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 9 (Abgabe: Mo., 23.1.2016, 10h)

Bitte beachten Sie die Hinweise zur Installation von **ug4** auf der Vorlesungshomepage.
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, Bilder und Ausgaben an

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

Sowohl Ausgaben aus Programmläufen, modifizierte Skripte, Visualisierungen sowie geschriebener C++ Quellcode sollen Teil Ihrer Abgabe sein.

Aufgabe 1 (Vektor- und Matrixnormen, 8P)
In einem ug4-Plugin sollen Vektor- und Matrix-Normen implementiert werden. Installieren Sie sich zunächst über 'ughub' das 'DemoPlugin'. Führen Sie dazu in Ihrem ug4-Verzeichnis folgende Kommandos aus:

```
ughub updatesources
ughub install DemoPlugin
```

Aktivieren Sie das Plugin in Ihrem build-System. In Ihrem ug4-build Ordner geben Sie dazu ein:

```
cmake -DDemoPlugin=ON .
make
```

Unter Windows 10 geben Sie 'nmake' statt 'make' ein. Sie sollten sowohl im cmake-Output als auch im Output von 'make' erkennen, dass das Plugin nun gebaut wird (siehe z.B. 'Info: Enabled Plugins' im cmake Output).

Erweitern Sie das 'Demo-Plugin' um folgende Funktionen, implementieren Sie diese und registrieren Sie sie in der Funktion 'InitUGPlugin_DemoPlugin':

```
number VecNormSup (vector_type& v);
number VecNorm2 (vector_type& v);
number MatNormRowSum (matrix_type& A);
```

Dabei soll 'VecNormSup' die Supremumsnorm für Vektoren berechnen, 'VecNorm2' die Euklidische Norm für Vektoren berechnen und 'MatNormRowSum' die Zeilensummennorm für Matrizen berechnen. Der Typ 'number' entspricht hierbei dem C++ Typ 'double'.
Sie können sich hierbei an den Beispielen in 'DemoPlugin' orientieren.
Kompilieren Sie nach Änderungen an dem 'DemoPlugin' ug4 jeweils wieder, indem Sie in Ihrem build-Verzeichnis 'make' (bzw 'nmake' unter Win10) eingeben.

Zum Testen Ihrer Funktionen erstellen Sie eine Kopie des Skripts

```
ug4/apps/Examples/laplace.lua
```

und erweitern dieses mittels des 'print' Befehls nach dem Lösungsvorgang (ca Zeile 135) um Normberechnungen für A, u und b. Geben Sie die berechneten Normen aus, z.B. mittels

```
print("VecNorm2(u): " .. VecNorm2(u))
```

Aufgabe 2 (Neumann-Problem, 4P + 2P + 6P)
Ausgehend vom Skript 'ug4/apps/Examples/laplace.lua' soll das Neumann-Problem umgesetzt werden:

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega = (0,1) \times (0,1) \\ \frac{\partial u}{\partial n} &= \varphi \text{ auf } \Gamma = \partial\Omega. \end{aligned}$$

a) Diskretisieren Sie Ω über ein 2×2 Viereckgitter. Weisen Sie separate Subsets für den nördlichen, den östlichen, den südlichen und den westlichen Rand zu. Desweiteren sollte es ein Subset für alle inneren Elemente, Kanten und Knoten geben. Zusätzlich sollte der zentrale Knoten bei $(0.5, 0.5)$ in einem separaten Subset (z.B. 'Dirichlet') liegen.
Bitte passen Sie im Simulationsskript die `requiredSubsets` entsprechend den von Ihnen gewählten Subsetnamen an (ca Zeile 50).

Das Neumann-Problem soll für

$$f(x,y) = -12 \cdot ((x-0.5)^2 + (y-0.5)^2)$$

mit folgenden Neumann Randbedingungen gelöst werden:

$$\begin{aligned} \text{South:} \quad & \varphi(x,y) = -4 \cdot (y-0.5)^3 \\ \text{North:} \quad & \varphi(x,y) = 4 \cdot (y-0.5)^3 \\ \text{West:} \quad & \varphi(x,y) = -4 \cdot (x-0.5)^3 \\ \text{East:} \quad & \varphi(x,y) = 4 \cdot (x-0.5)^3 \end{aligned}$$

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. "Inner").
Dem Objekt `neumannBnd` können dabei über `add` beliebig viele Randbedingungen hinzugefügt werden.

Setzen Sie außerdem für den zentralen Knoten eine Dirichlet-Bedingung mit Wert 0.

Ändern Sie die Zeile

```
solver = util.solver.CreateSolver(solverDesc)
```

zu

```
solver = util.solver.CreateSolver(''lu'')
```

(ca Zeile 122 im ursprünglichen Skript).

Lassen Sie Ihr Skript für die Verfeinerungsstufen '-numRefs 3', '-numRefs 4' und '-numRefs 5' laufen und visualisieren Sie die Lösung.
Was beobachten Sie bei der Lösung mit zunehmender Verfeinerung (nutzen Sie z.B. das 'Warp by Scalar' Tool in ParaView)?
Weshalb ist dieses Verhalten der Lösung bei zunehmender Verfeinerung zu erwarten?

Geben Sie nach dem Lösungsvorgang die Normen `VecNormSup` und `VecNorm2` für u sowie `MatNormRowSum` für A aus.

Hinweis: f wird als Quellterm bezeichnet und kann über 'elemDisc:set_source(...)' analog zum Beispiel in laplace.lua angegeben werden.

b) Ersetzen Sie Randbedingungen und Quellterm aus Aufgabenteil a) wie folgt:
Neumann-Randbedingungen:

$$\begin{aligned} \text{South:} \quad & \varphi(x,y) = 2 \cdot (y - 0.5) \\ \text{North:} \quad & \varphi(x,y) = -2 \cdot (y - 0.5) \\ \text{West:} \quad & \varphi(x,y) = 2 \cdot (x - 0.5) \\ \text{East:} \quad & \varphi(x,y) = -2 \cdot (x - 0.5) \end{aligned}$$

Quellterm:

$$f = -4$$

Für den zentralen Knoten soll wie in a) die Dirichlet-Bedingung 0 gesetzt werden.

Lassen Sie das Skript für 3, 4 und 5 Verfeinerungen laufen und vergleichen Sie die Lösungen (z.B. mittels ParaView).

Wie erklären Sie das von a) abweichende Verhalten der Lösung bei zunehmender Verfeinerung?

c) Im Folgenden soll die Korrektur durch Erweiterung des Systems auf alle Knoten verteilt werden (vergleiche Skript). Dazu sollen Matrix A, Lösung u und rechte Seite b analog zum Skript angepasst werden:

$$\overline{A} = \begin{pmatrix} A & 1 \\ 1 & 0 \end{pmatrix}, \quad \overline{u} = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \\ \lambda \end{pmatrix}, \quad \overline{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \\ 0 \end{pmatrix}$$

Implementieren Sie im 'DemoPlugin' eine Funktion

```
void ExtendEquationForNeumann(matrix_type& A,
                              vector_type& u,
                              vector_type& b)
```

die eine entsprechende Anpassung für A, u und b vornimmt. Dabei sollen A, u und b selbst verändert werden. Nutzen Sie für Vektoren u die Funktion

```
u.resize(newSize);
```

wobei `newSize` die neue Größe des Vektors angebe. Die Matrix A können Sie über den Befehl

```
A.resize_and_keep_values(newNumRows, newNumCols);
```

in der Größe verändern.

Passen Sie das Skript aus Aufgabenteil a) an, indem Sie die Dirichlet-Bedingung entfernen und stattdessen vor dem Lösungsvorgang mit Hilfe Ihrer Funktion das Gleichungssystem anpassen:

```
ExtendEquationForNeumann(A, u, b)
solver:init(A, u)
solver:apply(u, b)
```

WICHTIG: Nutzen Sie analog zu Aufgabenteil a) die Einstellung
`solver = util.solver.CreateSolver(''lu'')`

Zusätzlich zu den Normberechnungen aus Aufgabe a) soll nun auch vor dem Aufruf von `ExtendEquationForNeumann` die Zeilensummennorm von A ausgegeben werden.

Lassen Sie das Skript wieder für 3, 4 und 5 Verfeinerungen laufen und vergleichen Sie die Lösungen (z.B. mittels ParaView). Erklären Sie weshalb das Verhalten der Lösung in c) von dem in a) abweicht. Welche Variante würden Sie bevorzugen? Begründen Sie dies.

Erläutern Sie, wie die unterschiedlichen Normen von A aus Aufgabenteil a) und c) zu erklären sind.