MATH 286 Lab Report

Instructor: Thomas Honold

Group TA: Yuan Yue

Group members:

Li Chenhao	3190110385
Li Zihao	3190110098
Shen Keyi	3190110386
Zhu Kangyu	3190110361
Zhu Xiaohan	3190110352

We declare that this report is our own original work, and every work group member has a fair share in this work. For the preparation of the report we have not used any other resources than the Math 286 lecture material and the references cited in the report.

signs：

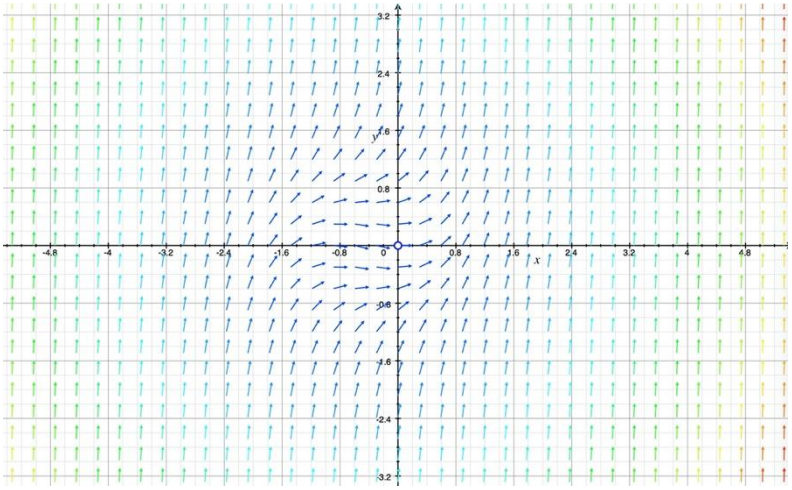
Problem statement:

$$y' = y^2 + t^2 + t, \qquad y(0) = 1 \qquad \text{(IVP1)}$$

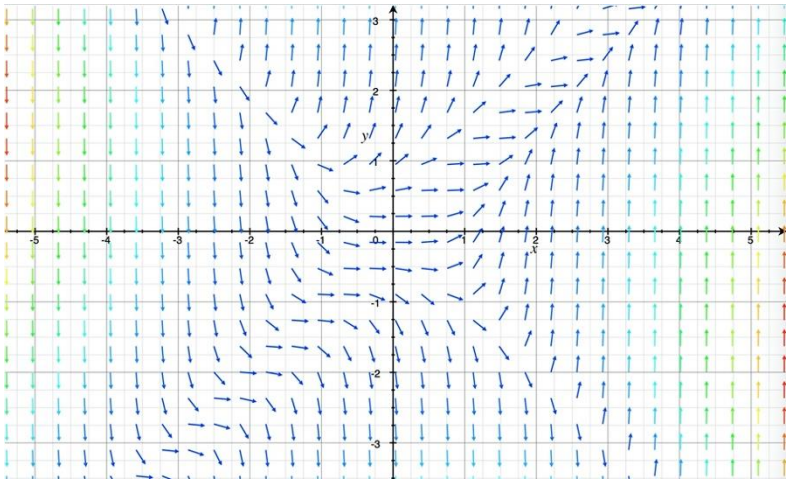
$$y' = (y - t)(y^2 - t^2) \qquad y(0) = 1 \qquad \text{(IVP2)}$$

Using a combination of numerical and analytical methods, determine the maximal solutions of (IVP1), (IVP2) as accurately as possible, including an accurate computation of their domains (a1, b1) and (a2, b2), respectively.

vector field:



vector field of IVP1



vector field of IVP2

1. Euler Method

1.1 Introduction

Euler method is probably the simplest way to solve a differential equation numerically. For the IVP

$$y' = f(t, y),\ y(t_0) = y_0 \qquad (1)$$

Euler method simply takes

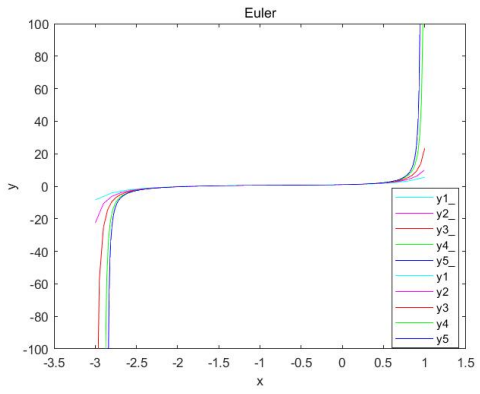
$$y(t + h) = y(t) + f(t, y(t)) * h \quad (2)$$

where h is the step size, positive or negative.

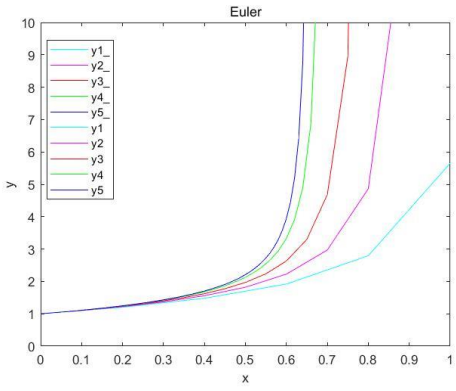
1.2 Result

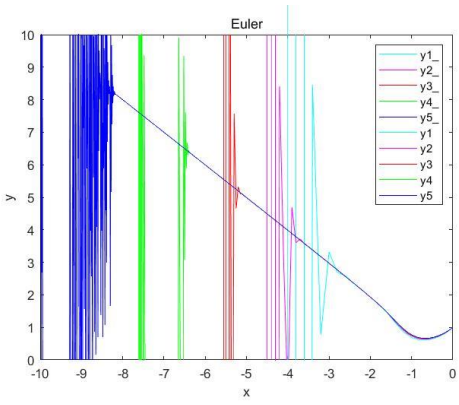
Euler method can be implemented very fast, but is very imprecise, sometimes even not applicable. For IVP1 Euler method is applicable, but for IVP2 it fails on the negative part.

IVP1:



IVP2:





1.3 Error analysis

Euler method comes from the Taylor approximation:

$$y(t + h) = y(t) + y'(t) * h\ + \frac{1}{2} * y''(c) * h^2 \qquad (3)$$

where $c \in [t, t + h]$. Therefore, for a small h we can ignore the quadratic term.

We assume that this is the only source of error, then

$$global\ error\ = \frac{1}{2} \sum_{k=1}^{M} y^{(2)}(c_k) * h^2 \approx \frac{1}{2} M y^{(2)}(c) h^2 = \frac{1}{2}(b - a) y^{(2)}(c) * h$$

Here we can see that the global error is within $O(h)$.

2. Improved Euler Method

2.1 Introduction

The improved Euler method is based on Euler method. For the IVP (1), improved Euler method takes

$$y\_predict(t+h) = y(t) + f(t, y(t)) * h \qquad (4)$$

$$y(t+h) = y(t) + \frac{h}{2}(f(t, y(t)) + f(t+h, y\_predict(t+h)) \qquad (5)$$

where h is the step size, positive or negative.

2.2 Theoretical background

Improve Euler Method comes from

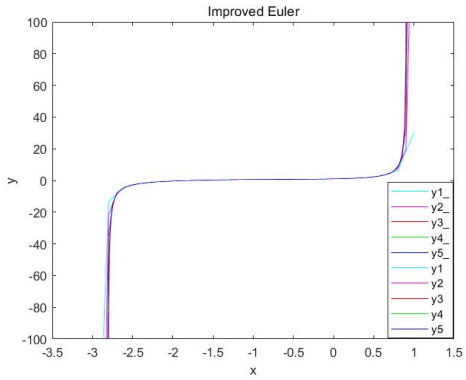
$$y(t_1) = y(t_0) + \int_{t_0}^{t_1} f(t, y(t))dt \approx y(t_0) + \frac{h}{2}(f(t_0, y(t_0) + f(t_1, y(t_1)) \qquad (6)$$

the "≈" comes from the approximation of integral by area of trapezoid.

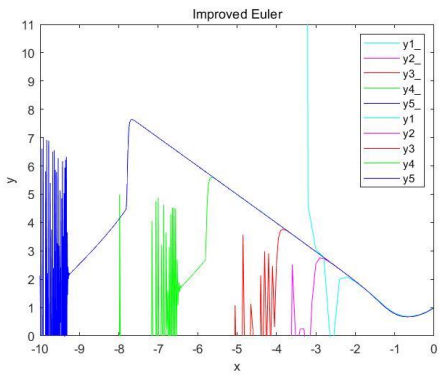
2.3 Result

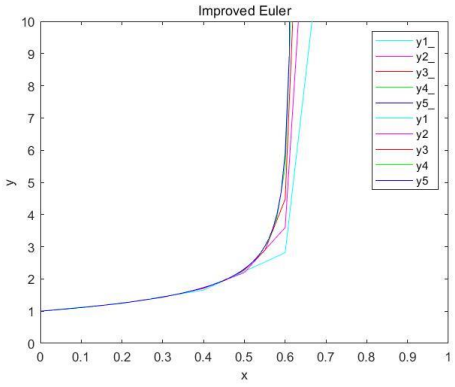
For IVP1 Euler method is applicable, but for IVP2 it fails on the negative part.

IVP1:



IVP2:





2.4 Error analysis

The approximation of integral by area of trapezoid gives the error

$$-\sum_{k=1}^{M} y^{(2)}(c_k)\frac{h^3}{12} \approx \frac{b-a}{12} y^{(2)}(c)h^2$$

Here we can see that the global error is within $O(h^2)$.

3. Runge-Kutta method

3.1 The Main Idea of Runge-Kutta

We know that for the IVP, we can use:

$$y_{n+1}(x) = y_n(x) + \int_{x_n}^{x_{n+1}} f\big(x, y(x)\big)dx$$

By the Mean Value Theorems, there must exist a ξ that $x_n \le \xi \le x_{n+1}$, we can substitute $\int_{x_n}^{x_{n+1}} f\big(x, y(x)\big)dx$ by $f\big(\xi, y(\xi)\big)h$, where h is the step size.

If we choose m points between x_n and x_{n+1} : $x_n \le t_1 \le t_2 \le \cdots \le t_m \le x_{n+1}$

Let $K_i = f\big(t_i, y(t_i)\big)$

We can use these points to calculate the area of $\int_{x_n}^{x_{n+1}} f\big(x, y(x)\big)dx$, here we construct a m-order Runge-Kutta.

Now we get $y_{n+1} = y_n + h\sum_i^m c_i K_i$, where c_i is the weighted combination coefficient.

In order to calculate K_i , we choose $t_1 = x_n$ and $K_1 = f\big(x_n, y(x_n)\big)$, then we can use weighted combination to calculate other K_i.

Let $t_i = t_1 + a_i h = x_n + a_i h$, we get the general form of Runge-Kutta:

$$K_1 = f(x_n + y_n)$$
$$K_2 = f(x_n + a_2 h, y_n + h b_{21} K_1)$$
$$K_3 = f\big(x_n + a_3 h, y_n + h(b_{31}K_1 + b_{32}K_2)\big)$$
$$\cdots$$
$$k_m = f\left(x_n + a_m h, y_n + h\big(b_{m_1}k_1 + b_{m_2}k_2 + \cdots + b_{mm-1}k_{m-1}\big)\right)$$

Where $\sum_{j=1}^{i-1} b_{ij} = a_i$ and $\sum_{i=1}^{m} c_i = 1$.

By increment function of Taylor expansion method, we can determine those a, b, and c.

3.2 Formula of nth-order Runge-Kutta method

The first order Runge-Kutta method is actually Euler method, and the second order Runge-Kutta method is actually improved Euler method.

3.2.1The third order Runge-Kutta method:

(a)

$$K_1 = f(x_n + y_n)$$
$$K_2 = f\left(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hK_1\right)$$
$$K_3 = f\left(x_n + \frac{2}{3}h, y_n + \frac{2}{3}hK_2\right)$$
$$y_{n+1} = y_n + \frac{h}{4}(K_1 + 3K_3)$$

(b)

$$K_1 = f(x_n + y_n)$$
$$K_2 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_1\right)$$
$$K_3 = f(x_n + h, y_n - hK_1 + 2hK_2)$$

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 4K_2 + K_3)$$

3.2.2 The fourth order Runge-Kutta method:

(a)

$$K_1 = f(x_n + y_n)$$

$$K_2 = f\left(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hK_1\right)$$

$$K_3 = f\left(x_n + \frac{2}{3}h, y_n - \frac{1}{3}hK_1 + hK_2\right)$$

$$K_4 = f(x_n + h, y_n + hK_1 - hK_2 + hK_3)$$

$$y_{n+1} = y_n + \frac{h}{8}(K_1 + 3K_2 + 3K_3 + K_4)$$

(b)

$$K_1 = f(x_n + y_n)$$

$$K_2 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_1\right)$$

$$K_3 = f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hK_2\right)$$

$$K_4 = f(x_n + h, y_n + hK_3)$$

$$y_{n+1} = y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

3.3 Use fourth order Runge-Kutta to solve lab IVP problem

3.3.1 Result of IVP1

We use the second one of the fourth order Runge-Kutta to solve (IVP1)

$$y' = y^2 + t^2 + t \quad y(0) = 1$$

(a) Take h = 0.3 from t = 0 to t = 0.9

Step 0 $t_0 = 0,\ y_0 = 1$

Step 1 $t_1 = 0.3\ \ y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 1.498836865170347

Step 2 $t_2 = 0.6\ \ y_2 = y_1 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 3.092616161249836

Step 3 $t_3 = 0.9\ \ y_3 = y_2 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 23.914533197543687

Compare the result with power series solutions:

t_i	Power series solution $y(t_i)$	Numerical solution y_i	Error $\lvert y(t_i) - y_i \rvert$
0	1	1	0
0.3	1.4992	1.498836865170347	0.00036313
0.6	3.1068	3.092616161249836	0.01418384
0.9	99.8573	23.914533197543687	75.9427668

(b) Take h = 0.1 from t = 0 to t = 0.9

Step 0 $t_0 = 0,\ y_0 = 1$

Step 1 $t_1 = 0.1\ \ y_1$= 1.116844289929774

Step 2 $t_2 = 0.2\ \ y_2$= 1.276584681611626

Step 3 $t_3 = 0.3\ \ y_3$= 1.499185425629429

Step 4 t_3 = 0.4 y_4= 1.819293075299801

Step 5 t_3 = 0.5 y_5= 2.304206211779880

Step 6 t_3 = 0.6 y_6= 3.106554937431314

Step 7 t_3 = 0.7 y_7= 4.666647714910317

Step 8 t_3 = 0.8 y_8= 8.990372045980527

Step 9 t_3 = 0.9 y_9= 53.696283140577293

Compare the result with power series solutions:

t_i	Power series solution $y(t_i)$	Numerical solution y_i	Error $\lvert y(t_i) - y_i \rvert$
0	1	1	0
0.1	1.1168	1.116844289929774	4.42899E-05
0.2	1.2766	1.276584681611626	1.53184E-05
0.3	1.4992	1.499185425629429	1.45744E-05
0.4	1.8193	1.819293075299801	6.9247E-06
0.5	2.3043	2.304206211779880	9.37882E-05
0.6	3.1068	3.106554937431314	0.000245063
0.7	4.6688	4.666647714910317	0.002152285
0.8	9.0342	8.990372045980527	0.043827954
0.9	99.8573	53.696283140577293	46.16101686

As we further decreased the step size, we found that the error become smaller and smaller which make the result more accurate.

3.3.2 Result of IVP2

We use the second one of the fourth order Runge-Kutta to solve (IVP2)

$$y' = (y - t)(y^2 - t^2) \quad y(0) = 1$$

(a) Take h = 0.2 from t = 0 to t = 0.9

Step 0 t_0 = 0, y_0 = 1

Step 1 t_1 = 0.2 $y_1 = y_0 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 1.251775109374377

Step 2 t_2 = 0.4 $y_2 = y_1 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 1.731223826996419

Step 3 t_3 = 0.6 $y_3 = y_2 + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4)$= 4.570965760523351

(b) Take h = 0.1 from t = 0 to t = 0.9

Step 0 t_0 = 0, y_0 = 1

Step 1 t_1 = 0.1 y_1= 1.111200906250208

Step 2 t_2 = 0.2 y_2= 1.251915521612879

Step 3 t_3 = 0.3 y_3= 1.442096338626661

Step 4 t_3 = 0.4 y_4= 1.733127294618528

Step 5 t_3 = 0.5 y_5= 2.314794729025541

Step 6 t_3 = 0.6 y_6= 5.426469594182638

3.4 Error analysis

By the relation of order and the highest accuracy given by Butcher in 1965, we get the form as followed:

Order	2	3	4	5	6	7	$n \geq 8$
The highest accuracy	$0(h^2)$	$0(h^3)$	$0(h^4)$	$O(h^4)$	$0(h^5)$	$O(h^6)$	$O(h^{n-2})$

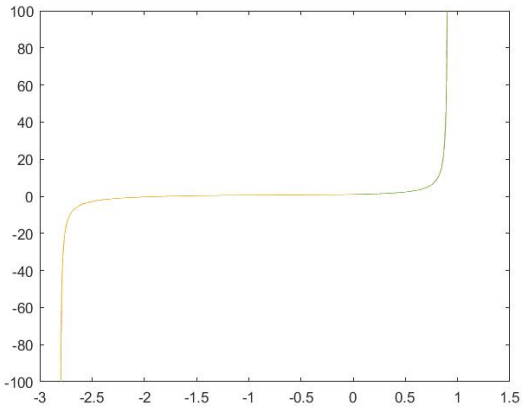
4. Linear multistep method

We have already had three numerical methods, Euler, Improved Euler, Runge-Kutta. However, they are all only use one point before to find out the next point, which means when you want (t_{k+1}, y_{k+1}) , then you only use (t_k, y_k). For this part, we introduce another kind of method, linear multistep method, including Milne-Simpson method, Adams-Bashforth-Moulton method, Hamming method, to solve these problems. Each method has two main formulars and uses 4 previous points to get next one (4^{th}order for example). The first formular is to estimate the y_{k+1} value, and the second one will use this value to get a new y_{k+1} value, which is more precise.

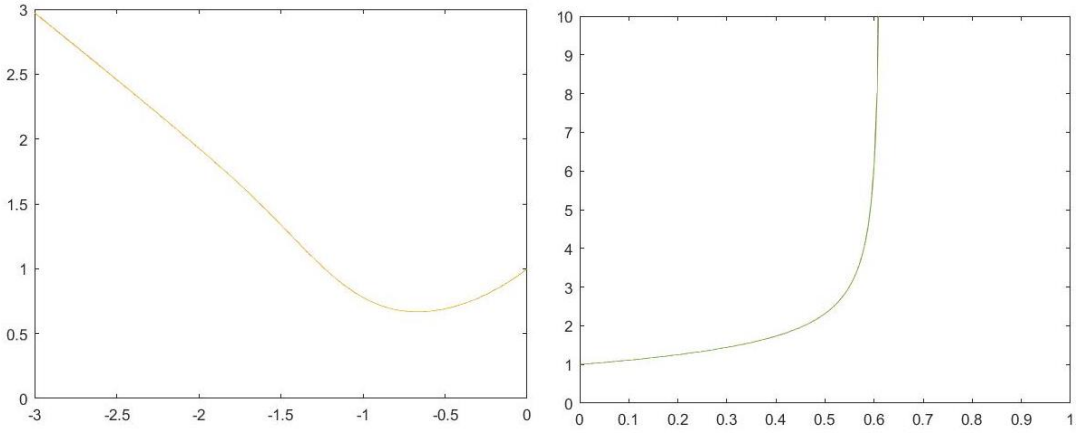
However, in these two problems, we only have one initial point, which means we need at least three more points to use 4^{th} order method. So firstly, we use Runge-Kutta method to get three more points, which are a bit smaller than 0 when we want to find solution for positive part. Then we can apply these three methods. For result, although using different methods, the result is very similar.

(The main code is attached in the appendix)

Result for IVP 1:



Result for IVP 2:



Negative part Positive part

4.1 Adams-Bashforth-Moulton method

The estimation is based on the calculation on $[t_k, t_{k+1}]$. By using Lagrange polynomials, based on (t_{k-3}, f_{k-3}), (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}), (t_k, f_k) we can get $y'_{k+1} = y_k +$

$\int_{t_k}^{t_{k+1}} f(t,y) = y_k + \frac{h}{24}(-9f_{k-3} + 37f_{k-2} - 59f_{k-1} + 55f_k)$. This means we know a new point $\left(t_{k+1}, f(t_{k+1}, y'_{k-1})\right)$, which can be used to find y_{k+1} again, by calculate the integration on $[t_k, t_{k+1}]$, $y_{k+1} = y_k + \int_{t_k}^{t_{k+1}} f(t,y) = y_k + \frac{h}{24}(f_{k-2} - 5f_{k-1} + 19f_k + 9f_{k+1})$. (The code is attached in the appendix)

4.2 Milne-Simpson method

Similar with Adams-Bashforth-Moulton method, The estimation is based on the calculation on $[t_{k-3}, t_{k+1}]$. By using Lagrange polynomials, based on (t_{k-3}, f_{k-3}), (t_{k-2}, f_{k-2}), (t_{k-1}, f_{k-1}), (t_k, f_k) we can get $y'_{k+1} = y_{k-3} + \int_{t_{k-3}}^{t_{k+1}} f(t,y) = y_{k-3} + \frac{4h}{3}(2f_{k-2} - f_{k-1} + 2f_k)$. This means we know a new point $\left(t_{k+1}, f(t_{k+1}, y'_{k-1})\right)$, which can be used to find y_{k+1} again, by calculate the integration on $[t_{k-1}, t_{k+1}]$, $y_{k+1} = y_{k-1} + \int_{t_{k-1}}^{t_{k+1}} f(t,y) = y_{k-1} + \frac{h}{3}(f_{k-1} + 4f_k + f_{k+1})$.

(The code is attached in the appendix)

4.3 Hamming method

Hamming method is also similar and the formular is

$$y'_{k+1} = y_{k-3} + \frac{4h}{3}(2f_{k-2} - f_{k-1} + 2f_k)$$

$$y_{k+1} = \frac{-1}{8}y_{k-2} + \frac{9}{8}y_k + \frac{3h}{8}(-f_{k-1} + 2f_k + f_{k+1})$$

(The code is attached in the appendix)

5. Power series representations of the solutions

5.1 Introduction

An analytic solution (or power series solution) of a scalar ODE is a "power series function"

$$y(t) = \sum_{n=0}^{\infty} a_n (t - t_0)^n, t \in I,$$

for some interval $I \subseteq \mathbb{R}$ of positive length and some $t_0 \in I$.

5.2 Method and result (take the first IVP as an example)

Because $f(t,y) = y^2 + t^2 + t$ is analytic, the solution must be analytic as well. i.e., we can assume $y(t) = \sum_{n=0}^{\infty} a_n t^n$, $y(0) = a_0 = 1$. And by the Cauchy's multiplication formula, we have $y(t)^2 = \sum_{n=0}^{\infty}(\sum_{k=0}^{n} a_k\, a_{n-k})t^n$.

Substitute $y(t), y(t)^2$ into the ODE and equate coefficients.$y'(t) = \sum_{n=1}^{\infty} n a_n t^{n-1} = \sum_{n=0}^{\infty}(n+1)a_{n+1}t^n, y^2 + t^2 + t = t^2 + t + \sum_{n=0}^{\infty}(\sum_{k=0}^{n} a_k\, a_{n-k})t^n$

$$\Longrightarrow (n+1)a_{n+1} = \begin{cases} 1 + \sum_{k=0}^{n} a_k\, a_{n-k}, n = 1,2 \\ \sum_{k=0}^{n} a_k\, a_{n-k}, n \neq 1,2 \end{cases}$$

For $n \geq 1$ this determines a_n from $a_k, k < n$, and thus together with the initial value

$y(0) = a_0 = 1$ provides a recursion formula for a_n, which can easily be programmed:

$$y(t) = 1 + t + \frac{3t^2}{2} + \frac{5t^3}{3} + \frac{19t^4}{12} + 1.75t^5 + 1.94444t^6 + 2.13095t^7 + 2.33482t^8 + \cdots$$

It is clear that all a_n are positive. Using induction, we can easily show that $a_n \geq 1$ for all n. Hence, the radius of convergence $\rho \leq 1$.

For the second IVP, it seems to be a little bit more complicated with the existence of y^3.But similarly, $y(t)^3 = \sum_{n=0}^{\infty}(\sum_{k=0}^{n} b_k\, a_{n-k})t^n$ which $b_n = \sum_{k=0}^{n} a_k\, a_{n-k}$ i.e., $y(t)^2 = \sum_{n=0}^{\infty} b_n t^n$. Then $y'(t) = \sum_{n=0}^{\infty}(n+1)a_{n+1}t^n$, $(y-t)(y^2-t^2) = \sum_{n=0}^{\infty}(\sum_{k=0}^{n} b_k\, a_{n-k})t^n -$ $\sum_{n=1}^{\infty} b_{n-1}t^n - \sum_{n=2}^{\infty} a_{n-2}t^n + t\text{^}3$. Besides, if we assume $b_n = 0$ for $n < 0$,

$$\Longrightarrow (n+1)a_{n+1} = \begin{cases} (\sum_{k=0}^{n} b_k\, a_{n-k}) - b_{n-1} - a_{n-2}, n \neq 3 \\ (\sum_{k=0}^{n} b_k\, a_{n-k}) - b_{n-1} - a_{n-2} + 1, n = 3 \end{cases}$$

Where $b_n = \sum_{k=0}^{n} a_k\, a_{n-k}$ and assume $b_n = 0$ for $n < 0$.

Based on the recursion formulas of a_n, b_n, we can get

$$y(t) = 1 + t + t^2 + t^3 + \frac{7}{4}t^4 + 2.45t^5 + 3.725t^6 + 5.63929t^7 + 4.40179t^8 + \cdots$$

5.3 Radius of convergence

For every power series $y(t) = \sum_{n=0}^{\infty} a_n(t-t_0)^n$, the radius of convergence $\rho = \frac{1}{L}$, where $L = \lim_{n\to\infty} sup\sqrt[n]{a_n}$. And if $\lim_{n\to\infty} \frac{|a_{n+1}|}{|a_n|}$ exists, $L = \lim_{n\to\infty} \frac{|a_{n+1}|}{|a_n|}$.

Use $n = 1000$, for the first IVP, $\rho \approx 0.91$; for the second IVP, $\rho \approx 0.625$

*6. Reduction to simpler ODE by appropriate substitution for the first IVP[1]

6.1 Introduction (Riccati transformation)

Riccati equations: $y' + p(t)y + q(t)y^2 + r(t) = 0$

Use substitution $y(t) = \frac{G(t)}{F(t)}$, then$[G' + p(t)G + r(t)F]F - [F' - q(t)G]G = 0$

Obviously, the choice of F, G is not unique, so we can choose some additional conditions that are convenient to solve. For example, we can make $[F' - q(t)G] = 0$.

Then we need to solve $[G' + p(t)G + r(t)F]F = 0$. Here, $G = \frac{F\prime}{q(t)}$, then $y(t) = \frac{G(t)}{F(t)} = \frac{F\prime}{q(t)F} = \frac{(\ln F)\prime}{q(t)}$, which is called Riccati transformation.

Then by solving $F'' + \left(p - \frac{q'}{q}\right)F' + qrF = 0$, we can get F, and hence y.

6.2 Result

In the first IVP, $p(t) = 0, q(t) = -1, r(t) = -(t+t^2)$. So, we use substitution $y(t) = -\frac{F\prime(t)}{F(t)}$. And $y(0) = -\frac{F'(0)}{F(0)} = 1$, $F(0) + F'(0) = 1$

$\Longrightarrow F'' + (t+t^2)F = 0$. Solve it by power series, assume $F(t) = \sum_{n=0}^{\infty} b_n t^n$, $b_0 = 1$, then we

1. https://zhuanlan.zhihu.com/p/113833745 (Chinese website)

get $b_1 = -1,\ b_2 = 0,\ b_3 = \frac{1}{6},$ and when $n \geq 4, n(n-1)b_n + b_{n-4} + b_{n-3} = 0$

And let $-F'(t) = \sum_{n=0}^{\infty} a_n t^n$, $a_n = -(n+1)b_{n+1}$. Let $y(t) = \sum_{n=0}^{\infty} c_n t^n$
We have $-F'(t) = y(t)F(t)$

$\Longrightarrow a_n = \sum_{k=0}^{n} c_k\, b_{n-k} \Longrightarrow c_n = \frac{a_n - \sum_{k=0}^{n-1} c_k b_{n-k}}{b_0}$, $(c_0 = \frac{a_0}{b_0})$.(From Lecture 30-34 Page 29/108)

Calculate first few terms, we get

	n	0	1	2	3	4	5
$-F'(t)$	a_n	1	0	$\frac{1}{2}$	0	$-\frac{1}{4}$	…
$F(t)$	b_n	1	-1	0	$-\frac{1}{6}$	0	$\frac{1}{20}$
$y(t)$	c_n	1	1	$\frac{3}{2}$	$\frac{5}{3}$	$\frac{19}{12}$	…

Based on the recursion formulas of a_n, b_n, c_n, we can get $y(t) = 1 + t + \frac{3t^2}{2} + \frac{5t^3}{3} + \frac{19t^4}{12} + \cdots$

This result is consistent with the power series solution obtained in 5.2

7. Asymptote analysis:

For the two IVPs

$y' = y^2 + t^2 + t, \quad y(0) = 1$ (IVP1)

$y' = (y-t)(y^2 - t^2) \quad y(0) = 1$ (IVP2)

1. vertical asymptote:

When y is a lot larger than t, they can be viewed as:

$y' = y^2,$ (IVP1)

with solutions $y = \frac{1}{a-t}$, with a to be determined by point (t, y).

According to our plot, IVP1 has two vertical asymptotes, positive and negative.

IVP 2 has only one positive vertical asymptote.

For IVP1 we have two precise points (0.9100, 7685.915313082049579) and (-2.810, -12055038) computed by RK-4, resulting the asymptote $x = 0.910103$ and $x = -2.8100001$.
For IVP2, we have a precise point (0.6142, 561.885927476), resulting the asymptote $x = 0.61597972$

2. asymptote for IVP2's negative part

From the plot we guess it's y = -t.

$$y' = (y-t)(y^2 - t^2) \Leftrightarrow \frac{y'}{t^3} = \frac{y^3}{t^3} - \frac{y^2}{t^2} - \frac{y}{t} + 1$$

For $t \to -\infty, \frac{y'}{t^3} = 0$, therefore if we assume $\frac{y}{t} = \lambda$, then

$0 = \lambda^3 - \lambda^2 - \lambda + 1 = (\lambda + 1)(\lambda - 1)^2$, from the plot $\lambda = -1$.

Therefore, the asymptote for the IVP2's negative part is $y = -t$.

Appendix

Code for Euler, improved Euler, RK-2, RK-3, RK-4, with step size 0.05

```
clear
clc
close all

%% initialize positive
syms x;
syms y;
syms f;         %
f(x,y) = y^2+x^2+x; % y'=f(x,y)
x0 = 0;
y0 = 1;           % y(0)=1

h = 0.05;          % step
x = x0:h:1;       % x
len = length(x);

%% initialize negative
syms x_;
syms y_;
syms f;
f(x_,y_) = y_^2+x_^2+x_; % y'=f(x,y)
x0_ = 0;
y0_ = 1;           % y(0)=1

h_ = -0.05;          % step
x_ = x0_:h_:(-3);       % x
len_ = length(x_);

%% Euler positive
y1 = zeros(size(x)); % y
y1(1) = y0;
for ii = 2:len
    K1 = f(x(ii-1),y1(ii-1));
    y1(ii) = y1(ii-1) + h*K1;
end

%% Euler negative
y1_ = zeros(size(x_)); % y
y1_(1) = y0_;
for ii = 2:len_
    K1 = f(x_(ii-1),y1_(ii-1));
```

```
    y1_(ii) = y1_(ii-1) + h_*K1;
end


%% improved Eular positive
y1_improved = zeros(size(x)); % y
y1_improved(1) = y0;
for ii = 2:len
    K1 = f(x(ii-1),y1_improved(ii-1));
    y1_improved(ii) = y1_improved(ii-1) + h*K1;
    K2 = f(x(ii),y1_improved(ii));
    y1_improved(ii) = y1_improved(ii-1) + h/2*(K1 + K2);
end


%% improved Eular negative
y1_improved_ = zeros(size(x_)); % y
y1_improved_(1) = y0_;
for ii = 2:len_
    K1 = f(x_(ii-1),y1_improved_(ii-1));
    y1_improved_(ii) = y1_improved_(ii-1) + h_*K1;
    K2 = f(x_(ii),y1_improved_(ii));
    y1_improved_(ii) = y1_improved_(ii-1) + h_/2*(K1 + K2);
end

%% 2K positive
y2 = zeros(size(x)); % y
y2(1) = y0;
for ii = 2:len
    K1 = f(x(ii-1),y2(ii-1));
    K2 = f(x(ii-1)+h/2,y2(ii-1)+h*K1/2);
    y2(ii) = y2(ii-1) + h*K2;
end

%% 2K negative
y2_ = zeros(size(x_)); % y
y2_(1) = y0_;
for ii = 2:len_
    K1 = f(x_(ii-1),y2_(ii-1));
    K2 = f(x_(ii-1)+h_/2,y2_(ii-1)+h_*K1/2);
    y2_(ii) = y2_(ii-1) + h_*K2;
end

%% 3K positive
```

```
y3 = zeros(size(x));
y3(1) = y0;
for ii = 2:len
    K1 = f(x(ii-1),y3(ii-1));
    K2 = f(x(ii-1)+h/2,y3(ii-1)+h*K1/2);
    K3 = f(x(ii-1)+h,y3(ii-1)+h*(K2*2-K1));
    y3(ii) = y3(ii-1) + h*(K1+4*K2+K3)/6;
end


%% 3K negative
y3_ = zeros(size(x_));
y3_(1) = y0_;
for ii = 2:len_
    K1 = f(x_(ii-1),y3_(ii-1));
    K2 = f(x_(ii-1)+h_/2,y3_(ii-1)+h_*K1/2);
    K3 = f(x_(ii-1)+h_,y3_(ii-1)+h_*(K2*2-K1));
    y3_(ii) = y3_(ii-1) + h_*(K1+4*K2+K3)/6;
end

%% 4K positive
y4 = zeros(size(x));
y4(1) = y0;
for ii = 2:len
    K1 = f(x(ii-1),y4(ii-1));
    K2 = f(x(ii-1)+h/2,y4(ii-1)+h*K1/2);
    K3 = f(x(ii-1)+h/2,y4(ii-1)+h*K2/2);
    K4 = f(x(ii-1)+h,y4(ii-1)+h*K3);
    y4(ii) = y3(ii-1) + h*(K1+2*K2+2*K3+K4)/6;
end

%% 4K negative
y4_ = zeros(size(x_));
y4_(1) = y0_;
for ii = 2:len_
    K1 = f(x_(ii-1),y4_(ii-1));
    K2 = f(x_(ii-1)+h_/2,y4_(ii-1)+h_*K1/2);
    K3 = f(x_(ii-1)+h_/2,y4_(ii-1)+h_*K2/2);
    K4 = f(x_(ii-1)+h_,y4_(ii-1)+h_*K3);
    y4_(ii) = y3_(ii-1) + h_*(K1+2*K2+2*K3+K4)/6;
end

%% plot
figure
```

```
plot(x_,y2_,'--r',x_,y3_,'--g',x_,y4_,'--b',x_,y1_,'--c',x_,y1_improved_,'--m')
hold on;
plot(x,y2,'--r',x,y3,'--g',x,y4,'--b',x,y1,'--c',x,y1_improved,'--m')
axis([0 1 -1 1]);
title('solution curve');
xlabel('x');
ylabel('y');
legend('2K','3K','4K','Euler','improved Euler')
```

Code for ABM method

```
function B = abmp(f,T, Y)
n=length (T);
if n<5
   return
end
F=feval(f,T (1: 4),Y(1:4) );
h=(T (2)-T (1));
for k=4 :n-1
   p=Y(k)+h/24*(F*[-9 37 -59 55]');
   T(k+1)=T(1)+h*k ;
   F=[F(2) F(3) F(4) feval(f,T(k+1),p)];
   Y(k+1)=Y(k)+h/24*(F*[1 -5 19 9]') ;
   F(4) =feval (f,T(k+1),Y(k+1) ) ;
end
B=[T' Y'];
end
```

Code for Milne method

```
function B=milnep(f,T,Y)

n=length (T);
if n<5
   return
end
F=zeros(1,4);
F=feval(f,T (1: 4),Y(1:4) );
h=(T (2)-T (1));
pold=0;
yold=0;
for k=4 :n-1
```

```
    p=Y(k-3)+(4*h/3)*(F*[0 2 -1 2]');
    pmod=p+28*(yold-pold)/29;
    T(k+1)=T(1)+h*k ;
    F=[F(2) F(3) F(4) feval(f,T(k+1),pmod)];
    Y(k+1)=Y(k-1)+h/3*(F*[0 1 4 1]') ;
    pold=p;
    yold=Y(k+1);
    F(4) =feval (f,T(k+1),Y(k+1) ) ;
end
B=[T' Y'];
end
```

Code for Hamming method

```
function B=hammingp(f,T,Y)

n=length (T);
if n<5
    return
end
F=zeros(1,4);
F=feval(f,T (1: 4),Y(1:4) );
h=(T (2)-T (1));
pold=0;
cold=0;
for k=4 :n-1
    p=Y(k-3)+(4*h/3)*(F*[0 2 -1 2]');
    pmod=p+112*(cold-pold)/121;
    T(k+1)=T(1)+h*k ;
    F=[F(2) F(3) F(4) feval(f,T(k+1),pmod)];
    c=(-Y(k-2)+9*Y(k))/8+(3*h/8)*(F*[0 -1 2 1]') ;
    Y(k+1)=c+9*(p-c)/121;
    pold=p;
    cold=c;

    F(4) =feval (f,T(k+1),Y(k+1) ) ;
end
B=[T' Y'];
End
```

Reference

[1] John H, Kurtis D. Numerical Methods Using MATLAB. Fourth Edition.
[2] 数学天才琪露诺.常见的可求解析解的微分方程. https://zhuanlan.zhihu.com/p/113833745 (Chinese website)