Computerpraktikum Wintersemester 2023/24

Themenvorschläge Numerische Simulation

Prof. Dr. Dominik Göddeke

Stand: 29.12.2023

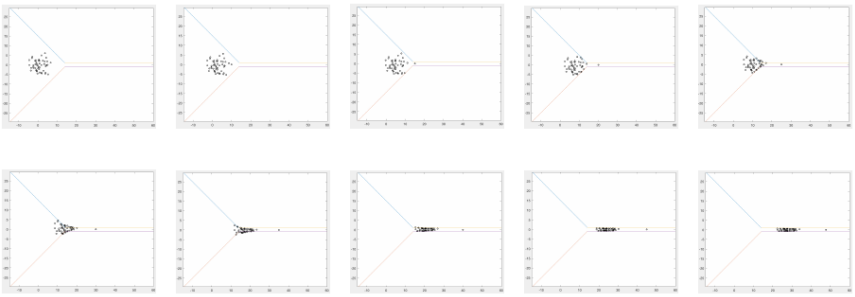
Organisatorisches

- Auf den folgenden Seiten finden Sie Themenvorschläge für das Numerik-Drittel des Computerpraktikums. Zu jedem Thema finden Sie ausführliche Literaturangaben, erste Implementierungshinweise, sowie eine grobe Einschätzung, welches Vorwissen hilfreich ist.
- Bei fast allen Themen ist eine Einarbeitung in noch unbekannte numerische Methoden nötig, wenigstens wenn Sie sich im 5. Fachsemester befinden. Dies ist Absicht, ich werde Sie aber sicher nicht alleine lassen. Für einen ersten Eindruck, der fast immer ausreicht für den *Einsatz* der Verfahren, stehen die Folien zu meiner Veranstaltung *Höhere Mathematik 4 / Numerik* zur Verfügung.
- Die Programmiersprache dieses Teils des Computerpraktikums ist Python. Es ist ausdrücklich erlaubt, alle Möglichkeiten von bspw. SciPy und NumPy zu nutzen, sofern sie die Aufgabe nicht trivialisieren. Die Dokumentation der von Ihnen erstellten Programme wird in die Benotung eingehen, in dem Sinne, dass unzureichende Dokumentation zu einer Abwertung führt. Es empfiehlt sich in vielen Fällen, parallel auch ein geteiltes Overleaf/Sync&Share/github/ILIAS Dokument zu erstellen, in dem Sie stichpunktartig Knackpunkte beim Verständnis sowie Designentscheidungen bei der Implementierung notieren. Dies wird die Erstellung der (benoteten) Abschlusspräsentation enorm vereinfachen.
- Mir ist wichtig, dass Sie strukturiert testend vorgehen. Validieren Sie Ihre Implementierungen, sowohl von Einzelkomponenten als auch des Gesamtsystems, anhand von Referenzlösungen. In der Numerik 1 wird beispielsweise häufig ein lineares Gleichungssystem mit bekannter Lösung gelöst. Dieser modulare „bottom-up“ Zugang spart enorm Zeit und Nerven. Es empfiehlt sich, die Gesamtstrategie zur Lösung der Aufgabe „top-down“ aufzustellen, und die Implementierung dann „bottom-up“ vorzunehmen.
- Ich erwarte neben der Validierung zusätzliche „Showcases“: Bei einigen Themen können dies Parameterstudien sein (Laufzeit in Abhängigkeit von Problemgröße, Einfluss eines freien Parameters), bei anderen Themen weitere simulierte Szenarien, die über eine reine Validierung hinausgehen, und bei wieder anderen Themen beides. Wir diskutieren dies in der Einzelbetreuung.
- Bei der Bewertung übernehme ich die Vorgehensweise von J. Dippon: Ca, 30 Minuten gemeinsame Präsentation inkl. Code-Demo, sowie ein gemeinsamer Aufschrieb. Hier reichen mir allerdings ca. 2 gemeinsame Seiten und 4–6 Seiten (plus Grafiken/Tabellen) pro Person. Wie Ihr Material sinnvoll in der Gruppe aufgeteilt werden kann wird gemeinsam besprochen. Die schriftliche Abgabe ist in der Woche nach den Präsentationen fällig.
- Wenn Sie die umfangreichen Sprechstundenangebote nicht nutzen, sind Sie selbst schuld :) – insbesondere erfolgt die detaillierte Ausgestaltung der Projekte individuell. Die Terminvorschläge sind noch lediglich Vorschläge, die noch vor der Erfindung der Energiesparwochen festgelegt wurden.

1 Modellierung und Simulation von Populismus

Vorkenntnisse Kontakt mit gewöhnlichen Differentialgleichungen, keine Numerik

Zusammenfassung Schwarmverhalten ist in der Natur vielfältig zu beobachten, bekannte Beispiele reichen vom Verhalten von großen Sardinenschwärmen[1] hin zur Beschreibung menschlichen Verhaltens in der Soziologie, beispielsweise zur Untersuchung von Wahlverhalten in sogenannten Follow-the-Leader Szenarien. Objekte in einem Schwarm bewegen sich gemeinsam und reagieren auf externe Ereignisse wie das Vorbeischwimmen eines Hais. Analog machen sich Menschen zunehmend eine Meinung zu eigen und ändern sie abrupt aufgrund bspw. einer Fernsehdebatte. Es hat den Anschein, als ob es eine zentrale Steuerung gibt, beziehungsweise als wäre der Schwarm / die Massenmeinung ein einzelner, denkender Organismus. Tatsächlich ist es so, dass bereits sehr einfache mathematische Modelle ein solches sogenanntes *emergentes Verhalten* ziemlich erfolgreich beschreiben können. Neben den Anwendungen in der Soziologie, der Biologie und vielen weiteren mehr kommen diese Modelle auch cineastisch zum Einsatz, so wurden beispielsweise bereits in der Gründerzeit der Computereffekte in Kinofilmen Fledermausschwärme in „Batman Returns (1992)“ oder die Stampede bei „König der Löwen (1994)“ auf diese Weise erstellt.



Details In diesem Projekt soll ein Partikelsimulator für Meinungsmache erstellt werden. Die Grundlage kann das Modell von Cucker und Smale [3] sein, für das sich tatsächlich einiges beweisen lässt. Hilfreich ist auch eine Arbeit zu Anführern und hierarchische Führung [2]. [2] Allerdings sollen explizit nicht Haie und Sardinen, sondern die Meinungsbildung betrachtet werden: Ein*e Meinungsträger*in ist modelliert durch ein Partikel, das andere Partikel beeinflusst.

Die Minimalziele umfassen eine Implementierung in 2D als Animation. Nehmen Sie an, dass es zwei Alternativen gibt (ja gegen nein, Pmurt gegen Nedib, Hörsaalschließer gegen Hörsaalöffner, . . .), codiert durch die Farbe der Individuen. Individuen sind zunächst neutral gestimmt, befinden sich gleichverteilt in einem „Raum“ und bewegen sich mehr oder weniger erratisch. Führen Sie dann eine*n „Stimmungsmacher*in“ ein, die/der sich auf einer vordefinierten Bahn durch den Raum bewegt, und all jene Individuen von ihrer/seiner Meinung überzeugt, die ihr/ihm zu nahe kommen. Sobald ein Individuum überzeugt wird, wird es ebenfalls stimmungsmachend, und folgt der/dem Überzeuger *in auf dessen Bahn. Verwenden Sie für die Zeitintegration das explizite Euler-Verfahren. Validieren Sie experimentell die Konvergenz des Modells anhand geeigneter Visualisierungen aggregierter Größen, beispielsweise den mittleren Abstand oder die Differenz der Normen der Zustimmungen (Geschwindigkeiten und Bewegungsrichtungen abweichend vom stimmungsmachenden Partikel) der Individuen.

[1] `https://www.youtube.com/watch?v=V5up0yaYz9E`

[2] Eine schöne Übersicht und einen vielleicht schnelleren Einstieg als die Originalveröffentlichung bieten diese Folien: `https://www.math.uni-potsdam.de/fileadmin/user_upload/Prof-Wahr/Zass/CDFA_Slides/DelebecquePedeches-Flocking.pdf`

Die konkrete Ausgestaltung kann in zwei Richtungen zielen: Bei einer gewissen Affinität zum Vergleich numerischer Verfahren werden mächtigere Lösungsverfahren gewählt, beispielsweise Runge-Kutta Verfahren und/oder implizite Verfahren zur Zeitintegration. Bei dieser Ausgestaltung stehen Studien von Genauigkeit und Laufzeit für ein konvergentes Szenario eher im Fokus. Bei einer gewissen Affinität zur Computergrafik wird eine interaktive Simulation erstellt, für die Sie sich beliebig austoben bei der Approximation realer Szenarien. Beispielsweise kann die/der „Anführer*in“ mit der Maus bewegt werden als Emulation des Fokus einer Wahlkampagne auf noch nicht überzeugte Gebiete. Denkbar ist auch, die Meinung nicht binär als $\{0, 1\}$ zu modellieren, sondern als Wert im Intervall $[0, 1]$, und dies durch Farbskalen abzubilden. Auch eine immer stärkere Überzeugung ist spannend, beispielsweise durch einen Kontakt mit einem gleichgesinnten Individuum. Orthogonal dazu kann mehr als ein*e Stimmungsmacher*in eingeführt werden, und die skizzierten Szenarien (oder Ihre eigenen Ideen) umgesetzt werden.

Ergänzende Literatur Einen schnellen Einstieg in Euler- und Runge-Kutta-Verfahren bieten die Folien zur VL8 der HM4 des Dozenten (ILIAS). `pygame` oder `matplotlib` können sich als hilfreich bei der Implementierung der Interaktivitätskomponente erweisen.

2 Musterbildung auf Tierfellen

Vorkenntnisse Erste Kontakte mit partiellen Differentialgleichungen und der zugehörigen Numerik, bspw. (laufender) Besuch der NUMDGL

Zusammenfassung Aus enzymkinetischen Überlegungen heraus, beispielsweise über Inhibitor-Aktivator Mechanismen, lassen sich zahlreiche Diffusions-Reaktionsmodelle konstruieren, üblicherweise in der folgenden Form:

$$\begin{cases} \partial_t u = D_u \Delta u + f(u, v) & \text{in } \Omega \times I \\ \partial_t v = D_v \Delta v + g(u, v) & \text{in } \Omega \times I \end{cases} \qquad \text{plus Anfangs- und Randwerte}$$

Hierbei sind $u(x, t), v(x, t)$ die Konzentrationen zweier Substanzen, D_u, D_v die zugehörigen Diffusionsparameter, $x \in \Omega =]a, b[^2$ das „Tierfell“, $t \in I =]0, T]$ ein Zeitintervall, und die Kopplung zwischen den Substanzen erfolgt durch die Reaktionsterme f und g, die üblicherweise nichtlinear sind. Diese Modelle können unter bestimmten Annahmen die Musterbildung auf Tierfellen beschreiben. Beispiele sind die Modelle von Schnakenberg oder von Gierer und Meinhardt. Im Lehrbuch von Eck et al. [5, Kap. 6.2.12] findet sich eine kurze Übersicht, eine ausführliche Diskussion bietet das Lehrbuch von Murray [8]. Aus mathematischer Sicht sehr spannend ist, dass das sich ausbildende Muster über den sogenannten Turing-Mechanismus analysierbar ist, also letztendlich eine linearisierte spektrale Stabilitätsanalyse, die stationäre Zustände in Abhängigkeit von Stabilitätsgebieten der einzelnen Parameter liefert.

Details In diesem Projekt soll ein Simulator für Tierfell-Muster erstellt werden, wobei pro Team nur ein Modell betrachtet werden soll. Die Stabilitätsanalyse ist nicht Bestandteil des Projekts, stattdessen sollen kritische Parameter aus der Literatur übernommen werden, was automatisch der Validierung dient. Die Simulation soll die Entstehung eines Musters über die Zeit visualisieren. Zur Diskretisierung des Diffusionsoperators im Ort kommen zentrale Finite Differenzen zweiter Ordnung zum Einsatz, eventuelle Konvektions- und Reaktionsterme in f und g werden stabilisiert, und die Diskretisierung in der Zeit erfolgt mit dem expliziten Euler-Verfahren. Die Nichtlinearität wird mit einem Newton-Verfahren oder einer Fixpunktiteration behandelt.

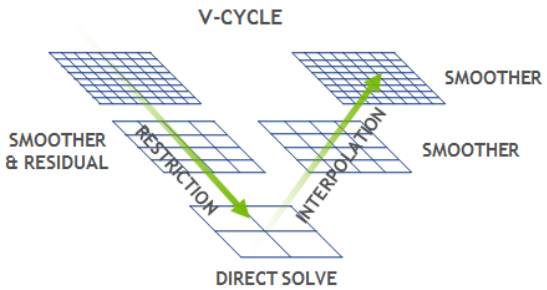
Die genaue Ausgestaltung wird nach der Einarbeitungsphase individuell besprochen, wobei ähnlich wie im vorherigen Thema zwei Ansätze denkbar sind: In einer anwendungsorientierten Ausgestaltung werden mehr Beispiele (und auch Gegenbeispiele) konstruiert, und die Ausbildung der Muster wird gründlicher erklärt. In einer methodischen Ausrichtung stehen beispielsweise Vergleiche zwischen unterschiedlichen Zeitintegrations- oder Nichtlinearitäts-Techniken im Mittelpunkt.

Ergänzende Literatur Einen schnellen Einstieg in die genannten numerischen Methoden bieten die Folien zur VL8, VL9 und VL10 der HM4 des Dozenten (ILIAS).

3 Mehrgitterverfahren

Vorkenntnisse Inhaltlich: Jacobi- und Gauß-Seidel Verfahren; Thematisch: (Numerische) Lineare Algebra, beweisbar optimale Lösungsverfahren

Zusammenfassung Bei diesem Thema steht nicht so sehr die Simulation im Vordergrund, sondern eher eine bestimmte numerische Verfahrensklasse: Einfache Defektkorrektur-Methoden wie das Jacobi- oder das Gauß-Seidel Verfahren konvergieren nur sehr langsam, vgl. die NUM1/2. In vielen Fällen kann jedoch ein hierarchischer Zugang Abhilfe schaffen, indem eine Fehlerkorrektur auf einem gröber aufgelösten Problem (rekursiv) vorgenommen wird, vergleiche die Skizze. Man kann beweisen, dass diese Verfahren für bestimmte Probleme nur konstant viele arithmetische Operationen pro Unbekannte durchführen bis zur Konvergenz, was sie natürlich ausgesprochen attraktiv macht, da die Konstante moderat klein ist.



Details In diesem Projekt soll dieses sogenannte Mehrgitter-Verfahren für ein einfaches Beispielproblem untersucht werden. Wir beschränken uns dazu auf das Poisson-Problem $-\Delta u = f$ in 1D und später 2D mit homogenen Dirichlet-Randbedingungen, und verwenden eine Diskretisierung mit zentralen Finiten Differenzen zweiter Ordnung. Weiter verwenden wir zunächst nur den sogenannten V-Zyklus, das gedämpfte Jacobi-Verfahren $x^{(k+1)} = x^{(k)} + \omega D^{-1}(b - Ax^{(k)})$ als Glätter, und bilineare Interpolation als Prolongation und die dazu adjungierte Restriktion durch gewichtete Mittelung. Alle vorkommenden Matrizen sollen in einem geeigneten dünnbesetzten Format gespeichert werden, vgl. `scipy.sparse`.

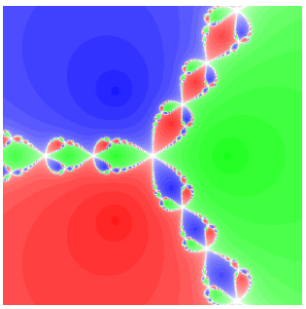
Wie danach weiter vorgegangen wird, diskutieren wir individuell. Denkbare Erweiterungen sind die Implementierung weiterer Glättungsoperatoren neben Jacobi, zusätzlicher Prolongations- und Restriktionsoperatoren oder anderer Zyklentypen. Ebenso denkbar sind ausführliche Parameterstudien für die freien Parameter Glättungsschritte und Dämpfungsparameter, sowie Laufzeitvergleiche gegen andere iterative und direkte Löser für dünn besetzte lineare Gleichungssysteme, die in SciPy zur Verfügung stehen. Auch der Übergang zu anderen Modellgleichungen wie $-\nabla \cdot A\nabla u = f$ mit einer Anisotropie-Matrix A kann das Projekt sinnvoll erweitern.

Ergänzende Literatur Einen schnellen Einstieg in die genannten numerischen Methoden bieten die Folien zur VL9 der HM4 des Dozenten (ILIAS). Ebenfalls steht ein Auszug aus einem Vorlesungsskript des Dozenten bereit, der das Prinzip in 1D motiviert (schamlos übernommen aus dem grandiosen Büchlein von Briggs [1]). Lehrbücher zur Theorie für weiterführende Fragestellungen sind beispielsweise: Trottenberg et al. [10], Wesseling [11] und Hackbusch [7]

4 Newton-Fraktale

Vorkenntnisse Kreativität und Definition Selbstähnlichkeit

Zusammenfassung Das Newton-Verfahren dient normalerweise dazu, Nullstellen nichtlinearer Funktionen zu approximieren. Dies ist bekanntlich aus der Algebra, und dort der Galois-Theorie, bereits für Polynome fünften Grades ein Problem, für das keine geschlossene Lösungsformel existiert, wenn wir uns auf Grundrechenarten und Wurzelziehen beschränken (Satz von Abel-Ruffini). Für Funktionen $f : \mathbb{C} \to \mathbb{C}$ besitzt es eine spannende künstlerische Komponente, die in diesem Projekt ergründet werden soll: Ähnlich wie bei den bekannten Mandelbrot- und Julia-Mengen kann das Konvergenzverhalten dazu verwendet werden, fraktale Strukturen zu erzeugen. Die Idee besteht darin, eine Funktion zu betrachten, die komplexe Nullstellen besitzt, beispielsweise $f(x) = x^3 - 1$. Wenn wir einen Ausschnitt der komplexen Ebene in N äquidistante Punkte $\{p_i \in \mathbb{C}\}_{i=1}^{N}$ zerlegen, d.h. ein Pixelgitter über die komplexe Ebene legen, können wir jeden dieser Punkte als Startwert des Newton-Verfahrens verwenden, und für jeden dieser Startwerte die Anzahl der Iterationen bis zur Konvergenz sowie die gefundene Nullstelle bestimmen. Wenn wir jeden Punkt p_i in einer Farbe färben, die mit der gefundenen Nullstelle korrespondiert, und mit einer Helligkeit, die mit der Anzahl der Iterationen bis zur Konvergenz korreliert, so wird der Punkt immer heller eingefärbt, je größer die Anzahl der Iterationen ist. Die Farbe weiß (maximale Helligkeit) wird für Startwerte verwendet, für die das Newton-Verfahren divergiert, vergleiche die folgende Abbildung:



Details Das Ziel dieses Projekts ist die Implementierung eines interaktiven Fraktal-Generators mit Hilfe des Newton-Verfahrens. Erläutern und implementieren Sie dazu zunächst das Newtonverfahren in 1D für komplexe Zahlen, und verstehen Sie das Konzept der lokalen Konvergenz. Ermitteln Sie experimentell einen geeigneten Schwellwert, ab dem Sie das Newton-Verfahren als divergent betrachten können.

Validieren Sie Ihre Implementierung anhand geeigneter Testfälle, indem Sie beispielsweise die Abbildung reproduzieren.

Im zweiten Schritt soll die Interaktivität ergänzt werden, in Form einer Zoomfunktion. Mit der Maus soll der betrachtete Ausschnitt in der komplexen Ebene verschoben werden können, und ein rechteckiger Bereich in der Visualisierung soll gezoomt werden können. Auch eine Möglichkeit zum Herauszoomen ist vorzusehen. Experimentieren Sie mit anderen Funktionen, und entwickeln Sie eine Intuition, wie das fraktale Muster aussehen könnte für Klassen von Funktionen. Schließlich ist eine Optimierung der Laufzeit vermutlich unabdingbar, wenigstens für große Pixelzahlen. Hier sind einige Parameterstudien hilfreich, sowie Plots der Funktionen, die Sie untersuchen.

Ergänzende Literatur Das Newton-Verfahren wird in jedem guten Lehrbuch der Numerik behandelt, die Werke von Dahmen und Reusken [4], Freund und Hoppe [6] und Schaback und Wendland [9] stehen als eBooks über die UB zur Verfügung. Einen schnellen Einstieg bietet auch VL4 der HM4 des Dozenten.

Literatur

[1] William L. Briggs. *A Multigrid Tutorial*. SIAM, 1987.

[2] Young-Pil Choi, Seung-Yeal Ha, und Zhuchun Li. Emergent dynamics of the Cucker-Smale flocking model and its variants. Technischer Bericht arXiv:1604.04887, arXiv, April 2016.

[3] Felipe Cucker und Steve Smale. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52(5), Mai 2007. DOI: 10.1109/TAC.2007.895842.

[4] Wolfgang Dahmen und Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer, 2. Auflage, 2008. DOI: 10.1007/978-3-540-76493-9.

[5] Christof Eck, Harald Garcke, und Peter Knabner. *Mathematische Modellierung*. Springer, 2. Auflage, 2011. DOI: 10.1007/978-3-642-18424-6.

[6] Roland W. Freund und Ronald W. Hoppe. *Stoer/Bulirsch: Numerische Mathematik 1*. Springer, 10. Auflage, 2007. DOI: 10.1007/978-3-540-45390-1.

[7] Wolfgang Hackbusch. *Multi-grid methods and applications*. Springer, Oktober 1985.

[8] James D. Murray. *Mathematical Biology*. Springer, 3. Auflage, 2002. DOI: 10.1007/b98868.

[9] Robert Schaback und Holger Wendland. *Numerische Mathematik*. Springer, 5. Auflage, September 2004. DOI: 10.1007/b137970.

[10] Ulrich Trottenberg, Cornelius W. Oosterlee, und Anton Schuller. *Multigrid*. Academic Press, 2001.

[11] Pieter Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, 1992.