Institut für Angewandte und Numerische Mathematik
Prof. Dr. Christian Wieners, Dipl.-Math. techn. Daniel Maurer

Numerische Mathematik 1 — Programmierpraktikum
Übungsblatt 4 vom 12. Dezember 2012
Wintersemester 2012/13

Programmieraufgabe 7 (Vektoriteration) (Abgabe)
Die Suchmaschine *Google* verwendet das sogenannte *PageRank*[1]-Verfahren, um die Sucheergebnisse einer Suchanfrage nach ihrer Wichtigkeit zu sortieren. Die Wichtigkeit einer Seite wird dabei über folgende Kriterien definiert:

- wie viele andere Seiten verlinken auf die Seite
- wie *wichtig* sind die Quellen, die auf diese Seite verlinken.

Das zweite Kriterium hat rekursiven Charakter, denn die Wichtigkeit einer Seite beeinflusst die Wichtigkeit aller von ihr verlinkten Seiten etc.

Modelliert wird dies anschaulich mit dem Surfverhalten eines *Random Surfers*[2], der (startend von einer zufälligen Seite) mit gleicher Wahrscheinlichkeit auf eine der von dieser Seite verlinkten Seite weitersurft. Die Wichtigkeit einer Seite ist dann die relative Häufigkeit, mit der der *Random Surfer* diese Seite besucht. Dies hat folgende Effekte:

- Eine Seite, die von vielen anderen Seiten verlinkt wird, hat eine hohe Wahrscheinlichkeit *"angesurft"* zu werden. Vielfach verlinkte Seiten werden also oft von dem *Random Surfer* besucht. Sie sind in diesem Sinne *wichtig*.
- Links von wichtigen Seiten haben einen höheren Wert als Links von unwichtigen Seiten, weil der *Random Surfer* oft in wichtigen Seiten verweilt.

Damit dieser Prozess nicht in einer Sackgasse (eine Seite ohne weitere Links) endet, wird das Modell erweitert. Durch "künstliches" Erzeugen von Links von einer Sackgasse zu allen anderen Seiten (einschließlich der Sackgasse selbst) wird dieses Problem behoben.

Außerdem kann es passieren, dass der *Random Surfer* an einer bestimmten Stelle "festhängt" – zum Beispiel, wenn sich zwei Seiten nur gegenseitig verlinken. Dadurch würde sich die relative Häufigkeit der besuchten Webseiten auf diese beiden Seiten verteilen – ein unerwünschter Effekt. Um dieses Problem zu umgehen, erlauben wir sog. *"teleports"*, das sind zufällige Sprünge des *Random Surfers* von der aktuellen Seite zu einer beliebigen Seite. Ein Parameter $\alpha \in]0,1[$ steuert dieses Verhalten. In jedem Zeitschritt folgt der *Random Surfer* mit Wahrscheinlichkeit $1-\alpha$ einem Link und springt mit Wahrscheinlichkeit α zu einer beliebigen Seite.

[1] benannt nach Larry Page, dem Erfinder und Mitgründer von Google

[2] Siehe z.B. in *"Google page rank and beyond"* von Amy N. Langville und Carl Dean Meyer, Princeton University Press, 2006

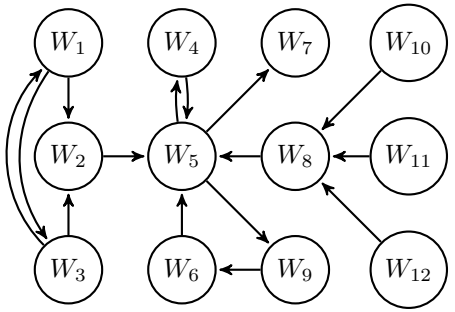
Die Linkstruktur kann durch einen gerichteten Graphen bzw. durch dessen Adjazenzmatrix dargestellt werden. Das Surfverhalten des *Random Surfers* kann dann durch eine Übergangsmatrix A dargestellt werden, deren Einträge a_{ij} die Wahrscheinlichkeit eines Übergangs von Seite i zu Seite j angeben.

Mathematisch gesprochen ist der *Random Surfer* ein diskreter Markov-Prozess. Aufgrund der Modellerweiterung mit den *"teleports"* ist die Übergangsmatrix A

- *stochastisch*, d.h. die Zeilensummen sind 1,
- *irreduzibel*, d.h. jede Seite kann von jeder anderen Seite aus erreicht werden,
- *aperiodisch*, d.h. der Graph enthält keine Zyklen mit fester Periodenlänge > 1.

Die Matrix besitzt daher eine sog. stationäre Verteilung, d.h. die Aufenthaltswahrscheinlichkeit des *Random Surfers* und somit auch relative Häufigkeit der Besuche jeder Seite konvergiert gegen einen festen Wert, wenn der *Random Surfer* lange genug simuliert wird.
Diese stationäre Verteilung ist genau der (eindeutige) Linkseigenvektor der Übergangsmatrix zum Eigenwert 1! Außerdem ist 1 der betragsgrößte Eigenwert und der einzige Eigenwert mit Betrag 1. Die Voraussetzungen der Potenzmethode zur Bestimmung der stationären Verteilung sind also gegeben. Dieser Eigenvektor wird auch PageRank Vektor genannt, der k-te Eintrag entspricht der Aufenthaltswahrscheinlichkeit des *Random Surfers*.

Wir betrachten nun folgenden gerichteten Graphen, der die Linkstruktur der Seiten $W_1, \ldots, W_{12}$ darstellt:



Für diesen Graphen sollen Sie mithilfe des PageRank Verfahrens die Wichtigkeit der Seiten W_1 bis W_{12} ermitteln. Gehen Sie dazu folgendermaßen vor:

(a) Stellen Sie die Adjazenzmatrix L des Graphen auf.

(b) Erzeugen Sie aus L eine stochastische, irreduzible, aperiodische Matrix A_α durch

(i) künstliche Links in Sackgassen, das heißt, suchen Sie nach Nullzeilen und "verlinken" Sie diese Seite zu allen anderen Seiten. Hierzu ist der Befehl `sum(X,DIM)` hilfreich.

(ii) Normalisierung der Zeilensummen. Verwenden Sie dazu den Befehl `repmat` auf einen Vektor der die Zeilensummen beschreibt, erzeugen damit eine Matrix der Größe von A und teilen die Matrix A komponentenweise.

(iii) Anwendung des *"teleport"*-Prinzips für $\alpha \in]0,1[$, das heißt mit Wahrscheinlichkeit $1-\alpha$ wird der *Random Surfer* einem Link folgen und mit Wahrscheinlichkeit α wird er irgendeine zufällige Seite aufrufen.

(c) Die Wahrscheinlichkeitsverteilung $v^{(n+1)}$ der Position des *Random Surfers* zum Zeitpunkt $n+1$ ergibt sich durch $v^{(n+1)} = v^{(n)} A_\alpha$. Eine stationäre Verteilung ist gegeben bei $vA_\alpha = v$. Wenden Sie zur Berechnung des Eigenvektors v zum Eigenwert 1 die Potenzmethode für den Startvektor $v^{(0)} = \frac{1}{12}(1,\ldots,1)$ an. Wählen Sie ein geeignetes Abbruchkriterium. Achten Sie dabei vor allem auf eine "sichere" Abbruchbedingung, die in jedem Fall irgendwann eintritt. Verwenden Sie die Parameter $\alpha_1 = 0.1$, $\alpha_2 = 0.4$ und $\alpha_3 = 0.6$ und vergleichen Sie die Ergebnisse.
Beurteilen Sie den Einfluss von α auf die Bewertung von W_2 und W_8.

(d) Da der Eigenwert $\lambda = 1$ bekannt ist, kann der Eigenvektor auch durch die Gleichung $vA = \lambda v = v = vI$ durch die Bestimmung des Nullraums von $(A-I)^T$ direkt berechnet werden (warum?). Berechnen sie den Eigenvektor zu $\lambda = 1$ auf diese Art (verwenden Sie dazu den Befehl `null(X)`) und vergleichen Sie die Laufzeiten beider Varianten.

Hinweis zur Programmierung in MATLAB: MATLAB verwendet Programmbibliotheken, die auf die Ausführung von Vektor- und Matrixoperationen optimiert sind. Verwenden Sie daher beim Programmieren mit MATLAB Vektor-Operationen anstelle von Schleifen, wo immer dies möglich ist (Dies gilt natürlich nicht für die Iterationsschleife der Potenzmethode in der obigen Aufgabe, ist aber ansonsten überall möglich!).

Abgabe der Programmieraufgaben:

Die bearbeiteten Programmieraufgaben können Sie mittwochs von 15:00 - 17:00 im Rechnerpool Geb. 01.93 (Kronenstr. 32, Raum 101) vorführen und erläutern. Dort haben Sie auch die Möglichkeit, unter Hilfestellung zu programmieren. Die Abgabe dieses Programmierblattes ist bis spätestens **Mittwoch, den 23. Januar 2013** möglich.

Programmieraufgabe 8 (Gerschgorin-Kreise)

(a) Schreiben Sie eine Prozedur `plotgerschgorin(A)`, die für eine quadratische Matrix A die Gerschgorin-Kreise gemeinsam mit den Eigenwerten (als Punkte) plottet.
Tipp: Mit `xplot(1:size(t,2),n) = xM(n) + r(n)*cos(t)` und `yplot(1:size(t,2),n) = r(n)*sin(t)` können Sie nacheinander die Gerschgorin-Kreise beschreiben, so dass diese mit `plot(xplot,yplot)` geplottet werden können. Dabei definiert `xM(n)` den Mittelpunkt und `r(n)` den Radius. Wählen Sie als Vektor `t = 0:0.001:2*pi`. Einen Plot können sie weiter benutzen, indem Sie `hold on` verwenden, so dass im gleichen Plot auch die Eigenwerte geplottet werden können. Beachten Sie, dass die x-Achse den Realteil und die y-Achse den Imaginärteil beschreibt.

(b) Testen Sie Ihre Prozedur mit den Matrizen

$$A_1 = \begin{pmatrix} 3 & 1 & 0 & 3 \\ 1 & 0 & 2 & 0 \\ 0 & 2 & -6 & 0 \\ 3 & 0 & 0 & -7 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 4 & 2 & 1 & 0 \\ -1 & 5 & 1 & 2 \\ -1 & -1 & 7 & 2 \\ 1 & 1 & 0 & -3 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 1 & 2 & 0 \\ 1 & 4 & 4 \\ 0 & \frac{1}{2} & 4 \end{pmatrix}.$$

(c) Welche der folgenden Aussagen für eine Matrix A sind richtig?

(i) Jeder Gerschgorin-Kreis von A enthält genau einen Eigenwert von A.

(ii) Jeder Gerschgorin-Kreis von A enthält mindestens einen Eigenwert von A.

(iii) Gibt es einen Gerschgorin-Kreis K von A, der keinen Punkt mit den anderen Gerschgorin-Kreisen gemeinsam hat, so enthält K genau einen Eigenwert von A.

(iv) Hat A nur reelle positive Eigenwerte, so liegen alle Gerschgorin-Kreise in der rechten Halbebene.

(v) Ist A strikt diagonaldominant mit positiven Diagonalelementen, dann liegen alle Eigenwerte in der rechten Halbebene.

(vi) Ist A strikt diagonaldominant, dann ist A regulär.

Service/Material:

Unter http://www.math.kit.edu/ianm3/lehre/numa12012w/ finden Sie die Homepage zur Vorlesung. Dort finden Sie neben den aktuellen Übungsblättern auch aktuelle Informationen zum Vorlesungsbetrieb.

Sprechstunden:

Prof. Dr. Christian Wieners: Dienstag, 09.30-10.30 Uhr und nach Vereinbarung
Dipl.-Math. techn. Daniel Maurer: Mittwoch, 14.30-15.30 Uhr und nach Vereinbarung