1. Using the fourth-order Runge-Kutte method, reproduce the results of Figs. 19.3, 19.4, and 19.5 for the following forced, damped Duffing equation:

\[
\begin{align*}
\frac{dy_1}{dt} &= y_2, \quad y_1(0) = 1, \\
\frac{dy_2}{dt} &= -\beta y_1 - \delta y_2 - \alpha y_1^3 + f \cos y_3, \quad y_2(0) = 0, \\
\frac{dy_3}{dt} &= 1, \quad y_3(0) = 0,
\end{align*}
\]

with \(\alpha = 1\), \(\beta = -1\), \(\delta = 0.22\), and \(f = 0.3\).

The subroutine \texttt{ode4}—which implements the fourth-order Runge-Kutte method algorithm—was used to approximate the solution to the above Duffing equation. Figure 1. is a MATLAB generated plot of \(y_2\) vs \(t\) for \(t \in [0, 200]\). Figure 2. is a MATLAB generated plot of \(y_2\) vs \(y_1\) for \(t \in [0, 200]\) and \(y_2\) vs \(y_1\) for \(t \in [100, 200]\). The parameters of the plots in both Figure 1. and Figure 2. are \(\alpha = 1\), \(\beta = -1\), \(\delta = 0.22\), and \(f = 0.3\).

Figure 1: Plot of \(y_2\) vs \(t\) for \(t \in [0, 200]\) with \(\alpha = 1\), \(\beta = -1\), \(\delta = 0.22\), and \(f = 0.3\); reproduction of Fig. 19.3
2. Use the fourth-order Runge-Kutte method to reproduce the results of Fig. 19.8 with $\alpha = 0$, $\beta = 10$, $\delta = 0.22$, and $f = 0.3$. Compare the approximations to $y_1(t = 100)$ given by the first, second, and fourth-order Runge-Kutte methods. Give a brief physical interpretation of the results.

The subroutine `ode4` was used to approximate the solution to the Duffing equation by implementing the fourth-order Runge-Kutte method. The subroutines `ode1` and `ode2` were also used to give first- and second-order approximations, respectively, to the solution to the Duffing equation for comparison with the fourth-order method. Figure 3. is a MATLAB generated plot of $y_2$ vs $t$ for $t \in [0, 200]$ and $y_2$ vs $y_1$ for $t \in [100, 200]$ with $\alpha = 0$, $\beta = 10$, $\delta = 0.22$, and $f = 0.3$. The approximations to $y_1(t = 100)$ given by the first, second, and fourth-order Runge-Kutte methods are presented in Table 1. below.

Table 1: Approximations to $y_1(t = 100)$ given by the first, second, and fourth-order Runge-Kutte methods

<table>
<thead>
<tr>
<th>Method</th>
<th>$y_1(t = 100)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ode4</code></td>
<td>2.8363077E-02</td>
</tr>
<tr>
<td><code>ode2</code></td>
<td>2.8361822E-02</td>
</tr>
<tr>
<td><code>ode1</code></td>
<td>2.7215973E-02</td>
</tr>
</tbody>
</table>
The value of $y_1(t = 100)$ is physically analogous to the position of the mass in a mass-spring-damper system described by the Duffing system of differential equations. The subroutines ode1, ode2, and ode4 give approximations—with increasing accuracy—to the position of such a mass at $t = 100$. The motion of the mass depends on the parameters $\alpha$, $\beta$, $\delta$, and $f$ which represent the values and different ratios of the spring constant, the damping power, and the magnitude of the external forcing.

3. Write a Fortran 90 function subroutine to evaluate the Taylor series approximation of $\cosh x$ about $x = 0$ with seven non-zero terms. Process this subroutine with the f2py Fortran to Python software. Demonstrate its execution within the Python environment as shown in Chapter 23.

The subroutine coshh evaluates the Taylor series expansion of $\cosh x$ about $x = 0$ with seven non-zero terms. The original source code is as follows.

```
coshh.f90
real function coshh(x)
  implicit none
  real, intent(in) :: x
  coshh = 1.+(x**2.)/2.+(x**4.)/24.+(x**6.)/720.
  +(x**8.)/40320.+x**10./3628800.+x**12./479001600.
end function coshh
```

After the module schrodinger was removed, the subroutine was processed into Python using the f2py Fortran to Python software, as shown in Chapter 23. Below are the Python operations executed to approximate $\cosh 3$.

```
[vsellner@remote205 ~]$ python
Python 2.6.6 (r266:84292, May 22 2015, 08:34:51)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-15)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import coshh
>>> print coshh.coshh(3)
10.067050186
```

The error between the seven-term Taylor series approximation to $\cosh(3)$ and the exact value is $5.6977e-05$. 

Figure 3: Plot of $y_2$ vs $t$ for $t \in [0, 200]$ and $y_2$ vs $y_1$ for $t \in [100, 200]$ with $\alpha = 0$, $\beta = 10$, $\delta = 0.22$, and $f = 0.3$; reproduction of Fig. 19.8.