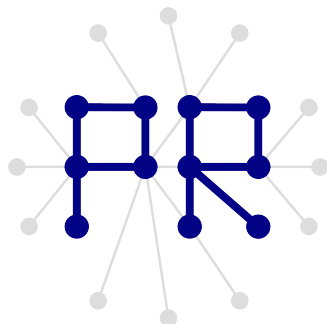
Einführung in die Informatik II

III.5 Optimierung

Prof. Dr.-Ing. Marcin Grzegorzek[1]

Forschungsgruppe für Mustererkennung
www.pr.informatik.uni-siegen.de

Institut für Bildinformatik
Universität Siegen



[1] Die im Rahmen dieser Lehrveranstaltung verwendeten Lernmaterialien wurden uns zum Großteil von Herrn Prof. Dr. Wolfgang Wiechert und Herrn Prof. Dr. Roland Reichardt zur Verfügung gestellt.

Inhaltsverzeichnis

Optimierung

Zunächst muss man sich über folgendes im Klaren sein:

- Was ist die Problemstellung?
- Was ist optimal?
- Was darf man ändern?

Beispiele:

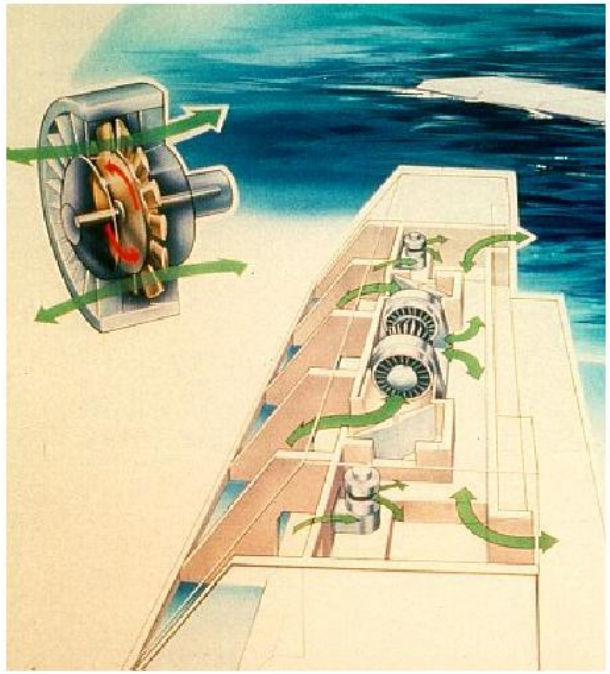
- BWL
- Maschinenbau
- Mathematik

Nutzung von Wellenenergie

Wasserspiegel steigt und sinkt
im umbauten Raum
Luft strömt durch Turbine

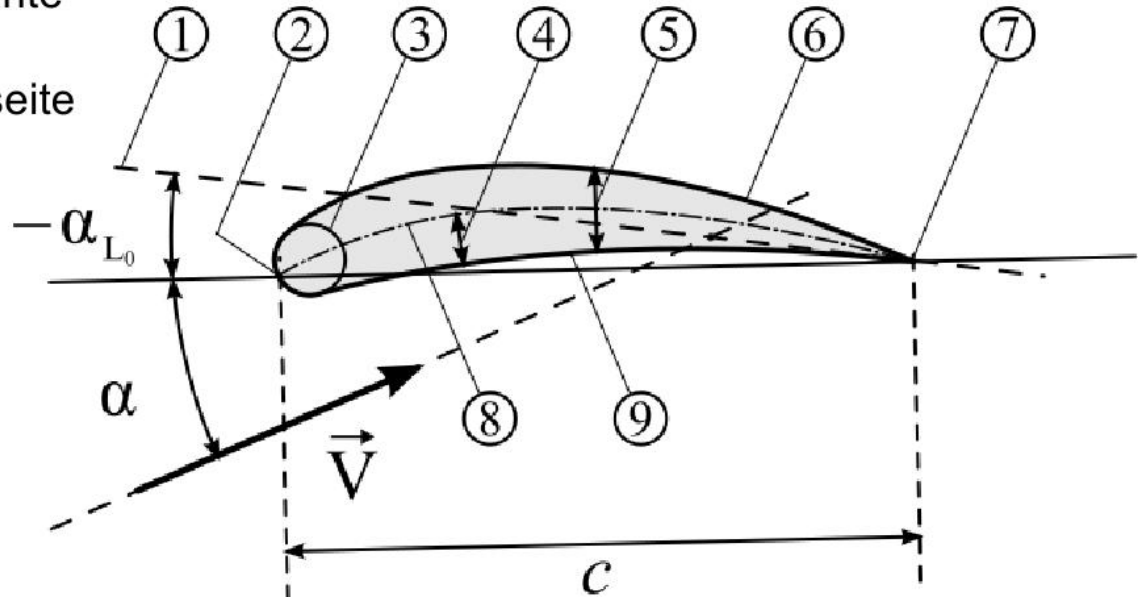
Ziel: Maximierung der
Wellenenergiewandlung

Änderung: Schaufelform



Geometrie eines NACA-Profils

1. Nullauftriebslinie
2. Flügelnase
3. Krümmungsradius der Flügelnase
4. Skelettlinie
5. maximale Profildicke
6. obere Profilseite
7. Profilhinterkante
8. Skelettlinie
9. untere Profilseite



Beispiel: Wellenkraftwerk

Messdaten:
- Wellenhöhe, Anströmgeschwindigkeit

Schaufelprofil:
- Simulation liefert Auftrieb

Optimierer:
- Profilform wird geändert, damit Auftrieb maximal wird.

Zielfunktion

Optimierung = **Minimierung** eines Funktionswertes

Den Funktionswert liefert die **Zielfunktion**

Die Zielfunktion kann **mehrere Parameter** haben.
Beispiel: Anströmgeschwindigkeit, Anströmwinkel, Profilform

Ziel der Optimierung formulieren

Vorsicht: Bei der Formulierung des Optimums werden schnell mehrere Ziele formuliert.
Beispiele:

- Das Bauteil soll eine **geringe Masse** und **hohe Festigkeit** haben.
- Das Produkt soll **preiswert** und **qualitativ hochwertig** sein.
- **Alle** Studierenden sollen in **Regelstudienzeit** das Studium mit **sehr guter Note** und **viel Fachwissen** abschließen.

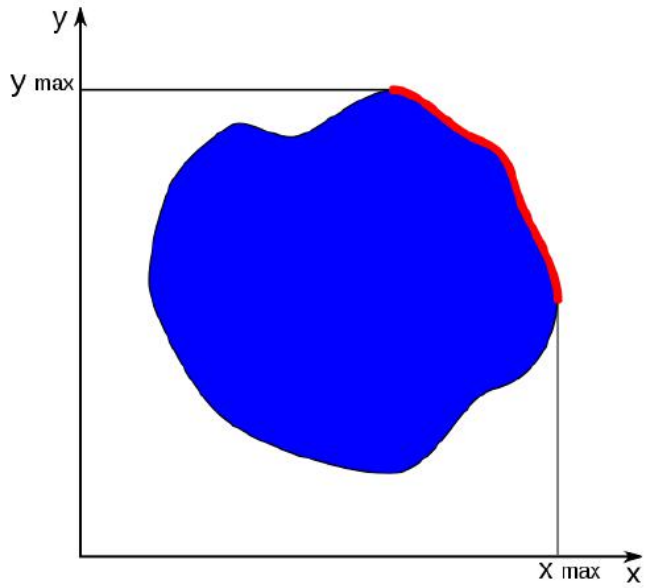
Fazit: Die Zielfunktion liefert einen Vektor als Ergebnis.

Pareto-Optimierung

Vektoroptimierung (auch Pareto-Optimierung genannt):
Die Werte mehrerer Zielfunktionen sind gleichzeitig zu optimieren.
Lösungen, die alle Komponenten der Zielfunktion gleichzeitig zu einem Optimum führen, sind in der Regel nicht vorhanden! Kompromiss

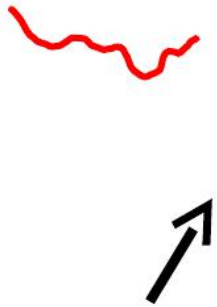
Beispiel:

x und y können nicht gleichzeitig maximal sein.



Lokal, Global

Bei Optimierungsproblemen kann es sein, dass man nicht das „Optimum“ findet.
Mit den Toolboxen von Matlab, die Ihnen zu Verfügung stehen, können Sie das globale Minimum nicht garantieren.

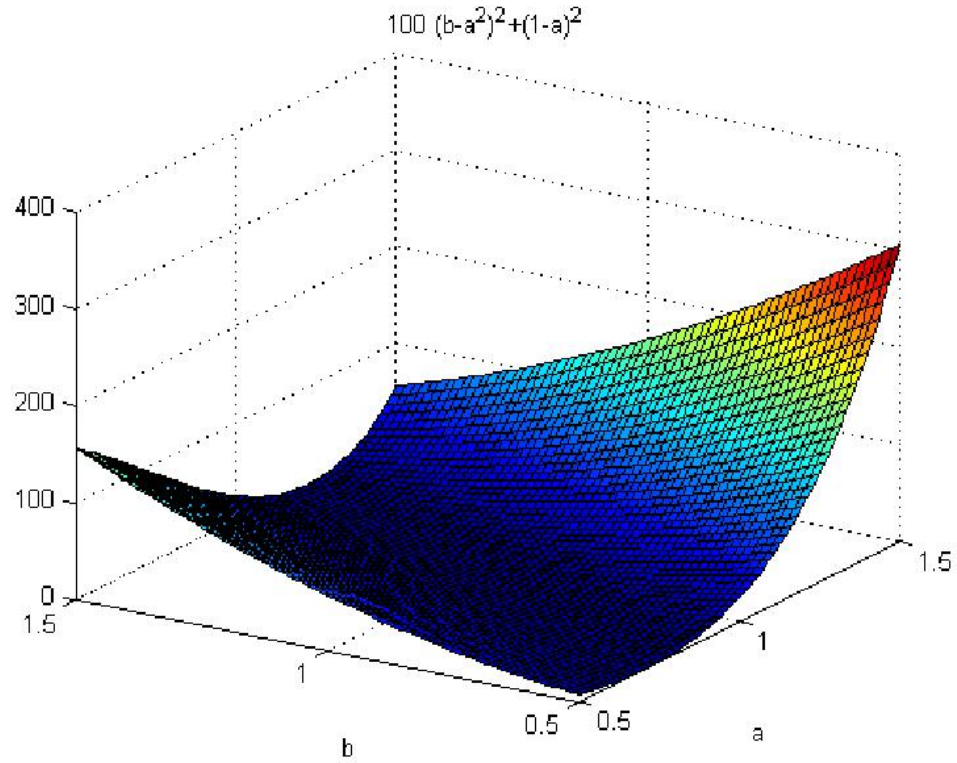


Lokales Minimum

Globales Minimum

Beispiel einer Zielfunktion

Parameter a und b sollen so gewählt werden, dass das Funktionsergebnis möglichst klein ist.



Implementierung in Matlab

Zielfunktion:

```
c=100*(b-a^2)^2+(1-a)^2
```

Schritt 1: Parameter durch Vektor ersetzen

```
x(1) = a;

x(2) = b;

c=100*(x(2)-x(1)^2)^2+(1-x(1))^2
```

Schritt 2: Funktion programmieren

```
function c=Zielfunktion(x)

c=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

Warum so umständlich?

Matlab kann nur Funktionen mit **einem Parameter** optimieren!
Der Übergabeparameter kann ein Vektor sein.
Der Rückgabeparameter muss ein skalarer Wert sein.

Also: **Alle Parameter in einen Vektor packen!**

Alternative Implementierung:

```
function c=Zielfunktion(x)

a=x(1);

b=x(2);

c=100*(b-a^2)^2+(1-a)^2;
```

function handle

Damit der Optimierer weiß welche Funktion er denn Minimieren soll, muss man den Funktionsnamen mitteilen.

Übergabeparameter mit Funktionsnamen
nennt man ***function handle***

Dem Funktionsnamen wird ein „@“ vorangestellt.

Beispiel: `@Zielfunktion`

Einfacher Optimierer

Matlab hat in der Standardversion einen einfachen Optimier namens `fminsearch`.

`fminsearch`: Find minimum of unconstrained multivariable function using derivative-free method

Finde Minimum einer Funktion mit mehreren Variablen ohne Nebenbedingungen unter Verwendung einer ableitungsfreien Methode.

fminsearch

```
x = fminsearch(fun,x0)
```

Übergabeparameter

`fun` = *function handle*, also @Funktionsname

`x0` = Vektor mit den Startwerten.

Rückgabeparameter

`x` = Vektor mit den optimalen Werten.

Beispiel (richtige Lösung ist `x=[1,1]`):

```
x=fminsearch(@Zielfunktion,[0 0])

x=[1.00000438589862,1.00001064099165]
```

fminsearch

```
[x,fval] = fminsearch(fun,x0)
```

Rückgabeparameter

`x` = Vektor mit den optimalen Werten.

`fval` = Funktionsergebnis bei optimalen Werten.

Beispiel:

```
[x,fval]=fminsearch(@Zielfunktion,[0 0])
```

```
x=[1.00000438589862,1.00001064099165]
```

```
fval=3.68617691517591e-010
```

(also Null)

Optionen für den Optimierer

Als zusätzlicher Parameter können Optionen gesetzt werden.
Diese Parameter werden mit dem Befehl `optimset` gesetzt.

Option	Beschreibung
MaxFunEvals	Maximale erlaubte Anzahl von Aufrufen der Zielfunktion
MaxIter	Maximale Anzahl der Iterationen
TolFun	Abbruch: Toleranz des Funktionswertes
TolX	Abbruch; Toleranz für x

Beispiel für Optionen

```
x=fminsearch(@Zielfunktion,…
[0 0],optimset('TolX',0.0000001))
```

Ergebnis:

```
x=[1.00000001628507,1.00000003175663]
```

```
[x,fVal]=fminsearch(@Zielfunktion,[0
0],optimset('Tolfun',1e-15))
```

Ergebnis:

```
fVal=3.31383040113604e-016
```

Fragestellung bei der Optionen:

Sollen das Funktionsergebnis oder die Parameter genauer bestimmt werden.

Beispiel: Funktion durch Messwerte

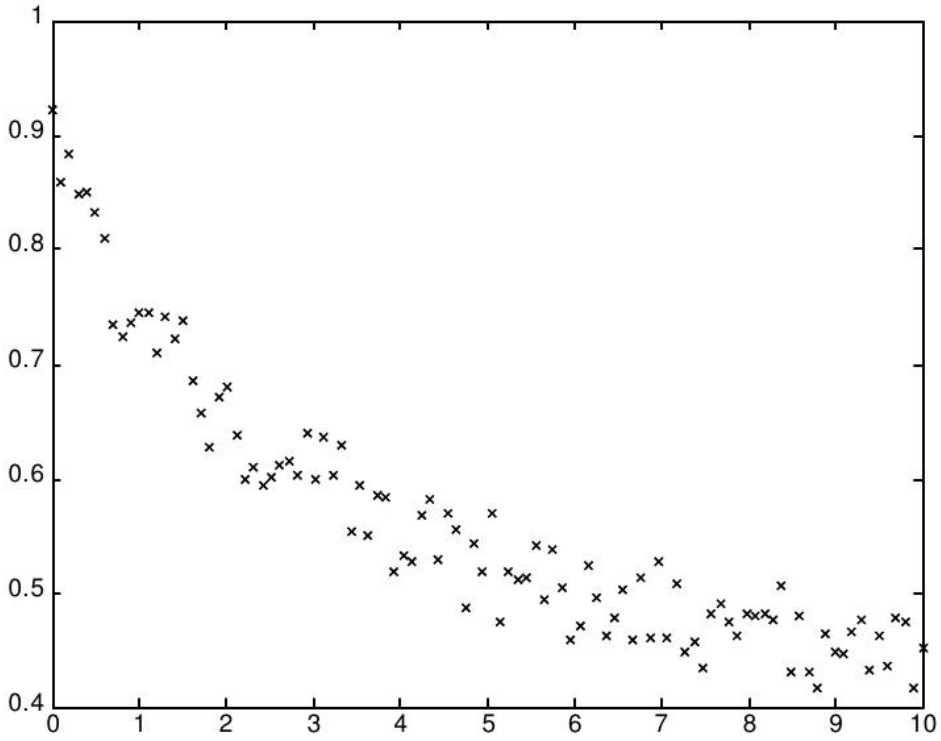
Folgende Messdaten seien gegeben:

Mögliche Funktion:

```
y=a/(x+b)^c
```

Frage:

```
a,b,c = ?
```



Vorgehensweise

Die Funktion soll „optimal“ durch die Messwerte gelegt werden.

Idee:

Funktionswerte werden an den Messpunkten (x) ausgerechnet (y).

Dann wird vergleichen wie gut die berechneten Werte mit den Messwerten übereinstimmen.

Je geringer die Abweichung ist, desto besser passt die Funktion an die Messwerte.

Bewertung

Fehlerquadratsumme (FQS)

1. F: Differenz der Werte (Ist-Soll = Fehler)
2. Q: Diesen Wert quadrieren. Damit werden große Abweichungen besonders „bestraft“
3. S: Alle Werte summieren

Die Fehlerquadratsumme ist die Rückgabevariable der Zielfunktion. Dieser Wert soll minimiert werden.

Problem

`fminsearch` kann der Zielfunktion nur einen Parameter übergeben.
Wie können der Zielfunktion die Messwerte mitgeteilt werden?

Lösung: Funktion in einer Funktion (*nested function*)

```
function y=f1(x)

        Messwert = 12;

        function y1=f2(x2);

                …Hier kann auf „Messwert“ zugegriffen
                werden

        end

end
```

Funktion für die Anpassung

```
function y=my_sample_function(x,parameters)
    % Allgemein: Die Parameter werden in einem Vektor übergeben.
    % Die einzelnen Parameter (a,b,c...) werden nun wieder aus dem Vektor
    % geholt.

    a=parameters(1);
    b=parameters(2);
    c=parameters(3);

    % Vektorisierte Beispielfunktion für die "richtigen" Werte
    % an den Stellen x ausrechnen.
    y=a./(x+b).^c;
end
```

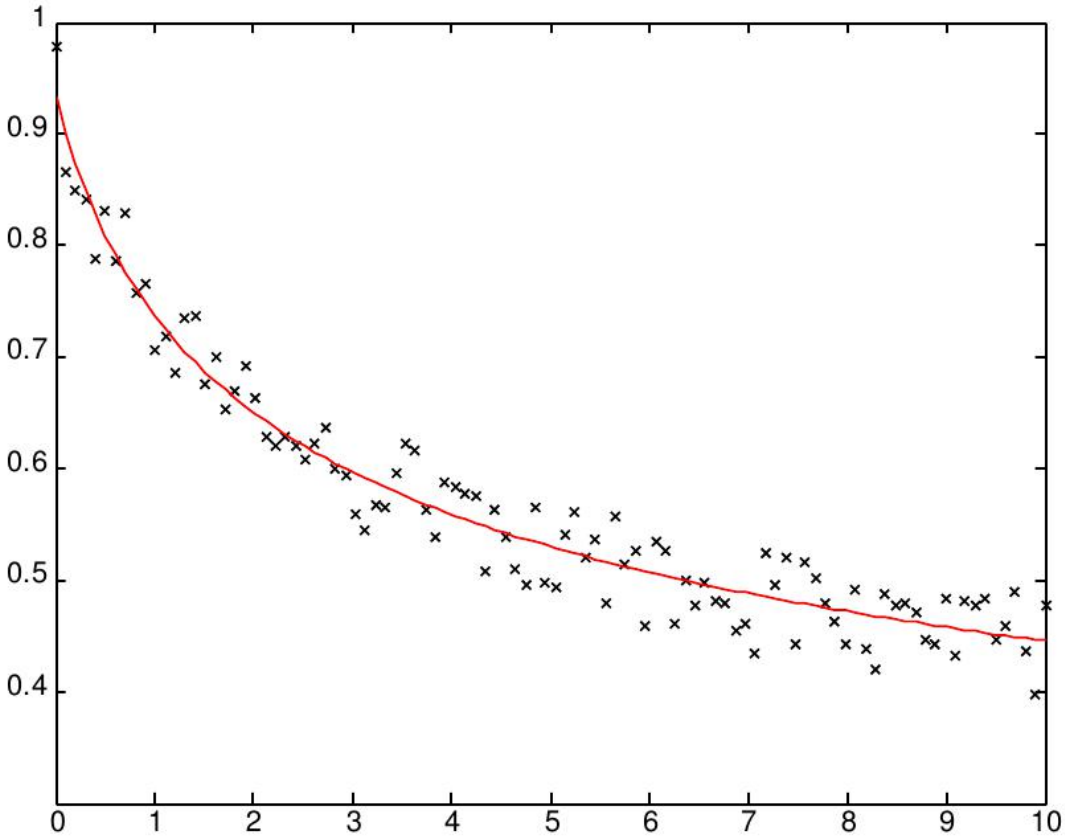
Zielfunktion

```
function min_me=my_min_function(parameters)
    % y-Werte mit den gegebenen Parametern ausrechnen

    % x ist gegeben, weil dies eine eingebette Funktion ist.

    y_now=my_sample_function(x,parameters);


    % Fehlerquadratsumme

    % y_Rauschen ist gegeben, weil dies eine eingebette Funktion ist.

    min_me = sum((y_Rauschen-y_now).^2);

end
```

Optimieren

```
function optBeispiel()
% Messwerte holen

[x,y_Rauschen]=generateValues();

% Startparameter für die Funktion
start_parameters=[1,1,1];

% Parameter optmieren
param_opt=fminsearch(@my_min_function,start_parameters)

    function min_me=my_min_function(parameters)
        … Siehe Folie vorher

    end

end
```

Ergebnis



Alternative

Matlab bietet ein Funktion, die genau auf diese Weise Funktionen an Messwerte anpasst:

```
x = lsqcurvefit(fun,x0,xdata,ydata)
```

`fun` = function handle

`X0` = Vektor mit Startwerten

`xdata` = Vektor mit den x-Koordinaten

`xdata` = Vektor mit den y-Koordinaten

Die Zielfunktion muss folgendes Aussehen haben

```
function F = myfun(x,xdata)
F = ...
```

...Berechne Funktionswerte an `xdata`

optimtool

Optimierungsverfahren

Zielfunktion

Grenzen für Parameter