

HANA Numerik SEP Projekt

Simon Stingelin (stiw@zhaw.ch)

15. Dezember 2022



Gruppe: Alguacil Dominique, Arslan Selim, Ercihan Kaya

Thema: Boussinesq Approximation

Lerninhalt:

- Berechnen der schwachen Gleichung.
- Anwenden der Methode der finiten Elemente auf ein physikalische Problem.
- Taylor-Hood Diskretisierung der Navier-Stokes Gleichung.
- Lösen eines gekoppelten nichtlinearen Randwert Problems.

Abgabe:

- Die Abgabe erfolgt als Kurzbericht in Form eines **PDF**-Dokuments über die **moodle** Abgabe bis zum in der **moodle** Abgabe definierten spätesten Abgabetermin.
- Die numerischen Resultate sind im Bericht dokumentiert und können mit lauffähigen Skripts nachvollzogen werden.

Bewertung:

Note 1	Note 3	Note 4	Note 4.5	Note 5	Note 5.5	Note 6
Keine Abgabe	Nicht bewertbar: die Aufgaben wurden nicht ernsthaft bearbeitet.	Die Aufgaben wurden knapp bearbeitet und sind nur im Ansatz dokumentiert.	Die Aufgaben wurden halbwegs bearbeitet und sind sehr knapp Ansatz dokumentiert.	Die Aufgaben wurden bearbeitet und dokumentiert.	Die Aufgaben wurden detailliert bearbeitet und dokumentiert.	Die Aufgaben wurden detailliert bearbeitet und dokumentiert. Es wurden noch zusätzliche Aspekte bearbeitet.

1 Aufgabestellung

Wir betrachten die Strömung in einem Heissluftballon (vgl. Titelbild) oder damit der Rechenaufwand überschaubar wird, die Wärme getriebene Strömung in einer Himmelslaterne (vgl. Abb. 1). Wobei wir die Form des Heissluftballons beibehalten und die Geometrie in einer rechenbaren Grösse definieren. Strömungsmechanisch betrachtet, ist durchaus auf Grund der Dimensionen eine turbulente und nicht eine laminare Strömung in einem Heissluftballon zu erwarten.

1.1 Zweidimensionale Lösung

In einem ersten Schritt betrachten wir das zweidimensionale Problem, welches danach auf ein rotationssymmetrisches Problem erweitert wird.



Abbildung 1: Himmelslaterne

Strömungen die durch Dichtevariation aufgrund von Temperaturschwankungen verursacht werden, können mit Hilfe der *Boussinesq-Approximation* (Temperaturschwankung nur in Dichte- und Druckvariation berücksichtigt) modelliert werden:

$$\partial_t T - \operatorname{div} \kappa \nabla T + \mathbf{u} \cdot \nabla T = 0 \quad (1a)$$

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \Delta \mathbf{u} = -\frac{1}{\rho} \nabla p - \mathbf{g} \alpha (T - T_0) \quad (1b)$$

$$\operatorname{div} \mathbf{u} = 0, \quad (1c)$$

wobei T das Temperaturfeld, \mathbf{u} das Strömungsfeld, p das Druckfeld und $\mathbf{g} = (0, g)^T$ das Gravitationsfeld bezeichne. Im Projekt sollen stationäre Lösungen der Boussinesq-Approximation berechnet werden. Initial wird der nichtlinearen Term $(\mathbf{u} \cdot \nabla) \mathbf{u}$ der Navier-Stokes Gleichung vernachlässigt. Wir gehen daher von einer kriechenden Strömung aus (Stokes-Flow) und suchen Lösungen des Gleichungssystems

$$-\operatorname{div}(\kappa \nabla T) + \mathbf{u} \cdot \nabla T = 0 \quad (2a)$$

$$-\nu \Delta \mathbf{u} = -\frac{1}{\rho} \nabla p - \mathbf{g} \alpha (T - T_0) \quad (2b)$$

$$\operatorname{div} \mathbf{u} = 0. \quad (2c)$$

Für die Modellparameter der beiden Gleichungen (1) und (2) gehen wir für die warme Luft im Ballon vom idealen Gasgesetz aus. In der Tabelle 1 sind die Modellparameter aufgelistet. Um die Berechenbarkeit des Projekts zu gewähren, skalieren wir die Geometrie kleiner. Die Dichte ρ , dynamische Viskosität η und Wärmeausdehnung α von Luft ist von der Temperatur abhängig gegeben, wobei darauf Acht gegeben werden muss, dass die Temperatur in Kelvin eingesetzt wird.

Wärmeleitfähigkeit	κ_{Luft}	0.0262 W/(m K)
Spezifische Gaskonstante	R_s	287.058 J/(kg K)
Atmosphärischer Druck	p_0	1013.25 mBar
Dichte trockene Luft	$\rho_{\text{Luft}} = \frac{p_0}{R_s T}$	kg/m ³ als ideales Gas
kinetische Gastheorie	$\eta_{\text{Gas}} = \frac{5\sqrt{\pi m k_B T}}{16\pi\sigma^2\Omega^{(2,2)*}}$	
dynamische Viskosität	$\eta_{\text{Luft}} \approx \frac{0.0182}{\sqrt{293.15}} \sqrt{T}$	mPa s
kinematische Viskosität	$\nu = \eta/\rho$	
Wärmeausdehnungskoeffizient	$\alpha_{\text{Luft}} = 1/T$	1/K

Tabelle 1: Parameter Annahmen für Luft, mehrheitlich basierend auf dem idealen Gasgesetz. Achtung: Temperatur T in Kelvin [K] einsetzen.

Auf der unteren Geraden (in der Geometrie als „inlet“ bezeichnet) gehen wir von einer Temperatur von 100°C aus. Über die Oberfläche des Ballons nehmen wir einen Wärmefluss an, welcher von der Temperatur abhängig ist. Auf der Symmetrie-Achse benutzen wir ideale Neumann Randbedingungen. Damit folgt für die *Randbedingungen*:

1. für das *Temperaturfeld* T

$$T(\Gamma_{\text{inlet}}) = 100^\circ\text{C} \quad (3a)$$

$$\kappa \frac{\partial}{\partial n} T(\Gamma_{\text{outer}}) = \beta (T - T_{\text{amb}}) \quad (3b)$$

$$\partial_n T(\Gamma_{\text{sym}}) = 0 \quad (3c)$$

2. für die *Strömungsverteilung* \mathbf{u}

$$\mathbf{u}(\Gamma_{\text{outer, inlet}}) = 0 \quad (3d)$$

$$u_x(\Gamma_{\text{sym}}) = 0 \quad (3e)$$

wobei $\Gamma_{\text{outer, inlet}} = \Gamma_{\text{outer}} \cup \Gamma_{\text{inlet}}$.

3. für die *Druckverteilung* p ist keine Randbedingung notwendig.

1.2 Axialsymmetrische Lösung

Wir erweitern nun die Gleichung auf axialsymmetrische Lösungen, welche Lösungen im \mathbb{R}^3 sind. Der Rechenaufwand für die Berechnung axialsymmetrischer Lösungen ist bedeutend kleiner als für volle 3d-Lösungen. Unter Benutzung von Zylinderkoordinaten (r, φ, z) und

der Annahme, dass die Lösung invariant in φ -Richtung ist (axialsymmetrisch), folgt das Gleichungssystem:

$$-\frac{1}{r} \frac{\partial}{\partial r} \left(\kappa r \frac{\partial T}{\partial r} \right) - \frac{\partial}{\partial z} \kappa \frac{\partial T}{\partial z} + u_r \frac{\partial T}{\partial r} + u_z \frac{\partial T}{\partial z} = 0 \quad (4a)$$

$$-\nu \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u_r}{\partial r} \right) + \frac{\partial^2 u_r}{\partial z^2} \right) + \nu \frac{u_r}{r^2} + u_r \frac{\partial u_r}{\partial r} + u_z \frac{\partial u_r}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial r} \quad (4b)$$

$$-\nu \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u_z}{\partial r} \right) + \frac{\partial^2 u_z}{\partial z^2} \right) + u_r \frac{\partial u_z}{\partial r} + u_z \frac{\partial u_z}{\partial z} = -\frac{1}{\rho} \frac{\partial p}{\partial z} - g \alpha T \quad (4c)$$

$$\frac{1}{r} \frac{\partial}{\partial r} (r u_r) + \frac{\partial u_z}{\partial z} = 0. \quad (4d)$$

In der Abbildung 2 sind die Lösungen für die beiden Situationen dargestellt. Die ebene Lösung muss als Lösung für ein Profil interpretiert werden, welches in z Richtung (aus der Ebene) beliebig weit ausgedehnt ist. Hingegen im axialsymmetrischen Fall (Abb. 2b) sehen wir die Lösung im axialsymmetrischen Volumen.

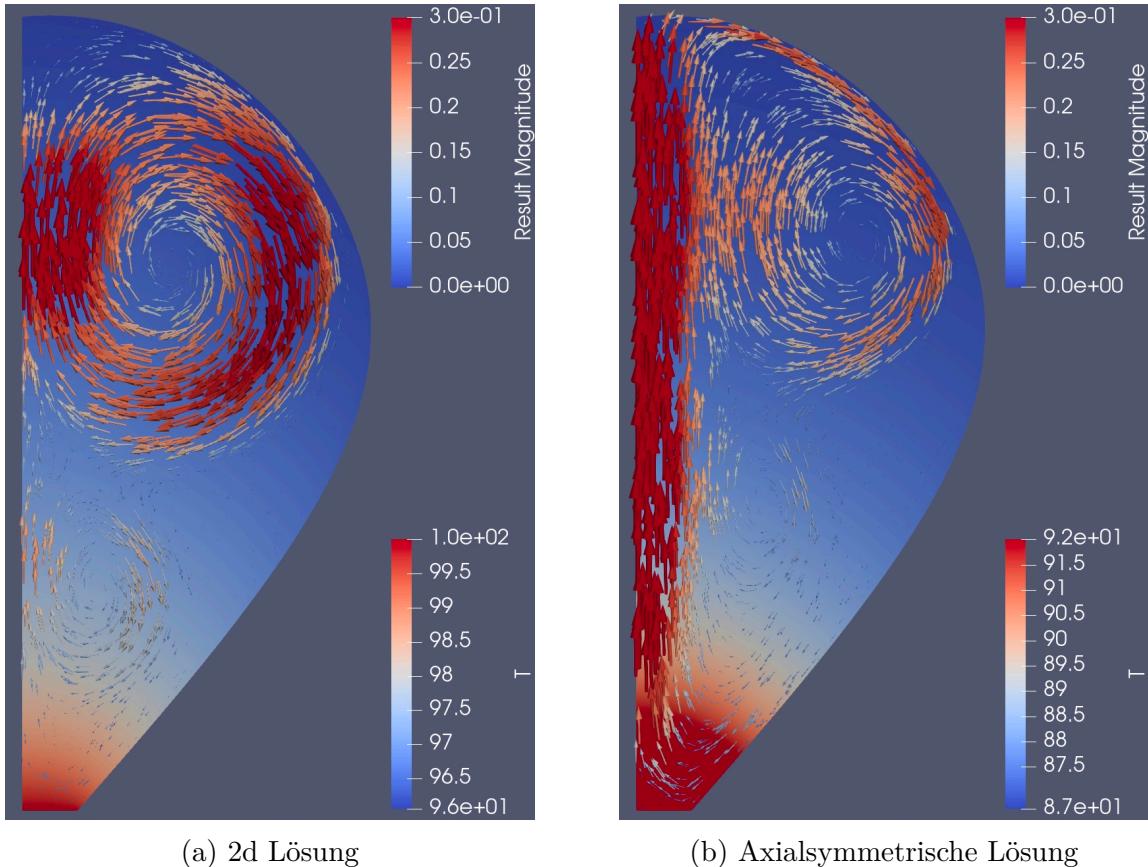


Abbildung 2: Vergleich der Lösungen

Input zur Numerik

Für die Diskretisierung der Strömung benutzen wir eine stetige H^1 -Vektor Diskretisierung mit Ordnung 2 plus zusätzliche innere Ordnung 3 und für den Druck eine unstetige

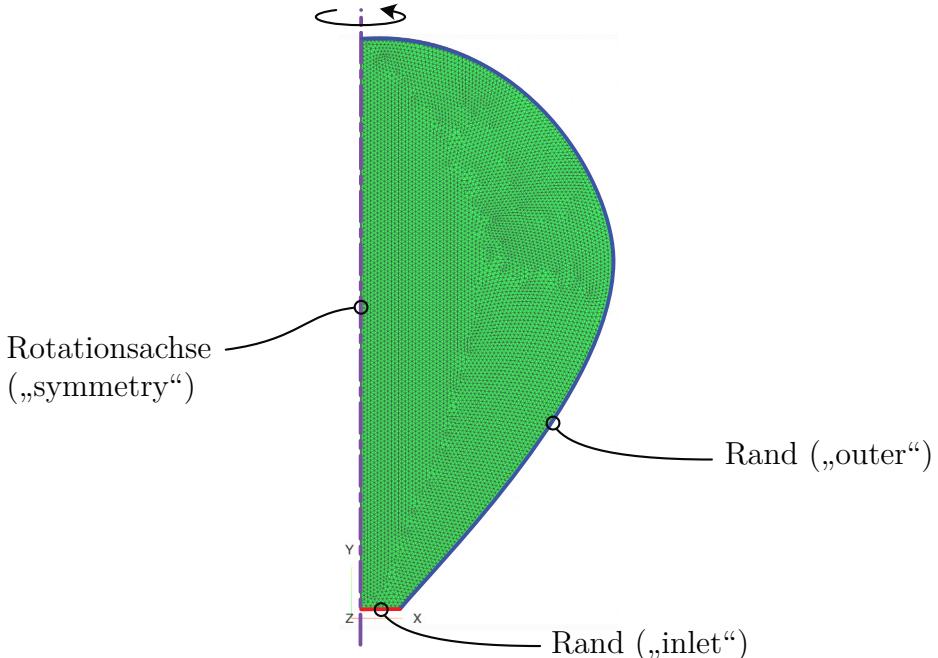


Abbildung 3: Rechengebiet, Mesh.

(diskontinuierliche) Diskretisierung mit Ordnung 1. Die klassische Taylor-Hood finite Element Diskretisierung benutzt sowohl für das Strömungsfeld (Ordnung 2) wie auch für den Druck (Ordnung 1) eine stetige Diskretisierung. (Sie können solche Lösungen berechnen, in dem Sie den FE-Space \mathbb{Q} auf H^1 ändern und das Flag `innerorder` im `Vr` und `Vz` löschen).

$$\mathbf{u}(\mathbf{x}) = \begin{pmatrix} u_r(\mathbf{x}) \\ u_z(\mathbf{x}) \end{pmatrix} \in V = P^{2+} \subset H^1(\Omega), \quad p(\mathbf{x}) \in Q = P^{1,dg} \subset L_2(\Omega).$$

```

1 order = 2
2 W = H1(mesh, order=order, dirichlet="inlet")
3 Vr = H1(mesh, order=order, orderinner=order+1,
4         dirichlet="sym|outer|inlet")
5 Vz = H1(mesh, order=order, orderinner=order+1,
6         dirichlet="outer|inlet")
7 Q = L2(mesh, order=order-1)
8 X = FESpace([W,Vr,Vz,Q])
9
10 r = x
11 z = y

```

Listing 1: FEM space definition

Die GridFunction setzt sich nun aus diesen vier Funktionen zusammen. Die Dirichlet Randbedingung für die Temperatur muss auf dem Rand „inlet“ gesetzt werden (vgl. Listing 2).

```

1 Temp = CoefficientFunction(gfx.components[0])
2 velocity = CoefficientFunction(gfx.components[1:3])
3 pressure = CoefficientFunction(gfx.components[3])
4

```

```

5 # Dirichlet Randbedingung fuer das Temperaturfeld
6 gfx.components[0].Set(100, definedon=mesh.Boundaries('inlet'))
7
8 Draw(Temp, mesh, 'T')
9 Draw(velocity, mesh, 'v')

```

Listing 2: Gridfunction inkl. Dirichlet Randbedingung für die Temperatur.

Für die Definition der schwachen Gleichung können Sie folgende Proxy-Funktionen (Test und Trial Funktionen) benutzen:

```

1 T,ur,uz,p = X.TrialFunction()
2 S,vr,vz,q = X.TestFunction()

```

Listing 3: test and trial function definition

Da wir die Parameter für Luft aus dem Idealen Gasgesetz benutzen, ist auch das Stokes-Gleichungssystem (2) nichtlinear. Würde man konstante Parameter nutzen (unabhängig von der Temperatur), so würden wir für die Strömung ohne Kopplung mit der Wärmeleitung ein lineares Gleichungssystem erhalten. Das bedeutet, dass Sie aufgrund der Kopplung und Temperaturabhängigkeit der Parameter in jedem Fall eine Newton-Methode benötigen. Implementiere Sie diese möglichst universell so, dass in der Sie die (nichtlineare) Bilinearform und Gridfunction übergeben (vgl. Listing 4).

```

1 def NewtonSteps(a, gfx, nMax=10, tol = 1e-9):
2     X = gfx.space
3     # vectors for newton steps
4     res = gfx.vec.CreateVector()
5     du = gfx.vec.CreateVector()
6
7     for it in range(nMax):
8         print('Iteration', it)
9         with TaskManager():
10             <<< snipp, do it self >>>

```

Listing 4: Newton-Methode

2 Aufgaben

Aufgabe 1

1. Wie lautet die schwache Formulierung des Gleichungssystems (2)?
2. (optional) Wie lautet die schwache Formulierung des Gleichungssystems (4)?

Vorgehen: Testen Sie jede Gleichung mit eigenen Testfunktionen S, v_x, v_y, q (vlg. Listing 3, wobei ur, uz mit ux, uy ersetzt werden muss).

- Die Stokes-Gleichung für die Strömung ist vektorwertig.

- Integrieren Sie die Laplace Operatoren jeweilen partiell, sprich wir haben nur Ableitungen 1. Grad in den schwachen Gleichungen.
- Den Gradienten des Drucks in der Gleichung (2b) integrieren wir ebenfalls partiell.
- $\Delta \mathbf{u}$ bedeutet den Laplace Operator Komponentenweise angewandt, daher

$$\Delta \mathbf{u} = \begin{pmatrix} \Delta u_x \\ \Delta u_y \end{pmatrix}$$

- Verifizieren Sie Ihre schwachen Gleichungen anhand des Listings 5. Weisen Sie die einzelnen Terme den Definitionen zu (**Achtung**: im Listing ist die Implementierung der axialsymmetrischen Gleichung).

Für die Implementierung (axialsymmetrischer Fall) der Gleichungen können Sie den Code in Listing 5 und 6 benutzen.

```

1 # Stokes flow
2 div_u = ur/r+grad(ur)[0]+grad(uz)[1]
3 div_v = vr/r+grad(vr)[0]+grad(vz)[1]
4
5 u = CoefficientFunction((ur,uz))
6 conv = u*grad(T)
7
8 a = BilinearForm(X, symmetric=False)
9 a += (kappa*grad(T)*grad(S)+conv*S)*r*dx
10 a += gamma*(T-Tamb)*S*r*ds('outer')
11 a += (nu*ur/r**2*vr + nu*grad(ur)*grad(vr) + nu*grad(uz)*grad(vz)
12     + 1/rho*p*div_v + g*alpha*grad(T)[1]*vz)*r*dx
13 a += div_u*q*r*dx

```

Listing 5: Definition bilinearform Stokes Gleichung

```

1 # Navier-Stokes flow
2 aNS = BilinearForm(X, symmetric=False)
3 aNS += (kappa*grad(T)*grad(S)+conv*S)*r*dx
4 aNS += gamma*(T-Tamb)*S*r*ds('outer')
5 aNS += (nu*ur/r**2*vr + nu*grad(ur)*grad(vr) + nu*grad(uz)*grad(vz)
6     + u*(grad(ur)*vr+grad(uz)*vz)
7     + 1/rho*p*div_v+g*alpha*grad(T)[1]*vz)*r*dx
8 aNS += div_u*q*r*dx

```

Listing 6: Definition bilinearform Navier-Stokes Gleichung

Aufgabe 2

1. Berechnen Sie mit Hilfe der **Newton-Iteration** die initiale Lösung für die **axial-symmetrische** Boussinesq Gleichung (4), jedoch ohne Konvektionsterm ($\mathbf{u} \cdot \nabla \mathbf{u}$) (Stokes-Strömung). Sie können dazu das Listings 5 benutzen. Wir benutzen hier eine zu $30 \times$ grössere Viskosität, welche wir im nächsten Schritt sukzessive verkleinern.

2. Berechnen Sie mit Hilfe der Lösung aus dem Schritt oben die Lösung der vollen axialsymmetrischen Boussinesq Gleichung (4), benutzen Sie das Listing 6. Reduzieren Sie sukzessive die Viskosität um die Lösung für Luft zu erhalten (vgl. Listing 7).

```

1 # initial with Stokes flow
2 NewtonSteps(a, gfx, nMax=15)
3
4 # iterativ for Navier-Stokes flow
5 scaleMus = 1+np.linspace(0,30**(.5),30)[::-1]**2
6
7 for s in scaleMus:
8     print(s)
9     scaleMu.Set(s)
10    ret = NewtonSteps(aNS, gfx, nMax=25, tol = 5e-9)
11    if ret > 0:
12        print('no convergence')
13        break

```

Listing 7: Berechnung mittlere Temperatur

Aufgabe 3

Berechnen Sie analog zur Aufgabe 2 die Lösungen der zweidimensionalen Boussinesq Gleichung. Sie können die Listings 5 und 6 entsprechend den schwachen Gleichungen aus der Aufgabe 1 modifizieren.

Aufgabe 4

- Welchen Auftrieb haben wir bei einer Umgebungstemperatur von $T_{\text{amb}} = 0^\circ\text{C}$ und atmosphärischem Druck p_0 im axialsymmetrischen Fall?
- Der Auftrieb ist gegeben durch den Massenunterschied der kalten und warmen Luft im Volumen des Ballons (siehe Listing 8).
- Wie hoch müsste der Ballon sein, um 200kg heben zu können, wenn wir annehmen, dass sich der Auftrieb linear skalieren lässt?

```

1 rhoT = 1013.25e-3*1e5/(287.058*(273.15+Temp))
2 rho0 = 1013.25e-3*1e5/(287.058*273.15)
3
4 massHotAir = 2*np.pi*Integrate(rhoT*r,mesh)
5 massColdAir = 2*np.pi*Integrate(rho0*r,mesh)

```

Listing 8: Berechnung der Masse

Aufgabe 5 (optional)

Berechnen Sie den Auftrieb abhängig von der Umgebungstemperatur $T_{\text{amb}} \in [-10^\circ, 30^\circ]$. Da die Umgebungstemperatur als Parameter gegeben ist, können Sie diese analog zur Viskosität variieren (Zeile 9 in Listing 7).

A Geometrie

```
1 from netgen.occ import *
2 from netgen.webgui import Draw as DrawGeo
3
4 # Geometry
5 # wir skalieren die Geometrie um die Rechenbarkeit zu gewaehren
6 scale = 0.1
7 geom = MoveTo(0,0)
8 geom = geom.LineTo(.2,0)
9 geom = geom.Spline([(0.85,0.7875),(1.25,1.8),(0.9,2.5),(0,2.85)])
10 geom = geom.Close().Face()
11 geom.faces.name = 'hotair'
12 geom.edges.name = 'outer'
13 geom.edges.Min(X).name = 'sym'
14 geom.edges.Min(Y).name = 'inlet'
15 geom = geom.Scale(gp_Pnt((0,0,0)), scale)
16 geo = OCCGeometry(geom, dim=2)
17
18 # Mesh
19 mesh = Mesh(geo.GenerateMesh(maxh=scale*0.025))
20 mesh.Curve(3) # curved elements on the boundary
```

Listing 9: Definition Geometrie

```
1 kappa = Parameter(0.0262)
2 scaleMu = Parameter(30)
3 mu = scaleMu*18.2e-6/np.sqrt(293.15)*sqrt(273.15+T)
4 rho = 1013.25e-3*1e5/(287.058*(273.15+T))
5 nu = mu/rho
6 g = 9.81
7 alpha = 1/(273.15+T)
8 gamma = Parameter(1e-3)
9 Tamb = Parameter(0)
```

Listing 10: Material Parameter