Computational Methods
CMSC/AMSC/MAPL 460

Ordinary differential equations

Ramani Duraiswami,
Dept. of Computer Science

Several slides adapted from Profs. Dianne O’Leary and Eric Sandt, TAMU
Ordinary differential equations

- Mathematical modeling involves posing models and then solving these models numerically or analytically.
- ODEs represent a powerful method of modeling, especially if things depend on rates of change.
- Rate of change of distance is velocity \( v = \frac{dx}{dt} \)
- Knowing the velocity as a function of \( \tau \) we can integrate using numerical quadrature.
- Rate of change of velocity is acceleration \( a = \frac{dv}{dt} = \frac{d^2x}{dt^2} \)
- Given initial conditions \( (v(0)=0, x(0)=0) \) find the location \( x(t) \) at time \( t \) given that the object falls with a constant acceleration of 10 m/s\(^2\).
Solution by simple integration

\[ \int dv = \int 10 dt \]
\[ \quad v = 10 t + c_1 \]
\[ \int dx/dt \ dt = \int 10t \ dt \]
\[ x = 10t^2/2 + c_1t + c_2 \]

- Use initial conditions
  - \( v(0) = 0 \) so \( c_1 \) is zero
  - \( x(0) = 0 \) so \( c_2 \) is zero
- Final solution \( x = 5t^2 \)
- Could handle more complex functions of \( t \) under the integral

\[ \frac{dx}{dt} = f(t) \quad x(t) = \int_0^t f(\tau) \ d\tau \]
What if simple integration would not work?

- Example: Let the velocity be a function of $x$ and $t$
- $\frac{dx}{dt} = f(x,t)$
- $x(t) = \int_0^t f(x(\tau), \tau) d\tau$
- Cannot be simply integrated
- This is the typical type of problem we need to solve in ODEs
- This is nonlinear because solution $x$ depends on itself
- The linear case could be solved using numerical quadrature
  - Previous chapter
Writing a 2nd order system in standard form

- ODEs in standard form: way they are input to software
- Written as a set of 1st order equations with initial conditions

\[
\begin{align*}
u'' &= g(t, u, u') \\
u(0) &= u_0 \\
u'(0) &= v_0
\end{align*}
\]

- where \( u_0 \) and \( v_0 \) are given.
- Let \( y_1 = u \) and \( y_2 = u' \). Then, in standard form:

\[
\begin{align*}
y_2' &= g(t, y_1, y_2) \\
y_1' &= y_2 \\
y_1(0) &= u_0 \\
y_2(0) &= v_0
\end{align*}
\]
Standard form

• We'll work with problems in *standard form*,
  
  \[ y' = f(t; y) \]
  
  \[ y(0) = y_0 \]

• where the function \( y \) has \( m \) components,

• \( y' \) means the derivative with respect to \( t \), and

• \( y_0 \) is a given vector of initial conditions (numbers).

• Writing this component-by-component yields
  
  \[ y'(1) = f_1(t, y(1) \ldots y(m)) \]
  
  ... 
  
  \[ y'(m) = f_m(t, y(1) \ldots y(m)) \]

with \( y(1)(t_0), \ldots, y(m)(t_0) \) given initial conditions
A modeling exercise: predator prey problems

- Eco-system (island) that contains rabbits and foxes
- Island has plenty of food for rabbits
- Rabbits reproduce like crazy and would fill-up the island
- Foxes eat rabbits
- Let $r(t)$ represent the number of rabbits and $f(t)$ the number of foxes.
- Model the number of rabbits and foxes on the island and decide if it will reach an equilibrium
Rabbit and fox population

- Rabbit population will grow at a certain rate
  - \( a \) is the natural growth rate of rabbits in the absence of predation,
- Rabbits will die as they are eaten by foxes. Let the rabbit die if it encounters a fox.
  - \( b \) is the death rate per encounter of rabbits due to predation,
- Fox population dies off if they cannot eat rabbits
  - \( c \) is the natural death rate of foxes in the absence of food (rabbits),
- Foxes reproduce if they have food
  - \( e \) is the efficiency of turning predated rabbits into foxes.
- Initial conditions \( R(0)=r_0 \) and \( F(0)=f_0 \)
- Volterra equations
  \[
  \frac{dR}{dt} = aR - bRF \\
  \frac{dF}{dt} = ebRF - cF
  \]
Standard form

- Volterra’s model

\[ \frac{dR}{dt} = 2R - \alpha RF \]

\[ \frac{dF}{dt} = \alpha RF - F \]

- Another example from the book

\[
\begin{align*}
\ddot{u}(t) &= -\frac{u(t)}{r(t)^3} \\
\ddot{v}(t) &= -\frac{v(t)}{r(t)^3}
\end{align*}
\]

where

\[ r(t) = \sqrt{u(t)^2 + v(t)^2} \]

The vector \( y(t) \) has four components,

\[ y(t) = \begin{bmatrix} u(t) \\ v(t) \\ \dot{u}(t) \\ \dot{v}(t) \end{bmatrix} \]

The differential equation is

\[
\dot{y}(t) = \begin{bmatrix} \dot{u}(t) \\ \dot{v}(t) \\ -\frac{u(t)}{r(t)^3} \\ -\frac{v(t)}{r(t)^3} \end{bmatrix}
\]

function ydot = twobody(t,y)

\[ r = \text{sqrt}(y(1)^2 + y(2)^2); \]

\[ \text{ydot} = [y(3); y(4); -y(1)/r^3; -y(2)/r^3]; \]
Solving differential equations: Euler’s method

• As in quadrature: use Taylor series

\[ y(t + h) = y(t) + hy'(t) + h^2/2 \, y''(\xi) \]

for some point \( \xi \) in \([t, t + h]\)

• Euler’s method

\[ y(t + h) = y(t) + hy'(t) + h^2/2 \, y''(\xi) \]

\[ y'(t) = f(t, y(t)) \]

• Note that

\[ y'(t) = f(t, y(t)) \]

• March forward

\[ y(h) = y_1 = y(0) + hy'(0) \]
\[ = y_0 + hf(t_0, y_0) \]
\[ y_{n+1} = y_n + hf_n \]
Example

Consider

\[
\frac{dy}{dx} = x + y
\]

The initial conditions is :

\[ y(0) = 1 \]

The analytical solution

\[ y(x) = 2e^x - x - 1 \]
Euler’s Method: First-order Taylor Method

\[ \frac{dy}{dx} = y' = f(x, y); \quad y(x_0) = y_0 \]

Straight line approximation
Euler’s Method Example

Consider

\[ \frac{dy}{dx} = x + y \]

The initial condition is: \( y(0) = 1 \)

The step size is: \( \Delta h = 0.02 \)

The analytical solution is:

\[ y(x) = 2e^x - x - 1 \]
Euler’s Method Example

The algorithm has a loop using the initial conditions and definition of the derivative

The derivative is calculated as:

\[ y'_i = x_i + y_i \]

The next y value is calculated:

\[ y_{i+1} = y_i + \Delta h \ y'_i \]

Take the next step:

\[ x_{i+1} = x_i + \Delta h \]
Euler’s Method Example

The results

<table>
<thead>
<tr>
<th>$x_n$</th>
<th>$y_n$</th>
<th>$y'_n$</th>
<th>$hy'_n$</th>
<th>Exact Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000</td>
<td>1.00000</td>
<td>0.02000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.02</td>
<td>1.02000</td>
<td>1.04000</td>
<td>0.02080</td>
<td>1.02040</td>
<td>-0.00040</td>
</tr>
<tr>
<td>0.04</td>
<td>1.04080</td>
<td>1.08080</td>
<td>0.02162</td>
<td>1.04162</td>
<td>-0.00082</td>
</tr>
<tr>
<td>0.06</td>
<td>1.06242</td>
<td>1.12242</td>
<td>0.02245</td>
<td>1.06367</td>
<td>-0.00126</td>
</tr>
<tr>
<td>0.08</td>
<td>1.08486</td>
<td>1.16486</td>
<td>0.02330</td>
<td>1.08657</td>
<td>-0.00171</td>
</tr>
<tr>
<td>0.1</td>
<td>1.10816</td>
<td>1.20816</td>
<td>0.02416</td>
<td>1.11034</td>
<td>-0.00218</td>
</tr>
<tr>
<td>0.12</td>
<td>1.13232</td>
<td>1.25232</td>
<td>0.02505</td>
<td>1.13499</td>
<td>-0.00267</td>
</tr>
<tr>
<td>0.14</td>
<td>1.15737</td>
<td>1.29737</td>
<td>0.02595</td>
<td>1.16055</td>
<td>-0.00318</td>
</tr>
<tr>
<td>0.16</td>
<td>1.18332</td>
<td>1.34332</td>
<td>0.02687</td>
<td>1.18702</td>
<td>-0.00370</td>
</tr>
<tr>
<td>0.18</td>
<td>1.21019</td>
<td>1.39019</td>
<td>0.02780</td>
<td>1.21443</td>
<td>-0.00425</td>
</tr>
<tr>
<td>0.2</td>
<td>1.23799</td>
<td>1.43799</td>
<td>0.02876</td>
<td>1.24281</td>
<td>-0.00482</td>
</tr>
</tbody>
</table>
Euler’s Method

The trouble with this method is

– Lack of accuracy
– Small step size needed for accuracy