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

Modeling and Simulation I

(Practical SIM1, WS 2018/19)

M. Huymayer, J. Wang, Dr. A. Nägel, Dr. M. Hoffer

Aufgabenblatt 7 Abgabe Montag, 28.1.2019, 16h

Im letzten Blatt haben Sie sich mit der Erstellung von räumlichen Gittern mit ProMesh beschäftigt. In diesem Blatt sollen Sie auf mit ProMesh erzeugten Gittern rechnen und erste Erfahrungen mit der Simulationsumgebung UG4 sammeln. Um Ihnen die Handhabung mit UG4 zu erleichtern, stellen wir Ihnen die Software mittels VRL-Plugin zur Verfügung. Laden Sie das VRL Projekt `ug4luashell-exported.vrlp` von der Homepage herunter und öffnen es mit VRL Studio. Die Plugins werden automatisch installiert. Das Projekt können Sie nach Neustart von VRL-Studio erneut öffnen. Laden Sie auch `ugroot.zip` herunter und entpacken es. In dem VRL-Projekt `ug4luashell-exported.vrlp` müssen Sie in der Komponente `UGConfigurator` die Pfade so anpassen, dass sie auf ihr ugroot Ordner ausgewählt ist und danach noch invoke drücken. Passen Sie auch alle weiteren Pfade an.

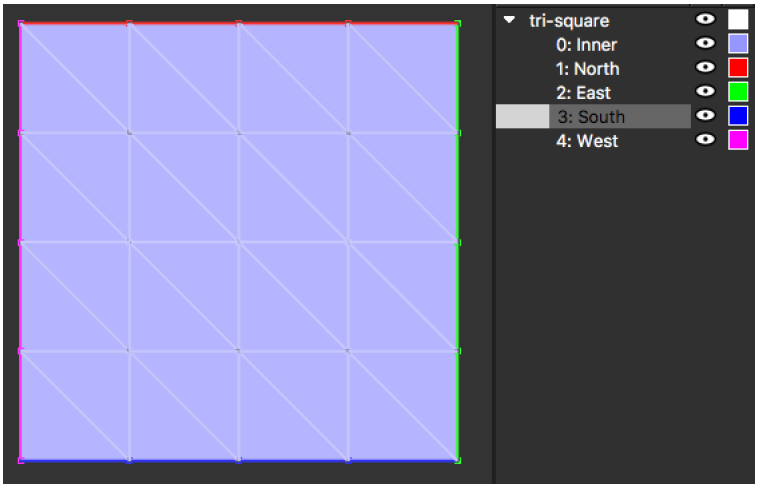
Aufgabe 1 (2+6+1+7 P)
Laden Sie das Lua-Skript laplace.lua von der Vorlesungshomepage herunter und bearbeiten Sie es in dem Lua Editor von VRL-Studio.

a) Erstellen Sie das dargestellte regelmäßige Dreiecksgitter mit 5×5 Knoten (Breite: 1, Höhe: 1, Zentrum: $(0.5, 0.5)$).

   Nutzen Sie dazu das Tool `Remeshing-Triangulation-ConvertToTriangles` direkt nach dem Erstellen des Quadrats. Um die Richtung der Kanten wie auf der Abbildung zu erreichen, nutzen Sie: `Selection-Edges-Selection Edges by Direction` und `Remeshing-Edges-Swap Edges`.

   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'.



b) Passen Sie im Lua Skript die rechte Seite des Modells an, indem Sie in Zeile 77 die Anweisung

```
elemDisc:set_source(0)
```

durch

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

ersetzen (statt $-\Delta u = 0$ betrachten wir also $-\Delta u = 1$). Für die östlichen Randwerte und westlichen Randbedingungen sollen folgende Dirichlet Randbedingungen gesetzt werden:

$$\begin{aligned} \text{West:} \quad & \varphi(x = 0, y) = 0 \\ \text{East:} \quad & \varphi(x = 1, y) = 1 \end{aligned}$$

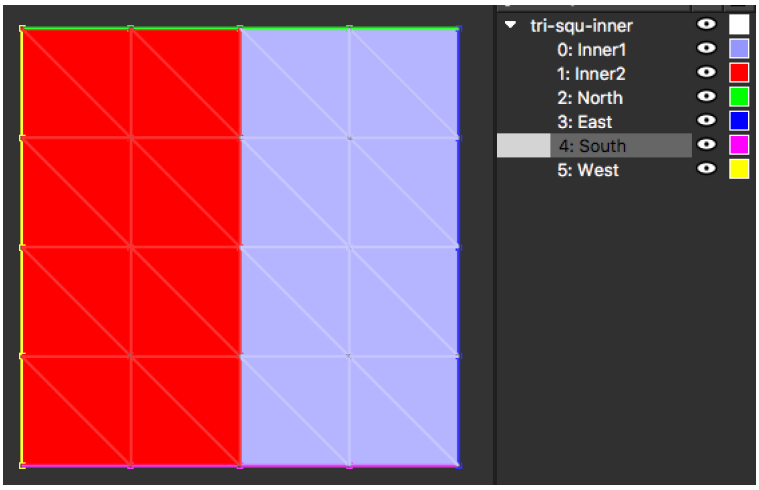
Passen Sie die Dirichlet Randbedingungen entsprechend im Skript an. Führen Sie das modifizierte Skript mit dem von Ihnen in Aufgabe 1 a erstellten Gitter zunächst ohne Verfeinerungen aus: In der Komponente `UGLuaShell` müssen Sie einen neuen Parameter setzen: wählen Sie die Funktion `set(String, double)` aus und setzen den Namen auf `numRefs` und den Wert zunächst auf 0.

Betrachten Sie die resultierende Matrix "Aufgabe1b_A2d.mat", sowie die rechte Seite "Aufgabe1b_rhs_laplace_2d.vec" mit dem ConnectionViewer. Nutzen Sie dazu entweder die VRL Komponente `ConnectionViewer`

`Component` oder starten Sie den ConnectionViewer unter VRL Tools. 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 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.

c) Erstellen Sie eine Kopie der Geometrie tri-square.ugx und modifizieren Sie die Geometrie, indem Sie dem Innenraum 2 Subsets zuweisen. Benennen Sie die Subsets entsprechend der folgenden Abbildung:



Speichern Sie die Geometrie unter tri-squ-inner.ugx ab.

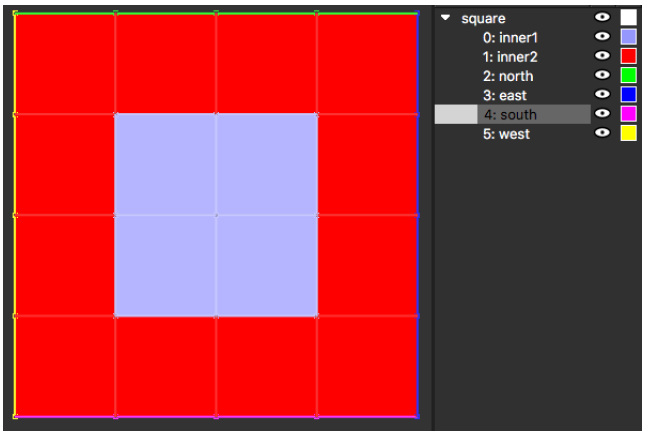
d) Für die beiden Teilgebiete `Inner1` und `Inner2` soll sich der Diffusionstensor der Poisson-Gleichung unterscheiden. Legen Sie dazu sowohl für das Gebiet `Inner1` als auch `Inner2` ein Object `ConvectionDiffusion` an und setzen Sie den Diffusionstensor mit: `elemDisc:set_diffusion(double diffusion)`. Die Variable `diffusion` soll für Inner1 auf 1 gesetzt werden und für Inner2 variiert werden mit 0.01, 1 und 100. Erstellen Sie entsprechend eine Kopie des lua-Skripts laplace.lua und machen Sie entsprechende Anpassungen. Betrachten Sie die resultierende Matrix "Aufgabe1d_A2d.mat", sowie die rechte Seite "Aufgabe1d_rhs_laplace_2d.vec"

mit dem ConnectionViewer. Wie sieht der Stern im Zentrum (0.5, 0.5) aus. Wie in Punkt (0.25, 0.5) und Punkt (0.75, 0.5)?

Wiederholen Sie den Test mit Viereckselementen statt Dreieckselementen. Beschreiben Sie den Stern in denselben Punkten. Geben Sie die resultierenden ConnectionViewer Dateien als "Aufgabe1d_quadr_A2d.mat" und "Aufgabe1d_quadr_rhs_laplace_2d.vec" ab.

Aufgabe 2 (Randbedingungen 1 + 4 P)
In dieser Aufgabe sollen Sie sich den Einfluss der Randbedingungen untersuchen.

a) Für die Diskretisierung erstellen Sie bitte für Ω folgendes Viereckgitter. Weisen Sie separate Subsets für den nördlichen, den östlichen, den südlichen und den westlichen Rand zu. Desweiteren sollte es zwei innere Subsets geben denen alle inneren Elemente, Kanten und Knoten zugewiesen werden wie in der nachfolgenden Abbildung.



b) Bitte passen Sie im Simulationsskript die `requiredSubsets` entsprechend den von Ihnen gewählten Subsetnamen an (ca Zeile 53). Für die östlichen und westlichen Randbedingungen sollen die gleichen Dirichlet Randbedingungen gesetzt werden, wie in Aufgabe 1. Passen Sie die Dirichlet Randbedingungen entsprechend im Skript an. Für die nördlichen und südlichen Ränder sollen Neumann-Null-Randbedingungen gelten. Setzen Sie auch hier den Diffusionskoeffizienten für "inner2" auf 1. Der Diffusionskoeffizient für "inner1" soll in mehreren Simulationen variiert werden. $D_{inner1} = 0.01$, $D_{inner1} = 1$ und $D_{inner1} = 100$. Untersuchen Sie die Lösung in ParaView (nutzen Sie z.B. das 'Warp by Scalar' Tool in ParaView) und im ConnectionViewer. Welche Beobachtung machen Sie?

Bonus: +2P
Setzen Sie folgende Neumann Randbedingung:

$$\begin{aligned} \text{South:} \quad & \varphi(x, y) = 1 \\ \text{North:} \quad & \varphi(x, y) = 1 \end{aligned}$$

Eine Neumann Randbedingung können Sie dabei wie folgt setzen:

```
neumannBnd = NeumannBoundary(''c'')
neumannBnd:add(''SomeCallback'', boundarySubset, innerSubset)
domainDisc:add(neumannBnd)
```

wobei `SomeCallback` eine lua-Funktion sei, die zu x und y Werten die Neumann-Randbedingung berechnet:

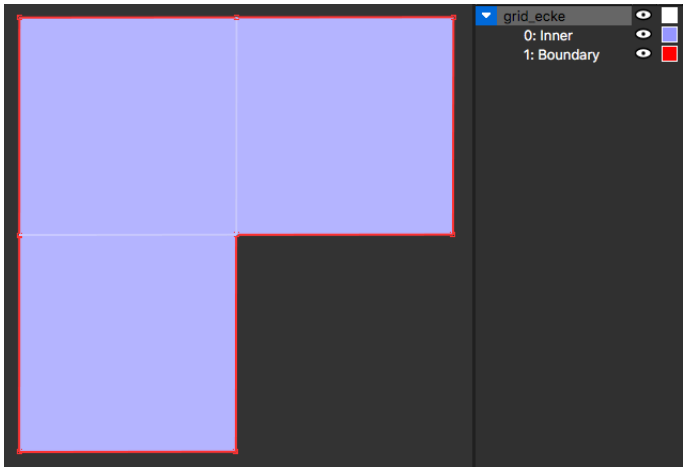
```
function SomeCallback (x, y, t)
    local value = ...
    return true, value
end
```

`boundarySubset` gebe den Namen des Rand-Subsets an, für das die Randbedingung definiert wird und `innerSubset` den Namen des inneren Subsets, auf dem die Gleichung definiert wurde (z.B. "inner2"). Dem Objekt `neumannBnd` können dabei über `add` beliebig viele Randbedingungen hinzugefügt werden.

Aufgabe 3 (2 + 6 P)
In dieser Aufgabe sollen Sie die Auswirkung der Geometrie auf Ihren Fehler untersuchen.

a) Erstellen Sie zunächst mit ProMesh eine L-förmige Geometrie (Zentrum $(0, 0)$, Höhe 2, Breite 2) mit einspringender Ecke. Im Falle einer solchen einspringenden Ecke besitzt die Ableitung der Lösung an der einspringenden Ecke eine Singularität. Weisen Sie *Subsets* gemäß der folgenden Darstellung zu.

Achtung: Achten Sie bitte auf die Ausrichtung der Geometrie!



b) Legen Sie die Poissongleichung

$$\begin{aligned} -\Delta u &= 0 \qquad && \text{in } \Omega \\ u &= g \qquad && \text{on } \Gamma \end{aligned}$$

in dem L-förmigen Gebiet zugrunde. Die Randbedingungen werden so gewählt, dass die exakte Lösung in Polarkoordinaten:

$$u(r,\varphi) = r^{\frac{2}{3}} \sin\left(\frac{2}{3}\varphi\right) \qquad \varphi \in [0, \frac{3}{2}\pi],\ r^2 = x^2 + y^2$$

entspricht. Implementieren Sie dazu eine lua Funktion `ExactSolution(x, y)`, die eine solchen Wert für eine Dirichlet Randbedingung zurückgibt.

Untersuchen Sie den L2-Fehler für Verfeinerungsstufen 4, 5, 6, 7, 8 und bilden Sie den absoluten als auch den relativen Fehler gegen die Verfeinerungsstufen ab. Nutzen Sie eine logarithmische Skala für absoluten und relativen Fehler. Den L2 Fehler erhalten Sie folgendermaßen:

```
l2error = L2Error("ExactSolution", u, "c", 0.0, 4)
```

Die L2 Norm können Sie wie folgt bestimmen:

```
l2norm = L2Norm( u, "c", 4)
```

Den relativen Fehler ermitteln Sie folgendermaßen: $l2error/l2norm$.

Tipps: Um die Randbedingung zu ermitteln beachten Sie: zur Umkehrung von Sinus und Kosinus wird deren Definitionsbereich auf das Intervall $[-\frac{\pi}{2}, \frac{\pi}{2}]$ für Sinus und auf $[0, \pi]$ für Kosinus eingeschränkt. Wenn dieser Bereich verlassen wird, ist eine Umformulierung nötig.

Anmerkung: Senden Sie den Quelltext als VRL-Studio Projekt (.vrlp Datei) und als Lua Skripte (.lua Datei), die erstellten Geometrien (.ugx bzw. .stl), die ConnectionViewer Dateien (.mat und .vec) und die Antworten zu den Fragen als E-Mail. Plots senden Sie bitte als pdf Dateien und Daten Ausgaben als Text Dateien.
Senden Sie Ihre Lösungen an `practical.sim1@gcsc.uni-frankfurt.de`.
Abgabe bis spätestens Montag, 28.1.2019, 16h.