Einführung in die Programmierung mit C++

Modern Family Tour: Parameterschätzer

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen

Inhalt

Inhalt

Modern Family

Einfache Funktionen und Namensbereiche


```
1  // naumann@stce.rwth-aachen.de
2
3  #include <iostream>
4  #include <cassert>
5
6  namespace models { // for models
7    float model(float p, float x) {
8      return p*x;
9    }
10 }
11
12 int main() {
13   using namespace std; // for convenience
14   float p=2, x=0;
15   cout << "x="; cin >> x;
16   assert(0<=x&&x<=10);
17   float y=models::model(p,x);
18   cout << "y=" << y << endl;
19   return 0;
20 }
```

- implizite Suche nach Symbolen (cin, cout, endl) im Namensbereich std innerhalb von main
- Definition und explizite Verwendung des Namensbereichs models
- darin nutzerdefinierte Funktionen zur Berechnung des Rückgabewerts (hier in y gespeichert) für gegebene Eingabeargumente (hier x und p)
- Verbesserte Struktur, Erweiterbarkeit, Wartbarkeit des Quellcodes

→ Live: Tour_MF003.cpp

Der erste Treffpunkt soll bei $(2, y)$ liegen.

Julia (Gretchen ist beim Joga) nimmt an, dass die Herren schon recht wackelig auf den Beinen sein werden.

Sie setzt $p = 2$ und lässt ihr Programm

$$y = p \cdot x = 2 \cdot 2 = 4$$

berechnen. Frohen Mutes begibt sie sich zum simulierten Treffpunkt $(2, 4)$.

Völlig unerwartet trifft sie Faust und Romeo jedoch schon bei $(2, 3)$ an. Die Jungs vertragen tatsächlich mehr als vermutet ...

→ Tafel: Diskrepanz im (x, y)-Koordinatensystem

Gretchen (Julia ist beim Pilates) lernt aus den am vorigen Freitag gemachten Beobachtungen.

Sie passt p so an, dass sie den Treffpunkt $(2, 3)$ der Vorwoche exakt vorhergesagt hätte.

Die Mathematik ist trivial. Die lineare Gleichung $y = p \cdot x$ muss bei gegebenen $x, y \in \boldsymbol{R}$ für $p \in \boldsymbol{R}$ gelöst werden, was direkt per Divison möglich ist, d.h.

$$p = \frac{y}{x} = \frac{3}{2} = 1.5 \quad .$$

→ Tafel: Korrektur im (x, y)-Koordinatensystem

Inhalt

Modern Family

Parameterschätzer für eine Beobachtungen


```
1  // naumann@stce.rwth-aachen.de
2
3  #include <iostream>
4  #include <cassert>
5
6  // global namespace
7  float estimate(float x, float y) {
8    assert(x!=0);
9    return y/x;
10 }
11
12 int main() {
13   float x=0,y=0;
14   std::cout << "x="; std::cin >> x;
15   assert(0<=x&&x<=10);
16   std::cout << "y="; std::cin >> y;
17   float p=estimate(x,y); // parameter estimation
18   std::cout << "p=" << p << std::endl;
19   return 0;
20 }
```

→ Live: Tour_MF004.cpp

```
:-) ./Tour_MF004.exe
x=2
y=3
p=1.5
```

Alternative: Daten in `data.in`

```
2 3
```

und

```
:-) ./Tour_MF004.exe < data.in
```

liefert (leicht derangierte) Ausgabe

```
x=y=p=1.5
```

Treffpunkt soll diesmal $(1, y)$ sein.

Die Computersimulation des den bisherigen Beobachtungen angepassten Modells schlägt wegen

$$y = 1.5 \cdot 1 = 1.5$$

$(1, 1.5)$ als Treffpunkt vor.

Faust und Romeo tauchen pünktlich um 23h am Punkt $(1, 2)$ auf - zum Glück in Rufweite...

→ Tafel: Diskrepanz im (x, y)-Koordinatensystem

Treffpunkt soll $(1.5, ?)$ sein. Julia (Gretchen ist, na klar, beim Joga) hat ein Problem!

Die beiden bisherigen Beobachtungen lieferten widersprüchliche Werte für p (1.5 bzw. 2). Das Modell scheint leider falsch zu sein :-(

Kopf hoch! Um eine grundlegende Änderung des Modells zu vermeiden beschliesst Julia (nach einem ausgiebigen Telefonat mit Gretchen), p so anzupassen, dass alle bisherigen Beobachtungen im Mittel gut vorhersagbar gewesen wären. Mit anderen Worten: p soll so gewählt werden, dass die Summe der Unterschiede zwischen den vorhergesagten und den eigentlichen Treffpunkten minimiert wird.

→ Tafel: Konflikt im (x, y)-Koordinatensystem

Die Summe der Distanzen zwischen Beobachtungen und entsprechenden Simulationen soll minimiert werden, d.h. bei jeweils zwei Werten

$$|p \cdot x_1 - y_1| + |p \cdot x_2 - y_2| \to \min \ .$$

Oft will man den Absolutbetrag $|\cdot|$ wegen des "unschönen Knicks" am Ursprung vermeiden und minimiert alternativ die "schön glatte" Summe der Fehlerquadrate

$$f(p, \mathrm{x}, \mathrm{y}) = (p \cdot x_1 - y_1)^2 + (p \cdot x_2 - y_2)^2 \quad .$$

Offensichtlich ist dieser Wert immer grösser als bzw. gleich 0 und sollte somit so nah wie möglich an 0 herangebracht werden.

AUFWACHEN! Ist $f(p, \mathrm{x}, \mathrm{y}) = 0$ möglich?

Inhalt

Julia erinnert sich, dass an einem stationären Punkt die erste Ableitung einer Funktion verschwindet. Sie leitet $f(p, x, y)$ nach p ab und erhält

$$\frac{df}{dp}(p, x, y) = 2 \cdot (x_1^2 + x_2^2) \cdot p - 2 \cdot (x_1 \cdot y_1 + x_2 \cdot y_2) = 0 \quad .$$

Jetzt p noch auf die linke Seite bringen und ...

$$p = \frac{2 \cdot (x_1 \cdot y_1 + x_2 \cdot y_2)}{2 \cdot (x_1^2 + x_2^2)} = \frac{2 \cdot 3 + 1 \cdot 2}{2^2 + 1^2} = \frac{8}{5} = 1.6 \quad .$$

An einem lokalen Minimum muss ausserdem die zweite Ableitung von f bzgl. p positiv sein.

$$\frac{d^2 f}{dp^2}(p, x, y) = 2 \cdot (x_1^2 + x_2^2) = 2 \cdot (2^2 + 1^2) = 10 > 0 \, .$$

→ Tafel: Kurvendiskussion

AUFWACHEN!

Was ergibt sich für $y = p \cdot x^2$?

Wach?

Was ergibt sich für $y = p \cdot x^2$?

$$f(p, x, y) = (p \cdot x_1^2 - y_1)^2 + (p \cdot x_2^2 - y_2)^2 \quad \Rightarrow$$

$$\frac{df}{dp}(p, x, y) = 2 \cdot \left(p \cdot (x_1^4 + x_2^4) - (x_1^2 \cdot y_1 + x_2^2 \cdot y_2)\right) = 0 \quad \Rightarrow$$

$$p = \frac{x_1^2 \cdot y_1 + x_2^2 \cdot y_2}{x_1^4 + x_2^4}$$

wobei

$$\frac{d^2 f}{dp^2}(p, x, y) = 2 \cdot (x_1^4 + x_2^4) \geq 0 \,.$$

Inhalt

Modern Family

Parameterschätzer für zwei Beobachtungen


```
1  // naumann@stce.rwth-aachen.de
2
3  #include <iostream>
4  #include <cassert>
5  #include <cmath> // mathematical functions (e.g, pow)
6
7  // parameter estimation / model calibration
8  float estimate(float x1, float y1, float x2, float y2) {
9    float d=pow(x1,2)+pow(x2,2);
10   assert(d!=0);
11   return (x1*y1+x2*y2)/d;
12 }
13
14 int main() {
15   using namespace std;
16   float x1=0,x2=0,y1=0,y2=0;
17   cout << "x1="; cin >> x1; cout << "y1="; cin >> y1;
18   cout << "x2="; cin >> x2; cout << "y2="; cin >> y2;
19   cout << "p=" << estimate(x1,y1,x2,y2) << endl;
20   return 0;
21 }
```

→ Live: Tour_MF005.cpp

```
:-) ./Tour_MF005.exe
x1=2
y1=3
x2=1
y2=2
p=1.6
```

Alternative: Daten in `data.in`

```
2 3
1 2
```

und

```
:-) ./Tour_MF005.exe < data.in
```

liefert (leicht derangierte) Ausgabe

```
x1=y1=x2=y2=p=1.6
```

1. Daten in `data.plot` (=`data.in`)

2. `gnuplot` script `plot.gp`

```
set terminal pdf # pdf output
set output "plot.pdf" # output file name
# plot data.plot and y=1.6*x; show domain [0:4]
plot [0:4] "data.plot", 1.6*x
```

3. Ausführen von `gnuplot` script

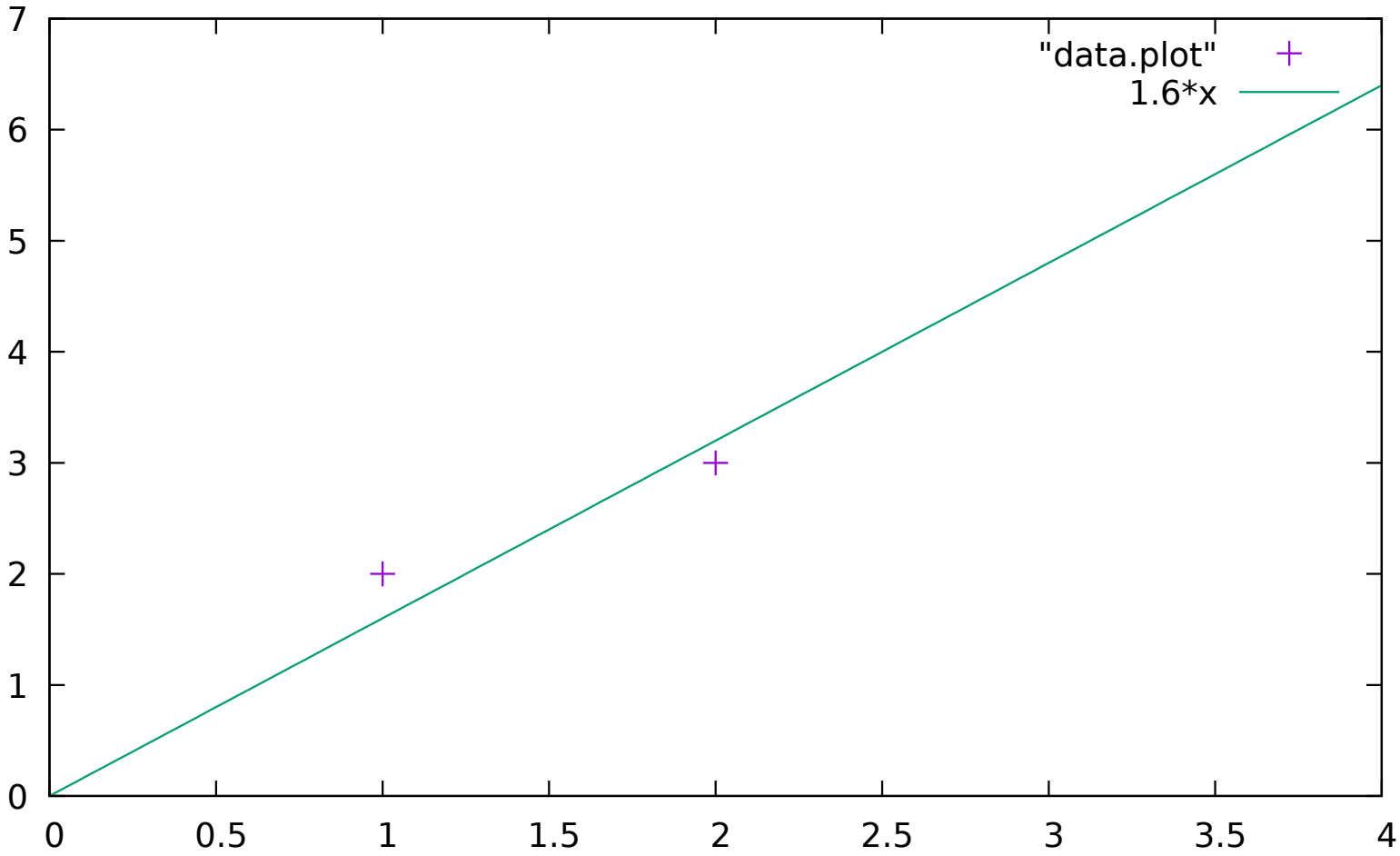
```
:-) gnuplot plot.gp
:-)
```

4. Betrachten von Datei `plot.pdf`, z.B.

```
:-) opera plot.pdf
```

Parameterschätzer für zwei Beobachtungen

Visualisierung mit `gnuplot`



Zusammenfassung

Einfache Funktionen und Namensbereiche

Parameterschätzer für eine Beobachtung

Kurvendiskussion

Parameterschätzer für zwei Beobachtungen