Numerical Calculus and Differential Equations

Trevor Spiteri
trevor.spiteri@um.edu.mt
http://staff.um.edu.mt/trevor.spiteri

Department of Communications and Computer Engineering
Faculty of Information and Communication Technology
University of Malta

7 April, 2008

Outline

Outline

The sum of a sequence

- The `sum` function can add all the elements of a vector.
- For example, the sum of $\begin{bmatrix} 1 & 3 & 0 & 10 \end{bmatrix}$ can be obtained as follows:

```
>> x = [1 3 0 10];
>> s = sum(x);
```

- This gives us `s` = 14.
- When the parameter to `sum` is a matrix, each column is treated independently.

The cumulative sum of a sequence

- The `cumsum` function returns the cumulative sum of a vector.
- For example, the cumulative sum of $\begin{bmatrix} 1 & 3 & 0 & 10 \end{bmatrix}$ can be obtained as follows:
```
>> x = [1 3 0 10];
>> cs = cumsum(x);
```
- This gives us `cs` $= \begin{bmatrix} 1 & 4 & 4 & 14 \end{bmatrix}$.
- When the parameter to `cumsum` is a matrix, each column is treated independently.

The difference of a sequence

- The `diff` function finds the difference between numbers next to each other in a vector.
- For example, the difference of $\begin{bmatrix}1 & 4 & 9 & 16 & 25\end{bmatrix}$ can be obtained as follows:
```
>> x = [1 4 9 16 25];
>> d = diff(x);
```
- This gives us d $= \begin{bmatrix}3 & 5 & 7 & 9\end{bmatrix}$.
- Passing the result to `d` again gives us:
```
>> d2 = diff(d);
```
- This gives us d2 $= \begin{bmatrix}2 & 2 & 2\end{bmatrix}$.
- These two steps can be combined using:
```
>> d2 = diff(x, 2);
```
- When the first parameter to `diff` is a matrix, each column is treated independently.

The product of a sequence

- The `prod` function can multiply all the elements of a vector.
- For example, the product of $\begin{bmatrix}1 & 3 & 0.5 & 10\end{bmatrix}$ can be obtained as follows:

```
>> x = [1 3 0.5 10];
>> p = prod(x);
```

- This gives us $\mathtt{p} = 15$.
- When the parameter to `prod` is a matrix, each column is treated independently.

The cumulative product of a sequence

- To `cumprod` function returns the cumulative product of a vector.
- For example, the cumulative product of $\begin{bmatrix} 1 & 3 & 0.5 & 10 \end{bmatrix}$ can be obtained as follows:

```
>> x = [1 3 0.5 10];
>> cp = cumprod(x);
```

- This gives us `cp` $= \begin{bmatrix} 1 & 3 & 1.5 & 15 \end{bmatrix}$.
- When the parameter to `cumprod` is a matrix, each column is treated independently.

Outline

Differentiating vectors

- Let $\mathbf{y} = \begin{bmatrix} 1 & 2 & 4 & 8 & 10 \end{bmatrix}$.
- To differentiate, use the `gradient` function.
- If the x-spacing is 1, we type:

```
>> dy = gradient(y);
```

- This gives us $\mathbf{g} = \begin{bmatrix} 1 & 1.5 & 3 & 3 & 2 \end{bmatrix}$.
- Note that:

$$
\begin{aligned}
g_1 &= (y_2 - y_1)/(x_2 - x_1) = (y_2 - y_1)/1 \\
g_2 &= (y_3 - y_1)/(x_3 - x_1) = (y_3 - y_1)/2 \\
g_3 &= (y_4 - y_2)/(x_4 - x_2) = (y_4 - y_2)/2 \\
g_4 &= (y_5 - y_3)/(x_5 - x_3) = (y_5 - y_3)/2 \\
g_5 &= (y_5 - y_4)/(x_5 - x_4) = (y_5 - y_4)/1
\end{aligned}
$$

Differentiating vectors with non-unit spacing

- Let $\mathbf{y} = \begin{bmatrix} 1 & 2 & 4 & 8 & 10 \end{bmatrix}$.
- Suppose that the x-spacing between each value of y is 0.5.
- For this non-unit spacing, we type:

```
>> dy = gradient(y, 0.5);
```

- This gives us $\mathbf{g} = \begin{bmatrix} 2 & 3 & 6 & 6 & 4 \end{bmatrix}$.
- Note that:

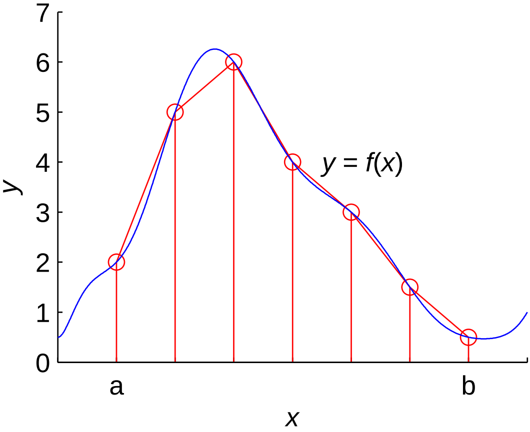
$$\begin{aligned}
g_1 &= (y_2 - y_1)/(x_2 - x_1) = (y_2 - y_1)/0.5 \\
g_2 &= (y_3 - y_1)/(x_3 - x_1) = (y_3 - y_1)/1 \\
g_3 &= (y_4 - y_2)/(x_4 - x_2) = (y_4 - y_2)/1 \\
g_4 &= (y_5 - y_3)/(x_5 - x_3) = (y_5 - y_3)/1 \\
g_5 &= (y_5 - y_4)/(x_5 - x_4) = (y_5 - y_4)/0.5
\end{aligned}$$

Differentiating polynomials

- The `polyder` function differentiates polynomials.
- For example, to differentiate $3x^2 - 2x + 5$, type:
```
>> p = [3 -2 5];
>> pd = polyder(p);
```
- The resulting polynomial is $\begin{bmatrix} 6 & -2 \end{bmatrix}$, that is, $6x - 2$.
- `polyder` can also differentiate polynomial fractions.
- To differentiate $\dfrac{x-2}{x^2+2}$, type:
```
>> num = [1 -2]; den = [1 0 2];
>> [numa, dena] = polyder(num, den);
```
- The result is $\texttt{numa} = \begin{bmatrix} -1 & 4 & 2 \end{bmatrix}$ and $\texttt{dena} = \begin{bmatrix} 1 & 0 & 4 & 0 & 4 \end{bmatrix}$, that is, $\dfrac{-x^2 + 4x + 2}{x^4 + 4x^2 + 4}$.

Trapezoidal integration

- $\sum \text{Area of trapezium} \approx \int_a^b f(x)$



The trapz function

- The `trapz` function computes an approximation of the integral using the trapezoidal method.
- `trapz` operates on data points, not on functions.
- If we have two vectors, **x** and **y**, we can write:

```
>> area = trapz(x, y);
```

Integration of a function

- When we have a function instead of data points, we do not use trapezoidal integration.
- The `quad` function uses an adaptive Simpson's rule.
- Let us compute the integral
$$\int_0^{\pi} \sin(x)\, dx$$
- This can be done using:
```
>> a = quad(@sin, 0, pi);
```
- This gives us $a = 2$.

Integrating polynomials

- The `polyint` function integrates polynomials.
- Unlinke `polyder`, `polyint` does not work on polynomial fractions.
- To integrate the polynomial $3x^2 - x + 1$, type:
  ```
  >> p = [3 -1 1];
  >> pint = polyint(p);
  ```
- The resulting polynomial is $\begin{bmatrix}1 & -0.5 & 1 & 0\end{bmatrix}$, that is, $x^3 - 0.5x^2 + x$.
- The constant of integration was assumed to be zero.
- To use 5 as the constant of integration, type:
  ```
  >> pint2 = polyint(p, 5);
  ```
- The new result is $\begin{bmatrix}1 & -0.5 & 1 & 5\end{bmatrix}$, that is, $x^3 - 0.5x^2 + x + 5$.

Outline

Ordinary differential equations

- An ordinary differential equation (ODE) contains functions of only one independent variable.
- An ODE can contain one or more derivatives of the same variable.
- Examples of ODEs:

$$\frac{dy(x)}{dx} = y(x)$$
$$\frac{d^2y(x)}{dx^2} + 3\frac{dy(x)}{dx} = \sin y(x) + x$$

- The following equation is not an ODEs, because it has two independent variables, y and z.

$$\frac{dy(x)}{dx} = y(x) + z(x)$$

- Note that $\dfrac{dy(x)}{dx}$ can be written as y', $\dfrac{d^2y(x)}{dx^2}$ as y'', and so on.

The derivative function

- A first order ODE is an ODE containing y' but no y'' or higher order derivatives.
- To solve such an ODE in MATLAB, first make the derivative the subject of the formula.

$$\begin{aligned} y' + y &= \cos x \\ y' &= \cos x - y \end{aligned}$$

- Then write a MATLAB function that computes this derivative.

yder.m

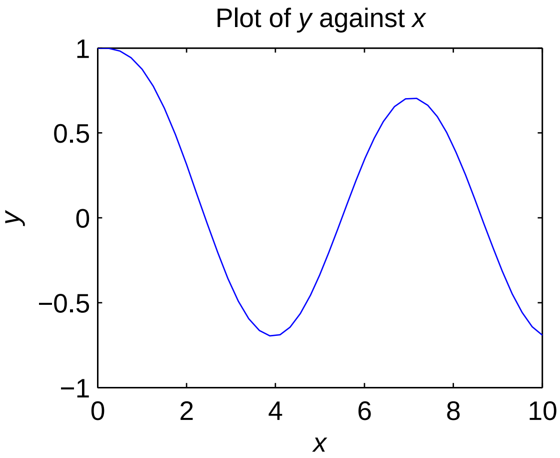
```
function yd = yder(x, y)
yd = cos(x) - y;
```

- Note that the first parameter is the controlled variable, x.
- This must be included even if it is not used to calculate y'.

Solving the ODE

- Once the `yder` function is ready, we can type:
  ```
  >> xspan = [0 10];
  >> y0 = 1;
  >> [xout yout] = ode45(@yder, xspan, y0);
  ```
- `xspan` contains the range of the integration.
- `y0` is the initial value of y.
- `yout` can be plotted agains `xout`.

The result



Different ODE solvers

- All the solvers have the same syntax as `ode45`.

Table : MATLAB ODE solvers

`ode45`	Nonstiff, medium order. Used most of the time.
`ode23`	Nonstiff, low order. Used for problems with crude error tolerances.
`ode113`	Nonstiff, variable order. Used for stringent error tolerances, or for computationally intensive problems.
`ode15s`	Stiff, variable order. Used if `ode45` is slow because the problem is stiff.
`ode23s`	Stiff, low order.
`ode23t`	Moderately stiff, low order.
`ode23tb`	Stiff, low order.

Changing higher-order ODEs to state-variable form

- To solve an ODE of order 2 or higher, the equation must first be written as a set of first-order equations.
- Consider the second-order equation
$$y'' + 3y' + 5y = \cos(10t)$$
- First, solve it for the highest derivative:
$$y'' = \cos(10t) - 5y - 3y'$$
- Next, define the variables $x_1 = y$ and $x_2 = y'$. Thus,
$$x_1' = x_2$$
$$x_2' = \cos(10t) - 5x_1 - 3x_2$$
- This form is sometimes called the state-variable form.

The derivative function for the state-variable form

- The equations are in state-variable form.

$$x_1' = x_2$$
$$x_2' = \cos(10t) - 5x_1 - 3x_2$$

- Next, write a function that computes the derivative.

xder.m

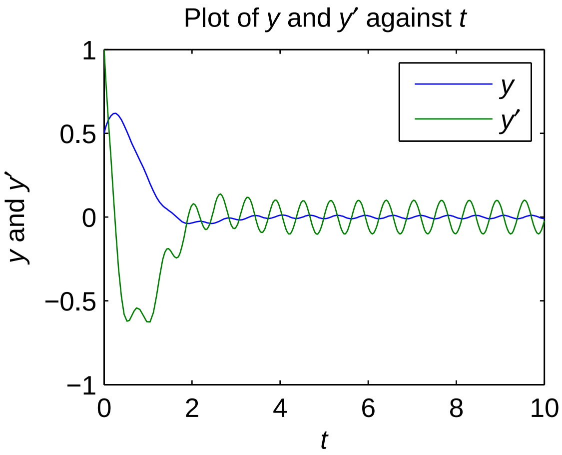
```
function xd = xder(t, x)
xd = zeros(2, 1);
xd(1) = x(2);
xd(2) = cos(10 * t) - 5 * x(1) - 3 * x(2);
```

- The return vector must be a column vector.

Solving the ODE

- Once the `xder` function is ready, we can type:
```
>> tspan = [0 10];
>> x0 = [0.5; 1];
>> [tout xout] = ode45(@xder, tspan, x0);
```
- `tspan` contains the range of the integration.
- `x0` is a column vector containing the initial values of $\mathbf{x}$.
- `tout` is a column vector containing the time instants of the result.
- `xout` has two columns.
- `xout(:, 1)` contains values of x_1, that is, values of y.
- `xout(:, 2)` contains values of x_2, that is, values of y'.
- To plot y against t, we can type:
```
>> plot(tout, xout(:, 1))
```

The result



Matrix method for state-variable form

- Suppose we have the ODE
$$50y'' + 3y' + 7y = f(t)$$
- In state-variable form with $x_1 = y$ and $x_2 = y'$, this becomes
$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -\tfrac{7}{50}x_1 - \tfrac{3}{50}x_2 + \tfrac{1}{50}f(t) \end{aligned}$$
- These equations can be written as:
$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{7}{50} & -\frac{3}{50} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{50} \end{bmatrix} f(t)$$
- In compact form that is
$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{B}f(t)$$
where $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{7}{50} & -\frac{3}{50} \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{50} \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$.

The derivative function for the matrix method

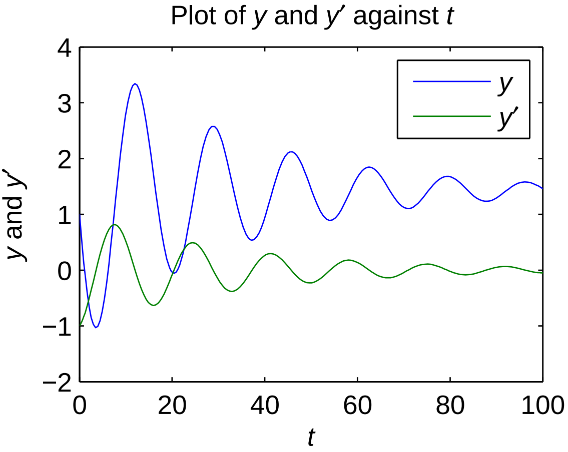
- Once the ODE is in matrix form, we write a function that computes the derivative.
- In our example, take $f(t) = 10$.

xder.m

```
function xd = xder(t, x)
A = [0 1; -7/50 -3/50];
B = [0; 1/50];
f = 10;
xd = A * x + B * f;
```

- To solve for $y \in [0, 100]$ with $y_0 = 1$ and $y'_0 = -1$, type:
  `>> [t, x] = ode45(@xder, [0 100], [1; -1]);`

The result



The characteristic roots

- The plot in the previous slide was decaying and oscillating.
- This behaviour depends on the characteristic roots of the equation.
- In state-variable form, the characteristic roots are the eigenvalues of the matrix $\mathbf{A}$.
- These can be obtained using:

```
>> A = [0 1; -7/50 -3/50];
>> r = eig(A);
```

- In this case, the roots are $-0.03 \pm 0.3730i$.
- The real part indicates the rate at which the function reaches steady-state response, $e^{-0.03t}$.
- The imaginary part indicates the frequency of the oscillations, $\cos(0.3730t + \phi)$.