exactly is equivalent to demanding that the local truncation error (LTE) of the scheme vanishes for every step of the solver. The local truncation error is given by

$$\tau(t_{n+r}) = \sum_{q=0}^{\infty} k^{q-1} \left(\sum_{j=0}^{r} \left(\frac{1}{q!} j^q \alpha_j - \frac{1}{(q-1)!} j^{q-1} \beta_j \right) \right) u^{(q)}(t_n). \tag{10}$$

Thus in this case for a method to solve the problem exactly it must be 4th order accurate i.e it must satisfy the order conditions $C_q = 0$ for $q \in \{0, 1, 2, 3, 4\}$. This is sufficient since the solution $u(t)$ belongs to the class of fourth order polynomials whose derivatives $u^{(k)}$ vanish for $k > 4, k \in \mathbb{N}$. The linear system of equations that arises should have solutions for a stencil using a number of points greater than $r = 5$ however using coefficients $\{\alpha_0, \alpha_1, \beta_0, \beta_1, \beta_2\}$ resulted in the trivial solution, thus it was necessary in order to obtain a proper solution to include a fixed $\alpha_2 = 1$. Further details on the derived FDM is provided in **d)**.

c)

We now consider the altered differential equation given by

$$y' = 4t\sqrt{y} - \lambda \left(y - \left(1 + t^2\right)^2 \right) \qquad\qquad y(0) = a \tag{11}$$

where $(\lambda, a) \in \mathbb{R}^2$. To find the cases where a numerical method that solves **b)** also exactly solves (11) we consider the following situations: When $a = 1$, (9) also solves this problem exactly regardless of the value of λ since the second term vanishes. Additionally the uniqueness property guarantees that that is the *only* solution. When $a \neq 1$ the second term does not vanish however. In this case in order to ensure that the same method will solve the equation exactly we need to require $\lambda = 0$. We therefore have the two possible cases

$$(a, \lambda) = (1, \mathbb{R}), \qquad\qquad (a, \lambda) = (\mathbb{R}^+, 0) \tag{12}$$

d)

The conditions in order to achieve a convergent LMM are described in **b)**. The first five order-coefficients have to equal zero and the roots of the resulting characteristic polynomial must be zero-stable. These criteria were satisfied by constructing a system of five unknowns either belonging to the right-hand-side (α's) or belonging to the left hand side (β's) of the IVP. The chosen method had coefficients $\{\alpha_0, \alpha_2, \beta_0, \beta_1, \beta_2\}$. The associated LMM then assumes the form

$$\alpha_2 U^{n+2} + \alpha_0 U^n = h \left(\beta_2 f(U^{n+2}, t_{n+2}) + \beta_1 f(U^{n+1}, t_{n+1}) + \beta_0 f(U^n, t_n) \right). \tag{13}$$

We note that this scheme is implicit since U^{n+2} is on both the right-hand side and left-hand side. Using (10) we arrived at the linear system given by

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & -1 \\ 0 & 1/2 & 0 & -1 & -2 \\ 0 & 1/6 & 0 & -1/2 & -2 \\ 0 & 1/24 & 0 & -1/6 & -8/6 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ -2 \\ -8/6 \\ -16/24 \end{bmatrix} \tag{14}$$

which has the non-trivial solution (where $\alpha_2 = 1$ has yielded the right hand side)

$$\{\alpha_0 = -1, \alpha_1 = 0, \alpha_2 = 1, \beta_0 = \frac{1}{3}, \beta_1 = \frac{4}{3}, \beta_2 = \frac{1}{3}\} \tag{15}$$

The roots of the characteristic polynomial to the homogeneous difference equation are then found from

$$\begin{aligned} \rho(\xi) &= \alpha_2\xi^2 + \alpha_1\xi + \alpha_0 \\ &= \xi^2 - 1 = 0 \\ \Rightarrow &\xi_1 = 1 \qquad \xi_2 = -1. \end{aligned} \tag{16}$$

These root are distinct and obey the root condition ($|\xi| \leq 1$) so we infer that the system is zero-stable. In summary this means that the presented LMM is *convergent* according to Dahlquist theorem (Theorem 6.3 in the book), as it is both *consistent* and exhibits zero-stability with the chosen coefficients. It should be noted that we initially attempted to find an explicit LMM which was convergent however while consistency was straight-forward to achieve we failed to discover a scheme which was at the same time zero-stable.

e)

Before writing the code we need to make sure that we have a method that works for all value on an interval. Looking at eq. 13 we realize that we need two previous points to estimate the next, which gives a problem when starting. The first value should be the initial condition, but to estimate the third value we end the second, due to the term with β_1, and we cannot get this from our current method!

We need a method to estimate the second value, which has the properties discussed in part 2,b to match our LMM. Here we look up the fourth order Runge Kutta method in the book, which is exact to fourth order and only needs one previous value to estimate the next. The scheme of the Runge Kutta method can be seen in example 5.13 in the book (page 126).

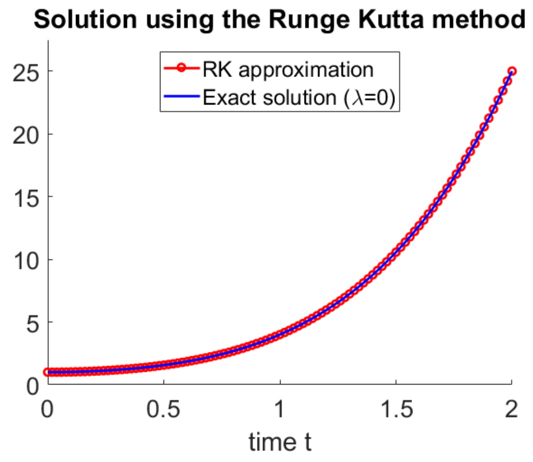
Going forward with the Runge Kutta method and our LMM, we now have a method where we get the first value from the initial condition, the second value from the Runge Kutta method (using the initial condition), and then use our LMM to estimate all the following values in the interval.

We use the suggestion on the last page of the assignment for which inputs to enter into our matlab function. So we build a solver which takes an function as a function handle (*func*), the start and end points of the time intervals (*tspan*), the desired number of step in the interval (n), and the initial condition (*y0*).
In the matlab function we start by defining the interval as a *linespace* and the find the step size (h). We set the first two values of the approximation (*yout*) as discussed above using initial condition and the Runge Kutta method. Then we run out LMM (in a for-loop) for all other values in the interval (starting at three). First in the loop we calculate all known terms (terms with $\{\alpha_0, \beta_0, \beta_1\}$). Then since out LMM is implicit we need to solve the eq. 13, which we turn into a zero-value problem by subtracting the right hand side on both sides, and using the matlab function *fzero* to solve for U^{n+2} (called z). Note that for some cases eq. 13 has no real solution, so a try-catch statement is used, to simply stop the program is *fzero* does not find any solution.
This code should be able to solve the desired equations exact, with the specification given in part 2,b, so now we just need to test if it works. Note that the *fzero* introduces an error to the method.

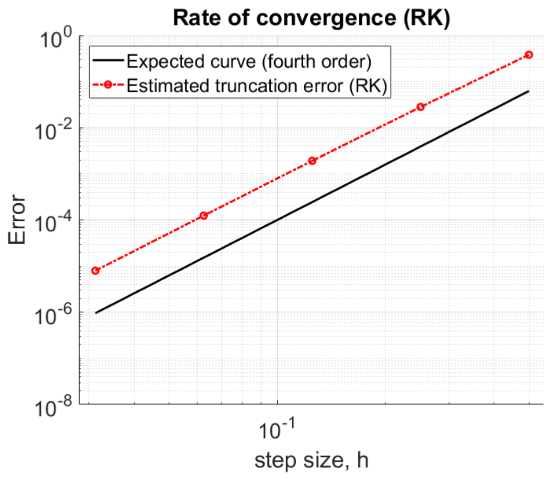
Code validation

It is important to validate the code, first to insure that there is no mistakes from typos, and to see if the method used works at all. To insure the robustness of the code it is important to analyze the rate of convergence. The rate of convergence can be off either by mistakes in the derivation of the method, or by unexpected error sources as e.g. numerical noise/round-off error.

First we need to check if the Runge Kutta method works, and that it have the correct rate of convergence. We do this by trying to solve eq. (5) in the assignment using only the Runge Kutta method (script *RKValidation* and *RKsolver*), for then to compare it with the exact solution found in part 2,a, which is illustrated in figure 3,a.



(a) Red curve shows the resulting Runge Kutta approximation (form the *RKsolver*) to the problem, which has the exact solution shown by the blue curve. The two curves can be seen to close to identical.



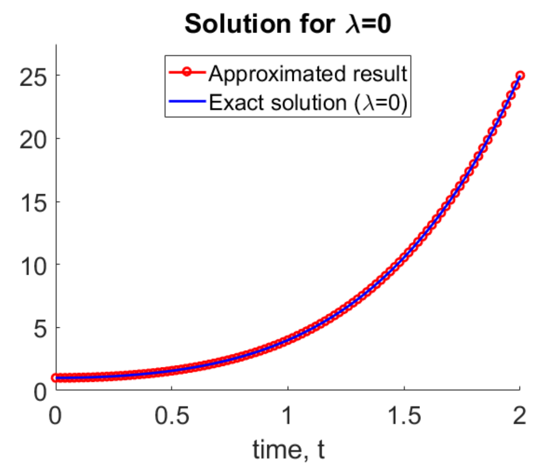
(b) Rate of convergence for the Runge Kutta method, where the red line is the estimated error and the black line the expected fourth order error.

Figure 3: Both plot a and b shows a nice match be between the approximated solution from the Runge Kutta method, and the exact and expected solution from theory.

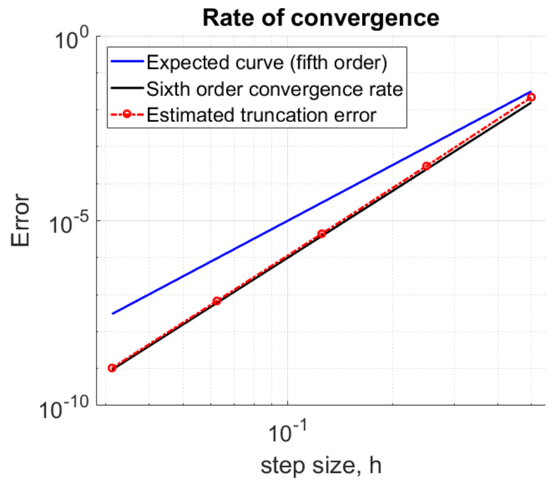
Figure 3 shows how the Runge Kutta method alone works nicely to approximate a solution (with $\lambda = 0$ and $y(0) = 1$). Running the script for different values of the step size (h), and subtracting the approximated value from the true solution gives us an estimate of the truncation error, a plot of this is shown in 3,b, where the rate of convergence can be seen to match the expected fourth order behavior.
This indicates the the Runge Kutta method works for our LMM. Since we only use the Runge Kutta method for a single step, would we expect the rate of convergence introduced by the Runge Kutta step to be an order higher (fifth order) then what is seen in figure 3,b.

Now we must validate the entire solver, which we again do by testing its ability to solve eq. (5) in the assignment, and comparing it to our exact solution found in part 2,a.

Figure 4 shows the result from our LMM-script. It is seen in plot (a) that it produces a nice approximation to the exact solution, however the rate of convergence is not as expected. We



(a) Red curve shows the resulting approximation using out LMM (form the *LMsolver*) to the problem, which has the exact solution shown by the blue curve. The two curves can be seen to close to identical.



(b) Rate of convergence for our LMM, where the red line is the estimated error and the blue line the expected fifth order error. The black line showing a sixth order convergence rate is seen to fit the estimated errors almost exact!

Figure 4: Our LMM seems to be able to produce an good approximation of the exact solution as hoped. However the rate of convergence is not as expected, it follows a sixth order curve better than the expected fifth order curve!

would expect the rate of convergence to be fifth order, since it should propagate the error from the single Runge Kutta step (fifth order local truncation error) through all the steps.
This rate of convergence is unexpected, since without the Runge Kutta step, should our solver be exact for these equations, so we expected only the Runge Kutta and numerical inaccuracy to effect the results. The unexpected part is that we get higher order, so our method is actually better than expected.

f)

We solve the differential equation given in (11) subject to a variety of conditions provided by all possible combinations of $a = \{1, 2, 10\}$, $\lambda = \{0, 1, 20\}$ and $h = \{0.025, 0.05, 0.1\}$. We are interested in inspecting the general properties of the solver on this differential equation as well as the behaviour that might occur as both the initial condition and the value of λ varies from the conditions described in (12). The following three figures (Figure 5, Figure 6, Figure 7) show these combinations. Each figure holds a constant step-size, within which the value of λ varies across the three plots. Each plot consists of three graphs, one for each value of a.
Inspection of the three figures reveal that in the first case of $\lambda = 0$ the solutions behave similarly to that of (8) as expected. The influence of a is a constant separation as explained by the derived result in **g)** provided in (10). This behaviour is not seen in the case of $\lambda = 1$ (the graphs do not overlap and only one color is seen) since the exponential causes the difference between the various solutions to the initial conditions to vanish fast. Despite the fact that this effect is dramatically enhanced in the latter case of $\lambda = 20$ the difference between the graphs in the immediate beginning is seen more clearly due to the scaling of the axis. Equivalent scaling for $\lambda = 1$ would reveal that the individual solutions converge much slower towards one another. For the solutions to $\lambda = 20$ we

see large oscillations that increase in size as time increases which is an indication of issues with the *absolute stability*. Notice also that the time domain for these cases are dramatically reduced and that as a increases in value the time domain for which the graph is plotted shrinks. This shrinkage also seems to be heavily influenced by step size. This is evident from the fact that for $h = 0.1$ the solution of $a = 10$ is only available until $t \approx 0.1\,\mathrm{s}$ while it is available only until $t \approx 1.3\,\mathrm{s}$ for $h = 0.025$. In practice this is a reflection of the inability for the MATLAB **fzero** function which we have used in our implementation to find the next-step solution U^{n+2} of the implicit problem

$$\underbrace{\left[\alpha_2 U^{n+2} - \beta_2 f^{n+2}(U^{n+2}, t^{n+2})\right]}_{\text{unknown term}} + \underbrace{\left[\alpha_0 U^n - \beta_0 f^n(U^n, t^n) - \beta_1 f^{n+1}(U^{n+1}, t^{n+1})\right]}_{\text{known term}} = 0 \qquad (17)$$

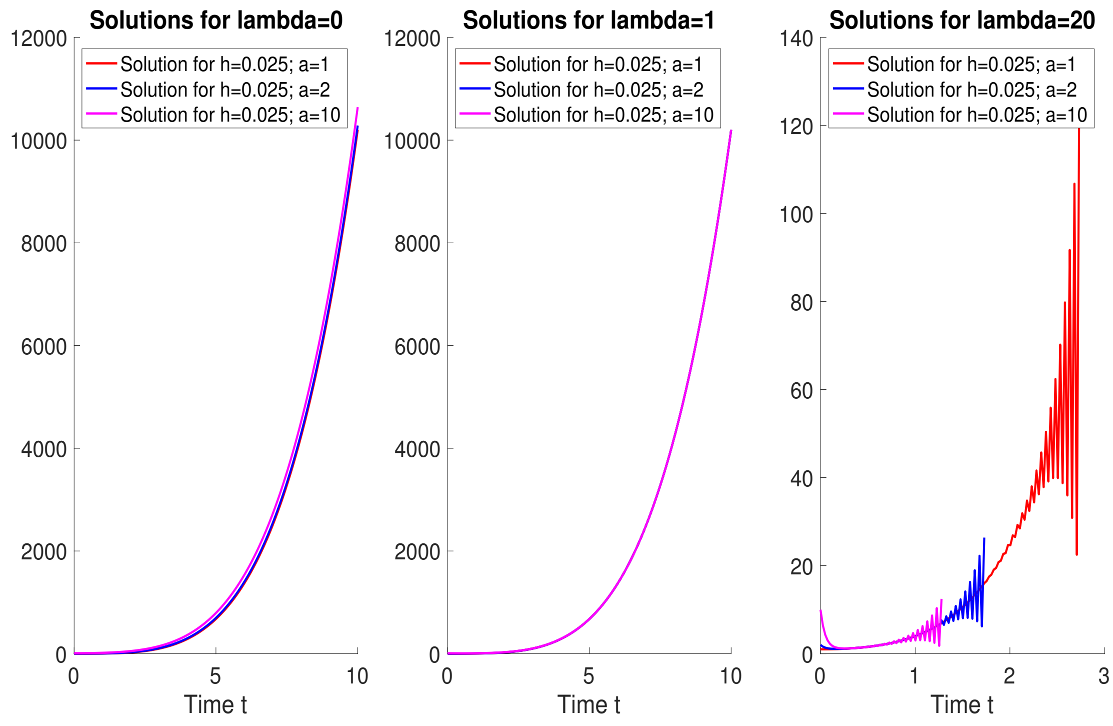


Figure 5: The solution to (11) with step size $h = 0.025$. The values of λ and a are varied across the three plots as explained by the associated titles and legends.

The absolute stability properties of our LMM can be investigated by looking at the roots ξ_i of the stability polynomial (18) whose norm fulfill $\|\xi_i\| = 1$. These define the boundary of the *region of absolute stability*. If the product $h\lambda$ does not lie within this region the numerical solution will oscillate with time and become increasingly inaccurate as we have seen emphasized in the case for $\lambda = 20$. The polynomial in this case is given by

$$\pi(\xi, h\lambda) = \rho(\xi) - (h\lambda)\sigma(\xi) = 0 \qquad \Leftrightarrow \qquad h\lambda = \frac{\xi^2 - 1}{\frac{1}{3}\xi^2 + \frac{4}{3}\xi + \frac{1}{3}} \qquad (18)$$

The boundary of the region can be found by parametrizing $\xi = \exp(i\theta)$. The region in the complex plane for this problem is shown in Figure 8. It is immediately clear from inspection that no values of $h\lambda$ will lie inside this region since h is real by definition and a $\lambda \in \mathbb{C}/\mathbb{R}$ will lead to a complex

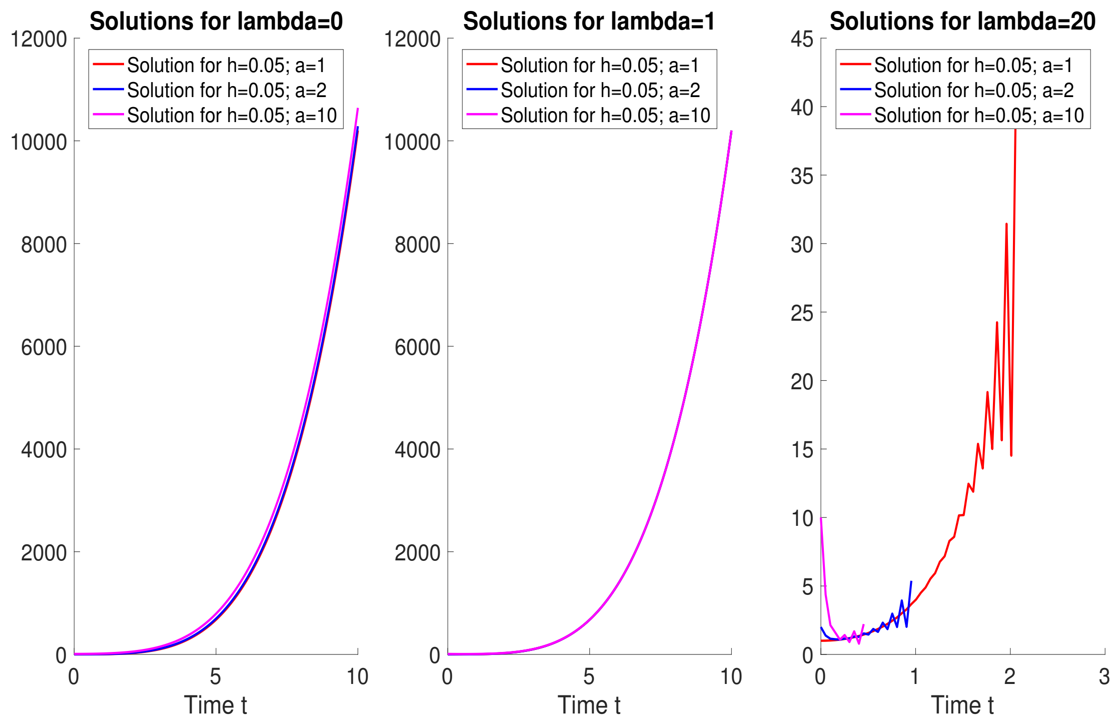


Figure 6: The solution to (11) with step size $h = 0.05$. The values of λ and a are varied across the three plots as explained by the associated titles and legends.

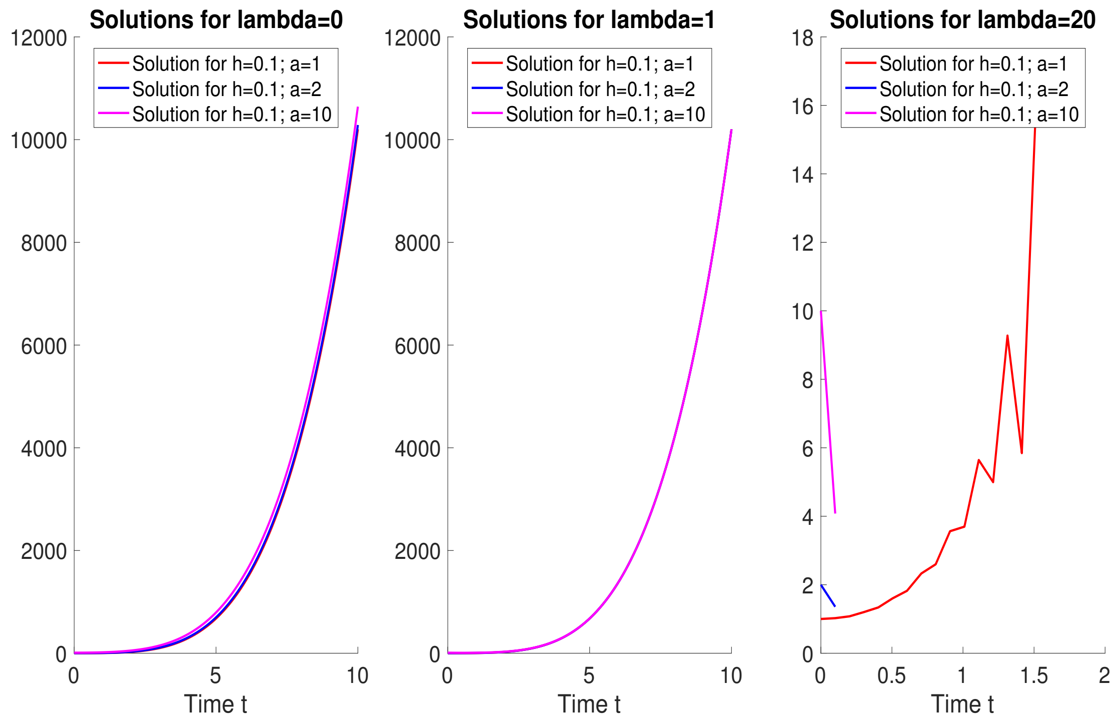


Figure 7: The solution to (11) with step size $h = 0.1$. The values of λ and a are varied across the three plots as explained by the associated titles and legends.

valued solution which is unwanted. As a consequent it seems that there is no way to ensure that the solution does not start to oscillate. At a first glance this does not seem to correspond with the observations of the solutions for $\lambda = 1$ however this is due to the relatively small value of λ. Figure 9 clearly demonstrates that for adequately large values of t the solutions also oscillate. The effect seems to be additionally pronounced for larger values of h possibly indicating that the degree of the oscillations increases as the distance between the region of absolute stability and $h\lambda$ increases.

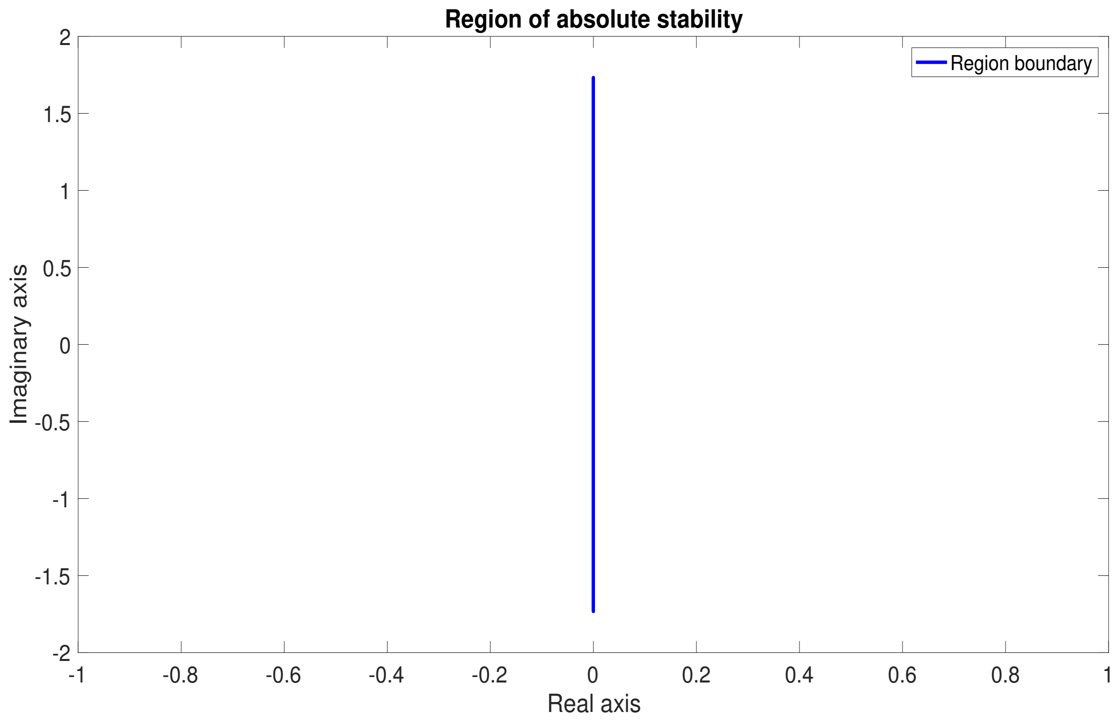


Figure 8: The boundary for the region of absolute stability. The boundary happens to span the region itself as it is just a straight line in the interval $I \in [-\frac{2\pi}{3}i, \frac{2\pi}{3}i]$

The time domain in which the solution exists decreases rapidly as λ increases and thus a serious drawback of the implemented method have been emphasized. It is clear that if we wish to solve (11) for non-trivial values of λ for any longer time domain another method would have to be used instead. Two possible candidates could be either the Adams-Moulton or Adams-Bashfort methods judging from the presented stability regions presented in the slides ***Absolute Stability*** of the fifth lecture of the course. They both exhibit absolute stability for some values of λ along the negative real axis thus it would be possible to solve the system in most cases by appropriately scaling the step size. Note that this rightly corresponds to positive values of λ in this case, as the derivation is based on the test fucking where the sign of λ is changed.

g)

The result in the previous subsection can be explored by introducing the function $z(t)$, into the modified problem.

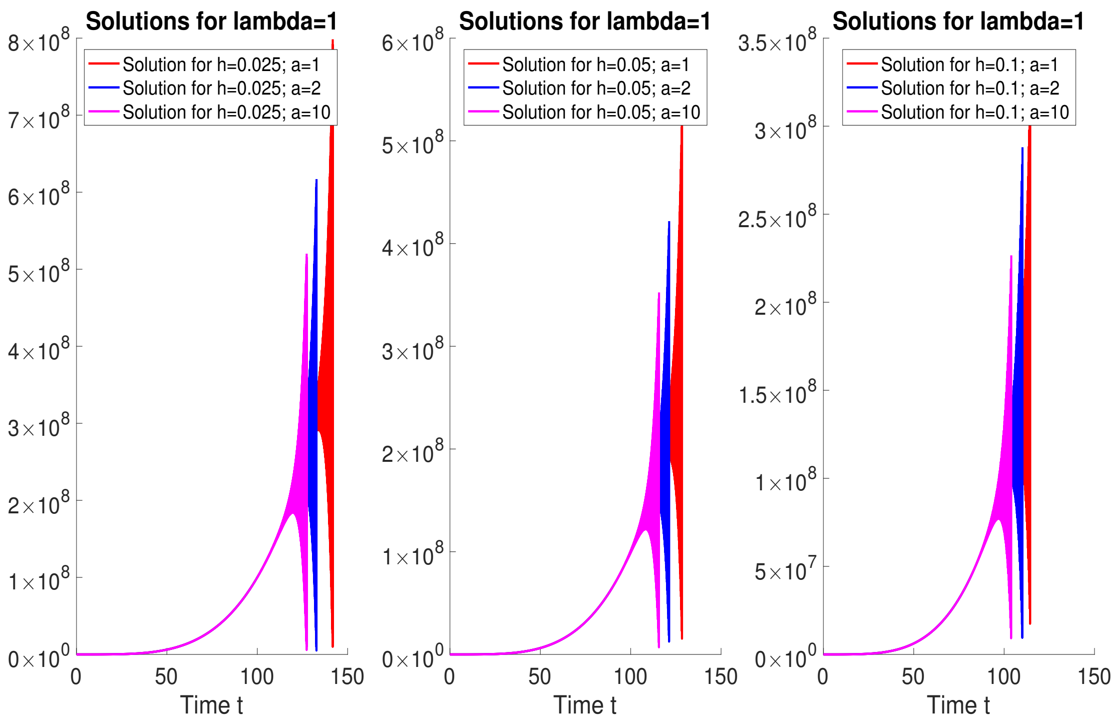


Figure 9: Solutions to (11) for $\lambda = 1$ for varying step size. The figure reveals that for large time scales oscillations also occur and that the error accumulation is influenced by the magnitude of the step size.

$$z(t) = y(t) - g(t) \tag{19}$$

Where $g(t)$ is the solution to the simplified problem (8). We insert this into the equation to obtain

$$(g(t) - z(t))' = 4t\sqrt{g(t) - z(t)} - \lambda(z(t)) \tag{20}$$

The square root can be approximated using the assumption $z(t) \approx 0$ into $\sqrt{g(t) - z(t)} = \sqrt{g(t)}$ and since the differential operator is linear we obtain

$$z'(t) - g'(t) = 4t\sqrt{g(t)} - \lambda z(t). \tag{21}$$

However we have that since $g(t)$ solves (8) $g'(t) = 4t\sqrt{g(t)}$ so this reduces to the simple homogeneous differential equation

$$z'(t) = -\lambda z(t) \tag{22}$$

We then see that the solution to this equation is an exponential. Applying the initial condition $z(0) = y(0) - g(0) = a - 1$ yields the particular solution

$$z(t) = (a - 1)\exp(-\lambda t) \tag{23}$$

Plotting the exponential solution together with a solution to eq. 11 shows how $z(t)$ influences the solution as time goes. Figure 10 shows how the $z(t)$ decreases as time goes for $\lambda = 20$. It can

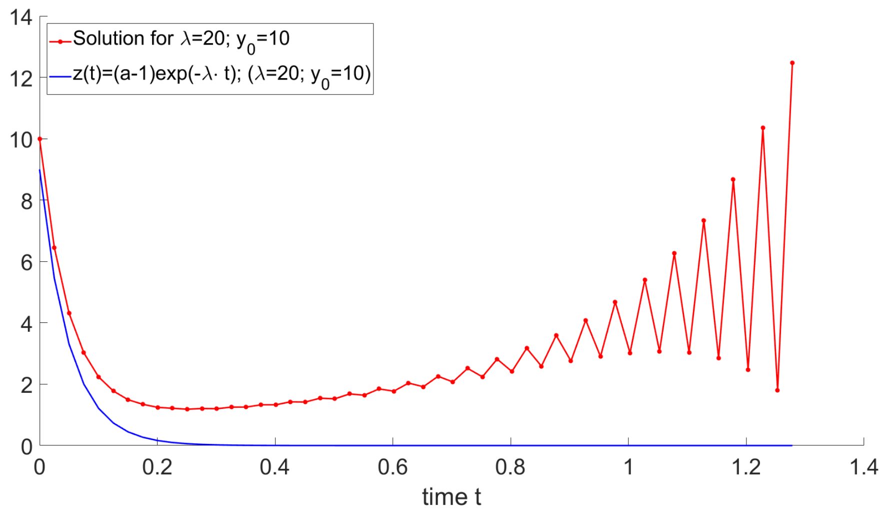


Figure 10: Plot of the solution from our LMM (red), and the size/influence noise $z(t)$ for $\lambda = 20$ and initial condition $y(0) = 10$.

be seen that any numerical noise ($z(y) \neq 0$) is being killed as time goes, and the solutions will converge toward the as time goes for any initial condition. This agrees with what is seen in the figure in part 2,f.

References

[LeVeque, 2007] LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics.