Numerical Treatment of Differential Equations: HW 1

Truman Ellis

January 28, 2011

Problem 1.1

Apply the method of proof of Theorems 1.1 and 1.2 to prove the convergence of the implicit midpoint rule and of the theta method.

Solution

Implicit Midpoint Rule We first need to calculate the local truncation error from

$$y(t_{n+1}) - y(t_n) - hf\left(t_n + \frac{h}{2}, y\left(t_n + \frac{h}{2}\right)\right). \tag{1}$$

Substituting $f(t_n + \frac{h}{2}, y(t_n + \frac{h}{2})) = y'(t_n + \frac{h}{2})$, and Taylor expanding

$$y'\left(t_n + \frac{h}{2}\right) = y'(t_n) + \frac{h}{2}h''(t_n) + O(h^2).$$

Now, Taylor expanding

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{1}{2}h^2y''(t_n) + O(h^3).$$

We can substitute these into Equation 1:

$$\begin{aligned}
&\cancel{y(t_n)} + h\cancel{y'(t_n)} + \frac{1}{2}h^2\cancel{y''(t_n)} + O(h^3) \\
&- \cancel{y(t_n)} \\
&- h(\cancel{y'(t_n)} + \frac{h}{2}h\cancel{y''(t_n)} + O(h^2)) && = O(h^3).
\end{aligned}$$

Now we can calculate the global error. Let us first Taylor expand $y(t_{n+1})$:

$$\begin{aligned}
y(t_{n+1}) &= y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(t_n) + \frac{h^3}{6}y'''(t_n) + \cdots \\
&= y(t_n) + h\underbrace{[y'(t_n) + \frac{h}{2}y''(t_n) + \frac{h^2}{6}y'''(t_n) + \cdots]}_{y'\left(t_n + \frac{h}{2}\right)} \\
&= y(t_n) + h\left[y'\left(t_n + \frac{h}{2}\right)\right].
\end{aligned}$$

Now we can subtract

$$\begin{aligned}
e_{n+1} &= y_{n+1} - y(t_{n+1}) \\
&= y_n - y(t_n) + h\left[f\left(t_{n+\frac{1}{2}}, \frac{1}{2}(y_n + y_{n+1})\right)\right] - h\left[y'\left(t_n + \frac{h}{2}\right)\right] + O(h^3) \\
&= e_n + h\left[f\left(t_n + \frac{h}{2}, \frac{1}{2}(y_n + y_{n+1})\right) - f\left(t_n + \frac{h}{2}, y(t_n + \frac{h}{2})\right)\right] + O(h^3).
\end{aligned}$$

Since f is Lipshitz,

$$\begin{aligned}
\|e_{n+1}\| &\leq \|e_n\| + h\left\|f\left(t_n + \frac{h}{2}, \frac{1}{2}(y_n + y_{n+1})\right) - f\left(t_n + \frac{h}{2}, y(t_n + \frac{h}{2})\right)\right\| + Ch^3 \\
&\leq \|e_n\| + h\lambda\left\|\frac{1}{2}(y_n + y_{n+1}) - y\left(t_n + \frac{h}{2}\right)\right\| + Ch^3 \\
&\leq \|e_n\| + h\lambda\left\|\frac{1}{2}(y_n + y_{n+1}) - \frac{1}{2}\left(y(t_n) + y(t_{n+1})\right)\right\| + Ch^3 \\
&\leq (1 + \frac{1}{2}\lambda h)\|e_n\| + \frac{1}{2}\lambda h\|e_{n+1}\| + Ch^3 \\
&\leq \frac{1 + \frac{1}{2}\lambda h}{1 - \frac{1}{2}\lambda h}\|e_n\| + \frac{C}{1 - \frac{1}{2}\lambda h}h^3
\end{aligned}$$

We can now claim that

$$\|e_n\| \leq \frac{C}{\lambda}\left[\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^n - 1\right]h^2\,.$$

Indeed for $n = 0$, $\|e_n\| = 0$ and

$$\begin{aligned}
\|e_{n+1}\| &\leq \left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)\|e_n\| + \frac{C}{1 - \frac{1}{2}\lambda h}h^3 \\
&= \frac{C}{\lambda}h^2\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^{n+1} - \frac{C}{\lambda}\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)h^2 + \frac{C}{\lambda}\frac{h^3\lambda}{1 - \frac{1}{2}h\lambda} \\
&= \frac{C}{\lambda}h^2\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^{n+1} + \frac{C}{\lambda}\left(-\frac{h^2 - \frac{1}{2}h^3\lambda + h^3\lambda}{1 - \frac{1}{2}h\lambda}\right) \\
&= \frac{C}{\lambda}h^2\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^{n+1} + \frac{C}{\lambda}h^2 \\
&= \frac{C}{\lambda}\left[\left(\frac{1 + \frac{1}{2}h\lambda}{1 - \frac{1}{2}h\lambda}\right)^{n+1} - 1\right]h^2\,.
\end{aligned}$$

Therefore the proof holds by induction, the implicit midpoint rule is order 2.

Theta Method We first need to calculate the local truncation error:

$$\begin{aligned}
&y(t_{n+1}) - [y(t_n) + h(\theta f(t_n, y(t_n)) + (1 - \theta)f(t_{n+1}, y(t_{n+1})))] \\
&= \left[\cancel{y(t_n)} + h\cancel{y'(t_n)} + \frac{1}{2}h^2y''(t_n) + O(h^3)\right] \\
&- \left[\cancel{y(t_n)} + h\theta\cancel{y'(t_n)} + (1 - \theta)h(\cancel{y'(t_n)} + hy''(t_n) + \frac{h^2}{2}y'''(t_n) + O(h^3))\right] \\
&= \left(\theta - \frac{1}{2}\right)h^2y''(tn) + \left(\frac{\theta}{2} - \frac{1}{3}\right)h^3y'''(t_n) + O(h^4)\,.
\end{aligned}$$

Therefore the local truncation is order h^2 for $\theta = \frac{1}{2}$ and order h otherwise.

Now for $\theta = \frac{1}{2}$, subtracting

$$\begin{aligned}
e_{n+1} &= y_{n+1} - y(t_{n+1}) \\
&= y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})) \\
&- y(t_n) - \frac{h}{2}(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))) + O(h^3) \\
&= e_n + \frac{h}{2}[f(t_n, y_n) - f(t_n, y(t_n))] \\
&+ \frac{h}{2}[f(t_{n+1}, y_{n+1}) - f(t_{n+1}, y(t_{n+1}))] + O(h^3)\,.
\end{aligned}$$

Since f is Lipshitz

$$\begin{aligned}\|e_{n+1}\| &\le \|e_n\| + \frac{1}{2}\lambda h[\|y_n - y(t_n)\| + \|y_{n+1} - y(t_{n+1})\|] + Ch^3 \\ &\le \frac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h}\|e_n\| + \frac{C}{1-\theta\lambda h}h^3 \\ &\le \frac{C}{\lambda}\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n - 1\right]h^2\,.\end{aligned}$$

This indeed holds for $n = 0$ and by induction for $n + 1$:

$$\begin{aligned}\|e_{n+1}\| &\le \left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)\frac{C}{\lambda}h^2\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n - 1\right] + \left(\frac{C}{1-\frac{1}{2}h\lambda}\right)h^3 \\ &= \frac{C}{\lambda}\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^{n+1} - 1\right]h^2\,.\end{aligned}$$

Therefore the theta method is order h^2 for $\theta = \frac{1}{2}$.

Now for $\theta \neq \frac{1}{2}$:

$$\begin{aligned}e_{n+1} &= y_{n+1} - y(t_{n+1}) \\ &= y_n + h\theta f(t_n, y_n) + h(1-\theta)f(t_{n+1}, y_{n+1}) - y(t_n) - hf(t_n, y(t_n)) + O(h^2) \\ &= y_n + h\theta f(t_n, y_n) - h(1-\theta)[f(t_n, y_n) + O(h)] - y(t_n) - hf(t_n, y(t_n)) + O(h^2) \\ &= e_n + h[f(t_n, y_n) - f(t_n, y(t_n))] + O(h^2)\,.\end{aligned}$$

Since f is Lipshitz

$$\begin{aligned}\|e_{n+1}\| &\le (1+h\lambda)\|e_n\| + Ch^2 \\ &\le \frac{C}{\lambda}\left[(1+h\lambda)^n - 1\right]h\,.\end{aligned}$$

This indeed holds for $n = 0$ and by induction for $n + 1$:

$$\|e_{n+1}\| \le (1+h\lambda)\frac{C}{\lambda}h[(1+h\lambda)^n - 1] + Ch^2 = \frac{C}{\lambda}h[(1+h\lambda)^{n+1} - 1]$$

Therefore the theta method is order h for $\theta \neq \frac{1}{2}$.

Problem 1.3

We solve the scalar linear system

$$y' = ay\,, \quad y(0) = 1\,.$$

a) Show that the 'continuous output' method

$$u(t) = \frac{1+\frac{1}{2}a(t-nh)}{1-\frac{1}{2}a(t-nh)}y_n\,, \quad nh \le t \le (n+1)h\,, \quad n = 0, 1, ...,$$

is consistent with the values of y_n and y_{n+1} which are obtained by the trapezoid rule.

Solution

According to the trapezoid rule

$$y_{n+1} = y_n + \frac{1}{2}h[ay_n + ay_{n+1}]\,.$$

Rearranging,

$$\left(1-\frac{1}{2}ha\right)y_{n+1}=\left(1+\frac{1}{2}ha\right)y_n$$
$$y_{n+1}=\frac{1+\frac{1}{2}ha}{1-\frac{1}{2}ha}y_n\,.$$

Evaluating

$$u(nh)=\frac{1+\frac{1}{2}a(0)}{1-\frac{1}{2}a(0)}y_n=y_n\,,$$

this is consistent with the trapezoid rule. Again, evaluating

$$u((n+1)h)=\frac{1+\frac{1}{2}a(h)}{1-\frac{1}{2}a(h)}y_n\,,$$

this is also consistent with the results from the trapezoid rule.

b) Demonstrate that u obeys the perturbed ODE

$$u'(t)=au(t)+\frac{\frac{1}{4}a^3(t-nh)^2}{\left[1-\frac{1}{2}a(t-nh)\right]^2}y_n\quad t\in[nh,(n+1)h]$$

with initial condition $u(nh)=y_n$. Thus, prove that

$$u((n+1)h)=\mathrm{e}^{ha}\left[1+\frac{1}{4}a^3\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2 d\tau\,\mathrm{d}\tau}{\left(1-\frac{1}{2}a\tau\right)^2}\right]y_n\,.$$

Solution

First we need to calculate

$$\begin{aligned}u'(t)&=-\frac{1+\frac{1}{2}a(t-nh)}{\left(1-\frac{1}{2}a(t-nh)\right)^2}\left(-\frac{1}{2}a\right)+\frac{\frac{1}{2}a}{1-\frac{1}{2}a(t-nh)}\\&=\frac{a}{\left(1-\frac{1}{2}a(t-nh)\right)^2}y_n\,.\end{aligned}$$

Now, combining terms in the right hand side,

$$\begin{aligned}u'(t)&=a\left[\frac{1+\frac{1}{2}a(t-nh)}{1-\frac{1}{2}a(t-nh)}y_n\right]+\frac{\frac{1}{4}a^3(t-nh)^2}{\left[1-\frac{1}{2}a(t-nh)\right]^2}y_n\quad t\in[nh,(n+1)h]\\&=\frac{\left(a-\cancel{\frac{1}{4}a^3(t-nh)^2}\right)+\cancel{\frac{1}{4}a^3(t-nh)^2}}{\left(1-\frac{1}{2}a(t-nh)\right)^2}y_n\\&=\frac{a}{\left(1-\frac{1}{2}a(t-nh)\right)^2}y_n\,.\end{aligned}$$

Therefore $u(t)$ does satisfy the perturbed ODE.

We now wish to find a solution via integrating factors. Let us rewrite

$$u'(t)\underbrace{-a}_{P(t)}u(t)=\underbrace{\frac{\frac{1}{4}a^3(t-nh)^2}{\left[1-\frac{1}{2}a(t-nh)\right]^2}y_n}_{Q(t)}\,.$$

Then,

$$M(t)=\mathrm{e}^{\int P(t)\,\mathrm{d}t}=\mathrm{e}^{-at}\,.$$

Now multiply both sides by $M(t)$, and using the product rule:

$$\mathrm{e}^{-at}u'(t) - a\mathrm{e}^{-at}u(t) = \mathrm{e}^{-at}Q(t)$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathrm{e}^{-at}u(t)\right) = \mathrm{e}^{-at}Q(t)\,.$$

Now integrate both sides from nh to $(n+1)h$:

$$\mathrm{e}^{-a(n+1)h}u((n+1)h) - \mathrm{e}^{-anh}u(nh) = \int_{nh}^{(n+1)h} \mathrm{e}^{-at}\frac{\frac{1}{4}a^3(t-nh)^2}{\left[1-\frac{1}{2}a(t-nh)\right]^2}y_n\,\mathrm{d}t\,.$$

Simplifying,

$$u((n+1)h) = \frac{1}{4}a^3\mathrm{e}^{a(n+1)h}\int_{nh}^{(n+1)h}\frac{\mathrm{e}^{-at}(t-nh)^2}{\left[1-\frac{1}{2}a(t-nh)\right]^2}y_n\,\mathrm{d}t + \mathrm{e}^{ah}\underbrace{u(nh)}_{y_n}\,.$$

Performing a change of variables to $\tau = t - nh$,

$$\begin{aligned}u((n+1h) &= \frac{1}{4}a^3\mathrm{e}^{a(n+1)h}\int_0^h\frac{\mathrm{e}^{-anh}\mathrm{e}^{-a\tau}\tau^2}{\left[1-\frac{1}{2}a\tau\right]^2}y_n\,\mathrm{d}\tau + \mathrm{e}^{ah}y_n\\ &= \frac{1}{4}a^3\mathrm{e}^{ah}\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2}{\left[1-\frac{1}{2}a\tau\right]^2}y_n\,\mathrm{d}\tau + \mathrm{e}^{ah}y_n\\ &= \mathrm{e}^{ah}\left[1+\frac{1}{4}a^3\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2}{\left[1-\frac{1}{2}a\tau\right]^2}y_n\,\mathrm{d}\tau\right]y_n\,.\end{aligned}$$

c) Let $e_n = y_n - y(nh)\,,\quad n = 0, 1, ...,$ Show that

$$e_{n+1} = \mathrm{e}^{ha}\left[1+\frac{1}{4}a^3\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau}{(1-\frac{1}{2}a\tau)^2}\right]e_n + \frac{1}{4}a^3\mathrm{e}^{(n+1)ha}\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau}{(1-\frac{1}{2}a\tau)^2}\,.$$

In particular, deduce that $a < 0$ implies that the error propagates subject to the inequality

$$|e_{n+1}| \le \mathrm{e}^{ah}\left(1+\frac{1}{4}\left|a\right|^3\int_0^h\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau\right)|e_n| + \frac{1}{4}\left|a\right|^3\mathrm{e}^{(n+1)ha}\int_0^h\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau\,.$$

Solution

We can trivially solve the original ODE for the exact solution:

$$y(t) = \mathrm{e}^{at}\,.$$

Then,

$$\begin{aligned}e_{n+1} &= y_{n+1} - y((n+1)h) = u((n+1)h) - \mathrm{e}^{a(n+1)h}\\ &= \mathrm{e}^{ah}\left(\frac{1}{4}a^3\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau}{(1-\frac{1}{2}a\tau)^2} \underbrace{\qquad\qquad}_{=e_n+y(nh)} + y_n - \underbrace{\mathrm{e}^{anh}}_{=y(nh)}\underbrace{\qquad\qquad}_{=e_n}\right)\\ &= \mathrm{e}^{ha}\left(\frac{1}{4}a^3\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau}{(1-\frac{1}{2}a\tau)^2}+1\right)e_n + \frac{1}{4}a^3\mathrm{e}^{(n+1)ha}\int_0^h\frac{\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau}{(1-\frac{1}{2}a\tau)^2}\,.\end{aligned}$$

In particular, when $a < 0$, the denominator of the integrals, $(1-\frac{1}{2}a\tau)^2 > 1$ and we can bound the error by letting the denominator be unity (thereby increasing the overall integral value). By replacing the integral denominators in the previous equation, we thus bound the error:

$$|e_{n+1}| \le \mathrm{e}^{ah}\left(1+\frac{1}{4}\left|a\right|^3\int_0^h\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau\right)|e_n| + \frac{1}{4}\left|a\right|^3\mathrm{e}^{(n+1)ha}\int_0^h\mathrm{e}^{-a\tau}\tau^2\,\mathrm{d}\tau\,.$$

Problem 1.5

Provided that $\mathbf{f}$ is analytic, it is possible to obtain from $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ and expression for the second derivative of $\mathbf{y}$, namely $\mathbf{y}'' = \mathbf{g}(t, \mathbf{y})$ where

$$\mathbf{g}(t, \mathbf{y}) = \frac{\partial \mathbf{f}(t, \mathbf{y})}{\partial t} + \frac{\partial \mathbf{f}(t, \mathbf{y})}{\partial \mathbf{y}} \mathbf{f}(t, \mathbf{y}) \,.$$

Find the orders of the methods

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n) + \frac{1}{2} h^2 \mathbf{g}(t_n, \mathbf{y}_n)$$

and

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{1}{2} h \left[\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n+1}, \mathbf{y}_{n+1}) \right] + \frac{1}{12} h^2 \left[\mathbf{g}(t_n, \mathbf{y}_n) - \mathbf{g}(t_{n+1}, y_{n+1}) \right] \,.$$

Solution

a) First, Taylor expand the exact solution

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2} \mathbf{y}''(t_n) + \frac{h^3}{6} \mathbf{y}'''(\xi) \,.$$

Now, substituting $\mathbf{y}''(t_n)$ for $\mathbf{g}(t_n, \mathbf{y}(t_n)$, compute

$$\begin{aligned} \mathbf{y}_{n+1} - \mathbf{y}(t_{n+1}) &= \cancel{\mathbf{y}(t_n)} + h\cancel{\mathbf{y}'(t_n)} + \frac{h^2}{2} \cancel{\mathbf{y}''(t_n)} \\ &\quad - \left[\cancel{\mathbf{y}(t_n)} + h\cancel{\mathbf{y}'(t_n)} + \frac{h^2}{2} \cancel{\mathbf{y}''(t_n)} + \frac{h^3}{6} \mathbf{y}'''(\xi) \right] \\ &= O(h^3) \,. \end{aligned}$$

Therefore the local truncation error is order h^2.

The global error can be found from:

$$\begin{aligned} e_{n+1} &= \mathbf{y}_{n+1} - \mathbf{y}(t_{n+1}) \\ &= \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n) + \frac{h^2}{2} \mathbf{g}(t_n, y_n) - \mathbf{y}(t_n) + h\mathbf{y}'(t_n) - \frac{h^2}{2} \mathbf{y}''(t_n) + O(h^3) \\ &= e_n + h[\mathbf{f}(t_n, y(t_n) + e_n) - \mathbf{f}(t_n, \mathbf{y}(t_n))] + \frac{h^2}{2} [\mathbf{g}(t_n, \mathbf{y}(t_n) + e_n) - \mathbf{g}(t_n, \mathbf{y}(t_n))] + O(h^3) \end{aligned}$$

From the triangle inequality and the Lipshitz condition and assuming $h < 1$, for appropriate λ:

$$\begin{aligned} \|e_{n+1}\| &\leq \|e_n\| + h \left\| \mathbf{f}(t_n, y(t_n) + e_n) - \mathbf{f}(t_n, \mathbf{y}(t_n)) \right\| + \frac{h^2}{2} \left\| \mathbf{g}(t_n, \mathbf{y}(t_n) + e_n) - \mathbf{g}(t_n, \mathbf{y}(t_n)) \right\| + Ch^3 \\ &\leq (1 + \underbrace{h\lambda_f + \frac{h^2}{2} \lambda_g}_{\leq h\lambda}) \|e_n\| + Ch^3 \\ &\leq \frac{C}{\lambda} h^2 [(1 + h\lambda)^n - 1] \,. \end{aligned}$$

This indeed holds for $n = 0$ and by induction for $n + 1$:

$$\|e_{n+1}\| \leq (1 + h\lambda) \frac{C}{\lambda} h^2 [(1 + h\lambda)^n - 1] + Ch^3 = \frac{C}{\lambda} h^2 [(1 + h\lambda)^{n+1} - 1]$$

Therefore the method is order h^2.

b) First we need to compute a few Taylor expansions:

$$\begin{aligned}
\mathbf{y}(t_{n+1}) &= \mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2}\mathbf{y}''(t_n) + \frac{h^3}{6}\mathbf{y}'''(t_n) + \frac{h^4}{24}\mathbf{y}^{(4)}(t_n) + \frac{h^4}{120}\mathbf{y}^{(5)}(t_n) + O(h^6)\,,\\
\mathbf{y}'(t_{n+1}) &= \mathbf{y}'(t_n) + h\mathbf{y}''(t_n) + \frac{h^2}{2}\mathbf{y}'''(t_n) + \frac{h^3}{6}\mathbf{y}^{4)}(t_n) + \frac{h^5}{24}\mathbf{y}^{(5)}(t_n) + O(h^5)\,,\\
\mathbf{y}''(t_{n+1}) &= \mathbf{y}''(t_n) + h\mathbf{y}'''(t_n) + \frac{h^2}{2}\mathbf{y}^{(4)}(t_n) + \frac{h^3}{6}\mathbf{y}^{5)}(t_n) + O(h^4)\,.
\end{aligned}$$

Now compute

$$\begin{aligned}
\mathbf{y}_{n+1} - \mathbf{y}(t_{n+1}) &= \mathbf{y}_n + \frac{1}{2}h\left[\mathbf{f}(t_n,\mathbf{y}_n) + \mathbf{f}(t_{n+1},\mathbf{y}_{n+1})\right] + \frac{1}{12}h^2\left[\mathbf{g}(t_n,\mathbf{y}_n) - \mathbf{g}(t_{n+1},\mathbf{y}_{n+1})\right]\\
&\quad - \left(\mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2}\mathbf{y}''(t_n) + \frac{h^3}{6}\mathbf{y}'''(t_n) + \frac{h^4}{24}\mathbf{y}^{(4)}(t_n) + \frac{h^4}{120}\mathbf{y}^{(5)}(t_n) + O(h^6)\right)\\
&= \mathbf{y}_n + \frac{1}{2}h\left[\mathbf{y}'(t_n) + \mathbf{y}'(t_{n+1})\right]\\
&\quad + \frac{1}{12}h^2\left[\mathbf{y}''(t_n) - \mathbf{y}''(t_{n+1})\right]\\
&\quad - \left(\mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2}\mathbf{y}''(t_n) + \frac{h^3}{6}\mathbf{y}'''(t_n) + \frac{h^4}{24}\mathbf{y}^{(4)}(t_n) + \frac{h^4}{120}\mathbf{y}^{(5)}(t_n) + O(h^6)\right)\\
&= \mathbf{y}_n + \frac{1}{2}h\left[\mathbf{y}'(t_n) + \mathbf{y}'(t_n) + h\mathbf{y}''(t_n) + \frac{h^2}{2}\mathbf{y}'''(t_n) + \frac{h^3}{6}\mathbf{y}^{4)}(t_n) + \frac{h^5}{24}\mathbf{y}^{(5)}(t_n) + O(h^5)\right]\\
&\quad + \frac{1}{12}h^2\left[\mathbf{y}''(t_n) - \mathbf{y}''(t_n) - h\mathbf{y}'''(t_n) - \frac{h^2}{2}\mathbf{y}^{(4)}(t_n) - \frac{h^3}{6}\mathbf{y}^{5)}(t_n) - O(h^4)\right]\\
&\quad - \left(\mathbf{y}(t_n) + h\mathbf{y}'(t_n) + \frac{h^2}{2}\mathbf{y}''(t_n) + \frac{h^3}{6}\mathbf{y}'''(t_n) + \frac{h^4}{24}\mathbf{y}^{(4)}(t_n) + \frac{h^4}{120}\mathbf{y}^{(5)}(t_n) + O(h^6)\right)\\
&= (1-1)\,\mathbf{y}(t_n) + \left(\frac{1}{2} + \frac{1}{2} - 1\right)h\mathbf{y}'(t_n) + \left(\frac{1}{2} + \frac{1}{12} - \frac{1}{12} - \frac{1}{12}\right)\frac{h^2}{2}\mathbf{y}''(t_n)\\
&\quad + \left(\frac{1}{4} - \frac{1}{12} - \frac{1}{6}\right)\frac{h^3}{6}\mathbf{y}'''(t_n) + \left(\frac{1}{12} - \frac{1}{24} - \frac{1}{24}\right)\frac{h^4}{24}\mathbf{y}^{(4)}(t_n)\\
&\quad + \left(\frac{1}{48} - \frac{1}{72} - \frac{1}{120}\right)\frac{h^4}{120}\mathbf{y}^{(5)}(t_n) + O(h^6)\\
&= O(h^5)\,.
\end{aligned}$$

Therefore the local truncation error is order h^4.

Subtracting

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \frac{h}{2}[\mathbf{f}(t_n,\mathbf{y}(t_n)) + \mathbf{f}(t_{n+1},\mathbf{y}(t_{n+1}))] + \frac{h^2}{12}[\mathbf{g}(t_n,\mathbf{y}(t_n)) + \mathbf{g}(t_{n+1},\mathbf{y}(t_{n+1}))] + O(h^5)$$

from y_{n+1}, we obtain

$$\begin{aligned}
e_{n+1} &= e_n + \frac{h}{2}\left\{[\mathbf{f}(t_n,\mathbf{y}_n) - \mathbf{f}(t_n,\mathbf{y}(t_n))] + [\mathbf{f}(t_{n+1},\mathbf{y}_{n+1}) - \mathbf{f}(t_{n+1},\mathbf{y}(t_{n+1}))]\right\}\\
&\quad + \frac{h^2}{12}\left\{[\mathbf{g}(t_n,\mathbf{y}_n) - \mathbf{g}(t_n,\mathbf{y}(t_n))] + [\mathbf{g}(t_{n+1},\mathbf{y}_{n+1}) - \mathbf{g}(t_{n+1},\mathbf{y}(t_{n+1}))]\right\} + O(h^5)\,.
\end{aligned}$$

From Lipshitz and the triangle inequality,

$$\begin{aligned}
\|e_{n+1}\| &\le \|e_n\| + \frac{h}{2}\left\{\|\mathbf{f}(t_n,\mathbf{y}_n) - \mathbf{f}(t_n,\mathbf{y}(t_n))\| + \|\mathbf{f}(t_{n+1},\mathbf{y}_{n+1}) - \mathbf{f}(t_{n+1},\mathbf{y}(t_{n+1}))\|\right\}\\
&\quad + \frac{h^2}{12}\left\{\|\mathbf{g}(t_n,\mathbf{y}_n) - \mathbf{g}(t_n,\mathbf{y}(t_n))\| + \|\mathbf{g}(t_{n+1},\mathbf{y}_{n+1}) - \mathbf{g}(t_{n+1},\mathbf{y}(t_{n+1}))\|\right\} + Ch^5\\
&\le \frac{1 + \frac{1}{2}h\lambda_f + \frac{1}{12}h^2\lambda_g}{1 - \frac{1}{2}h\lambda_f - \frac{1}{12}h^2\lambda_g}\|e_n\| + \frac{C}{1 - \frac{1}{2}h\lambda_f - \frac{1}{12}h^2\lambda_g}h^5\,.
\end{aligned}$$

Assuming $h < 1$, the inequality is still satisfied if we substitute $\frac{1}{2}h\lambda_f + \frac{1}{12}h^2\lambda_g$ with appropriate $\frac{1}{2}h\lambda$:

$$\begin{aligned}\|e_{n+1}\| &\leq \frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\|e_n\| + \frac{C}{1-\frac{1}{2}h\lambda}h^5 \\ &\leq \frac{C}{\lambda}\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n - 1\right]h^4\,.\end{aligned}$$

This is indeed satisfied for $n = 0$ and by induction for $n + 1$:

$$\begin{aligned}\|e_{n+1}\| &\leq \left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)\frac{C}{\lambda}h^4\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^n - 1\right] + \left(\frac{C}{1-\frac{1}{2}h\lambda}\right)h^5 \\ &= \frac{C}{\lambda}\left[\left(\frac{1+\frac{1}{2}h\lambda}{1-\frac{1}{2}h\lambda}\right)^{n+1} - 1\right]h^4\,.\end{aligned}$$

Therefore the method is order h^4.

Problem 1.7

Repeated differentiation of the ODE, for analytic $\mathbf{f}$ yields explicit expressions for functions $\mathbf{g}_m$ such that

$$\frac{\mathrm{d}^m\mathbf{y}(t)}{\mathrm{d}t^m} = \mathbf{g}_m(t,\mathbf{y}(t))\,, \quad m = 0, 1, ...$$

Hence $\mathbf{g}_0(t,\mathbf{y}) = \mathbf{y}$ and $\mathbf{g}_1(t,\mathbf{y}) = \mathbf{f}(t,\mathbf{y})$.

a) Assuming for simplicity that $\mathbf{f} = \mathbf{f}(\mathbf{y})$ (i.e. that the ODE system is autonomous), derive $\mathbf{g}_3$.

Solution

$$\begin{aligned}\mathbf{g}_3(t,\mathbf{y}) &= \frac{\mathrm{d}^3\mathbf{y}(t)}{\mathrm{d}t^3} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{f}(\mathbf{y}))\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{y}}\mathbf{f}\right) \\ &= \frac{\mathrm{d}^2\mathbf{f}}{\mathrm{d}y^2}\mathbf{f}^2 + \left(\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}y}\right)^2\mathbf{f}\,.\end{aligned}$$

b) Prove that the mth *Taylor method*

$$\mathbf{y}_{n+1} = \sum_{k=0}^{n}\frac{1}{k!}h^k\mathbf{g}_k(t_n,\mathbf{y}_n)\,, \quad n = 0, 1, ...,$$

is of order m for $m = 1, 2, ...$

Solution

The exact Taylor expansion of $y(t_{n+1})$ can be written

$$y(t_{n+1}) = \sum_{k=0}^{\infty}\frac{1}{k!}h^k\mathbf{g}_k(t_n,\mathbf{y}_n) = \sum_{k=0}^{m}\frac{1}{k!}h^k\mathbf{g}_k(t_n,\mathbf{y}_n) + \frac{1}{(m+1)!}h^{m+1}\mathbf{y}^{(m+1)}(\xi)\,.$$

Calculate:

$$\begin{aligned}y_{n+1} - y(t_{n+1}) &= \sum_{k=0}^{m}\frac{1}{k!}h^k\mathbf{g}_k(t_n,\mathbf{y}_n) - \left(\sum_{k=0}^{m}\frac{1}{k!}h^k\mathbf{g}_k(t_n,\mathbf{y}_n) + \frac{1}{(m+1)!}h^{m+1}\mathbf{y}^{(m+1)}(\xi)\right) \\ &= -\frac{1}{(m+1)!}h^{m+1}\mathbf{y}^{(m+1)}(\xi) = O(h^{m+1})\,.\end{aligned}$$

Therefore the local truncation error is of order h^m.

The global error is

$$\begin{aligned} e_{n+1} &= \sum_{k=0}^{m} \frac{1}{k!} h^k g_k(t_n, y_n) - \sum_{k=0}^{m} \frac{1}{k!} h^k g_k(t_n, y(t_n)) + O(h^{m+1}) \\ &= \sum_{k=0}^{m} \frac{1}{k!} h^k (g_k(t_n, y_n) - g_k(t_n, y(t_n)) + O(h^{m+1}) \,. \end{aligned}$$

The Lipshitz condition gives us

$$\|e_{n+1}\| \leq \left(1 + \sum_{k=1}^{m} \frac{1}{k!} h^k \lambda_k\right) \|e_n\| + C h^{m+1} \,.$$

For $h < 1$ and appropriate λ we can say

$$\begin{aligned} \|e_{n+1}\| &\leq (1 + h\lambda) \|e_n\| + C h^{m+1} \\ &\leq \frac{C}{\lambda} h^m [(1 + h\lambda)^n - 1] \,. \end{aligned}$$

This indeed holds for $n = 0$ and by induction for $n + 1$:

$$\|e_{n+1}\| \leq (1 + h\lambda) \frac{C}{\lambda} h^m [(1 + h\lambda)^n - 1] + C h^{m+1} = \frac{C}{\lambda} h^m [(1 + h\lambda)^{n+1} - 1]$$

Therefore the method is order h^m.

c) Let $\mathbf{f}(\mathbf{y}) = \Lambda \mathbf{y} + \mathbf{b}$, where the matrix Λ and the vector $\mathbf{b}$ are independent of t. Find the explicit form of $\mathbf{g}_m$ for $m = 0, 1, \ldots$ and thereby prove that the mth Taylor method reduces to the recurrence

$$\mathbf{y}_{n+1} = \left(\sum_{k=0}^{m} \frac{1}{k!} h^k \Lambda^k\right) \mathbf{y}_n + \left(\sum_{k=1}^{m} \frac{1}{k!} h^k \Lambda^{k-1}\right) \mathbf{b} \,, \quad n = 0, 1, \ldots$$

Solution

Recall $\mathbf{y}' = \mathbf{f}(\mathbf{y})$ and

$$\mathbf{g}_m = \frac{\mathrm{d}^m \mathbf{y}}{\mathrm{d}t^m} \,.$$

Then

$$\mathbf{g}_2 = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{f}(\mathbf{y})) = \frac{\mathrm{d}}{\mathrm{d}t}(\Lambda \mathbf{y} + \mathbf{b}) = \Lambda \mathbf{f} = \Lambda^2 \mathbf{y} + \Lambda \mathbf{b} \,,$$

and

$$\mathbf{g}_3 = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{g}_2) = \frac{\mathrm{d}}{\mathrm{d}t}(\Lambda^2 \mathbf{y} + \Lambda \mathbf{b}) = \Lambda^2 \mathbf{f} = \Lambda^3 \mathbf{y} + \Lambda^2 \mathbf{b} \,.$$

A pattern becomes apparent, thus

$$\mathbf{g}_m = \Lambda^m \mathbf{y} + \Lambda^{m-1} \mathbf{b} \,.$$

We can substitute this into the formula for the mth order Taylor method and splitting the sum:

$$\begin{aligned} \mathbf{y}_{n+1} &= y_n + \sum_{k=1}^{m} \frac{1}{k!} h^k (\Lambda^m \mathbf{y}_n + \Lambda^{m-1} \mathbf{b}) \\ &= \left(\sum_{k=0}^{m} \frac{1}{k!} h^k \Lambda^k\right) y_n + \left(\sum_{k=1}^{m} \frac{1}{k!} h^k \Lambda^{k-1}\right) \mathbf{b} \,. \end{aligned}$$

Euler's Method

Test Problem 1

$$\mathbf{f}(t, \mathbf{y}) = (-2y_1, -3y_2)$$
$$\mathbf{y}(0) = (1, 2)$$

Exact Solution:

$$\mathbf{y}(t) = (y_1(0)\mathrm{e}^{-2t}, y_2(0)\mathrm{e}^{-3t})$$

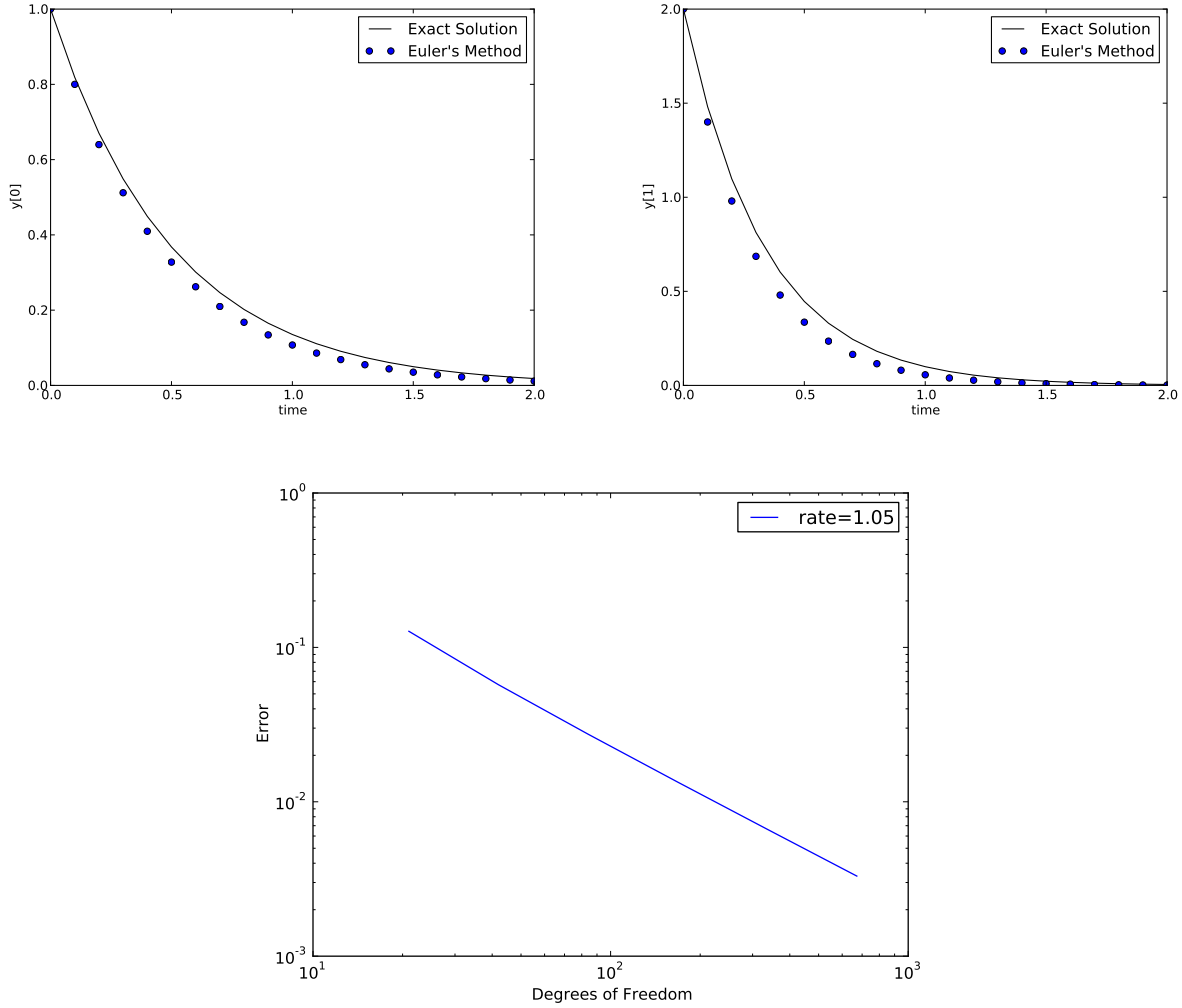


Figure 1: Problem 1

Test Problem 2

$$\mathbf{f}(t, \mathbf{y}) = (\frac{1}{2}, -t)$$
$$\mathbf{y}(0) = (3, 4)$$

Exact Solution:

$$\mathbf{y}(t) = (y_1(0) + 0.5t, y_2(0) - 0.5t^2)$$

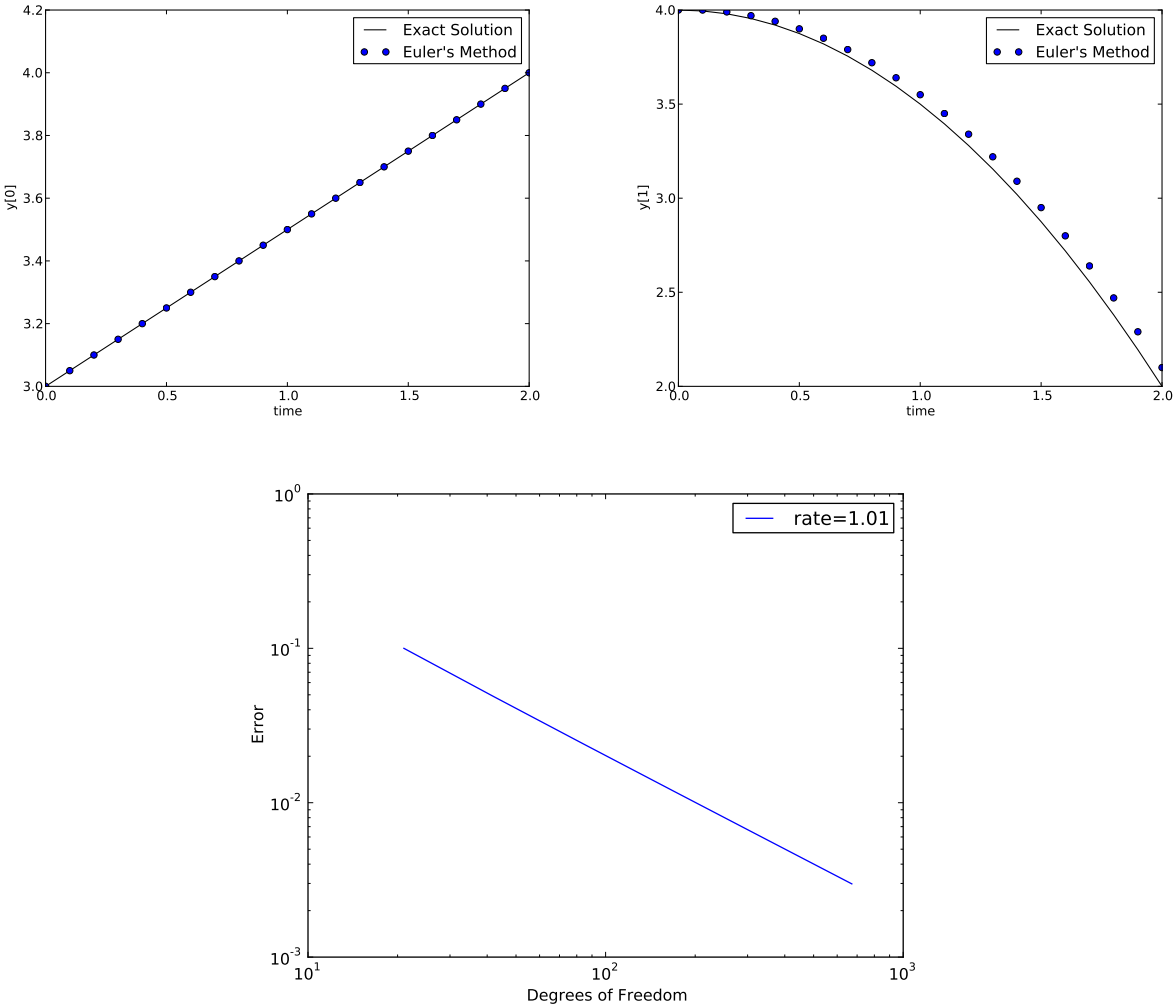


Figure 2: Problem 2

Test Problem 3

$$\mathbf{f}(t, \mathbf{y}) = -\mathbf{y}^2 t$$
$$\mathbf{y}(0) = 2$$

Exact Solution:

$$\mathbf{y}(t) = y(0)/(1 + 0.5t^2 y(0))$$

Test Problem 4

$$\mathbf{f}(t, \mathbf{y}) = (-y_1, y_1)$$
$$\mathbf{y}(0) = (1, 2)$$

Exact Solution:

$$\mathbf{y}(t) = (y_1(0)\mathrm{e}^t, y_1(0)(1 - \mathrm{e}^t) + y_2(0))$$

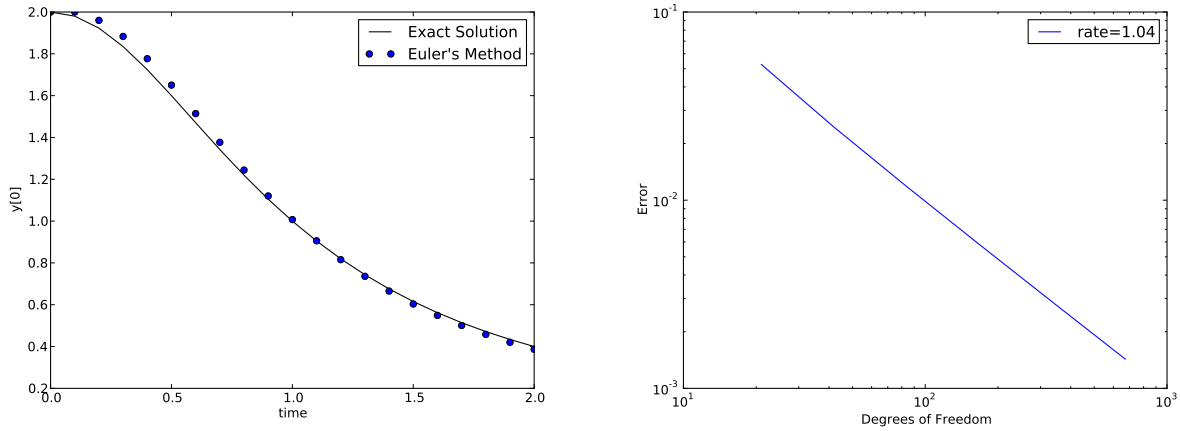
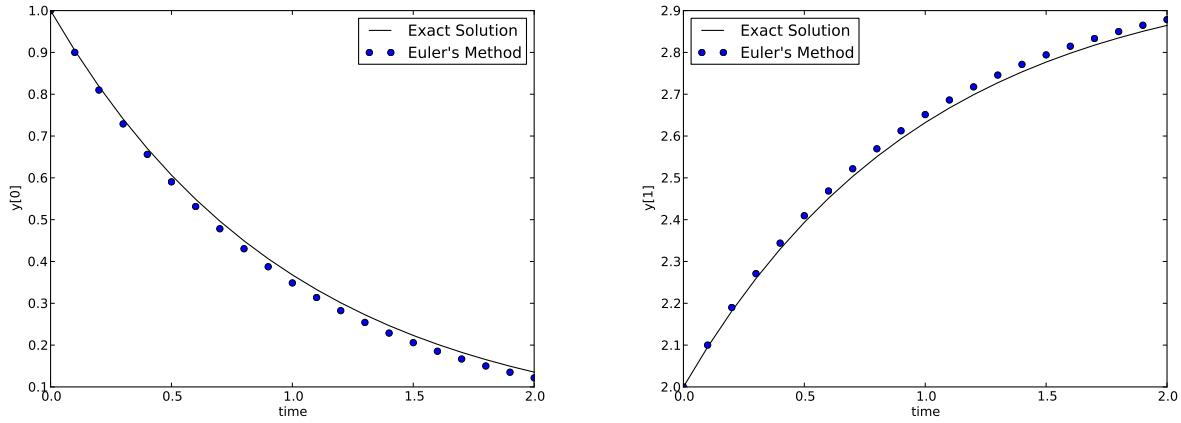


Figure 3: Problem 3



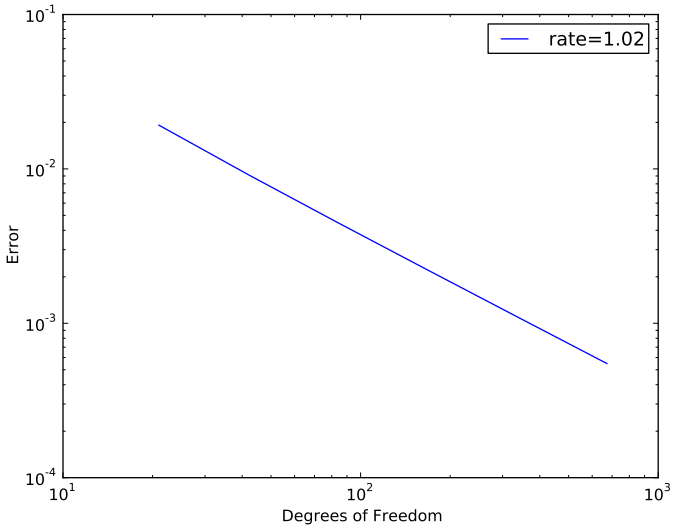


Figure 4: Problem 4

Test Problem 5

$$\mathbf{f}(t, \mathbf{y}) = -1/y)$$
$$\mathbf{y}(0) = 1$$

Exact Solution:

$$\mathbf{y}(t) = \sqrt{1 - 2t}$$

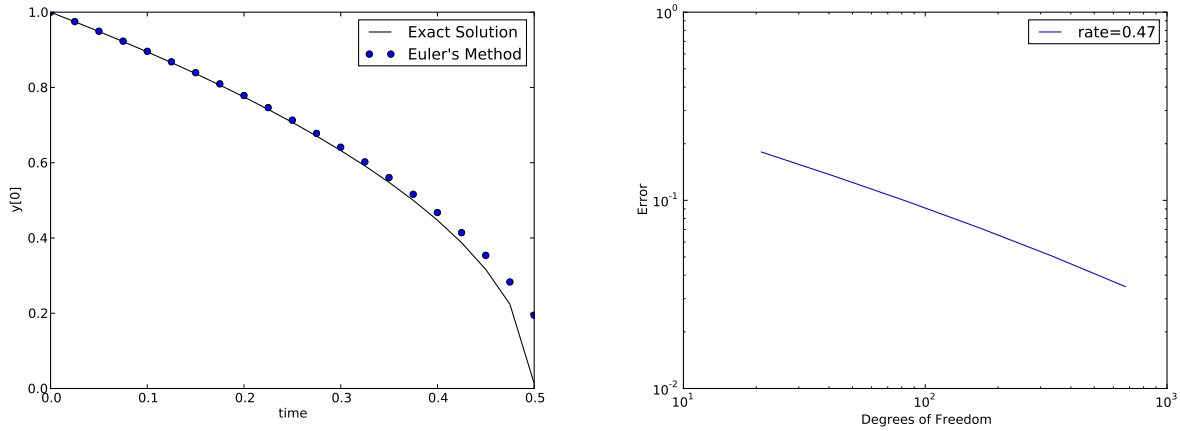


Figure 5: Problem 5

Problem 5 is not Lipshitz in y, therefore the convergence is not optimal.

Euler.cpp

```
// Euler's Method
// Written by: Truman Ellis
// Numerical Treatment of Differential Equations
// Spring 2011

#include "Python.h"
#include <iostream>
#include <vector>
#include <memory>
#include <fstream>
#include <math.h>

using namespace std;

class DifferentialEquation
{
    public:
        DifferentialEquation(){}
        virtual void eval (double t, vector<double> y, vector<double> &f) = 0;
        virtual void exact(double t0, double t1, vector<double> &y) = 0;
};


class DE0 : public DifferentialEquation
{
    // f(t,y) = -1/y, d=1
    public:
        void eval (double t, vector<double> y, vector<double> &f)
        {
            f[0] = -1/y[0];
        }

        void exact(double t0, double t1, vector<double> &y)
        {
            y[0] = sqrt(1-2*t1);
        }
};


class DE1 : public DifferentialEquation
{
    // f(t,y) = (-2y[0],-3y[1]), d=2
    public:
        void eval (double t, vector<double> y, vector<double> &f)
        {
            f[0] = -2*y[0];
            f[1] = -3*y[1];
        }

        void exact(double t0, double t1, vector<double> &y)
        {
            double dt = t1-t0;
            y[0] = y[0]*exp(-2*dt);
            y[1] = y[1]*exp(-3*dt);
        }
};


class DE2 : public DifferentialEquation
{
    // f(t,y) = (1/2, -t), d=2
    public:
        void eval (double t, vector<double> y, vector<double> &f)
        {
            f[0] = 0.5;
            f[1] = -t;
        }

        void exact(double t0, double t1, vector<double> &y)
        {
            y[0] = y[0] + 0.5*(t1-t0);
            y[1] = y[1] - 0.5*(t1*t1-t0*t0);
        }
};


class DE3 : public DifferentialEquation
{
    // f(t,y) = (y[0]^2t,y[0]*y[1]), d=1
    public:
        void eval (double t, vector<double> y, vector<double> &f)
        {
            f[0] = -y[0]*y[0]*t;
        }
```

```
        void exact(double t0, double t1, vector<double> &y)
        {
            y[0] = y[0]/(1 + 0.5*(t1*t1-t0*t0)*y[0]);
        }
};


class DE4 : public DifferentialEquation
{
    // f(t,y) = (y[0]^2t,y[0]*y[1]), d=2
    public:
        void eval (double t, vector<double> y, vector<double> &f)
        {
            f[0] = -y[0];
            f[1] = y[0];
        }

        void exact(double t0, double t1, vector<double> &y)
        {
            double e = exp(t0-t1);
            double yy = y[0];
            y[0] = yy*e;
            y[1] = yy*(1-e)+y[1];
        }
};


class EulerSolution
{
    private:
        shared_ptr<DifferentialEquation> de;
        double tStart;
        double tStop;
        int Nsteps;
        double h;
        vector<double> y0;
        int dim;
        vector< vector<double> > solution;
        vector<double> time;
        vector< vector<double> > EXsolution;
        double error;

    public:
        EulerSolution(shared_ptr<DifferentialEquation> de_, double tStart_,
                      double tStop_, int Nsteps_, vector<double> y0_): de(de_),
                      tStart(tStart_), tStop(tStop_), Nsteps(Nsteps_), y0(y0_)
        {
            h = (tStop-tStart)/double(Nsteps-1);
            dim = y0.size();
            solution.push_back(y0);
            time.push_back(tStart);
        }

        double solve()
        {
            vector<double> f(2);
            for (int n=0; n < Nsteps; ++n){
                double t = tStart + n*h;
                time.push_back(t+h);
                de->eval(t, solution[n], f);
                for (int d = 0; d < dim; ++d){
                    f[d] = solution[n][d] + f[d]*h;
                }
                solution.push_back(f);
            }
            error = exact();
            return error;
        }

        double exact()
        {
            double t0 = tStart;
            double t1 = tStart;
            vector<double> EXy = y0;
            error = 0;
            for (int n=0; n < Nsteps; ++n){
                de->exact(t0, t1, EXy);
                EXsolution.push_back(EXy);
                for (int d=0; d< dim; ++d){
                    error = max(error, fabs(EXy[d]-solution[n][d]));
                }
                t0 = t1;
                t1 += h;
            }
```

```
            return error;
        }

        void write()
        {
            ofstream output("output.txt");
            output << "#time        Euler: y[0]  Exact: y[0]";
            cout << "#time        Euler: y[0]  Exact: y[0]";
            for (int d = 1; d < dim; ++d)
            {
                cout << "  Euler: y["<<d<<"]  Exact: y["<<d<<"]";
                output << "  Euler: y["<<d<<"]  Exact: y["<<d<<"]";
            }
            cout << endl;
            output << endl;
            cout.setf(ios::fixed,ios::floatfield);
            output.setf(ios::fixed,ios::floatfield);
            cout.precision(10);
            output.precision(10);
            for(int i=0; i < Nsteps; i++){
                output << time[i] << " ";
                cout << time[i] << " ";
                for(int d=0; d < dim; ++d){
                    output << solution[i][d] << " " << EXsolution[i][d] << " ";
                    cout << solution[i][d] << " " << EXsolution[i][d] << " ";
                }
                output << endl;
                cout << endl;
            }
            output.close();
        }
};

int main()
{
    int probNum;
    int probDim;
    int Nsteps;
    double tStart;
    double tStop;
    vector<double> y0;
    vector<double> f;
    shared_ptr<DifferentialEquation> de;

    cout << "Which test problem (0, 1, 2, 3, 4): ";
    cin >> probNum;
    switch(probNum)
    {
        case 0:
            de = shared_ptr<DifferentialEquation> (new DE0);
            break;
        case 1:
            de = shared_ptr<DifferentialEquation> (new DE1);
            break;
        case 2:
            de = shared_ptr<DifferentialEquation> (new DE2);
            break;
        case 3:
            de = shared_ptr<DifferentialEquation> (new DE3);
            break;
        case 4:
            de = shared_ptr<DifferentialEquation> (new DE4);
            break;
        default:
            return 1;
    }
    cout << "Please enter problem dimension: ";
    cin >> probDim;
    cout << "Please enter start time: ";
    cin >> tStart;
    cout << "Please enter stop time: ";
    cin >> tStop;
    cout << "Please enter number of steps: ";
    cin >> Nsteps;
    cout << "Please enter "<<probDim<<" initial conditions: ";
    for (int d = 0; d < probDim; ++d){
        double ic;
        cin >> ic;
        y0.push_back(ic);
    }

    EulerSolution euler(de, tStart, tStop, Nsteps, y0);
    double error = euler.solve();
    euler.write();
    cout << "error: " << error << endl;
```

```
    vector<double> errorArray;
    vector<double> refArray;

    ofstream output("errors.txt");
    output.setf(ios::fixed,ios::floatfield);
    output.precision(10);
    output << Nsteps << " " << error << endl;
    refArray.push_back(Nsteps);
    errorArray.push_back(error);
    for (int ref = 1; ref <= 5; ++ref){
        Nsteps = 2*Nsteps;
        EulerSolution eulerRef(de, tStart, tStop, Nsteps, y0);
        refArray.push_back(Nsteps);
        error = eulerRef.solve();
        errorArray.push_back(error);
        output << Nsteps << " " << error << endl;
    }
    output.close();

    Py_Initialize();
    FILE *fp = fopen ("EulerPlot.py", "r");
    PyRun_SimpleFile(fp, "EulerPlot.py");
    Py_Exit(0);


    return 0;
}
```

EulerPlot.py

```
# -*- coding: utf-8 -*-
# Euler's Method
# Written by: Truman Ellis
# Numerical Treatment of Differential Equations
# Spring 2011

from pylab import *

data=loadtxt('output.txt')

for d in range(1, (data.shape[1]+1)/2):
    figure(d)
    plot(data[:,0],data[:,2*d],'k-', label='Exact Solution')
    plot(data[:,0],data[:,2*d-1],'o', label="Euler's Method")
    legend(loc='best')
    xlabel('time')
    ylabel('y['+str(d-1)+']')

figure()
convergenceData = loadtxt('errors.txt')
(m,b) = polyfit(log(convergenceData[:,0]),log(convergenceData[:,1]),1)
loglog(convergenceData[:,0],convergenceData[:,1])
xlabel('Degrees of Freedom')
ylabel('Error')
legend(('rate=%.2f' % (-m,),), loc='best')
show()
```