Partielle Differentialgleichungen und Numerische Software

Frederik Fuhrmann, Uwe Kleis und Wolfgang Mackens

Arbeitsbereich Mathematik der TU Hamburg-Harburg
Kasernenstraße 12, D-21073 Hamburg
http://www.tu-harburg.de/mat

Zusammenfassung

Software zur Lösung partieller Differentialgleichungen wird exemplarisch besprochen und Zugangsmöglichkeiten werden benannt. Wünschenswerte Eigenschaften zukünftiger Löser werden gesammelt.

1 Einleitung

Mit partiellen Differentialgleichungen (PDEs = Partial Differential Equations) werden die mathematischen Probleme beschrieben, deren Lösung zu beherrschen die Konkurrenzfähigkeit der heutigen Hochtechnologie entscheidend mitbestimmt. Kein Wunder also, daß viel Zeit und Aufwand darauf verwendet wird, anwendungsorientierte Software zur Lösung partieller Differentialgleichungen zu entwickeln, mit der Anwender für die konkreten PDEs ihres Arbeitsgebietes Näherungslösungen erzeugen können, ohne je Theorie oder Numerik „ihrer“ PDEs erlernen zu müssen. Prä- und Postprozessoren übernehmen die Arbeit bei der Umsetzung des spezifischen Anwendungsproblems in eine PDE und übersetzen nach der numerischen Behandlung die gewonnene Lösungsapproximation in Objekte einer virtuellen Wirklichkeit zurück, so daß der Anwender das Ergebnis nicht nur leicht interpretieren kann, sondern sehr häufig mit der tatsächlichen Wirklichkeit verwechselt und überhaupt nicht mehr die Möglichkeit von Fehlern in Rechnung stellt.

Es scheint uns durchaus lohnend, darüber nachzudenken, inwieweit eine Illusion der exakten Beschreibung von Wirklichkeit zweckmäßig ist und ob das damit erreichte Kaschieren immer verbleibender Ungenauigkeiten nicht vielleicht als gefährlich einzustufen ist. Möglicherweise muß man über Verfremdungseffekte nachdenken, welche helfen, den Modellcharakter des jeweils

Dargestellten im Bewußtsein zu halten. Es wäre durchaus möglich, dieses Thema hier zu vertiefen, weil neben Erzeugern kommerzieller Software auch Forschungsvorhaben an den Universitäten darauf ausgerichtet sind, sich mit der Erzeugung von Problemlöseumgebungen (PSEs = Problem Solving Environments) zu befassen, die dem Anwender die Kenntnis von Mathematik ersparen sollen [?].

Wir wollen hier aber nur am Rande diese letzten Hochschulaktivitäten streifen und auch auf die (oft nur intern PDEs verwendenden) Produkte der Software-Industrie nicht eingehen. Erstens ist die Anzahl kommerzieller Codes einfach zu groß: So findet man z. B. schon 1988 in [?] über 950 verschiedene Finite-Element-Codes zusammengetragen, und allein im Europort-Projekt [?] werden 33 große Industrie-Codes auf Parallelrechner portiert. Zweitens sind diese Pakete meist auf relativ eng begrenzte Spezialaufgaben zugeschnitten und so in darauf ausgerichtete Problemlöseumgebungen eingebettet, daß allgemeine Aufgabenstellungen damit nur schwer angehbar sind. Drittens wird die Qualität dieser Produkte nicht allein an der Güte gemessen, mit der das beschriebene Stück Wirklichkeit durch PDEs modelliert wird und diese numerisch behandelt werden. Für den Anwender sind die entscheidenderen Qualitäten oft außerhalb des mathematisch numerisch bedeutsamen Teils zu suchen. So kann ein Software-Produkt z. B. dadurch als außerordentlich nützlich empfunden werden, daß es eine fachbezogene (Modell-) Datenbank anbietet oder daß es gängige gesetzliche Bestimmungen für die Modellierung darlegen kann, wobei nicht ausgeschlossen ist, daß der numerische Anteil des Werkzeugs dem Standard von vor 20 Jahren entspricht und nach heutigen Maßstäben eventuell total ineffizient oder möglicherweise sogar unsicher ist.

Es geht in dieser Arbeit stattdessen um Software, mit der partielle Differentialgleichungen und Differentialgleichungssysteme, die als mathematische Gebilde offen vorliegen, numerisch effizient mit modernen Methoden gelöst werden können. Wenn der potentielle Anwender von Lösungsmethoden einem anderen Wissenschaftszweig als der Mathematik entstammt, kann man annehmen, daß er die Gleichungen zur Beschreibung eines Sachverhaltes seines Forschungsbereiches selbst hergeleitet hat. Da man oft auch davon ausgehen muß [?], daß er in seinem eigenen Studium kaum Ausbildung in Theorie und Numerik partieller Differentialgleichungen erhalten hat, und seine Kenntnisse über PDEs daher nicht sehr groß sind, muß man mit zwei sich gegenseitig verstärkenden Schwierigkeiten rechnen: Einmal muß man annehmen, daß ein guter Überblick über die aktuelle Methodenlandschaft fehlt, so daß nicht selten in unbekümmerter Weise die exotischsten Gleichungskombinationen aufgestellt werden, für die dann anschließend nur sehr schwer geeignete Software zu finden ist (wenn es sie denn überhaupt gibt). Zum zweiten verleitet die fehlende Kenntnis dazu, die jeweils zuerst gefundene halbwegs funktio-

nierende numerische Methode zu verwenden, was zu einer ineffizienten und unsicheren Numerik führen kann sowie in der Folge zur Verschwendung personeller Ressourcen und Rechenzeit.

Ungeachtet all dieser Probleme scheint die Modellierung mit partiellen Differentialgleichungen eine der derzeit wichtigsten Methoden in den Ingenieur- und Naturwissenschaften zu sein, und die numerische Behandlung dieser Gleichungen erscheint charakteristisch für das Arbeitsgebiet des „Scientific Computing". So waren etwa die letzten beiden Veranstaltungen der Fachgruppen „Scientific Computing" von DMV und GAMM völlig von der numerischen Behandlung partieller Differentialgleichungen geprägt. Auf dem Workshop „Scientific Computing in der Theoretischen Physik" [?] behandelten 67% der Beiträge PDEs. Von den 30 Arbeiten des Sammelbandes [?] über „Scientific Computing in Chemical Engineering" betreffen 28 Differentialgleichungen und 17 partielle Differentialgleichungen.

Die kontinuierlich zunehmende Bedeutung der partiellen Differentialgleichungen in der numerischen Praxis läßt sich auch aus dem Auftreten von PDE-relevanten Themen in den wissenschaftlichen Zeitschriften entnehmen. So findet man z. B. in der „Numerischen Mathematik" seit 1975 einen Anstieg des PDEs betreffenden Anteils der Arbeiten von 17% über 31% im Jahre 1985 auf 61% im Jahre 1995. In anderen Zeitschriften ist dieser Anstieg nicht so dramatisch aber doch deutlich.

Auch bei der Beratung von Anwendern steigt der Anteil der PDEs ständig. Dabei treten gehäuft Probleme auf, die nicht mehr mit leicht verfügbaren Mitteln lösbar sind. Waren noch vor wenigen Jahren fast alle PDE-Probleme mit dem Verweis auf PLTMG [?] aus der Welt zu schaffen, wird man heute mit Fragen nach der Lösung räumlich mehrdimensionaler Systeme parabolischer Gleichungen oder von Systemen aus Gleichungen unterschiedlichen Typs konfrontiert.

Da der in numerischen Bibliotheken wie NAG oder IMSL vorhandene Fundus an PDE-Lösern schnell nicht mehr ausreicht, versucht man sich bei der Suche nach geeigneter Software natürlich des Internets zu bedienen – und wird hinsichtlich des damit erreichten Informationszuflusses (Information pro eingesetzter Sitzungszeit) sehr enttäuscht, wenn man nicht über geeignete Vorinformation verfügt. Verwendet man nämlich eine der bekannten Suchmaschinen, um mit dem Schlagwort „PDE" eine Suche durchführen zu lassen, so erhält man unter einer die interessanten Informationen zuschüttenden Riesenanzahl von größtenteils nichtssagenden Hinweisen auch solch unerwartete Angebote wie **P**roduct **D**ata **E**xchange oder auch **P**roduct **D**esign **E**ngineering. Unter „partial differential equations" erhält man (je nach Suchmaschine) keine oder fast unendlich viele (160 157) Treffer. Reduziert man diese Menge durch die zusätzliche Forderung nach „software", so geht die

Treffermenge immer noch in die Tausende. Hierunter sind sicher auch viele interessante Informationen, bei der Mehrzahl der Meldungen handelt es sich aber um Home-Pages von Institutionen (etwa Universitäten im angloamerikanischen Netzraum), die sich mit partiellen Differentialgleichungen und – in irgendeiner Form – auch mit Software befassen. Wir haben viel gelesen, aber keine Seite gefunden, die einen allgemeinen Überblick über Software zur Behandlung Partieller Differentialgleichungen bietet. Selbst GAMS [?] und netlib [?] bieten zu diesem Gebiet eher wenig, und auch die MathNet-Links [?] des Berliner Konrad-Zuse-Zentrums helfen nicht viel weiter. Als derzeit besten Einstiegspunkt sehen wir die von Prof. Rüde in Augsburg erstellte Seite zu „Numerischer Software“ auf dem GSCI-Server [?] an. Von hier findet man u. a. schnell Zugang zu Softwarelisten für Finite Elemente [?], Strömungssimulation [?] und Mehrgitter-Methoden [?]. Da die GSCI-Software-Seite aber nicht speziell auf PDEs ausgerichtet ist, kann man von dort nicht gezielt nach Software für konkrete Aufgabenstellungen suchen.

Diesem Defizit wollen wir abhelfen, indem wir – zunächst unter Ausschluß der vollen Öffentlichkeit – probeweise eine PDE-WWW-Seite

http://www.tu-harburg.de/mat/pde/pde.html, Passwort: P&D&E

einrichten und – bei Akzeptanz und entsprechender Lieferung von Information[1] – allgemein zugänglich machen und pflegen.

Zum Zeitpunkt der Sommerschule wird man auf dieser Seite zunächst nur auf die Informationen zugreifen können, die wir bis dahin selbst eruieren und klassifizieren konnten. Erfolgreich kann diese Seite auf Dauer nur durch die Mitarbeit vieler werden.

Im zweiten Abschnitt dieses Aufsatzes versuchen wir darzustellen, in welch unterschiedlichen Organisationsformen PDE-Software heute angeboten wird und stellen einige Löser zusammen. In Abschnitt 3 listen wir einige Wünsche an PDE-Software und ihr Entwicklungsumfeld auf, deren Erfüllung Anwendern und Erstellern die Arbeit erleichtern könnte. Im vierten Abschnitt konkretisieren wir unseren Beitrag zur Verbesserung des Zugriffs auf PDE-Software und -Information.

2 Formen von Software für PDEs

Während der letzten 10 bis 15 Jahre hat sich – einhergehend mit der Weiterentwicklung von Hardware und systemnaher Software – die Form der angebotenen Software für PDEs zum Teil sehr gewandelt.

[1] siehe dazu Abschnitt 4

Die „klassische Form“ numerischer Software ist die der FORTRAN-Routine, die zur Lösung eines Problems in ein eigenes Treiber-Programm eingebunden werden muß. Dabei sind die Differentialgleichungen sowie die Anfangs-, Rand- oder Nebenbedingungen jeweils durch Unterprogramme bereitzustellen. Diese aufrufbare Routinenform wird auch weiterhin gebraucht werden, da mit ihr Subprobleme in einem umfassenden Umfeld – etwa zur Gestaltoptimierung oder bei Kontrollproblemen – gelöst werden müssen.

Standard-Quellen solcher Routinen sind die Zeitschrift „Transactions on Mathematical Software“(TOMS), die zugehörige Programmsammlung [?], mathematische Bibliotheken wie die NAG- [?] und die IMSL-Bibliotheken[2] [?] oder entsprechende Bibliotheken der Rechnerhersteller. Als weitere leistungsfähigere Routinen und Sammlungen von Routinen seien hier KASKADE [?], PDEX1M [?], FIDISOL/CADSOL [?] und VECFEM [?] für elliptische und parabolische Probleme und CLAWPACK [?] für hyperbolische Probleme beispielhaft genannt.

Üblicherweise werden neben den Routinen auch Beispieltreiber mit Beispielproblemen angeboten, die der Benutzer für seine Probleme anpassen kann. Diese Beispiele sind meist so ausgewählt, daß der Löser dafür gute Ergebnisse liefert. Für den Anwender ist es aber auch wichtig zu wissen, wann die Methoden versagen und was typische Problemfälle sind. Auch in welcher Weise die Routinen versagen oder wie sie in Bedrängnis reagieren, ist interessant. Es wäre sicher von Nutzen, wenn in Zukunft durch Beispiele auch problematisches Verhalten demonstriert würde. Besonders bei naiven Benutzern würde die Information über die Fehlbarkeit der gerade benutzten Software sicherlich eine vorsichtigere Wertung der Ergebnisse bewirken.

Sowohl positive als auch negative Beispiele lassen sich sicher sehr viel leichter und schneller durchspielen und verstehen, wenn für ihre Darstellung Graphische Benutzeroberflächen (GUIs = Graphical User Interfaces) bereitstehen. Für die Programme KASKADE und PDEX1M wurde am Konrad-Zuse-Zentrum in Berlin gerade (zusammen mit einigen anderen Routinen) eine gemeinsame GUI [?] mit Hilfe von Tcl/Tk [?] erstellt. Die Erstellung solcher Oberflächen kann eine durchaus komplexe Aufgabe sein [?], und es wäre daher sicher nützlich, wenn für wichtige Problembereiche Programme mit GUIs, die klare Schnittstellen für mathematische Teilaufgaben haben, zusammen mit größeren Problemsammlungen zentral angeboten würden. Zum einen würde sich dadurch die Arbeit an den GUIs stärker bezahlt machen; auch kleinere Arbeitsgruppen könnten dann z. B. ihre Löser für lineare oder

[2] Allerdings muß man gerade zu diesen beiden Bibliotheken sagen, daß sie in Bezug auf PDEs (insbesondere, was hyperbolische Gleichungen angeht) bislang recht „dünn besetzt“ waren. In der NAG findet man z. B. erst ab Mark 17 überhaupt Routinen für hyperbolische Probleme.

nichtlineare Gleichungssysteme in graphischer Umgebung an vielen relevanten Beispielen testen und mit konkurrierenden Methoden vergleichen. Zum anderen ergäbe sich dadurch als sicher sehr nützlicher Nebeneffekt eine automatische Normierung der Schnittstellen der die GUIs nutzenden Programme.

Das Programmpaket CLAWPACK benutzt für die graphische Ausgabe seiner Ergebnisse das MATLAB-System [?]. VECFEM bietet Schnittstellen zu mehreren kommerziellen Visualisierungstools an.

Eine alternative (aber natürlich auch in Kombination mit Graphik mögliche) Steigerung des Bedienungskomforts größerer Routinensammlungen besteht in der Verwendung einer Skriptsprache zur Steuerung der Routinenaufrufe. Eines der ersten Systeme, das eine solche Sprache verwendete, um das zu lösende Problem, die zu verwendende Methode und die Art der Darstellung des Ergebnisses abzurufen, war ELLPACK [?] zur Lösung elliptischer Probleme.[3] In ähnlicher Weise ist PDE/PROTRAN [?] zu bedienen, mit dem elliptische, parabolische, hyperbolische und Eigenwert-Probleme angehbar sind.

Heute haben sicher alle interaktiven Systeme eine mit diesen Skriptsprachen vergleichbare Kommandosprache. Ein neueres Beispiel ist das System UG3 [?] vom Institut für Computeranwendungen III der Universität Stuttgart. UG3 ist eine große Bibliothek aus C++-Routinen für die Entwicklung adaptiver Mehrgitter-Verfahren auf unstrukturierten Gittern. Für zweidimensionale Diffusionsgleichungen, lineare Elastizitätsprobleme, inkompressible Navier-Stokes Gleichungen und dreidimensionale Konvektions-Diffusionsgleichungen kann UG3 direkt als interaktiver Löser mit vollem GUI eingesetzt werden. Für andere Probleme können die Einzelroutinen zur Gittergenerierung und -verfeinerung, zur Diskretisierung, zur (Multigrid-) Lösung und zur graphischen Darstellung getrennt genutzt werden, da sie weitgehend problemunabhängig gehalten wurden. UG3 bietet ferner eingeschränkten Zugang zum parallelen Rechnen.

Das Konzept der Sammlung aufeinander abgestimmter Teilwerkzeuge verfolgen auch die ebenfalls noch jungen Konkurrenzprojekte PETSc [?] und DIFFPACK [?]. PETSc beschränkt sich dabei fast vollständig auf das Angebot von Lösern für schon diskretisierte Systeme (derzeit werden nur Hilfen für Linienmethoden angeboten), konzentriert sich hier aber auch auf deren parallele Realisierung. DIFFPACK bietet zusätzlich einen großen Fundus an Diskretisierungshilfen.

Alle neueren Entwicklungen sind übrigens objektorientiert, was die zu-

[3] Heute wird offenbar nur noch das Nachfolgesystem //ELLPACK [?] angeboten, dessen an ELLPACK anknüpfender Name verdeckt, daß auch andere PDEs damit behandelbar sind (s.u.).

nehmende Bedeutung dieses Programmierparadigmas für moderne Numerik-Entwicklungen unterstreicht. Zumeist verwenden die Pakete für die lineare Algebra auch die einschlägigen effizienten Programm-Sammlungen BLAS [?], LAPACK [?] und ScaLAPACK [?], welche durch optimale maschinennahe Implementierungen große Performance-Gewinne erzielen können. Für die Kommunikation auf Message-Passing Parellelrechnern werden entsprechende Standard-Bibliotheken, etwa MPI [?] mit BLACS [?], verwendet.

Ein weiteres wichtiges Konzept bei moderneren PDE-Lösern ist das der Adaptivität. Eines der ersten adaptiven Pakete zur Mehrgitterlösung elliptischer zweidimensionaler Probleme ist PLTMG [?]. Dem drittgenannten Autor hat PLTMG in seinen ersten Versionen zuverlässig viel Arbeit bei der Beratung von Ingenieuren abgenommen.

Einerseits wird mit adaptiven Methoden eine relative Sicherheit in der Erreichung hinreichender Approximationsgenauigkeiten angestrebt ([?], [?]). Auf der anderen Seite kann man aber oft auch nur mit adaptiven Methoden die Anzahl der für die gewünschte Genauigkeit benötigten Modellfreiheitsgrade soweit senken, daß die Probleme mit den heutigen Ressourcen an Speicher und Rechenleistung überhaupt behandelbar werden [?].

Die oft recht raffinierten Gitter- und Ordnungsadaptionen lassen sich heute noch nicht sehr gut von der konkreten Anwendungsklasse abstrahieren. Dies läßt Aktivitäten, Black-Box-Löser für allgemeine partielle Differentialgleichungssysteme zu entwickeln, oder gar das PDELab-Projekt [?], in dem ein Werkzeug zur Generierung von Problemlöseumgebungen für partielle Differentialgleichungssysteme entwickelt werden soll, als noch recht leise Zukunftsmusik erscheinen. Dies gilt insbesondere unter dem Aspekt, daß solche PSEs nicht gerade bescheiden konzipiert werden [?]:

> A PSE is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerfull processors, and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily. Overall, they create a framework that is all things to all people: they solve simple or complex problems, support rapid prototyping or detailed analysis, and can be used in introductory education or at the frontiers of science.

Sicher hat das derzeitig schon entwickelte //ELLPACK [?] als ein PSE-Prototyp beeindruckende informatische Systemeigenschaften, und ganz sicher werden die in [?] entwickelten und gesammelten Konzepte für die Einbindung

von Leistungen verschiedener Softwarekomponenten in große (naturwissenschaftliche) Software-Systeme einmal sehr nützlich werden. Die Arbeit an aktuellen PDE-Problemen läßt uns aber zweifeln, ob nicht noch zu viele elementare numerische Probleme einer großen Integration von Lösungsmethoden entgegenstehen. So ist die Behandlung von PDE-Systemen immer noch problematisch, und bei der Kopplung von Lösern für Teilsysteme [?] ist man noch weit von so zuverlässigen Algorithmen entfernt, daß an einen automatischen Einsatz gedacht werden könnte. Hinreichend zuverlässige Verfahren zur iterativen Lösung der linearisierten Gleichungen, die durch Diskretisierung aus Reaktions-Diffusions-Systemen entstehen, stehen immer noch aus.

Der Einsatz „Künstlicher Intelligenz“ in Problemlöseumgebungen, wie er u. a. in [?] und [?] geplant wird und wie er in EVE schon teilweise realisiert sein soll [?], ist deshalb vielleicht auf gut bekanntem Terrain (wie bei zweidimensionalen elliptischen Differentialgleichungen) gut möglich. Für allgemeine Probleme wäre wohl sicher erst einmal zu klären, ob es denn überhaupt Alternativen gibt, aus denen ein intelligentes System auswählen könnte.

Wir würden momentan eher dafür votieren, die PDE-Intelligenz statt auf Systemseite auf der Anwenderseite zu fördern.

3 Wünsche an PDE-Software und ihre Erstellung

Zur Erstellung vieler der oben angesprochenen PDE-Lösungswerkzeuge (wie allgemein von numerischer Software) werden Kenntnisse sowohl in angewandter Informatik als auch in der Numerik benötigt. Drückt man es positiv aus, so arbeiten Ersteller numerischer Software interdisziplinär. Realistischer sitzen sie nicht nur zwischen den Disziplinen sondern oft auch „zwischen allen Stühlen“. In Großbritannien und USA z. B. wäre ihr Platz im numerischen Kern der Computer Science, was eine entsprechend positive Bewertung ihrer Arbeit nach sich zöge und auch Zuarbeit durch die Gesamtdisziplin sicherte.

Der erste Wunsch zur PDE-Software ist einer zum wissenschaftlichen Umfeld ihrer Ersteller, daß es nämlich gelänge, die Kommunikation zwischen Informatik und Numerik in Deutschland so sehr zu verbessern, daß eine gemeinsame Fachdisziplin „Numerische Informatik“ von beiden Seiten die Anerkennung erhielte, die ihrer Bedeutung für die Wirtschaft entspricht.

Unser zweiter Wunsch ist, daß zwischen den vielen derzeitigen PDE-Software-Projekten ein wenig Kompatibilität hergestellt wird. Der potentielle Anwender profitiert derzeit nämlich nicht von der Vielfalt des Angebotes, weil er es sich bei der Komplexität der Systeme kaum leisten kann, mehrere

Systeme zu installieren, Erfahrungen damit zu gewinnen und sie schließlich alle auf sein eigentliches Problem anzuwenden. Dabei wäre es nützlich, wenn nicht nur Daten sondern auch Routinen zwischen den Systemen leicht ausgetauscht werden könnten.

Auch unser dritter Wunsch betrifft Kompatibilität, diesmal von Programmiersprachen. Die Diskussionen, ob FORTRAN nun C++ überlegen ist oder umgekehrt, ob man den Preis eines Laufzeitfaktors 5 für das objektorientierte Programmieren zahlen will, ob man den tatsächlich zahlt, ob man überhaupt Objektorientiertheit braucht und ob C++ durch FORTRAN2000 nicht ohnehin wieder abgelöst werden wird, all diese Diskussionen scheinen uns unproduktiv. Die verschiedenen Sprachparadigmen haben alle ihre genuinen Anwendungsbereiche, in denen sie besondere Vorteile haben [**?**]. Es steht deshalb wohl nicht zur Debatte, *die eine* für PDE-Software besonders geeignete Programmiersprache auszuwählen oder zu entwerfen. Es muß darum gehen, konzeptionell einfache Hilfsmittel zur effizienten Kopplung von Code-Stücken in unterschiedlichen Sprachen bereitzustellen.

Abschließend listen wir stichwortartig einige kurze (z. T. schon geäußerte) Wünsche auf:

- Algorithmen sollten mit Hilfe von GUIs schnell zugänglich und testbar gemacht werden. Mitgelieferte Beispiele sollten dabei auch mögliche Fehlfunktionen oder Schwächen des Löser offen darlegen.

- Beispielsammlungen zu verschiedenen Aufgabenklassen mit eigenem Treiber und zugehöriger GUI sowie genauer Schnittstellendefinition für den einzusetzenden Löser sollten erstellt werden, um damit verschiedene Löser vergleichen zu können. Solche Sammlungen können eine Normierungen der oberen Schnittstellen von Lösern bewirken.

- Schnittstellennormierungen nach unten für die durch Löser einzusetzenden Hilfsroutinen (lineare Algebra, nichtlineare Gleichungssysteme) sind ebenso nützlich, da diese Routinen dann leicht gewechselt werden können und die PDE-Löser als Vergleichstreiber für diese Hilfsprobleme dienen können.

- Es sollten – wo immer dies möglich ist – Standard-Routinen (BLAS, LAPACK, ScaLAPACK, BLACS, MPI,...) eingesetzt werden, um eine gute Performance auf der jeweiligen Zielmaschine zu gewährleisten und um die Portierbarkeit der Programme zu verbessern.

- Die Programme sollten zur Erhöhung der Flexibilität einen starken Modularisierungsgrad aufweisen. Module sollten separat dokumentiert sein. Dokumentationen schließen Literaturangaben ein.

- Bei Lösern für nichtlineare PDEs sollte vermehrt automatische Differentiation [?] eingesetzt werden.
- Methoden zur Kopplung von Teilsystemlösern sollten stärker untersucht werden. Hier liegt ein Weg zu Lösern für allgemeine gekoppelte Systeme.
- Interaktive Löser sollten Benutzer (noch) stärker auf mögliche Fehlerquellen hinweisen.

4 Eine PDE-WWW-Seite

Wir wollen die PDE-WWW-Informationen so konzipieren, daß nicht nur PDE-erfahrene Mathematiker diese nutzen können, sondern auch andere Anwender von PDEs [?]. Die Ordnung der Information in einem Entscheidungsbaum [?] scheint uns bei PDEs nicht nützlich, da Interessenten unterschiedlicher fachlicher Herkunft mit unterschiedlichen Grundfragen an das System herangehen, so daß man mehrere Bäume mit entsprechenden Wurzeln zur Verfügung stellen müßte.

Wir haben uns daher entschlossen, die Information mit Stichwörtern zu klassifizieren und den Anwender nach diesen Stichwörtern fragen zu lassen. Zur Unterstützung des Benutzers sollen dabei Stichwortmenüs angeboten werden. Diese sind wieder in Untermenüs gegliedert. Als Hauptmenüs sind derzeit vorgesehen: **Mathematische Klassifikation**, **Lösungsmethode**, **Anwendungsgebiet** und **Rechner-Architektur und -Plattform**.

Für Anfänger ist vorgesehen, daß die Stichworte auf Nachfrage vom System erklärt werden.
Uns ist klar, daß dies alles eine Menge Arbeit sein wird.[4] Darum hoffen wir auch auf die Mitarbeit vieler und bitten um Klassifikationen von PDE-Software sowie die Erstellung von Hilfe-Seiten zu Stichwörtern sowie andere Hinweise. Konstruktive Kritik ist immer gern willkommen.

Literatur

[1] S. Artlich und W. Mackens, *Newton Coupling of Fixed Point Iterations*, in [?]

[4] Hinsichtlich der Klassifikation von Algorithmen der Linearen Algebra, der nichtlinearen Gleichungssysteme, der Optimierung, der Gittererzeugung, der gewöhnlichen Differentialgleichungen und Parallelisierungstechniken erwarten wir Unterstützung von den entsprechenden „Harburger Sommerschulen“ [?].

[2] R. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations; Users's Guide 6.0*, SIAM, Philadelphia 1990

[3] I. Babuska, *Adaptive Mathematical Modelling*, pp. 1-14 in [**?**]

[4] P. Baras, J. Blum, J.C. Paumier, P. Witomski and F. Rechenmann, *EVE: An Object-Centered Knowledge-Based PDE Solver*, pp. 1-18 in [**?**]

[5] C. Bischof and A. Carle, *Users' Experience with Adifor 2.0*, Argonne Preprint ANL/MCS-P589-0496

[6] The Basic Linear Algebra Communication Subprograms, http://math.nist.gov/cgi-bin/gams-serve/list-package-components/BLACS.html

[7] The Basic Linear Algebra Subroutines, http://www.netlib.org/blas/index.html

[8] *CFD Seite, MPI Garching*, http://www.mpa-garching.mpg.de/~tomek/CFD/CFD_codes.html

[9] *CLAWPACK*, http://www.amath.washington.edu/~rjl/clawpack.html

[10] P. Deuflhard, H.C. Hege and E. Sedlmayr (eds.), *Scientific Computing in der Theoretischen Physik*, Collected Abstracts, Tagung Berlin 1994, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Technical Report 94-1 (März 1994)

[11] P. Deuflhard, J. Lang und U. Nowak, *Adaptive Algorithms in Dynamical Simulation*, ZIB Preprint SC 95-16 (May 1995)

[12] DIFFPACK homepage, http://www.oslo.sintef.no/avd/33/3340/diffpack

[13] B. Engquist and T. Smedsaas (eds.), *PDE Software: Modules, Interfaces and Systems*, North-Holland, Amsterdam – New York – Oxford 1984

[14] B. Erdmann, J. Lang und R. Roitzsch, *KASKADE Manual – Version 2.0* , Technical Report TR 93-5, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1993

[15] Europort, *List of Ported Codes*, http://www.gmd.de/SCAI/europort/CODES.html

[16] FIDISOL/CADSOL,
http://www.rz.uni-karlsruhe.de/~numerik/fidisol.cadsol/index.html

[17] *Finite Element Server*,
http://www.engr.usask.ca/~macphed/finite/fe_resources/
fe_resources.html

[18] J.E. Flaherty, P.J. Paslow, M.S. Shephard and J.D. Vasilakis (eds.), *Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia 1989

[19] J. Fröhlich and J. Lang, *Twodimensional Cascadic Finite Element Computations of Combustion Problems*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 96-5, Februar 1996

[20] P.W. Gaffney and E.N. Houstis (eds.), *Programming Environments for High-Level Scientific Problem Solving*, Proceedings of the IFIP TC2/WG 2.5 Working Conference September 1991 at Karlsruhe, North-Holland, Amsterdam 1992

[21] GAMS,
http://gams.nist.gov

[22] W. Hackbusch and G. Wittum (eds.), *Numerical Treatment of Coupled Systems*, Notes on Numerical Fluid Mechanics Vol 51, Vieweg-Verlag, Wiesbaden 1995

[23] Harburger Sommerschulen,
http://www.tu-harburg.de/mat/sommerschulen.html

[24] E.N. Houstis and J.R. Rice, *Parallel ELLPACK: A Development and Problem Solving Environment for High Performance Computing Machines*, pp. 229-241 in [**?**]

[25] E.N. Houstis, J.R. Rice and R. Vichnevetsky (eds.), *Intelligent Mathematical Software Systems*, North Holland, Amsterdam 1990

[26] E.N. Houstis, J.R. Rice and R. Vichnevetsky (eds.), *Expert Systems for Scientific Computing*, North Holland, Amsterdam 1992

[27] The IMSL Homepage,
http://www.vni.com/adt.dir/adt.html

[28] F. Keil, W. Mackens, H. Voß und J. Werther (eds.), *Scientific Computing in Chemical Engineering*, Springer, Berlin 1996

[29] LAPACK: Linear Algebra PACKage,
http://www.netlib.org/lapack/index.html

[30] W. Mackens, S.M. Rump (eds.), *Software Engineering im Scientific Computing*, erscheint 1996 bei Vieweg-Verlag, Wiesbaden

[31] J. Mackerle and B. Frederiksson, *Handbook of Finite Element Software*, Studentlitteratur, Lund 1988

[32] M. Machura, *Problem Solving Environment for Partial Differential Equations: The User Perspective*, pp. 171-191 in [**?**]

[33] Math-Net Links to the mathematical world,
http://elib.zib-berlin.de:88/Math-Net/Links/math.html

[34] MATLAB,
http://www.mathworks.com/matlab.html

[35] *Mehrgitter-Seite*,
http://na.cs.yale.edu/mgnet/www/mgnet.html

[36] H.D. Mittelmann, Decision Tree for Optimization Software,
http://plato.la.asu.edu:80/guide.html

[37] MPI, *A Message-Passing Interface Standard*,
http://www.mcs.anl.gov/mpi/mpi-report-1.1/mpi-report.html

[38] NAG: The Numerical Algorithm Group,
http://www.nag.co.uk:70

[39] netlib,
http://www.netlib.org

[40] U. Nowak, *PDEX1M — A Software Package for the Numerical Solution of Parabolic Systems in One Space Dimension*, pp. 163-169 in [**?**]

[41] U. Nowak, U. Pöhle und R. Roitzsch, *Eine graphische Oberfäche für numerische Programme*, erscheint 1996 in [**?**]

[42] U. Nowak, U. Pöhle, R. Roitzsch, B.E. Sucrow, *Formale Spezifikation des ZIB-GUI mit Hilfe von Graph-Grammatiken*, erscheint 1996 in [**?**]

[43] PDELab-Project,
http://www.cs.purdue.edu/research/cse/pdelab

[44] PETSc — The Portable Extensible Toolkit for Scientific Computing, http://www.mcs.anl.gov/petsc/petsc.html

[45] PSE Homepage – Purdue University, http://www.cs.purdue.edu/research/cse/pses

[46] J.R. Rice and R.F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer Verlag, New York 1985

[47] U. Rüde (ed), *German Scientific Computing* http://Scicomp.math.uni-augsburg.de/scicomp/gsci/software/software.html

[48] ScaLAPACK, http://www.netlib.org/scalapack/index.html

[49] G. Sewell, *Analysis of a Finite Element Method: PDE/PROTRAN*, Springer-Verlag, New York 1985

[50] TOMS: Collected Algorithms of the ACM, http://www.netlib.org/toms/Overview.html

[51] UG3, http://www.ica3.uni-stuttgart.de/∼ug

[52] VECFEM, http://www.uni-karlsruhe.de/ ∼vecfem

[53] S. Weerawarana, *Problem Solving Environments for Partial Differential Equation Based Applications*, PhD Thesis, Purdue University, August 1994

[54] J. van Welij and R.E. Huddleston, *The Applicability of General Software for Real World Problems - A Discussion with Introductory Remarks by J. van Welij and R.E. Huddleston*, pp. 373-377 in [?]

[55] Brent B. Welch, *Practical Programming in Tcl and Tk*, Prentice Hall, Upper Saddle River (1995)