CS370 Assignment 2

Daniel Burstyn (20206120)

Feb 16, 2009

1 Analytic Problems

1. a) $x_1'(t) = w_1(t)$
   $x_2'(t) = w_2(t)$
   $w_1'(t) + c_1(w_1(t) - w_2(t)) + k_1(x_1(t) - x_2(t)) + k_2x_1(t) = sin(t)$
   $w_2'(t) + c_2(x_1(t) - x_2(t)) - c_1(w_1(t) - w_2(t)) = 0$
   With initial conditions: $x_1(0) = x_2(0) = w_1(0) = w_2(0) = 0$

   b) We need to rearrange our functions like so:
   $w_1'(t) = sin(t) - c_1(w_1(t) - w_2(t)) - k_1(x_1(t) - x_2(t)) - k_2x_1(t)$
   $w_2'(t) = c_1(w_1(t) - w_2(t)) - c_2(x_1(t) - x_2(t))$
   Thus, we get our vectors:
   $$y = \begin{pmatrix} x_1(t) \\ x_2(t) \\ w_1(t) \\ w_2(t) \end{pmatrix}, f = \begin{pmatrix} w_1(t) \\ w_2(t) \\ sin(t) - c_1(w_1(t) - w_2(t)) - k_1(x_1(t) - x_2(t)) - k_2x_1(t) \\ c_1(w_1(t) - w_2(t)) - c_2(x_1(t) - x_2(t)) \end{pmatrix}$$

2. For this question we have $y'(t) = f(t, y) = -\lambda y$ to test the stability for:
   $y_{n+1} = y_n + \frac{h}{4}[f(t_n, y_n) + 3f(t_n + \frac{2h}{3}, y_n + \frac{2}{3}hf(t_n, y_n))]$

$$\begin{aligned}
y_{n+1} &= y_n + \frac{h}{4}\left[f(t_n, y_n) + 3f\left(t_n + \frac{2h}{3}, y_n + \frac{2}{3}hf(t_n, y_n)\right)\right] \\
&= y_n + \frac{h}{4}\left[-\lambda y_n - 3\lambda\left(y_n - \frac{2}{3}h\lambda y_n\right)\right] \\
&= y_n\left[1 - \frac{h\lambda}{4} - \frac{3h\lambda}{4} + \frac{h^2\lambda^2}{2}\right] \\
&= y_n\left[1 - h\lambda + \frac{h^2\lambda^2}{2}\right] \\
&= y_{n-1}\left[1 - h\lambda + \frac{h^2\lambda^2}{2}\right]^2 \\
&= \dots \\
&= y_0\left[1 - h\lambda + \frac{h^2\lambda^2}{2}\right]^{n+1}
\end{aligned}$$

Thus we can see that the solution decays to zero as $n \to \infty$ if and only if $|1 - h\lambda + \frac{h^2\lambda^2}{2}| \leq 1$
Solving this inequality we get:

$$-h\lambda + h^2\frac{\lambda^2}{2} \leq 0 \quad \Rightarrow \quad h \geq 0 \text{ and } h \leq \frac{2}{\lambda}$$

and

$$0 \leq 2 - h\lambda + h^2\frac{\lambda^2}{2} \quad \Rightarrow \quad \text{Nothing, since this is always true}$$

Therefore, we can conclude that this method is conditionally stable for $h \in \left[0, \frac{2}{\lambda}\right]$.

3. a) For reference, we have the taylor series, which is:
$y(t_{n+1}) = y(t_n) + y'(t_n)h + y''(t_n)\frac{h^2}{2} + y'''(t_n)\frac{h^3}{6} + \ldots$
Our function gives us:

$$\begin{aligned}
\frac{y_{n+1} - y_{n-1}}{2h} &= f(t_n, y_n) \\
y(t_{n+1}) &= y(t_{n-1}) + 2hy'(t_n) \\
&= \left(y(t_n) - y'(t_n)h + y''(t_n)\frac{h^2}{2} - y'''(t_n)\frac{h^3}{6} + \ldots\right) \\
&\quad +2h\left(y'(t_n) + y''(t_n)h + y'''(t_n)\frac{h}{2} + \ldots\right) \\
&= y(t_n) + y'(t_n)h + y''(t_n)\frac{5h^2}{2} + \ldots
\end{aligned}$$

We can see that this differs from the taylor expansion in the $y''(t_n)$ term, thus we can conclude that the local error is $O(h^2)$.

b) Like above, our method is $y_{n+1} = y_{n-1} + 2hy'_n$. We will use $y'(t) = f(t, y) = -\lambda y$, so that we have:

$$y_{n+1} = y_{n-1} - 2h\lambda y_n$$

Applying perturbation analysis, we get:

$$e_{n+1} = e_{n-1} - 2h\lambda e_n$$

Solve this recurrence relation by setting $e_n = \mu^n$:

$$\mu^2 + 2h\lambda\mu - 1 = 0$$

which gives

$$\begin{aligned}
\mu &= \frac{-2h\lambda \pm \sqrt{4h^2\lambda^2 + 4}}{2} \\
&= -h\lambda \pm \sqrt{h^2\lambda^2 + 1}
\end{aligned}$$

After some algebra, we can show that this method is *unconditionally unstable*.

2 Programming Problems

4. We find that given the values set in the assingment sheet, the pursuer will catch the target at $t = 12.9226$. See attached plot, and documented code.

5. The resulting table:

Tolerance	T	No. of Ftn Evals
10^{-3}	10.1699	326
10^{-4}	12.9243	558
10^{-5}	12.9225	523
10^{-6}	12.9226	703
10^{-7}	12.9225	917
10^{-8}	12.9225	1149
10^{-9}	12.9225	1573

   We can plainly see that as we lower the error tolerance, the number of function evaluations increases. Since our tolerance for error is so low it takes a much larger number of evaluations to find an acceptable solution. We can also see that, with the exception of the first entry, our T values are approximately equal. This makes sense, since the solution to the system should not depend on the error tolerance, for tolerance that is sufficiently small. Of course, this shows that 10^{-3} is not small enough, since we get a solution that is obviously incorrect.

6. Using the same table format as question 5, we get the table:

Tolerance	T	No. of Ftn Evals
10^{-5}	24.3609	15325
10^{-6}	18.6414	13009
10^{-7}	12.9225	12793

   Here we immediately notice that the number of function evaluations is much larger than in 5. This is because ode45 is a nonstiff solver, and thus uses many more evaluations to arrive at a solution. Since we have so many evaluations, our global errors get much higher, and so we need a lower tolerance in order to arrive at the correct solution. This explains the other solutions times that we got using ode45.

7. The evasion strategy I chose to implement is as follows:
   If the distance between the pursuer and the target is less than some constant, d_0, then the target should make a sharp turn in the opposite direction as normal. Besides that, the code is the same as the other programming problems.

   The reason for this evasion strategy is that, the target can turn more sharply than the pursuer, so when the pursuer gets close, we make a sharp turn away from where the pursuer expects us to be. This cause the pursuer to end up generally further away, and facing the wrong direction, allowing the target to escape.

   Setting d_0 is a matter of trial and error, and I've attached the plots for a few possibilities.