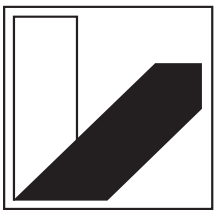


UNIVERSITÄT
BAYREUTH

Finite Differenzen und die 2-dim. Wärmeleitungsgleichung

Bachelorarbeit

von

Arthur Fleig

FAKULTÄT FÜR MATHEMATIK, PHYSIK UND INFORMATIK
MATHEMATISCHES INSTITUT

Datum: 30. September 2011

Betreuung:
Prof. Dr. L. Grüne
Dipl.-Math. T. Jahn
Dipl.-Wirtschaftsmath. K. Worthmann

In Erinnerung an meine Oma († 16. Februar 2011)

Danksagung

Zunächst möchte ich mich gerne bei den Lehrstühlen “Angewandte Mathematik” und “Ingenieurmathematik” bedanken, die mir stets mit ihrer fachlichen Kompetenz beratend zur Seite standen. Besonders dankbar bin ich Herrn Prof. Dr. Grüne und Thomas Jahn, da sie immer ein offenes Ohr für meine Fragen hatten. Sehr bemerkenswert finde ich Herrn Jahns Engagement für die Studierenden auch außerhalb der üblichen Bürozeiten.

Weiterhin bedanke ich mich bei meiner Familie und meiner Verlobten Eleni für die Unterstützung bei allem, was ich tue. Ohne deren Fürsorge und Liebe wäre das Studierendenleben bedeutend schwieriger und ich bin sehr froh, dass ich mich jederzeit auf sie verlassen kann.

Schließlich bin ich meinen Freunden an der Universität Bayreuth dankbar, durch die die letzten drei Jahre wesentlich angenehmer waren.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Arbeit beschäftigt sich mit der numerischen Lösung partieller Differentialgleichungen (PDGL). Dafür gibt es mehrere Verfahren. Hier wird auf das "Finite Differenzen"-Verfahren im $\mathbb{R}^2$ eingegangen. Während diese Methode auf "einfachen" Gebieten – wie zum Beispiel einem Rechteck – leicht zu implementieren ist, wird es bei "schwierigeren" Gebieten zunehmend unangenehmer. Hier werden vorwiegend die "unangenehmen" Gebiete untersucht. Praktisches Kernstück dieser Arbeit ist ein "Gittergenerator", der zunächst ein Gitter auf einem vorgegebenen Gebiet in $\mathbb{R}^2$ erzeugt, auf dem die partielle DGL gelöst werden soll und anschließend die nötigen Differenzenapproximationen berechnet.

1.1 Problemstellung

PDGLn werden oft verwendet, um physikalische Vorgänge mathematisch zu modellieren. So wird ein Diffusionsprozess einer Substanz in einem Medium, etwa die Wärmeausbreitung in einem Wärmeleiter, durch die sogenannte Wärmeleitungsgleichung beschrieben.

Definition 1.1 *(Wärmeleitungsgleichung, Anfangs-/Randwert-Problem)*

(a) Die d-dimensionale Wärmeleitungsgleichung *lautet*

$$\partial_t u(t,x) - \Delta u(t,x) = f(t,x) \tag{1.1}$$

mit $\partial_t u := \frac{\partial u}{\partial t}, \Delta u := \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2}$ *wobei* $u : [0,\infty[\times\mathbb{R}^d \to \mathbb{R}$ *gesucht und* $f(t,x)$ *vorgegeben.*

(b) Sei $\Omega \subset \mathbb{R}^d$ *ein beschränktes Gebiet mit Rand* $\Gamma = \Gamma_D \dot\cup \Gamma_N$ *und* $T > 0$. Γ_N *sei so, dass die vom Gebiet* Ω *ins Äußere zeigende Normalenrichtung* ν *erklärt werden kann. Ein* Anfangs-/Randwert-Problem (ARWP) einer Wärmeleitungsgleichung *ist definiert durch*

$$\partial_t u(t,x) - \Delta u(t,x) = f(t,x), (t,x) \in]0,T[\times\Omega \tag{1.2}$$

$$u(t,x) = g_D(t,x), (t,x) \in]0,T] \times \Gamma_D \quad \textit{(Dirichlet-Randbedingung)} \tag{1.3}$$

$$\frac{\partial u}{\partial \nu}(t,x) = g_N(t,x), (t,x) \in]0,T] \times \Gamma_N \quad \textit{(Neumann-Randbedingung)} \tag{1.4}$$

$$u_{|t=0} = h(x) \quad \textit{(Anfangsdaten)} \tag{1.5}$$

wobei $f :]0,T[\times\Omega \to \mathbb{R}, g_D :]0,T] \times \Gamma_D \to \mathbb{R}, g_N :]0,T] \times \Gamma_N \to \mathbb{R}, h : \bar{\Omega} \to \mathbb{R}$ *vorgegeben sind und* $u : [0,T] \times \bar{\Omega} \to \mathbb{R}$ *gesucht ist.*

Bemerkung 1.2
Die Wärmeleitungsgleichung spielt in dieser Arbeit eine vorrangige Rolle. Doch es werden auch PDGLn der Form

$$c_1\partial_t u + c_2\partial_{x_1} u + c_3\partial_{x_2} u + c_4\partial_{x_1x_1} u + c_5\partial_{x_2x_2} u + c_6 u = f \tag{1.6}$$

mit $c_i \in \mathbb{R}, i \in \{1,\dots,6\}$ *behandelt. Zur besseren Übersicht wurden die Argumente* (t,x) *diesmal ausgelassen. Dabei ist* $u(t,x)$ *gesucht und* $f(t,x)$ *vorgegeben,* $t \in [0,\infty[$ *und* $x := (x_1,x_2) \in \mathbb{R}^2$. *Ein zugehöriges ARWP erhält man, in dem man die Definition 1.1(b) im Fall* $d = 2$ *betrachtet und anstelle der Gleichung in (1.2) (1.6) einsetzt. Beachte, dass die 2-dimensionale Wärmeleitungsgleichung ein Spezialfall von (1.6) ist.*

Definition und Bemerkung 1.3
$\frac{\partial u}{\partial \nu}$ *aus Definition 1.1(b) bezeichnet die Richtungsableitung von* u *in Richtung des nach außen gerichteten Normalenvektors* ν *von* Γ_N *und heißt* Normalenableitung. *Diese existiert, wenn der Rand* Γ_N *glatt ist.*

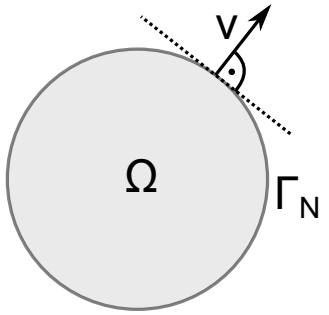


Abbildung 1.1: Normalenvektor

Definition und Bemerkung 1.4
Neben der Dirichlet-Randbedingung (1.3) und der Neumann-Randbedingung (1.4) gibt es für ein vorgegebenes $\alpha \in \mathbb{R}$ *die* Cauchy-Randbedingung

$$\frac{\partial u}{\partial \nu}(t,x) + \alpha u(t,x) = g_C(t,x),$$

eine Kombination von Dirichlet- und Neumann-Randbedingung. Cauchy-Randbedingungen werden hier nicht berücksichtigt, können jedoch im Prinzip analog behandelt werden.

Für festes t ist die Lösung eine Funktion $u : \bar{\Omega} \to \mathbb{R}$, das heißt die Lösung ist ein Element eines ∞-dimensionalen Funktionenraums. Das Problem ist also ∞-dimensional (vgl. [1]). Um es mit numerischen Methoden auf dem Computer zu lösen, muss das Problem auf eine endlichdimensionale Approximation transformiert werden. Dafür gibt es mehrere Ansätze. Welchen Ansatz man wählt, hängt stark vom Ortsgebiet Ω ab. In dieser Arbeit wird die Methode der "Finiten Differenzen" im Fall $d = 2$ behandelt. Die theoretische Grundlage dafür, speziell für die Abschnitte 2.1 und 2.2, bildete das Vorlesungsskript [2].

Kapitel 2

Finite Differenzen (2D)

Die Grundidee des "Finiten Differenzen"-Verfahrens ist wie folgt: Zunächst wird das Gebiet Ω aus Definition 1.1(b) diskretisiert. In den Gitterpunkten werden dann die Ortsableitungen durch sogenannte Differenzenapproximationen ersetzt. Daraus erhält man ein großes System gewöhnlicher Differentialgleichungen, das die partielle Differentialgleichung annähernd wiedergibt und mit bekannten Algorithmen gelöst werden kann. Da nur das Ortsgebiet diskretisiert wird und die gewöhnlichen Differentialgleichungen weiterhin kontinuierlich in der Zeit sind, spricht man von einer "Semidiskretisierung".
Das Gebiet Ω im Spezialfall $d = 2$ wird dabei wie folgt diskretisiert:

Definition 2.1

(a) Sei $\Omega \subset \mathbb{R}^2$ wie in Def. 1.1(b). Dann ist die Menge der inneren Gitterpunkte *mit Schrittweiten h_x in x-Richtung und h_y in y-Richtung gegeben durch*

$$G := \Omega \cap \{(x, y) \in \mathbb{R}^2 \mid x = z_1 h_x, y = z_2 h_y, z_1, z_2 \in \mathbb{Z}\} \tag{2.1}$$

(b) Analog ist mit

$$\partial G := \partial\Omega \cap \{(x, y) \in \mathbb{R}^2 \mid (x = z_1 h_x \vee y = z_2 h_y), z_1, z_2 \in \mathbb{Z}\} \tag{2.2}$$

die Menge der Randpunkte des Gitters G *aus (a) gegeben. Das* Gitter *lässt sich insgesamt darstellen als*

$$\bar{G} := G \cup \partial G \tag{2.3}$$

2.1 Differenzenapproximationen für innere Punkte

Ausgehend von einem inneren Punkt $p_c := (x, y) \in G$ (c für "center") eines Gitters G aus Definition 2.1 erhält man allgemein die folgenden vier Nachbarpunkte:

$$\begin{aligned} p_e &:= (x + h_e, y) \\ p_w &:= (x - h_w, y) \\ p_n &:= (x, y + h_n) \\ p_s &:= (x, y - h_s) \end{aligned} \tag{2.4}$$

Dabei legen $h_e, h_w \in]0, h_x]$ und $h_n, h_s \in]0, h_y]$ die "Richtung" ("east", "west", "north" und "south") fest.

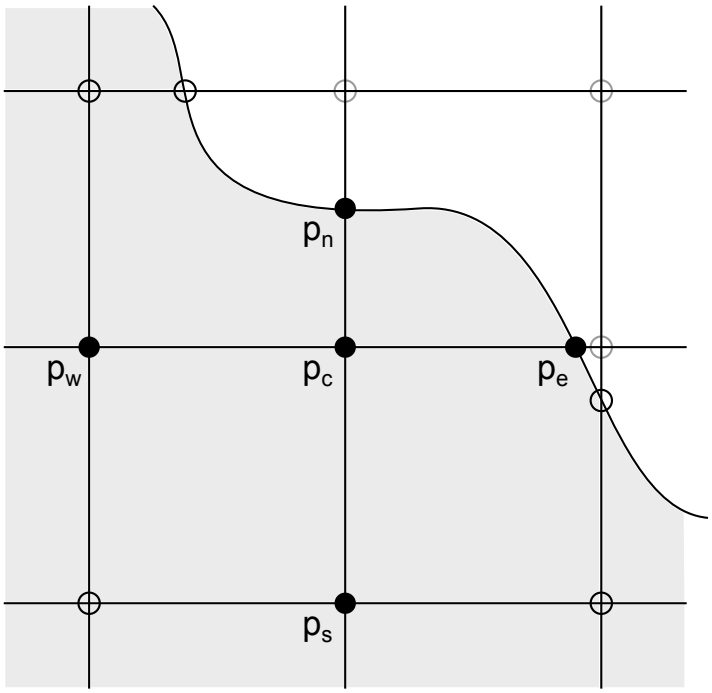


Abbildung 2.1: Gitterpunkt mit vier Nachbarpunkten

Für festes $t \in \mathbb{R}$ bezeichne u_c, u_e, u_w, u_n und u_s die Lösung u der PDGL in den entsprechenden Gitterpunkten, also zum Beispiel $u_c := u(t, p_c) = u(p_c)$. Für ein hinreichend oft differenzierbares u, das in diesem Kapitel generell vorausgesetzt wird, lässt sich nun die Lösung in den Gitterpunkten mit Hilfe einer Taylorentwicklung der Ordnung 2 von u um p_c in p_e approximieren:

$$\begin{aligned} u_e &= u_c + \partial_x u(p_c)(p_e - p_c)_1 + \frac{1}{2}\partial_x^2(p_c)(p_e - p_c)_1^2 + \mathcal{O}((p_e - p_c)_1^3) \\ &= u_c + \partial_x u(p_c)h_e + \frac{1}{2}\partial_x^2(p_c)h_e^2 + \mathcal{O}(h_e^3) \end{aligned} \tag{2.5}$$

Man beachte, dass jeweils entweder x oder y fest ist und daher die eindimensionale Taylor-Entwicklung ausreicht. Entsprechend bezeichnet $(p_e - p_c)_1$ die erste Komponente des Vektors $p_e - p_c$. Analog gilt für die anderen Nachbarpunkte:

$$u_w = u_c - \partial_x u(p_c) h_w + \frac{1}{2}\partial_x^2(p_c) h_w^2 + \mathcal{O}(h_w^3) \tag{2.6}$$

$$u_n = u_c + \partial_y u(p_c) h_n + \frac{1}{2}\partial_y^2(p_c) h_n^2 + \mathcal{O}(h_n^3) \tag{2.7}$$

$$u_s = u_c - \partial_y u(p_c) h_s + \frac{1}{2}\partial_y^2(p_c) h_s^2 + \mathcal{O}(h_s^3) \tag{2.8}$$

Damit lassen sich nun verschiedene Ableitungen, zum Beispiel $\partial_x^2 u$ und $\partial_y^2 u$, im Gitterpunkt p_c approximieren. Diese sollen sich als Linearkombination von u_e, u_w, u_n, u_s und u_c bilden lassen, d.h.

$$\alpha_e u_e + \alpha_w u_w + \alpha_n u_n + \alpha_s u_s + \alpha_c u_c \tag{2.9}$$

wobei $\alpha_i \in \mathbb{R}, i \in \{e, w, n, s, c\}$.

Einsetzen von (2.5)-(2.8) in (2.9) liefert unter Vernachlässigung der Restterme:

$$\begin{aligned}\alpha_e u_e + \alpha_w u_w + \alpha_n u_n + \alpha_s u_s + \alpha_c u_c = {} & (\alpha_e + \alpha_w + \alpha_n + \alpha_s + \alpha_c) u_c \\ & + (\alpha_e h_e - \alpha_w h_w)\partial_x u(p_c) \\ & + (\alpha_n h_n - \alpha_s h_s)\partial_y u(p_c) \\ & + \frac{1}{2}(\alpha_e h_e^2 + \alpha_w h_w^2)\partial_x^2 u(p_c) \\ & + \frac{1}{2}(\alpha_n h_n^2 + \alpha_s h_s^2)\partial_y^2 u(p_c)\end{aligned}$$

Nun lassen sich Bedingungen an die Koeffizienten α_i herleiten, so dass die jeweils gewünschte Ableitung im Punkt p_c approximiert wird. Exemplarisch wird das in Beispiel 2.2 gezeigt. Doch zunächst eine abkürzende Schreibweise:

Notation
Zur Abkürzung werden in diesem Abschnitt folgende Bezeichnungen verwendet:

$$\alpha := (\alpha_e, \alpha_w, \alpha_n, \alpha_s, \alpha_c)^T, v := (u_e, u_w, u_n, u_s, u_c)^T$$

Beispiel 2.2

(a) Es soll $\partial_x^2 u(p_c)$ approximiert werden, d.h. folgende Bedingungsgleichungen sind zu erfüllen:

$$\begin{aligned}\alpha_e + \alpha_w + \alpha_n + \alpha_s + \alpha_c &= 0 \\ (\alpha_e h_e - \alpha_w h_w) &= 0 \\ (\alpha_n h_n - \alpha_s h_s) &= 0 \\ \frac{1}{2}(\alpha_e h_e^2 + \alpha_w h_w^2) &= 1 \\ \frac{1}{2}(\alpha_n h_n^2 + \alpha_s h_s^2) &= 0\end{aligned}$$

In Matrixform umgeschrieben ist das

$$A\alpha = b$$

mit $b := (0,0,0,1,0)^T$ *und*

$$A := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ h_e & -h_w & 0 & 0 & 0 \\ 0 & 0 & h_n & -h_s & 0 \\ \frac{1}{2}h_e^2 & \frac{1}{2}h_w^2 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2}h_n^2 & \frac{1}{2}h_s^2 & 0 \end{pmatrix} \tag{2.10}$$

Als eindeutige Lösung des LGS erhält man

$$\alpha = \left(\frac{2}{(h_e + h_w)h_e}, \frac{2}{(h_e + h_w)h_w}, 0, 0, -\frac{2}{h_e h_w} \right)^T,$$

also mit $h := \max\{h_e, h_w\}$ *insgesamt*

$$\partial_x^2 u(p_c) = \alpha^T \cdot v + \mathcal{O}(h^2) \approx \alpha^T \cdot v$$

(b) Analog erhält man für $\Delta u(p_c) := \partial_x^2 u(p_c) + \partial_y^2 u(p_c)$ *mit A aus (a) und* $b := (0,0,0,1,1)^T$

$$\alpha = \left(\frac{2}{(h_e + h_w)h_e}, \frac{2}{(h_e + h_w)h_w}, \frac{2}{(h_n + h_s)h_n}, \frac{2}{(h_n + h_s)h_s}, -\frac{2(h_e h_w + h_n h_s)}{h_e h_w h_n h_s} \right)^T$$

(c) Für $\partial_x u(p_c)$ *erhält man im Falle* $h_w = h_e = h_x$ *gerade den zentralen Differenzenquotienten*

$$\partial_x u(p_c) = \frac{u(x + h_x, y) - u(x - h_x, y)}{2h_x} + \mathcal{O}(h_x^2) \approx \frac{u(x + h_x, y) - u(x - h_x, y)}{2h_x}$$

Bemerkung 2.3

(a) Beachte, dass die Matrix (2.10) wegen $h_e, h_w \in]0, h_x]$ *und* $h_n, h_s \in]0, h_y]$ *regulär ist.*

(b) Gemischte Ableitungen zweiter Ordnung lassen sich mit diesem Ansatz nicht approximieren, da sie nicht in den Taylor-Approximationen auftauchen. Damit gemischte Ableitungen in der Taylorentwicklung auftreten, wird mindestens ein weiterer Punkt benötigt, der nicht ausschließlich in x- oder y-Richtung liegt. Mit diesem weiteren Punkt lässt sich analog ein LGS aufstellen, um die nötigen 6 Koeffizienten zu berechnen.

(c) Die quadratische Ordnung des Restterms in Beispiel 2.2(a) erhält man, indem man die Taylor-Entwicklung bis zur Ordnung 3 erweitert und ausnutzt, dass sich die Terme der Ordnung 3 durch unterschiedliche Vorzeichen wegheben.

2.2 Differenzenapproximationen für Randpunkte

In Definition 1.1(b) wurden die Dirichlet- und die Neumann-Randbedingung eingeführt.

Bemerkung 2.4
Dirichlet-Randpunkte sind leicht behandelbar, da dort die Lösung u vorgegeben ist. Damit ist in diesem Fall nichts weiter zu tun, insbesondere ist keine Differenzenapproximation nötig.

Die Neumann-Randpunkte $p \in \partial G \cap \Gamma_N$ sind sowohl in der Theorie als auch in der Implementierung wesentlich schwieriger zu handhaben als innere Gitterpunkte. Prinzipiell muss man die Normalenableitung aus der Neumann-Bedingung (1.4) ebenfalls durch Differenzenapproximationen ersetzen. Die Vorgehensweise hier gleicht dabei dem Vorgehen im Vorlesungsskript [2], an dem sich diese Arbeit hauptsächlich orientiert: Die DGL (1.2) wird nicht nur auf den inneren Gitterpunkten diskretisiert, sondern auch auf den Neumann-Randpunkten. Die Idee der Fortsetzung wird auch in [3, S. 26ff] und [4, S. 50ff] verfolgt, um die Normalenableitung auf einfachen Gebieten wie dem Einheitsquadrat mit einer höheren Ordnung zu approximieren, nämlich $\mathcal{O}(h^2)$, wenn h die Schrittweite in x- und y-Richtung ist. Man kann hoffen, dass diese Fortsetzung auch auf irregulären Gittern zu guten Ergebnissen führt.
Ziel ist es also, analog zum Abschnitt 2.1 Differenzenapproximationen für die partiellen Ableitungen in (1.2) in Neumann-Randpunkten zu erhalten. Da die Nachbarpunkte eines Randpunktes jedoch nicht zwangsweise nur in x- oder y-Richtung liegen, treten bei der Taylor-Entwicklung im Allgemeinen gemischte Ableitungen auf. Damit werden nun sechs statt nur fünf Größen benötigt, um eine eindeutige Linearkombination zur Approximation der Ableitungen zu erhalten. Zu den vier Nachbarpunkten wird die Neumann-Randbedingung (1.4) hinzugenommen. Die Normalenableitung lässt sich schreiben als

$$\frac{\partial u}{\partial \nu}(x,y) = \partial_x u(x,y)\cos(\phi) + \partial_y u(x,y)\sin(\phi). \tag{2.11}$$

In Abbildung 2.2 ist der am häufigsten auftretende Fall dargestellt: Ein Neumann-Randpunkt p_c liegt auf nur einer Gitterlinie und es lassen sich vier Nachbarpunkte in unmittelbarer Umgebung finden.

Bemerkung 2.5
In Definition 1.1(b) wurde gefordert, dass der vom Gebiet Ω ins Äußere zeigende Normalenvektor ν erklärt werden kann. In den Randpunkten, in denen keine eindeutige Normale existiert, ist die Neumann-Randbedingung (1.4) nicht wohldefiniert und damit das Problem nicht gut gestellt. Natürlich kann in diesem Fall auch keine Approximation hergeleitet werden. Ein solcher Fall sind die Punkte p_c und p_e in Abbildung 2.3.

Ausgehend von einem Neumann-Randpunkt $p_c := (x,y) \in \partial G \cap \Gamma_N$ erhält man nun die folgenden vier Nachbarpunkte:

$$\begin{aligned} p_e &:= (x_e, y_e) = (x - h_{ex}, y - h_{ey}) \\ p_w &:= (x_w, y_w) = (x - h_{wx}, y - h_{wy}) \\ p_n &:= (x_n, y_n) = (x - h_{nx}, y - h_{ny}) \\ p_s &:= (x_s, y_s) = (x - h_{sx}, y - h_{sy}) \end{aligned} \tag{2.12}$$

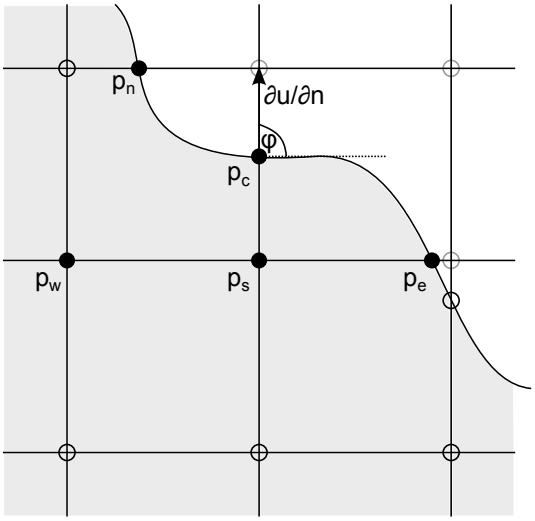


Abbildung 2.2: Randpunkt mit vier Nachbarpunkten

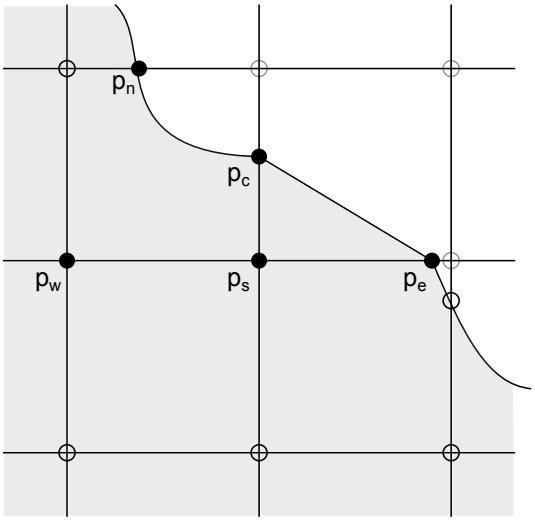


Abbildung 2.3: Normale im Randpunkt nicht eindeutig

Diesmal sind mit $h_{ex}, h_{wx}, h_{nx}, h_{sx} \in [-2h_x, 2h_x]$ und $h_{ey}, h_{wy}, h_{ny}, h_{sy} \in [-2h_y, 2h_y]$ die "Richtungen" nicht a priori bekannt. Zu beachten ist, dass keine Punkte zusammenfallen dürfen, d.h. gefordert ist $\|p_i - p_j\|_2 > 0$ für $i, j \in \{c, e, w, n, s\}$ mit $i \neq j$.

Bemerkung 2.6

(a) Die Einschränkung auf $[-h_x, h_x]$ bzw. $[-h_y, h_y]$ ist im Allgemeinen nicht möglich, vgl. Abbildung 2.4. In diesem Fall ist der vierte Nachbarpunkt zwangsweise weiter entfernt und wäre unter der Voraussetzung $h_{ey}, h_{wy}, h_{ny}, h_{sy} \in [-h_y, h_y]$ nicht erreichbar.

(b) Abbildung 2.4 zeigt einen weiteren bedeutenden Unterschied zu inneren Punkten. Während für innere Gitterpunkte schnell "kanonische" Nachbarpunkte gefunden werden konnten,

können und müssen die Nachbarpunkte von Neumann-Randpunkten gewählt werden.

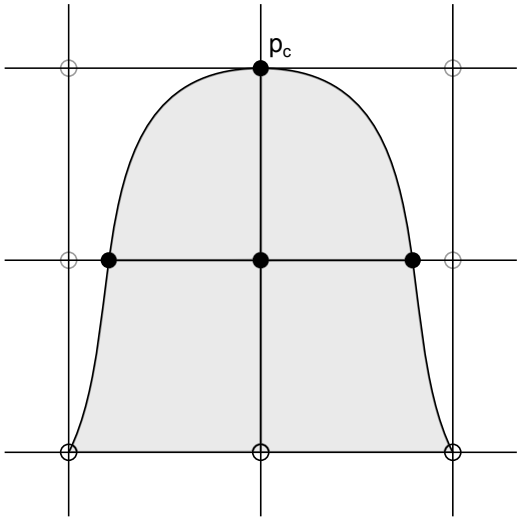


Abbildung 2.4: Weiter entfernte Nachbarpunkte

Dieses Mal wird die mehrdimensionale Taylor-Entwicklung bemüht, um die gewünschte Ableitung in p_c zu approximieren. Mit der aus Abschnitt 2.1 bekannten Schreibweise gilt unter Vernachlässigung der Restterme:

$$u_e \approx u_c - \partial_x u(p_c)h_{ex} - \partial_y u(p_c)h_{ey} + \frac{1}{2}\partial_x^2(p_c)h_{ex}^2 + \frac{1}{2}\partial_y^2(p_c)h_{ey}^2 + \partial_x\partial_y u(p_c)h_{ex}h_{ey} \tag{2.13}$$

$$u_w \approx u_c - \partial_x u(p_c)h_{wx} - \partial_y u(p_c)h_{wy} + \frac{1}{2}\partial_x^2(p_c)h_{wx}^2 + \frac{1}{2}\partial_y^2(p_c)h_{wy}^2 + \partial_x\partial_y u(p_c)h_{wx}h_{wy} \tag{2.14}$$

$$u_n \approx u_c - \partial_x u(p_c)h_{nx} - \partial_y u(p_c)h_{ny} + \frac{1}{2}\partial_x^2(p_c)h_{nx}^2 + \frac{1}{2}\partial_y^2(p_c)h_{ny}^2 + \partial_x\partial_y u(p_c)h_{nx}h_{ny} \tag{2.15}$$

$$u_s \approx u_c - \partial_x u(p_c)h_{sx} - \partial_y u(p_c)h_{sy} + \frac{1}{2}\partial_x^2(p_c)h_{sx}^2 + \frac{1}{2}\partial_y^2(p_c)h_{sy}^2 + \partial_x\partial_y u(p_c)h_{sx}h_{sy} \tag{2.16}$$

Zusammen mit (2.11) soll nun die folgende Linearkombination für die Näherung verwendet werden:

$$\beta_e u_e + \beta_w u_w + \beta_n u_n + \beta_s u_s + \beta_c u_c + \beta_d \frac{\partial u}{\partial \nu}(p_c) \tag{2.17}$$

wobei $\beta_i \in \mathbb{R}, i \in \{e, w, n, s, c, d\}$
Wie im vorherigen Abschnitt lassen sich Bedingungsgleichungen für verschiedene Ableitungen herleiten und die Koeffizienten β_i durch das Lösen eines LGS erhalten. Analog zur Matrix (2.10) leitet man die Matrix

$$B := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ -h_{ex} & -h_{wx} & -h_{nx} & -h_{sx} & 0 & \cos(\phi) \\ -h_{ey} & -h_{wy} & -h_{ny} & -h_{sy} & 0 & \sin(\phi) \\ \frac{1}{2}h_{ex}^2 & \frac{1}{2}h_{wx}^2 & \frac{1}{2}h_{nx}^2 & \frac{1}{2}h_{sx}^2 & 0 & 0 \\ \frac{1}{2}h_{ey}^2 & \frac{1}{2}h_{wy}^2 & \frac{1}{2}h_{ny}^2 & \frac{1}{2}h_{sy}^2 & 0 & 0 \\ h_{ex}h_{ey} & h_{wx}h_{wy} & h_{nx}h_{ny} & h_{sx}h_{sy} & 0 & 0 \end{pmatrix} \tag{2.18}$$

her.

Bemerkung 2.7
Die Matrix (2.18) kann singulär werden. Für eine eindeutige Lösung muss die Regularität sichergestellt werden. Die Regularität hängt von der Auswahl der Nachbarpunkte ab. In Abbildung 2.5 ist eine Auswahl dargestellt, die zur Singularität von (2.18) führt.

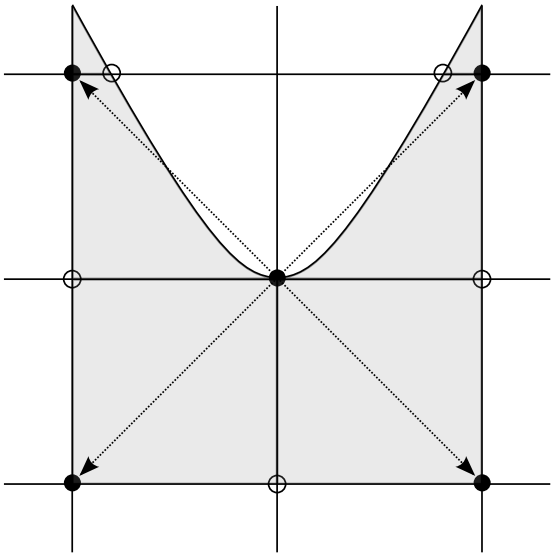


Abbildung 2.5: Eine schlechte Wahl der Nachbarpunkte

Es ist also eine geschickte Auswahl der Nachbarpunkte für Neumann-Randpunkte nötig, die

- die Regularität der Matrix (2.18) sicherstellt und
- den Fehler, den man durch das Vernachlässigen der Restterme in den Taylorentwicklungen macht, möglichst klein hält. Dazu sind nahegelegene Nachbarpunkte wünschenswert.

Details zu verschiedenen Auswahlverfahren folgen in Abschnitt 3.2.3. An dieser Stelle wird angenommen, dass zumindest die Matrix (2.18) regulär ist. Zum Abschluss dieses Abschnitts soll nun $\Delta u(p_c)$ approximiert werden.

Notation
Zur Abkürzung werden in diesem Abschnitt folgende Bezeichnungen verwendet:

$$\beta := (\beta_e, \beta_w, \beta_n, \beta_s, \beta_c, \beta_d)^T, w := (u_e, u_w, u_n, u_s, u_c, \frac{\partial u}{\partial \nu}(p_c))^T$$

Beispiel 2.8
Es soll $\Delta u(p_c)$ approximiert werden. In Matrixform geschrieben ist das $B\beta = c$ mit B aus (2.18) und $c := (0, 0, 0, 1, 1, 0)^T$. Sofern B regulär ist, erhält man die Lösung[1] *mit*

$$N := 2 \det(B),$$

[1]Das Gleichungssystem wurde mit der Computersoftware "Maple" gelöst. Ein entsprechendes "Maple-Worksheet" findet sich auf der beigelegten CD-ROM.

$$\begin{aligned}
\beta_e = -\frac{1}{N}\Big(& (h_{wx}\sin(\phi) - h_{wy}\cos(\phi))\left[h_{nx}h_{ny}\left(h_{sx}^2 - h_{sy}^2\right) - h_{sx}h_{sy}\left(h_{nx}^2 - h_{ny}^2\right)\right] \\
& + (h_{nx}\sin(\phi) - h_{ny}\cos(\phi))\left[h_{sx}h_{sy}\left(h_{wx}^2 - h_{wy}^2\right) - h_{wx}h_{wy}\left(h_{sx}^2 - h_{sy}^2\right)\right] \\
& + (h_{sx}\sin(\phi) - h_{sy}\cos(\phi))\left[h_{wx}h_{wy}\left(h_{nx}^2 - h_{ny}^2\right) - h_{nx}h_{ny}\left(h_{wx}^2 - h_{wy}^2\right)\right]\Big) \\
\beta_w = \frac{1}{N}\Big(& (h_{ex}\sin(\phi) - h_{ey}\cos(\phi))\left[h_{nx}h_{ny}\left(h_{sx}^2 - h_{sy}^2\right) - h_{sx}h_{sy}\left(h_{nx}^2 - h_{ny}^2\right)\right] \\
& + (h_{nx}\sin(\phi) - h_{ny}\cos(\phi))\left[h_{sx}h_{sy}\left(h_{ex}^2 - h_{ey}^2\right) - h_{ex}h_{ey}\left(h_{sx}^2 - h_{sy}^2\right)\right] \\
& + (h_{sx}\sin(\phi) - h_{sy}\cos(\phi))\left[h_{ex}h_{ey}\left(h_{nx}^2 - h_{ny}^2\right) - h_{nx}h_{ny}\left(h_{ex}^2 - h_{ey}^2\right)\right]\Big) \\
\beta_n = -\frac{1}{N}\Big(& (h_{sx}\sin(\phi) - h_{sy}\cos(\phi))\left[h_{ex}h_{ey}\left(h_{wx}^2 - h_{wy}^2\right) - h_{wx}h_{wy}\left(h_{ex}^2 - h_{ey}^2\right)\right] \\
& + (h_{ex}\sin(\phi) - h_{ey}\cos(\phi))\left[h_{wx}h_{wy}\left(h_{sx}^2 - h_{sy}^2\right) - h_{sx}h_{sy}\left(h_{wx}^2 - h_{wy}^2\right)\right] \\
& + (h_{wx}\sin(\phi) - h_{wy}\cos(\phi))\left[h_{sx}h_{sy}\left(h_{ex}^2 - h_{ey}^2\right) - h_{ex}h_{ey}\left(h_{sx}^2 - h_{sy}^2\right)\right]\Big) \\
\beta_s = \frac{1}{N}\Big(& (h_{nx}\sin(\phi) - h_{ny}\cos(\phi))\left[h_{ex}h_{ey}\left(h_{wx}^2 - h_{wy}^2\right) - h_{wx}h_{wy}\left(h_{ex}^2 - h_{ey}^2\right)\right] \\
& + (h_{ex}\sin(\phi) - h_{ey}\cos(\phi))\left[h_{wx}h_{wy}\left(h_{nx}^2 - h_{ny}^2\right) - h_{nx}h_{ny}\left(h_{wx}^2 - h_{wy}^2\right)\right] \\
& + (h_{wx}\sin(\phi) - h_{wy}\cos(\phi))\left[h_{nx}h_{ny}\left(h_{ex}^2 - h_{ey}^2\right) - h_{ex}h_{ey}\left(h_{nx}^2 - h_{ny}^2\right)\right]\Big) \\
\beta_c = -\frac{1}{N}\Big(& \left[h_{ex}h_{ey}(h_{sy}^2 - h_{sx}^2) - h_{sx}h_{sy}(h_{ey}^2 - h_{ex}^2)\right]\left[(h_{nx} - h_{wx})\sin(\phi) - (h_{ny} - h_{wy})\cos(\phi)\right] \\
& + \left[h_{ex}h_{ey}(h_{ny}^2 - h_{nx}^2) - h_{nx}h_{ny}(h_{ey}^2 - h_{ex}^2)\right]\left[(h_{wx} - h_{sx})\sin(\phi) - (h_{wy} - h_{sy})\cos(\phi)\right] \\
& + \left[h_{wx}h_{wy}(h_{sy}^2 - h_{sx}^2) - h_{sx}h_{sy}(h_{wy}^2 - h_{wx}^2)\right]\left[(h_{ex} - h_{nx})\sin(\phi) - (h_{ey} - h_{ny})\cos(\phi)\right] \\
& + \left[h_{wx}h_{wy}(h_{ey}^2 - h_{ex}^2) - h_{ex}h_{ey}(h_{wy}^2 - h_{wx}^2)\right]\left[(h_{nx} - h_{sx})\sin(\phi) - (h_{ny} - h_{sy})\cos(\phi)\right] \\
& + \left[h_{wx}h_{wy}(h_{ny}^2 - h_{nx}^2) - h_{nx}h_{ny}(h_{wy}^2 - h_{wx}^2)\right]\left[(h_{sx} - h_{ex})\sin(\phi) - (h_{sy} - h_{ey})\cos(\phi)\right] \\
& + \left[h_{nx}h_{ny}(h_{sy}^2 - h_{sx}^2) - h_{sx}h_{sy}(h_{ny}^2 - h_{nx}^2)\right]\left[(h_{wx} - h_{ex}\sin(\phi) - (h_{wy} - h_{ey})\cos(\phi)\right]\Big) \\
\beta_d = -\frac{1}{N}\Big(& (h_{sy}^2 - h_{sx}^2)\big[h_{wx}h_{wy}(h_{ey}h_{nx} - h_{ex}h_{ny}) + h_{nx}h_{ny}(h_{ex}h_{wy} - h_{ey}h_{wx}) + h_{ex}h_{ey}(h_{wx}h_{ny} \\
& - h_{wy}h_{nx})\big] + (h_{ny}^2 - h_{nx}^2)\big[h_{wx}h_{wy}(h_{ex}h_{sy} - h_{ey}h_{sx}) + h_{sx}h_{sy}(h_{wx}h_{ey} - h_{wy}h_{ex}) \\
& + h_{ex}h_{ey}(h_{wy}h_{sx} - h_{wx}h_{sy})\big] + (h_{ey}^2 - h_{ex}^2)\big[h_{wx}h_{wy}(h_{sx}h_{ny} - h_{sy}h_{nx}) + h_{nx}h_{ny}(h_{sy}h_{wx} \\
& - h_{sx}h_{wy}) + h_{sx}h_{sy}(h_{nx}h_{wy} - h_{ny}h_{wx})\big] + (h_{wy}^2 - h_{wx}^2)\big[h_{sx}h_{sy}(h_{ny}h_{ex} - h_{nx}h_{ey}) \\
& + h_{ex}h_{ey}(h_{nx}h_{sy} - h_{ny}h_{sx}) + h_{nx}h_{ny}(h_{sx}h_{ey} - h_{sy}h_{ex})\big]\Big)
\end{aligned}$$

Im Spezialfall $h_{wy} = h_{ey} = h_{sy}, h_{sx} = 0, \phi = \frac{\pi}{2}$ *(vgl. Abbildung 2.2) ergibt sich*

$$\beta = \left(\frac{2}{(h_{ex} - h_{wx})h_{ex}}, -\frac{2}{(h_{ex} - h_{wx})h_{wx}}, 0, \frac{2(h_{ex}h_{wx} + h_{sy}^2)}{h_{ex}h_{wx}h_{sy}^2}, -\frac{2}{h_{sy}^2}, \frac{2}{h_{sy}}\right)^T,$$

also insgesamt

$$\Delta u(p_c) \approx \beta^T \cdot w$$

2.3 Das System gewöhnlicher Differentialgleichungen

In den Abschnitten 2.1 und 2.2 wurden jeweils die Koeffizienten für die Approximation in einem Gitterpunkt hergeleitet. Dies wird nun für alle Punkte gemacht, in denen die Lösung u des ARWPs aus Definition 1.1(b) bzw. Bemerkung 1.2 unbekannt ist, also für die inneren Gitterpunkte und die Neumann-Randpunkte.
Gegeben sei also ein Gitter $\bar{G}$ wie in Definition 2.1. Sei P_I die Menge der inneren Punkte, P_D die der Dirichlet-Randpunkte und P_N die der Neumann-Randpunkte. Seien weiter

$$N_I := |P_I|\,, N_N := |P_N|\,, N_D := |P_D|\,.$$

Die inneren Gitterpunkte $p_1, \ldots, p_{N_I} \in P_I$, die Neumann-Randpunkte $p_{N_I+1}, \ldots, p_{N_I+N_N} \in P_N$ und die Dirichlet-Randpunkte $p_{N_I+N_N+1}, \ldots, p_{N_I+N_N+N_D} \in P_D$ seien jeweils von links nach rechts und von unten nach oben nummeriert.[1] Jeder Gitterpunkt $p_i \in P_I \cup P_N$ hat die vier Nachbarpunkte p_i^e, p_i^w, p_i^n und p_i^s. u_i bezeichne die Lösung in einem Punkt p_i. Für $p_i \in P_I$ sei α^i der Koeffizientenvektor, der die gewünschten partiellen Ableitungen (bis zu zweiter Ordnung, allerdings keine gemischten Ableitungen) approximiert (vgl. Beispiel 2.2). Analog β^i für $p_i \in P_N$ (vgl. Beispiel 2.8).
Damit stehen alle benötigten Größen zur Verfügung, um eine PDGL mit einem $(N_I + N_N)$-dimensionalen System aus gewöhnlichen Differentialgleichungen zu approximieren. Im Falle von Bemerkung 1.2 lässt sich die partielle Differentialgleichung

$$c_1\partial_t u + c_2\partial_{x_1} u + c_3\partial_{x_2} u + c_4\partial_{x_1x_1} u + c_5\partial_{x_2x_2} u + c_6 u = f$$

mit ausgelassenen Argumenten (vgl. (1.6)) in den Gitterpunkten approximieren mit

$$c_1\partial_t\tilde{u} + A\tilde{u} = b \Leftrightarrow c_1\partial_t\tilde{u} = -A\tilde{u} + b, \tag{2.19}$$

wobei $\tilde{u} \in \mathbb{R}^{N_I+N_N}$ ein Vektor (sog. "Zustandsvektor") ist und die Ableitung nach der Zeit komponentenweise zu verstehen ist. Die Anfangsdaten erhält man mit

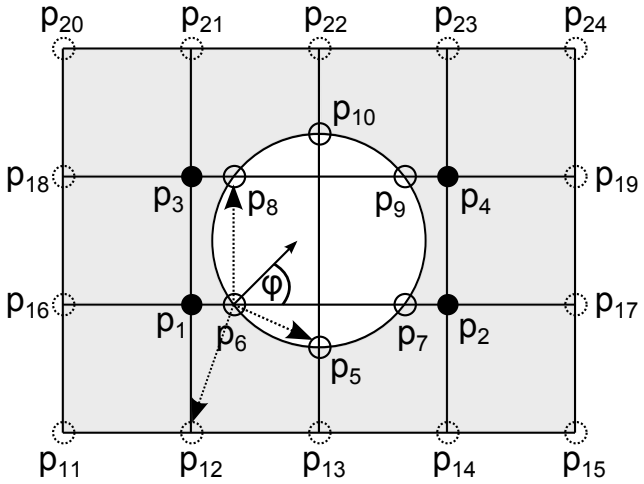
$$\tilde{u}_0 = \big(h(p_1), \ldots, h(p_{N_I+N_N})\big)^T \tag{2.20}$$

aus (1.5). Die Komponenten der Koeffizientenvektoren α^i und β^i lassen sich durch das Lösen eines LGS mit der Matrix (2.10) bzw. (2.18) und entsprechenden rechten Seiten errechnen, vgl. Beispiele 2.2 und 2.8.[2] Sie treten – ggf. multipliziert mit den entsprechenden Vorfaktoren $c_j, j = \{2, \ldots, 6\}$ – entweder in dem Vektor $b \in \mathbb{R}^{N_I+N_N}$ oder in der Matrix $A \in \mathbb{R}^{(N_I+N_N)\times(N_I+N_N)}$ auf, je nach dem, ob der jeweilige Nachbarpunkt ein Dirichlet-Randpunkt ist oder nicht. Die Bedingung an die Normalenableitung in den Neumannpunkten fließt in den Vektor b ein, ebenso wie die rechte Seite f. Diese Vorgehensweise soll am folgenden Beispiel demonstriert werden:

[1] Die Nummerierung beeinflusst die Gestalt und damit einige Eigenschaften der Matrix A.

[2] In der Praxis wurden die Gleichungssysteme mit der Computersoftware "Maple" allgemein gelöst und die Lösungsformeln fest im Gittergenerator implementiert.

Beispiel 2.9
Man betrachte das Gebiet Ω bzw. das Gitter G wie in der unteren Abbildung dargestellt und das ARWP aus Definition 1.1(b), wobei Γ_D der äußere Rand des Rechtecks und Γ_N der innere Rand des Kreises sei.



Das Problem lässt sich approximieren mit

$$\partial_t \tilde{u} = -A\tilde{u} + b = -\begin{pmatrix} \alpha_c^1 & 0 & \alpha_n^1 & 0 & 0 & \alpha_e^1 & 0 & 0 & 0 & 0 \\ 0 & \alpha_c^2 & 0 & \alpha_n^2 & 0 & 0 & \alpha_w^2 & 0 & 0 & 0 \\ \alpha_s^3 & 0 & \alpha_c^3 & 0 & 0 & 0 & 0 & \alpha_e^3 & 0 & 0 \\ 0 & \alpha_s^4 & 0 & \alpha_c^4 & 0 & 0 & 0 & 0 & \alpha_w^4 & 0 \\ 0 & 0 & 0 & 0 & \beta_c^5 & \beta_n^5 & \beta_e^5 & 0 & 0 & 0 \\ \beta_w^6 & 0 & 0 & 0 & \beta_e^6 & \beta_c^6 & 0 & \beta_n^6 & 0 & 0 \\ 0 & \beta_e^7 & 0 & 0 & \beta_w^7 & 0 & \beta_c^7 & 0 & \beta_n^7 & 0 \\ 0 & 0 & \beta_w^8 & 0 & 0 & \beta_s^8 & 0 & \beta_c^8 & 0 & \beta_e^8 \\ 0 & 0 & 0 & \beta_e^9 & 0 & 0 & \beta_s^9 & 0 & \beta_c^9 & \beta_w^9 \\ 0 & 0 & \beta_w^{10} & 0 & 0 & 0 & 0 & \beta_s^{10} & \beta_e^{10} & \beta_c^{10} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \end{pmatrix}$$

$$+ \begin{pmatrix} f(t,p_1) - \alpha_w^1 g_D(t,p_{16}) - \alpha_s^1 g_D(t,p_{12}) \\ f(t,p_2) - \alpha_s^2 g_D(t,p_{14}) - \alpha_e^2 g_D(t,p_{17}) \\ f(t,p_3) - \alpha_w^3 g_D(t,p_{18}) - \alpha_n^3 g_D(t,p_{21}) \\ f(t,p_4) - \alpha_e^4 g_D(t,p_{19}) - \alpha_n^4 g_D(t,p_{23}) \\ f(t,p_5) - \beta_d^5 g_N(t,p_5) - \beta_s^5 g_D(t,p_{13}) - \beta_w^5 g_D(t,p_{12}) \\ f(t,p_6) - \beta_d^6 g_N(t,p_6) - \beta_s^6 g_D(t,p_{12}) \\ f(t,p_7) - \beta_d^7 g_N(t,p_7) - \beta_s^7 g_D(t,p_{13}) \\ f(t,p_8) - \beta_d^8 g_N(t,p_8) - \beta_n^8 g_D(t,p_{21}) \\ f(t,p_9) - \beta_d^9 g_N(t,p_9) - \beta_n^9 g_D(t,p_{22}) \\ f(t,p_{10}) - \beta_d^{10} g_N(t,p_{10}) - \beta_n^{10} g_D(t,p_{22}) \end{pmatrix}$$

und Anfangsdaten $\tilde{u}_0 = \big(h(p_1), \ldots, h(p_{10})\big)^T \in \mathbb{R}^{10}$.
Beachte: Für Neumann-Randpunkte können andere Nachbarpunkte gewählt werden, solange die Matrix (2.18) regulär ist.

Bemerkung 2.10
In den Dirichlet-Randpunkten $p_i \in P_D$ genügt es, die Funktionen h (für $t = 0$) und g_D (für $t > 0$) aus Definition 1.1(b) auszuwerten.

Bemerkung 2.11
Falls in (1.6)

$$c_1 = 0$$

gilt und die Funktionen f, g_D und g_N von t unabhängig sind, muss man kein System gewöhnlicher Differentialgleichungen lösen. Es reicht, das lineare Gleichungssystem

$$A\tilde{u} = b \tag{2.21}$$

einmalig zu lösen.

Wenn man das Gitter aus Beispiel 2.9 verfeinert, ist die Struktur der Matrix besser erkennbar. Abbildung 2.6 zeigt die Struktur einer zum Beispiel 2.9 gehörenden 526×526-Matrix. Dunkle Stellen sind Nicht-Null-Einträge, während weiße Flächen Null-Einträge darstellen. Die Matrix ist sehr dünn besetzt: In der i-ten Zeile gibt es maximal fünf Nicht-Null-Einträge, nämlich die Koeffizienten α_j^i bzw. β_j^i, $j \in \{e, w, n, s, c\}, i \in \{1, \ldots, 526\}$.

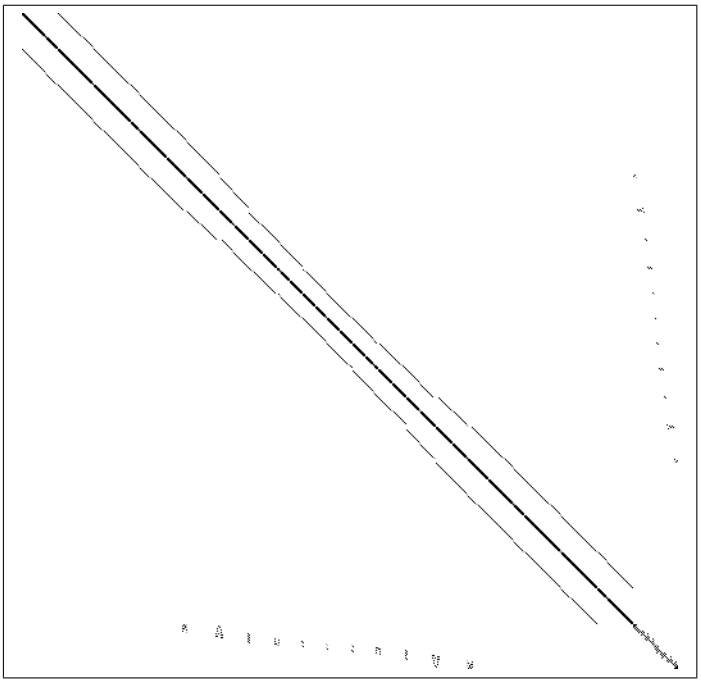


Abbildung 2.6: Matrixstruktur

Bemerkung 2.12
Sei $\Gamma_N = \emptyset$ und damit $\partial\Omega = \Gamma_D$, d.h. es existieren keine Neumann-Randpunkte. In diesem Fall fallen die "Unregelmäßigkeiten" in der Matrix unten und rechts weg. Nimmt man

zusätzlich an, dass $\bar{\Omega}$ ein Rechteck ist, so erhält man eine sogenannte Bandmatrix, vgl. Abbildung 2.7. Für Bandmatrizen gibt es spezielle schnelle Algorithmen zum Lösen linearer Gleichungssysteme.

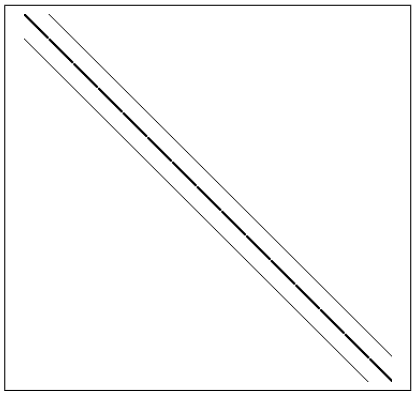


Abbildung 2.7: Bandmatrix

Bemerkung 2.13
Bei einer anderen Nummerierung der Gitterpunkte kann die Matrixstruktur erheblich abweichen. In Abbildung 2.8 werden die inneren Gitterpunkte und die Neumann-Randpunkte ebenfalls von links nach rechts und von unten nach oben nummeriert. Statt jedoch die inneren Gitterpunkte zuerst zu nummerieren, werden Neumann-Randpunkte und innere Punkte zeitgleich nach der Position im Gitter geordnet. Die daraus entstehende Matrixstruktur bei einer zum Beispiel 2.9 gehörenden 526×526-Matrix ist in Abbildung 2.9 dargestellt. Dabei wurde ausschließlich die Nummerierung verändert, alle anderen Parameter sind gleich.

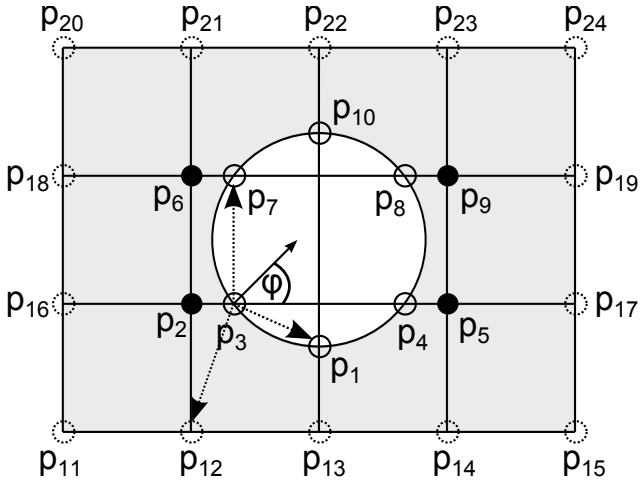


Abbildung 2.8: Alternative Gitterpunktnummerierung

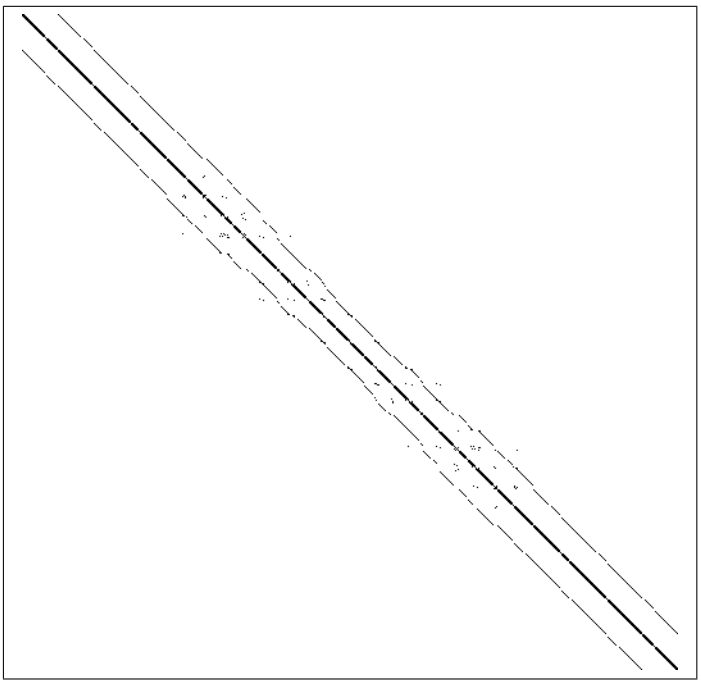


Abbildung 2.9: Matrixstruktur bei alternativer Nummerierung

Kapitel 3

Der Gittergenerator

Der Gittergenerator ist das praktische Herzstück dieser Arbeit. Er diskretisiert ein beschränktes Gebiet $\Omega \subset [0, \infty[^2$ und approximiert in den Gitterpunkten die gewünschten partiellen Ableitungen (bis zu zweiter Ordnung, allerdings keine gemischten Ableitungen) mit Hilfe der Finite-Differenzen-Methode. Damit lassen sich die Anfangs-/Randwertprobleme aus Definition 1.1(b) und Bemerkung 1.2 behandeln. Das Programm schreibt die Informationen der Matrix A und des Vektors b aus Abschnitt 2.3 in Textdateien, die anschließend zum Beispiel in einen DGL-Löser eingelesen werden können.
Weitere Textdateien werden erstellt, um dem Anwender die Visualisierung des Gebietes zu ermöglichen. In Verbindung mit den enthaltenen .plt-Dateien lässt sich zum Beispiel im frei verfügbaren Programm "Gnuplot" nicht nur das Gebiet selbst visualisieren, sondern auch die Nachbarschaftsbeziehungen zwischen den einzelnen Punkten.

3.1 Funktionsweise im Überblick

Bevor der Gittergenerator im Detail vorgestellt wird, ein kurzer Überblick. Das Programm erfüllt im Wesentlichen zwei Aufgaben, nämlich

1. die Diskretisierung eines Gebietes und

2. die Berechnung der Differenzenapproximationen in den Gitterpunkten.

Diese Aufgaben erledigt der Gittergenerator in je zwei Phasen, also insgesamt vier Phasen. Ausgehend von einem vorgegebenen $\Omega \subset [0, \infty[^2$ wird im ersten Schritt ein Rechteck-Gitter generiert, wobei das Rechteck Ω enthält. In Phase 2 wird das Rechteck-Gitter mit Hilfe von Funktionen, die Ω beschreiben, "zurechtgeschnitten" und damit Ω diskretisiert. Daneben wird für neu hinzugekommene Neumann-Randpunkte der Winkel ϕ der Normalenableitung $\frac{\partial u}{\partial \nu}$ berechnet. Nun folgt die entscheidende Phase 3, in der für Neumann-Randpunkte Nachbarn ausgewählt werden. Abschließend werden in Phase 4 die gewünschten partiellen Ortsableitungen in den Gitterpunkten gemäß Kapitel 2 approximiert und alle benötigten Informationen in Textdateien geschrieben.

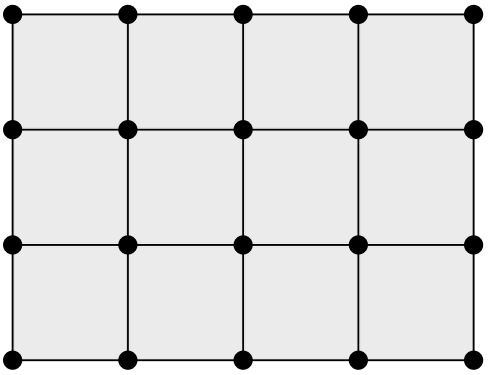


Abbildung 3.1: Phase 1: Erzeugen des Rechteck-Gitters

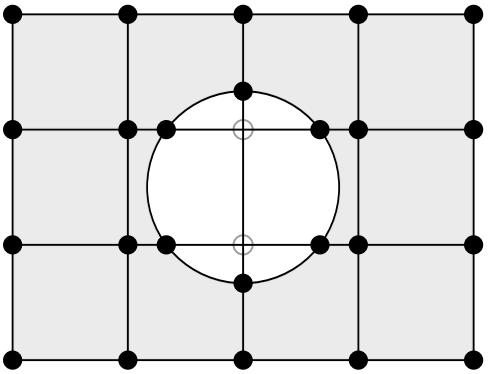


Abbildung 3.2: Phase 2: Zurechtschneiden des Gitters

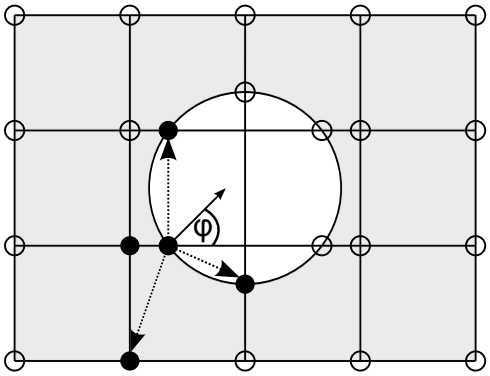


Abbildung 3.3: Phase 3: Behandlung der Neumann-Randpunkte

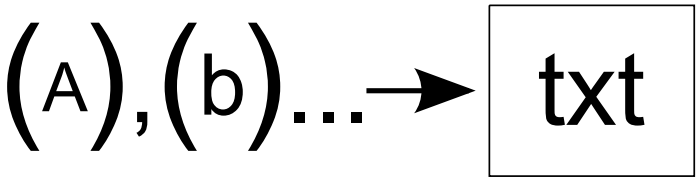


Abbildung 3.4: Phase 4: Differenzenapproximation und Dateiausgabe

Zum weiteren Verständnis sind einige Details zum Gittergenerator bzw. dessen Programmcode hilfreich. Dabei geht es um die Daten, die einen Gitterpunkt charakterisieren und, damit zusammenhängend, wie Gitterpunkte "von ihren Nachbarn wissen", d.h. wie die Nachbarbeziehungen zwischen den Punkten verwaltet werden. Dazu muss man wissen, dass die Punkte in einem Vektor abgelegt werden. Der Zugriff auf die einzelnen Gitterpunkte erfolgt über eine eindeutige Adresse, nämlich ihre Position in diesem Vektor. Diese Adressen (oder Positionen) werden wiederum in Listen gespeichert. Es gibt je eine Liste für

- (ip) innere Punkte,
- (bp_D) Dirichlet-Randpunkte,
- (bp_N) Neumann-Randpunkte und
- (dp) Punkte, die aus dem Gitter entfernt wurden.

Wenn ein Punkt aus dem Gitter entfernt wird, gibt das Programm seine Position im Vektor zum Überschreiben frei, indem die Adresse zu der Liste (dp) hinzugefügt wird und aus der entsprechenden Liste (ip), (bp_D) oder (bp_N) gelöscht wird. Fügt man einen neuen Randpunkt in das Gitter ein, so muss er in dem Vektor untergebracht werden. Das Programm prüft dabei zunächst anhand der Liste (dp), ob im Vektor überschreibbarer Speicherplatz vorhanden ist. Falls nicht, wird neuer Speicherplatz angefordert und der neue Punkt an das Ende des Vektors gesetzt. In jedem Fall wird die Position des Randpunktes im Vektor in der entsprechenden Liste (bp_D) oder (bp_N) gespeichert. Dieses Vorgehen ermöglicht es, den Speicherplatz effizient zu nutzen.
Jeder Gitterpunkt kann nun die folgenden Daten speichern:

- Koordinaten des Punktes (x, y)
- Adressen der Nachbarpunkte ("east", "west", "north", "south")
- Punkttyp ("inner", "boundary")
- Randtyp ("none", "Dirichlet", "Neumann") (für Randpunkte)
- Winkel ϕ (für Neumann-Randpunkte)
- Gitterpunktnummer

Die "Adressen der Nachbarpunkte" verweisen jeweils auf die Position des Nachbarpunktes in dem Vektor. Nachbarbeziehungen werden also hergestellt, in dem die Adresse des Nachbarpunktes im Gitterpunkt gespeichert wird. Entsprechend "kennt" ein Gitterpunkt immer nur maximal vier andere Gitterpunkte. Der "Gitterpunktnummer" kommt erst bei der Matrix-Generierung in Phase 4 eine Bedeutung zu, die in Abschnitt 3.2.4 näher betrachtet wird. Damit stehen nun genug Informationen bereit, um die Funktionsweise des Gittergenerators im Detail zu betrachten.

3.2 Funktionsweise im Detail

3.2.1 Phase 1: Erzeugen des Rechteck-Gitters

Im ersten Schritt wird ein Rechteck-Gitter erzeugt. Dazu gibt der Anwender zwei Eckpunkte $p_{lr} := (x_{lr}, 0)$ ("lower right") und $p_{ul} := (0, y_{ul})$ ("upper left") sowie die Anzahl der Gitterpunkte in x- und y-Richtung $N_x + 1$ bzw. $N_y + 1$ vor. Mit den Schrittweiten

$$h_x := \frac{x_{lr}}{N_x} \text{ und } h_y := \frac{y_{ul}}{N_y} \tag{3.1}$$

wird dann das Rechteck-Gitter

$$\bar{G} = \{(x, y) \in [0, x_{lr}] \times [0, y_{ul}] \mid x = n_1 h_x, y = n_2 h_y, n_1, n_2 \in \mathbb{N}_0\} \tag{3.2}$$

generiert. Des Weiteren kann der Anwender für jede Kante des Rechtecks separat festlegen, ob diese ein Dirichlet- oder ein Neumann-Rand sein soll. Für die Neumann-Randpunkte wird der Winkel ϕ der Normalenableitung $\frac{\partial u}{\partial \nu}$ ausnahmsweise bereits in Phase 1 berechnet, da dies für das Rechteck sehr simpel ist: $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. Jeder innere Gitterpunkt bekommt die "kanonischen" Nachbarpunkte aus Abschnitt 2.1 zugewiesen. Randpunkte haben in dieser Phase weniger als vier Nachbarpunkte. Bei den Dirichlet-Randpunkten ist das kein Problem: sie werden nicht benötigt, weil die Lösung u bereits vorgegeben ist. Ein Neumann-Randpunkt hingegen braucht nach Abschnitt 2.2 alle vier Nachbarpunkte; diese werden in Phase 3 ausgewählt.

Bemerkung 3.1

(a) Gitter können nur im ersten Quadranten erzeugt werden. Das mathematische Problem muss ggf. transformiert werden.

(b) (0,0) ist immer vorgegeben als linke untere Ecke in Phase 1. Dies ist jedoch keine große Einschränkung, da das Gitter in Phase 2 ohnehin angepasst wird.

3.2.2 Phase 2: Zurechtschneiden des Gitters

Das aktuelle Gitter soll nun zurechtgeschnitten werden. Technisch realisiert wird dies durch das Entfernen von Gitterpunkten und das Einfügen von neuen Randpunkten in das bestehende Gitter. Dazu gibt der Anwender

- eine stetig differenzierbare Funktion $F : \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto F(x, y)$,
- deren Ableitungen $\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}$ und
- einen Randtyp (Dirichlet oder Neumann)

an. In jedem Gitterpunkt $(x, y) \in \bar{G}$ wird dann die Funktion F ausgewertet, wobei per Konvention die Menge

$$\{(x, y) \in \bar{G} \mid F(x, y) < 0\}$$

aus dem bestehenden Gitter entfernt wird und die Menge

$$\{(x, y) \in \bar{G} \mid F(x, y) = 0\}$$

einen Rand bildet. Es gibt nun drei Fälle, die bei dieser Iteration über die Gitterpunkte auftreten können:

1. $F(x, y) > 0$
   In diesem Fall ist nichts zu tun, da diese Punkte per Konvention weiterhin im Gitter enthalten sind.

2. $F(x, y) = 0$
   Diese Punkte bleiben ebenfalls im Gitter enthalten. Das weitere Vorgehen hängt jedoch davon ab, ob (x, y) bisher ein innerer Gitterpunkt oder ein Randpunkt war.

   (a) (x, y) sei ein innerer Gitterpunkt
   Dann wird dieser zu einem Randpunkt mit dem Randtyp, den der Anwender angegeben hat.

   (b) (x, y) sei ein Randpunkt
   Hier muss weiter differenziert werden.

      i. Der Randtyp des Randpunktes stimme mit dem Randtyp, den der Anwender angegeben hat, überein.
      Für Dirichlet-Randpunkte muss nichts getan werden.
      Für Neumann-Randpunkte muss der Winkel ϕ der Normalenableitung $\frac{\partial u}{\partial \nu}(x, y)$ erneut berechnet werden.

      ii. Der Randtyp des Randpunktes stimme mit dem Randtyp, den der Anwender angegeben hat, nicht überein.
      In diesem Fall kann es zu Problemen bei der Differenzenapproximation kommen, da einige Fehlerabschätzungen ihre Gültigkeit verlieren, wenn Dirichlet- und Neumann-Randpunkte nahe beieinander sind. Der Dirichlet-Randpunkt wird im Programm immer bevorzugt, d.h. falls der Randtyp des Randpunktes "Neumann" ist, wird dieser auf "Dirichlet" geändert und falls der Randtyp des Randpunktes "Dirichlet" ist, bleibt dieser und wird nicht in "Neumann" umgeändert. In beiden Fällen wird dem Anwender eine entsprechende Warnung ausgegeben.

3. $F(x, y) < 0$
   In diesem Fall wird der Gitterpunkt per Konvention entfernt und die Nachbarpunkte des zu löschenden Gitterpunktes untersucht. Dabei muss für jeden gespeicherten Nachbarpunkt (x^*, y^*) eine weitere Fallunterscheidung gemacht werden:

   (a) $F(x^*, y^*) > 0$
   Da alle Gitterpunkte auf mindestens einer Gitterlinie liegen, gilt entweder $x^* = x$ oder $y^* = y$.

i. $x^* = x$; o.E. sei $y < y^*$, sonst vertausche die Intervallgrenzen.
$\Rightarrow \exists \hat{y} \in]y, y^*[: F(x^*, \hat{y}) = 0$ nach Zwischenwertsatz. Diese Nullstelle kann nun mit verschiedenen numerischen Verfahren approximiert werden. Im Gittergenerator wird eine Kombination aus dem Newton- und dem Bisektionsverfahren angewendet. Beachte, dass die x-Komponente fest ist und man daher diese Verfahren im eindimensionalen Fall verwenden kann. In diesem Fall wird für das Newton-Verfahren die Ableitung $\frac{\partial F}{\partial y}$ benötigt. Sowohl Newton- als auch Bisektionsverfahren werden als bekannt vorausgesetzt. Die Kombination dieser Verfahren soll jedoch kurz erläutert werden: Falls das Newton-Verfahren nicht die "richtige" Nullstelle findet und aus dem Intervall $]y, y^*[$ "herausspringt", wird das Intervall gemäß dem Bisektionsverfahren halbiert. Auf dem neuen Intervall wird dann ein Newton-Verfahren angewendet, wobei der Startpunkt immer die Mitte des aktuellen Intervalls ist. Dies geschieht so lange, bis eine gültige Lösung $\hat{y} \in]y, y^*[$ genau genug approximiert wird oder die maximale Anzahl an Intervallteilungen bzw. Newtoniterationen erreicht ist. Anschließend wird der Verweis von (x^*, y^*) auf (x, y) als Nachbarpunkt gelöscht und eine gegenseitige Nachbarschaftsbeziehung zwischen (x^*, y^*) und $(x^*, \hat{y})$ hergestellt. Somit hat der neue Randpunkt $(x^*, \hat{y})$ genau einen bekannten Nachbarn.

ii. $y^* = y$; o.E. sei $x < x^*$, sonst vertausche die Intervallgrenzen.
Analog mit $\hat{x}$.

(b) $F(x^*, y^*) = 0$
In diesem Fall muss keine Nullstelle gesucht werden, da sie bereits bekannt ist. Es wird lediglich die Nachbarschaftsbeziehung von (x^*, y^*) zu (x, y) gelöscht. Weitere Maßnahmen werden an dieser Stelle nicht unternommen, da im Verlauf der Iteration über alle Gitterpunkte der Punkt (x^*, y^*) ohnehin gemäß Fall 2 behandelt wird bzw. wurde.

(c) $F(x^*, y^*) < 0$
In diesem Fall muss nichts gemacht werden. Aus Effizienzgründen wird jedoch die Nachbarschaftsbeziehung von (x^*, y^*) zu (x, y) gelöscht. Wenn dann in der Iteration über die Gitterpunkte der Punkt (x^*, y^*) behandelt wird und Fall 3 eintritt, ist (x, y) nicht mehr als dessen Nachbar gespeichert. Somit spart man sich eine Auswertung $F(x, y)$.

Die Berechnung des Winkels ϕ für die Neumann-Randpunkte (x, y) erfolgt mittels

$$\phi = \operatorname{atan2}\left(-\frac{\partial F}{\partial y}(x, y), -\frac{\partial F}{\partial x}(x, y)\right) \in]-\pi, \pi],$$

einer Variante der Arkustangensfunktion. Sie hat gegenüber dem Arkustangens den Vorteil, den Winkel im korrekten Quadranten ermitteln zu können, da sie die Vorzeichen beider Komponenten berücksichtigt.

Bemerkung 3.2

(a) Es ist möglich, mehrere Funktionen F hintereinander an ein vorhandenes Gitter zu übergeben und Phase 2 damit mehrfach hintereinander auszuführen. Das macht es leichter, mit relativ einfachen Funktionen kompliziertere Gebiete zu erzeugen.

(b) Es gibt keine Möglichkeit, neue innere Gitterpunkte hinzuzufügen. Damit bietet es sich an, bei der Konstruktion von F den Blickwinkel zu wechseln: Anstelle zu beschreiben, welche Gitterpunkte weiterhin enthalten sein sollen, setzt man an den Gitterpunkten an, die entfernt werden sollen. So ist es beispielsweise leicht, aus einem vorhandenen Gitter einen Kreis mit Mittelpunkt (m_1, m_2) und Radius $\sqrt{r}$ "auszuschneiden":

$$F(x, y) := (x - m_1)^2 + (y - m_2)^2 - r$$

(c) In die Funktion F können zusätzlich feste Parameter eingebaut werden, wie im obigen Beispiel für den Kreis.

(d) Im obigen Fall 3a ist keine Aussage über die Eindeutigkeit der Nullstelle im Intervall $]y, y^*[$ bzw. $]x, x^*[$ gemacht worden. Die Eindeutigkeit muss aber gelten, damit das Gebiet sinnvoll diskretisiert werden kann. Um die Eindeutigkeit sicherzustellen, muss eine hinreichend feine Diskretisierung gewählt werden, der Grad der Feinheit ist a priori jedoch nicht klar. Eine geeignete Wahl der Schrittweite bleibt dem Anwender überlassen.

(e) Damit das Newton-Verfahren konvergiert, ist es hinreichend, dass F in beiden Variablen zwei mal stetig differenzierbar ist.

3.2.3 Phase 3: Behandlung der Neumann-Randpunkte

Ist das gewünschte Gebiet diskretisiert, müssen nun die für die Differenzenapproximation benötigten Nachbarpunkte bestimmt werden. Am Ende der Phase 2 hat jeder innere Gitterpunkt seine vier Nachbarn wie in Abschnitt 2.1. Bei den Dirichlet-Randpunkten spielen sie keine Rolle und so bleiben noch die Neumann-Randpunke. Diese haben am Ende der Phase 2 meist nur einen bekannten Nachbarpunkt, die weiteren müssen erst "gefunden" werden. Erschwerend kommt hinzu, dass nicht beliebige Nachbarpunkte in Frage kommen. Zum einen muss die Regularität der Matrix (2.18) sichergestellt werden, zum anderen sollten die Nachbarpunkte nicht zu weit weg sein, damit der Fehler, den man durch das Vernachlässigen der Restterme in den Taylorentwicklungen (2.13) - (2.16) in Abschnitt 2.2 macht, möglichst klein bleibt. Weiterhin sollten die Punkte auch nicht zu nah gewählt werden, denn je näher die Punkte liegen, umso größer werden die Komponenten der Koeffizientenvektoren α und β im Betrag. Dies hat eine höhere Kondition der Matrix A zur Folge. Eine höhere Kondition bedeutet, dass das Problem numerisch schwieriger zu handhaben ist. Die Ziele sind also, für jeden Neumann-Randpunkt

1. Gitterpunkte in der Nähe zu finden und als Kandidaten zu markieren und

2. aus dieser Liste von Kandidaten vier "möglichst gute" Nachbarpunkte auszuwählen.

Das erste Ziel wird mit einem etwas technischen Algorithmus erreicht, dessen Details nicht sonderlich ausschlaggebend sind. Sicher gibt es andere Verfahren, die zum Ziel führen. Außerdem ist es kein mathematisches Problem. Doch die Beschreibung des Algorithmus hilft, die Probleme und Grenzen des Gittergenerators weiter zu verstehen und soll dem interessierten Leser nicht vorenthalten werden.
Ausgehend von dem Neumann-Randpunkt (x_0, y_0) geht man zu einem vorhandenen Nachbarpunkt (x_1, y_1). Von dort aus geht es weiter zu einem Nachbarpunkt (x_2, y_2) von (x_1, y_1), usw., wobei man nur $credit \in \mathbb{N}$ mal in jede Richtung gehen kann, d.h. *credit* mal jeweils den "east"-, "west"-, "north"- und "south"-Nachbarpunkt auswählen kann. Hat man seine Züge aufgebraucht, so geht es einen Schritt zurück. Dabei wird in jedem "Rückwärts"-Schritt keine Richtung "abgezogen", sondern die entgegengesetzte Richtung "gutgeschrieben". Geht man also einen Schritt nach "north" zurück, könnte man wieder einen Schritt in Richtung "south" tun. Um eine Endlosschleife zu verhindern, wird in jedem Schritt überprüft, ob der zu besuchende Punkt bereits auf der Kandidatenliste steht. Falls ja, wird stattdessen eine andere Richtung probiert, sofern man noch Züge in diese Richtung übrig hat. Immer wenn ein Punkt nicht besucht wird, entweder weil er bereits bekannt ist oder weil es keinen Nachbarpunkt in dieser Richtung gibt, wird diese Richtung nicht "abgezogen". Sind die Punkte in allen Richtungen, in die man gehen kann, bekannt, so wird ein weiterer Rückwärts-Schritt getan. Dies geschieht so lange, bis alle erreichbaren Punkte in der Umgebung bekannt sind. Die Größe der Umgebung wird durch die vom Anwender vorgegebene Zahl *credit* bestimmt.

Bemerkung 3.3

(a) Trifft man in der oben beschriebenen Vorgehensweise in einem Schritt auf einen Neumann-Randpunkt, so wird dieser Punkt natürlich in die Kandidatenliste aufgenommen. Allerdings muss man in diesem Fall einen Schritt zurückgehen, auch wenn man noch in andere Richtungen gehen könnte. Der Grund dafür ist, dass man ab Phase 3 a priori nicht weiß, auf welche Nachbarpunkte ein Neumann-Randpunkt verweist. Abhängig von der Auswahlstrategie können katastrophale Ergebnisse entstehen. Ein Beispiel für ein unglückliches Szenario ist in Abbildung 3.5 dargestellt: Sehr weit entfernte Punkte werden als "Nachbarn" ausgewählt, was wiederum zu schlechten Resultaten führt, da die vernachlässigten Restterme in den Taylorentwicklungen groß werden.

(b) Es gibt Fälle, in denen nicht genügend Kandidaten gefunden werden. Sie treten beispielsweise bei exotischen Gebieten und großer Schrittweite auf, vgl. Abbildung 3.6. Das Programm muss dabei beendet werden, weil man ohne die Nachbarpunkte nicht weiterkommt. Alternativ wäre eine Löschung des Punktes denkbar, aber auch das wäre nicht befriedigend.

(c) Man sollte n möglichst klein wählen, um nicht zu ferne Nachbarpunkte zu bekommen. Allerdings reicht $n = 1$ nicht aus, vgl. Abbildung 2.4. Der Standard ist daher $n = 2$.

Damit wäre das erste Ziel erreicht. Nun möchte man aus der Kandidatenliste eine möglichst gute Nachbarpunktkombination auswählen. "Möglichst gut" heißt dabei, dass das Endergebnis, also die Lösung des DGL-Systems, möglichst nah an der exakten Lösung sein sollte.

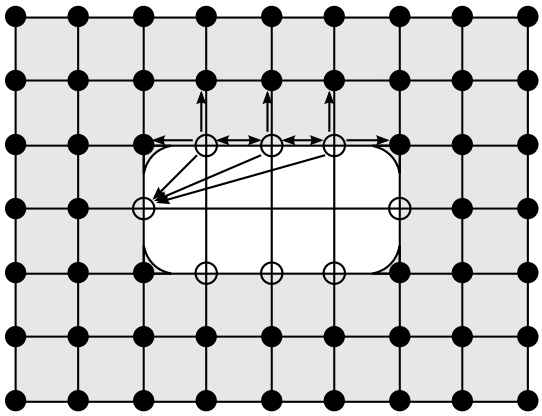


Abbildung 3.5: Suche nach Nachbarpunkten: “Kettenreaktion”

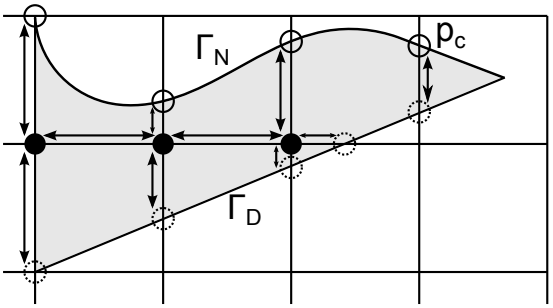


Abbildung 3.6: Suche nach Nachbarpunkten: Nicht genügend Nachbarpunkte

Bis dahin fließen jedoch noch viele weitere Faktoren ein. Aus diesem Grund wurde das Problem heuristisch angegangen: Verschiedene Auswahlstrategien wurden entwickelt, die auf verschiedene Aspekte Wert legen. Die Ansätze waren:

- Wähle die (in der euklidischen Norm) nächstgelegenen vier Nachbarpunkte aus.
- Wähle diejenigen (in der euklidischen Norm) nächstgelegenen vier Nachbarpunkte aus, die einen vorgegebenen Mindestabstand zum Neumann-Randpunkt haben.
- Wähle die Nachbarpunkte so aus, dass jeweils ein Punkt möglichst links, rechts, oben oder unten liegt. Die Idee war, die Differenzenapproximation, die bei den inneren Punkten verwendet wird, nachzubauen.
- Bevorzuge Randpunkte und wähle dann die inneren Gitterpunkte mit dem kleinsten Adresswert in dem Vektor, in dem alle Gitterpunkte hinterlegt sind.
- Wähle die Nachbarpunkte so, dass die Kondition $\text{cond}_\infty(B)$ minimal wird, wobei B die Matrix (2.18) ist und die Kondition mit

$$\text{cond}_\infty(A) := ||A||_\infty ||A^{-1}||_\infty$$

für reguläre Matrizen A definiert ist. Damit sollte untersucht werden, ob sich die Kondition der Matrix B positiv auf die Kondition der großen Matrix A aus (2.19) auswirkt.[1]

- Wähle die Nachbarpunkte so, dass die gewichtete Kondition $\gamma \cdot \text{cond}_\infty(B)$ minimal wird, wobei B die Matrix (2.18) ist und $\gamma \in \mathbb{R}$ noch festgelegt wird.
- Bevorzuge sogenannte "Neumann-randnahe" Punkte. Das sind innere Gitterpunkte, die auf mindestens einen Neumann-Randpunkt verweisen.

Zusätzlich können die beiden folgenden Tests durchgeführt werden.

Definition 3.4 *(Tests)*
Eine Menge aus vier Gitterpunkten besteht den

(a) Regularitätstest $:\Leftrightarrow$ *(2.18) regulär*

(b) Betatest $:\Leftrightarrow \beta_i \neq 0, i \in \{e, w, n, s, c, d\}$

Wenig überraschend ist die Forderung nach der Regularität der Matrix (2.18). Zu akademischen Zwecken sind jedoch Strategien eingebaut, die auf den Regularitätstest verzichten. Der Betatest dürfte etwas verwundern, weil man auf den ersten Blick keine Notwendigkeit dafür sieht. Er hat sich jedoch in der Praxis bewährt: Je nach Gebiet und Auswahlstrategie kommt es vor, dass einige Komponenten von β Null sind, d.h. $\exists i \in \{e, w, n, s, c, d\} : \beta_i = 0$. Dies kann dazu führen, dass die Matrix A aus Abschnitt 2.3 singulär wird und somit das in Bemerkung 2.11 erwähnte lineare Gleichungssystem nicht gelöst werden kann und die Ergebnisse unbrauchbar werden. Je häufiger einzelne Komponenten von β Null sind, umso höher ist die Gefahr, dass die Matrix singulär wird. Allerdings weiß man a priori nicht, welche Einträge zu einer Singularität führen. Daher werden alle Einträge überprüft, einschließlich β_d, damit man sicherstellen kann, dass die Neumann-Randbedingung erfüllt ist. Sollte es vorkommen, dass es keine mögliche Auswahl an Nachbarpunkten aus der Auswahlliste gibt, die den Betatest besteht, wird der Betatest für diesen Neumann-Randpunkt übersprungen, um wenigstens zu versuchen, ein brauchbares Ergebnis zu liefern. Der Anwender wird darüber mit einer Warnung informiert. Der Betatest kann nur von Strategien ausgeführt werden, die auch den Regularitätstest durchführen. Der Anwender kann jedoch einstellen, ob der Betatest durchgeführt werden soll. Es gibt noch weitere Einstellungsmöglichkeiten, die im Laufe der Arbeit entstanden sind und sich für manche Auswahlverfahren als hilfreich erwiesen haben. Diese und bereits erwähnte Optionen sind in Tabelle 3.1 zusammengefasst. Sie spielen in den nun kommenden Strategiebeschreibungen eine wichtige Rolle.

Bemerkung 3.5

(a) In den Strategien wird oft nach einem minimalen Element gesucht. Dazu werden die Listen der potentiellen Nachbarpunkte nach dem entsprechenden Kriterium sortiert, z.B. nach dem Abstand. Diese Aufgabe erledigt der "sort"-Befehl der "Standard Template Library" für `C++` *für Listen, vgl. [5]. Der als Hilfsmittel benutzte Algorithmus regelt nach eigenen Kriterien, wie er mit gleichen Werten verfahren soll. Daher wird an dieser Stelle keine Aussage darüber gemacht, welche Werte in diesem Fall bevorzugt werden.*

[1] Numerische Experimente unterstützen diese Hoffnung, vgl. Kapitel 4.

(b) Die Option "deleteOldNeighbours" hat nur Auswirkungen auf "non-validated"-Strategien. Die "validated"-Strategien überschreiben immer die vorhandenen Nachbarpunkte.

Option	Beschreibung
strategy	Damit bestimmt der Anwender, welcher der oben genannten Ansätze verfolgt wird. Gleichzeitig legt er damit fest, ob der Regularitätstest durchgeführt wird.
doBetatest	Entscheidet darüber, ob der Betatest durchgeführt wird.
hThresholdDelDistNeighbours	Diese Zahl gibt an, welchen Wert die größte Schrittweite $\max\{h_x, h_y\}$ haben darf, bis zu der weiter entfernte Punkte **nicht** von vornherein ausgeschlossen werden. Weiter entfernte Punkte sind alle Kandidaten mit Abstand $> \frac{3}{2}\max\{h_x, h_y\}$ zum Neumann-Randpunkt.
deleteOldNeighbours	Diese Option legt fest, ob bereits vorhandene Verweise auf Nachbarpunkte überschrieben werden sollen.

Tabelle 3.1: Einstellungsmöglichkeiten der Auswahlstrategien

Die Auswahlverfahren lassen sich unterteilen in jene, die den Regularitätstest durchführen ("validated") und jene, die es nicht tun ("non-validated"). Betrachtet werden zunächst letztere, da sie eine einfachere Struktur haben. Es sind vier solcher Strategien implementiert.

Strategie 1: NV_NEAREST
Verfügbare Optionen: deleteOldNeighbours
Diese Strategie wählt die bezüglich der euklidischen Norm nächstgelegenen Punkte aus. Die Hoffnung besteht darin, den Fehler, den man durch das Weglassen der Restterme in den Taylorentwicklungen macht, möglichst gering zu halten. Neben ihrer Einfachheit hat diese Strategie den Vorteil, dass durch das Auswahlkriterium keine potentiellen Nachbarpunkte ausgeschlossen werden und man somit nicht Gefahr läuft, zu viele Punkte auszuschließen.

Strategie 2: NV_NEAREST_MIN_DIST
Verfügbare Optionen: deleteOldNeighbours
Ähnlich zu Strategie 1 ist das Auswahlkriterium der euklidische Abstand. Allerdings versucht diese Strategie, zu nahe gelegene Punkte als Nachbarpunkte auszuschließen. Als Grenzwert hat sich in der Praxis $\frac{1}{4}\max\{h_x, h_y\}$ bewährt, wobei h_x und h_y wie in (3.1) definiert sind. Es werden nur Punkte entfernt, wenn noch mindestens so viele Punkte zur Auswahl stehen wie benötigt werden. Die Punkte mit dem kleinsten Abstand werden zuerst entfernt. Im Wesentlichen ist es eine Erweiterung von Strategie 1. Sie wurde entworfen, um eine weitere Fehlerquelle zu berücksichtigen. Zwar wird der Fehler in den Taylorentwicklungen durch die nahen Punkte möglicherweise reduziert, aber je näher die Punkte liegen, umso größer wird die Kondition der Matrix A. Mit dieser Strategie wird versucht, einen Mittelweg zu gehen.

Strategie 3: NV_OPTIMIZE_DIRECTION
Verfügbare Optionen: deleteOldNeighbours
Diese Strategie wählt die Nachbarpunkte so aus, dass jeweils ein Punkt möglichst links, rechts, oben oder unten liegt. Dazu wird die Umgebung um den Nachbarpunkt in vier Teilgebiete unterteilt, die die Richtungen "east", "west", "north" und "south" annährend wiedergeben sollen, vgl. Abbildung 3.7. Gibt es in einem Teilgebiet keine Punkte, so wird in dieser Richtung kein Nachbar zugewiesen. Sollten am Ende des Durchlaufs Nachbarpunkte benötigt werden, wird für die restlichen Punkte die Strategie 2 aufgerufen und eine entsprechende Meldung an den Anwender ausgegeben. Die Idee dieser Strategie war, die Differenzenapproximation, die bei den inneren Punkten verwendet wird, nachzubauen. Es hat sich als schlechte Idee herausgestellt, da dadurch die Matrix (2.18) singulär bzw. sehr schlecht konditioniert wird. Diese Strategie ist nicht empfehlenswert und wurde nur als schlechtes Beispiel beibehalten.

Strategie 4: NV_PREFER_BOUNDARY
Verfügbare Optionen: deleteOldNeighbours
Mit diesem Verfahren werden Randpunkte bevorzugt ausgewählt. Sollten dann noch Nachbarpunkte benötigt werden, werden sie mit inneren Gitterpunkten aufgefüllt. In beiden Fällen wird der Punkt mit der niedrigsten Position in dem Vektor, in dem alle Punkte gespeichert werden, ausgewählt. Dies hat keinen besonderen Grund, sondern sollte vielmehr eine technische Vereinfachung sein. Sie hat jedoch Folgen. Dadurch werden bei den inneren Gitterpunkten tendenziell diejenigen gewählt, die sich links unten im Gitter befinden. Bei den Randpunkten lässt sich diesbezüglich keine eindeutige Aussage machen, da Randpunkte an beliebige Stellen in den Vektor eingefügt werden können, sofern der entsprechende Platz zum Überschreiben freigegeben ist. Besonders wichtig bei dieser Strategie ist die Größe der Umgebung, die nach Nachbarpunkten abgesucht wird. Ist der *credit*-Wert zu hoch und die Umgebung damit zu groß, werden auch sehr weit entfernte Punkte bevorzugt, wenn sie Randpunkte sind. Der Fehler, den man durch das Weglassen der Restterme in den Taylorentwicklungen macht, wird dabei größer und führt oftmals zu schlechten Resultaten.

Die implementierten "validated"-Strategien sind technisch etwas aufwändiger, da einige Sicherheitsmechanismen eingebaut werden mussten. So können durch das Entfernen von zu weit entfernten Nachbarn, durch den Betatest und durch das Auswahlkriterium jeweils so viele Kandidaten gelöscht werden, dass nicht mehr genügend übrig bleiben. Daher werden an geeigneten Stellen Sicherheitskopien gemacht. Auch werden die Punkte etwas anders gehandhabt. Durch den Regularitätstest werden jeweils vier Nachbarpunkte, die den Test bestehen, zu einer Menge zusammengefasst. Anstelle jeden Nachbarpunkt nur einmal zu betrachten, betrachtet man jetzt diese Mengen, d.h. Kombinationen von jeweils vier Nachbarpunkten. Folglich müssen die Auswahlkriterien an diese neue Form angepasst werden. Diese etwas andere Formulierung kann, zusammen mit dem in Bemerkung 3.5 erwähnten "sort"-Algorithmus, trotz gleichem Ansatz zu unterschiedlichen Ergebnissen zwischen "validated"-

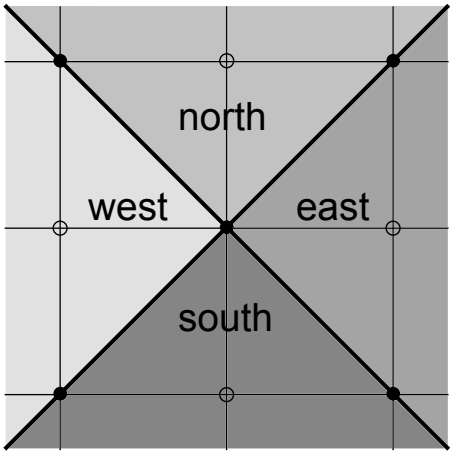


Abbildung 3.7: Teilgebieteinteilung für Strategie 3

und "non-validated"-Strategien führen, auch wenn beide Ergebnisse im Sinne des Regularitätstests zulässig wären. Dieses Phänomen tritt auf, wenn die Werte, nach denen sortiert wird, gleich sind. Es könnte durch Einfügen eines weiteren, sicher eindeutigen Auswahlkriteriums beseitigt werden. Ein Beispiel hierfür wäre die Position, die ein Punkt in dem Vektor hat, in dem alle Gitterpunkte gespeichert werden. Um die folgenden fünf Strategien nicht komplizierter zu machen als sie sind, wurde auf diese Maßnahme verzichtet.

Strategie 5: V_NEAREST
Verfügbare Optionen: doBetatest
Diese Strategie ist sehr ähnlich zu Strategie 1. Auch hier ist der euklidische Abstand das Auswahlkriterium. Die wichtigen Unterschiede sind die Durchführung des Regularitätstests und – auf Wunsch – die des Betatests. Technisch bedingt wird hier das Minimum über die Summe der vier Abstände zu den Nachbarpunkten gesucht, also

$$\min \sum_{i \in \{e,w,n,s\}} d_i,$$

wobei d_i der euklidische Abstand vom aktuellen Neumann-Randpunkt zum jeweiligen potentiellen Nachbarpunkt ("east", "west", "north", "south") sei. Dieses Verfahren ist relativ einfach und löscht, wie auch das "non-validated"-Analogon, keine Kandidaten durch das Auswahlkriterium. Damit bietet sich diese Strategie als Notfalllösung an, falls durch andere Kriterien alle gültigen Kandidaten herausgefiltert werden.

Strategie 6: V_NEAREST_MIN_DIST
Verfügbare Optionen: doBetatest
Die "validated"-Variante der Strategie 2 geht genauso vor wie Strategie 5 mit

dem einzigen Unterschied, eine im Sinne des Regularitätstests gültige Nachbarpunktkombination zu verwerfen, wenn mindestens ein Nachbarpunkt einen Abstand $\leq \frac{1}{4}\max\{h_x, h_y\}$ zum aktuellen Neumann-Randpunkt hat.

Strategie 7: V_COND
Verfügbare Optionen: doBetatest, hThresholdDelDistNeighbours
Bisher haben die meisten vorgestellten Strategien einen geometrischen Ansatz. Diese Strategie versucht eine andere Herangehensweise: Sicher ist, dass die Matrix (2.18), im Folgenden mit B bezeichnet, bei der Auswahl der Nachbarpunkte eine wichtige Rolle spielt. Sie darf beispielsweise nicht singulär werden. Wenn sie jedoch sehr schlecht konditioniert ist, hat das ebenso üble Auswirkungen. Also wird hier versucht, diese Kondition zu minimieren, d.h.

$$\min \ \mathrm{cond}_\infty(B).$$

Die Hoffnung besteht, dass sich die Kondition der Matrix B positiv auf die Kondition der großen Matrix A aus (2.19) auswirkt und damit ein besseres Endergebnis hervorbringt. Trotz des etwas anderen Ansatzes spielt auch hier die Entfernung der Punkte vom Neumann-Randpunkt eine wichtige Rolle. Je näher die Punkte, desto schlechter wird die Kondition. Daher neigt diese Strategie, weiter entfernte Punkte zu bevorzugen, was wiederum im Konflikt mit dem Wunsch steht, nähere Punkte auszuwählen. Aus diesem Grund werden hier auf Wunsch von vornherein weiter entfernte Punkte ausgeschlossen.

Strategie 8: V_COND_W
Verfügbare Optionen: doBetatest, hThresholdDelDistNeighbours
Diese Erweiterung von Strategie 7 knüpft an der Problematik an, dass weiter entfernte Punkte bevorzugt werden. Die Idee ist hier, den Wert der Kondition mit dem Abstand zu gewichten. Das Gewicht, das hier gewählt wurde, ist der maximale Abstand des Neumann-Randpunktes zu den vier Nachbarpunkten. Insgesamt ergibt sich damit

$$\min \left[\left(\max_{i \in \{e,w,n,s\}} d_i \right) \mathrm{cond}_\infty(B) \right],$$

wobei d_i der euklidische Abstand vom aktuellen Neumann-Randpunkt zum jeweiligen Nachbarpunkt ("east", "west", "north", "south") sei. Diese Gewichtung kann sicherlich noch verbessert werden. Auch hier besteht die Möglichkeit, weiter entfernte Punkte auszuschließen.

Strategie 9: V_PREFER_INNER_NEAR_NEUMANN
Verfügbare Optionen: doBetatest, hThresholdDelDistNeighbours
Die Idee für das letzte hier vorgestellte Verfahren entstand aus dem Wunsch,

weiterhin den Kompromiss zwischen nahen Punkten und guter Kondition zu suchen und der Hoffnung, innere Gitterpunkte wären bei der Auswahl "geeigneter" als Neumann-Randpunkte. Die Hoffnung rührt daher, dass inneren Gitterpunkten "kanonische" Nachbarn zugewiesen werden können und sie daher "robuster" erscheinen. Wie der Name der Strategie suggeriert, werden hier innere Gitterpunkte, die Neumann-Randpunkte als Nachbarn eingetragen haben, bevorzugt. Technisch bedingt müssen auch hier immer Mengen von jeweils vier Nachbarpunkten betrachtet werden. Die üblichen Bezeichnungen "east", "west", "north" und "south" seien hier durch die Buchstaben e, w, n und s ausgedrückt, damit klar ist, was mit "Kandidat i" im Folgenden gemeint ist. Es wird das Minimum

$$\min \sum_{i \in \{e,w,n,s\}} f_i,$$

gebildet, wobei

$$f_i := \begin{cases} -1, & \text{falls Kandidat } i \text{ auf mind. einen Neumann-Randpunkt verweist} \\ 0, & \text{sonst} \end{cases}$$

Auch bei diesem Verfahren lassen sich zu weit entfernte Nachbarpunkte von vornherein ausschließen.

Nachdem die Auswahlkriterien in den Strategien vorgestellt wurden, soll zum Abschluss der allgemeine technische Ablauf für die Verfahren 5 bis 9, insbesondere die Sicherheitsmechanismen, näher erläutert werden. Da die Strategien 5 und 6 keine Option bieten, weiter entfernte Kandidaten von vornherein auszuschließen, sind sie etwas leichter zu handhaben und werden daher zuerst präsentiert. Für alle potentiellen Nachbarpunktkonstellationen wird zunächst der Regularitätstest durchgeführt. Jede Konstellation, die den Test nicht besteht, wird ausgeschlossen. Wird der Regularitätstest für keine Kombination von Nachbarpunkten bestanden, muss mit einer Fehlermeldung an dieser Stelle abgebrochen werden. Ansonsten führt der Gittergenerator für die verbleibenden Konstellationen den Betatest aus, sofern gewünscht. Auch hier wird jede Kombination von Nachbarpunkten, die den Test nicht besteht, ausgeschlossen. Falls danach keine potentiellen Nachbarn mehr vorhanden sind, muss auf den Betatest verzichtet werden in der Hoffnung, trotzdem ein brauchbares Resultat zu erhalten. In diesem Fall werden alle Nachbarpunktkonstellationen, die den Regularitätstest bestanden haben, mit Hilfe einer Sicherheitskopie wieder in Betracht gezogen. Erst jetzt kommt eines der obigen Auswahlkriterien zum Zuge. Dabei können im Allgemeinen weitere gültige Nachbarpunktkonstellationen gelöscht werden. Sollte keine Konstellation mehr übrig bleiben, bieten sich hier zwei Möglichkeiten an:

1. Überspringe den Betatest, sofern er durchgeführt und die Sicherheitskopie noch nicht verwendet wurde oder

2. verwende ein anderes Auswahlkriterium.

Implementiert wurde die zweite Lösung mit der Intuition, dass der Betatest wichtiger sei als das Auswahlkriterium. Es wird auf die Strategie 5 ausgewichen, da durch deren Auswahlkriterium keine Kandidaten gelöscht werden. Die Strategien 7 bis 9 laufen ähnlich ab mit dem Unterschied, dass vor dem Regularitätstest ggf. zu weit entfernte Nachbarpunkte gelöscht werden. Sie werden nicht erst nach dem Regularitätstest gelöscht. Zwar geht man damit das Risiko ein, schlimmstenfalls den Test erneut durchzuführen. In der Regel erhält man aber so ein schnelleres Ergebnis, da der Test nicht für alle Punkte durchgeführt werden muss. Der darauffolgende Betatest hat eine hohe Priorität: Schlägt dieser fehl, wird versucht, die weiter entfernten Punkte wieder hinzuzunehmen. Wird der Betatest erneut nicht bestanden, muss darauf verzichtet werden. Wie zuvor wird auf ein anderes Auswahlkriterium zurückgegriffen, falls durch das ursprüngliche Auswahlkriterium alle möglichen Konstellationen aussortiert wurden. Dies ist jedoch lediglich ein theoretisches Gerüst für mögliche zukünftige Auswahlkriterien, denn die aktuellen Strategien 7 bis 9 verwerfen in diesem Schritt keine Nachbarpunkte mehr. Abbildung 3.8 fasst die Abläufe zusammen.

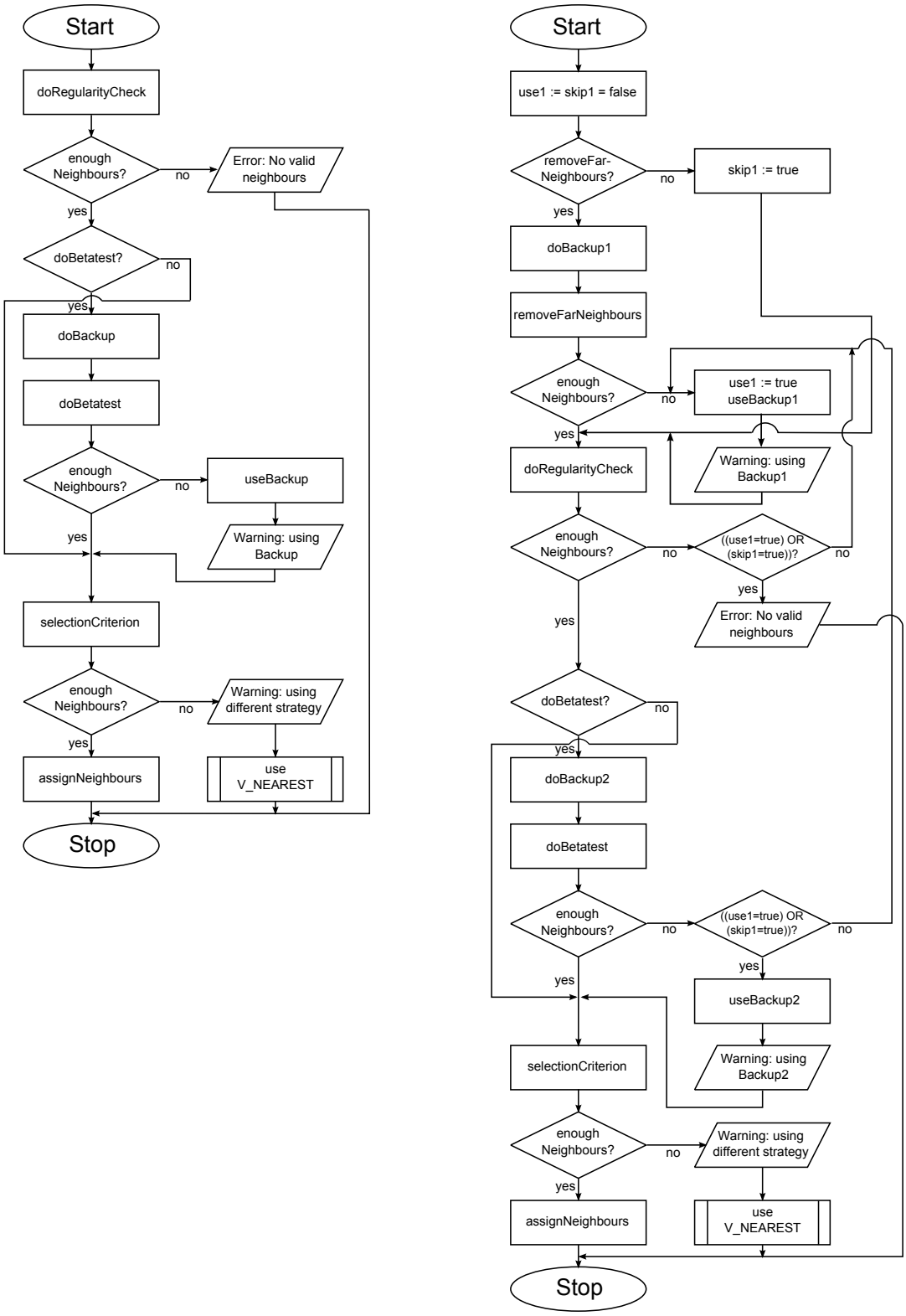


Abbildung 3.8: Grober Ablauf für die Strategien 5 und 6 (links) und 7 bis 9 (rechts)

3.2.4 Phase 4: Differenzenapproximation und Dateiausgabe

Nachdem sichergestellt wurde, dass jeder innere Gitterpunkt und jeder Neumann-Randpunkt vier Nachbarn hat, bleibt im letzten Schritt des Gittergenerators nur noch die Generierung der Matrix A und des Vektors b aus Abschnitt 2.3. Die Hauptschwierigkeit besteht darin, dies möglichst speichereffizient zu gestalten. Insbesondere sollte man die Dünnbesetztheit der Matrix ausnutzen.
Zunächst sollte man jedoch wissen, wo genau in der Matrix die Koeffizienten α_i bzw. β_i für einen gegebenen Punkt auftauchen. Dies hängt von der Nummerierung der Gitterpunkte ab. Für einen Menschen stellt es kein Problem dar, die aktuellen Gitterpunkte nach einem vorgegebenen System zu nummerieren und daraufhin die Matrix aufzustellen. Der Computer hingegen braucht etwas Hilfestellung, da er mit den vorhandenen Informationen aus den verwendeten Datenstrukturen auskommen muss. Eine Information wäre die eindeutige Kennzeichnung der Punkte durch ihre Position in dem Vektor, in dem alle Punkte gespeichert sind. Diese Nummern können jedoch nicht verwendet werden, weil in dem Vektor ggf. Punkte enthalten sind, die nicht mehr im Gitter vorhanden sind. Das kann vorkommen, weil diese Punkte gemäß Abschnitt 3.2.2 nicht aus dem Vektor entfernt werden, sondern lediglich zum Überschreiben freigegeben werden. Eine ständige Anpassung des Vektors, so dass er immer genauso lang ist wie die Anzahl der noch im Gitter enthaltenen Punkte, wäre zu aufwändig und ineffizient. Für eine effiziente Lösung muss man also eine Möglichkeit finden, nur die Punkte, die jetzt noch im Gitter enthalten sind, zu nummerieren. Die implementierte Lösung ermöglicht eine Dateiausgabe der Matrix, ohne speziell Speicherplatz dafür zu reservieren. Es wird lediglich Speicherplatz für die fünf bzw. sechs Werte von α bzw. β benötigt, die immer wieder überschrieben werden können.
Der Trick besteht darin, die Positionen der Punkte in den Listen (ip) und (bp_N) auszunutzen. In diesen Listen sind nur die noch im Gitter enthaltenen Punkte vorhanden. Der Gittergenerator beherrscht dabei die zwei in Abschnitt 2.3 vorgestellten[2] Möglichkeiten der Gitterpunktnummerierung, die kurz wiederholt werden. Seien dazu P_I die Menge der inneren Punkte und P_N die der Neumann-Randpunkte. Die beiden Ansätze sind:

1. Nummeriere die inneren Gitterpunkte und die Neumann-Randpunkte jeweils von links nach rechts und von unten nach oben. Damit haben die inneren Gitterpunkte die Nummern 1 bis $|P_I|$, die Neumann-Randpunkte die Nummern $|P_I| + 1$ bis $|P_I| + |P_N|$.

2. Nummeriere die inneren Gitterpunkte und die Neumann-Randpunkte zusammen von links nach rechts und von unten nach oben. Damit wird kein Unterschied mehr zwischen inneren Gitterpunkten und Neumann-Randpunkten gemacht. Sie haben allgemein die Nummern 1 bis $|P_I| + |P_N|$.

Bemerkung 3.6
Beachte, dass die Dirichlet-Randpunkte in der Implementierung nicht speziell nummeriert werden müssen, weil sie nicht in der Matrix A auftauchen. Es reicht, deren Position in dem Vektor, in dem alle Punkte gespeichert werden, zu kennen.

[2] vgl. Beispiel 2.9 und Bemerkung 2.13

Nun müssen also alle Punkte aus (ip) und (bp_N) durchgegangen und nach deren Koordinaten im Gitter sortiert werden. Im Fall 1 ist die Liste (ip) bereits richtig sortiert. Das liegt daran, dass keine inneren Punkte hinzukommen können und die Punkte in Phase 1 richtig sortiert in die Liste (ip) eingetragen wurden. Im Fall 2 werden die Punkte aus (ip) und (bp_N) aus technischen Gründen in eine neue Liste $(merged)$ zusammengelegt. Die Nummer des Punktes wird in dem Wert "Gitterpunktnummer" abgelegt und sollte nicht mit dem Wert verwechselt werden, der die Position in dem Vektor, in dem alle Punkte gespeichert werden, beschreibt. Nachdem die Nummerierung abgeschlossen ist, iteriert der Gittergenerator nacheinander über die Listen (ip) und (bp_N) (Fall 1) bzw. über die Liste $(merged)$ (Fall 2). In jeder Iteration wird der Koeffizientenvektor α (innere Gitterpunkte) bzw. β (Neumann-Randpunkte) berechnet. Da die Matrix A dünnbesetzt ist, lohnt es sich, nur die Nicht-Null-Einträge der Matrix in der Form

(Zeilennummer, Spaltennummer, Wert)

in eine Datei zu schreiben. Null-Einträge werden nicht abgespeichert. Die Zeilen- und Spaltennummern ergeben sich aus der Gitterpunktnummer des aktuellen Punktes und den Gitterpunktnummern der Nachbarpunkte. Die Koordinaten der Punkte werden in eine separate Datei abgelegt. Die Ausgabe des Vektors b hängt davon ab, ob die Funktionen f, g_D und g_N aus Def. 1.1(b) von t abhängig sind oder nicht. Sofern sie von t unabhängig sind, lässt sich der Vektor b explizit ausrechnen und in Form eines Spaltenvektors ausgeben. Andernfalls wird eine große Tabelle mit allen Daten ausgegeben, die man für die Berechnung des Vektors b in einem externen Programm für jeden Zeitschritt t braucht. Das wären im einzelnen

- die Gitterpunktnummer,
- die Koordinaten des Gitterpunktes,
- die Komponenten des Koeffizientenvektors α bzw. β und
- die Koordinaten der Nachbarn "east", "west", "north" und "south", sofern der entsprechende Wert von α_i bzw. β_i für ein $i \in \{e, w, n, s\}$ ungleich Null ist.

Im externen Programm müssen dann nur noch die Funktionen f, g_D und g_N in jedem Zeitschritt t ausgewertet und mit den entsprechenden Koeffizienten aus der Tabelle verrechnet werden.

3.2.5 Einstellungen für den Anwender – ein Überblick

Dieser Abschnitt soll lediglich eine Übersicht der Parameter geben, auf die der Anwender Einfluss nehmen kann. Alle wichtigen Einstellungen finden sich in der folgenden Tabelle. Nicht aufgeführt sind die Funktionen F und deren partielle Ableitungen $\frac{\partial F}{\partial x}$ und $\frac{\partial F}{\partial y}$ aus Phase 2, da sie nicht einfach durch Setzen eines Parameters konstruiert werden.

Parameter	Beschreibung
$x_{lr} \in \mathbb{R}_+$	x-Koordinate des Eckpunktes $(x_{lr}, 0)$
$y_{ul} \in \mathbb{R}_+$	y-Koordinate des Eckpunktes $(0, y_{ul})$
$N_x \in \mathbb{N}$	Anzahl der Teilintervalle in x-Richtung
$N_y \in \mathbb{N}$	Anzahl der Teilintervalle in y-Richtung
boundDefault $\in \{1,2\}^4$	Legt für jede Seite des Rechtecks in Phase 1 den Randtyp ("Dirichlet" oder "Neumann") fest
credit $\in \mathbb{N}$	Dieser Wert gibt an, wie oft der Nachbarsuchalgorithmus jeweils den "east"-, "west"-, "north"- und "south"-Nachbarpunkt auswählen kann
strategy $\in \{1, \ldots, 9\}$	Wahl der Strategie
doBetatest $\in \{true, false\}$	Entscheidet darüber, ob der Betatest durchgeführt wird.
hThresholdDelDistNeighbours $\in \mathbb{R}$	Diese Zahl gibt an, welchen Wert die größte Schrittweite $\max\{h_x, h_y\}$ haben darf, bis zu der weiter entfernte Punkte **nicht** von vornherein ausgeschlossen werden. Weiter entfernte Punkte sind alle Kandidaten mit Abstand $> \frac{3}{2}\max\{h_x, h_y\}$ zum Neumann-Randpunkt.
deleteOldNeighbours $\in \{true, false\}$	Diese Option legt fest, ob bereits vorhandene Verweise auf Nachbarpunkte überschrieben werden sollen.
useAltNumbering $\in \{true, false\}$	Damit bestimmt der Anwender, welche der beiden in Abschnitt 3.2.4 erwähnten Nummerierungsansätze verwendet werden sollen.
timeDependent $\in \{true, false\}$	Gibt an, ob mindestens eine der Funktionen f, g_D und g_N von t abhängig sind.

Tabelle 3.2: Einstellungsmöglichkeiten des Anwenders

Kapitel 4

Anwendungen und Bewertung

4.1 Anwendungsbeispiele

Der Schwerpunkt dieser Arbeit liegt auf der Wahl von Gebieten, die über ein Rechteck oder einen Kreis hinausgehen und einen Neumann-Rand besitzen. Auf allen ausgewählten Gebieten mussten sich die insgesamt neun Strategien auf verschieden feinen Gittern beweisen. In den meisten Fällen lässt sich die Lösung nur numerisch berechnen. Ohne das Vorhandensein einer Referenzlösung lassen sich kaum präzise Aussagen über die Güte der errechneten Lösungen machen. Um dennoch zumindest heuristische Aussagen machen zu können, wurden im Vorfeld speziell das Beispiel 4.1 und Varianten davon (zum Beispiel andere Neumann-Randbedingungen) sowohl mit dem Gittergenerator als auch mit dem kommerziellen "Finite Elemente"-Löser "Comsol Multiphysics" behandelt und die Lösungen anhand charakteristischer Werte wie dem (absoluten) Maximum der Lösung verglichen. Dabei wurde festgestellt, dass die Lösungen qualitativ übereinstimmen. Je nach Gitterfeinheit und Wahl der Strategie waren die Maximalwerte der Lösungen auf einige Dezimalstellen genau identisch. Die übrigen im Folgenden aufgeführten Beispiele wurden nur mit dem Gittergenerator behandelt. Hier wurden weitere Kriterien zur Hilfe genommen, die sich aus der grafischen Darstellung der numerischen Lösungen ergeben:

- Sieht die numerische Lösung "glatt" aus, wie man es von der Lösung der (stationären) Wärmeleitungsgleichung erwarten würde?

- Verhält sich die Lösung insbesondere am Neumann-Rand wie man es aus physikalischen Gründen erwarten würde? Beispielsweise bedeutet bei der (stationären) Wärmeleitungsgleichung die homogene Neumann-Bedingung $\frac{\partial u}{\partial \nu} = 0$ auf Γ_N physikalisch, dass Γ_N "perfekt isoliert" ist, d.h. an dieser Stelle entweicht keine Wärme und es kommt auch keine von außen hinzu.

Sämtliche Beispiele wurden so gewählt, dass nach Bemerkung 2.11 nur ein lineares Gleichungssystem gelöst werden muss. Zur Weiterverarbeitung der Textdateien des Gittergenerators, speziell zur Lösung der linearen Gleichungssysteme und für die grafische Darstellung

der numerischen Lösung, wurde das Computeralgebrasystem "Maple" verwendet. Die entsprechenden Worksheets sind auf der beigelegten CD-ROM gespeichert.
Relevante Parameter und Vergleichskriterien werden unter den Plots der numerischen Lösung auf dem jeweils vorgegebenen Gitter in der Form

$$(N_x, N_y, \text{strategy}, \max u, \text{cond}_\infty(A))$$

angezeigt, wobei A die Matrix A aus (2.21) sei. Für die Schrittweiten gilt, sofern in den Beispielen nicht anders angegeben:

$$(h_x, h_y) = \left(\frac{12}{N_x}, \frac{8}{N_y}\right)$$

Da die Nummerierung der Gitterpunkte keinen Einfluss auf die Lösung des linearen Gleichungssystems hat, wurde stets die erste Nummerierungsmethode verwendet, vgl. Abschnitt 3.2.3. Sofern nicht anders erwähnt, wurden weiterhin folgende Optionen verwendet:

credit	$= 2$
doBetatest	$= true$
hThresholdDelDistNeighbours	$= 0$
deleteOldNeighbours	$= false$
timeDependent	$= false$

Der Gittergenerator wurde auf einem Rechner mit Windows 7 Professional (32 bit) und der IDE "CodeBlocks" in der Version 8.02 (mit "MinGW") programmiert und kompiliert. Die vom Gittergenerator erstellten Textdateien können je nach Betriebssystem und Compiler variieren, insbesondere könnten im Extremfall zum Beispiel aufgrund von Rechenungenauigkeiten andere Nachbarpunktkonstellationen ausgewählt werden oder es könnte in manchen Fällen – je nach Compiler – "Infinity" ausgegeben werden.

4.1.1 Die 2-dim. Wärmeleitungsgleichung (stationär)

Man betrachte zunächst die 2-dim. Wärmeleitungsgleichung. Um kein System gewöhnlicher Differentialgleichungen lösen zu müssen, wird hier der stationäre Fall der Gleichung betrachtet. Damit reicht es laut Bemerkung 2.11, ein Gleichungssystem zu lösen. Die Differentialgleichung mit Randwerten sei für verschiedene Gebiete $\Omega \subset [0, \infty[^2$ mit Rand $\partial\Omega = \Gamma_D \dot\cup \Gamma_N$ definiert durch:

$$\begin{aligned} -\Delta u &= 2 \text{ auf } \Omega \\ u &= 0 \text{ auf } \Gamma_D \\ \frac{\partial u}{\partial \nu} &= 0 \text{ auf } \Gamma_N \end{aligned} \tag{4.1}$$

Beispiel 4.1
Das erste Gebiet ist ein Querschnitt eines Behälters, der eine Röhre mit Radius $r = \sqrt{3}$ enthält, vgl. Abbildung 4.1. Γ_N sei der Rand des inneren Kreises, Γ_D der äußere Rand.

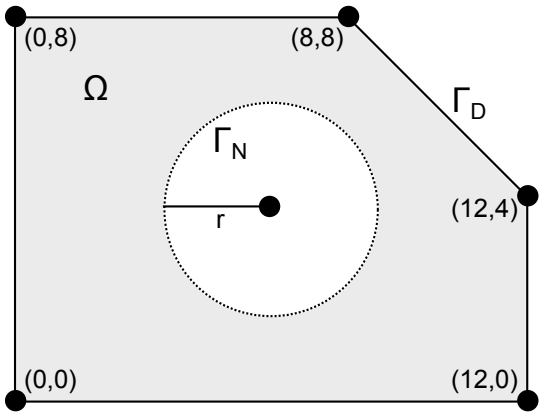


Abbildung 4.1: Unregelmäßiges Gebiet Nr. 1: Behälterquerschnitt

Die folgenden Abbildungen stellen die numerische Lösung auf dem jeweiligen Gitter grafisch dar. Zum Vergleich: Der Finite-Elemente-Löser lieferte als Maximalwert der Lösung

$$\max u = 9.526701$$

sowohl bei einer gröberen als auch bei einer recht feinen Diskretisierung.

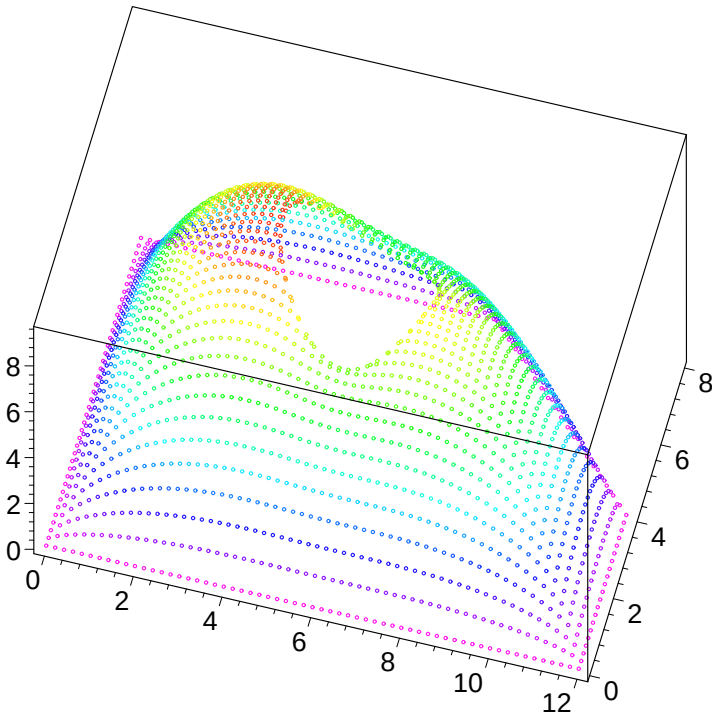


Abbildung 4.2: Bsp. 4.1, (60, 40, NV_NEAREST, 9.517956201, 6424.871609)

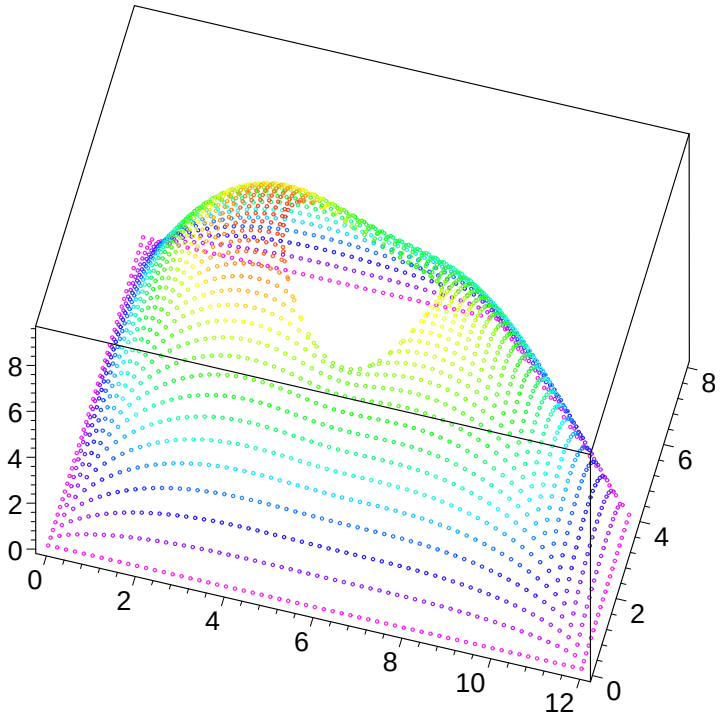


Abbildung 4.3: Bsp. 4.1, (60, 40, NV_NEAREST_MIN_DIST, 9.517708740, 6293.745710)

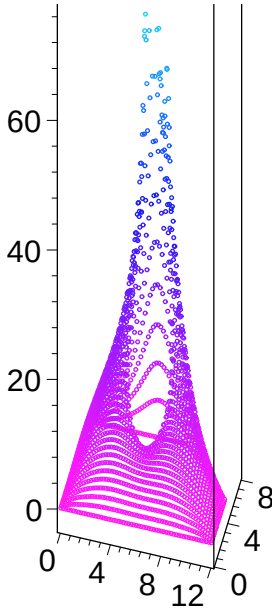


Abbildung 4.4: Bsp. 4.1, (60, 40, NV_PREFER_BOUNDARY, 181.8044281, 92552887.06)

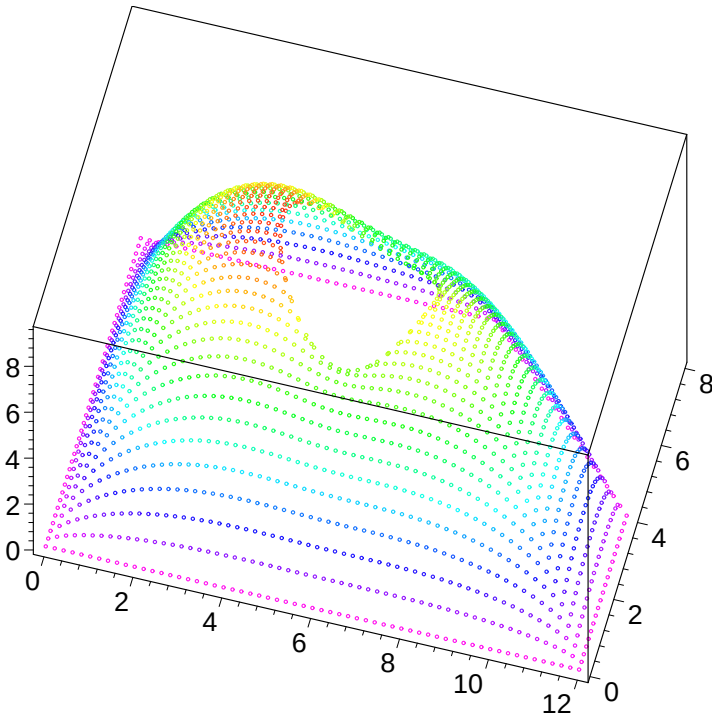


Abbildung 4.5: Bsp. 4.1, (60, 40, V_NEAREST, 9.531970857, 6440.016424)

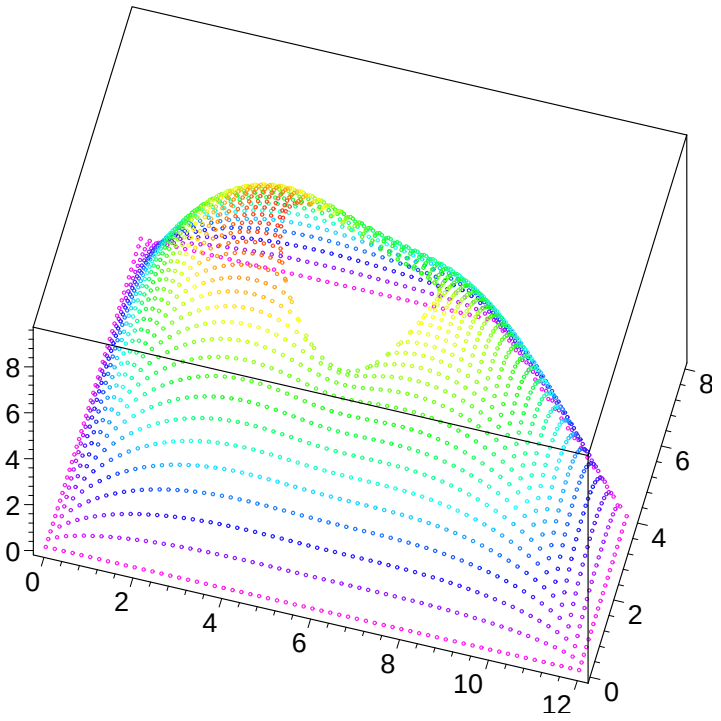


Abbildung 4.6: Bsp. 4.1, (60, 40, V_NEAREST_MIN_DIST, 9.531720991, 6311.822052)

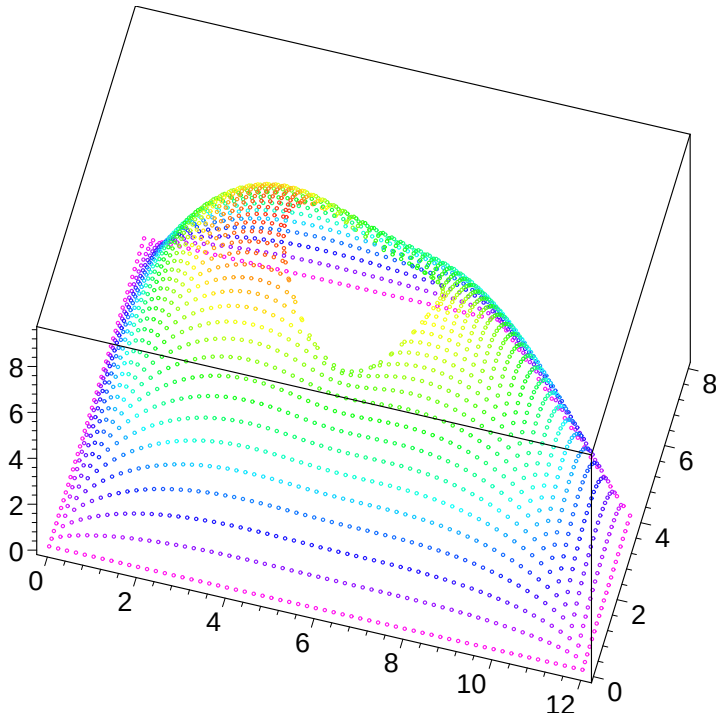


Abbildung 4.7: Bsp. 4.1, (60, 40, V_COND, 9.544484642, 4348.894677)

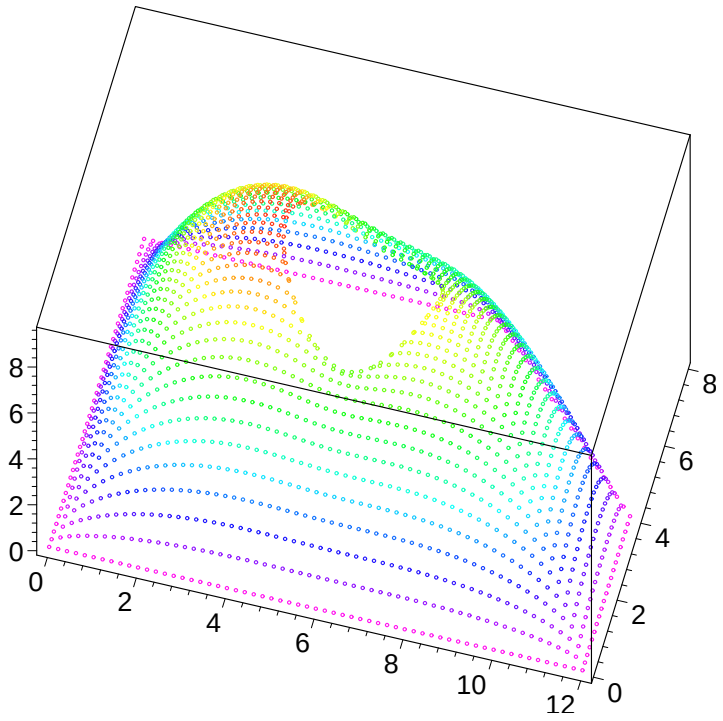


Abbildung 4.8: Bsp. 4.1, (60, 40, V_COND_W, 9.544659546, 4351.764360)

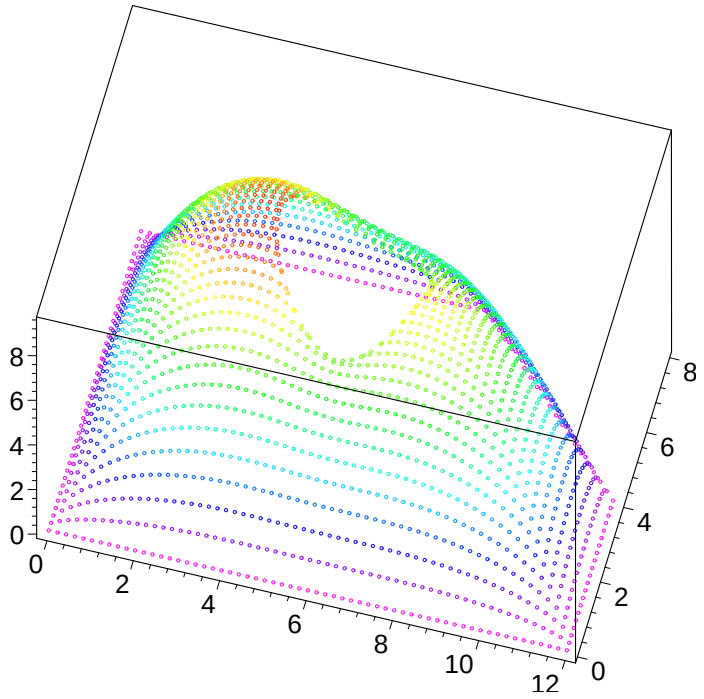


Abbildung 4.9: Bsp. 4.1, (60, 40, V_PREFER_INNER_NEAR_NEUMANN, 9.542473873, 11849.94285)

Man sieht, dass die meisten Plots ähnlich sind. Die Strategie "NV_PREFER_BOUNDARY" erweist sich als untauglich; der zugehörige Plot wurde nach oben hin abgeschnitten. Die Strategie "NV_OPTIMIZE_DIRECTIONS" generierte nicht einmal eine gültige Matrix A. Dies war zu erwarten, da die Matrix (2.18) sehr wahrscheinlich in mehreren Punkten singulär wird. Die Strategien "V_COND" und "V_COND_W" weisen den niedrigsten Wert für $\text{cond}_\infty(A)$ auf. Möglicherweise hat die Kondition der Matrix (2.18) Auswirkungen darauf.

(N_x, N_y)	$\dim A$	Strategie	$\max u$	$\text{cond}_\infty(A)$
$(60, 40)$	1938	NV_NEAREST	9.517956201	6424.871609
$(60, 40)$	1938	NV_NEAREST_MIN_DIST	9.517708740	6293.745710
$(60, 40)$	1938	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(60, 40)$	1938	NV_PREFER_BOUNDARY	181.8044281	92552887.06
$(60, 40)$	1938	V_NEAREST	9.531970857	6440.016424
$(60, 40)$	1938	V_NEAREST_MIN_DIST	9.531720991	6311.822052
$(60, 40)$	1938	V_COND	9.544484642	4348.894677
$(60, 40)$	1938	V_COND_W	9.544659546	4351.764360
$(60, 40)$	1938	V_PREFER_INNER_NEAR_NEUMANN	9.542473873	11849.94285
n/a	n/a	*Finite-Elemente-Methode*	9.526701	n/a

Tabelle 4.1: Zusammenfassung für Beispiel 4.1 mit $(N_x, N_y) = (60, 40)$

Bei einer höheren Auflösung von $(N_x, N_y) = (120, 80)$ zeichnet sich dasselbe Bild ab. Abbildung 4.10 steht daher stellvertretend für diejenigen Strategien, die schon oben "gut" abgeschnitten haben. Auch hier versagen die Strategien "NV_PREFER_BOUNDARY" und "NV_OPTIMIZE_DIRECTIONS". Tabelle 4.2 fasst die interessanten Werte zusammen.

(N_x, N_y)	$\dim A$	Strategie	$\max u$	$\text{cond}_\infty(A)$
$(120, 80)$	7812	NV_NEAREST	9.526969713	97669.81714
$(120, 80)$	7812	NV_NEAREST_MIN_DIST	9.521039332	97269.19649
$(120, 80)$	7812	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(120, 80)$	7812	NV_PREFER_BOUNDARY	141.7146415	6221070363
$(120, 80)$	7812	V_NEAREST	9.529043814	97630.34926
$(120, 80)$	7812	V_NEAREST_MIN_DIST	9.521615218	556446.7997
$(120, 80)$	7812	V_COND	9.528178054	17332.22432
$(120, 80)$	7812	V_COND_W	9.528194363	17332.25399
$(120, 80)$	7812	V_PREFER_INNER_NEAR_NEUMANN	9.527370855	102577.6441
n/a	n/a	*Finite-Elemente-Methode*	9.526701	n/a

Tabelle 4.2: Zusammenfassung für Beispiel 4.1 mit $(N_x, N_y) = (120, 80)$

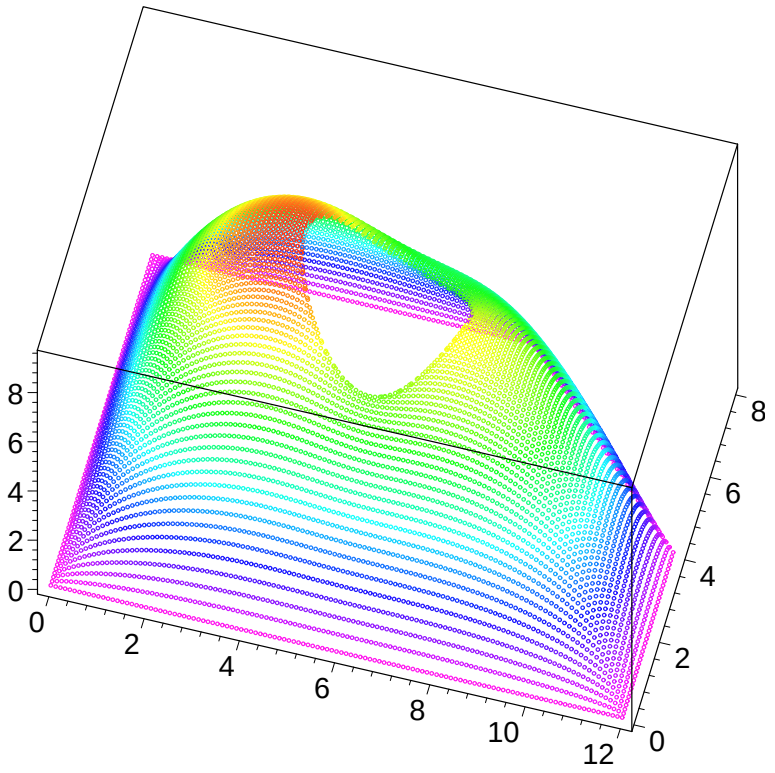


Abbildung 4.10: Bsp. 4.1, (120, 80, V_COND_W, 9.528194363, 17332.25399)

Beispiel 4.2
Man betrachte weiterhin die stationäre Wärmeleitungsgleichung (4.1), diesmal jedoch auf dem Gebiet aus Abbildung 4.11 mit $r = \frac{1}{\sqrt{2}}$.

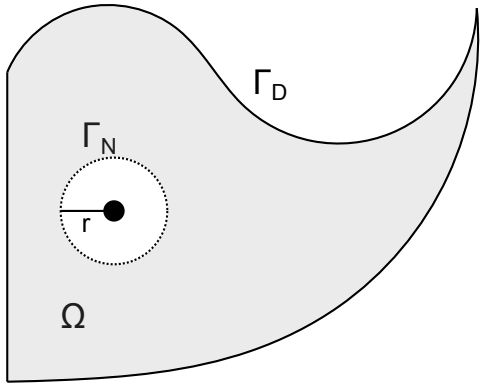


Abbildung 4.11: Unregelmäßiges Gebiet Nr. 2

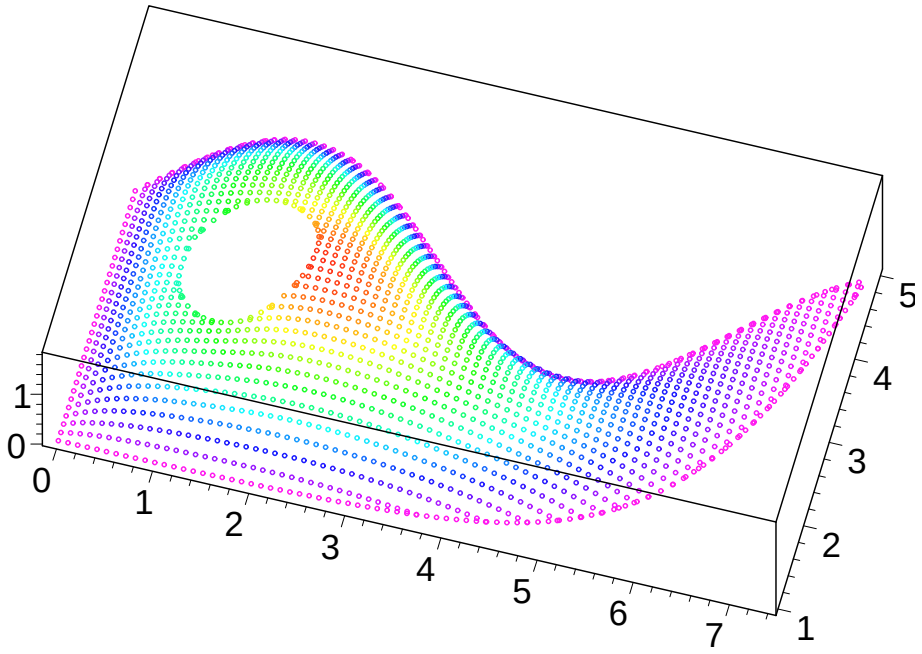


Abbildung 4.12: Bsp. 4.2, (120, 80, NV_PREFER_BOUNDARY, 1.810376454, 327261.9602)

Während diesmal die Strategien “NV_NEAREST”, “NV_NEAREST_MIN_DIST” und “NV_OPTIMIZE_DIRECTION” keine gültigen Matrizen A liefern, überrascht “NV_PREFER_BOUNDARY” mit einem brauchbaren Ergebnis. Alle anderen Strategien haben als Ergebnis einen ähnlichen Plot wie Abbildung 4.12. Daher werden die interessanten Werte lediglich in einer Tabelle abgetragen. In diesem Beispiel führen die Strategien “V_COND” und

"V_COND_W" nicht zu einer wesentlich niedrigen Kondition von A, verschlechtern sie aber auch nicht. Interessant ist auch, dass die beiden Strategien "V_COND" und "V_COND_W" gleiche Werte für $\max u$ und $\mathrm{cond}_\infty(A)$ liefern. Das kann darauf hindeuten, dass die Gewichtung noch nicht optimal ist, da sie – zumindest in diesem Fall – keinen wesentlichen Einfluss hat.

(N_x, N_y)	$\dim A$	Strategie	$\max u$	$\mathrm{cond}_\infty(A)$
$(120, 80)$	1729	NV_NEAREST	n/a	n/a
$(120, 80)$	1729	NV_NEAREST_MIN_DIST	n/a	n/a
$(120, 80)$	1729	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(120, 80)$	1729	NV_PREFER_BOUNDARY	1.810376454	327261.9602
$(120, 80)$	1729	V_NEAREST	1.812350050	238500.3293
$(120, 80)$	1729	V_NEAREST_MIN_DIST	1.812350050	238629.3592
$(120, 80)$	1729	V_COND	1.812581738	237231.5956
$(120, 80)$	1729	V_COND_W	1.812581738	237231.5956
$(120, 80)$	1729	V_PREFER_INNER_NEAR_NEUMANN	1.812471828	236804.7422

Tabelle 4.3: Zusammenfassung für Beispiel 4.2 mit $(N_x, N_y) = (120, 80)$

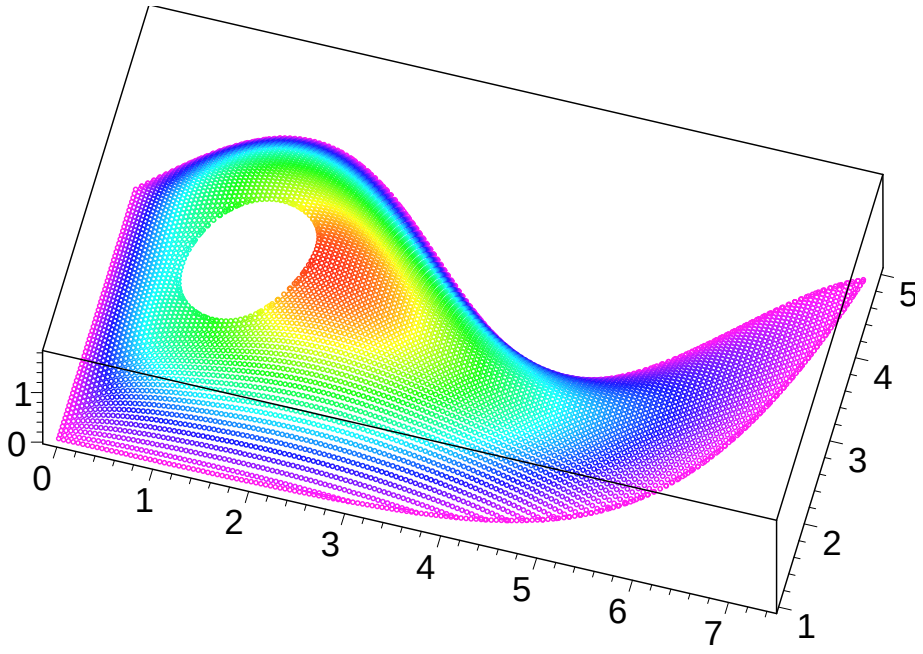


Abbildung 4.13: Bsp. 4.2, (240, 160, NV_NEAREST_MIN_DIST, 1.807987533, 1721491.379)

Wieder wird das Gitter verfeinert, diesmal auf $(N_x, N_y) = (240, 160)$. Die grafischen Ergebnisse verändern sich nicht wesentlich. Allerdings liefert diesmal "NV_NEAREST_MIN_DIST" ein Ergebnis, vgl. Abb. 4.13. Es scheint sich langsam heraus zu kristallisieren, dass das Abstandskriterium an sich nicht schlecht ist, aber ohne den Regularitätstest es Glückssache ist,

ob ein brauchbares Resultat herauskommt oder nicht. Ausnahmsweise liefert “V_COND” eine sehr hohe Kondition und das, obwohl sich die Werte für “V_COND” und “V_COND_W” für $(N_x, N_y) = (120, 80)$ garnicht unterscheiden; es scheint ein Sonderfall zu sein.

(N_x, N_y)	$\dim A$	Strategie	$\max u$	$\text{cond}_\infty(A)$
$(240, 160)$	6904	NV_NEAREST	n/a	n/a
$(240, 160)$	6904	NV_NEAREST_MIN_DIST	1.807987533	1721491.379
$(240, 160)$	6904	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(240, 160)$	6904	NV_PREFER_BOUNDARY	1.813259854	1796413.254
$(240, 160)$	6904	V_NEAREST	1.808174696	1719994.877
$(240, 160)$	6904	V_NEAREST_MIN_DIST	1.807167248	1860965.304
$(240, 160)$	6904	V_COND	1.807498105	2084663.071
$(240, 160)$	6904	V_COND_W	1.809328748	1582317.794
$(240, 160)$	6904	V_PREFER_INNER_NEAR_NEUMANN	1.810058761	1556517.821

Tabelle 4.4: Zusammenfassung für Beispiel 4.2 mit $(N_x, N_y) = (240, 160)$

Nun soll der Einfluss der Option “hThresholdDelDistNeighbours” untersucht werden. Bisher wurden weiter entfernte Nachbarn immer gelöscht, da der Wert auf Null gesetzt war. Zum Vergleich wird dieser Wert nun so hoch gesetzt, dass Kandidaten nicht von vornherein wegen ihrer Entfernung ausgeschlossen werden. Dabei werden nur die drei Strategien berücksichtigt, die diese Option bieten.

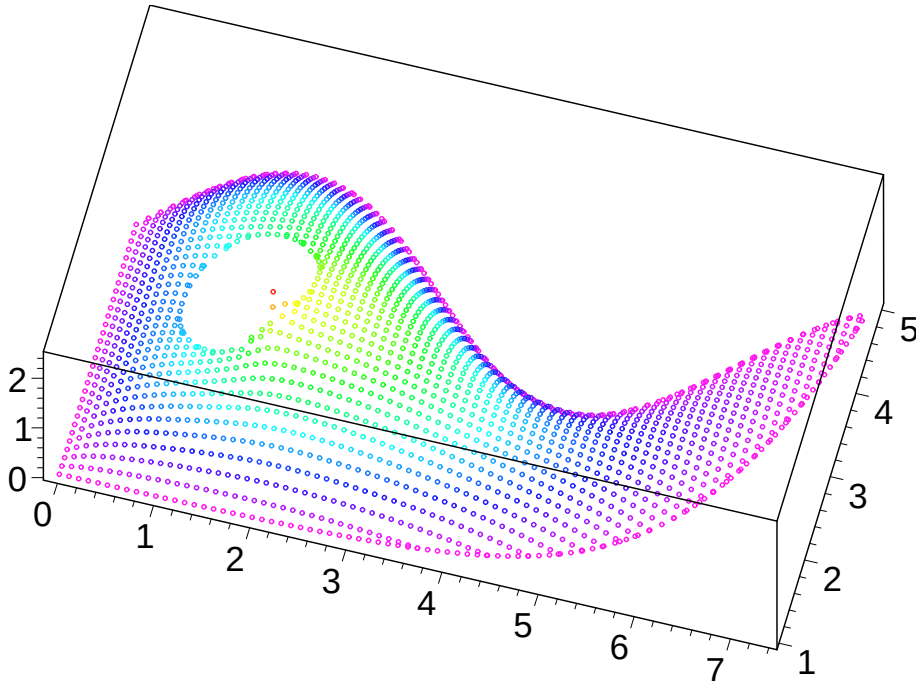


Abbildung 4.14: Bsp. 4.2, (120, 80, V_COND, 2.4871065, 2089576.7)

Man vergleiche die Abbildung 4.14 mit 4.12. Lässt man weiter entfernte Nachbarn zu, kann es vor allem bei den Strategien "V_COND" und "V_COND_W" vorkommen, dass die Ergebnisse merklich schlechter werden. Es liegt nahe, dass dieses Phänomen vor allem bei gröberen Gittern auftritt. Bei feineren Gittern sind die Punkte so nahe beieinander, dass die Entfernung keine Rolle mehr spielt. Es wäre sogar möglich, dass es bei zunehmend feineren Gittern empfehlenswert wäre, das Ausschließen von "weit entfernten" Nachbarn abzuschalten. In Tabelle 4.5 sind weitere Werte zu finden. Eine "1" in der R-Spalte bedeutet, dass zu weit entfernte Nachbarn gelöscht werden und eine "0" entsprechend, dass sie erhalten bleiben.

(N_x, N_y)	$\dim A$	Strategie	R	$\max u$	$\text{cond}_\infty(A)$
$(120, 80)$	1729	V_COND	1	1.8125817	237231.59
$(120, 80)$	1729	V_COND	0	2.4871065	2089576.7
$(120, 80)$	1729	V_COND_W	1	1.8125817	237231.59
$(120, 80)$	1729	V_COND_W	0	1.9560589	1214070.4
$(120, 80)$	1729	V_PREFER_INNER_NEAR_NEUMANN	1	1.8124718	236804.74
$(120, 80)$	1729	V_PREFER_INNER_NEAR_NEUMANN	0	1.8094723	263049.10
$(240, 160)$	6904	V_COND	1	1.8074981	2084663.0
$(240, 160)$	6904	V_COND	0	1.8153815	2209856.3
$(240, 160)$	6904	V_COND_W	1	1.8093287	1582317.7
$(240, 160)$	6904	V_COND_W	0	1.8143220	1572459.6
$(240, 160)$	6904	V_PREFER_INNER_NEAR_NEUMANN	1	1.8100587	1556517.8
$(240, 160)$	6904	V_PREFER_INNER_NEAR_NEUMANN	0	1.8092666	26813823

Tabelle 4.5: Auswirkungen von "hThresholdDelDistNeighbours" für Beispiel 4.2

Beispiel 4.3
Diesmal wird nicht nur ein weiteres Gebiet betrachtet (vgl. Abbildung 4.15; der Radius des Kreises r ist $\frac{1}{\sqrt{10}}$), auch die rechten Seiten in (4.1) werden verändert. Die neue Gleichung lautet:

$$\begin{aligned} -\Delta u &= 0 \ \textit{auf}\ \Omega \\ u &= \frac{1}{2} \ \textit{auf}\ \Gamma_D \\ \frac{\partial u}{\partial \nu} &= 1 \ \textit{auf}\ \Gamma_N \end{aligned} \tag{4.2}$$

Versuchsweise wird die Option "doBetatest" ausgeschaltet. Das Resultat: Zwei der "validated"-Strategien liefern nun eine singuläre Matrix. Der Betatest sollte also nach Möglichkeit immer angeschaltet bleiben! Abbildung 4.16 ist qualitativ repräsentativ für alle Strategien, die eine reguläre Matrix A erzeugen. Erstaunlicherweise sieht das Ergebnis für "NV_PREFER_BOUNDARY" trotz der sehr hohen Kondition noch relativ gut aus. Die Ergebnisse der anderen Strategien deuten jedoch auf eine etwas abweichende Lösung hin und liefern vermutlich das bessere Ergebnis.

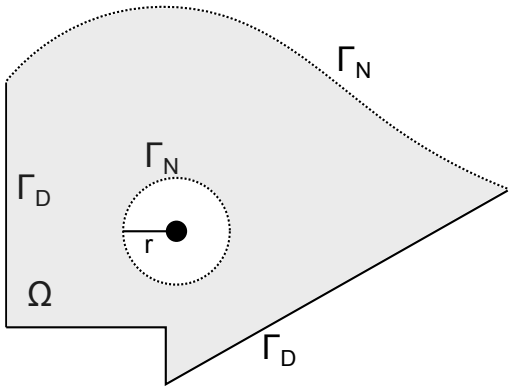


Abbildung 4.15: Unregelmäßiges Gebiet Nr. 3

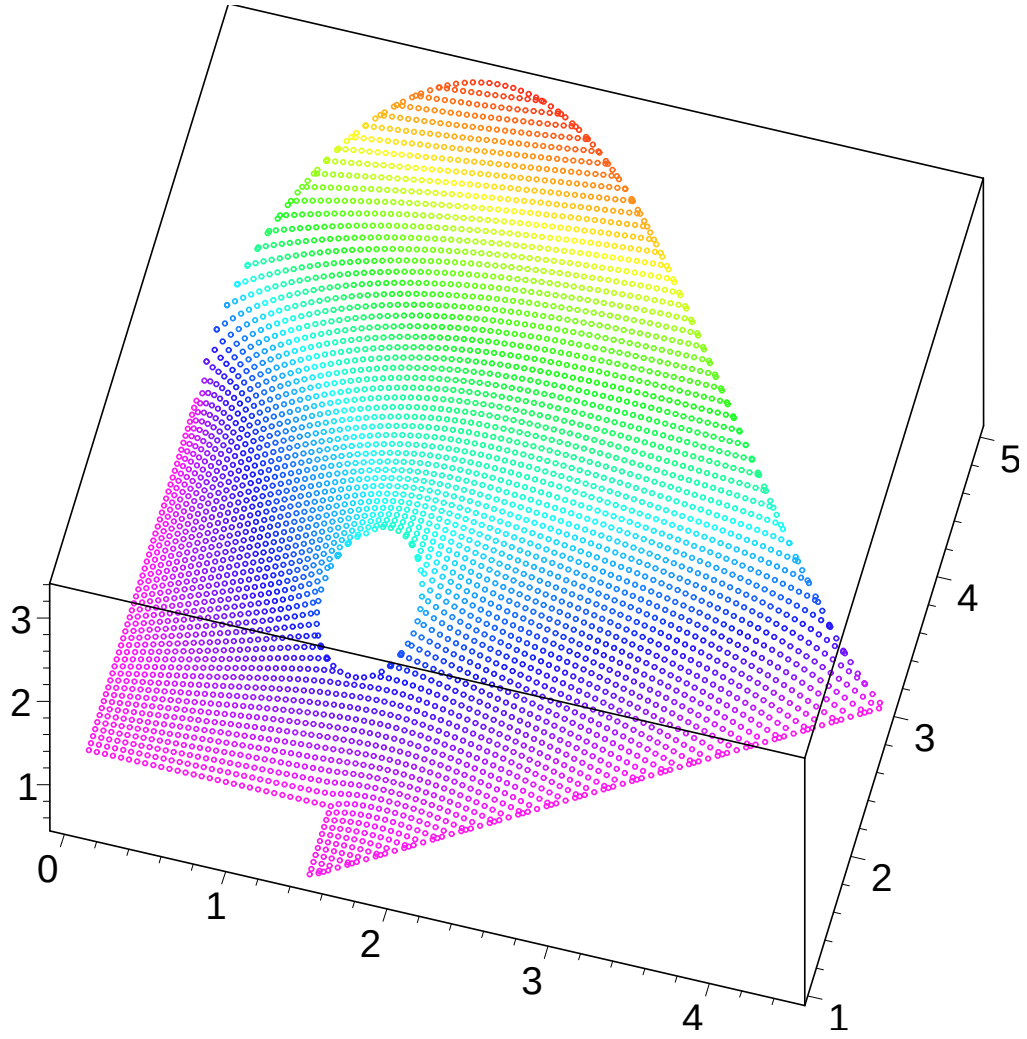


Abbildung 4.16: Bsp. 4.3, (240, 160, NV_PREFER_BOUNDARY, 3.355340028, 0.271451e13)

(N_x, N_y)	$\dim A$	Strategie	$\max u$	$\text{cond}_\infty(A)$
$(240, 160)$	4381	NV_NEAREST	n/a	n/a
$(240, 160)$	4381	NV_NEAREST_MIN_DIST	n/a	n/a
$(240, 160)$	4381	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(240, 160)$	4381	NV_PREFER_BOUNDARY	3.355340028	0.271451e13
$(240, 160)$	4381	V_NEAREST	n/a	n/a
$(240, 160)$	4381	V_NEAREST_MIN_DIST	3.411945448	301834.0570
$(240, 160)$	4381	V_COND	3.411691778	259630.9265
$(240, 160)$	4381	V_COND_W	3.411500160	253473.6463
$(240, 160)$	4381	V_PREFER_INNER_NEAR_NEUMANN	n/a	n/a

Tabelle 4.6: Zusammenfassung für Beispiel 4.3 mit $(N_x, N_y) = (240, 160)$

4.1.2 Die 1-dim. Wärmeleitungsgleichung

Bisher wurde lediglich eine Ortsdiskretisierung durchgeführt. Jetzt soll versucht werden, die 1-dim. Wärmeleitungsgleichung vollständig zu lösen. Das hier betrachtete ARWP lautet:

$$\begin{aligned} \partial_t u(x,t) - \partial_{xx} u(x,t) &= 0 \qquad \text{für } 0 < x < 1, 0 < t < 1 \\ u(0,t) = u(1,t) &= 0 \qquad \text{für } 0 < t < 1 \\ u(x,0) &= \sin(\pi x) \text{ für } 0 \leq x \leq 1. \end{aligned} \tag{4.3}$$

Man kann zeigen, dass die exakte Lösung

$$u(x,t) = \exp^{-\pi^2 t} \sin(\pi x) \tag{4.4}$$

lautet. Damit lässt sich nun die Güte der numerischen Lösung (genauer) messen.
Für den Gittergenerator muss man das Problem etwas umformulieren. Dabei wird die t-Komponente als weitere Ortskomponente in y-Richtung aufgefasst. Das ist auch der Grund für die Schreibweise $u(x,t)$ anstelle von $u(t,x)$. Als Gebiet wählt man $\Omega =]0,1[^2$. Die Anfangsbedingung kann in diesem Fall leicht in eine Dirichlet-Randbedingung umgeformt werden. Für $t = 1$ und $0 < x < 1$ wurden in (4.3) keine Werte vorgegeben. Der Gittergenerator braucht jedoch an dieser Stelle einen Rand mit Randbedingung. Eine Dirichlet-Randbedingung wäre unsinnig. Die Idee ist hier, die homogene Neumann-Randbedingung

$$\frac{\partial u}{\partial \nu}(x,1) = 0 \text{ für } 0 < x < 1$$

zu verwenden. Der Grund dafür ist die physikalische Interpretation der Neumann-Randbedingung bei der Wärmeleitungsgleichung: Ist diese auf den Wert Null vorgeschrieben, so ist der Rand "perfekt isoliert", d.h. am Rand entweicht keine Wärme und es kommt auch keine von außen hinzu. Die Normale zeigt hier in y-, also in t-Richtung. Folglich besteht die Hoffnung, dass durch diese Neumann-Randbedingung die gleiche zeitliche Entwicklung gewährleistet ist wie in (4.3), da keine Einflüsse von "außen" vorhanden sind. Insgesamt

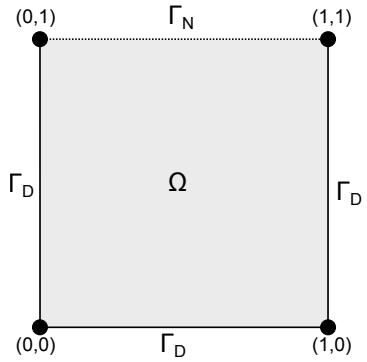


Abbildung 4.17: Einheitsquadrat

erhält man also das Gebiet wie in Abbildung 4.17 dargestellt, wobei die Punkte $(0,1)$ und $(1,1)$ jeweils zum Dirichlet-Rand gehören.
Das RWP lautet schließlich:

$$\begin{aligned} \partial_t u - \partial_{xx} u &= 0 && \text{auf } \Omega \\ u &= \sin(\pi x) && \text{auf } \Gamma_D \\ \frac{\partial u}{\partial \nu} &= 0 && \text{auf } \Gamma_N \end{aligned} \tag{4.5}$$

Die exakte Lösung ist in Abbildung 4.18 dargestellt. Anstelle der numerischen Lösungen, die auch hier nach Bemerkung 2.11 durch das Lösen eines LGS berechnet werden, wird nun der Fehler im Betrag in den Gitterpunkten veranschaulicht, also

$$|u(x,t) - \tilde{u}(x,t)|, (x,t) \in \bar{G},$$

wobei $\tilde{u}(x,t)$ die numerische Approximation von u im Gitterpunkt (x,t) des Gitters G sei. Für die Schrittweiten gilt hier: $(h_x, h_y) = (\frac{1}{N_x}, \frac{1}{N_y})$

(N_x, N_y)	$\dim A$	Strategie	$\max \lvert u - \tilde{u} \rvert$	$\text{cond}_\infty(A)$
$(60, 120)$	7080	NV_NEAREST	n/a	n/a
$(60, 120)$	7080	NV_NEAREST_MIN_DIST	n/a	n/a
$(60, 120)$	7080	NV_OPTIMIZE_DIRECTION	n/a	n/a
$(60, 120)$	7080	NV_PREFER_BOUNDARY	n/a	n/a
$(60, 120)$	7080	V_NEAREST	0.000497482	1848.537408
$(60, 120)$	7080	V_NEAREST_MIN_DIST	0.000497482	1848.550150
$(60, 120)$	7080	V_COND	0.000497482	1827.303127
$(60, 120)$	7080	V_COND_W	0.000497482	1827.372124
$(60, 120)$	7080	V_PREFER_INNER_NEAR_NEUMANN	0.000497482	2442.143981

Tabelle 4.7: Zusammenfassung für 1-dim. Wärmeleitungsgleichung

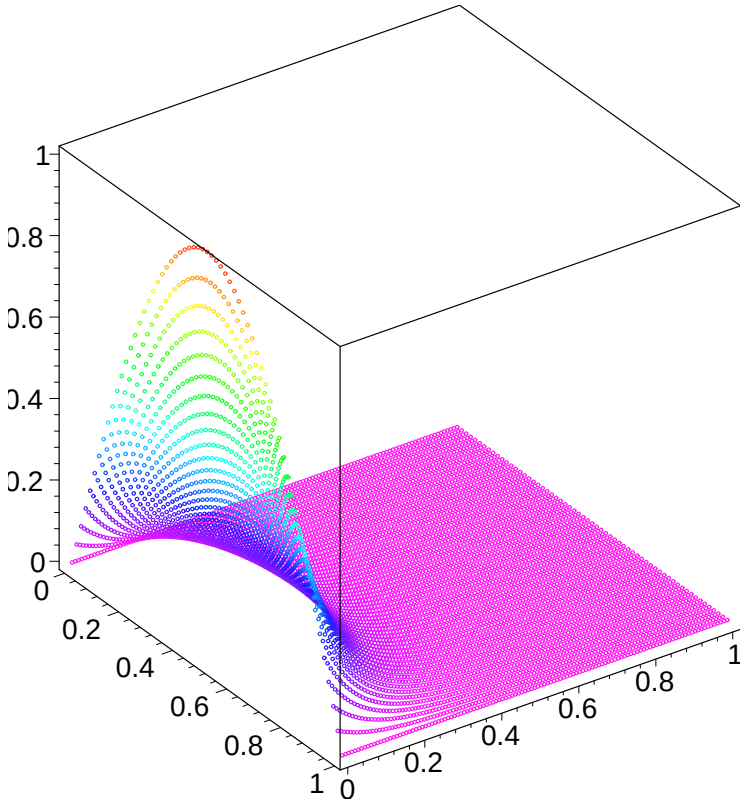


Abbildung 4.18: Wärmeleitungsgleichung (1D), u (exakt)

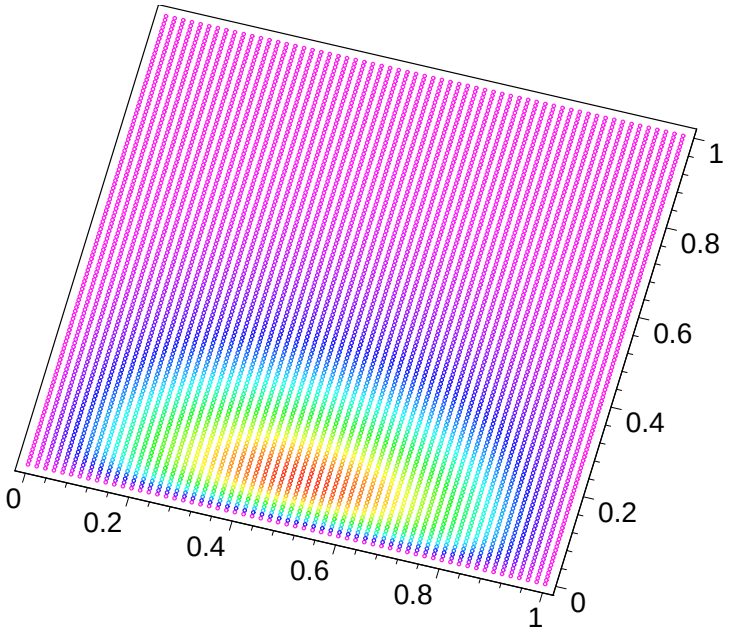


Abbildung 4.19: Wärmeleitungsgleichung (1D), $|u - \tilde{u}|$

Selbst bei diesem relativ einfachen Gebiet versagen alle "non-validated"-Strategien. Alle anderen liefern praktisch identische Ergebnisse. Die Strategien liefern ähnliche Werte für $\text{cond}_\infty(A)$. Insgesamt lässt sich die 1-dim. Wärmeleitungsgleichung relativ gut approximieren.

4.2 Abschließende Bewertung

Mit dieser Arbeit sollte untersucht werden, wie weit man in der Praxis mit der auf recht einfachen Ideen basierenden Finite-Differenzen-Methode kommt. Normalerweise wird diese Methode bei vergleichsweise angenehmen Problemstellungen angewendet, bei denen es zum Beispiel gar keine Neumann-Randpunkte gibt. In diesem Fall gibt es theoretische Resultate, die die Konvergenz der Finite-Differenzen-Methode unter gewissen Regularitätsanforderungen zeigen. Doch selbst hier wird es schnell komplizierter, sobald man auf unregelmäßige Gebiete stößt. Gerade diese sind aber in Anwendungen oftmals unabdingbar, genauso wie Neumann-Randpunkte.
Diese Arbeit macht nochmal deutlich, dass eine Neumann-Randbedingung einen erheblichen Mehraufwand produziert. An den obigen Beispielen hat man jedoch sehen können, dass das Finite-Differenzen-Verfahren selbst bei unregelmäßigen Gittern und Neumann-Randbedingung relativ erfolgreich sein kann. Daher sollte diese Methode – zumindest aus praktischer Sicht – nicht sofort verworfen werden, wenn die Problemstellung etwas komplexer wird. Man hat allerdings auch gesehen, dass es eine Gratwanderung zwischen einer "guten" und einer "katastrophalen" Lösung ist. Der Schlüssel für mögliche "gute" Lösungen ist eine geschickte Wahl der Nachbarpunkte in den Neumann-Randpunkten. Von den neun vorgestellten Strategien zur Auswahl der Nachbarpunkte sind prinzipiell alle "validated"-Strategien, also jene, die den Regularitätstest durchführen, empfehlenswert. Man sollte darauf achten, dass auch der Betatest durchgeführt wird; ohne ihn können schnell Probleme auftreten. Letztendlich liegt es aber am Anwender, die für ein konkretes Beispiel optimalen Strategien inklusive Optionen auszuwählen. Genau das ist wohl aber auch das größte praktische Problem an diesem Gittergenerator bzw. an der Methode, die dahintersteht. Schließlich hätte man gerne eine Anwendung, die ein gegebenes Problem sicher löst.
Ein weiteres großes Hindernis sind die aus der Theorie benötigten Anforderungen an die Lösung der partiellen Differentialgleichung, speziell die Glattheitsanforderungen. Wenn man den Weg in Richtung mehr Realismus einschlägt, stößt man damit recht schnell an die Grenzen der Finiten Differenzen. In diesem Fall müsste man einen ganz anderen theoretischen Ansatz wählen und auf sogenannte "schwache Lösungen" und beispielsweise die "Finite-Elemente-Methode" zurückgreifen. Die Stichwörter werden hier nur erwähnt und sollen zur weiteren Recherche anregen. An dieser Stelle bleibt festzuhalten, dass man – zumindest den praktischen Resultaten nach zu urteilen – den realistischen Anwendungen auch mit der Finite-Differenzen-Methode ein Stückchen näher gekommen ist.

Anhang A

Inhalt der CD-ROM

Die beiliegende CD-ROM ist in neun Verzeichnisse unterteilt. Zu finden sind neben der Bachelorarbeit als PDF-Datei alle verwendeten Maple-Worksheets, der Quellcode des Gittergenerators, eine bereits kompilierte Version des Gittergenerators für Windows 7 (32 bit) und Linux (openSuse 11.2, 64 bit), Skripte zum Aufruf des Gittergenerators für die Beispiele aus Kapitel 4 und die .plt-Dateien für Gnuplot, um die vom Gittergenerator diskretisierten Gebiete zu visualisieren. Als Referenz ist ein .zip-Archiv mit den für die Grafiken und Tabellen in Kapitel 4 verwendeten, vom Gittergenerator erstellten Textdateien vorhanden. Eine Zusammenfassung findet sich in Tabelle A.1.

Bemerkungen zum Inhalt der CD-ROM:

- Beim Aufruf des Gittergenerators werden Dateien erstellt. Daher ist es wichtig, dass bei Ausführung des Gittergenerators bzw. einer .cmd- oder .sh-Datei Schreibzugriff besteht.
- Die Maple-Worksheets, die in den "example"-Ordnern zusammen mit den .cmd- bzw. .sh-Dateien zu finden sind, sind darauf angewiesen, dass vorher die .cmd- bzw. .sh-Datei ausgeführt worden ist, damit sich alle vom Gittergenerator erstellten Dateien in den richtigen Ordnern befinden. Alternativ muss man in den Maple-Worksheets die Verzeichnisse anpassen. Die Worksheets erstellen Dateien, so dass ein Schreibzugriff nötig ist.
- Die .plt-Dateien für Gnuplot müssen sich im gleichen Ordner befinden wie die vom Gittergenerator erstellten Textdateien, damit das Gebiet mit Gnuplot visualisiert werden kann. Eventuell ist also ein Kopieren bzw. Verschieben der .plt-Dateien nötig.
- Die komprimierten Textdateien in dem .zip-Archiv sind alle mit Windows 7 (32 bit) erstellt worden.

Ordner	Dateiname	Beschreibung
bachelorarbeit	bachelorarbeit.pdf example_textfiles.zip	Die Bachelorarbeit im PDF-Format Alle für die Beispiele aus Kapitel 4 erstellten Textdateien.
example_1	example1_laplace.cmd example1_laplace_linux.sh plot_laplace_60x40.mw plot_laplace_120x80.mw	Skripte zum Aufruf des Gittergenerators mit Parametern für das Beispiel 4.1 und zugehörige Maple-Worksheets zum Lösen der linearen Gleichungssysteme und für die grafische Darstellung.
example_2	example2_laplace.cmd example2_laplace_linux.sh plot_laplace_120x80.mw plot_laplace_240x160.mw plot_laplace_noRemoval.mw	Analog zu example_1 für Beispiel 4.2.
example_3	example3_laplace.cmd example3_laplace_linux.sh plot_laplace_noBeta_240x160.mw	Analog zu example_1 für Beispiel 4.3.
example_4	example4_wlg_1dim.cmd example4_wlg_1dim_linux.sh plot_wlg_1dim_60x120.mw	Analog zu example_1 für die 1-dim. Wärmeleitungsgleichung aus Abschnitt 4.1.2.
gittergen_binary	gittergen_v10.exe gittergen_v10.out	vorkompilierter Gittergenerator für Windows 7 (32 bit) und Linux (openSuse 11.2, 64 bit)
gittergen_source	example_functions.cpp example_functions.h main.cpp	Quellcode des Gittergenerators, aufgeteilt in alle wichtigen Funktionen und Beispielfunktionen F für die Gebietsbeschreibung und die rechten Seiten f, g_D und g_N.
gnuplot	pfeile.plt plot_dots.plt plot_normal.plt	Dateien zur Visualisierung des betrachteten Gebiets inklusive Nachbarschaftsbeziehungen in Gnuplot.
maple	differenzenapprox_innen.mw differenzenapprox_neumann.mw plot.mw	Maple-Worksheets zur Herleitung der Differenzenapproximationen, Lösen der Gleichungssysteme aus Kapitel 2 und für die grafische Darstellung.

Tabelle A.1: Überblick über den Inhalt der CD-ROM

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Jahn, T.: *Implementierung numerischer Algorithmen auf CUDA-Systemen*, Universität Bayreuth, Diplomarbeit, 2010

[2] Büskens, C.: *Linienmethoden.* 2003. – Vorlesungsskript WS 2002/2003

[3] Reinhardt, H.-J.: *Differenzenapproximationen partieller Differentialgleichungen.* 1997. – Online verfügbar auf http://www.uni-siegen.de/fb6/numerik/skripts/skripte/ (Dateiname: DPDscript.zip); zuletzt aufgerufen am 22.09.2011, 19:10 Uhr

[4] Hackbusch, W.: *Theorie und Numerik elliptischer Differentialgleichungen, Erste Fassung.* 2005. – Online verfügbar auf http://www.mis.mpg.de/publications/other-series/ln/lecturenote-2805.html; zuletzt aufgerufen am 22.09.2011, 19:09 Uhr

[5] *The C++ Resources Network.* – http://www.cplusplus.com/reference/stl/; zuletzt aufgerufen am 22.09.2011, 19:09 Uhr

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Bayreuth, den 30. September 2011

(Arthur Fleig)