Manual for the `polyfit` function

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

October 5, 2012

1 General description

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

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

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. By default `polyfit` produces 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

```
    [polyFunction, yCalc, statParams] = polyfit(xVar, yVar,
polyDeg, matlabForm, doGraph, polyChar)
```

Arguments (function input) for `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.
- `polyDeg` is a scalar (`polyDeg = n`), the degree of the polynomial.
- `matlabForm` (optional): if present makes the function to produce an output in the matlab form, see the output argument `polyFunction`. To invoke this option add the argument `MComp = "Y"`.
- `doGraph` (optional): if present produces a graphic with the input values (as circles) and a line produced with the result `polyFunction`. To invoke this option add the argument `doGraph = "Y"`.

- `polyChar` (optional): a character to be used in the polynomial result `polyFunction`. By default `polyChar` will be `"x"`, but it can be changed. For example to use z add the argument `polyChar = "z"`.

WARNING: the degree of the polynomial should be less than the number of $x-y$ data pairs, i.e. $m > n$. For example if there are 6 data pairs they can be fit with polynomials of degree 1, 2, 3, 4, or 5, but not of degree 6.

Results (function output), for `m` data sets and a polynomial of degree `n`:

- `polyFunction` in the default form is a Scilab polynomial of degree `n` that adjust the input data `y(x)`. If the `matlabForm` option was used it becomes a vector with `n+1` elements containing the values of a_i, that is `polyFunction = [an ... a2 a1 a0]`

- `yCalc` is a column vector with `m` elements corresponding to the `y` values calculated with `x1, x2, ..., xm`

- `statParams` is a vector with statistical parameters, `statParams = [St Sr stdv r2 Syx]` 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
$$y_i^{\text{calc}} = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n \tag{4}$$
    comes from eq. 1 applied to x_i.

  - `stdv`: standard deviation, defined as
$$\left(\frac{S_t}{m-1} \right)^{1/2} \tag{5}$$

  - `r2`: correlation coefficient, defined as
$$r^2 = \frac{S_t - S_r}{S_t}. \tag{6}$$
    If the `r2` value is close to 1 the correlation is good, and the opposite if close to 0.

  - `Syx`: standard error, defined as
$$S_{yx} = \left(\frac{S_r}{m-(n+1)} \right)^{1/2} \tag{7}$$

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

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{8}$$

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]
```

Matlab-like output can be obtained adding `matlabForm = ''Y''`, and graphic comparison with `doGraph = ''Y''`. For example

```
polyfit(x_lst, y_lst, 3, matlabForm=''Y'', doGraph=''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

$$f_{\text{obj}} = \sum_{i=1}^{m} \left[y_i - \left(a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n \right) \right]^2 \tag{9}$$

meaning that $f_{\text{obj}} = f_{\text{obj}}(a_0, a_1, \ldots, a_n)$. But instead of using eq. 9 the problem is recasted in the generalized form

$$y = a_0 z_0 + a_1 z_1 + a_2 z_2 + \ldots + a_n z_n \tag{10}$$

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

$$\frac{\partial f_{\text{obj}}}{\partial a_i} = 0 \tag{11}$$

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

$$\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\right)\mathbf{A} = \left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) \tag{12}$$

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

$$\mathbf{Z} = \begin{bmatrix} z_{10} & z_{11} & z_{12} & \cdots & z_{1n} \\ z_{20} & z_{21} & z_{22} & \cdots & z_{2n} \\ z_{30} & z_{31} & z_{32} & \cdots & z_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ z_{m0} & z_{m1} & z_{m2} & \cdots & z_{mn} \end{bmatrix}. \tag{13}$$

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

Solution of eq. 12 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. Instead the QR factorization is used to transform eq. 12 into

$$\mathbf{RA} = \mathbf{Q}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) \tag{14}$$

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

$$\mathbf{B} = \mathbf{Q}\left(\mathbf{Z}^{\mathrm{T}}\mathbf{Y}\right) = [b_{0,}b_{1,} \ldots, b_n]' \tag{15}$$

and

$$\mathbf{R} = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & \cdots & r_{0,n} \\ 0 & r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ 0 & 0 & r_{2,2} & \cdots & r_{2,n} \\ \vdots & & & & \vdots \\ 0 & \cdots & 0 & r_{n-1,n-1} & r_{n-1,n} \\ 0 & \cdots & 0 & 0 & r_{n,n} \end{bmatrix} \tag{16}$$

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.

Warning

This function is provided as-is, the author does not provide any guarantee about its results.