Numerical Computing II

Homework 22: Initial Value Problems for Ordinary Differential Equations

Marty Fuhry

May 6, 2010

Exercise 22.2

To show that the Backwards Euler's Method is $O(h)$, we begin by showing the following:

$$\begin{aligned} u_n &= (\frac{1}{1-\lambda h})^n u_0 \\ &= (1 + \lambda h + O(h^2))^n u_0 \\ e^{\lambda nh + nO(h^2)} u_0 &= (1 + \lambda h + O(h^2))^n u_0. \end{aligned}$$

So we have equality with $O(h^2)$ between u_n and $e^{n\lambda h + nO(h^2)}$. Use this equality to solve the global error. Note that $t_n - t_0 = nh$.

$$\begin{aligned} u_n - u(t_n) &= u_n - u_0 e^{\lambda(t_n - t_0)} \\ &= u_n - u_0 e^{\lambda nh} \\ &= u_0 e^{\lambda nh + nO(h^2)} - u_0 e^{\lambda nh}. \end{aligned}$$

We know that $nh = t_n - t_0$ is a constant, so $nO(h^2) = O(nh^2) = (t_n - t_0)O(h) = O(h)$. Then,

$$e^{nO(h^2)} = e^{O(h)} = 1 + O(h).$$

Substituting this in for our global error yields:

$$\begin{aligned} u_n - u(t_n) &= u_0 e^{\lambda nh} e^{nO(h^2)} - u_0 e^{\lambda nh} \\ &= u_0 e^{\lambda nh} e^{O(h)} - u_0 e^{\lambda nh} \\ &= u_0(1 + O(h)) e^{\lambda nh} - u_0 e^{\lambda nh} \\ &= u_0 e^{\lambda nh} + u_0 O(h) - u_0 e^{\lambda nh} \\ &= u_0 O(h) \\ &= O(h) \end{aligned}$$

□

Exercise 22.3

We need to explicitly solve for u_1 in the trapezoidal method $u_1 = 1/2(u_0 + hf(t_0, u_0) + u_1 + hf(t_1, u_1))$. Multiplying this out and solving u_1 gives us:

$$\begin{aligned}
u_1 &= u_0 + \frac{h}{2}(f(t_0, u_0) + f(t_1, u_1)) \\
&= u_0 + \frac{h}{2}(\lambda u_0 + \lambda u_1) \\
&= u_0 + \frac{h\lambda}{2}u_0 + \frac{h\lambda}{2}u_1 \\
u_1 - \frac{h\lambda}{2}u_1 &= u_0 + \frac{h\lambda}{2}u_0 \\
u_1(1 - \frac{h\lambda}{2}) &= u_0(1 + \frac{h\lambda}{2}) \\
u_1 &= u_0\frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}.
\end{aligned}$$

Now, we need to manipulate u_1 by solving it as a geometric series:

$$\begin{aligned}
u_1 &= u_0(1 + \frac{h\lambda}{2})(\frac{1}{1 - h\lambda}) \\
&= u_0(1 + \frac{h\lambda}{2})(1 + \frac{h\lambda}{2} + (\frac{h\lambda}{2})^2 + (\frac{h\lambda}{2})^3 + O(h^4)) \\
&= u_0(1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{2(h\lambda)^3}{8} + \frac{(h\lambda)^3}{8} + O(h^4)) \\
&= u_0(1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{3(h\lambda)^3}{8} + O(h^4)).
\end{aligned}$$

Now, using the exponential expansion:

$$u_0 e^{h\lambda} = u_0(1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + O(h^4))$$

we can finally solve the local error:

$$\begin{aligned}
u_1 - u(t_1) =& u_0(1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{3(h\lambda)^3}{8} + O(h^4)) - \\
& u_0(1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + O(h^4)) \\
=& O(h^3).
\end{aligned}$$

□

Exercise 22.5

$$u_{j+1} = u_j + h(\alpha u'_{j+1} + \beta u'_j)$$

We use this method with the conditions $u'_l = f(t_l, u_l)$. We begin with the function $u(t) = 1$, which means that $f(t_l, u_l) = 0 \forall t_i, u_i$. Choose $t_0 = 0, t_1 = h$ and then $u(t_0) = 0, u(t_1) = 0$.

$$\begin{aligned}
u'_0 &= f(t_0, u_0) = 0 \\
u'_1 &= f(t_1, u_1) = 0 \\
u_1 &= u_0 + h(\alpha u'_1 + \beta u'_0) \\
&= u_0 + h(\alpha 0 + \beta 0) \\
u_1 &= u_0 = 0.
\end{aligned}$$

Now we try $u(t) = t$, which means that $f(t_l, u_l) = 1 \forall t_i, u_i$. Choose $t_0 = 0, t_1 = h$ and then $u(t_0) = 0, u(t_1) = h$.

$$\begin{aligned}
u'_0 &= f(t_0, u_0) = 1 \\
u'_1 &= f(t_1, u_1) = 1 \\
u_1 &= u_0 + h(\alpha f(t_1, u_1) + \beta f(t_0, u_0)) \\
&= u_0 + h(\alpha 1 + \beta 1) \\
h &= h(\alpha + \beta) \\
1 &= \alpha + \beta
\end{aligned}$$

We have our first equation, then. Now we try $u(t) = t^2$, which means that $f(t_l, u_l) = 2t$. Choose $t_0 = 0, t_1 = h$ and then $u(t_0) = 0, u(t_1) = h^2$.

$$\begin{aligned}
u'_0 &= f(t_0, u_0) = 0 \\
u'_1 &= f(t_1, u_1) = 2h \\
u_1 &= u_0 + h(\alpha u'_1 + \beta u'_0) \\
h^2 &= 0 + h(\alpha 2h + \beta 0) \\
h^2 &= \alpha 2h^2 \\
\frac{1}{2} &= \alpha = \beta.
\end{aligned}$$

Then, try $u(t) = t^3$, which means that $f(t_l, u_l) = 3t^2$. Choose $t_0 = 0, t_1 = h$ and then $u(t_0) = 0, u(t_1) = h^3$.

$$
\begin{aligned}
u_0' &= f(t_0, u_0) = 0 \\
u_1' &= f(t_1, u_1) = 3h^2 \\
u_1 &= u_0 + h(\alpha u_1' + \beta u_0') \\
h^3 &= 0 + h(\alpha 3h^2 + \beta 0) \\
h^3 &= \alpha 3h^3 \\
\frac{1}{3} &= \alpha \implies \beta = \frac{2}{3}.
\end{aligned}
$$

But this doesn't agree with our other equations. $\alpha = \beta = 1/2$ works for all polynomials up to and including t^2. When we reach the polynomial t^3, we need $\alpha = 1/3$, which conflicts for polynomials of lower degree.

Exercise 22.6

Apply each method to:

$$
\begin{aligned}
u'(t) &= -25u(t) \\
t &\geq t_0 \\
u(t_0) &= u_0.
\end{aligned}
$$

Trapezoidal Method

The trapezoidal method is given by:

$$u_{j+1} = u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, u_{j+1})).$$

We apply this rule for the function $f(t,u) = \lambda u$ and observe:

$$\begin{aligned}
u_{j+1} &= u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, u_{j+1})) \\
&= u_j + \frac{h}{2}(\lambda u_j + \lambda u_{j+1}) \\
&= u_j + \frac{h}{2}\lambda u_j + \frac{h}{2}\lambda u_{j+1}) \\
u_{j+1} - \frac{h}{2}\lambda u_{j+1} &= u_j + \frac{h}{2}\lambda u_j \\
u_{j+1}(1 - \frac{h}{2}\lambda) &= u_j + \frac{h}{2}\lambda u_j \\
u_{j+1} &= u_j \frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda} \\
u_{j+1} &= u_0 (\frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda})^j.
\end{aligned}$$

This converges if and only if

$$\frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda} < 1. \tag{1}$$

Since $h > 0$ and $\lambda < 0$, (1) is equivalent to

$$\begin{aligned}
1 + \frac{h}{2}\lambda &< 1 - \frac{h}{2}\lambda \\
h\lambda &< 0 \\
-25h &< 0.
\end{aligned}$$

In this case, it works for all positive h.

Heun's Method

Heun's method is given by:

$$u_{j+1} = u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, \hat{u_{j+1}})).$$

Note that $\hat{u_{j+1}}$ is taken from Euler's method

$$\hat{u_{j+1}} = u_j + hf(t_j, u_j).$$

Rewriting Heun's method, we get:

$$u_{j+1} = u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, u_j + hf(t_j, u_j))).$$

We apply this rule for our function given by $f(t, u) = \lambda u$ and observe:

$$\begin{aligned}
u_{j+1} &= u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, u_j + hf(t_j, u_j))) \\
&= u_j + \frac{h}{2}(f(t_j, u_j) + f(t_{j+1}, u_j + h\lambda u_j)) \\
&= u_j + \frac{h}{2}(\lambda u_j + \lambda(u_j + h\lambda u_j)) \\
&= u_j + \frac{h}{2}(2\lambda u_j + h\lambda^2 u_j)) \\
&= u_j + h\lambda u_j + \frac{h^2}{2}\lambda^2 u_j)) \\
&= u_j(1 + h\lambda + \frac{h^2}{2}\lambda^2) \\
&= u_0(1 + h\lambda + \frac{h^2}{2}\lambda^2)^j.
\end{aligned}$$

This converges if and only if

$$\begin{aligned}
1 + h\lambda + \frac{h^2}{2}\lambda^2 &< 1 \\
2h\lambda + h^2\lambda^2 &< 0 \\
h\lambda(2 + h\lambda) &< 0.
\end{aligned}$$

Since $h\lambda < 0$, we need $2 + h\lambda > 0$ to satisfy the inequality. Then,

$$\begin{aligned}
2 + h\lambda &> 0 \\
h\lambda &> -2 \\
h &< \frac{2}{25}.
\end{aligned}$$

We get this from $\lambda = -25$. So, when $h < \frac{2}{25}$, Heun's method converges in this case.

Example 22.7: Explicit Multistep Method

The Explicit Multistep Method from Example 22.7 is given by

$$u_{j+1} = u_j + h(\alpha u'_j + \beta u'_{j-1}).$$

Use the fact that $u_l' = f(t_l, u_l) = \lambda u_l$, and apply this rule for $f(t, u) = \lambda u$.

$$\begin{aligned} u_{j+1} &= u_j + h(\alpha u_j' + \beta u_{j-1}') \\ &= u_j + h(\alpha\lambda u_j + \beta\lambda u_{j-1}) \\ &= u_j + h\alpha\lambda u_j + h\beta\lambda u_{j-1}. \end{aligned}$$

We can't solve explicitly for this function since it is implicit, but we can use our knowledge of the values of u_j and u_{j-1} to show that since $\lambda = -25$, we have a decreasing function. Therefore, $u_j < u_{j-1}$. Now, we have an upper bound on our u_{j+1} value.

$$\begin{aligned} u_{j+1} &= u_j + h\alpha\lambda u_j + h\beta\lambda u_{j-1} \\ &< u_{j-1}(1 + h\alpha\lambda + h\beta\lambda). \end{aligned}$$

This inequality is equal to

$$u_0(1 + h\alpha\lambda + h\beta\lambda)^{j-1}$$

which converges if and only if

$$\begin{aligned} 1 + h\alpha\lambda + h\beta\lambda) &< 1 \\ h\alpha\lambda + h\beta\lambda) &< 0 \\ h\lambda(\alpha + \beta) &< 0 \\ h(\alpha + \beta) &> 0. \end{aligned}$$

Notice the inequality was flipped since $\lambda = -25$. So this converges if h is positive and the sum of the coefficients $\alpha + \beta$ is positive.

Conclusions

The only method which required h to be small was Heun's Method. Both the Trapezoidal Method and the Explicit Multistep Method from Example 22.7 converged for all positive values of h (so long as the sum of the coefficients $\alpha + \beta$ was positive for the Explicit Multistep Method). Therefore, both the Trapezoidal Method and the Explicit Multistep Method allowed the largest step length by not setting any bounds on the size of h.

Exercise 22.8

$$u'' - \epsilon(1 - u^2)u' + u = 0 \tag{2}$$

We begin by setting these up into a system of linear ordinary differential equations. Let $v = u'$, making $v' = u''$. Then,

$$v' = \epsilon(1 - u^2)v - u. \tag{3}$$

We can build the following system of equations based on (2) and (3):

$$\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} v \\ \epsilon(1-u^2)v - u \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & \epsilon(1-u^2) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

The following code can be used to model the system of ordinary differential equations.

```
1 function ydot = f(t,y)
2     eps = 1;
3     ydot = [0, 1; -1, eps*(1-y(1)^2)];
4     ydot = ydot*y;
5 end
```

ODE Solver *lsode*

We can use the `lsode` solver in Octave to solve the following function with $\epsilon = 1$. Run the following commands.

```
> y0 = [1/2 1/2]';
> t  = linspace(0,25,500);
> y  = lsode("f", y0, t);
```

See the final pages for the graphs this method produced. Note the regularity of these graphs, especially for when $\epsilon = .1$ and $\epsilon = .01$. The graphs are very normal without any "weird" nonstandard curves produced by the other methods.

ODE Solver *ode23*

The `ode23` ODE Solver worked very well and produced a much more interesting set of graphs than the `lsode` solver produced. The graphs for when $\epsilon = 1$ look nearly identical, but for the other two cases, we have a very neat looking start conforming to a diverging (or so it seems) pattern in the oscillations.

ODE Solver *ode45*

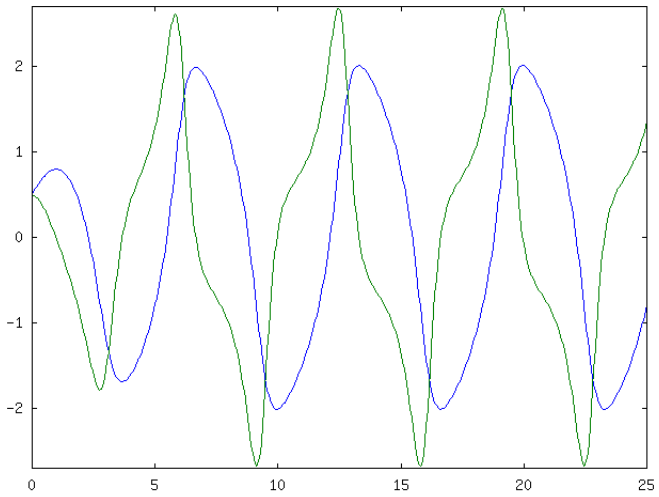
This method produced perhaps the "second best looking" graphs next to the uniformly beautiful graphs that `lsode` produced. These graphs still have sharp cusps, but for the most part they look neat and quite good without much bizarre oscillations like in `ode23`.

ODE Solver *ode78*

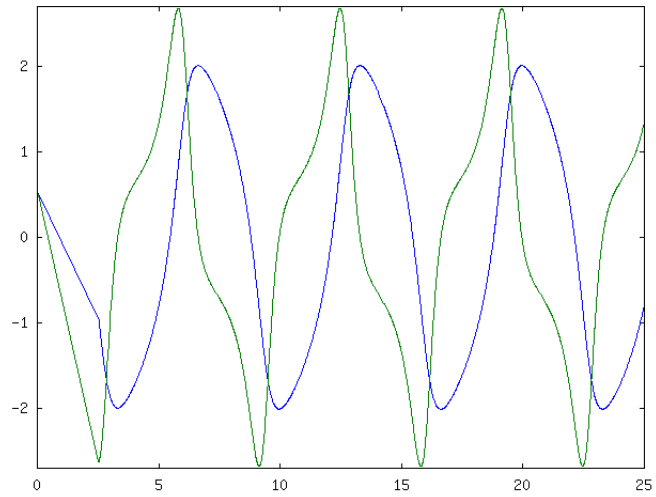
Since Octave does not have a `ode15` method (or at least, doesn't have one in the standard odesolver package), I chose to use `ode78` as a fourth example. This method produces the sharpest graphs of any of the solvers discussed here. Perhaps simply being the "highest order" solver of the bunch doesn't mean that it will produce the best looking graphs.

Conclusions

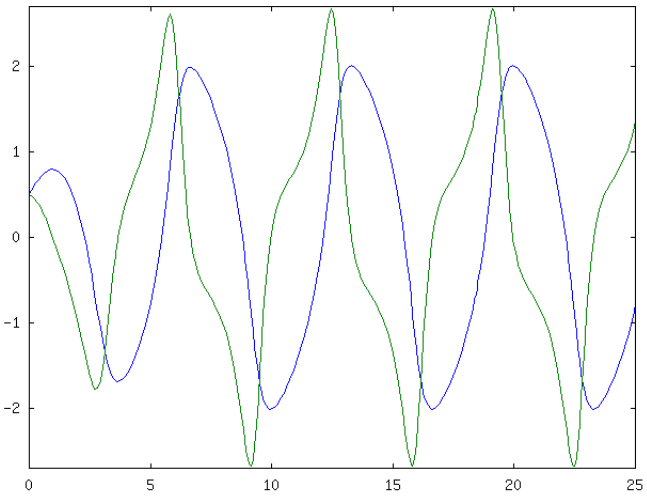
The fastest solvers were probably `ode45` and `lsode`. The slowest solver may have been `ode23`, though the computed examples here aren't quite big enough to provide a meaningful insight into the time complexity of the ODE Solvers' algorithms. In Octave's help files, I read that `lsode` has both a relative and absolute tolerance of 1.49×10^{-8} and uses a stiff integration method. The rest of the solvers, `ode45`, `ode23`, and `ode78` all use a relative tolerance of 10^{-3}.



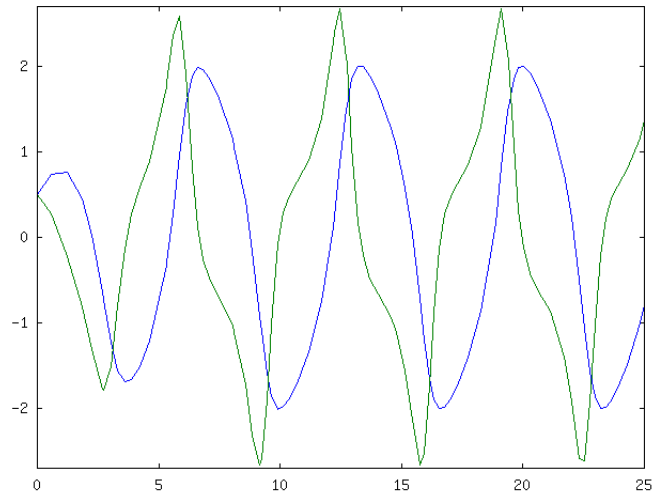
(a) *lsode*



(b) *ode23*

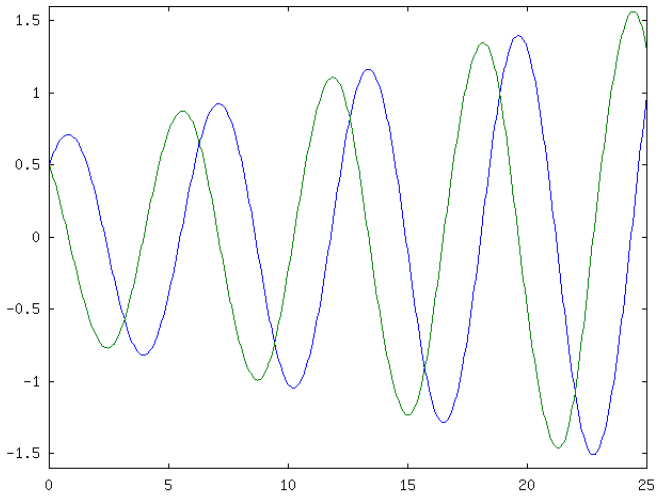


(c) *ode45*

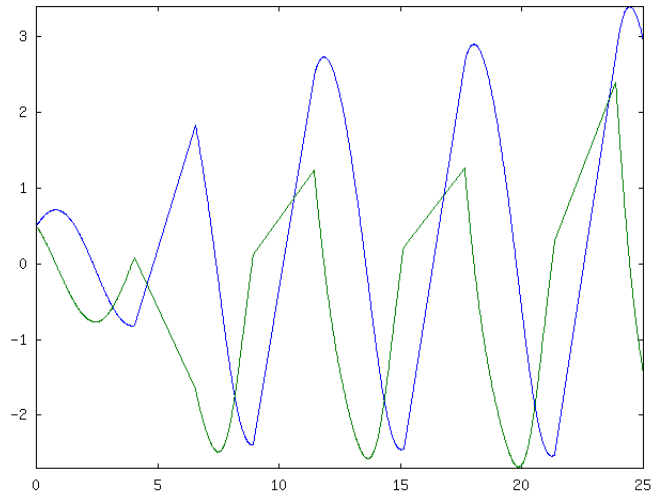


(d) *ode78*

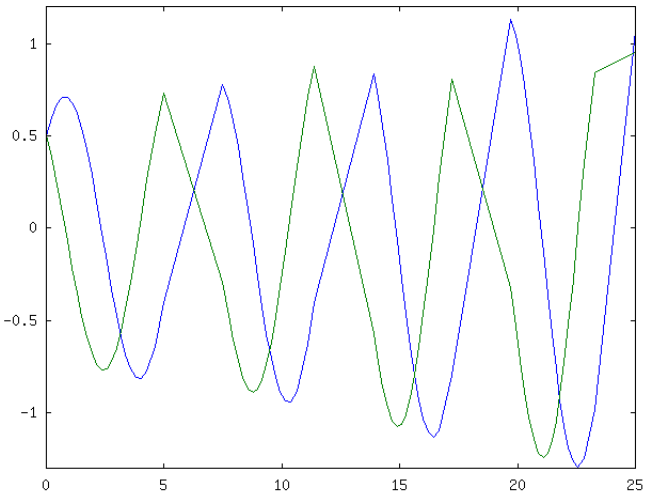
Figure 1: Graph of (3) with $\epsilon = 1$



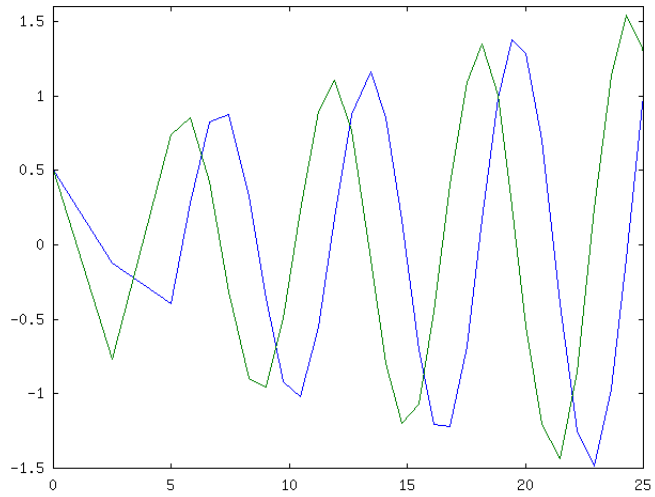
(a) *lsode*



(b) *ode23*

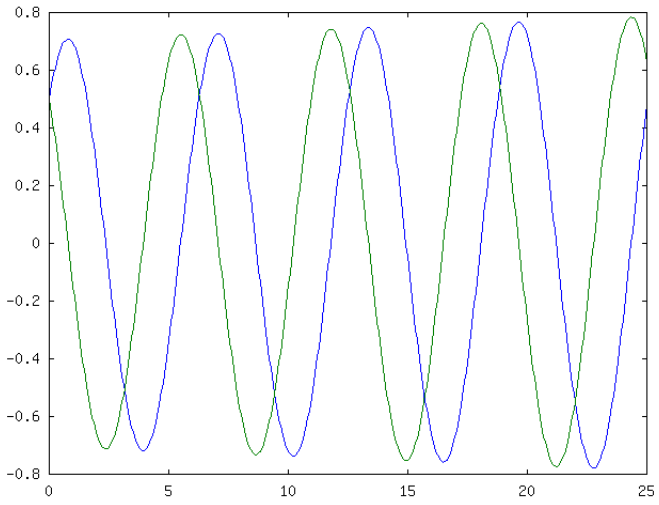


(c) *ode45*

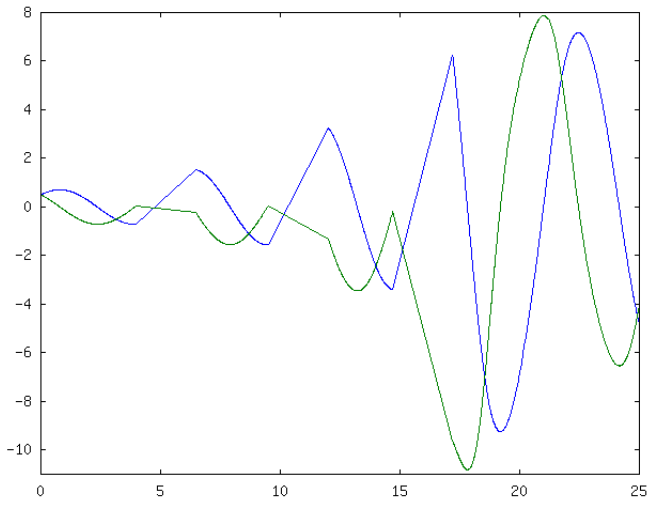


(d) *ode78*

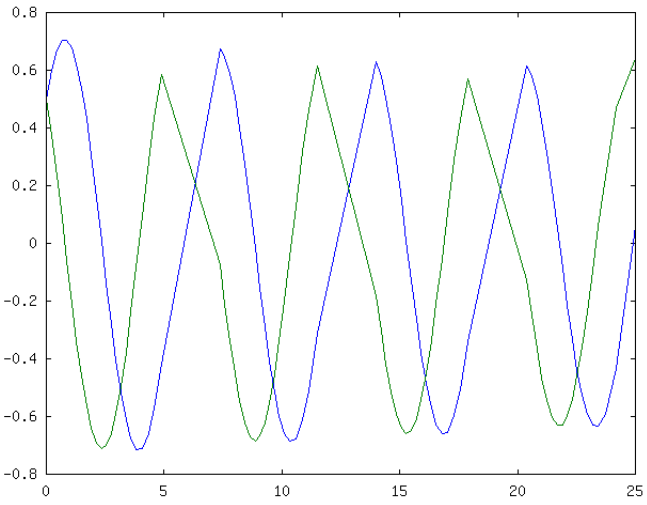
Figure 2: Graph of (3) with $\epsilon = .1$



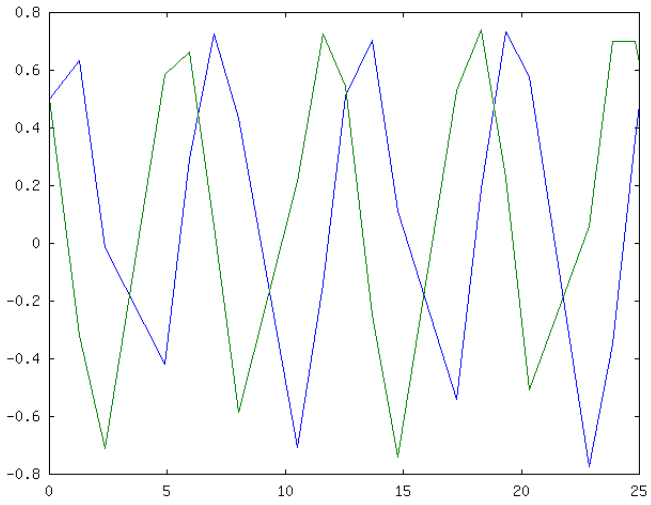
(a) *lsode*



(b) *ode23*



(c) *ode45*



(d) *ode78*

Figure 3: Graph of (3) with $\epsilon = .01$