Ü 5 Fünfte Übungseinheit

Inhalt der fünften Übungseinheit:

- Nichtlineare Datenmodelle
- Kurze Wiederholung: lineare und nichtlineare Datenmodelle
- Bonus-Material: MATLAB-Werkzeuge zum Anpassen von Funktionen an Daten
- Alternativen zur Minimierung der Fehlerquadrate: Robuste Regression,
- Anwendungen der Singulärwert-Zerlegung: Total Least Squares, Datenkomprimierung

Ü 5.1 Überbestimmte nichtlineare Systeme, Gauß-Newton-Verfahren

Das Gauß-Newton-Verfahren findet Näherungslösungen (im Sinn der kleinsten Fehlerquadrate) für überbestimmte nichtlineare Systeme. Die Grundidee ist, ähnlich wie bei der Lösung nichtlinearer Gleichungssysteme mit dem Newton-Verfahren, mit der *Jacobimatrix* des nichtlinearen Systems iterativ Korrekturterme zu berechnen. Die Aufgaben 41 und 42 zeigen ausführlich den Rechenweg.

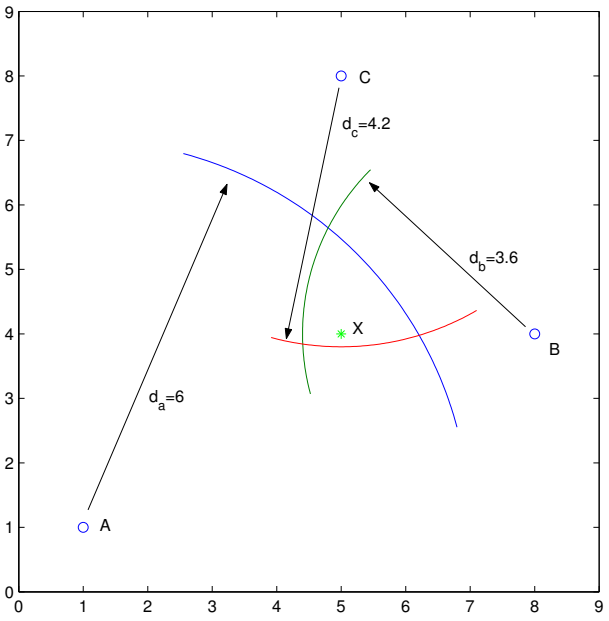
Aufgabe 41: Standortbestimmung durch Trilateration

Die Abstände von drei festen Punkten A, B, C in der xy-Ebene zu einem unbekannten Punkt X sind (etwas ungenau) bekannt. Gesucht ist eine möglichst gute Positionsbestimmung.

Punkt	x	y	Entfernung
	1	1	6
	8	4	3,6
	5	8	4,2

Die entsprechenden Gleichungen lauten:

$$\begin{aligned} \sqrt{(x_1-1)^2+(x_2-1)^2} &= 6 \\ \sqrt{(x_1-8)^2+(x_2-4)^2} &= 3.6 \\ \sqrt{(x_1-5)^2+(x_2-8)^2} &= 4.2 \end{aligned}$$



Den drei Gleichungen entsprechen drei Kreise im $\mathbb{R}^2$. Sie haben keinen gemeinsamen Schnittpunkt. Dementsprechend gibt es keine exakte Lösung des überbestimmten Gleichungssystems.

Schreiben Sie ein MATLAB-Programm, das die kleinste-Quadrate-Anpassung für den Standort findet.

Bemerkung: Diese Aufgabe ist die 2-dimensionale Vereinfachung einer Positionsbestimmung im Raum. Die GPS-Technik beruht auf diesen geometrischen Grundlagen. Ein GPS-Empfänger misst die Abstände (genauer: Signallaufzeiten) zu mehreren Satelliten. Die Signale sind stark

verrauscht, erst Ausgleichsrechnung und zusätzliche Datenfilterung gewährleisten eine auf einige Meter genaue Position.

Kurzfassung: Aufgabenstellung und Lösungsweg

Gegeben: überbestimmtes nichtlineares System

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}\ , \quad \mathbf{x} \in \mathbb{R}^n\ , \quad \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m, \quad m > n$$

Iterative Lösung des linearisierten Systems: Ausgehend von Startvektor $\mathbf{x}^{(0)}$ bestimmt man eine Korrektur $\Delta\mathbf{x}$.

Die Rechenvorschrift des Newton-Verfahrens für $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ ergibt ein überbestimmtes lineares System mit der Jacobimatrix D_f

$$D_f \cdot \Delta\mathbf{x} = -\mathbf{f}(\mathbf{x})$$

Verbesserte Lösung $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x}$.

Konkret für die vorliegende Aufgabe:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \sqrt{(x_1-1)^2+(x_2-1)^2} - 6 \\ \sqrt{(x_1-8)^2+(x_2-4)^2} - 3.6 \\ \sqrt{(x_1-5)^2+(x_2-8)^2} - 4.2 \end{bmatrix}, \qquad D_f = \begin{bmatrix} \frac{x_1-1}{\sqrt{(x_1-1)^2+(x_2-1)^2}} & \frac{x_2-1}{\sqrt{(x_1-1)^2+(x_2-1)^2}} \\ \frac{x_1-8}{\sqrt{(x_1-8)^2+(x_2-4)^2}} & \frac{x_2-4}{\sqrt{(x_1-8)^2+(x_2-4)^2}} \\ \frac{x_1-5}{\sqrt{(x_1-5)^2+(x_2-8)^2}} & \frac{x_2-8}{\sqrt{(x_1-5)^2+(x_2-8)^2}} \end{bmatrix}$$

Mit Startvektor $\mathbf{x} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$ erhält man

$$\mathbf{f}\left(\begin{bmatrix} 5 \\ 4 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ -3/5 \\ -1/5 \end{bmatrix}, \ D_f = \begin{bmatrix} \frac{4}{5} & \frac{3}{5} \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \ \text{lin. Syst.} \begin{bmatrix} \frac{4}{5} & \frac{3}{5} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/5 \\ 1/5 \end{bmatrix}$$

Ergibt $\Delta x_1 = 1/25, \Delta x_2 = 7/25 \quad \longrightarrow$ verbesserte Position $[5.04; 4.28]$.

Sie wiederholen diese Rechenschritte und iterieren, bis sich die Position im Rahmen einer vernünftig gewählten Fehlerschranke nicht mehr ändert. Vergleichen Sie dazu den Übungs-Abschnitt Ü 3.4: der Rechengang und vor allem auch die Implementierung in MATLAB sind nahezu gleich. Unterschied: die Matrix des Gleichungssystems ist nun nicht mehr quadratisch und auch nicht konstant; sie hängt von den jeweils aktuellen Näherungswerten für $\mathbf{x}$ ab.

Natürlich sagt dieses Kochrezept nichts zur Theorie überbestimmter linearer Systeme oder zu den Eigenschaften der so berechneten Ausgleichslösung. Die Idee, ein nichtlineares System mittels Jacobi-Matrix linearisiert anzunähern, wird Ihnen aber in der Praxis ständig begegnen.

Aufgabe 42: Gauß-Newton-Verfahren in Wikipedia

Die Angabe zum folgenden Beispiel stammt aus der englischen bzw. französischen Wikipedia (Stichworte *Gauss-Newton algorithm* bzw. *Algorithme de Gauss-Newton*). Lösen Sie die Aufgabe mit folgender Anleitung in MATLAB.[17]

[17] Wer dieses Beispiel in der englischen oder französischen Wikipedia-Seite durchliest und bei der Abbildung der Modellkurve auf "more details" klickt, findet fertigen MATLAB-Code zur Lösung des Beispiels und Zeichnen der Kurve. Aber das Verstehen von fremdem Code ist auch nicht einfach.

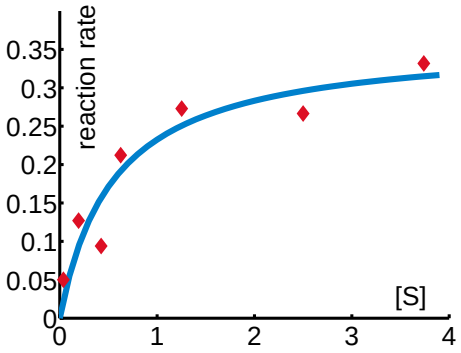
Biologie-Experimente zur Beziehung zwischen Substanz-Konzentration x und Reaktionsrate y in einer durch Enzyme vermittelten Reaktion haben die Daten in folgender Tabelle ergeben:

x	0.038	0.194	0.425	0.626	1.253	2.500	3.740
y	0.05	0.127	0.094	0.2122	0.2729	0.2665	0.3317

Gesucht ist eine Kurve (Modellfunktion) der Form

$$y = a\frac{x}{b+x}$$

die im Sinn der kleinsten Quadrate die Daten am besten approximiert. Die Parameter a und b sind zu bestimmen.
Einsetzen der Daten ergibt sieben nichtlineare Gleichungen in den beiden Unbekannten a, b.



$$a\frac{0.038}{b+0.038} - 0.05 = 0$$
$$a\frac{0.194}{b+0.194} - 0.127 = 0$$
$$\vdots$$
$$a\frac{3.740}{b+3.740} - 0.3317 = 0$$

Das Gleichungssystem in Vektor-Schreibweise:

$$\mathbf{f}(\mathbf{x}) = 0 \text{ mit } \mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^7$$

Die Jacobi-Matrix D_f dieses Systems ist eine 7×2-Matrix. Zeile i enthält die partiellen Ableitungen der i-ten Gleichung nach den Unbekannten a und b.

$$(D_f)_{i1} = \frac{x_i}{b+x_i}, \quad (D_f)_{i2} = -\frac{ax_i}{(b+x_i)^2}$$

Der weitere Rechenweg verläuft völlig analog zur Anleitung im Übungs-Abschnitt Ü 3.4 und der Musterlösung auf Seite 27 im Skriptum: Wählen Sie als Startwert `[a;b]=[0.9;0.2]`, werten Sie $\mathbf{f}$ und D_f aus. Die kleinste-Quadrate-Näherung aus dem überbestimmten Gleichungssystem $D_f\mathbf{\Delta x} = -\mathbf{f}$ liefert den Korrekturvektor $\mathbf{\Delta x}$.

Beachten Sie: bei einer quadratischen Jacobi-Matrix liefert der MATLAB-Befehl `Df(x0)\f(x0)` die Lösung des Gleichungssystems; hat die Jacobi-Matrix mehr Zeilen als Spalten (überbestimmtes System), liefert derselbe Befehl *nicht* die Lösung (es gibt keine exakte Lösung!), sondern eine Anpassung mit geringstmöglichem Restfehler (die „am wenigsten falsche Antwort“)!

Bonus-Frage: Unser Newton-Musterprogramm findet die verbesserte Näherung mit dem MATLAB-Befehl

```
x = x0 -  Df(x0)\f(x0);     % Newton-Schritt
```

Das Wikipedia-Beispiel nennt die Jacobi-Matrix J_f und verwendet für denselben Schritt (in MATLAB-Code geschrieben) den Befehl

```
x = x0 - (J'*J)\(J'*f(x0));
```

Warum einerseits `Df(x0)\f(x0)` und andererseits `(J'*J)\(J'*f(x0))`? Was wird da jeweils gerechnet? Vor- und Nachteile der beiden Varianten im Vergleich?

Aufgabe 43: Kalorimeterversuch

Der Datensatz `Kalorimeter.dat` auf der Übungsseite enthält Wertepaare (Zeit t in Minuten, Temperatur T in Celsius) zu einem Kalorimeterversuch, gemessen ab der Einbringung eines Versuchsobjektes. Die Temperatur sinkt zuerst rasch, steigt dann aber wieder und gleicht sich langsam der Umgebungstemperatur an.

Der Verlauf der Temperatur T als Funktion der Zeit t soll durch ein Modell der Form

$$T(t) = T_0 + C_1 \exp(-\lambda_1 t) + C_2 \exp(-\lambda_2 t)$$

beschrieben werden. Bestimmen Sie die Parameter dieses nichtlinearen Modells ausgehend von den Startwerten

$$T_0 = 6 \qquad C_1 = -1 \qquad \lambda_1 = 1/10 \qquad C_2 = 20 \qquad \lambda_2 = 2$$

(drei Iterationen des Newton-Verfahrens reichen aus.) Zeichnen Sie Messpunkte und Modellfunktion.

Ü 5.2 Kurze Wiederholung: lineare und nichtlineare Datenmodelle

Die Aufgaben zur Anpassung von Funktionen an Daten beschränkten sich in der vorigen Einheit auf *lineare Modelle*. Dabei bedeutet „linear“: die gesuchte Anpassung f ist eine *Linearkombination* von n Basisfunktionen $\phi_1, \phi_2, \ldots, \phi_n$ in der Form

$$f = a_1\phi_1 + a_2\phi_2 + \cdots + a_n\phi_n \quad .$$

Wichtig: Die Basisfunktionen $\phi_1, \ldots, \phi_n$ können beliebige, auch nichtlineare Funktionen sein – es sind die gesuchten Koeffizienten $a_1, \ldots, a_n$, die nur linear im Ansatz auftreten!

Verwirrender Weise wird für die Anpassung einer Ausgleichs-Geraden auch der Begriff „lineare Regression“ verwendet. Unter „polynomiale Regression“ versteht man die Anpassung eines Polynoms an gegebene Daten, wobei es sich aber um ein lineares Datenmodell handelt.

Unterscheiden Sie:

- Typ der Anpassungsfunktionen: Gerade ("lineare Regression"), Polynom ("polynomiale Regression"), Winkel- oder Exponentialfunktionen...
- Verknüpfung der Ansatzfunktionen: linear, nichtlinear. Dazu gehören die Begriffe lineares/nichtlineares Datenmodell.

Vergleichen Sie: Lineare Modelle

Aufgabe 38: Ansatz $z(x, y) = a_1 + a_2x + a_3x^2 + a_4y$.
Hier sind die Basisfunktionen $\phi_1(x, y) = 1, \phi_2(x, y) = x, \phi_3(x, y) = x^2, \phi_4(x, y) = y$.

Aufgabe 39: Ansatz $t(d, T_0) = a_1 + a_2 d + a_3 T_0 + a_4 d T_0$.
Basisfunktionen sind $\phi_1(d, T_0) = 1, \phi_2(d, T_0) = d, \phi_3(d, T_0) = T_0, \phi_4(d, T_0) = d \cdot T_0$.

Aufgabe 40: Ansatz $y = a_1 + a_2 \cos\left(x\frac{\pi}{6}\right) + a_3 \sin\left(x\frac{\pi}{6}\right)$
Basisfunktionen sind $\phi_1(x) = 1, \phi_2(x) = \cos\left(x\frac{\pi}{6}\right), \phi_3(x) = \sin\left(x\frac{\pi}{6}\right)$.

Nichtlineare Modelle

Aufgabe 42: Ansatz $y = a\frac{x}{b+x}$. Hier sind die gesuchten Koeffizienten a und b nichtlinear verknüpft!

Aufgabe 43: Ansatz $T(t) = T_0 + C_1 \exp(-\lambda_1 t) + C_2 \exp(-\lambda_2 t)$.
Wären λ_1 und λ_2 bekannt, dann wäre es ein linearen Modell mit Parametern T_0, C_1 und C_2. Weil auch die beiden λ_i gesucht sind, wird die Aufgabe nichtlinear.

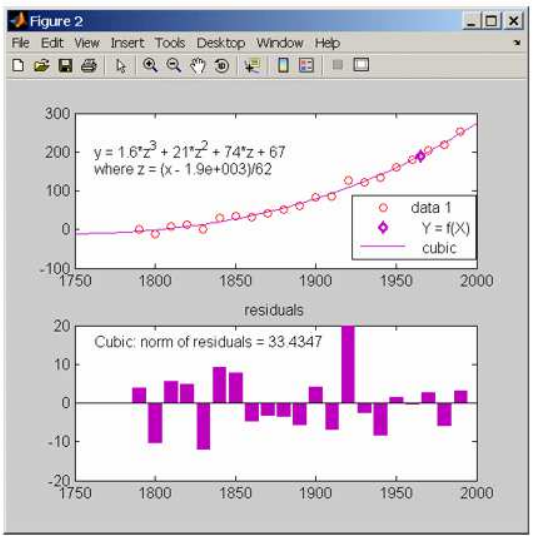
Kapitel Ü 5.1 in diesen Unterlagen zeigt einen Standard-Lösungsweg zur nichtlinearen Anpassung. Der folgende Abschnitt Ü 5.3 zeigt weitere Möglichkeiten und stellt MATLAB-Werkzeuge vor. Vielleicht finden Sie diese Werkzeuge einfacher und intuitiver zu verwenden als den Lösungsweg aus Kapitel Ü 5.1.

Ü 5.3 MATLAB-Werkzeuge zum Anpassen von Funktionen an Daten

(Bonus-Material für Interessierte: Im Vergleich zu älteren Versionen bietet MATLAB inzwischen tolle neue Werkzeuge. Schaut euch 's an, es zahlt sich aus!)

Zur MATLAB-Grundausstattung gehört das *Basic Fitting Tool*. Damit sollten Sie unbedingt umgehen können. Machen Sie sich gleich einmal damit vertraut: Finden Sie die entsprechende Anleitung in der MATLAB-Hilfe (Vers. 2021b) unter MATLAB › Data Import and Analysis › Descriptive Statistics › Interactive Fitting oder (besser, weil dieser Pfad in älteren Versionen anders lautet) suchen Sie direkt nach dem Stichwort *Interactive Fitting*.

Sie finden dort ein durchgearbeitetes Beispiel, das die Verwendung des *Basic Fitting Tools* für interaktives Anpassen von Kurven an Datenpunkte erklärt.



Arbeiten das Beispiel in der MATLAB-Hilfe durch und stellen Sie die Approximation graphisch dar, etwa so wie in der nebenstehenden Abbildung.

Darüber hinaus bietet MATLAB in der *Statistics and Machine Learning Toolbox* und in der *Curve Fitting Toolbox* viele Werkzeuge zur Datenanpassung, weit mehr als wir in diesen Übungen behandeln können. Die Aufgaben 44 und 46 zeigen beispielhaft, welche Möglichkeiten die Befehle `cftool` und `polytool` bieten. Aufgabe 47 erwähnt den Befehl `robustfit`.

Aufgabe 44: Basic Fitting Tool im Vergleich zu Curve Fitting Toolbox

Hier lösen Sie das nichtlineare Ausgleichsproblem aus Aufgabe 42 mit MATLAB-Toolboxen.

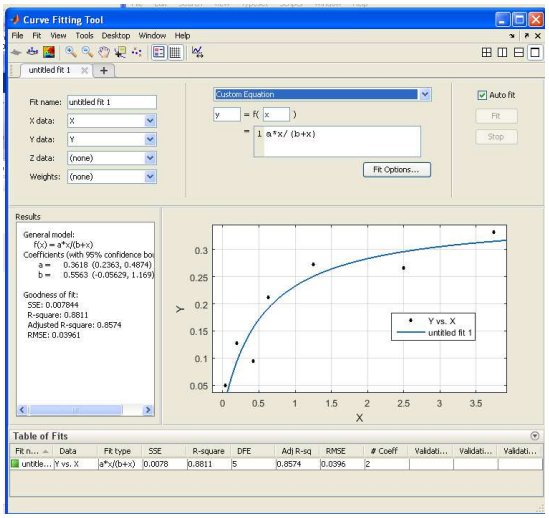
Legen Sie mit MATLABs *basic fitting tool* eine Ausgleichsgerade, ein quadratisches und dann ein kubisches Ausgleichspolynom durch folgende Datenpunkte und plotten Sie die Ergebnisse.

```
X = [0.038 0.194 0.425  0.626  1.253  2.500  3.740];
Y = [0.05  0.127 0.094 0.2122 0.2729 0.2665 0.3317];
```

Es handelt sich hier um die Daten aus Aufgabe 42; dort finden Sie auch eine Abbildung mit gut angepasster Kurve. Welche anderen Optionen des *basic fitting tool* liefern plausible Kurven? Was bedeuten die Fehlermeldungen, die Sie ab Grad 7 bekommen?

Es stellt sich heraus: Kein Ausgleichspolynom kann die Datenpunkte befriedigend approximieren. Das liegt teils daran, dass die Datenpunkte aufgrund der Messunsicherheit weit gestreut sind, aber auch daran, dass sich im gemessenen Experiment die y-Werte mit zunehmendem x asymptotisch einem konstanten Wert nähern. Kein Polynom kann so ein asymptotisches Verhalten beschreiben.

Der Befehl `cftool` öffnet die *Curve Fitting App*. Hier können Sie neben Polynomen auch weitere Modellfunktionen anpassen. Versuchen Sie es: Wählen Sie links oben unter *X Data* und *Y Data* die entsprechenden Datenvektoren aus den im Workspace vorhandenen Variablen. Oben Mitte können Sie den Funktionstyp wählen. Probieren Sie zuerst *Polynomial* mit verschiedenen Polynom-Graden. Wählen Sie anschließend *Custom Equation* und geben Sie den Funktionstern `a*x/(b+x)` ein. (Das ist die Modellfunktion aus Aufgabe 42!) Ihre Anpassung sollte so aussehen wie hier gezeigt.



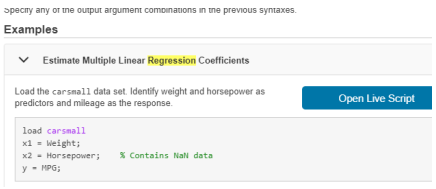
Zusätzlich zu den berechneten Werten der Koeffizienten a und b liefert MATLAB auch ein Konfidenzintervall. Interpretation: Falls den Daten tatsächlich ein Modell der Form $y = ax/(b + x)$ zugrunde liegt, aber die y-Daten zufallsbedingt verrauscht sind, dann berechnet MATLAB *Schätzungen* $\hat{a}$ und $\hat{b}$ für die Parameter a beziehungsweise b. Zum Beispiel:

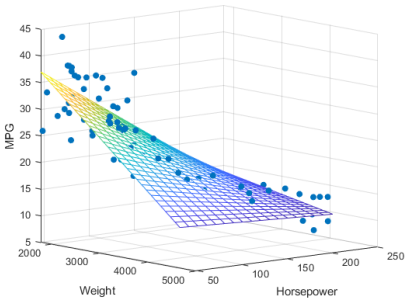
```
Coefficients (with 95% confidence bounds):
       a =      0.3618  (0.2363, 0.4874)
       b =      0.5563  (-0.05629, 1.169)
```

Die Interpretation „Die tatsächlichen Werte von a und b liegen mit 95%-iger Sicherheit im berechneten Intervall“ ist so nicht korrekt; die tatsächlichen Werte von a und b liegen entweder drinnen oder nicht – da ist kein Spielraum für Wahrscheinlichkeiten. Es ist umgekehrt: die berechneten Grenzen des Konfidenzintervalls sind unsicher. Jede unabhängige Wiederholung der Messung liefert (für die immer gleichen Modellparameter a und b) einen neuen Satz von Datenpunkten mit anderen zufallsbedingten Fehlern. MATLABs Schätz-Methode berechnet dazu Konfidenzintervalle mit jeweils etwas anderen Grenzen. Meistens (in 95% der Fälle) schätzt

Matlab die Fehlergrenzen richtig, aber in 5% der Fälle überdeckt das berechnete Konfidenzintervall die wahren Werte nicht.

Aufgabe 45: Der Befehl *regress* aus der Statistics Toolbox





Die Matlab-Hilfe Zum Stichwort „regress“ zeigt ein Beispiel mit Daten zu Gewicht, Motorleistung und Benzinverbrauch von Autos und findet dazu eine Anpassungs-Funktion. Vergleichen Sie mit Aufgabe 39: dort wird ein Datenmodell derselben Art berechnet, die Form der Matrix ist gleich.

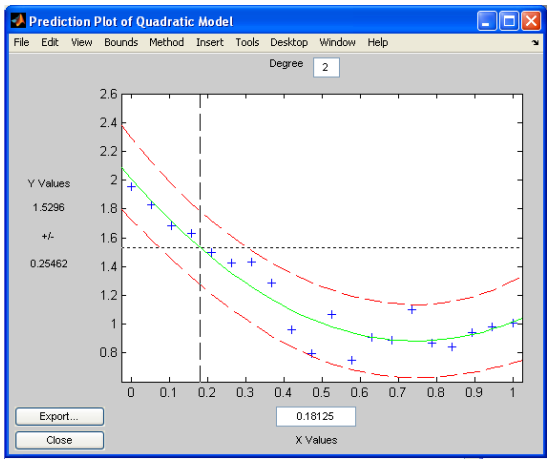
Sie können in der MATLAB-Hilfe auf *Open Live Script* klicken und gelangen so in den *Live Editor*, der mehr Möglichkeiten bietet als der Standard-Editor. Die Dateiendung `*.mls` kennzeichnet Live-Skript-Dateien.
Wenn Sie neugierig sind, probieren Sie den Live-Editor aus. Wenn Sie in der gewohnten Arbeitsumgebung bleiben wollen, speichern Sie dieses Musterbeispiel mittels *save as* als Dateityp *MATLAB Code Files*, Dateiendung `*.m`

Orientieren Sie sich an diesem Muster und erstellen Sie für die Daten aus Aufgabe 39 (Durchmesser, Ausgangstemperatur, Kochzeit) mit dem `regress`-Befehl eine Anpassung und eine ähnliche Graphik wie oben (mit Kochzeit als z-Achse).

Aufgabe 46: Der Befehl *polytool* aus der Statistics toolbox

Die folgenden MATLAB-Befehlszeilen erzeugen entlang der Funktion $y = 2 - 3x + 2x^2$ Datenpunkte, die durch zufällige, normalverteilte Störungen verrauscht sind.

```
n=20;
x=linspace(0,1,n);
y=2 - 3*x + 2*x.^2 + 0.1*randn(1,n);
```



Der Befehl `polytool(x,y)` öffnet ein Fenster ähnlich dem basic fitting tool. Erzeugen Sie damit ein Bild wie die nebenstehende Abbildung und erklären Sie:

- Was bedeuten die verschiedenen Kurven?
- Wozu dient das verschiebbare Achsenkreuz, was sind die angezeigten X-Values und Y-Values?
- Wenn Sie auf Export... klicken, was bedeuten „Parameters“ und Parameters CI“?

Ü 5.4 Robuste Regression

Aufgabe 47: Robuste Regression

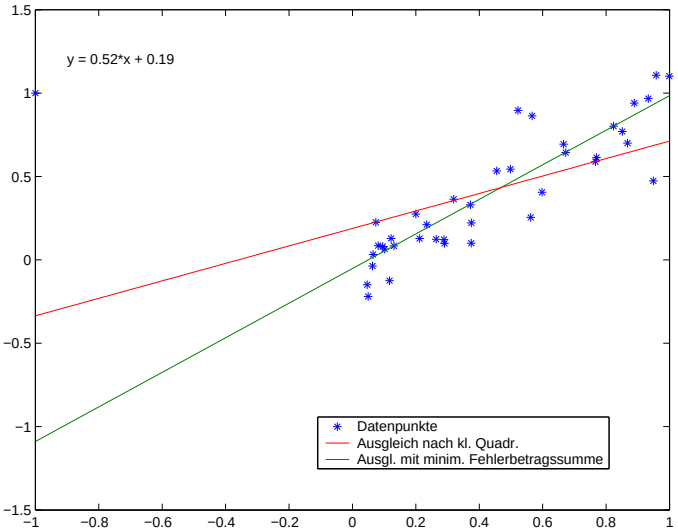
Lineare Regression nach der Methode der kleinsten Quadrate reagiert empfindlich auf Ausreißer in den Datenpunkten. Die folgenden MATLAB-Befehle erzeugen einen Datensatz von Punkten, die–bis auf einen Ausreißer–annähernd linearen Trend zeigen.

```
x = rand(40,1);
y = x + 0.2*randn(40,1);        Die Funktion randn erzeugt normalverteilte Abweichungen!
x(1) = -1; y(1)=1;
```

Schreiben Sie eine Script-Datei, die einen solchen Datensatz erzeugt und führen Sie folgende Schritte durch:

1. Stellen sie die Datenpunkte mit dem Symbol `'*'` in einem Diagramm dar.
2. Kapitel 6.3 des Vorlesungsskriptums beschreibt, wie man (nach der Methode der kleinsten Quadrate) die Parameter a und b der Ausgleichsgeraden $y = a + bx$ bestimmt. Implementieren Sie diese Formeln in Ihrem Skriptfile und berechnen Sie damit a und b.
3. Zeichnen Sie mit den Werkzeugen des Menues „Tools–Basic Fitting“ eine Ausgleichsgerade ein. Stellen Sie die Formel im Diagramm dar und vergleichen Sie mit den vorher berechneten Werten a und b.
4. Minimieren der Fehler*quadrate* ist das Standardverfahren zur Regression. Warum macht es keinen Sinn, die *Summe* der Fehler zu minimieren? Kapitel 6.6 des Vorlesungsskriptums beschreibt ein Regressionsverfahren, das die Summe der Fehler*beträge* minimiert. Dazu gibt es ein Musterprogramm als Datei `linregrob.m`. Zeichnen Sie die so bestimmte Gerade im Diagramm ein.

Ihr Diagramm sollte etwa so aussehen:



Übrigens: Die MATLAB-Hilfe zum Befehl `robustfit` aus der Statistics Toolbox zeigt als Beispiel einen ganz ähnlichen Fall wie die Aufgabe 47. Der Befehl `robustfit` bietet im Vergleich zur Datei `linregrob.m` mehr Optionen. Wenn es Sie interessiert, können Sie die Aufgabe 47 auch mit `robustfit` lösen und die Ergebnisse mit den `linregrob.m`-Resultaten vergleichen.

Auch `cftool` aus der *Curve Fitting Toolbox*, siehe Aufgabe 44 beherrscht robustes Fitting: wählen Sie dort *Polynomial, Degree 1, Robust Bisquare* oder *LAR*.