Assignment 2

02610 Optimization and Data Fitting – Anders Hørsted (s082382)

Question 1: Fitting to Air Pollution Data

In this question we are going to fit sums of sines and cosines to the Air Pollution Data given in the assignment. First a function `NOfit` that calculates the parameters and the residuals for the fit is created.

Question 1.1

The function should have the call `[x_star, r_star] = NOfit(t,y,n)`, but for convenience it is written to also return the design matrix A. The function `NOfit` relies on `get_A` to calculate the actual design matrix. The function `get_A` is used later when the fits are plotted.

```
function [xstar, rstar, A] = NOfit(t, y, n)

    omega = 2*pi/24;
    m = length(t);

    if m ~= length(y)
        error('The length of t and y should match')
    end

    A = get_A(t, n);

    xstar = (A'*A)\(A'*y);
    rstar = y - A*xstar;
end
```

Code Listing 1: `NOfit` function to fit sine, cosines of order `n`

```
function [A] = get_A(t, n)

    if mod(n, 2) == 0
        error('Only odd number of basis functions');
    end

    omega = 2*pi/24;
    m = length(t);
    A = zeros(m, n);
    fns = {@sin, @cos};
    for i=1:m
        for j=1:n
            % Cycle between sine and cosine
            f = fns{mod(j,2)+1};
            A(i,j) = f(floor(j/2)*omega*t(i));
        end
    end
end
```

Code Listing 2: Helper function `get_A` used to generate design matrix

Question 1.2

See appendix A.1 for code used in this exercise.

The `NOfit` function is now tested. Using the data given in the assignment text a 3rd order cosine is fitted which gives the model.

$$M(\mathbf{x}, t) = 186.81 - 44.94 \sin(\omega t) - 93.43 \cos(\omega t)$$

To confirm the implementation of `NOfit` the residual 2-norm is checked and it is the expected $||\, \mathbf{r}^* \,||_2 = 292.558$. Since the implementation seems to be right, the fitted model is plotted along with the data in figure 1 .

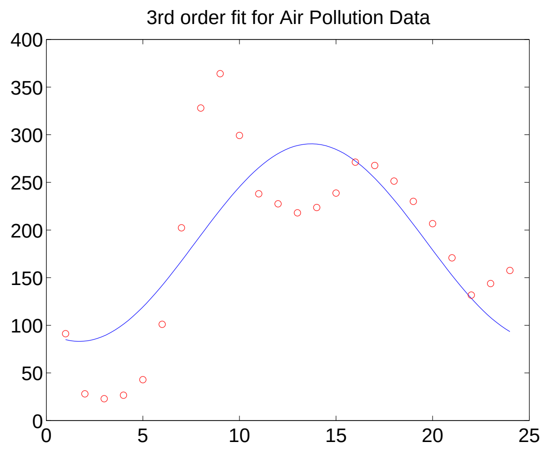

Figure 1: 3rd order fitted model M

Question 1.3

See appendix A.2 for code used in this exercise.

The optimal order of the fit must be determined. Using the test for random signs and the test for correlation, for the orders $n = 3, 5, 7, 9, 11, 13$ gives the results shown in figure 2.

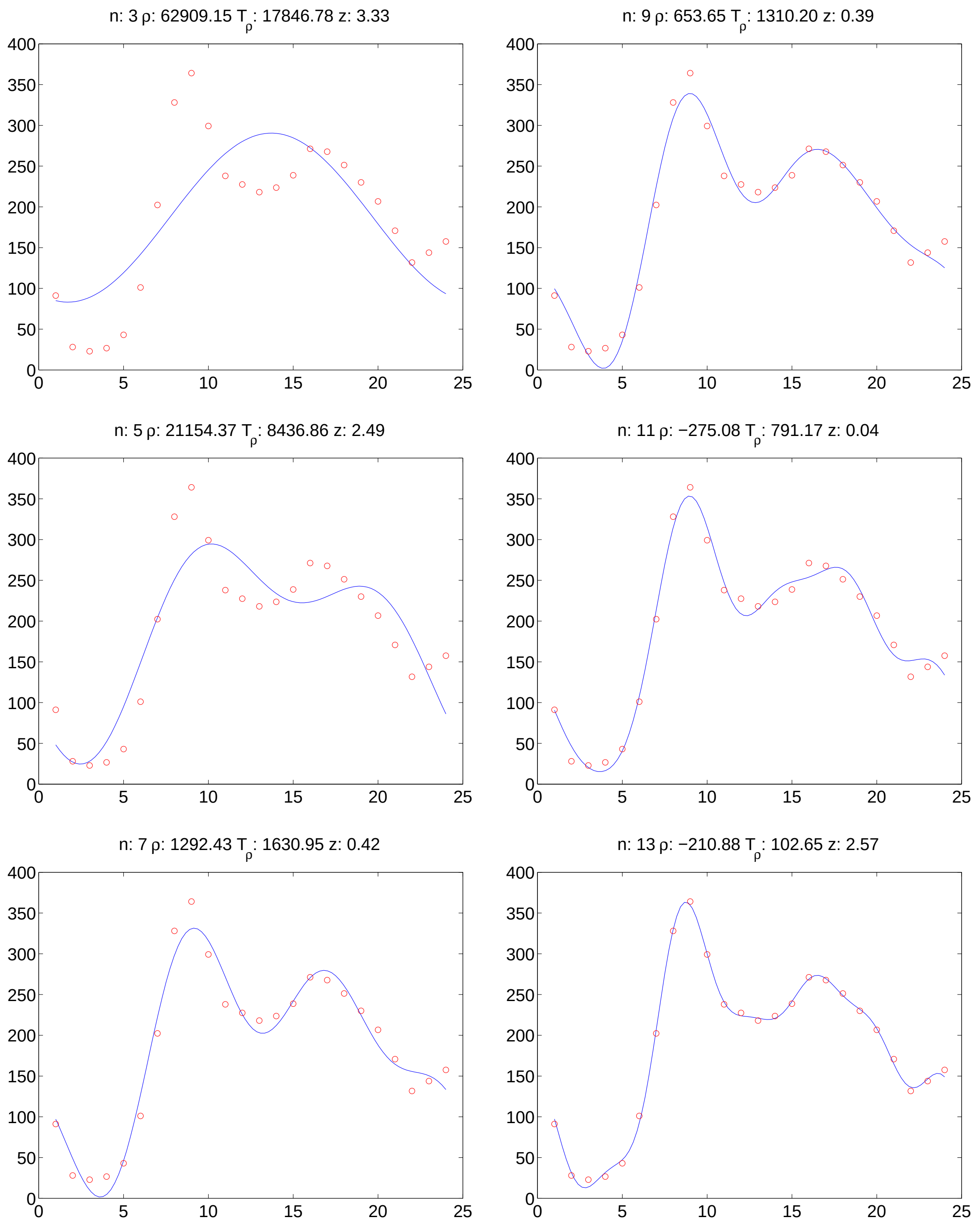

Figure 2: Fitted models for $n = 3, 5, 7, 9, 11, 13$ including residual test results

Question 2

In this problem a chemical reaction rate is modelled as a function of the concentration of a substrate. The predicted reaction rate is modelled as

$$\hat{y} = \frac{\theta_1 x}{\theta_2 + x}$$

where x is the concentration and θ_1 and θ_2 are the parameters of interest. Given the 12 measurements of reaction rate and corresponding concentration we then model the measured reaction rate y by

$$y = \hat{y} + e \qquad (1)$$

where $e \sim N(0, \sigma^2)$ and σ^2 is unknown.

Question 2.1

See appendix A.4 for code used in this exercise.

First a plot of the experimental data (x, y) and another plot of the inverse data $(1/x, 1/y)$ are created and shown in figure 3. From the plots it do look as if a straight line could be fitted to the inverse data.

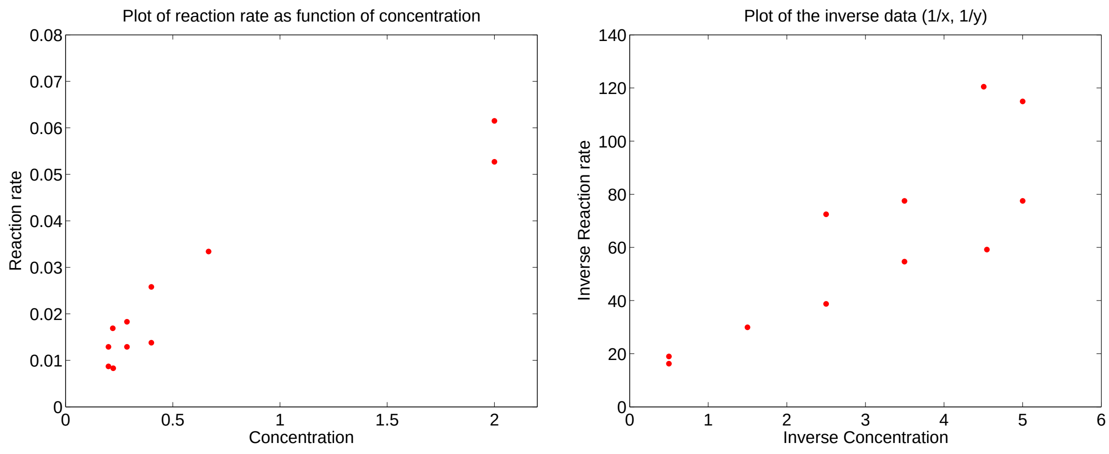

Figure 3: Plot of experimental data (x, y) and inverse data $(1/x, 1/y)$ for question 2

Since the relationship between the inverse reaction rate and the inverse concentration could be linear, a first attempt at finding the parameters θ_1 and θ_2 is to look at $\frac{1}{\hat{y}}$ as a function of $\frac{1}{x}$. This gives

$$\frac{1}{\hat{y}} = \frac{\theta_2 + x}{\theta_1 x} = \frac{\theta_2}{\theta_1}\frac{1}{x} + \frac{1}{\theta_1}$$

and by setting $\lambda_1 = \frac{1}{\theta_1}$ and $\lambda_2 = \frac{\theta_2}{\theta_1}$, we get a linear model

$$\frac{1}{\hat{y}} = \lambda_1 + \lambda_2 \frac{1}{x} \tag{2}$$

We don't know $\frac{1}{\hat{y}}$ but we can find an estimate for the parameters λ_1 and λ_2 by fitting $\frac{1}{y}$ linear on $\frac{1}{x}$. From the estimates of λ_1 and λ_2 we can then find θ_1 and θ_2. This method is implemented and shown in listing 3

```
function [theta, lambda] = calc_chemical_reaction_params_linear(x, y)

    lambda = zeros(2,1);
    theta = zeros(2,1);

    A = [ones(length(x), 1) 1./x];
    lambda = (A'*A)\(A'*(1./y));

    theta(1) = 1/lambda(1);
    theta(2) = lambda(2)/lambda(1);

end
```

Code Listing 3: Function that finds θ_1 and θ_2 by fitting $\frac{1}{y}$ linear on $\frac{1}{x}$

Using the function in listing 3 gives the parameter estimates

$$\theta^*_{LS} \approx \begin{pmatrix} 0.14 \\ 2.54 \end{pmatrix}$$

and from these parameter estimates we plot the model $\hat{y}$ along with the original data in figure 4

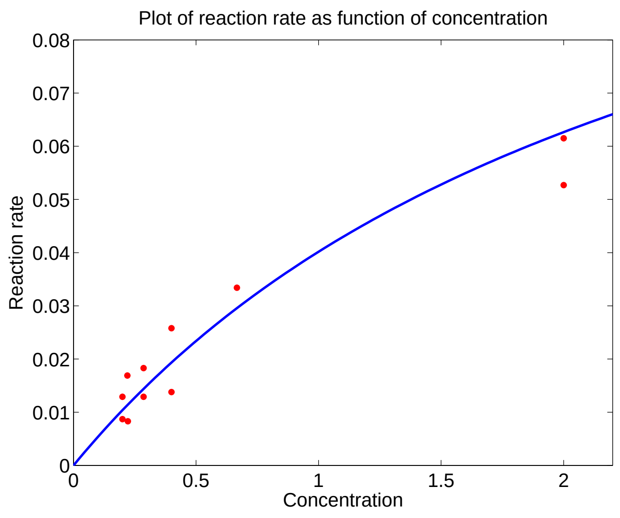

Figure 4: Plot of original data and the model $\hat{y}$, with parameters estimated from the linear model (2)

From the figure it is seen that the fitted model isn't explaining the data perfect. The problem is that by fitting $\frac{1}{y}$ linear on $\frac{1}{x}$ we implicitly assume that $\frac{1}{y}$ can be written as a sum of a linear model of $\frac{1}{x}$ and an gaussian error term. This is the same as assuming that $\frac{1}{y}$ is normal distributed, but from our original model (1) we get that y is normal distributed with mean $\hat{y}$ and variance σ^2. We therefore found the parameter estimate θ^*_{LS} by assuming that the inverse of a normal distributed variable, is also normal distributed. This isn't correct and as a result we found that the fit wasn't perfect.

Question 2.2

See appendix A.5 for code used in this exercise.

In this question the parameters θ is found by solving the non-linear least squares problem.

$$\phi(\theta) = \frac{1}{2}\sum_{i=1}^{n} ||\, y_i - f(\theta; x_i)\, ||_2^2 \tag{3}$$

First a contour plot of $\phi(\theta)$ is created and θ^*_{LS} is shown with a dot in the plot. The contour plot is shown in figure 5 and it looks as if there are values for θ that gives smaller values for $\phi(\theta)$ than θ^*_{LS}.

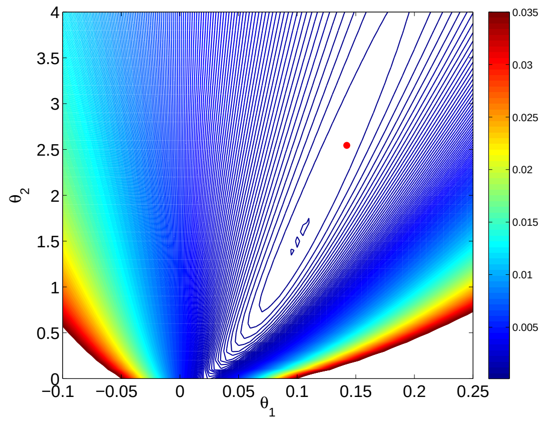

Figure 5: Contour plot of ϕ defined in (3) with θ^*_{LS} shown

To find an better estimate than θ^*_{LS}, a nonlinear least squares method is used instead of the linear least squares method from the previous question. Specifically the Levenberg-Marquardt (L-M) algorithm is used to find θ^*. By running the L-M algorithm, the parameter estimate θ^* is found as

$$\theta^* \approx \begin{pmatrix} 0.10 \\ 1.51 \end{pmatrix}$$

and estimates for $\hat{\sigma}^2$ and $\mathrm{Cov}[\theta^*]$ is found as

$$\hat{\sigma}^2 \approx 9.54e\text{-}06, \quad \text{Cov}[\theta^*] \approx \begin{bmatrix} 0.000111 & 0.00275 \\ 0.00275 & 0.0742 \end{bmatrix}$$

From $\hat{\sigma}^2$ and $\text{Cov}[\theta^*]$ a 95% confidence intervals for θ^* can be found which gives

$$\text{Conf}_{95\%}(\theta_1^*) \approx [0.0804; 0.122], \quad \text{Conf}_{95\%}(\theta_2^*) \approx [0.98; 2.05]$$

The optimal estimate θ^* is now plotted in a contour plot of $\phi(\theta)$ and is shown in figure 6. From the figure it looks as if the new estimate gives a lower value for $\phi(\theta)$, than the estimate from the previous question.

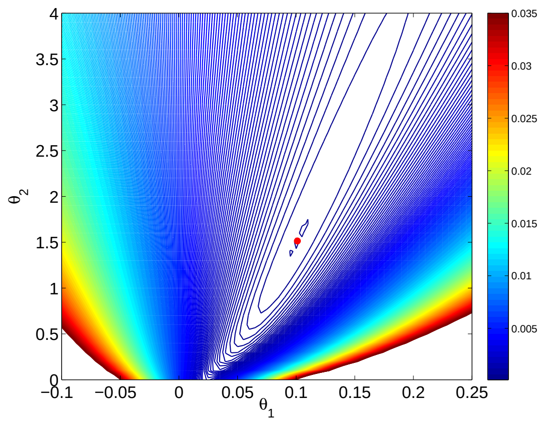

Figure 6: Contour plot of ϕ defined in (3) with θ^* shown

To better visualize the improvement the Michaelis-Menten model is now plotted for both θ_{LS}^* and θ^* along with the measured data. The plot is found in figure 7 and the model with θ^* as parameters seems to fit the data best.

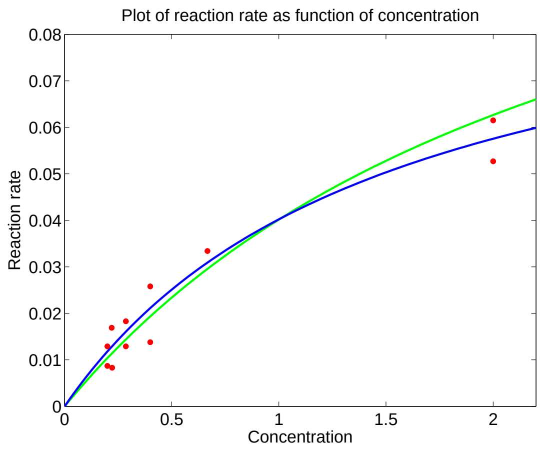

Figure 7: Plot of linear and non-linear fitted models

Question 2.3

See appendix A.6 for code used in this exercise.

We continue to work with the chemical reaction used in question 2.1 and 2.2, but instead of measurements of reaction rate and concentration, we now have measurements of concentrations x_i at different times t_i. Denoting the real concentration at time t as $x(t)$ the measured concentration is modelled as

$$y(t_i) = \hat{y}(\theta; t_i) + e$$

where $e \sim N(0, \sigma^2)$ and $\hat{y}(\theta; t_i) = x(t_i; \theta)$ is the concentration predicted by

$$\frac{dx(t)}{dt} = -\frac{\theta_1 x(t)}{\theta_2 + x(t)}, \quad x(0) = 10.0 \tag{4}$$

The parameters θ should now be estimated by

$$\min_{\theta} \phi(\theta) = \frac{1}{2} \sum_{i=1}^{n} ||\, y(t_i) - \hat{y}(\theta; t_i)\, ||_2^2 \tag{5}$$

using the measurements in `MMBatchData.mat`. The data is plotted in figure 8

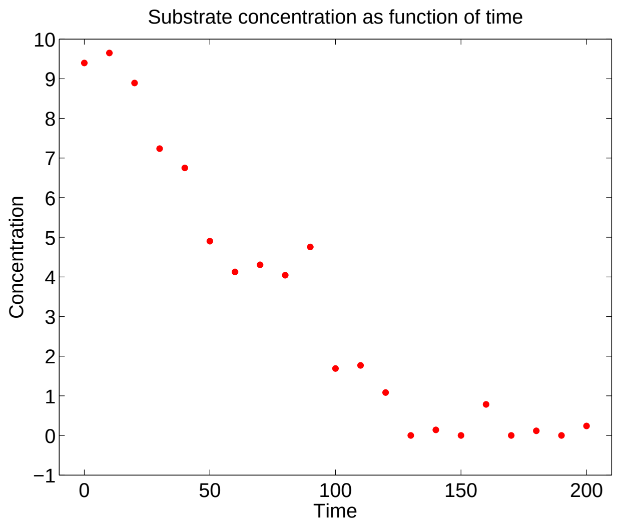

Figure 8: Plot of measurements of concentration for different times

To actually estimate θ a Matlab function that computes $\hat{y}(\theta; t_i)$ for a given θ and all t_i, is implemented and shown in listing 4. The function takes a vector `t` containing all t_is and another vector `p` that is the parameter θ. Using the Matlab function `ode45` the solution of (4) is found for all t_i, and since $\hat{y} = x$ we can directly return the solution obtained from `ode45`

```
function yhat = ex23_yhat(t, p)
```

```
    x0 = 10;
    model = @(t, x)-p(1)*x./(p(2)+x);
    [T, X] = ode45(model, t, x0, []);
    yhat = X;
end
```

Code Listing 4: Function to compute $\hat{y}(\theta, t_i)$, for a given θ and all t_i

We are now able to compute $\hat{y}(\theta; t_i)$ and therefore we can also compute $\phi(\theta)$ for different values of θ. Therefore a contour plot of $\phi(\theta)$ can be created. The contour plot is shown in figure 9. From the plot there seems to be a minimum near (0.1, 1).

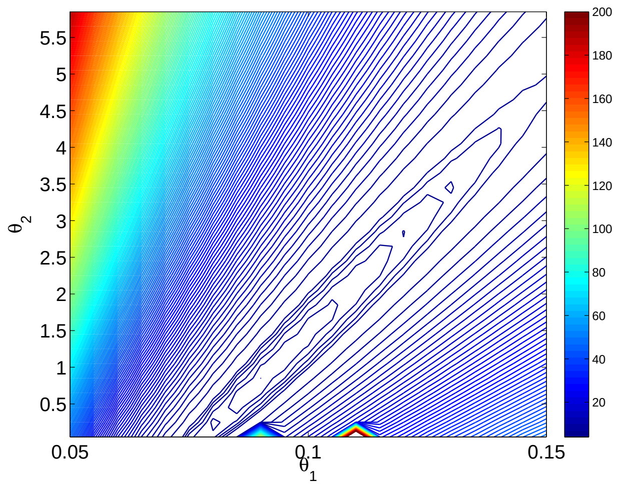

Figure 9: Contour plot for ϕ defined in (5)

To actually find a minimum for $\phi(\theta)$ a nonlinear least squares algorithm is used. To get good performance we need to supply the least squares function with the derivative of the residuals. We get that

$$\frac{\partial(y_i - \hat{y}(\theta; t_i))}{\partial\theta} = -\frac{\partial\hat{y}(\theta; t_i)}{\partial\theta}$$

so we only need to determine $\frac{\partial}{\partial\theta}\hat{y}(\theta; t_i)$. A function that calculates $\frac{\partial}{\partial\theta}\hat{y}(\theta; t_i)$ (and $\hat{y}(\theta; t_i)$) for all t_i is therefore implemented and shown in listing 5.

```
function z = ex23_z(t, p)
    z0 = [10 0 0];

    function zdot = model(t, z, p)
        x = z(1,1);
        Sp = z(2:3, 1);

        x_plus_p2 = p(2)+x;
        x_plus_p2_sq = x_plus_p2.^2;
        xdot = -p(1)*x./x_plus_p2;

        dfdx = -p(1)./x_plus_p2 + p(1)*x./x_plus_p2_sq;
        dfdp = [-x./x_plus_p2;  p(1)*x./x_plus_p2_sq];

        Spdot = dfdx*Sp + dfdp;

        zdot = [xdot; Spdot(:)];
    end

    [T, X] = ode45(@model, t, z0, [], p);
    z = X;
end
```

Code Listing 5: Function to calculate $\hat{y}$ and $\frac{\partial \hat{y}}{\partial \theta}$ for a given θ and all t_i

The implementation is based on page 11-17 in the slides for lecture 12. The implementation utilizes the fact that since $\hat{y} = x$ we have that (using the notation of the slide) $S_\theta(t) = \frac{\partial}{\partial \theta}\hat{y}(\theta; t)$. By calculating z (as defined on slide 16) we get excatly the information we need to run a non-linear least squares method.

The parameters θ is now found by using the `marquardt` method from the IMM optimbox with the settings `tau=1e-3`, `tolg=1e-7`, `tolx=1e-12` and `maxeval=100`. The function uses 35 iterations to find the parameters

$$\theta \approx \begin{pmatrix} 0.09 \\ 0.77 \end{pmatrix}$$

When the parameters are found, $\hat{\sigma}^2$ and $\text{Cov}[\theta]$ can be estimated, which gives

$$\hat{\sigma}^2 \approx 0.21, \quad \text{Cov}[\theta] \approx \begin{bmatrix} 7.28e\text{-}05 & 0.00445 \\ 0.00445 & 0.293 \end{bmatrix}$$

Using $\hat{\sigma}^2$ and $\text{Cov}[\theta]$, confidence intervals for the parameter estimates can be calculated, which gives

$$\text{Conf}_{95\%}(\theta_1) \approx [0.0725; 0.106], \quad \text{Conf}_{95\%}(\theta_2) \approx [-0.29; 1.83]$$

A Appendices

All Matlab source code is included in the appendices. All the source code including the LaTex code used for the report can also be found at `https://github.com/alphabits/dtu-fall-2011/tree/master/02610/assignment-2`.

A.1 Question 1.2

```
load_ex1;

n = 3;

[xstar, rstar] = NOfit(t, y, n);

file = fopen('../tables/3rd-order-fitted-model.tex', 'w');
fprintf(file, 'M(\\myvec{x}, t) = %.02f %+.02f \\sin(\\omega t) %+.02f \\cos(\\omega t)', xstar);
fclose(file);

plot_fit(t, y, n, xstar, rstar, '3rd order fit for Air Pollution Data');
saveeps('../media/3rd-order-fit.eps');
```

Code Listing 6: ex12.m

```
function [] = plot_fit(t, y, n, x, r, plot_title)
    tplot = linspace(1,24,100);
    A = get_A(tplot, n);
    fit = A*x;
    fs = 18;
    set(gca, 'fontsize', fs);
    plot(tplot, fit, '-b', t, y, 'or');
    title(plot_title, 'fontsize', fs);
end
```

Code Listing 7: plot_fit.m

A.2 Question 1.3

```
load_ex1;


for n=3:2:13
    [x, r] = NOfit(t, y, n);
    plot_fit_with_res_analysis(t, y, n, x, r);
    saveeps(sprintf('../media/order-determination-%d.eps', n));
end
```

Code Listing 8: ex13.m

```
function [] = plot_fit_with_res_analysis(t, y, n, x, r)
    z = run_score(r);
    [rho, Trho] = correlation_score(r);
    plot_title = sprintf('n: %d \\rho: %.02f T_\\rho: %.02f z: %.02f', ...
                         n, rho, Trho, z);
    plot_fit(t, y, n, x, r, plot_title);
end
```

Code Listing 9: plot_fit_with_res_analysis.m

```
function score = run_score(seq)
    m = length(seq);
    u = sum(seq(1:(end-1)).*seq(2:end) < 0) + 1;
    nplus = sum(seq>0);
    nminus = sum(seq<=0);
    mu = (2*nplus*nminus)/m + 1;
    sd = sqrt((mu-1)*(mu-2)/(m-1));
    score = abs(u-mu)/sd;
end
```

Code Listing 10: run_score.m

A.3 Question 1.4

```
load_ex1;

m = length(d(:,1));
sstars = zeros(1,11);


for i=1:11
    n = 2*i + 1;
    [x, r, A] = NOfit(d(:,1), d(:,2), n);
    sstars(i) = norm(r)/sqrt(m-n);
end

plot(3:2:23, sstars, 'r.', 'MarkerSize', 16);
set(gca, 'FontSize', 16);
xlabel('n');
ylabel('s^*');
title('Estimated standard deviation for different n');
saveeps('../media/ex14-sd.eps');
```

Code Listing 11: ex14.m

A.4 Question 2.1

```
dat = load('../data/reaction-rates.txt');

x = dat(:,1);
y = dat(:,2);

fs = 16;

plot(x, y, 'r.', 'MarkerSize', 16);
set(gca, 'FontSize', fs);
axis([0 2.2 0 0.08]);
xlabel('Concentration');
ylabel('Reaction rate');
title('Plot of reaction rate as function of concentration');
%saveeps('../media/ex21-plot.eps');

plot(1./x, 1./y, 'r.', 'MarkerSize', 16);
set(gca, 'FontSize', fs);
axis([0 6 0 140]);
xlabel('Inverse Concentration');
ylabel('Inverse Reaction rate');
title('Plot of the inverse data (1/x, 1/y)');
%saveeps('../media/ex21-plot-inv.eps');

[theta, lambda] = calc_chemical_reaction_params_linear(x, y);
ex2_plot_model_with_data(theta, x, y);
%saveeps('../media/ex21-linear-model.eps');

fid = fopen('../tables/param-estimates-ex21.tex', 'w');
fprintf(fid, '\\theta_{LS}^* \\approx \\begin{pmatrix} %0.2f \\\\ %0.2f \\end{pmatrix} \n', theta);
fclose(fid);

% plot(x_inv_preds, y_inv_preds, 'b-', 1./x, 1./y, 'ro');
% axis([0 6 0 140]);
```

Code Listing 12: ex21.m

```
function [theta, lambda] = calc_chemical_reaction_params_linear(x, y)

    lambda = zeros(2,1);
    theta = zeros(2,1);

    A = [ones(length(x), 1) 1./x];
    lambda = (A'*A)\(A'*(1./y));

    theta(1) = 1/lambda(1);
    theta(2) = lambda(2)/lambda(1);

end
```

Code Listing 13: calc_chemical_reaction_params_linear.m

```
function [] = ex2_plot_model_with_data(theta, x, y, line_spec)
    x_preds = linspace(0,2.2,100);
    y_preds = ex2_yhat(x_preds, theta);

    plot(x_preds, y_preds, line_spec, x, y, 'r.', 'MarkerSize', 16, 'LineWidth', 2);
    set(gca, 'FontSize', 16);
    axis([0 2.2 0 0.08]);
    xlabel('Concentration');
    ylabel('Reaction rate');
    title('Plot of reaction rate as function of concentration');
end
```

Code Listing 14: ex2_plot_model_with_data.m

A.5 Question 2.2

```
addpath('/home/anders/dtu/E11/02610/src/immoptibox');

dat = load('../data/reaction-rates.txt');

x = dat(:,1);
y = dat(:,2);

[theta_lin, lambda] = calc_chemical_reaction_params_linear(x, y);

x_con = -0.1:0.005:0.25;
y_con = 0:0.1:4;
v = 0:0.0001:0.035;
[X_con, Y_con] = meshgrid(x_con, y_con);

phi_fun = ex22_phi(x, y);
phi = arrayfun(phi_fun, X_con, Y_con);

contour(X_con, Y_con, phi, v, 'linewidth', 1);
set(gca, 'FontSize', 16);
xlabel('\theta_1');
ylabel('\theta_2');
colorbar;
hold on;
plot(theta_lin(1), theta_lin(2), 'r.', 'MarkerSize', 20);
hold off;
saveeps('../media/ex22-contour-linear.eps');


[xs, info, perf] = marquardt(@residual_jacobian_ex22, theta_lin, [0 1e-7 1e-12 1e3], x, y);
theta_marq = xs(:, end);
[r_marq, J_marq] = residual_jacobian_ex22(theta_marq, x, y);
sigma_est_marq = 0.5*sum(r_marq.^2)/(length(x)-length(theta_marq));

calculate_and_save_covariance_info(theta_marq, sigma_est_marq, J_marq, 'ex22', '\\theta^*');


ex2_plot_model_with_data(theta_lin, x, y, 'g-');
hold on;
ex2_plot_model_with_data(theta_marq, x, y, 'b-');
hold off;
saveeps('../media/ex22-models-with-data.eps');
```

```
contour(X_con, Y_con, phi, v, 'linewidth', 1);
set(gca, 'FontSize', 16);
xlabel('\theta_1');
ylabel('\theta_2');
colorbar;
hold on;
plot(theta_marq(1), theta_marq(2), 'r.', 'MarkerSize', 20);
hold off;
saveeps('../media/ex22-contour-marquardt.eps');
```

Code Listing 15: ex22.m

```
function phi_fun = ex22_phi(x, y)

    function phi = calc_phi(theta1, theta2)
        yhat = ex2_yhat(x, [theta1, theta2]);
        phi = 0.5*sum((y-yhat).^2);
    end

    phi_fun = @calc_phi;

end
```

Code Listing 16: ex22_phi.m

```
function [r J] = residual_jacobian_ex22(theta, x, y)
    xplustheta2 = theta(2) + x;

    r = y - theta(1)*x./xplustheta2;
    J = -[x./xplustheta2  -theta(1)*x./xplustheta2.^2];
end
```

Code Listing 17: residual_jacobian_ex22.m

```
function [cov_theta, err_theta, min_theta, max_theta] = calculate_and_save_covariance_info(theta, sigma, J, name, the
    cov_theta = sigma*((J'*J)\eye(length(theta)));
    err_theta = 1.96*sqrt(diag(cov_theta));
    min_theta = theta - err_theta;
    max_theta = theta + err_theta;

    % Save latex representation of the results
    param_template = [theta_repr, ' \\approx \\begin{pmatrix} %0.2f \\\\ %0.2f \\end{pmatrix} \n'];
    saveline(['param-estimates-', name, '.tex'], param_template, theta);
    sigma_template = '\\\\hat{\\\\sigma}^2 \\\\approx %0.3g \n';
    sigma_line = sprintf(sigma_template, sigma);
    sigma_line = strrep(sigma_line, 'e-', 'e\\mbox{-}');
    saveline(['sigma-estimate-', name, '.tex'], sigma_line, []);
    cov_theta_tmpl = ['\\text{Cov}[', theta_repr, '] \\approx '...
                      '\\begin{bmatrix} %0.3g & %0.3g \\\\ %0.3g & %0.3g \\end{bmatrix}'];
    saveline(['covariance-', name, '.tex'], cov_theta_tmpl, cov_theta);
    conf_ints_tmpl = ['\\text{Conf}_{95\\%%}(', theta_repr, '_1) \\approx [%0.3g; %0.3g], ', ...
                      '\\quad \\text{Conf}_{95\\%%}(', theta_repr, '_2) \\approx [%0.3g; %0.3g]'];
    saveline(['confidence-', name, '.tex'], conf_ints_tmpl, ...
             [min_theta(1), max_theta(1), min_theta(2), max_theta(2)]);
end
```

Code Listing 18: calculate_and_save_covariance_info.m

A.6 Question 2.3

```
addpath('/home/anders/dtu/E11/02610/src/immoptibox');

data = load('../data/MMBatchData.mat');
data = data.data;

x = data(:,1);
y = data(:,2);


plot(x, y, 'r.', 'MarkerSize', 16);
set(gca, 'FontSize', 16);
xlabel('Time');
ylabel('Concentration');
title('Substrate concentration as function of time');
axis([-10 210 -1 10]);
saveeps('../media/ex23-data.eps');


odefun = @(t, x, p)-p(1)*x./(p(2)+x);
[T, X] = ode45(odefun, [0 200], 10, [], [0.14 2.54]);
[T2, X2] = ode45(odefun, 0:10:200, 10, [], [0.14 2.54]);

x_con = 0:0.05:0.5;
y_con = 0:0.2:4;
x_con = 0.05:0.005:0.15;
y_con = 0.05:0.2:6;
v = [0:0.2:5, 5:1:200];
[X_con, Y_con] = meshgrid(x_con, y_con);

phi_fun = get_phi_fun(x, y, @ex23_yhat);
phi = arrayfun(phi_fun, X_con, Y_con);

contour(X_con, Y_con, phi, v, 'linewidth', 1);
set(gca, 'FontSize', 16);
xlabel('\theta_1');
ylabel('\theta_2');
colorbar;
saveeps('../media/ex23-contour.eps');
```

Code Listing 19: ex23.m

A.7 Question 2.3 marquardt

```
addpath('/home/anders/dtu/E11/02610/src/immoptibox');

data = load('../data/MMBatchData.mat');
data = data.data;

x = data(:,1);
y = data(:,2);


[thetas, info, perf] = marquardt(@residual_jacobian_ex23, [0.1 1.5], [0 1e-7 1e-12 0], x, y);
theta = thetas(:, end);
[r, J] = residual_jacobian_ex23(theta, x, y);
sigma_est = 0.5*sum(r.^2)/(length(x)-length(theta));
[covar, err, min_theta, max_theta] = calculate_and_save_covariance_info(theta, sigma_est, J, 'ex23', '\\theta');
```

Code Listing 20: ex23_marquardt.m

```
function yhat = ex23_yhat(t, p)
    x0 = 10;
    model = @(t, x)-p(1)*x./(p(2)+x);
    [T, X] = ode45(model, t, x0, []);
    yhat = X;
end
```

Code Listing 21: ex23_yhat.m

```
function z = ex23_z(t, p)
    z0 = [10 0 0];

    function zdot = model(t, z, p)
        x = z(1,1);
        Sp = z(2:3, 1);

        x_plus_p2 = p(2)+x;
        x_plus_p2_sq = x_plus_p2.^2;

        xdot = -p(1)*x./x_plus_p2;

        dfdx = -p(1)./x_plus_p2 + p(1)*x./x_plus_p2_sq;
        dfdp = [-x./x_plus_p2;  p(1)*x./x_plus_p2_sq];

        Spdot = dfdx*Sp + dfdp;

        zdot = [xdot; Spdot(:)];
    end

    [T, X] = ode45(@model, t, z0, [], p);
    z = X;
end
```

Code Listing 22: ex23_z.m

```
function phi_fun = get_phi_fun(x, y, yhat_fun)

    function phi = calc_phi(theta1, theta2)
        yhat = yhat_fun(x, [theta1, theta2]);
        phi = 0.5*sum((y-yhat).^2);
    end

    phi_fun = @calc_phi;

end
```

Code Listing 23: get_phi_fun.m

```
function [r J] = residual_jacobian_ex23(theta, x, y)
    Z = ex23_z(x, theta);

    r = y - Z(:,1);
    J = -Z(:,2:3);
end
```

Code Listing 24: residual_jacobian_ex23.m

A.8 Helper functions

```
function [] = saveeps(filename)
    print('-depsc', '-loose', filename);
end
```

Code Listing 25: saveeps.m

```
function [] = saveline(filename, linetemplate, variables)
    f = fopen(['../tables/', filename], 'w');
    fprintf(f, linetemplate, variables);
    fclose(f);
end
```

Code Listing 26: saveline.m

References

[1] Jorge Nocedal & Stephen J. Wright, *Numerical Optimization.* Springer Science+Business Media, 2nd Edition, 2006.

[2] Kaj Madsen & Hans Bruun Nielsen, *Introduction to Optimization and Data Fitting.* DTU IMM, 1st Edition, 2010.

[3] Hans Bruun Nielsen, *Checking Gradients.* DTU IMM, 1st Edition, 2000, `http://www2.imm.dtu.dk/~hbn/Software/checkgrad.ps`.