Eine interaktive Matlab-Umgebung für iterative Gleichungslöser

T. Betcke und H. Voss
{t.betcke,voss}@tu-harburg.de

7. Oktober 2002

Zusammenfassung

Viele Anwendungsprobleme führen auf große, dünnbesetzte Gleichungssysteme, die oft nicht mehr effektiv mit Hilfe einer LU-Zerlegung gelöst werden können. Stattdessen werden meistens Krylovraum-Verfahren eingesetzt, bei denen in jeder Iteration eine Matrix-Vektor Multiplikation durchgeführt wird. Diese läßt sich bei dünnbesetzten Matrizen oft mit einem Aufwand von $O(n)$ implementieren, wobei n die Dimension der Matrix ist. Zur Demonstration der verschiedenen Krylovraumverfahren wurde eine interaktive Matlab-Umgebung geschaffen, mit der ohne tiefergreifende Kenntnisse von Matlab verschiedene iterative Löser für symmetrische und unsymmetrische Probleme getestet werden können. Die dabei benutzten Testmatrizen stammen von Matrix-Market, welches eine umfangreiche Sammlung im Internet für Matrizen aus verschiedensten Anwendungsgebieten ist. Dabei werden für jede Matrix detaillierte Informationen wie z.B. die Kondition oder die Struktur geliefert.

1 Installation

Zur Installation unter Windows genügt es, das gepackte Zip-Archiv in ein beliebiges Verzeichnis zu entpacken. Es ist nur darauf zu achten, daß die Verzeichnisstruktur im Archiv beim Entpacken erhalten bleibt. Dies kann z.B. mit dem Programm WinZip sichergestellt werden. Unter Unix steht entsprechend ein tar.gz.-Archiv zur Verfügung. Die Unix-Version wurde unter HP-UX erstellt. Unter anderen Systemen kann es manchmal dazu kommen, daß die Elemente der graphischen Oberfläche leicht verschoben sind. Dies läßt sich aber mit dem GUI-Builder von Matlab leicht beheben. Nach dem Entpacken sind mehrere Verzeichnisse entstanden. Das Haupt-Verzeichnis heißt *iter_solver*. Dieses Verzeichnis enthält sämtliche benötigten Matlab-Skripte. Innerhalb von *iter_solver* ist ein Verzeichnis *matrices*, welches die Unterverzeichnisse *eigene*, *prec*, *symm*, *unsymm*

enthält. Im Verzeichnis *eigene* kann der Benutzer eigene Matrizen speichern, die er untersuchen möchte. Die Verzeichnisse *symm* und *unsymm* enthalten die vorgegebenen symmetrischen bzw. nicht-symmetrischen Matrizen. Im Verzeichnis *prec* können eigene Präkonditionierer gespeichert werden. Die Speicherorte für die Verzeichnisse sind am Anfang der Routine *startSolve.m* definiert und könne auch beliebig verändert werden. Bei einer Installation für mehrere Benutzer ist es z.B. sinnvoll, die vorgegebenen Matrizen nur einmal zentral zu speichern, um Speicherplatz zu sparen. Gestartet wird das Programm von Matlab aus im Verzeichnis *iter_solver* mit dem Befehl *it_environment*.

2 Die Oberfläche

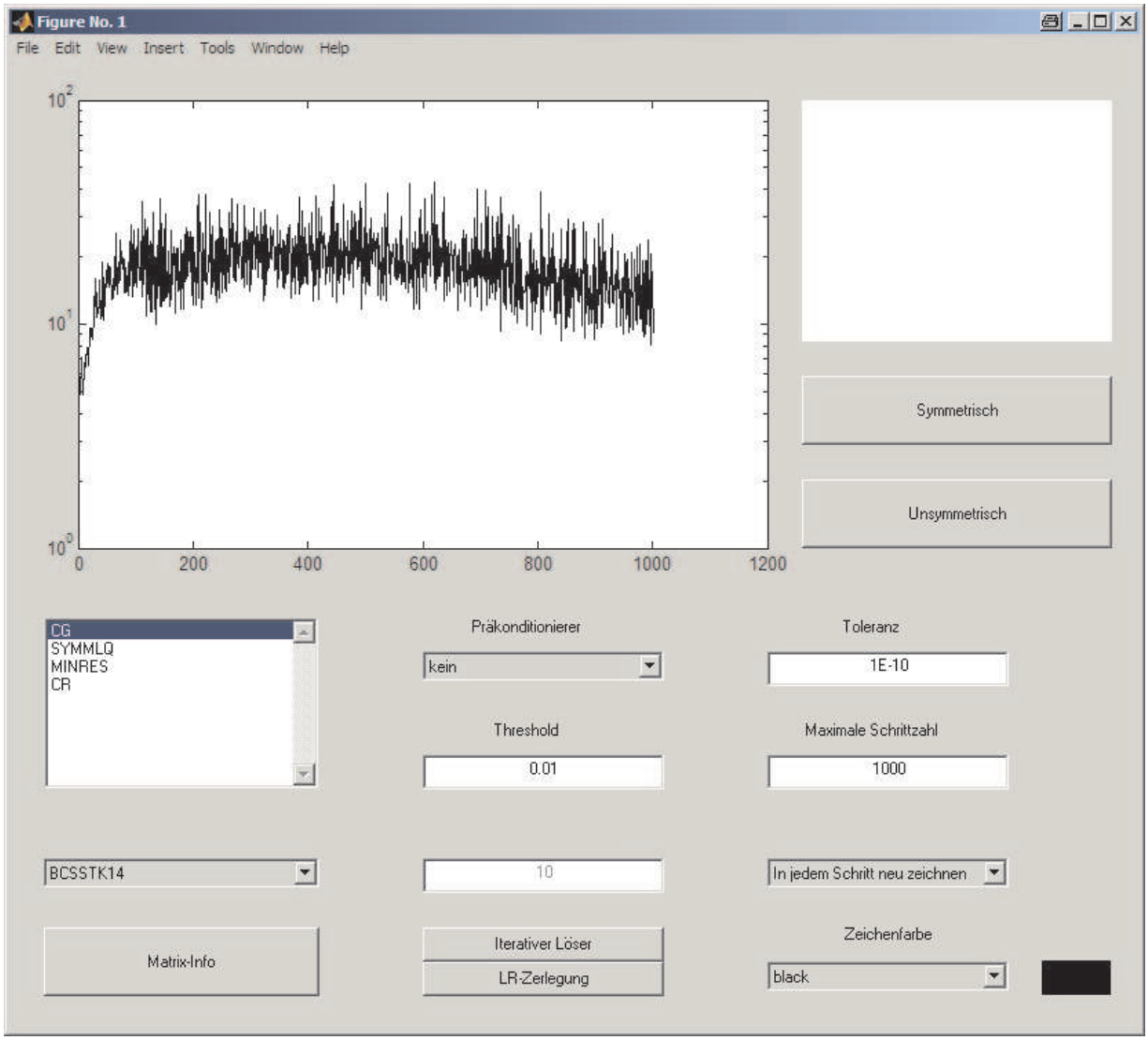

Abbildung 1: Die Programmoberfläche

In Bild 1 ist die Oberfläche des Programms dargestellt. Oben links wird der

Konvergenzverlauf in einem eigenem Fenster dargestellt. In dem kleinen Fenster rechts erscheinen Informationen über die benötigte Zeit für den ausgewählten Löser. Die beiden großen Buttons *Symmetrisch* und *Unsymmetrisch* sind dafür da, zwischen symmetrischen und nichtsymmetrischen Problemen umzuschalten. In der unteren Hälfte des Fensters lassen sich die Optionen für die Löser einstellen. Die Optionen sollem im Folgenden erklärt werden.

- **Auswahl des Lösers**
  Im Fenster links unter dem Konvergenzverlauf werden die in dem Programm implementierten Löser angezeigt. Die Auswahlmöglichkeiten hängen davon ab, ob symmetrische oder unsymmetrische Probleme ausgewählt wurden.
- **Auswahl der Matrix**
  Unterhalb des Fensters für den Löser kann eine Testmatrix ausgewählt werden. Die in dem Programm angegebenen Matrizen sind alle der Internetsammlung 'Matrix Market' entnommen. Es kann auch eine eigene Testmatrix eingegeben werden. Dies soll später erklärt werden.
- **Informationen über die gewählte Matrix**
  Mit dem Button *Matrix Info* lassen sich aus dem Internet detaillierte Informationen über die ausgewählte Matrix abrufen. Dazu muß eine Internetverbindung bestehen und Matlab für die Zusammenarbeit mit einem Webbrowser konfiguriert sein. Dies ist normalerweise standardmäßig der Fall.
- **Präkonditionierer**
  In diesem Feld kann ein Präkonditionierer ausgewählt werden. Standardmäßig steht für symmetrische Löser die unvollständige Cholesky-Zerlegung (ICC) und für nichtsymmetrische Probleme die unvollständige LR-Zerlegung (ILU) zur Verfügung. Für ICC bzw. ILU bestehen 2 Möglichkeiten. Zum einen gibt es die 'No Fill-In' Variante, bei der der Präkonditionierer die gleiche Sparse-Struktur besitzt wie die Matrix. Diese Variante wird mit *ILU* bzw. *ICC* angewählt. Weiterhin gibt es die Möglichkeit mit *ICC(THRESHOLD)* bzw. *ILU(THRESHOLD)* eine Drop-Toleranz anzugeben, mit der bestimmt wird, ab welchem Betrag ein Element bei der unvollständigen Cholesky- oder LR-Zerlegung zu null gemacht wird. In dem Feld *Threshold* kann diese Toleranz angegeben werden. Weiterhin kann ein eigener Präkonditionierer angegeben werden. Hierbei öffnet sich beim Starten des Lösers ein Fenster, bei dem nach der LR-Zerlegung des Präkonditinierers gefragt wird. Das Speicherformat von eigenen Matrizen wird später näher erläutert. Präkonditionerer sind nicht bei allen Lösern implementiert. Wird ein Löser gewählt, bei dem keine Möglichkeit zur Präkonditionierung implementiert ist, kann kein Präkonditionierer ausgewählt werden.
- **Weitere Optionen für den Löser**
  Unter dem Fenster für die Auswahl des Threshold können bei Bedarf weitere Optionen für einzelne Löser eingebunden werden. So wird bei BICGSTAB der ℓ-Parameter hier eingegeben und bei GMRES bestimmt, nach wieviel Schritten ein Neustart erfolgt.
- **Toleranz**

In diesem Feld wird das relative Residuum angegeben, bei dem der Löser gestoppt wird.

- **Maximale Schrittzahl**
  Hier kann die maximale Schrittzahl angegeben werden, die ein Löser hat, um das geforderte Residuum zu erreichen.
- **Iterativer Löser**
  Mit diesem Button wird der gewählte iterative Löser gestartet. Nach Beendigung wird der Konvergenzverlauf und die benötigte Zeit angezeigt. Als rechte Seite des Gleichungssystems wird immer der Vektor bestehend aus lauter einsen gewählt.
- **LR-Zerlegung**
  Mit diesem Button wird eine LR-Zerlegung der gewählten Matrix gestartet. Damit kann die Zeit für die LR-Zerlegung der gewählten Matrix mit der Zeit für einen iterativen Löser verglichen werden.
- **Steuerung der Anzeige des Konvergenzverlaufs**
  Um den Konvergenzverlauf zwischen verschiedenen Lösern zu vergleichen, kann unter dem Feld für die maximale Schrittzahl eingestellt werden, ob das Fenster bei jedem Löser neu gezeichnet, also alte Konvergenzverläufe vergessen werden, oder ob die Grafik eingefroren werden soll. Dies heißt, daß ein alter Konvergenzverlauf im Fenster erhalten bleibt. Dazu kann in dem unteren Feld auch noch die Zeichenfarbe eingestellt werden, um den Konvergenzverlauf zwischen verschiedenen Lösern zu vergleichen.

3 Eingabe eigener Matrizen / Datenformate

Eigene Matrizen können dem Programm mit Hilfe des 'Matrix-Market'-Formats bekannt gemacht werden. Dazu stehen die Funktionen *mmread* und *mmwrite* zur Verfügung. Die Funktion *mmwrite* speichert Matrizen im 'Matrix-Market'-Format und die Funktion *mmread* liest Matrizen im 'Matrix-Market'-Format ein. Das Speichern einer Matrix 'A' kann mit *mmread* folgendermaßen geschehen.

```
mmwrite('A.mtx',A);
```

Der erste Parameter ist der Dateiname, in dem die Matrix gespeichert werden soll. Der zweite Parameter ist der Name der Variable, welche die Matrix enthält.

Das Lesen von Matrizen im 'Matrix-Market'-Format funktioniert ganz ähnlich. Um den Inhalt der Datei 'foo.mtx' in die Variable 'foo' einzulesen, genügt der Befehl

```
foo=mmread('foo.mtx');
```

Um jetzt eine eigen Matrix mit der Oberfläche zu untersuchen, muß diese mit der Endung '.mtx' im Verzeichnis *eigene* gespeichert sein. Wird num im Auswahlfeld für Matrizen der Punkt 'Eigene' gewählt, erscheint beim Starten des Lösers eine Aufforderung den Namen der Matrix einzugeben. Dabei muß der Matrixname ohne die Dateiendung eingegeben werden. Ist also im Verzeichnis *eigene* die Datei 'foo.mtx' gespeichert, muß im Dialogfeld der Name 'foo' eingegeben werden. Entsprechend funktioniert auch die Eingabe eigener Präkonditionierer. Hier werden die Dateien im Verzeichnis *prec* erwaretet.

4 Übungsaufgaben

Die Aufgaben 1 bis 5 beschäftigen sich mit nichtsymmetrischen Problemen. Entsprechend sollten Sie hier im Demonstrationsprogramm auf den Button 'Unsymmetrisch' drücken, um die entsprechenden Verfahren und Matrizen zu erhalten. Für die letzte Aufgabe müssen Sie den Button 'Symmetrisch' drücken, um eine Auswahl an symmetrischen Problemen und entsprechenden Verfahren zu erhalten.

Aufgabe 1: (Thema: Matrix-Market)

Machen Sie sich mit Matrix-Market vertraut. Diese Seite liefert eine umfangreiche Kollektion an Testmatrizen aus verschiedenen Anwendungsbereichen. Zu jeder Matrix, die man sich von dort herunterladen kann, sind zusätzlich umfassende Informationen gespeichert wie z.B. Normen, Konditionsschätzungen, ein Plot der Matrix usw. Matrix-Market finden Sie unter
'http://www.math.nist.gov/MatrixMarket/'. Desweiteren können Sie über alle Testmatrizen im Matlab-Demonstrationsprogramm Infos von Matrix-Market erhalten, indem Sie auf den Button 'Matrix-Info' drücken.

Aufgabe 2: (Thema: Gauß im Vergleich zu iterativen Verfahren)

Auf den ersten Blick scheinen iterative Verfahren gegenüber der Gauß-Elimination viele Vorteile zu haben. Ist die Matrix dünn besetzt, benötigt eine Matrix-Vektor-Multiplikation nur $O(n)$ Operationen, und oft kommt man mit viel weniger als n Schritten mit dem iterativen Verfahren zu einer guten Approximation der Lösung. Doch ist die Sache leider nicht ganz so einfach. In vielen Anwendungsfällen sind optimierte Varianten der Gauß-Elimination noch schneller als gängige iterative Löser. Dies soll im Folgenden näher betrachtet werden.
Benutzen Sie zur Untersuchung die Matrix 'ck400'. Vergleichen Sie die Zeit, die Sie mit der LR-Zerlegung benötigen, mit der Zeit bei verschiedenen iterativen Lösern. Probieren Sie die Löser auch mit Präkonditionierung aus und versuchen Sie, eine Kombination aus Präkonditionierer und iterativen Löser zu finden, mit der Sie möglichst wenig Zeit zum lösen benötigen.

Aufgabe 3: (Thema: Iterative Löser im Vergleich)

In den Vorträgen haben Sie verschiedene iterative Löser kennengelernt. Dabei stellt sich natürlich die Frage, welcher Löser am besten ist. Wie so oft gibt es aber auch hier keine klare Antwort. Schauen Sie sich dazu die Matrizen 'fs_541_1' und 'bfw398a' einmal genauer an und lösen Sie sie mit verschiedenen Verfahren. Versuchen Sie herauszufinden, welches Verfahren bei den Matrizen am wenisten FLOPS bei einer Genauigkeit von 1E-10 benötigt und vergleichen Sie auch die verschiedenen Verfahren bei beiden Matrizen untereinander.

Zu diesem Thema gibt es übrigens einen Artikel von Nachtigal, Reddy und Trefethen (*How fast are nonsymmetric matrix iterations ?*), in dem sie die drei Verfahren CGNR, GMRES und CGS vergleichen und Matrizen konstruieren, bei denen jeweils ein Verfahren besser ist als die beiden anderen Verfahren. Dies hat zur Folge, daß sich nicht allgemein bei nichtsymmetrischen Matrizen sagen läßt, welches Verfahren am besten ist.

Aufgabe 4: (Thema: QMR-2 und QMR-3 im Vergleich.)

Im Vortrag wurde Ihnen das QMR-Verfahren mit 2-Term-Rekursion und mit 3-

Term-Rekursion vorgestellt und gesagt, daß die 2-Term-Rekursion oft stabiler ist. Dies soll jetzt einmal demonstriert werden. Untersuchen Sie dazu die Matrix 'fs_680_1' mit den beiden QMR-Verfahren bei einer Toleranz von 1E-5. Danach sollten Sie einmal die Residuenverläufe der beiden Verfahren für die Matrix 'cavity01' vergleichen.

Aufgabe 5: (Thema: Präkonditionierung.)

Richtig beliebt sind iterative Verfahren erst mit dem Aufkommen von guten Präkonditionierern geworden. Die möglichen Verbesserungen durch Präkonditionierung sollen jetzt einmal genauer untersucht werden. Schauen Sie sich dazu zunächst die Matrix 'fs_680_1' genauer an und probieren Sie verschiedene Präkonditionierer aus. Variieren Sie auch den Parameter 'Threshold'. Vergleichen Sie nun die benötigte Zeit der Verfahren mit Präkonditionierung und ohne Präkonditionierung. Untersuchen Sie nun die Matrix 'impcol_d'. Was passiert, wenn Sie als Präkonditionierer 'ILU' auswählen ? Versuchen Sie einen Parameter für den Threshold zu finden, für den die Wahl des Präkonditionierers 'ILU(Threshold)' zum Erfolg führt.

Hinweis: Wem die Bedeutung der unvollständigen LU-Zerlegung noch unklar ist, findet in der Matlab-Hilfe einige Informationen darüber. Die entsprechende Matlab-Funktion heißt 'LUINC'.

Aufgabe 6: (Thema: Symmetrisch indefinite Matrizen.)

Für symmetrisch positiv definite Matrizen ist CG die beste Wahl. Sehr oft läßt sich CG auch für indefinite Probleme anwenden. Desweiteren stehen auch verschiedene andere Verfahren für indefinite Systeme zur Verfügung. Im Demonstrationsprogramm finden Sie eine Reihe symmetrisch positiv definiter aber auch symmetrisch indefiniter Matrizen. Machen Sie sich mit Hilfe dieser Testmatrizen über Verfahren für symmetrische Matrizen vertraut. Für positiv definite Matrizen können Sie beispielsweise mit der Matrix 'gr_30_30' beginnen, die einen diskreten Laplace-Operator auf einem 30×30 Gitter darstellt. Für indefinite Systeme ist die Matrix BCSSTK26 empfehlenswert. An dieser Matrix können Sie auch gut sehen, daß Präkonditionierung manchmal unerläßlich ist.

5 Übersicht über die Testmatrizen

Symmetrische Matrizen

Name	Dimension	Kondition (geschätzt)	Definitheit
BCSSTK14	1806	1.3E10	positiv definit
BCSSTK16	4884	65	positiv definit
BCSSTK19	817	2.8E11	indefinit
BCSSTK21	3600	4.5E7	indefinit
BCSSTK22	138	1.7E5	indefinit
BCSSTK23	3134	6.9E12	indefinit
BCSSTK24	3562	65	indefinit
BCSSTK26	1922	2.3E8	indefinit
BCSSTK28	4410	65	indefinit
BCSSTM20	485	2.6E5	positiv definit
gr_30_30	900	3.8E2	positiv definit
nos1	237	2.5E7	positiv definit
nos5	468	2.9E4	positiv definit
nos7	729	4.1E9	positiv definit
s1rmq4m1	5489	3.21E6	positiv definit
s2rmq4m1	5489	1.15E8	positiv definit

Unsymmetrische Matrizen

Name	Dimension	Kondition (geschätzt)
west0067	67	3E2
west0132	132	6.4E11
west0497	497	1.4E12
watt__1	1859	5.38E9
sherman2	1080	1.4E12
jpwh_991	991	7.3E2
impcol_d	425	1.9E3
gre_115	115	1.5E2
fs_541_1	541	7.6E4
bfw398a	398	7.6E3
bfw782a	782	4.6E3
cavity01	317	6.9E4
cavitiy11	2597	Nicht verfügbar
ck400	400	2.2E6
dw2048	2048	5.3E3
dwb512	2500	4.5
fs_680_1	680	2.1E4

6 Quellenangaben für die Löser

Der BICGSTAB-Löser stammt von G. L. Sleijpen und kann unter 'www.math.uu.nl/people/sleijpen' heruntergeladen werden. 'QMR-2' ist erhältlich unter 'www.netlib.org/templates' als der Routinen in den 'Templates for the Solution of Linear Systems'. Die folgenden Löser wurden von H. Voss implementiert.

Löser	Dateiname
CG	cg_p.m
MINRES	minres.m
CR	cr.m
GMRES	gmresm_lp.m
BICG	bicg_lp.m
TFQMR	tfqmr_lp.m
QMRBCGSTAB	qmrbcgst_lp.m
CGNR	cgnr.m
CGS	cgs_lp.m
QMR-3	qmr_lp.m