DEPARTMENT MATHEMATIK
UND INFORMATIK
DER UNIVERSITÄT ZU KÖLN

Prof. Dr. A. Klawonn
J. Knepper, M. Sc.

Sommersemester 2019

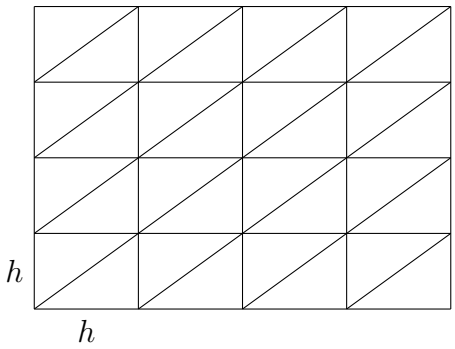
15. Mai 2019

7. Übung zu Wissenschaftliches Rechnen II

Aufgabe 1 (Locking): (4 Punkte)
Das Einheitsquadrat $\overline{\Omega} = [0,1]^2$ sei, wie abgebildet, gleichmäßig in Dreiecke mit Seitenlänge h zerlegt. Es sei $V_0^h(\Omega) \subset (H_0^1(\Omega))^2$ der Finite-Elemente-Raum der $\mathcal{P}_1$-Elemente mit homogenen Dirichletrandwerten auf $\partial\Omega$. Zeigen Sie, dass

$$v^h \in \{w^h \in V_0^h(\Omega) : \ \mathrm{div}(w^h) = 0\} \quad \Rightarrow \quad v^h \equiv 0 \text{ auf } \overline{\Omega}$$

gilt.



Aufgabe 2 (Produktraum: Koerzivität): (4 Punkte)
Wir definieren den Produktraum $V := (H_0^1(\Omega))^d \times L^2(\Omega)$ mit der zugehörigen Norm, $||(u,p)||_V^s = ||u||_{H^1(\Omega)}^s + \|p\|_{L^2(\Omega)}^s$, $s = 2$. Betrachten Sie die Bilinearform $A : V \times V \to \mathbb{R}$ mit

$$A((u,p),(v,q)) := 2\mu(\varepsilon(u),\varepsilon(v))_{L^2(\Omega)} + (q,\mathrm{div}(u))_{L^2(\Omega)} + (p,\mathrm{div}(v))_{L^2(\Omega)} - t^2(p,q)_{L^2(\Omega)}.$$

Zeigen Sie, dass $A(\cdot,\cdot)$ nicht V-elliptisch ist.

Aufgabe 3 (Bubble-Funktion): $(4 + 1 = 5$ Punkte)
Es sei $T = (a_1, \ldots, a_n)$ ein Dreieck oder Tetraeder und

$$\hat{L}_i(x) := \det \begin{pmatrix} a_1 & \ldots & a_{i-1} & x & a_{i+1} & \ldots & a_n \\ 1 & \ldots & 1 & 1 & 1 & \ldots & 1 \end{pmatrix}.$$

Durch $\varphi_i(x)$ seien die nodalen $\mathcal{P}_1$-Basisfunktionen gegeben.

1. Zeigen Sie, dass
$$L_i(x) = \varphi_i(x), \quad L_i(x) = \frac{\hat{L}_i(x)}{\hat{L}_i(a_i)},$$
gilt und somit die Bubble-Funktion durch $b(x) = c\prod_i \varphi_i(x)$, $c \in \mathbb{R}_{\neq 0}$, dargestellt werden kann.

2. Plotten Sie in Matlab eine Bubble-Funktion auf dem Referenzdreieck. Normieren Sie diese, sodass $b(c) = 1$ erfüllt ist, wobei c der Schwerpunkt des Dreiecks sei. Für die Gittergenerierung können Sie den folgenden Code nutzen:

```
[xx,yy] = meshgrid(linspace(0,1,30));
x = [xx(:),yy(:)];
x = x(x(:,1) + x(:,2) <= 1,:);
tri = delaunay(x(:,1),x(:,2));
```

Programmieraufgabe (Stokes): $(3 + 7 + 10 = 20$ Punkte)

a) Verwenden Sie die Programme des 6. Übungsblattes, a.i) und a.ii) und vergleichen Sie die Konditionszahlen der assemblierten Systeme mit dem `condest`-Befehl in Matlab (Schätzung bzgl. der 1-Norm).

b) Assemblieren Sie nun die einzelnen Blöcke des Sattelpunktproblems zur Aufgabe a.i) (6. Übungsblatt) in FEniCS und fügen Sie diese in Matlab zusammen. Setzen Sie dort die Randwerte, berechnen Sie die Lösung mit GMRES und plotten Sie die Geschwindigkeits- und Drucklösung auf dem $\mathcal{P}_1$-Gitter.

Hinweise:

1. Die Nummerierung in Python ist nullbasiert.
2. Die FEniCS-Triangulierung `mesh.cells()` wird als Integer-Array gespeichert. Um sie in Matlab z.B. für `triplot` verwenden zu können, muss sie im `double`-Format gespeichert sein.
3. Ein Numpy-Array wird als Zeilenvektor exportiert.

Schreiben Sie daher am Anfang Ihres Matlab-Skriptes etwas in der folgenden Art:

```
b = b';
triP1 = double(triP1)+1;
```

c) Lösen Sie die Stokesschen Gleichungen

$$\begin{aligned} -\Delta u + \nabla p &= f && \text{in } \Omega = (0,1)^2, \\ \operatorname{div}(u) &= 0 && \text{in } \Omega, \\ u &= u|_{\partial\Omega} && \text{auf } \partial\Omega, \end{aligned}$$

wobei die rechte Seite durch

$$f(x,y) = \begin{pmatrix} 28\pi^2 \sin(4\pi x)\cos(4\pi y) \\ -36\pi^2 \cos(4\pi x)\sin(4\pi y) \end{pmatrix}$$

gegeben ist und die Lösung (u, p) durch

$$\begin{aligned} u(x,y) &= \begin{pmatrix} \sin(4\pi x)\cos(4\pi y) \\ -\cos(4\pi x)\sin(4\pi y) \end{pmatrix}, \\ p(x,y) &= \pi \cos(4\pi x)\cos(4\pi y). \end{aligned}$$

Es gilt $\int_\Omega p(x,y)\,\mathrm{d}x = 0$. Berechnen Sie die Lösung in FEniCS basierend auf der Formulierung mit einem Lagrangeschen Multiplikator für die Integralmittelwertbedingung an den Druck.

Implementieren Sie die folgenden Elemente, gepaart mit bestimmten Gitterauflösungen:

	Element	n
1	$\mathcal{P}_1$–$\mathcal{P}_0$	18
2	$\mathcal{P}_1$–$\mathcal{P}_1$	20
3	$\mathcal{P}_2$–$\mathcal{P}_0$	11
4	$\mathcal{P}_2$–$\mathcal{P}_1$ (Taylor-Hood)	11
5	$\mathcal{P}_2$–$\mathcal{P}_2$	10
6	MINI	13
7	Crouzeix-Raviart	13
8	$\mathcal{P}_1$–$\mathcal{P}_1$–stab.	20
9	$\mathcal{P}_2$–$\mathcal{P}_2$–stab.	10

- MINI: $(\text{Bubble} + \mathcal{P}_1)^2 \times \mathcal{P}_1$

  Das Geschwindigkeitselement kann wie folgt implementiert werden:

```
P1 = FiniteElement('P', mesh.ufl_cell(), degree = 1)
bubble = FiniteElement("Bubble", mesh.ufl_cell(), degree = 3)
element_vel = VectorElement( NodalEnrichedElement(bubble,P1),
   dim=mesh.geometric_dimension())
```

- Crouzeix-Raviart (CR): $\text{CR}^2 \times \mathcal{P}_0$

  Achtung: Es gibt in der Literatur auch ein anderes Element, welches als Crouzeix-Raviart-Element bezeichnet wird. Hier haben wir in der Geschwindigkeit den Crouzeix-Raviart-Raum, d.h. $\mathcal{P}_1$-Ansatzfunktionen auf den Elementen, die stetig in den Kantenmittelpunkten sind (nicht-konform).

- Die inf-sup-stabilisierten[1] Elemente (8 und 9) gleicher Geschwindigkeit- und Druckordnung werden genau wie die nicht-stabilisierten implementiert; es müssen jedoch die Bilinearform `a` und das lineare Funktional `F` modifziert werden:

```
h = 0.5*(mesh.hmax()+mesh.hmin())
beta = 0.2
delta = beta*h*h
a -= delta*dot(grad(q), grad(p))*dx
F -= delta*dot(grad(q), f)*dx
```

  Bemerkung: Wir nutzen nicht die volle Stabilisierung entsprechend [3] für $\mathcal{P}_2 - \mathcal{P}_2$-Elemente.

  Achtung: Die Vorzeichen müssen konsistent zum Vorzeichen der Divergenzbedingung sein. Wird $\text{div}(u) = 0$ mit $+\int_\Omega \text{div}(u) q \, \mathrm{d}x$ anstatt $-\int_\Omega \text{div}(u) q \, \mathrm{d}x$ in die Variationsformulierung über den Produktraum integriert, so müssen auch die Vorzeichen oben invertiert werden, d.h. die Stabilisierung wird dann mit `a +=` und `F +=` integriert. Implementieren sie also folgende Variationsformulierung: Finde $(u, p) \in \hat{X}^h \times M^h$, sodass für alle $(v, q) \in X^h \times M^h$ gilt:

$$\begin{aligned} \big(\nabla u, \nabla v\big)_{L^2(\Omega)} - \big(\,\text{div}(v), p\big)_{L^2(\Omega)} &= \big(v, f\big)_{L^2(\Omega)}, \\ -\big(\,\text{div}(u), q\big)_{L^2(\Omega)} - \delta\big(\nabla q, \nabla p\big)_{L^2(\Omega)} &= -\delta\big(\nabla q, f\big)_{L^2(\Omega)}. \end{aligned}$$

Nutzen Sie zur Diskretisierung einmal das `mshr`-Modul mit `generate_mesh(domain ,n)` und einmal `UnitSquareMesh(n,n)`. Berechnen Sie mit `errornorm` den relativen L^2-Fehler

[1] FEniCS - Stokes stabilized
[2] S. 147 in Elman et al., *Finite Element and Fast Iterative Solvers*
[3] 2nd Ed., 2014.; S. 288 in Donea und Huerta, *Finite Element Methods for Flow Problems*, 2003.

zur exakten Lösung (nutzen Sie stets `degree = 7` für die Quadraturordnung). Geben Sie die Fehler zu den zwei Gittern und den jeweils 9 Verfahren an und geben Sie zudem stets die Anzahl Freiheitsgrade an. Kommentieren Sie die Ergebnisse.

Abgabe:

- **Theorie:** Bis Mittwoch, 22. Mai 2019, 14:00 Uhr,
- **Programm:** Bis Mittwoch, 29. Mai 2019, 14:00 Uhr,

im entsprechenden Kasten in Raum 3.01 des Mathematischen Instituts.