Goethe-Center for Scientific Computing (G-CSC)
Goethe-Universität Frankfurt am Main

# Modelling and Simulation I

(Practical SIM1, WS 2016/17)

M. Hoffer, Dr. S. Reiter, Dr. A. Vogel

**Aufgabenblatt 7 (Abgabe: Mo., 9.1.2016, 10h)**

Bitte beachten Sie die Hinweise zur Installation von **ug4** und **ConnectionViewer** auf der Vorlesungshomepage. Stellen Sie sicher, dass beide Programme auf Ihrem Rechner installiert und ausführbar sind.
Für Hinweise im Umgang mit der Skriptsprache *Lua* beachten Sie bitte das nach der Installation von 'Examples' auf Ihrem Rechner verfügbare Skript `ug4/apps/Examples/lua-programming.lua`.

**Hinweis:** Um Konflikte mit der Versionskontrolle zu vermeiden sollten Sie heruntergeladene Skripte zunächst kopieren und dann die kopierte Version editieren. Für jede Aufgabe sollten Sie dabei ein separates Skript erstellen. Sollte es bei Aktualisierungen zu Problemen kommen, können Sie die problematischen Skripte löschen und nochmals aktualisieren. Stellen Sie dann sicher, zuvor eine Sicherheitskopie des betroffenen Skripts erstellt zu haben.

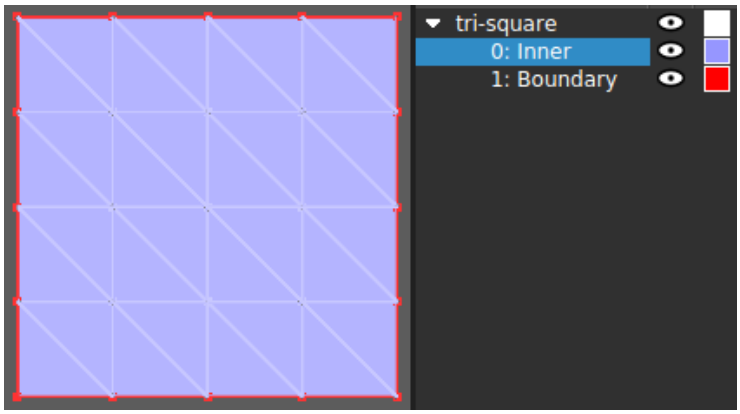
Bitte aktualisieren Sie Ihre ug4 Version vor dem Bearbeiten der Aufgaben, indem Sie in Ihrem ug4 Ordner oder einem Unterordner folgendes ausführen:

```
ughub gitpull
```

Bitte schicken Sie erstellte Skripte und Ausgaben an

`practical.sim1@gcsc.uni-frankfurt.de`

**Aufgabe 1** (Geometrie, 2P)
Erstellen Sie das dargestellte regelmäßige Dreiecksgitter mit $5 \times 5$ Knoten (Breite: 1, Höhe: 1, Zentrum: $(0.5, 0.5)$).
Nutzen Sie dazu das Tool `Remeshing-Triangulation-ConvertToTriangles` direkt nach dem Erstellen des Quadrats.
Achten Sie auf die korrekte Zuweisung der Subsets und darauf dass jeder Knoten, jede Kante und jedes Flächenelement einem Subset zugewiesen sind.
Speichern Sie das erstellte Gitter als 'tri-square.ugx'.




**Aufgabe 2** (6P)
Erstellen Sie zunächst eine Kopie des Skripts "ug4/apps/Examples/laplace.lua" und passen Sie die rechte Seite des Modells an, indem Sie in Zeile 101 die Anweisung

```
elemDisc:set_source(''Source''..dim..''d'')
```

durch

```
elemDisc:set_source(1)
```

ersetzen (statt $-\Delta u = f$ betrachten wir also $-\Delta u = 1$).
Führen Sie das modifizierte Skript mit dem von Ihnen in A1 erstellten Gitter zunächst ohne Verfeinerungen aus:

```
ugshell -ex laplace-copy.lua -grid tri-square.ugx -numRefs 0
```

Betrachten Sie die resultierende Matrix "A2d.mat", sowie die rechte Seite "rhs_laplace_2d.vec" mit dem ConnectionViewer. Führen Sie dazu z.B.

```
java -jar SOMEPATH/ConnectionViewer.jar A2d.mat
```

aus. Vergleichen Sie die Matrix und rechte Seite qualitativ mit der in der Vorlesung besprochenen Finite Differenzen Methode. Wo sehen Sie Parallelen? Wie verhalten sich Matrix und rechte Seite in Abhängigkeit von der Gitterweite $h$, wenn Sie das Problem mit höherer Feinheit lösen (-numRefs 1, bzw. -numRefs 2)?

Mit welchem Faktor müssten Sie $A$ multiplizieren, damit die Matrix mit der der aus der Vorlesung bekannten zugehörigen Finite Differenzen Methode übereinstimmen würde? Geben Sie diesen Faktor in Abhängigkeit von `numRefs` für das Gitter aus Aufgabe 1 an.

**Aufgabe 3** (Eigenwerte, 8P)
Gehen Sie nun wieder vom ursprünglichen 'laplace.lua' Skript aus und nutzen Sie für alle Berechnungen das in Aufgabe 1 erstellte Gitter.

Implementieren Sie eine Funktion `PoissonEigenvektor(x, y)`, die abhängig von einer Koordinate $(x, y)$ den zugehörigen Wert des aus der Vorlesung (Satz 12) bekannten Eigenvektors zum größten Eigenwert der aus dem Fünfpunktstern zum Poisson-Problem resultierenden Matrix $K_h$ liefert.

Nutzen Sie die Funktion `Interpolate(''PoissonEigenvektor'', u, ''c'')`, um die `GridFunction` $u$ mit den Werten des Eigenvektors zu füllen. Speichern Sie $u$ anschließend als 'u.vec'.

Skalieren Sie die Matrix $A$ über die Funktion `MatScale(A, s)`, wobei statt 's' der in Aufgabe 2 bestimmte Skalierungsfaktor eingesetzt werden soll, so dass $A$ anschließend mit der Matrix $K_h$ übereinstimmt.

Mittels `A:apply(b, u)` lässt sich $b := Au$ berechnen. Speichern Sie $b$ als 'b.vec' und vergleichen Sie die Koeffizienten von 'u.vec' und 'b.vec' im ConnectionViewer. Stellen Sie eine Verbindung zum größten Eigenwert der Matrix $K_h$ aus der Vorlesung her:

$$\lambda_{\nu,\mu} = \frac{4}{h^2}\left(sin^2\left(\frac{\nu\pi h}{2}\right) + sin^2\left(\frac{\mu\pi h}{2}\right)\right)$$

**Aufgabe 4** (Konvektion, 4P)
Gehen Sie wieder vom ursprünglichen 'laplace.lua' Skript aus und aktivieren Sie direkt nach der Erstellung die Konvektionskomponente des Konvektions-Diffusions-Diskretisierungsobjekts mittels

```
elemDisc:set_velocity(''CustomVelocity'')
elemDisc:set_upwind(FullUpwind())
```

Erstellen Sie die zugehörige Funktion `CustomVelocity(x, y)`, die abhängig von der Koordinate eines Knotens zwei Rückgabewerte `u,v` zurückgibt (`return u,v`).
Lassen Sie Ihr angepasstes Skript auf der Geometrie aus Aufgabe 1 mit 5 Verfeinerungen (-numRefs 5) für folgende Ausdrücke für $u$ und $v$ laufen und visualisieren Sie die Ergebnisse mit ConnectionViewer:

$$u := s * y$$
$$v := -s * x$$

mit $s = -1000, -10, 1, 100$.