NUMERICAL METHODS FOR SCIENTIFIC COMPUTING

Antonella Falini

INTRODUCTION to MATLAB

Part IV

L.M. Geo-Engineering, year 2018/2019

Outline

- Anonymous Functions
- Newton Interpolating polynomial
- Lagrange Interpolating polynomial
- Piecewise linear interpolation
- Interpolation with Matlab

Anonymous Functions

- *Anonymous functions* allow you to create a simple function without creating an M-file.
- Syntax:

  `fhandle = @(arglist) expression ;`
- `fhandle` function handle; it can be used to invoke the function;
- `(arglist)` a list of input arguments separated by commas;
- `expression` any single valid MATLAB expression.

Example

- Define the following function as anonymous in Matlab: $f(x, y) = x^2 + y^2$.
- We need to use the command `f=@(x,y) x.^2 + y.^2`
- Evaluate $f(2, 3)$
- In the command window simply type `f(2,3)`

NEWTON INTERPOLATING POLYNOMIAL

Example

- We want to estimate the natural logarithm of 2 using linear interpolation.
- The given pairs are:

x	1	6
$f(x)$	$\log(1)$	$\log(6)$

- The *Newton linear-interpolation formula*,

$$N(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

- Compute the percentage relative error:

$$\frac{\log(2) - N(2)}{\log(2)} \times 100$$

NEWTON INTERPOLATING POLYNOMIAL

Example

- We want to estimate the natural logarithm of 2 using a quadratic Newton polynomial.
- The given pairs are:

x	1	4	6
$f(x)$	$\log(1)$	$\log(4)$	$\log(6)$

- The *Newton quadratic-interpolation formula*,

$$N(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2).$$

with, $b_1 = f(x_1)$, $b_2 = \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}$ and $b_3 = \dfrac{\frac{f(x_3)-f(x_2)}{x_3-x_2} - \frac{f(x_2)-f(x_1)}{x_2-x_1}}{x_3 - x_1}$.

- Compute the percentage relative error:

LAGRANGE INTERPOLATING POLYNOMIAL

Example

- Use the linear Lagrange polynomial to interpolate the values:

x	0	20
$f(x)$	3.85	0.800

- *Linear Lagrange interpolating polynomial*,

$$f_1(x) = \frac{x - x_2}{x_1 - x_2} f(x_1) + \frac{x - x_1}{x_1 - x_2} f(x_2)$$

LAGRANGE INTERPOLATING POLYNOMIAL

- In general the Lagrange polynomial of degree $n-1$ which interpolates n pairs is computed by,

$$f_{n-1}(x) = \sum_{i=1}^{n} L_i(x) f(x_i)$$

with

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Example

- Use quadratic Lagrange polynomial to interpolate the pairs,

x	0	20	40
$f(x)$	3.85	0.800	0.212

- Plot the input data and the resulting Lagrange polynomial.

PIECEWISE LINEAR INTERPOLATION

- Given n data points, for every interval i the linear spline s_i is constructed as:

$$s_i(x) = a_i + b_i(x - x_i),$$

where $x_i, i = 1 \ldots n$ are given points and

$$\begin{aligned} a_i &= f_i \equiv f(x_i) \\ b_i &= \frac{f_{i+1} - f_i}{x_{i+1} - x_i} \end{aligned}$$

Example

- Given the following data, construct a linear interpolating spline:

x	3.0	4.5	7.0	9.0
f	2.5	1.0	2.5	0.5

- Plot the points and the resulting spline.

Interpolation with MatLab

- The built-in function `interp1` provides a number of different types of piecewise one-dimensional interpolation.

  `si = interp1(x, f, xi, 'method')`
- x and f are the vectors containing values that has to be interpolated,
- *si* vector containing the results of the interpolation, evaluated at the points in the vector *xi*
- 'method':
  - `'nearest'` : piecewise constant.
  - `'linear'`: the method uses straight lines to connect the points
  - `'spline'`: piecewise cubic spline interpolation with not-a-knot end conditions.

Example

- Use `interp1` to interpolate the following values, changing the used method and plot the results.

t	0	20	40	56	68	80	84	96	104	110
v	0	20	20	38	80	80	100	100	125	125

Interpolation with MatLab

- To perform interpolation in 2D setting we can use the built-in function `interp2`

  `zi= interp2(x, y, z, xi, yi, 'method')`
- x and y are the matrices containing the coordinates of the points at which the values in the matrix z are given.
- zi: a matrix containing the results of the interpolation evaluated at the points in the matrices xi and yi.
- 'method' is the desired method : linear, nearest, spline.

Interpolation with MatLab

An application

- In order to resample the pixels in an image, to increase the resolution or change the size, or to smooth out the pixels after zooming, an interpolation process may be applied.

Example

- Let us load the following image:

```
A = imread('ngc6543a.jpg');

imshow(A)
```

- We need to create some grid points. To this end we are going to use the pixels of the image, but first we need to convert them to double precision

```
F = griddedInterpolant(double(A));
```

Interpolation with MatLab

An application

Example

- As you zoom in on an image, the pixels in the region of interest become larger and larger and detail in the image is quickly lost. You can use image resampling to smooth out these zooming artifacts.

```
imshow(A(1:570,10:600,:),'InitialMagnification','fit')

zoom(10)

title('Original Image, 10x Zoom')
```

- To create query points we need to create grid vectors,

```
[sx,sy,sz] = size(A);
xq = (1:0.1:sx)';
yq = (1:0.1:sy)';
zq = (1:sz)';
```

Interpolation with MatLab

An application

Example

- Query the interpolant F to reproduce this zoomed image (approximately) with 10x higher resolution. Compare the results from several different interpolation methods.

```
figure

F.Method = 'linear';

vq = uint8(F({xq,yq,zq}));

imshow(vq(1:5700,150:5900,:),'InitialMagnification','fit')

zoom(10)
title('Linear method')
```

Interpolation with MatLab

An application

Example

- Query the interpolant F to reproduce this zoomed image (approximately) with 10x higher resolution. Compare the results from several different interpolation methods.

```
figure

F.Method = 'cubic';

vq = uint8(F({xq,yq,zq}));

imshow(vq(1:5700,150:5900,:),'InitialMagnification','fit')

zoom(10)
title('Cubic method')
```

Interpolation with MatLab

An application

Example

- Query the interpolant F to reproduce this zoomed image (approximately) with 10x higher resolution. Compare the results from several different interpolation methods.

```
figure

F.Method = 'spline';

vq = uint8(F({xq,yq,zq}));

imshow(vq(1:5700,150:5900,:),'InitialMagnification','fit')

zoom(10)
title('Spline method')
```