Computernumerik - Exercise 2/Task 3

[redacted]

May 8, 2016

1 Aufgabe 3 - Interpolation:

In TUWEL steht eine Datei daten.csv für Sie zum Downloaden bereit. Importieren Sie die Datei in Matlab. Interpretieren Sie die erste Spalte als N Stützstellen x_j und die zweite Spalte als dazugehörige Funktionswerte f_j.
Bestimmen Sie zu den Stützpunkten dasjenige Polynom $p(x) = a_0 + a_1 x + a_2 x^2$ zweiten Grades, welches die Summe der Fehlerquadrate

$\sum_{j=1}^{N} (p(x_j) - f_j)^2$

minimiert. [2]

1.1 Quadratisches Ausgleichsproblem:

Wenn man aus einer Messung einen Datensatz mit N Samples zu den Messzeitpunkten $x_1, x_0, ..., x_N$ und den Messwerten $\tilde{f}(x_1), \tilde{f}(x_2), ..., \tilde{f}(x_N)$ hat, kann man aus diesen Werten durch Interpolation eine allgemeines quadratisches Polynom festlegen, das die gemessenen Daten anhand einer Metrik optimal annähert.

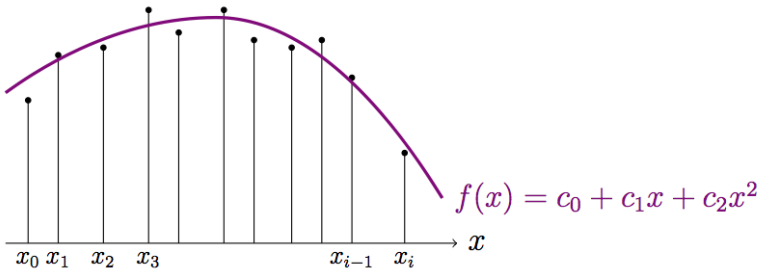

Figure 1: Allgemeines Quadratisches Polynom und Messwerte

In unserem Beispiel wollen wir ein Polynom finden das Summe der Fehlerquadrate minimiert. Das bedeutet wir suchen die Koeffizienten c_0, c_1 und c_2 für das Polynom $f(x) = c_0 + c_1 x + c_2 x^2$ die den Ausdruck

$E(c_0, c_1, c_2) = \sum_{j=1}^{N}(c_0 + c_1 x_i + c_2 x_i^2 - \tilde{f}(x_i))^2$

minimieren. Bildlich entspricht das dem Finden der Koeffizienten eines Polynoms das die Summe der Fläche der Quadrate, die durch den Abstand der Messwerte und des Polynoms aufgespannt werden, minimiert wird.

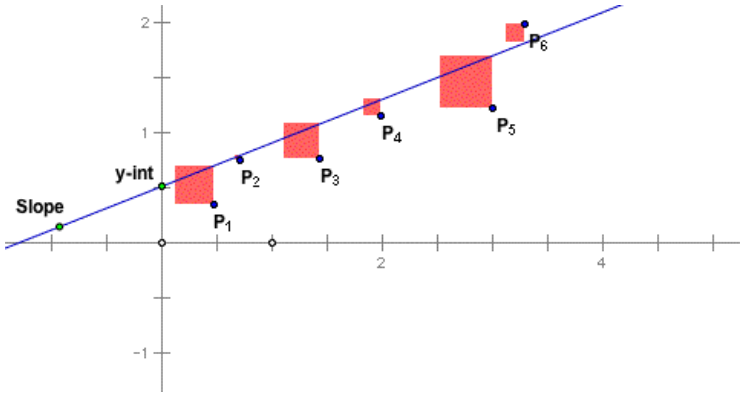

Figure 2: Fehlerquadrate von einer Geraden und Messwerten

Um die Extremwerte zu erhalten muss man die Ableitungen nach c_0, c_1, c_2 von $E(c_0, c_1, c_2)$ null setzen.

$\frac{\partial E(c_0,c_1,c_2)}{\partial c_0} = 0,\ \frac{\partial E(c_0,c_1,c_2)}{\partial c_1} = 0,\ \frac{\partial E(c_0,c_1,c_2)}{\partial c_2} = 0$

Durch Ableiten und Umformen erhält man ein Gleichungssystem mit 3 Gleichungen und 3 Unbekannten.

$\sum_{j=1}^{N}(c_0 + c_1 x_i + c_2 x_i^2 - \tilde{f}(x_i)) = 0$
$\sum_{j=1}^{N}(c_0 + c_1 x_i + c_2 x_i^2 - \tilde{f}(x_i))x_i = 0$
$\sum_{j=1}^{N}(c_0 + c_1 x_i + c_2 x_i^2 - \tilde{f}(x_i))x_i^2 = 0$

Durch weitere Umformung erhält man ein Gleichungssystem, welches man auch als Matrix anschreiben kann.

$(N+1)c_0 + (\sum_{j=1}^{N} x_i)c_1 + (\sum_{j=1}^{N} x_i^2)c_2 = \sum_{j=1}^{N} \tilde{f}(x_i)$
$(\sum_{j=1}^{N} x_i)c_0 + (\sum_{j=1}^{N} x_i^2)c_1 + (\sum_{j=1}^{N} x_i^3)c_2 = \sum_{j=1}^{N} x_i\tilde{f}(x_i)$
$(\sum_{j=1}^{N} x_i^2)c_0 + (\sum_{j=1}^{N} x_i^3)c_1 + (\sum_{j=1}^{N} x_i^4)c_2 = \sum_{j=1}^{N} x_i^2\tilde{f}(x_i)$

Man sieht sofort das man das System auch als $M * \vec{x} = \vec{b}$ auffassen kann und diese System einfach lösen kann.

$$M = \begin{pmatrix} m+1 & \sum_{j=1}^{N} x_i & \sum_{j=1}^{N} x_i^2 \\ \sum_{j=1}^{N} x_i & \sum_{j=1}^{N} x_i^2 & \sum_{j=1}^{N} x_i^3 \\ \sum_{j=1}^{N} x_i^2 & \sum_{j=1}^{N} x_i^3 & \sum_{j=1}^{N} x_i^4 \end{pmatrix}, \vec{b} = \begin{pmatrix} \sum_{j=1}^{N} \tilde{f}(x_i) \\ \sum_{j=1}^{N} x_i\tilde{f}(x_i) \\ \sum_{j=1}^{N} x_i^2\tilde{f}(x_i) \end{pmatrix}$$

und $\vec{x} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}$ als gesuchter Vektor.

1.2 Matlab Code:

Der folgende Matlab code führt die in Abschnitt 1.1 erklärte Methode durch.

```
%% Load CSV Data
data = csvread('daten.csv');

%% Retrieve Data Columns
x = data(:,1); % x_j
fx = data(:,2); % f_j

%% Equation Matrix
%%% Equation system calculated by calculating the maxima of the least
%%% squared error saved in Matrix M
M = [ size(x,1)+1,    sum(x),     sum(x.^2);
       sum(x),        sum(x.^2),  sum(x.^3);
       sum(x.^2),     sum(x.^3),  sum(x.^4)];
%% Result Vector
%%% Vector b calculated by doing mathematical transformations of the
%%% equation system as described in Section 1.1
sol = [sum(fx); sum(x.*fx); sum(x.^2.*fx)];

%% Solving the Equation System
%%% c := b in Section 1.1
c = M\sol;

%% Preparing Quadratic Approximation
%%% Creating a linearly spaced set of points and calculate the ...
    f_j(x) at
%%% the given points
x_points = linspace(min(x), max(x), size(x,1));
fit = c(1) + c(2) .* x_points + c(3) .* x_points.^2;

%% Plotting Approximation of Least Squared Error and Measured ...
    Sample Points
%%% Plot the sample points and the calculated polynome given by the
%%% calculation of the minimum of the squared errors.
figure(1)
hold on
scatter(x, fx);
plot(x_points, fit, 'r');
hold off
```

Die Ergebnisse mit dem CSV als Inputdaten sind zu sehen in Figure 3.

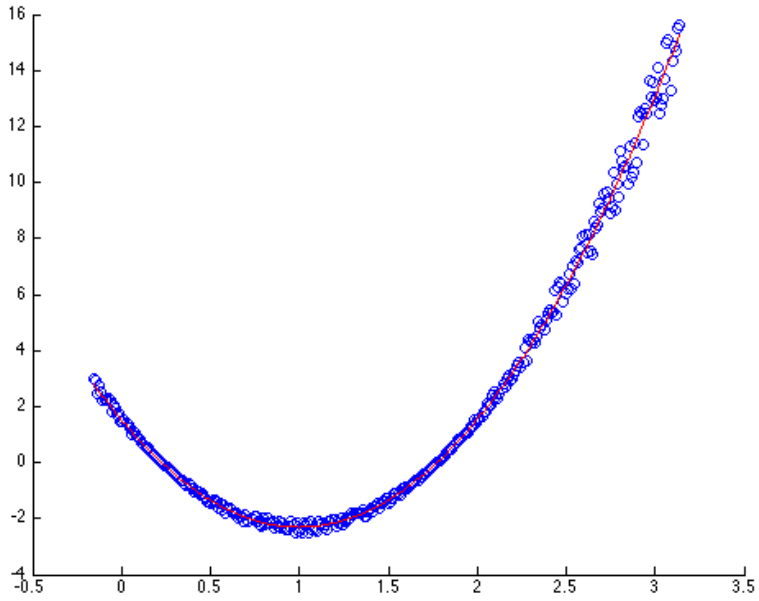

Figure 3: CSV Daten Scatterplot mit Polynom das die quadratischen Fehlersummen minimiert.

Anstatt das ganze selbst in Matlab zu implementieren kann man auch ganz einfach die Funktion *polyfit* [1] verwenden bei der man lediglich den Grad des Polynoms sowie die Messdaten angeben muss.

References

[1] Polyfit matlab documentation. `https://de.mathworks.com/help/matlab/ref/polyfit.html`. Accessed: 2016-05-08.

[2] Gabriela Schranz-Kirlinger. Computernumerik und numerische mathematik, Februar 2014.