Numerical Solution of Ordinary Differential Equations

Kurt Bryan and SIMIODE

Example 1: Many ODEs cannot be solved in an analytical or closed-form. For example, consider the ODE u'(t) = sin(u(t)) - t*u(t):

In[1]:=
```
de = u'[t] == Sin[u[t]] - t * u[t]
```

Mathematica can't provide an analytical solution, say with u(0) = 1:

In[2]:=
```
DSolve[{de, u[0] == 1}, u, t]
```

Yet the existence-uniqueness theorem applies and guarantees that a solution exists. In this case we can obtain a numerical approximation to the solution by using the ***NDSolve*** command. In this case we'll solve on the interval t = 0 to t = 5, as

In[15]:=
```
sol = NDSolve[{de, u[0] == 1}, u, {t, 0, 5}]
```

To evaluate the solution at a given time t, say t = 2.0, execute

In[4]:=
```
u[2] /. sol
```

Or we can define "usol[t]" as a function

In[12]:=
```
usol[t_] = u[t] /. sol
```

In[13]:=
```
usol[2]
```

To plot the solution evaluate

In[16]:=
```
Plot[Evaluate[u[t] /. sol], {t, 0, 5}, PlotRange → All]
```

Example 2: The various classical methods---Euler's method, improved Euler (RK2 here), and the RK4 method---can be implemented using the ***NDSolve*** command. In each case below the step size is 0.1.

In[17]:=
```
eulersol = NDSolve[{de, u[0] == 1}, u, {t, 0, 5},
  StartingStepSize → 0.1, Method → {"FixedStep", Method → "ExplicitEuler"}]
```

In[27]:=
```
rk2sol = NDSolve[{de, u[0] == 1}, u, {t, 0, 5},  StartingStepSize → 0.1,
  Method → {"FixedStep", Method → {"ExplicitRungeKutta", "DifferenceOrder" → 2}}]
```

In[22]:=
```
rk4sol = NDSolve[{de, u[0] == 1}, u, {t, 0, 5},  StartingStepSize → 0.1,
  Method → {"FixedStep", Method → {"ExplicitRungeKutta", "DifferenceOrder" → 4}}]
```

A quick comparison. The numerical solution "sol" is an adaptive step size method and probably the most accurate.

In[33]:=
```
u[5] /. sol (*Accurate Solution*)

u[5] /. eulersol (*Euler estimate *)

u[5] /. rk2sol (*Improved Euler*)
```

```
u[5] /. rk4sol (* RK4 Estimate *)
```