Computational Methods and Modelling

Antonio Attili & Edward McCarthy

antonio.attili@ed.ac.uk
ed.mccarthy@ed.ac.uk

School of Engineering
University of Edinburgh
United Kingdom

Solution tutorial 5
Fitting and interpolation

Exercise 1: Linear regression

- The problem can be solved applyting the formula:

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \tag{1}$$

$$a_0 = \bar{y} - a_1\bar{x} \tag{2}$$

where $\bar{y} = \left(\sum_{i=1}^{n} y_i\right)/n$ and $\bar{x} = \left(\sum_{i=1}^{n} x_i\right)/n$ are the means of y and x, respectively.

- With this formula we compute the coefficient a_0 and a_1 of the line:

$$y = a_0 + a_1 x \tag{3}$$

Exercise 2: Spline interpolation

- The following code can be used to generate the spline. Note that also a linear interpolation is shoown in the code.

''

```
# code for import of data and module not included here!!!
##############
# array containing the points where we want to evaluate the
# intepolation
x_int = np.linspace(0,1,num=64)

# generate linear interpolant
f_lin = interpolate.interp1d(x, y, kind='linear')
# evaluate linear interpolan at the desiderd points
y_int_lin = f_lin(x_int)

# generate spline interpolant
f_spline  = interpolate.splrep(x, y, s=0)
# evaluate spline interpolan at the desiderd points
y_int_spline = interpolate.splev(x_int, f_spline, der=0)

# plot results
plt.figure()
plt.plot(x,y,'gh',ms=10)
plt.plot(x_int,y_int_lin,'r.',x_int,y_int_spline,'b.')
plt.xlabel('x')
plt.ylabel('y')
# plot a zoom
plt.figure()
plt.plot(x,y,'gh',ms=10)
plt.plot(x_int,y_int_lin,'r.',x_int,y_int_spline,'b.')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(0.05,0.3)
plt.ylim(0.5,1.5)
plt.show()
```