1. (a) Let $\mathbf{Q}$ and $\mathbf{b}$ be as follows:

$$\mathbf{Q} = \begin{pmatrix} 6 & 4 \\ 4 & 6 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -5 \\ -6 \end{pmatrix}.$$

Then $g(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{b}^T\mathbf{x}$.

(b) We have

$$\nabla g(\mathbf{x}) = \begin{pmatrix} 6x_1 + 4x_2 + 5 \\ 6x_2 + 4x_1 + 6 \end{pmatrix} = \begin{pmatrix} 6x_1 + 4x_2 + 5 \\ 4x_1 + 6x_2 + 6 \end{pmatrix},$$

and we implement the functions in MATLAB as follows:

```
Q = [6 4; 4 6];
b = [-5; -6];
g = @(x) 1/2*x'*Q*x - b'*x;
gradg = @(x) Q*x - b;
```

(c) Running `eig(Q)` in MATLAB shows that the largest eigenvalue is $\lambda_{\max} = 10$. So, the largest range of values of α for which the algorithm is globally convergent is $0 < \alpha < \frac{1}{5}$.

(d)
```
% Define the function and its initial parameters.
Q = [6 4; 4 6];
b = [-5; -6];
g = @(x) 1/2*x'*Q*x - b'*x;
gradg = @(x) Q*x - b;
alpha = 1/10;
count = 0;

% Starting point;
x = [0; 0];
gx = g(x);

% Set previous value to inf to ensure at least one iteration is performed
gp = Inf; % gp = g previous

% Perform fixed step line search
while abs(gp - gx) >= 1e-6
    gp = gx;
    x = x - alpha * gradg(x);
    gx = g(x);
    count = count + 1;
end
fprintf('After %d iterations, ', count)
fprintf('the minimiser is [%.6f, %.6f].\n', x)
```

After 26 iterations the minimiser is $[-0.300756, -0.799244]$.

(e)
```
% Define the function and its initial parameters.
Q = [6 4; 4 6];
b = [-5; -6];
g = @(x) 1/2*x'*Q*x - b'*x;
gradg = @(x) Q*x - b;
count = 0;

% Starting point;
x = [0; 0];
gx = g(x);

% Set previous value to inf to ensure at least one iteration is performed
gp = Inf; % gp = g previous

% Perform steepest descent
while abs(gp - gx) >= 1e-6
    gp = gx;
    gg = gradg(x);
    alpha = (gg'*gg)/(gg'*Q*gg);
    x = x - alpha * gg;
    gx = g(x);
    count = count + 1;
end
fprintf('After %d iterations, ', count)
fprintf('the minimiser is [%.6f, %.6f].\n', x)
```

After 6 iterations the minimiser is $[-0.299995, -0.799987]$.

2. (a) As $h(\mathbf{x})$ is the sum of two squares, both of which are zero only when $x_1 = x_2 = 1$, it follows that $\mathbf{x} = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$ is the unique minimiser.

(b) $\nabla h(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{pmatrix} = \begin{pmatrix} 400x_1^3 + 2x_1 - 400x_1x_2 - 2 \\ 200x_2 - 200x_1^2 \end{pmatrix}$

$D^2h(\mathbf{x}) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}$

Then in MATLAB:

```
h = @(x) 100*(x(2) - x(1)^2)^2 + (x(1) - 1)^2;
gh = @(x) [400*x(1)^3 + 2*x(1) - 400*x(1)*x(2) - 2
           200*x(2) - 200*x(1)^2];
hh = @(x) [1200*x(1)^2 - 400*x(2) + 2, -400*x(1); -400*x(1), 200];
```

(c)
```
% Define the function, gradient and hessian.
h = @(x) 100*(x(2) - x(1)^2)^2 + (x(1) - 1)^2;
gh = @(x) [400*x(1)^3 + 2*x(1) - 400*x(1)*x(2) - 2;
           200*x(2) - 200*x(1)^2];
hh = @(x) [1200*x(1)^2 - 400*x(2) + 2, -400*x(1); -400*x(1), 200];

% Starting point
x = [0; 0];
count = 0;
% Set initial difference to inf so at least one iteration is performed.
diff = inf;

% Perform Newton's method
while (diff >= 10e-10)
    xp = x;
    x = x - hh(x)\gh(x); % In MATLAB, A\B computes inv(A)*B
    diff = abs((h(x) - h(xp))/h(xp));
```

```
    count = count + 1;
end
fprintf('After %d iterations, ', count)
fprintf('the minimiser is [%.6f, %.6f].\n', x)
```

After 3 iterations , the minimiser is [1.000000, 1.000000]

(d)
```
% Step size
alpha = 1/100;

% Starting point
x = [0; 0];
% Set initial difference to infinity so at least one iteration is
% performed.
diff = inf;
% Keep track of iterations
count = 0;

% Perform fixed step size line search
while (diff >= 10e-10)
    xp = x;
    x = x - alpha*gh(x);
    diff = abs((h(x) - h(xp))/h(xp));
    count = count + 1;
end
fprintf('After %d iterations, ', count)
fprintf('the minimiser is [%.6f, %.6f].\n', x)
```

After 41 iterations, the minimiser is [-Inf, Inf].
The algorithm did not converge and instead shoots off to infinity.

(e)
```
% Function, initial simplex and parameters
h = @(x) 100*(x(2) - x(1)^2)^2 + (x(1) - 1)^2;
x1 = [0; 0];
x2 = [0; 1];
x3 = [1; 0];
count = 0;
max_iterations = 100;
alpha = 1;
gamma = 2;
rho = 1/2;
sigma = 1/2;

% Perform the downhill simplex method
for k = 1:max_iterations
    % Sort so that h(x1) < h(x2) < h(x3)
    if h(x2) < h(x1)
        temp = x1; x1 = x2; x2 = temp;
    end
    if h(x3) < h(x1)
        temp = x1; x1 = x3; x3 = temp;
    end
    if h(x3) < h(x2)
        temp = x2; x2 = x3; x3 = temp;
    end

    % Calculate centre of mass
    xo = (x1 + x2)/2;
```

```
    % Calculate reflection
    xr = xo + alpha*(xo - x3);

    if h(x1) < h(xr) && h(xr) < h(x2)
        x3 = xr;
    elseif h(xr) < h(x1)
        % Calculate expansion
        xe = xo + gamma*(xr - xo);
        if h(xe) < h(xr)
            x3 = xe;
        else
            x3 = xr;
        end
    else
        if h(xr) < h(x3)
            % Calculate outside contraction
            xc = xo + rho*(xr - xo);
        else
            % Calculate inside contraction
            xc = xo + rho*(x3 - xo);
        end
        if h(xc) < h(x3)
            x3 = xc;
        else
            % Shrink
            x2 = x1 + sigma*(x2 - x1);
            x3 = x1 + sigma*(x3 - x1);
        end
    end
end
fprintf('The minimiser is [%.6f, %.6f].\n', x1)
```

The minimiser is [1.000000, 1.000000]

3. Replace the definition of the function h in the previous part with the following:

```
x_data = [0.28 0.76 0.93 1.88 3.03 4.73 4.90];
y_data = [1.62 1.22 1.80 1.03 1.17 0.70 0.22];
h = @(coeffs) sum((y_data - coeffs(1).*exp(coeffs(2).*x_data)).^2);
```

The minimiser is [1.788263, -0.234402], so the model is $y = 1.788263e^{-0.234402x}$.