Institut für Numerische Simulation der Universität Bonn
Prof. Dr. Mario Bebendorf

Praktikum im Sommersemester 2013

Programmierpraktikum numerische Algorithmen (P2E1)
(Numerische Lösung der Wärmeleitungsgleichung)

Betreuer: Christian Kuske[1]

Blatt 1

1 Einführung

In diesem Programmierpraktikum soll eine numerische Lösung $u\colon \Omega \subset \mathbb{R}^d \times [0,T] \to \mathbb{R}$ der Wärmeleitungsgleichung

$$\begin{aligned} -\Delta u + c\frac{\partial u}{\partial t} &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{auf } \partial\Omega \end{aligned} \tag{1}$$

gefunden werden. Hierbei ist $\mathbb{R} \ni c > 0$ die Temperaturleitfähigkeit, $g \in \partial\Omega$ eine Funktion die Anfangs- und Randwerte vorgibt, $\frac{\partial u}{\partial t}$ die partielle Ableitung von u nach der Zeit t und Δ bezeichne den d-dimensionalen LAPLACE-Operator.

Bei der Wärmeleitungsgleichung handelt es sich um eine parabolische partielle Differentialgleichung zweiter Ordnung. Ein Überblick über die verschiedenen Typen von Differentialgleichungen wird im folgenden Abschnitt gegeben.

Einschränkung

Der Einfachheit halber wird die Gleichung (1) für $d = 2$ betrachtet. Die Übertragung der später vorgestellten Verfahren auf mehr als zwei Dimensionen lässt sich ohne große Änderungen im Programmcode bewerkstelligen.

Zusätzlich wird das Gebiet auf den zweidimensionalen Einheitswürfel beschränkt.

Insgesamt ergibt sich:

$$\begin{aligned} u\colon [0,1]^2 \times [0,T] &\to \mathbb{R}, \\ (x,y) \times t &\mapsto u(x,y,t). \end{aligned}$$

Zur Bestimmung der numerischen Lösung der Gleichung (1) wird diese zunächst für den stationären (zeitunabhängigen, $t = t_0$) Fall ermittelt. Das heißt, es wird $\frac{\partial u}{\partial t} = 0$ gesetzt.

[1] We6 4.002; Tel.: 0228/73-60454; Email: kuske@ins.uni-bonn.de

Damit lautet Gleichung (1)

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega := [0,1]^2, \\ u &= g \quad \text{auf } \partial\Omega. \end{aligned} \tag{2}$$

Diese Gleichung ist eine elliptische partielle Differentialgleichung zweiter Ordnung. Zu beachten ist, dass bei der Lösung der Gleichung (2) c keine Rolle spielt, da ein stationäres Problem gelöst wird und c lediglich die Ausbreitung der Temperatur beschreibt.

Diskretisierung

Da die Gleichung (2) kontinuierlich ist und sich somit nicht im Computer darstellen lässt, muss sie diskretisiert werden. Genaueres findet sich in Abschnitt 3.

Durch den Übergang von Ω zum (inklusive Rand) diskretisierten Gebiet Ω_h mit Gitterweite h ergibt sich das Gleichungssystem

$$L_h u_h = f_h. \tag{3}$$

Die dadurch entstehende Matrix $L_h \in \mathbb{R}^{n^2 \times n^2}$ und rechte Seite $f_h \in \mathbb{R}^{n^2}$ lassen sich berechnen. Die diskrete Lösung u_h stellt damit eine Näherung an u aus Gleichung (2) dar.

Da die Matrix L_h wenige Einträge ungleich Null besitzt, bietet sich ein iteratives Verfahren zum Lösen der Gleichung (3) an.

2 Differentialgleichungen

Eine Differentialgleichung ist eine Gleichung einer unbekannten Funktion von einer oder mehreren Variablen, die diese mit ihren Ableitungen in Bezug setzt. Solche Gleichungen beschreiben häufig Vorgänge in der Physik, finden aber auch Anwendung in anderen Bereichen.

Beispiel 2.1. *Bei der Berechnung von Zinsen hängt der Zuwachs proportional von der momentanen Geldmenge ab, das heißt*

$$\frac{dK(t)}{dt} = rK(t).$$

Hierbei ist r die Zinsrate und $K(t)$ das Kapital zum Zeitpunkt t. Es handelt sich um eine gewöhnliche Differentialgleichung erster Ordnung. Damit die Lösung eindeutig ist, müssen Anfangswerte $K(t_0) = k_0$ festgelegt werden.

Gewöhnliche Differentialgleichungen

Wie im obigen Beispiel zu sehen ist, hängen gewöhnliche Differentialgleichungen von lediglich einer Variablen ab, wobei die Ordnung den höchsten Grad der Ableitung angibt.

Partielle Differentialgleichungen

Bei dieser Art von Differentialgleichungen ist eine Abhängigkeit in mehreren Variablen gegeben (mind. zwei), wobei zusätzlich partielle Ableitungen in mindestens zwei Variablen auftauchen. Auch in diesem Fall bezeichnet die Ordnung den höchsten Grad der Ableitung.

In diesem Praktikum wird sich vorwiegend mit partiellen Differentialgleichung zweiter Ordnung befasst. Eine allgemeine Darstellung in d Variablen sieht wie folgt aus:

$$-\sum_{i,j=1}^{d} a_{ij}(x)\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d} b_i(x)\frac{\partial u}{\partial x_i} + c(x)u = f(x). \tag{4}$$

Im Unterschied zu den gewöhnlichen Differentialgleichungen gibt es drei wesentliche Typen von partiellen Differentialgleichungen zweiter Ordnung. Mit $(A(x))_{ij} := a_{ij}(x)$ ergibt sich:

- Die Gleichung (4) heißt elliptisch, falls $A(x)$ postiv definit für alle x ist.
- Die Gleichung (4) heißt hyperbolisch, falls $A(x)$ einen negativen und $d-1$ positive Eigenwerte für alle x hat.
- Die Gleichung (4) heißt parabolisch, falls $A(x)$ positiv semidefinit aber nicht definit ist und der Rang von $(A(x), b(x))$ gleich d für alle x ist.

Der d-dimensionale LAPLACE-Operator besteht aus der Summe aller partiellen Ableitungen zweiten Grades in einer Dimension, das heißt

$$\Delta := \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}. \tag{5}$$

Aus den Gleichungen (1) und (2) werden mit Hilfe von (4) und (5) folgende Matrizen für $d = 2$ erzeugt:

$$A_p(x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad b_p(x) = \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix}, \qquad A_e(x) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Die Matrix $A_p(x)$ und der Vektor $b_p(x)$ erfüllen die dritte Bedingung der Gleichung (4) und damit ist Gleichung (1) eine parabolische partielle Differentialgleichung zweiter Ordnung in drei Variablen. Dabei ist die dritte Variable die Abhängigkeit der Zeit. Die Gleichung (2) ist elliptisch.

Diese Strukturen bleiben für beliebiges d erhalten.

3 Diskretisierung

Eine Diskretisierung ist der Übergang von einem kontinuierlichen Modell (1), (2) zu einem diskreten Modell (3), das durch endlich viele Daten beschrieben werden kann. Dabei entsteht ein sogenannter Diskretisierungsfehler $e_h := u - u_h$, der die Abweichung des kontinuierlichen Problems zum diskretisierten angibt. Dieser Fehler sollte bei einer Verringerung der Gitterweite $h = \frac{1}{n-1}$ gegen Null streben, wobei n die Anzahl der Diskretisierungspunkte darstellt.

Finite Differenzen

Da die Gleichungen (1) und (2) Ableitungen enthalten, stellt sich die Frage, wie sich diese mit Hilfe von diskreten Datenpunkten berechnen beziehungsweise annähern lassen. Dazu dient die Methode der Finiten Differenzen.

Diese Methode ist durch die Definition der Ableitung über den Differenzenquotienten

$$\frac{du}{dx} := \lim_{h \to 0} \frac{u(x+h) - u(x)}{h}$$

motiviert.

Für die erste Ableitung im Eindimensionalen gibt es drei verschiedene Typen von Finiten Differenzen:

- $(\partial^+ u)(x) := \dfrac{u(x+h) - u(x)}{h}$ Vorwärtsdifferenz
- $(\partial^- u)(x) := \dfrac{u(x) - u(x-h)}{h}$ Rückwärtsdifferenz
- $(\partial^0 u)(x) := \dfrac{u(x+h) - u(x-h)}{2h}$ zentrale Differenz

Für die zweite Ableitung in einer Dimension lassen sich die Finiten Differenzen wie folgt definieren:

- $(\partial^+(\partial^- u))(x) = (\partial^-(\partial^+ u))(x) := \dfrac{u(x+h) - 2u(x) + u(x-h)}{h^2}$
- $(\partial^0(\partial^0 u))(x) := \dfrac{u(x+2h) - 2u(x) + u(x-2h)}{4h^2}$

Alle anderen Kombinationen von ∂^+, ∂^- und ∂^0 sind ebenfalls möglich, führen allerdings auf Verfahren mit einer schlechteren Konsistenzordnung. In der Praxis wird für die zweite Ableitung meist $(\partial^+(\partial^- u))(x)$ verwendet, wobei diese dann mit dem sogenannten Differenzenstern

$$h^{-2} \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

abgekürzt wird.

Konsistenz

Ist ein numerisches Verfahren konsistent, so approximiert dieses Verfahren das gegebene Problem. Wie schnell dies geschieht, lässt sich durch die Konsistenzordnung beschreiben.

Eine Diskretisierung $-L_h u_h = f_h$ von $-\Delta u = f$ heißt konsistent von Ordnung k, falls für $h \to 0$

$$\|\tilde{R}_h \Delta u - L_h R_h u\|_\infty \leq c h^k \|u\|_{C^{k+2}(\bar{\Omega})}$$

für alle $u \in C^{k+2}(\bar{\Omega})$ gilt. Hierbei sind $\tilde{R}_h, R_h$ geeignete Restriktionen auf ein Gitter Ω_h und c eine von h, u unabhängige Konstante.

$$\begin{array}{ccc} u & \xrightarrow{-\Delta} & f \\ {\scriptstyle R_h}\downarrow & & \downarrow{\scriptstyle \tilde{R}_h} \\ u_h & \xrightarrow{-L_h} & f_h \end{array}$$

Stabilität

Ist ein numerisches Verfahren stabil, so ist es unempfindlich gegenüber kleinen Störungen der Daten. Das bedeutet, dass die Störungen das Verfahren nicht beeinflussen und auftretende Rundungsfehler die Lösung nicht verfälschen.

Eine Diskretisierung L_h heißt stabil, falls

$$\sup_{h>0} \|L_h^{-1}\|_\infty < \infty$$

gilt.

Bemerkung 3.1 (Konvergenz)**.** *Es lässt sich zeigen, dass aus der Konsistenz und der Stabilität die Konvergenz des Verfahrens folgt. Dabei entspricht die Konvergenzordnung der Konsistenzordnung.*

Gleichungssystem

Die Finiten Differenzen führen auf schwach besetzte Matrizen. Beispielsweise entsteht aus der Differenz $(\partial^+(\partial^- u))(x)$ auf Gleichung (2) für $d = 1$ angewendet folgendes Gleichungssystem:

$$\frac{1}{h^2}\begin{pmatrix} 1 & & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & & 1 \end{pmatrix} \cdot \begin{pmatrix} u_h(0) \\ u_h(h) \\ \vdots \\ u_h((n-2)h) \\ u_h(1) \end{pmatrix} = \begin{pmatrix} g_0/h^2 \\ 0 \\ \vdots \\ 0 \\ g_1/h^2 \end{pmatrix} \in \mathbb{R}^n. \tag{6}$$

Hierbei ist das Gebiet $[0, 1]$ durch n Ãđquidistante Punkte unterteilt und g_0, g_1 sind feste Randwerte an den Intervallgrenzen Null und Eins. Die Randwerte werden jeweils durch h^2 geteilt, da die Gleichungen aus (2) in die Gleichung (3) übertragen werden. Durch das Addieren der ersten und der letzten Zeile auf die zweite und die vorletzte Zeile, erhält man eine symmetrische Systemmatrix:

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} u_h(h) \\ u_h(2h) \\ \vdots \\ u_h((n-3)h) \\ u_h((n-2)h) \end{pmatrix} = \begin{pmatrix} g_0 \\ 0 \\ \vdots \\ 0 \\ g_1 \end{pmatrix} \in \mathbb{R}^{n-2}.$$

Dieses Gleichungssystem lässt sich mit dem Gauss-Seidel-Verfahren lösen. Dabei ist zu beachten, dass aufgrund der Lokalität der Finiten Differenzen - die Ableitung an der Stelle x ist lediglich von den Nachbarpunkten $x + h$ und $x - h$ abhängig - die Matrix nicht aufgestellt werden muss. Es genügt, das Gauss-Seidel-Verfahren für die Matrixeinträge ungleich Null zu berechnen. Die Lösungen $u_h(0), u_h(1)$ am Rand müssen dabei nicht berücksichtigt werden, da diese bereits festgelegt sind.

4 Aufgaben

a) Bestimme den Differenzenstern für den Laplace-Operator in zwei Variablen und stelle das lineare Gleichungssystem zu Gleichung (2) auf, wobei die Matrix symmetrisch sein soll!

b) Implementiere eine Matrix-Vektor-Multiplikation für die schwach-besetzte Diskretisierung des Laplace-Operators in symmetrischer Form:

- `void applyLaplace(const unsigned int n, const double * const x, double * const y)`,
  wobei $y = L_h x$ berechnet wird und L_h symmetrisch positiv definit ist.

c) Implementiere zur Lösung dieses Gleichungssystems das Gauss-Seidel-Verfahren! Verwende dazu die folgenden Funktionen:

- `void initRHS(const unsigned int n, double * const f)`,
  wobei die rechte Seite des symmetrischen Gleichungssystem $L_h u = f_h$ initialisiert wird.
- `void GaussSeidel(const unsigned int n, const double * const f, double * const u, double * const eps, unsigned int * const iter)`,
  wobei das Gleichungssystem $L_h u = f_h$ gelöst wird. Das Verfahren soll abbrechen,

sobald $\|u^{(k+1)} - u^{(k)}\|_\infty < \varepsilon$ ist oder nach maximal `iter` Iterationsschritten. Die erreichte Genauigkeit und die dabei benötigten Iterationsschritte sollen in den jeweiligen Variablen gespeichert werden.

d) Teste das Verfahren für $50 \leq n \leq 500$ und $10^{-8} \leq \varepsilon \leq 10^{-3}$ und verwende jeweils die Randdaten $g(x, y) = 100$ für $(x, y) \in (0, 1) \times \{0\}$, $g(x, y) = 40$ für $(x, y) \in \{0, 1\} \times (0, 1)$ und $g(x, y) = 20$ für $(x, y) \in (0, 1) \times \{1\}$! Stelle die Ergebnisse grafisch dar und überprüfe mit Hilfe von Aufgabenteil b) den relativen Fehler der Lösung; benutze dazu:

- `extern void createVTKFile(const unsigned int n, const double * const u).`
- `double nrm2(const unsigned int n, const double * const x)`, wobei der Rückgabewert der euklidischen Norm $\|x\|_2$ entspricht.
- `double diff(const unsigned int n, const double * const x, const double * const y)`, wobei der Rückgabewert der euklidischen Norm $\|x - y\|_2$ entspricht.

e) Vergleiche die Zahl der nötigen Iterationsschritte sowie die Zeit bei steigendem n mit Hilfe der bereitgestellten Funktion

- `extern "C" double cputime(const double).`