Numerische Lineare Algebra
mit einer Einführung in MATLAB

Gerhard Opfer WS 1998/1999

Bogen 9: Ausgabetermin am 17. Dezember 1998

Abgabetermin: am 7. Januar 1999

Inhalt von Woche 9: Fehlerabschätzung für die Nachiteration bei Rechnung mit einfacher Genauigkeit, Beispiele für die Wirkung der Nachiteration und der Äquilibrierung (Präkonditionierung mit Diagonalmatrizen), grundlegende Aufgaben der numerischen linearen Algebra, sog. `BLAS` (Basic Linear Algebra Soubroutines), `BLAS1` $(a\mathbf{x}+\mathbf{y})$, `BLAS2` $(\mathbf{Ax}+\mathbf{y})$, `BLAS3` $(\mathbf{AB}+\mathbf{C})$, Verhältnis von Zahl der Operationen zu Zahl der Speicherzugriffe, Beschleunigung der Matrix-Matrix-Multiplikation durch Blockbildung, Strassens Algorithmus.

Präsenzaufgaben:

Aufgabe P25. *Nachiteration* besteht aus den folgenden Schritten:

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{r} = \mathbf{Ax} - \mathbf{b}, \quad \mathbf{Ad} = \mathbf{r}, \quad \mathbf{x}_{\text{Nachit}} = \mathbf{x} - \mathbf{d}. \tag{9.1}$$

Man führe die Nachiteration aus an verschiedenen, bereits bekannten Matrizen hoher Kondition und vergleiche $\mathbf{x}$ mit $\mathbf{x}_{\text{Nachit}}$. Benutzen Sie Hinweise aus Aufgabe 16.

Aufgabe P26. Prüfen Sie an Beispielen, ob die Bedingung (9.2) für die komponentenweise Rückwärtsstabilität der Nachiteration beim Gleichungssystem $\mathbf{Ax} = \mathbf{b}$ zutrifft. Die Größe ε ist dabei die Maschinengenauigkeit. Berechnen Sie die links in (9.2) stehende Größe.

$$\kappa_\infty(\mathbf{A})\frac{\max_j(|\mathbf{A}|\,|\mathbf{x}|)_j}{\min_j(|\mathbf{A}|\,|\mathbf{x}|)_j} < \frac{1}{\varepsilon} \tag{9.2}$$

Aufgabe P27. Vergleichen Sie das Lösen eines 2×2 Gleichungssystems nur nach Stabilitätsgesichtspunkten bei Lösung nach Gauß und nach Cramer. Hinweis: Wählen Sie eine fast singuläre Matrix $\mathbf{A}$ und eine rechte Seite $\mathbf{b}$ so, daß $\mathbf{b} \approx \mathbf{A}(:,2)$. Folgerung: Cramers Regel ist nicht rückwärts stabil.

Hausaufgaben:

Aufgabe 17. (i) Beweisen Sie durch vollständige Induktion, daß Strassens Algorithmus (Programm 9.1, s. u.) die Matrixmultiplikation $\mathbf{C} = \mathbf{A} * \mathbf{B}$ richtig ausführt. (ii) Stellen Sie mit dem angegebenen Programm die Rechenzeiten $t_{\texttt{matlab}}(n)$ und die Zahl der Operationen $f_{\texttt{matlab}}(n)$ (in `flops`) fest für Matrizen der Größe $n = 2^m$, $m = 0, 1, \ldots, 6$ und vergleichen Sie das Ergebnis $f_{\texttt{matlab}}(n)$ auch mit $2n^3$ (bilden Sie die Quotienten). Falls ein schneller Rechner zur Verfügung steht, können Sie auch $n = 128$ versuchen. Formulieren Sie als Ergebnis eine Tabelle. Schreiben Sie auch den benutzten Rechnertyp auf. Beachten Sie, daß die Befehlssequenz `flops,tic,toc,flops` bereits 18 `flops` in Rechnung stellt. Verwenden Sie also die Reihenfolge `tic,flops,...,flops,toc`. (iii) Bauen Sie in das Programm 9.1 einen globalen Multiplikationszähler ein, und schlüsseln Sie die `flops` auf in `mult` und (`add, sub`). (iv) Wiederholen Sie alles mit komplexen Matrizen. Die Zahlen $2n^3$ sind dann durch $8n^3$ zu ersetzen. Machen Sie wg. $8n^3$ vorher einige Tests.

Aufgabe 18. Zur Multiplikation von $\mathbf{C} = \mathbf{A} * \mathbf{B}$ nach Strassens Algorithmus werden $T(n)$ Operationen (`mult, add, sub`) benötigt, wenn die angegebenen Matrizen die Ordnung $n = 2^m$, $m = 0, 1, \ldots$ haben. Für $T(n)$ gilt

$$T(n) = 7\,T\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2, \quad T(1) = 1. \tag{9.3}$$

(i) Leiten Sie eine explizite Formel für $T(n)$ her. Hinweis: Betrachten Sie $\tilde{T}(m) := T(2^m)$. (ii) Die übliche Matrixmultiplikation benötigt $\tau(n) := n^2(2n-1)$ Operationen. Bestimmen Sie das kleinste $n > 1$ mit

$T(n) \leq \tau(n)$. (iii) Wie ist der Unterschied zwischen $T(n)$ und den (in der vorigen Aufgabe) experimentell festgestellten $f_{\texttt{matlab}}(n)$.
Es ist zu beachten, daß `matlab` für eine (reelle) Matrix∗Matrix-Operation $2n^3$ Operationen berechnet, weil Skalarprodukte mit $2n$ Operationen und nicht mit $2n-1$ Operationen berechnet werden. Tatsächlich braucht ein reelles Skalarprodukt n `mult` und $n-1$ `add`, also $2n-1$ Operationen, wie angegeben.

Programm 9.1: Strassens Algorithmus als `matlab`-Programm

```
function [C]=strassen(A,B,n);
if n==1, C=A*B; else
  m=n/2;
  A11=A(1:m,1:m); A12=A(1:m,m+1:n); A21=A(m+1:n,1:m); A22=A(m+1:n,m+1:n);
  B11=B(1:m,1:m); B12=B(1:m,m+1:n); B21=B(m+1:n,1:m); B22=B(m+1:n,m+1:n);
  P1=strassen(A12-A22,B21+B22,m); P2=strassen(A11+A22,B11+B22,m);
  P3=strassen(A11-A21,B11+B12,m); P4=strassen(A11+A12,B22,m);
  P5=strassen(A11,B12-B22,m);     P6=strassen(A22,B21-B11,m);
  P7=strassen(A21+A22,B11,m);     C11=P1+P2-P4+P6;C12=P4+P5;C21=P6+P7;
  C22=P2-P3+P5-P7;                  C=[C11 C12;C21 C22];
end; %if
```

Zusatzaufgabe: Lassen Sie das folgende Programm 9.2 laufen und versuchen Sie, es zu verstehen.

Programm 9.2: Kleines Weihnachtsprogramm

```
%Ein matlab-Programm zum Weihnachtsfest 1998 f\"ur die H\"orer der Vorlesung
"Numerische lineare Algebra mit einer Einf\"uhrung in matlab".

c = pi*(1+sqrt(5))/2;  x=0:70000;
z=cumsum(exp(i*(c*x.*x+.4))); plot(z,'g'); axis image; hold on;
zz = z(1:700:70000); plot(zz,'ro','MarkerSize',4,'LineWidth',2);
[M,I] = max(imag(z));
plot(z(I),'y*','MarkerSize',10,'LineWidth',2);
axis('off');
```

Hinweis: Beide Programme sind, wie bekannt, aus dem Internet abrufbar.

(Version of 17. 12. 1998 at 494 minutes since midnight, machine used: ultra)