Übungsblatt 7: Wärmeleitung in 2D (Dirichlet RBn)

Aufgabe 1: Dirichlet-Randbedingungen

1.1 Implementieren Sie die Funktion

Python

```
applyDirichletBCs(dofs, K, r)
```

Julia

```
applyDirichletBCs!(dofs, K, r)
```

mit der das lineare Gleichungssystem so modifiziert wird, dass die Lösungen für die Freiheitsgrade in `dofs` gleich null sind. Verwenden Sie das auf Moodle bereitgestellte Notebook **'dirichlet_test.ipynb'**.

1.2 Erzeugen Sie mit Gmsh ein FE-Netz für ein kreisförmiges Gebiet mit dem Radius $r = 2.2\,\mathrm{m}$. Die Elementgröße soll $h = 0.2\,\mathrm{m}$ betragen. Es trägt zur Übersichtlichkeit bei, wenn Sie die Gmsh-Dateien in einen Unterordner `gmsh` legen.

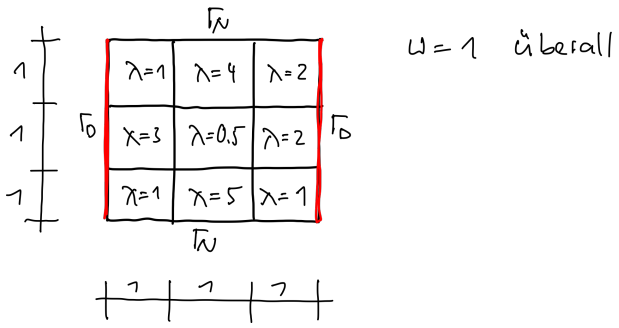
Zur Kontrolle: Sie sollten rund 500 Knoten und 1000 Elemente erhalten.

1.3 Berechnen Sie in der Datei **'heat_plate.ipynb'** die Näherungslösung der Temperaturverteilung für die Kreisplatte (Parameter λ und w frei gewählt). Vergleichen Sie die maximale Temperatur mit der exakten Lösung

$$w_{\max} = \frac{w \cdot r^2}{4 \cdot \lambda}.$$

Zusatzaufgabe: Mithilfe der Funktion der Temperaturverteilung (Notizen zur Vorlesung) können Sie die exakte Lösung in den Knotenpunkten berechnen und damit den Diskretisierungsfehler plotten.

1.4 Berechnen Sie die Wärmeverteilung für das dargestellte Wärmeleitungsproblem.



Sie müssen hierfür die Funktion `assembleKr` dahingehend anpassen

1.5 Wandeln Sie die Matrix **K** so um, dass nur die von null verschiedenen Einträge gespeichert werden. Vergleichen Sie die Zeit, die zum Lösen des LGS benötigt wird für die Ausgangsmatrix und die konvertierte Matrix.

In Python:

```python
import scipy.sparse as sp
...
KSparse = sp.csr_matrix(K)
%time That = np.linalg.solve(K, r)
%time That = sp.linalg.spsolve(KSparse, r)
```

In Julia:

```julia
using SparseArrays
...
KSparse = sparse(K)
@time That =  K \ r;
@time That =  KSparse \ r;
```