

# Übung 2

## Computational Physics III

Matthias Plock (552335)

Paul Ledwon (561764)

10. Mai 2018

### Inhaltsverzeichnis

|  |          |
|--|----------|
| <b>1 Aufgabe 1</b>   | <b>1</b> |
| 1.1 Einleitung . . . . .   | 1        |
| 1.2 Implementierung des Matrix-Vektor-Produktes für den Laplace Operator . . . . . | 1        |
| 1.3 Implementierung von Hilfsfunktionen . . . . .                                  | 2        |
| 1.4 Implementierung der Methode der Konjugierten Gradienten . . . . .              | 2        |
| <b>2 Aufgabe 2</b>   | <b>3</b> |

## 1 Aufgabe 1

### 1.1 Einleitung

Implementiert wird die Methode der Konjugierten Gradienten aus Übung 1 in der Programmiersprache C.

Im Folgenden wird auf die drei Teilaufgaben eingegangen.

### 1.2 Implementierung des Matrix-Vektor-Produktes für den Laplace Operator

Zu implementieren war das Matrix-Vektor-Produkt

$$w = Av,$$

wobei  $A$  der Laplace-Operator ist und  $v$  ein Vektor mit Unbekannten. Bei einer Problemgröße von  $N \times N$  hat  $A$  die Dimension  $(2N + 2) \times (2N + 2)$ ,  $v$  hat die Dimension  $2(2N + 2)$ . Die extra Dimensionen tauchen auf, da die Randbedingungen mit einbezogen werden.

Die Implementierung ist in der Funktion `void laplace_2d(double *w, double *v)` zu finden.

Die Funktion wird einen Unbekannten-Vektor  $v$  (Abb. 1) aufgerufen, das Resultat in die Shell ausgegeben (Abb. 2).

```
v =
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 10.30 19.80 10.50 11.50 8.10 25.50 7.40 23.60 0.00
0.00 4.10 20.50 18.60 17.10 24.20 25.10 22.70 7.00 0.00
0.00 12.40 19.40 8.40 24.80 2.70 23.20 23.10 14.10 0.00
0.00 11.80 9.00 4.60 9.90 5.10 15.90 20.10 15.40 0.00
0.00 10.20 5.00 1.30 18.30 4.90 8.80 16.30 9.00 0.00
0.00 3.70 9.30 0.50 2.30 8.80 23.30 9.40 21.20 0.00
0.00 17.10 17.80 20.50 19.80 15.50 18.00 8.40 1.70 0.00
0.00 1.40 13.00 11.60 6.50 3.30 6.10 22.00 13.50 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
||v|| = 118.21852647
```

Abbildung 1: Der Unbekannten-Vektor  $v$ .

```

w =
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 -17.30 -37.90 7.90 -10.30 28.80 -61.40 42.20 -80.00 0.00
0.00 26.80 -20.10 -17.90 10.70 -43.80 -4.80 -28.20 32.40 0.00
0.00 -14.30 -27.30 33.80 -61.10 66.50 -26.00 -12.30 -10.90 0.00
0.00 -15.60 4.80 10.20 13.20 13.00 -6.40 -9.70 -18.40 0.00
0.00 -20.30 9.80 23.20 -54.80 21.40 25.20 -17.90 16.90 0.00
0.00 21.80 -10.20 31.40 38.20 10.80 -48.20 31.60 -64.70 0.00
0.00 -45.50 -11.30 -32.30 -34.40 -12.10 -18.70 17.50 36.30 0.00
0.00 24.50 -21.20 -6.40 8.70 14.90 18.90 -60.00 -30.30 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
||w|| = 248.22340744

```

Abbildung 2: Resultat des Funktionsaufrufs `void laplace_2d(double *w, double *v)` mit dem Unbekannten-Vektor  $v$ .

Um das Resultat zu verifizieren wenden wir den Laplace-Operator auf den Unbekannten-Vektor  $v$  (ohne die Randbedingungen) an. Das Resultat ist in Abb. 3 zu finden.

```

[ [-17.3 -37.9 7.9 -10.3 28.8 -61.4 42.2 -80. ]
[ 26.8 -20.1 -17.9 10.7 -43.8 -4.8 -28.2 32.4]
[ -14.3 -27.3 33.8 -61.1 66.5 -26. -12.3 -10.9]
[ -15.6 4.8 10.2 13.2 13. -6.4 -9.7 -18.4]
[ -20.3 9.8 23.2 -54.8 21.4 25.2 -17.9 16.9]
[ 21.8 -10.2 31.4 38.2 10.8 -48.2 31.6 -64.7]
[ -45.5 -11.3 -32.3 -34.4 -12.1 -18.7 17.5 36.3]
[ 24.5 -21.2 -6.4 8.7 14.9 18.9 -60. -30.3]]

```

Abbildung 3: Anwendung des Laplace-Operators aus dem letzten Übungsblatts auf den Unbekannten-Vektor  $v$  (ohne Randbedingungen).

Vergleich der Abbildungen 2 und 3 zeigt, dass die implementierte Methode das selbe Resultat liefert, wie die Methode der letzten Übung.

### 1.3 Implementierung von Hilfsfunktionen

Es wurden Hilfsfunktionen implementiert um ein Skalarprodukt zu berechnen (`double scalar_product(double *u, double *v)`), um zwei Vektoren zu addieren (`void vector_addition(double *u, double *v, double *w)`) sowie um einen Vektor zu skalieren (`void scale_vector(double prefactor, double *v, double *w)`).

### 1.4 Implementierung der Methode der Konjugierten Gradienten

In `void conjugate_gradient_laplace(double *x, double *b)` ist die Methode der Konjugierten Gradienten implementiert. Mit ihr wird das  $x$  zum Unbekannten-Vektor  $v$  aus Abb. 1 bestimmt. Das Resultat konvergiert nach 27 Schritten (Konvergenz wird angenommen, wenn das Residuum  $\|r\| < 1 \times 10^{-15}$  ist) und ist in Abb. 4 zu finden.

```

tol (1.741376e-16 < 1.000000e-15) reached after 27 steps
x =
 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
 0.00 -15.24 -27.67 -31.78 -34.98 -36.26 -38.40 -29.25 -20.28 0.00
 0.00 -22.97 -43.87 -53.98 -60.39 -63.53 -62.61 -50.92 -28.28 0.00
 0.00 -28.66 -50.38 -61.26 -71.96 -70.69 -72.48 -60.84 -34.93 0.00
 0.00 -28.91 -48.31 -60.34 -70.69 -72.07 -72.60 -61.91 -36.50 0.00
 0.00 -26.86 -44.60 -56.51 -68.50 -69.20 -68.02 -57.63 -33.75 0.00
 0.00 -23.72 -41.72 -51.31 -59.28 -63.32 -63.84 -50.52 -31.89 0.00
 0.00 -22.60 -37.96 -47.23 -51.71 -52.15 -50.22 -39.32 -22.08 0.00
 0.00 -11.63 -22.50 -27.43 -28.37 -27.84 -27.56 -26.07 -15.41 0.00
 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
||x|| = 381.97886569

```

Abbildung 4:  $x$  zum Unbekannten-Vektor  $v$  aus Abb. 1, berechnet mit der Methode der Konjugierten Gradienten.

Das Resultat wird mit der Python-Implementierung aus Übung 1 verifiziert (Abb. 5).

```

[[-15.24 -27.67 -31.78 -34.98 -36.26 -38.4 -29.25 -20.28]
 [-22.97 -43.87 -53.98 -60.39 -63.53 -62.61 -50.92 -28.28]
 [-28.66 -50.38 -61.26 -71.96 -70.69 -72.48 -60.84 -34.93]
 [-28.91 -48.31 -60.34 -70.69 -72.07 -72.6 -61.91 -36.5 ]
 [-26.86 -44.6 -56.51 -68.5 -69.2 -68.02 -57.63 -33.75]
 [-23.72 -41.72 -51.31 -59.28 -63.32 -63.84 -50.52 -31.89]
 [-22.6 -37.96 -47.23 -51.71 -52.15 -50.22 -39.32 -22.08]
 [-11.63 -22.5 -27.43 -28.37 -27.84 -27.56 -26.07 -15.41]]

```

Abbildung 5:  $x$  berechnet mit der Python-Implementierung.

Vergleich der Abbildungen 4 und 5 zeigt, dass die implementierte Methode des selbe Ergebnis liefert, wie die Python Implementierung des letzten Übungsblattes.

## 2 Aufgabe 2