

## Vorgeschlagene Struktur für die Programmieraufgabe des 9. Übungsblatts für Einführung in Numerische Mathematik Sommersemester 2012

Schreib die Funktion neben Ihrem Namen. Alle Funktionen sollen die (folgende) Funktion `globals` (Keeling) aufrufen, die die globalen Variablen für alle definiert:

```
function globals
global N      ... % Anzahl der Zellen
x            ... % Gitterpunkte
h            ... % Zellenbreite
ep           ... % Abrundung fuer |x| ~= sqrt(x^2 + ep)
mue          ... % Regularisierungsparameter
alf          ... % Schrittbreite
tol          ... % relativer Abbruchstoleranz
Methode      ... % Optimierungsmethode
v            ... % Vektor der verrauschten Werte der Eigabe
u            ... % Vektor der entrauschten Werte der Ausgabe
du           ... % aktuellste Aenderungsrichtung fuer u
Jh           ... % aktuellster Wert der Zielfuntion
F            ... % Gradient, d.h. F = DJh/h
DF           ... % Jacobi-Matrix von F
D            ... % F = DJh/h = u - v + (mu/h^2)D*u
```

Der gesamte Code hat eine pyramidenförmige Struktur. Die Treiber-Funktion (ganz oben in der Pyramide) heisst Entrauschen (Keeling).

Entrauschen ruft auf:

- Start (Lepuschitz)
- Loese (Pieber)
- Ergebnis (Adanc)

Start initialisiert den Code und definiert:

```
N      % Anzahl der Zellen
x      % Gitterpunkte
h      % Zellenbreite
ep     % Abrundung fuer |x| ~= sqrt(x^2 + ep)
mue    % Regularisierungsparameter
tol    % relativer Abbruchstoleranz
v      % das verrauschte Signal, das entrauscht werden soll
Methode % 1 => Abstieb, 2 => Implizit, 3 => Newton
```

Ergebnis stellt die verrauschte Kurve und die entrauschte Kurve zusammen grafisch dar. Weitere Information kann berichtet werden, z.B. über die verwendeten Parameter.

Loese ruft die folgenden auf,

- KriegD (Wegger)
- KriegF (Hrassnigg)
- KriegDF (Knittelfelder)
- Liniensuche (Holler)

und verwendet eine gewisse *Methode*, um  $Jh$  zu minimieren. Loese sieht ungefähr so aus:

$u = v$

Methode = 1:

```
KriegF aufrufen, um F zu holen
du = -F
Liniensuche aufrufen, um eine gute Schrittweite alf zu finden
u = u + alpha*du
wiederhole bis Abbruchstoleranz erreicht
```

Methode = 2:

```
KriegD aufrufen, um D zu holen
du = (I + mue*D/h^2) \ v
Liniensuche aufrufen, um eine gute Schrittweite alf zu finden
u = u + alpha*du
wiederhole bis Abbruchstoleranz erreicht
```

Methode = 3:

```
KriegF aufrufen, um F zu holen
KriegDF aufrufen, um DF zu holen
du = -DF \ F
Liniensuche aufrufen, um eine gute Schrittweite alf zu finden
u = u + alf*du
wiederhole bis Abbruchstoleranz erreicht
```

KriegD berechnet  $D$  in Abhängigkeit von  $u$ , wobei  $F = DJh/h = u - v + (\mu/h^2)D*u$ .

KriegF berechnet  $F$  in Abhängigkeit von  $u$ , wobei  $F = DJh/h = u - v + (\mu/h^2)D*u$ .

KriegDF berechnet die Jakobi-Matrix  $DF$  in Abhängigkeit von  $u$ , wobei  $F = DJh/h = u - v + (\mu/h^2)D*u$ .

Liniensuche nimmt  $u$  und  $du$  her und ruft auf:

- KriegJh (Glowatschnig)

Ein Wert  $alf \in (0, 2)$  wird zurückgegeben, wobei  $Jh(u + alf*du)$  ungefähr am kleinsten ist. Die einfachste Liniensuche erfolgt über eine Abtastung des Intervalls  $(0, 2)$ .

KriegJh berechnet  $Jh$  in Abhängigkeit von  $u$ .