Manual for the `polyfit` function

Javier I. Carrero (jicarrerom@unal.edu.co)

December 14, 2009

1 General description

Given a set of m (x_i, y_i) data points `polyfit` finds the n coefficients for

$$y = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n \tag{1}$$

that best fit $y(x)$ in the sense of minimizing the sum of the residuals $\left(y_i - y_i^{\text{calc}}\right)^2$ where y_i^{calc} represents the value calculated with eq. 1. Polynomial degree n is set as an input argument of the function, main output of `polyfit` has two possible forms, the default one is a Scilab polynomial representing eq. 1, but if the `Mcomp` input option is used `polyfit` returns the coefficients in Matlab's style, as a vector `[an .. a1 a0]`.

2 Function arguments

The standard syntax is

```
[paj, ycalc, statpar] = polyfit(x, y, Mcomp, GraphChk, pChar)
```

Input arguments (given `m` sets of data and a polynomial of degree `n`):

- `x` is a `m` component vector (column or row), each element in it represents a value of the independent variable.
- `y` is a `m` component vector (column or row), each element in it represents a value of the dependent variable.
- `Mcomp` (optional) if present `Mcomp` produces an output `paj` in the Matlab's style, i.e. a vector `[an .. a1 a0]`. To invoke this option write the optional argument as `Mcomp = 'Y'` .
- `GraphChk` (optional) if present produces a graphic in which the calculated values of `y` are plotted against the perfect fit, i.e. against the `ycalc=y` line. To invoke this option simply write the argument as `GraphChk = 'Y'`.

- `pChar` (optional) is an alternate character variable for polynomial in the output, which is `x` by default, for example to obtain a polynomial in terms of `s` the function must be called with `pChar = 's'`.

Output arguments (given `m` sets of data and a polynomial of degree `n`):

- `paj` in the default form is scilab polynomial object of degree `n`. To evaluate such output the `horner` function must be used, for example to get the value of `paj` with `x=2.5` the call is `horner(paj, 2.5)`. If the Matlab's style was selected in the input `paj` will have the form `[an .. a1 a0]`.
- `ycalc` is a column vector with `m` elements corresponding to the `y` values calculated with eq. 1 applied to `x`.
- `statpar` is a vector with statistical parameters, `statpar = [St Sr stdv r2]` where
  - `St`: is defined as
$$S_t = \sum_{i=1}^{m} (y_i - \bar{y})^2 \tag{2}$$
    where $\bar{y}$ is the average of y
  - `Sr`: is the sum of the m residuals, defined as
$$S_r = \sum_{i=1}^{m} \left(y_i - y_i^{\text{calc}} \right)^2 \tag{3}$$
    where y_i^{calc} comes from eq. 1 applied to x_i.
  - `stdv`: standard deviation, defined as
$$\left(\frac{S_t}{m-1} \right)^{1/2} \tag{4}$$
  - `r2`: correlation coefficient, defined as
$$r^2 = \frac{S_t - S_r}{S_t} \tag{5}$$
  - `Syx`: standard error, defined as
$$S_{yx} = \left(\frac{S_r}{m-(n+1)} \right)^{1/2} \tag{6}$$

3 Example

Given the data

```
x_lst = [0 1 2 3 4 5]
y_lst = [2.1 7.7 13.6 27.2 40.9 61.1]
```

find the degree-3 polynomial that best fit the data in the form

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{7}$$

It is not necessary to recast the data. The command

```
polyfit(x_lst, y_lst, 3)
```

produces

```
2.2507937 + 3.3994709x + 1.2912698x^2 + 0.0759259x^3
```

Complete calling

```
[pol, yc, estad]=polyfit(x_lst, y_lst, 3)
```

produces

```
pol = 2.2507937 + 3.3994709x + 1.2912698x^2 + 0.0759259x^3
yc = [2.2507937 7.0174603 14.822222 26.120635 41.368254
61.020635]
estad = [2513.3933 3.3730159 22.420497 0.9986580 1.2986562]
```

For Matlab-like output and graphic comparison add `Mcomp = 'Y'` and `GraphChk = 'Y'`, for example

```
polyfit(x_lst, y_lst, 3, Mcomp='Y', GraphChk='Y')
```

produces

```
0.0759259 1.2912698 3.3994709 2.2507937
```

and a graphic comparison.

4 Mathematical background

Fitting of eq. 1 is based on the minimization of the objective function f_{obj} defined as

$$f_{\text{obj}} = \sum_{i=1}^{m} \left[y_i - \left(a_0 + a_1 x_i + a_2 x_i^2 + ... + a_n x_i^n \right) \right]^2 \tag{8}$$

meaning that $f_{\text{obj}} = f_{\text{obj}}(a_0, a_1, \ldots, a_n)$. But instead of using eq. 8 the problem recast in the generalized form

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2 + ... + a_n z_n \tag{9}$$

making $z_0(x) = 1$, $z_1(x) = x$, $z_2(x) = x^2$, ..., $z_n(x) = x^n$, this way the minimization based on

$$\frac{\partial f_{\text{obj}}}{\partial a_i} = 0 \tag{10}$$

for $i = 0, 1, 2, \ldots, n$ generates a matrix equation of the form

$$\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right)\mathbf{A} = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) \tag{11}$$

where the unknown values of a_i grouped in $\mathbf{A} = [a_0, a_1, \ldots, a_n]'$ depend on $\mathbf{Y} = [y_1, y_2, \ldots, y_m]'$ and

$$\mathbf{Z} = \begin{bmatrix} z_{10} & z_{11} & z_{12} & \cdots & z_{1n} \\ z_{20} & z_{21} & z_{22} & \cdots & z_{2n} \\ z_{30} & z_{31} & z_{32} & \cdots & z_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{m0} & z_{m1} & z_{m2} & \cdots & z_{mn} \end{bmatrix}. \tag{12}$$

For further explanation see Chapra and Canale's "Numerical Methods for Engineers, 5th ed., ch. 17 (McGraw-Hill, 2005).

Solution of eq. 11 using Scilab's \ operator is not advisable because the sums of powers of x tend to produce terms in the matrix with notorious differences in order of magnitude, making unreliable the matrix inversion. Instead the QR factorization is used to transform eq. 11 into

$$\mathbf{R}\mathbf{A} = \mathbf{Q}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) \tag{13}$$

where $\mathbf{R}$ is an upper triangular matrix, meaning that the a_i values can be calculated recursively from a_n down to a_0, with the definitions

$$\mathbf{B} = \mathbf{Q}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) = [b_{0,}b_{1,}\ldots,b_n]' \tag{14}$$

and

$$\mathbf{R} = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & \cdots & r_{0,n} \\ 0 & r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ 0 & 0 & r_{2,2} & \cdots & r_{2,n} \\ \vdots & & & & \vdots \\ 0 & \cdots & 0 & r_{n-1,n-1} & r_{n-1,n} \\ 0 & \cdots & 0 & 0 & r_{n,n} \end{bmatrix} \tag{15}$$

the a_n coefficient comes from

$$a_n = b_n / r_{n,n},$$

the a_{n-1} coefficient comes from a_n

$$a_{n-1} = \left(b_{n-1} - r_{n-1,n}a_n\right)/r_{n-1,n-1},$$

and so on, down to a_0 and filling the $\mathbf{A}$ variable.