Programmierblatt 4.

Bearbeiten bis: **Sonntag, 31.05.2020**

Mit den nun auf den ersten drei Blättern erarbeiteten Methoden wollen wir uns der Lösung des Bernoullischen freien Randproblems widmen. Die Benutzung einer Randintegralformulierung mit der Darstellung der Ränder per Parametrisierungen hat dabei den Vorteil, dass keine Gebietsdiskretisierung benötigt wird. Dadurch ergibt sich ein äusserst elegantes Verfahren.

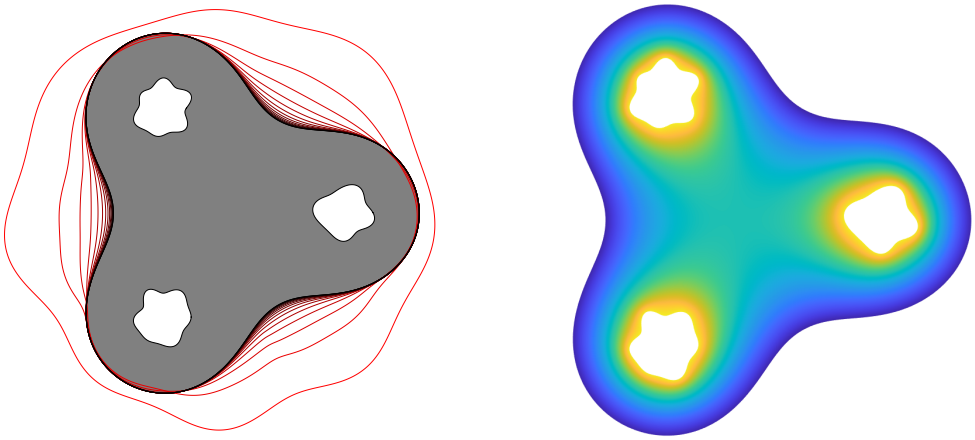
Sei $\Omega \subset \mathbb{R}^2$ ein zusammenhängendes und beschränktes Gebiet mit positivem Genus, $\operatorname{diam}(\Omega) < 1$ und einem C^k-glatten Rand, wobei $k \geq 2$ ist. Sei weiter $\Gamma_1 \cup \cdots \cup \Gamma_r = \Gamma := \partial\Omega$ die Zerlegung des Randes in dessen r Zusammenhangskomponenten, wobei Γ_1 der äussere Gebietsrand und Γ_ℓ mit $\ell \geq 2$ die inneren Gebietsränder sind. Das Randwertproblem

$$\begin{aligned} \Delta\, u(\mathbf{x}) &= 0 && \text{für } \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= g(\mathbf{x}) && \text{für } \mathbf{x} \in \Gamma_i := \Gamma_2 \cup \cdots \cup \Gamma_r, \\ \partial_{\mathbf{n}}\, u(\mathbf{x}) &= h(\mathbf{x}, \mathbf{n}) && \text{für } \mathbf{x} \in \Gamma_f := \Gamma_1. \end{aligned} \tag{1}$$

wird zu einem *freien Randproblem*, wenn wir den Rand Γ_f so bestimmen wollen, dass neben (1) auch noch

$$u(\mathbf{x}) = 0 \quad \text{für } \mathbf{x} \in \Gamma_f \tag{2}$$

gilt.



Iterative Lösung

Aussgehend von einem Rand $\Gamma_1^{(k)}$, gegeben durch eine glatte, periodische Parametrisierung $\boldsymbol{\gamma}_1^{(k)} \colon [0,1] \to \Gamma_1^{(k)}$, erhalten wir im $(k+1)$-ten Schritt den Rand $\Gamma_1^{(k+1)}$ gemäss der Update-Formel

$$\boldsymbol{\gamma}_1^{(k+1)}(s) = \boldsymbol{\gamma}_1^{(k)}(s) + d_k(s)\mathbf{z}(s)$$

für die vorgegebene Richtung $\mathbf{z} \colon [0,1] \to \mathbb{R}^2$. Um das Update $d_k(s)$ zu bestimmen, ist es naheliegend

$$u^{(k)}\big(\boldsymbol{\gamma}_1^{(k+1)}(s)\big) \stackrel{!}{=} 0$$

zu fordern, wobei $u^{(k)}$ die Lösung von (1) mit dem Rand $\Gamma_f = \Gamma_1^{(k)}$ ist. Da dies aber auf eine nichtlineare Gleichung führt, linearisieren wir $u^{(k)}$ um den Punkt $\boldsymbol{\gamma}_1^{(k)}(s)$ mit Hilfe

der Taylor-Entwicklung und fordern stattdessen

$$0 = u^{(k)}\big(\boldsymbol{\gamma}_1^{(k)}(s)\big) + d_k(s)\Big\langle \nabla_{\mathbf{x}}\, u^{(k)}\big(\boldsymbol{\gamma}_1^{(k)}(s)\big), \mathbf{z}(s)\Big\rangle.$$

Damit erhalten wir also

$$d_k(s) = -\frac{u^{(k)}\big(\boldsymbol{\gamma}_1^{(k)}(s)\big)}{\Big\langle \nabla_{\mathbf{x}}\, u^{(k)}\big(\boldsymbol{\gamma}_1^{(k)}(s)\big), \mathbf{z}(s)\Big\rangle}. \tag{3}$$

Diskretisierung des freien Randes

Unter der Annahme, dass der gesuchte Rand Γ_1 ein sternförmiger Rand mit Sternzentrum $\mathbf{0}$ ist, lassen sich die Parametrisierungen $\boldsymbol{\gamma}_1^{(k)}$ als

$$\boldsymbol{\gamma}_1^{(k)}(s) = r_k(s)\begin{bmatrix}\cos(2\pi s)\\ \sin(2\pi s)\end{bmatrix} \tag{4}$$

mit einer glatten, periodischen Radiusfunktion $r_k\colon [0,1] \to \mathbb{R}_{>0}$ ansetzen. Damit ist es nun weiter notwendig $\mathbf{z}(s)$ jeweils radial zu wählen, damit die Update-Formel wieder sicher einen sternförmigen Rand ergibt. Wir setzen daher nun speziell

$$\mathbf{z}(s) = \begin{bmatrix}\cos(2\pi s)\\ \sin(2\pi s)\end{bmatrix},$$

weswegen dann jeweils die neue Radiusfunktion durch

$$r_{k+1}(s) = r_k(s) + \tau d_k(s)$$

gegeben ist; hierbei sei zusätzlich $\tau \in (0,1]$ ein geeigneter Dämpfungsparameter. Die Radiusfunktion und damit auch das Update setzen wir als trigonometrisches Polynom mit N Termen an

$$a_0 + \sum_{j=1}^{M} a_j \cos(2\pi j s) + \sum_{j=1}^{\tilde{M}} b_j \sin(2\pi j s),$$

wobei $N = 1 + M + \tilde{M}$ mit $M = \tilde{M}$ oder $M = 1 + \tilde{M}$ ist.

Diskretisierung des Randwertproblems

Es bleibt nun zu beschreiben, wie man in jedem Schritt (3) auswerten kann, dabei verzichten wir aufgrund der Übersichtlichkeit jeweils folgend auf das Setzen des Iterationssuperskripts $^{(k)}$. Mit Hilfe der direkten Randintegralgleichungsmethode von Blatt 3 können wir die vollständigen, approximativen Cauchy-Daten von (1) in n_1 parametrisch äquidistanten Punkten auf Γ_1, $(\boldsymbol{\rho}_1^D, \boldsymbol{\rho}_1^N)$, bestimmen. Gemäss

$$\Big\langle \nabla_{\mathbf{x}}\, u\big(\boldsymbol{\gamma}_1(s)\big), \mathbf{z}(s)\Big\rangle = \partial_{\mathbf{n}}\, u\big(\boldsymbol{\gamma}_1(s)\big)\langle \mathbf{n}_{\boldsymbol{\gamma}_1(s)}, \mathbf{z}(s)\rangle + \partial_{\mathbf{t}}\, u\big(\boldsymbol{\gamma}_1(s)\big)\langle \mathbf{t}_{\boldsymbol{\gamma}_1(s)}, \mathbf{z}(s)\rangle$$

lässt sich die Ableitung vom jeweiligen u in Richtung $\mathbf{z}(s)$ mit den Normalen- und Tangentialableitung darstellen. Wenn man bedenkt, dass

$$\partial_{\mathbf{t}}\, u\big(\boldsymbol{\gamma}_1(s)\big) = \frac{\mathrm{d}_s\, u\big(\boldsymbol{\gamma}_1(s)\big)}{\|\boldsymbol{\gamma}_1'(s)\|} \quad \text{und} \quad \mathbf{t}_{\boldsymbol{\gamma}_1(s)} = \frac{\boldsymbol{\gamma}_1'(s)}{\|\boldsymbol{\gamma}_1'(s)\|}$$

gelten, so hat man, abgesehen von dem $\mathrm{d}_s\, u\big(\boldsymbol{\gamma}_1(s)\big)$, schon alle benötigten Grössen an den n_1 Punkten auf Γ_1, um (3) dort auswerten zu können. Ausgehend von den Werten $\boldsymbol{\rho}_1^D = [u_i]_{i=1}^{n_1}$ berechnen wir dazu das trigonometrische Interplolationspolynom

$$\tilde{u}(s) = a_0 + \sum_{j=1}^{m} a_j \cos(2\pi j s) + \sum_{j=1}^{\tilde{m}} b_j \sin(2\pi j s),$$

mit $n_1 = m + \tilde{m}$ wobei $m = \tilde{m}$ oder $m = 1 + \tilde{m}$ und leiten dann $\tilde{u}(s)$ nach s ab, was auf das trigonometrische Polynom

$$\mathrm{d}_s\, \tilde{u}(s) = \sum_{j=1}^{\tilde{m}} (2\pi j b_j) \cos(2\pi j s) + \sum_{j=1}^{m} (-2\pi j a_j) \sin(2\pi j s)$$

führt, welches dann an den n_1 Punkten ausgewertet werden kann. Die so für die n_1 Punkte von Γ_1 ausgerechneten Werte von d_k können dann benutzt werden, um ein trigonometrische Interplolationspolynom für d_k zu bestimmen, welches dann noch nach N Termen abgeschnitten wird, um dann das Radiusfunktionupdate zu erhalten. Dabei entspricht das Abschneiden des trigonometrischen Polynoms der L^2-Bestapproximation.

Trigonometrische Polynome und Interplolation

Ein trigonometrisches Polynom mit n Termen kann durch

$$a_0 + \sum_{j=1}^{m} a_j \cos(2\pi j s) + \sum_{j=1}^{\tilde{m}} b_j \sin(2\pi j s)$$

dargestellt werden, wobei $n = m + \tilde{m}$ mit $m = \tilde{m}$ oder $m = 1 + \tilde{m}$ ist. Dies entspricht einer Darstellung mit Koeffizienten bezüglich der wohl offensichtlichsten Basis. Andernseits kann das trigonometrisches Polynom mit n Termen auch interpolatorisch durch

$$\sum_{j=1}^{n} v_j L_j^{(n)}(s)$$

dargestellt werden, wobei $L_j^{(n)}$ die trigonometrischen Lagrange-Polynome sind. Dabei sind die Koeffizienten v_j genau die Werte bei den Punkten $s = (j-1)/n$. Diese zwei Darstellung sind dabei mit einer Basistransformation verknüpft, welche abgesehen von Umformungen mit $\mathcal{O}(n)$ Aufwand durch die *schnelle Fourier Transformation* (FFT) respektive ihre Inverse gegeben ist. Deswegen können diese Transformationen also mit nur $\mathcal{O}(n \log n)$ Aufwand durchgeführt werden. Ebenso lässt sich damit eine effiziente Auswertung der trigonometrischen Polynome an k äquidistanten Punkten durchführen.

Aufgabe 1.

Damit die Radiusfunktion nun jeweils durch die FFT ausgewertet werden können, ändern wir, wie wir die Beschreibungen der Parametrisierungen $\boldsymbol{\gamma}_\ell$ der Randkomponenten Γ_l in dem `structure array mbp` für dieses Blatt abspeichern. Neu sollen dabei die Felder `gamma`, `dgamma`, `d2gamma` und `ngamma` von `mbp` wie folgt vektorwertig auswertbar sein, wobei $s_i := (i-1)/n$:

$$\begin{aligned}
\texttt{mbp(}\ell\texttt{).gamma(}n\texttt{)} &= \begin{bmatrix} \boldsymbol{\gamma}_\ell(s_1) & \cdots & \boldsymbol{\gamma}_\ell(s_n) \end{bmatrix} \\
\texttt{mbp(}\ell\texttt{).dgamma(}n\texttt{)} &= \begin{bmatrix} \boldsymbol{\gamma}'_\ell(s_1) & \cdots & \boldsymbol{\gamma}'_\ell(s_n) \end{bmatrix} \\
\texttt{mbp(}\ell\texttt{).d2gamma(}n\texttt{)} &= \begin{bmatrix} \boldsymbol{\gamma}''_\ell(s_1) & \cdots & \boldsymbol{\gamma}''_\ell(s_n) \end{bmatrix} \\
\texttt{mbp(}\ell\texttt{).ngamma(}n\texttt{)} &= \begin{bmatrix} \mathbf{n}_{\boldsymbol{\gamma}_\ell(s_1)} & \cdots & \mathbf{n}_{\boldsymbol{\gamma}_\ell(s_n)} \end{bmatrix}
\end{aligned}$$

Ändern Sie Ihre Funktion von Blatt 3

```
function mbp = add_star_shaped_parametrisation(mbp, xc, no, r, dr, d2r),
```

welches dem `structure array mbp` die sternförmigen Parametrisierung

$$s \mapsto \texttt{xc} + r(s) \begin{bmatrix} \cos(2\pi s) \\ \sin(2\pi s) \end{bmatrix}$$

anhängt, so dass sie der neuen Darstellung folgt. Dazu werden nun die Funktion r und ihre ersten zwei Ableitungen als `function handles` `r`, `dr` und `d2r` übergeben, welche wie folgt vektorwertig auswertbar sein sollen:

$$\begin{aligned} \texttt{r}(n) &= \begin{bmatrix} r(s_1) & \cdots & r(s_n) \end{bmatrix} \\ \texttt{dr}(n) &= \begin{bmatrix} r'(s_1) & \cdots & r'(s_n) \end{bmatrix} \\ \texttt{d2r}(n) &= \begin{bmatrix} r''(s_1) & \cdots & r''(s_n) \end{bmatrix}. \end{aligned}$$

Erstellen Sie weiter eine analoge Funktion

```
function mbp = ...
          set_star_shaped_parametrisation(mbp, l, xc, no, r, dr, d2r),
```

welche aber die `l`-te Parametrisierung von `mbp` ersetzt.

Aufgabe 2.

Erstellen Sie eine Funktion

`function dfs = fourier_differentiate_coefficients(fs)`,

welche das durch die Einträge in `fs` definierte, trigonometrische Polynom ableitet und dessen Koeffizienten als `dfs` zurückgibt.

Hinweis. *Die Reihenfolge der abgespeicherten Fourierkoeffizienten müssen Sie dabei der Funktion* `fourier_evaluate_on_uniform` *oder* `fourier_compute_coefficients`*, welche die oben beschriebene Transformation von Darstellungen berechnen, entnehmen.*

Aufgabe 3.

Erstellen Sie eine Funktion

`function [r, dr, d2r] = get_fourier_star_description(rs)`,

welches aus den n Radiusfunktionswerten bei $s = (j-1)/n$ die trigonometrisch polynominterpolierende Radiusfunktion und deren zwei ersten Ableitungen als `function handles` `r`, `dr` und `d2r` zurückgibt, welche wie folgt vektorwertig auswertbar sein sollen:

$$\begin{aligned} \texttt{r}(n) &= \begin{bmatrix} r(s_1) & \cdots & r(s_n) \end{bmatrix} \\ \texttt{dr}(n) &= \begin{bmatrix} r'(s_1) & \cdots & r'(s_n) \end{bmatrix} \\ \texttt{d2r}(n) &= \begin{bmatrix} r''(s_1) & \cdots & r''(s_n) \end{bmatrix} \end{aligned}$$

Hinweis. *Benutzen Sie dazu Ihre Funktion* `fourier_differentiate_coefficients` *sowie* `fourier_evaluate_on_uniform` *und* `fourier_compute_coefficients`*.*

Aufgabe 4.

Ändern Sie Ihre Funktion von Blatt 3

`function dcd = nystroem_cauchy(mbp, ns, g, h, btf)`,

so dass sie mit der neue Darstellung von dem `structure array` `mbp` funktioniert. Ändern Sie ebenso, dass der Rückgabewert der Funktion, das `structure array` `dcd`, neu auch das Feld `dgamma` mit den ausgewerteten $\gamma'_\ell(s_i)$ enthält.

Aufgabe 5.

Schreiben Sie eine Funktion

```
function [mbp, dcd, rsfs] = ...
                        bernoulli(mbp, rsf, g, h, ns, tau, tol, maxit),
```

welches das freie Randproblem approximativ wie oben beschrieben löst. Dabei werden das Gebiet als das `structure array mbp` und die Funktionen g und h als `function handles g` und `f` übergeben. Weiter sind `ns` die Anzahl Randpunkte für jeden Rand für das Nyström-Verfahren, `tau` der Dämpfungsfaktor, `tol` eine Abbruchstoleranz, wobei nach dem k-ten Schritt abgebrochen wird, falls $\max_{j=1}^{N}\big(|d_k(s_j)|\big) <$ `tol` ist, und `maxit` die maximale Anzahl an Iterationsschritten. In dem Zeilenvektor `rsf` sollen die N Radiusfunktionsauswertungen der 0-ten Iteration der Parametrisierung des Randes Γ_1 stehen.

In dem zurückgegebenen `structure array mbp` soll das erreichte Lösungsgebiet und in dem `structure array dcd` die dazugehörige, approximative Lösung von (1) stehen. In der Matrix `rsfs` geben Sie die N Radiusfunktionsauswertungen für jede Iteration der Parametrisierung des Randes Γ_1 zeilenweise zurück.

Definieren Sie Ihr eigenes freie Randproblem zu Testzwecken. Dafür müssen Sie beachten, dass überall $g > 0$ und $h < 0$ sind und dass der äussere Rand wirklich aussen liegt. Sie finden dazu auch in der Beilage drei Testprobleme.