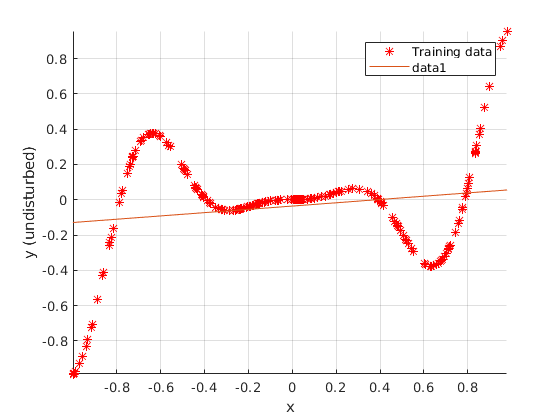
Dokumentation der Ergebnisse:

Aufgabe 1:

Ergebnis des Makros *'anlernen\_1D\_ungestoert.makrog'* ohne jegliche Einstellung (default Einstellungen)*:*



Frage 1:

Kann ein Polynom 1./2./3. Ordnung das Problem lösen?

Polynom 1. Ordnung:

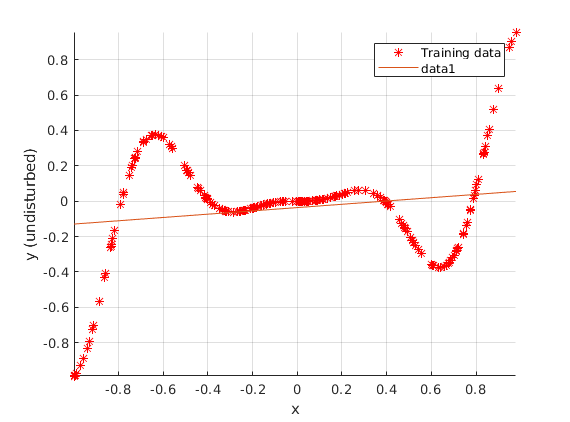
Fitness of regression:

Mean absolute value: 0.197196

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0

Correlation coefficient between true value and estimation: 0.174659

Coefficient of determination R^2 between true value and estimation: 0.0305058



Polynom 2. Ordnung:

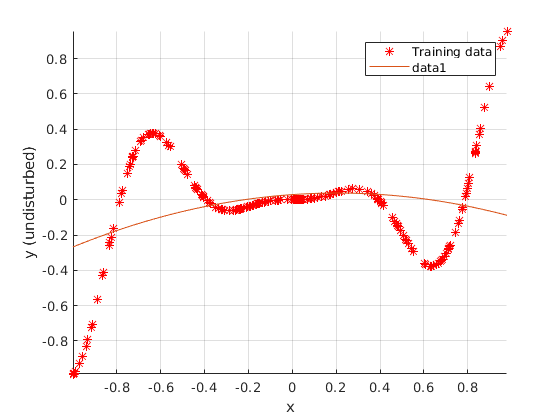
Fitness of regression:

Mean absolute value: 0.191139

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0.00273444

Correlation coefficient between true value and estimation: 0.267431

Coefficient of determination R^2 between true value and estimation: 0.0715193



Polynom 3. Ordnung:

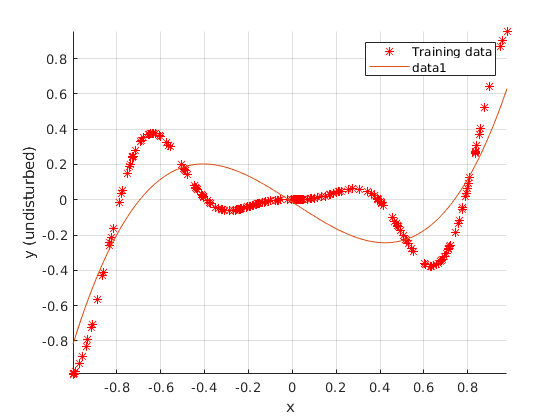
Fitness of regression:

Mean absolute value: 0.16439

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0.142296

Correlation coefficient between true value and estimation: 0.774918

Coefficient of determination R^2 between true value and estimation: 0.600498

*Analyse der Ergebnisse:*

Der eigentlich ausschlaggebende Wert für die Güte der Regression ist für das Polynom 3. Grades am besten mit einem '**mean absolute value'** von 0.16439. Demnach ist die beste Regression für unser Problem von 3. Ordnung. Schaut man sich allerdings dann die Plots zu den Regressionen an, erkennt man, dass das Polynom 3. Ordnung der durch die Messpunkte beschriebenen funktion am nächsten kommt, allerdings im Bereich von 0 +/-0.3 die lineare Regression wesentlich besser die Messpunkte abbildet

*Warum tritt dieser Effekt auf?*

Betrachtet man sich die Messpunkte genauer, so stellt man fest, dass im Bereich um 0 +/- 0.3 eine wesentlich höhere Dichte von Messpunkten vorzufinden ist als es im Bereich der y-Ausschläge der Fall ist. Somit nähert sich die lineare Regression in diesem Bereich wesentlich besser an die Messpunkte an als die 2. oder 3. Ordnung. (Da die weiter entfernten Messpunkte, mangels der Aufnahmefrequenz, keinen so großen Einfluss auf die Regression hat)

Aus diesem Grund ist auch der mean absolut value von 1. zu 3. Ordnung auch nicht so extrem groß wie man ihn möglicher Weise erwarten würde.

*Welche Schlussfolgerung lässt sich ableiten?*

Welches Regressionsmodell für eine Anwendung geeignet ist hängt vom Anwendungfall, bzw vom Arbeitsbereich ab. Für den Fall, dass in der Anwendung des entwickelten Algorithmus nur Werte zwischen +/- 0.3 vorkommen, ist eine lineare Regression die beste Approximation. Kommen auch Werte außerhalb dieses Bereichs vor, ist die Regression 3. Ordnung die bessere Wahl.

*Was lernen wir daraus?*

Nur weil eine Beurteilungsmetrik ein gutes Ergebnis ausgibt, muss dies nicht zwangsläufig heißen, dass das Ergebnis brauchbar ist. Eine Visualisierung ist in jedem Fall sinnvoll um das Problem zu verstehen und die Qualität eines Ergebnis zu interpretieren. Das Verlassen auf einen Wert nur einer Metrik kann zu Trugschlüssen führen.

Letztlich muss man ein Modell immer im Bezug zum Anwendungsfall auswählen. Das setzt aber ein Problemverständnis, als auch ein Verständnis der Lösung vorraus.

Frage 1:

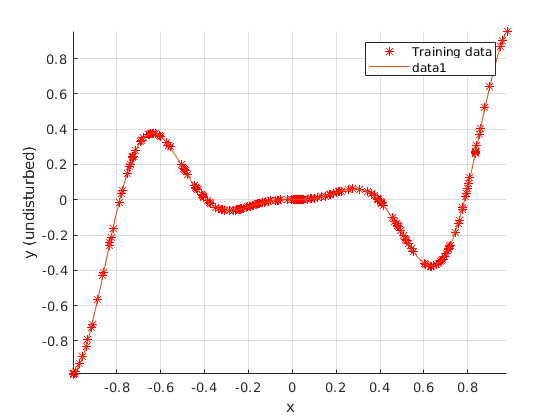
Kann ein Neuronales Netz das Problem lösen?

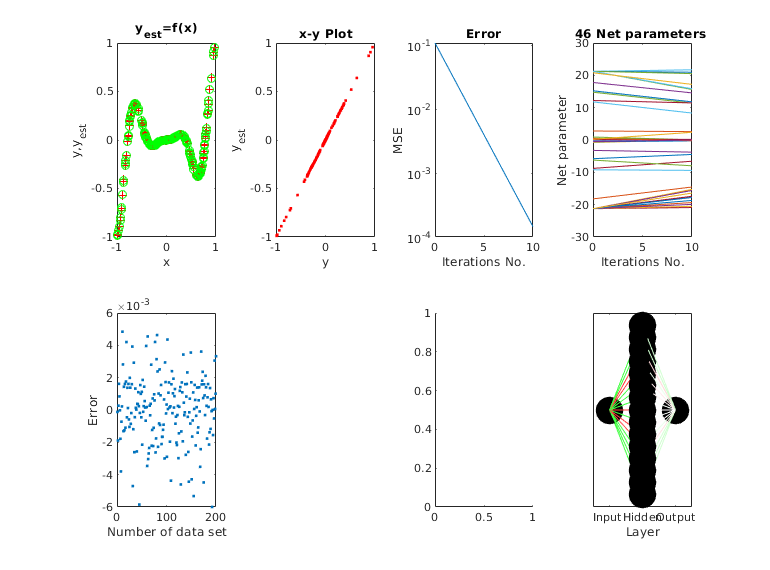
Mean absolute value: 0.000773872

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0.995962

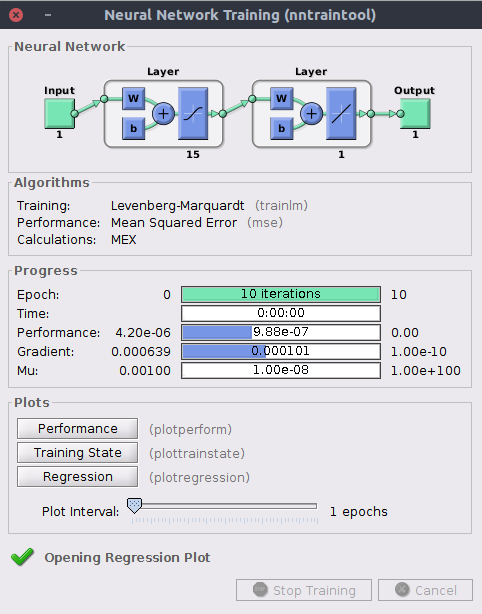
Correlation coefficient between true value and estimation: 0.999994

Coefficient of determination R^2 between true value and estimation: 0.999989





Man kann sehen, dass das default - Neuronale Netz die Messpunkte exakt wieder geben kann. Außerdem befindet sich der Trainingsfehler im Bereich von 10^-4. Die Netzparameter sind teilweise auf 0 trainiert und werden kleiner, was für ein erfolgreiches Training steht.



Frage 1:

Was sind geeignete Einstellungen für

* Anzahl Neuronen pro Schicht (Number of neurons per layer),
* Anzahl Lernepochen (Number of learning epochs),
* Fehlerziel (Error goal),
* MLP-Lernalgorithmus (MLP-Lernalgorithmus) und
* Normierung (Normalization, siehe Bild bei Frage 2)?

Anzahl Neuronen pro Schicht (20 Epochen):

4: mean absolute value: 0.0160823

5: mean absolute value: 0.00585046

6: mean absolute value: 0.00930023

Also sind 5 Neuronen extrem gut. Eine weitere Verbesserung stellt sich nicht maßgebend ein. Mehr neuronen bewirken nur mehr Rechenzeit.

Anzahl Lernepochen: (mit 5 Neuronen)

13: mean absolute value: 0.00589547

14: mean absolute value: 0.00578784

15: mean absolute value: 0.0059805

16: mean absolute value: 0.00594389

17: mean absolute value: 0.00592229

18: mean absolute value: 0.00586446

*Auffällig:* Je mehr neuronen ich habe, desto weniger Epochen brauche ich! Bei 3 Epochen ist das Ergebnis mit 6 Neuonen besser:

3 Epochen, 6 Neuonen: mean absolute value = 0.00169391

Außerdem sieht die geschätzte Funktion viel besser aus!

-> Wähle 14 Epochen, da sich danach nicht mehr viel an der Funktion tut und das Netz sogar schlechter wird.

-> Allgemein: Versuche die y-Daten und die y\_estimated (durch Netz geschätze Funktion) in lineare Abhängigkeit zu bringen. Das wird gut ab 14.

Auffällig:

Je mehr Neuronen, desto weniger Epochen brauche ich für ein brauchbares Ergebnis! Bei 10 Neuronen kann ich auf 11 Epochen runter.

-> Bessere Anpassungsfähigkeit durch mehr Freiheitsgrade.

-> Overfitting findet früher statt!

Fehlerziel:

Braucht 10^-5 für brauchbares Ergebnis!

-> Abbruch der Berechnung, wenn Netz konvergiert ist.

Optimierer:

1: Levenberg-Marquet, Quasi Newton funktionieren gut

2: RPROP, Gradientenabstieg und adaptive Backprob funktionieren nicht gut mit der gebenen Konfiguration!

-> 1) konvergieren schneller, sind aber rechenaufwendiger für sehr große Netze! (Mathematisch bedingt.)

-> Für 2) lässt sich keine zufriedenstellende Lösung mit den vorher getätigten Einstellungen finden. Auch bei Anpassung der Neuronen und Epochen und Errorziel wieder auf 0 setzen findet sich keine konvergierte Lösung.

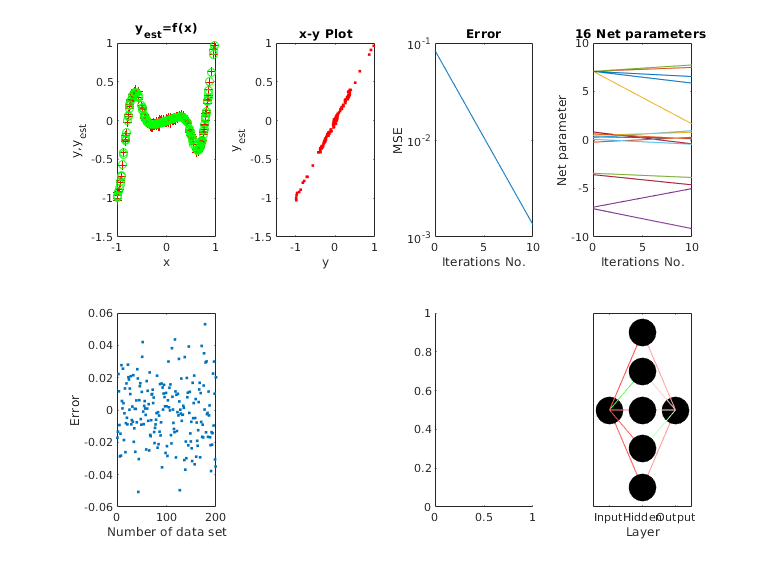
Normalisierung:

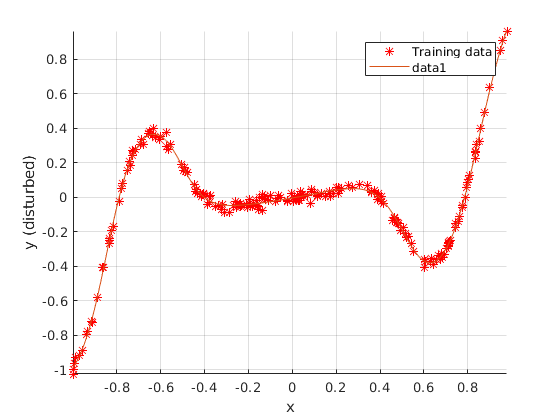
Bei den stabilen Verfahren aus 1) verändert sich nicht wirklich etwas durch eine zusätzliche Normalisierung.

Bei den nicht konvergierenden Verfahren 2) wird das Ergebnis etwas besser, wenn man Normalisierung anschaltet. Um ein einigermaßen akzeptables Ergebnis zu erhalten muss ich extrem viel mehr Epochen trainieren. (ca. 1000 für 7 Neuronen mit Gradien Decent und mean=0, STD=unchanged) [Wir wissen ja durch den Marquet-Algorithmus, dass wir mit 7 Neuronen eine zufriedenstellende Lösung finden können.]

Frage 4: Was passiert bei den besten Einstellungen beim Anlernen mit gestörten

Eingangsdaten?

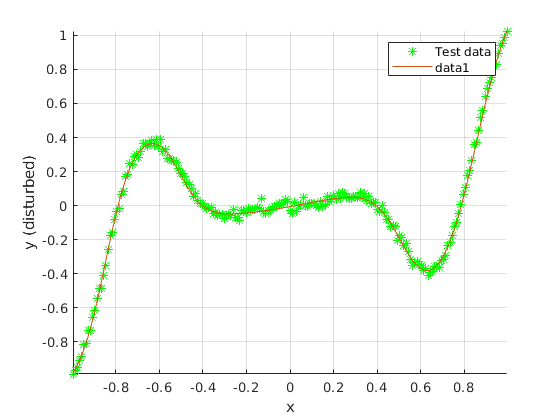


Man sieht, dass das Netz quasi eine Regression durch die verrauschten Punkte legt. Man könnte die Wirkung als Tiefpassfilterung bezeichnen. Das ist zu erwarten gewesen, da das Netz versucht die Informationen zu generalisieren.

In den Trainingsdaten ist auffällig, dass die Netzparameter teilweise divergieren.

Aufgabe 5: Was passiert bei den besten Einstellungen bei der Anwendung auf

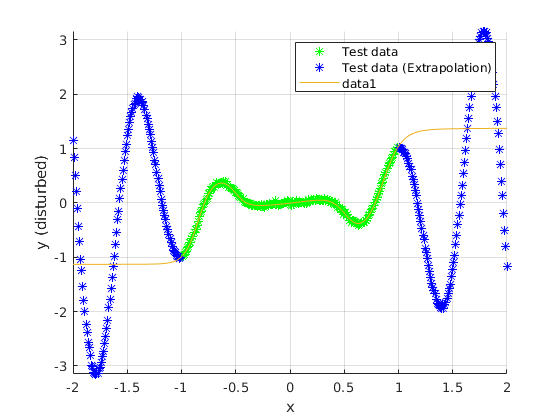
(unbekannte) Testdaten? Interpretieren Sie die Ergebnisse!

Die Daten werden gut durch das Netz repräsentiert. Die Abweichung ist gering und der Kurvenverlauf entspricht in etwa dem der Trainingsdaten.

Das ist zu erwarten, da alle Test-Daten in der Nähe der Daten, mit denen trainiert wurde liegen. Dem entsprechend kann das Netz die Daten in diesem Bereich extrem gut repräsentieren, da es darauf trainiert wurde dies zu können.

Aufgabe 6: Was passiert bei den besten Einstellungen bei der Anwendung auf

(unbekannte) Testdaten mit Extrapolation? Interpretieren Sie die Ergebnisse!



Im Bereich, in dem die Testdaten nahe bei den Trainingsdaten liegen funktioniert die Regression mit dem Netz sehr gut. Sobald eine Eingabe kommt, mit der das Netz nicht trainiert wurde, ist der Fehler extrem groß. (Bereich x = {< -1 & > 1})

-> Ich kann ein trainiertes Netz nur auf Daten anwenden, die es während des Trainings schonmal gesehen hat. Auf völlig unbekannten Eingabedaten, die es vorher noch nie gesehen hat kommen idR falsche Ausgaben raus!

-> Der Anwendungsbereich eines Netzes ist auf den Bereich beschränkt, der durch die Trainingsdaten abgedeckt ist.

Aufgabe 1 - Zweidimensionale Modellbildung:

Frage 1:

Kann ein Polynom 1., 2. oder 3. Ordnung das Problem lösen?

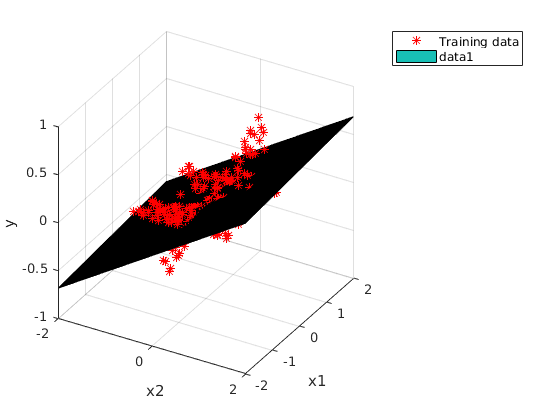
1. Ordnung:

Mean absolute value: 0.15798

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0

Correlation coefficient between true value and estimation: 0.656089

Coefficient of determination R^2 between true value and estimation: 0.430453



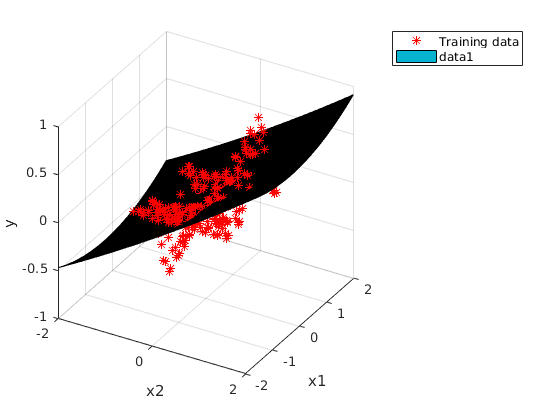
2. Ordnung:

Mean absolute value: 0.158011

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0

Correlation coefficient between true value and estimation: 0.658573

Coefficient of determination R^2 between true value and estimation: 0.433718



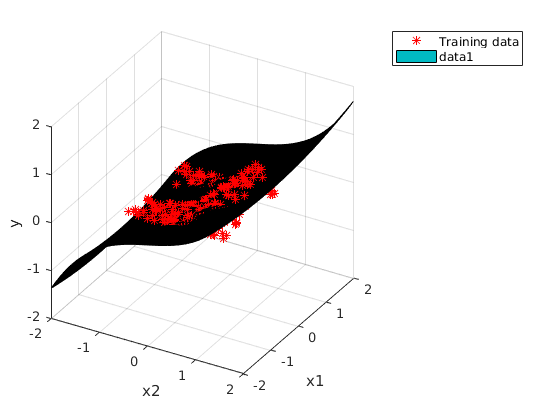
3. Ordnung:

Mean absolute value: 0.154345

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0

Correlation coefficient between true value and estimation: 0.662503

Coefficient of determination R^2 between true value and estimation: 0.43891

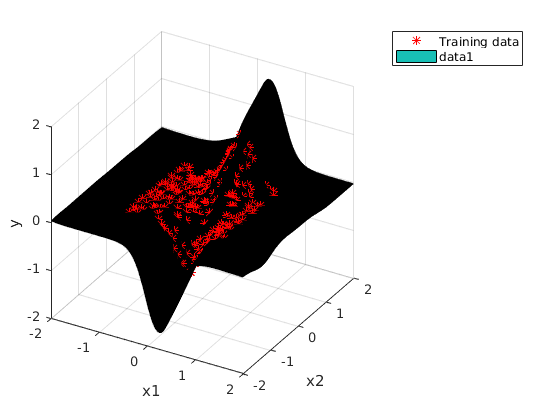
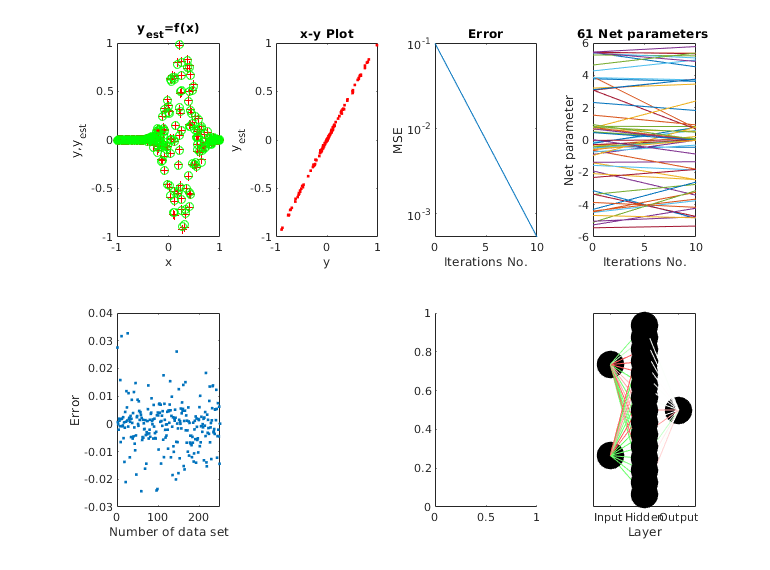
ANN:

Mean absolute value: 0.00483853

Fitness improvement for the mean absolute error to the trivial estimation (0-1): 0.967649

Correlation coefficient between true value and estimation: 0.999756

Coefficient of determination R^2 between true value and estimation: 0.999513



Interpretation:

Polynom:

Das Polynom 1. Ordnung hat eine bessere mean absolute value als das 2. Ordnung. Der Grund ist der selbe wie bei der eindimensionalen Variante: Die Dichte an Messpunkten im Bereich, in dem die Daten annähernd einen linearen Zusammenhang haben ist höher als an den Rändern, die eine wesentlich höhere Diffenz zwischen zwei Funktionswerten von Messpunkten aufweisen.

Da die Funktion die durch die Messpunkte abgebildet werden zusätzlich noch symetisch ist, legt sich die Ebene gut durch eine große Anzahl von Punkten im linearen Bereich. Dieser, sehr dicht abgebildete, lineare Bereich wird von den Funktionen höherer Ordnung nicht so gut repräsentiert. Daraus erklärt sich, dass das Polynom zweiten Grades etwas schlechter ist als das ersten Gerades.

ANN:

Das ANN lernt bei genügend Freiheitsgraden und Trainingsdaten eine gute repräsentation der Daten.

Aufgabe 3:

Aufgabe 1:

Eindimensionales Beispiel: Was passiert bei einer zu kleinen Anzahl von

Lerndaten (z.B. 10...20)? Datentupel 1- N manuell auswählen unter

Menu – Edit – Select – Data points using numbers

und dann Modell neu entwerfen

Set-Up:

Kann nicht mit (voreingestellten Datentupeln) Makros aufgerufen werden!

Berechnung:

Menü -> Data Mining -> Regression -> Apply

Visualisierung:

Menü -> View -> Regression -> Input Variables, Output Variables and Regression Function

Auswertung:

Polynom 3. Grades:

10 Datenpunkte

mean absolute value: 0.129886

20 Datenpunkte

mean absolute value: 0.164808

200 Datenpunkte

mean absolute value: 0.16439

Polynom 1. Grades:

10 Datenpunkte

mean absolute value: 0.15425

Polynom 2. Grades:

10 Datenpunkte

mean absolute value: 0,147373

20 Datenpunkte;

mean absolute value: 0.166291

Analyse:

Polynome:

Je mehr Datenpunkte ich habe, desto schlechter wird die Regression. Das liegt daran, dass die Funktion weniger Datenpunkten 'gerecht' werden muss. Dem entsprechend ist der mav niedriger bei kleinen Datenmengen.

ANN:

Bei extrem wenigen Datenpunkten (bis 5 Stück) kommen noch sehr ähnliche Ergebnisse zu der tatsächlichen Funktion raus! Das ist erstaunlich!

*Woran liegt das?*

Aufgabe 2:

Zweidimensionales Beispiel:

Fragen 3 und 5 für das eindimensionale Beispiel

TBD - nur wenn Zeit ist

Aufgabe 3:

Zeitreihenprognose für Vorhersage Energieverbrauch

(Projekt: u:\scixminer\prj\building.prjz),

TIPPS: Dokumentation lesen und Betreuer fragen....

TBD - nur wenn Zeit ist