Numerical Analysis, FMN011

Carmen Arévalo

Lund University

carmen@maths.lth.se

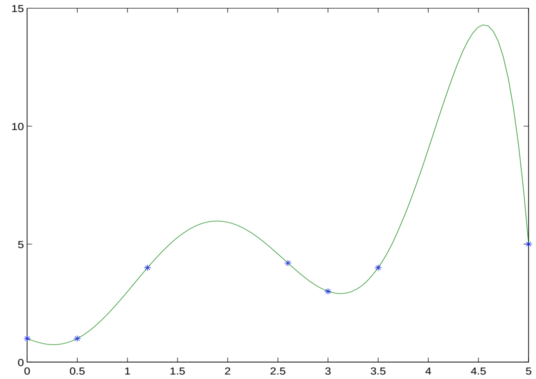
Piecewise interpolation, Splines

Single vs. Piecewise interpolation polynomials

Interpolate the following 7 points.

x_i	0.0	0.5	1.2	2.6	3.0	3.5	5
y_i	1	1	4	4.2	3	4	5

Interpolation with $p \in \Pi_6$



When the number of data points is large, it is often unsuitable to use polynomials of high degree

Linear piecewise polynomials

Instead of a single interpolating polynomial, we can use several low degree polynomials over subintervals.

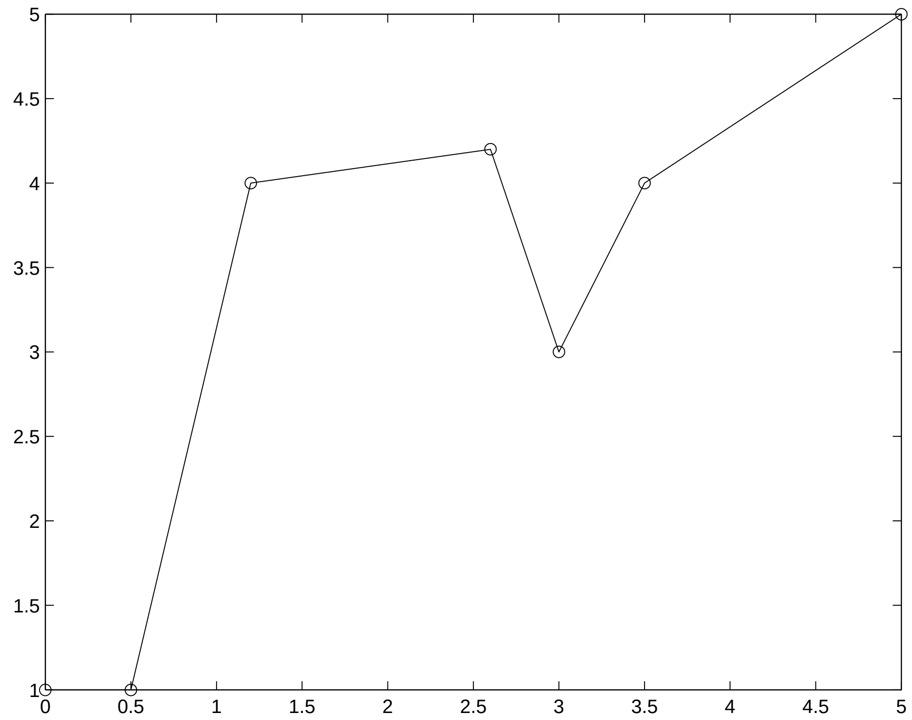
Here a different linear polynomial is used in each $[x_{i-1}, x_i]$

$$S(x) = \begin{cases} S_1(x) & = & a_1 + b_1(x - 0.0) & x \in [0.0, 0.5] \\ S_2(x) & = & a_2 + b_2(x - 0.5) & x \in [0.5, 1.2] \\ & \vdots & & \vdots \\ S_6(x) & = & a_6 + b_6(x - 3.5) & x \in [3.5, 5.0] \end{cases}$$

The coefficients a_j and b_j are computed from the interpolation conditions:

$$S_j(x_{j-1}) = y_{j-1}; \quad S_j(x_j) = y_j$$

Interpolation with a piecewise linear polynomial (6 pieces)



Splines

Polynomial pieces joined together with certain smoothness conditions.

- S is a piecewise polynomial of degree m:

$$S(x) = \begin{cases} S_1(x) & x \in [x_1, x_2] \\ S_2(x) & x \in [x_2, x_3] \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [x_{n-1}, x_n] \end{cases}$$

  The points x_i are the knots. The functions S_i are polynomials of degree m.

- S has $m-1$ continuous derivatives.

Cubic splines

Data points: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

$$
\begin{aligned}
S_1(x) &= a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3, \\
&\qquad\qquad\qquad \text{for } x \in [x_1, x_2] \\
S_2(x) &= a_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3, \\
&\qquad\qquad\qquad \text{for } x \in [x_2, x_3] \\
&\vdots \\
S_{n-1}(x) &= a_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 \\
&\quad +d_1(x - x_{n-1})^3, \qquad \text{for } x \in [x_{n-1}, x_n]
\end{aligned}
$$

Properties of cubic splines

- $S \in \Pi_3$ on each $[x_i, x_{i+1}], \quad i = 1, \ldots, n-1$
- $S(x_k) = y_k, \quad i = 1, \ldots, n$
- S, S' and S'' continuous on $[a, b]$.

Existence of cubic splines

Interpolation conditions:

$S_i(x_i) = y_i, \qquad (i = 1, \ldots, n-1)$

$S_{n-1}(x_n) = y_n$

Continuity conditions:

$S_i(x_{i+1}) = S_{i+1}(x_{i+1})$

$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$

$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \qquad (i = 1, \ldots, n-2)$

Total number of conditions to be fulfilled: $4n - 6$

Constructing a spline means finding the coefficients $a_i, b_i, c_i, d_i \quad (i = 1, \ldots, n-1)$

Total number of coefficients to be determined:

$4n - 4 \Rightarrow$ 2 free conditions or degrees of freedom.

Some types of Cubic Splines

Natural cubic spline:
$S''(x_1) = 0$ and $S''(x_n) = 0$

Clamped cubic spline:
$S'(x_1) = v_1$ and $S'(x_n) = v_n$

Curvature-adjusted cubic spline:
$S''(x_1) = v_1$ and $S''(x_n) = v_n$

Not-a-knot cubic spline:
$S_1'''(x_2) = S_2'''(x_2)$ and $S_{n-2}'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$
(this is MATLAB's default when using the `spline` command)

Determining the a_i coefficients of a cubic spline

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad x \in [x_i, x_{i+1}], \quad (i = 1 : n-1)$$

From the interpolation conditions

$$S_i(x_i) = y_i, \qquad (i = 1, \ldots, n-1),$$

$$a_i = y_i, \qquad (i = 1, \ldots, n-1)$$

Determining b_i and d_i

Define: $h_i = x_{i+1} - x_i$ and $c_n = S''_{n-1}(x_n)$

$$S''_i(x) = 2c_i + 6d_i(x - x_i)$$

$$S''_i(x_{i+1}) = S''_{i+1}(x_{i+1}) \Rightarrow 2c_i + 6d_ih_i = c_{i+1} \Rightarrow d_i = \frac{c_{i+1} - c_i}{3h_i}$$

$$y_{i+1} = y_i + b_ih_i + c_ih_i^2 + d_ih_i^3 \Rightarrow b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{3}(c_{i+1} + 2c_i)$$

Determining c_i for the natural spline

$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}) \Rightarrow$

$$\begin{aligned}
h_1c_1 + 2(h_1+h_2)c_2 + h_2c_3 &= 3\left(\frac{\Delta_2}{h_2} - \frac{\Delta_1}{h_1}\right) \\
&\vdots \\
h_{n-2}c_{n-2} + 2(h_{n-2}+h_{n-1})c_{n-1} + h_{n-1}c_n &= 3\left(\frac{\Delta_{n-1}}{h_{n-1}} - \frac{\Delta_{n-2}}{h_{n-2}}\right)
\end{aligned}$$

where $\Delta_i = y_{i+1} - y_i$

For the natural spline:

$$S_1''(x_1) = S_{n-1}''(x_n) = 0 \Rightarrow c_1 = 0, c_n = 0$$

Matrix form of equations for c_i

$$
\begin{bmatrix}
1 & 0 & 0 & & & \\
h_1 & u_2 & h_2 & & & \\
 & \ddots & \ddots & \ddots & & \\
 & & h_{n-2} & u_{n-2} & h_{n-1} \\
 & & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n
\end{bmatrix}
=
\begin{bmatrix}
0 \\ w_2 \\ \vdots \\ w_{n-1} \\ 0
\end{bmatrix}
$$

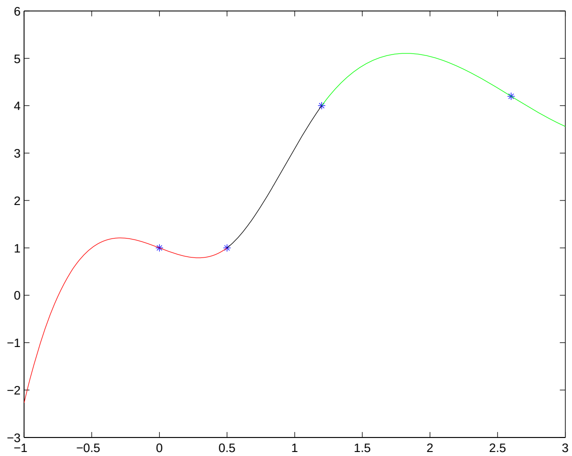
where $u_i = 2(h_i + h_{i-1})$ and $w_i = 3\left(\frac{\Delta_i}{h_i} - \frac{\Delta_{i-1}}{h_{i-1}}\right)$.

The matrix is strictly diagonally dominant, so the system has a unique solution.

Example

Use cubic splines to represent the following data:

x_i	0.0	0.5	1.2	2.6
y_i	1	1	4	4.2



Of all twice differentiable functions that interpolate a given set of data points, the cubic spline has less wiggle!

End Conditions

The top and bottom rows of the system must be replaced

Curvature-adjusted spline: $2c_1 = v_1, 2c_n = v_n$

$$\left[\begin{array}{cccccccc|c} 2 & 0 & 0 & 0 & \dots & \dots & 0 & 0 & v_1 \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & 2 & v_2 \end{array}\right]$$

Clamped spline: $S'(x_1) = v_1$ and $S'(x_n) = v_n$

$$\left[\begin{array}{cccccc|c} 2h_1 & h_1 & 0 & \dots & 0 & 0 & 3(\Delta_1/h_1 - v_1) \\ 0 & 0 & 0 & \dots & h_{n-1} & 2h_{n-1} & 3(v_n - \Delta_{n-1}/h_{n-1} \end{array}\right]$$

Not-a-knot spline: $d_1 = d_2$ and $d_{n-2} = d_{n-1}$

$$\left[\begin{array}{cccccccc|c} h_2 & -(h_1+h_2) & h_1 & \dots & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & \dots & h_{n-1} & -(h_{n-2}+h_{n-1}) & h_{n-2} & 0 \end{array}\right]$$

Splines in MATLAB

If we use the commands

```
> sp=spline(x,y) % this creates the spline structure
> [breaks,coeff,L,K]=unmkpp(sp) % "unmakes" (shows) structure
```

we obtain the following structure in MATLAB

```
    >> sp=spline(x,y)
sp =
      form: 'pp'
    breaks: [1x10 double]
     coefs: [9x4 double]
    pieces: 9
     order: 4
       dim: 1
```

This says that we have 9 cubic polynomials, and we can get their coefficients:

```
>> [breaks,coeff,L,K]=unmkpp(sp)
breaks =
  Columns 1 through 4
        0.97           1.12           2.92           3.00
  Columns 5 through 8
        3.33           3.97           6.10           8.39
  Columns 9 through 10
        8.56           9.44
coeff =
       16.67         -27.30         -11.10           2.58
       16.67         -19.99         -18.02           0.43
    -1276.96          69.75          71.30           0.06
      287.09        -249.36          56.34           5.74
      -23.26          38.48         -14.13           7.44
```

```
          1.33        -6.27         6.53         8.07
         -0.87         2.19        -2.13         6.38
         12.21        -3.81        -5.83         2.51
         12.21         2.60        -6.04         1.44
L =
     9
K =
     4
```

Here we read that the sixth cubic polynomial in our spline is `coeff(6,:)`:

$$1.33(x-3.97)^3 - 6.27(x-3.97)^2 + 6.53(x-3.97) + 8.07$$

Example: solution with a single interpolating polynomial

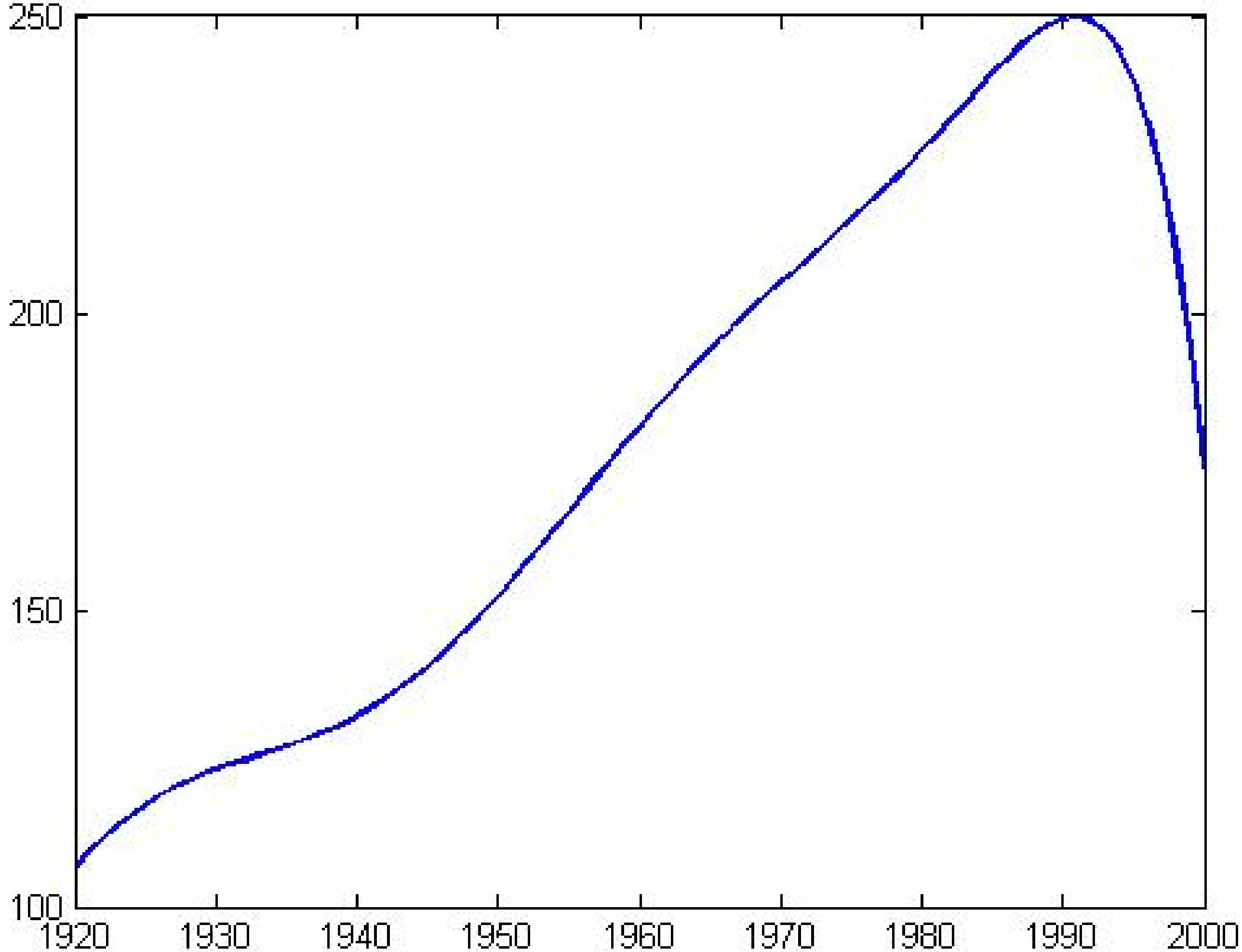
U.S.A. population (millions)

1920	1930	1940	1950	1960	1970	1980	1990
106.5	123.1	132.1	152.3	180.7	205.0	227.2	249.5

Let's solve the extrapolation problem to estimate the U.S.A. population in 2000:

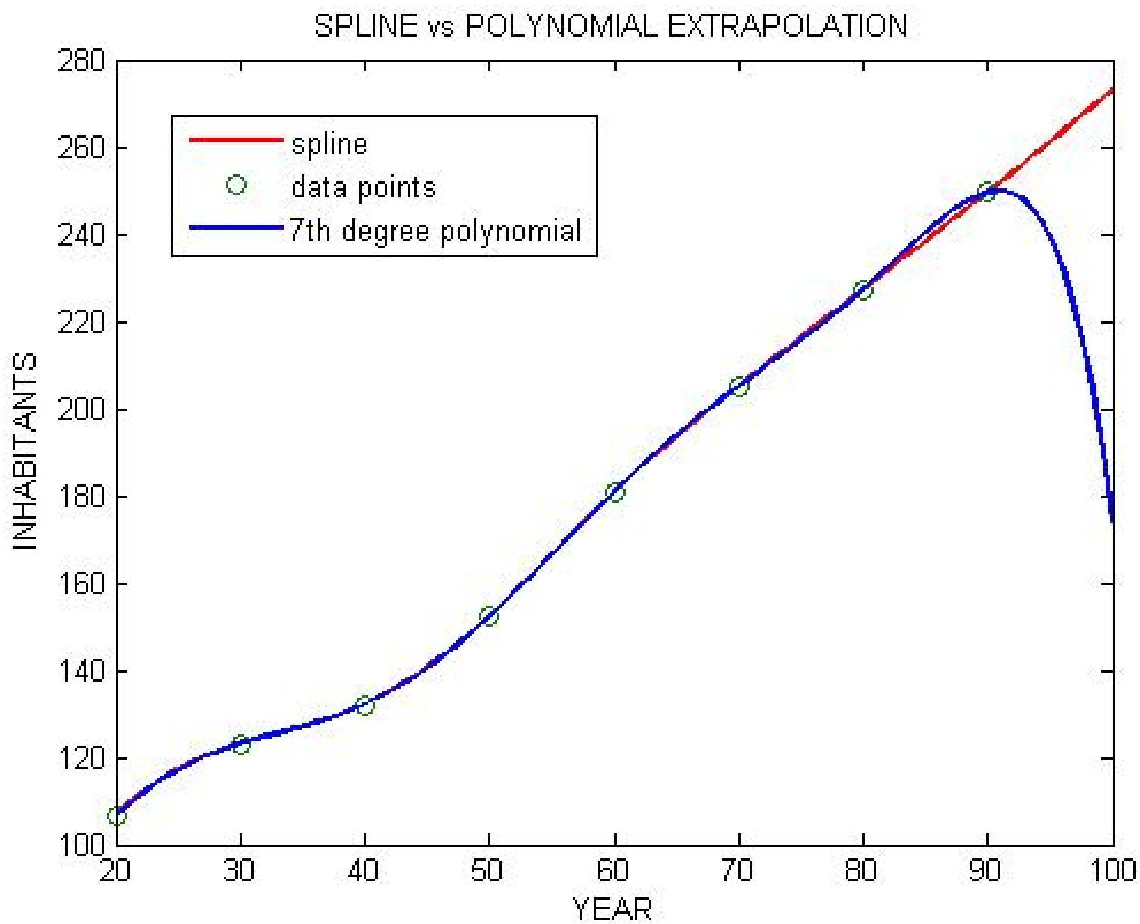
Interpolate with a 7th degree polynomial: $p(2000) = 173.7$

Seventh order polynomial gives wrong prediction (**extrapolation**) for 2000!!!


250
200
150
100
1920
1930
1940
1950
1960
1970
1980
1990
2000

Solution with cubic spline interpolation

```
>> x=20:10:90;
>> y=[106.5 123.1 132.1 152.3 ...
       180.7 205 227.2 249.5];
>> pp=interp1(x,y,100,'spline')
pp =
       273.05
```


SPLINE vs POLYNOMIAL EXTRAPOLATION
spline
data points
7th degree polynomial
INHABITANTS
YEAR
280
260
240
220
200
180
160
140
120
100
20
30
40
50
60
70
80
90
100