Nonlinear Least Squares Regression

Nonlinear regression attempts to fit a general model function

$$y = f(x; \mathbf{p})$$

that may depend nonlinearly on the parameter vector $\mathbf{p} = (p_1, p_2, \ldots, p_M)$ to a set of data with y-values $y_1, y_2, \ldots, y_N$ at the corresponding N x-values $x_1, x_2, \ldots, x_N$.

The least squares method still applies, with the (non-quadratic) function

$$\Phi(\mathbf{p}) = \frac{1}{2} |\mathbf{F}(\mathbf{p}) - \mathbf{F}_0|^2$$

with

$$\mathbf{F}(\mathbf{p}) = (f(x_1; \mathbf{p}), f(x_2; \mathbf{p}), \ldots, f(x_N; \mathbf{p}))$$

and

$$\mathbf{F}_0 = (y_1, y_2, \ldots, y_N)$$

to be minimised with respect to the parameter vector $\mathbf{p}$.

We require the zero gradient condition:

$$\nabla\Phi(\mathbf{p}) = \mathbf{J}^T(\mathbf{F}(\mathbf{p}) - \mathbf{F}_0) = 0$$

to hold at the minimum.

Introducing the vector

$$\mathbf{b}(\mathbf{p}) = -\nabla\Phi(\mathbf{p}) = -\mathbf{J}^T(\mathbf{F}(\mathbf{p}) - \mathbf{F}_0)$$

the zero gradient condition can be written as

$$\mathbf{b}(\mathbf{p}) = 0$$

The problem can be solved using the Newton-Raphson iterations described on page 33. The process starts with an initial parameter vector $\mathbf{p}$ and is corrected to be $\mathbf{p} + \Delta\mathbf{p}$ in the next iteration. The increment parameter vector $\Delta\mathbf{p}$ satisfies the linear equation

$$\mathbf{J}^b(\mathbf{p}) \cdot \Delta\mathbf{p} = -\mathbf{b}(\mathbf{p})$$

where $\mathbf{J}^b(\mathbf{p})$ denotes the Jacobian of the vector $\mathbf{b}$ with components given by

$$J^b_{ij} = \frac{\partial b_i}{\partial p_j} = -\frac{\partial^2 \Phi}{\partial p_i \partial p_j} = -H_{ij}$$

Therefore

$$\mathbf{J}^b = -\mathbf{H}$$

The increment parameter vector therefore satisfies the system of linear equations

$$\mathbf{H} \cdot \Delta\mathbf{p} = \mathbf{b}$$

i.e.

$$\Delta\mathbf{p} = -\mathbf{H}^{-1}\mathbf{J}^T(\mathbf{F}(\mathbf{p}) - \mathbf{F}_0)$$

As discussed as on page 83 $\mathbf{H}$ can be approximated by $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ for a system with moderate nonlinearity.

A set of illustrative MATLAB code is given as follows.

MATLAB program: nonlinfit.m

```
function p = nonlinfit(x,y,p0,tol)
% function call: p = nonlinfit(x,y,p0,tol)
% p - output row of optimaised parameters
% x - input row of x-data
% y - input row of y-data
% p0 - input row of initial parameters
% tol - tolerance

N = length(x);

% Choose a model function:
f = @model_func_exp;
%f = @model_func_lin;

% F0 - vector F0 contains the y-data
F0 = y;

% M - number of parameters
M = length(p0);

% p - parameter vector, initialised to be p0
p = p0 % displayed
```

```
% norm_dp is the norm of the incriment parameter vector p
% and is initialised to be greater than tol so that the
% iteration will be done at least once
norm_dp = 10*tol;

while norm_dp > tol
    for i = 1:N
        F(i) = f(x(i), p, 0);
        for j = 1:M
            J(i,j) = f(x(i), p, j);
        end
    end
    H = J'*J;
    b = -J'*(F-F0)';
    dp = (H\b)';
    norm_dp=sqrt(dp*dp');
    p = p + dp % displayed for each iteration
end
```

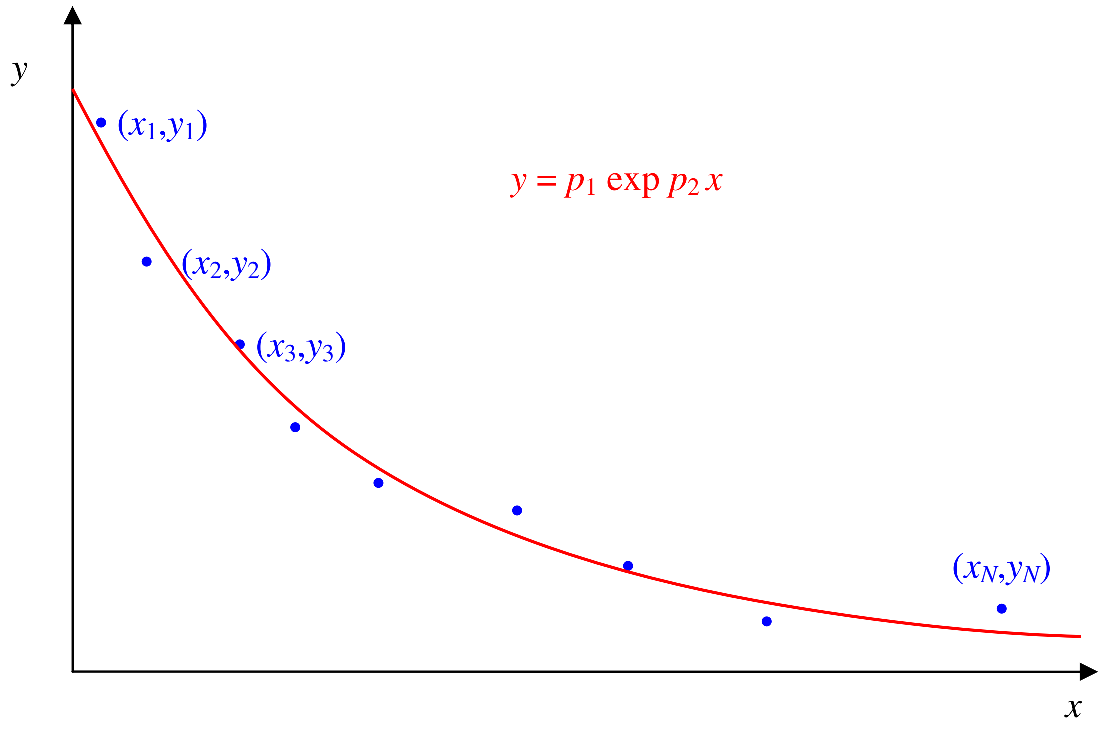
MATLAB function: model_func_lin.m

```
function y = model_func_lin(x, p, k)
% y = model_func_lin(x, p, k)
% y - output y-value
% x - input x-value
% k - differentialtion index:
%     if k=0 then y = f(x,p) = p(1)+p(2)*x
%     if k=i>0 then y = d f(x,p)/d p(i)
if k==0
    y = p(1)+p(2)*x;
elseif k==1
    y = 1;
elseif k==2
    y = x;
end
```

MATLAB function: model_func_exp.m

```
function y = model_func_exp(x, p, k)
% y = model_func_exp(x, p, k)
% y - output y-value
% x - input x-value
% k - differentialtion index:
%     if k=0 then y = f(x,p) = p(1)*exp(p(2)*x)
%     if k=i>0 then y = d f(x,p)/d p(i)
if k==0
    y = p(1)*exp(p(2)*x);
elseif k==1
    y = exp(p(2)*x);
elseif k==2
    y = p(1)*x*exp(p(2)*x);
end
```

Nonlinear regression using an exponential function


y
(x_1,y_1)
(x_2,y_2)
(x_3,y_3)
(x_N,y_N)
$y = p_1 \exp p_2 x$
x