Abgabe bis Di 19. Januar 12:00 im Kasten bei Raum A306

Der Quellcode muss eingereicht werden. Der Code muss unter Python 2.7 oder 3.x laufen. Schreiben/Kommentieren Sie Ihr Programm so, dass die Matrix, die Sie zerlegen, leicht geändert werden kann. Aus dem Quellcode muss hervor gehen, dass Sie den Crout-Algorithmus verwenden. **(25 Zusatzpunkte)**

Blatt 9

Programmieraufgabe: Berechnen der Determinante mit einer LU Zerlegung

Zur Berechnung einer Determinante ist die Berechnung mit der Leibnitzformel unbrauchbar, da die Anzahl der Permutationen über die man summieren muss wächst wie $n!$ wobei n die Dimension des Vektorraumes ist.

Eine weitere Methode besteht in der sogenannten LU-Zerlegung einer Matrix. Ein Beispiel für eine LU Zerlegung ist der Jordan-Gauß-Algorithmus. Man erhält so eine obere (upper) Dreiecksmatrix U und eine untere (lower) Dreiecksmatrix L so dass

$$LU = A.$$

Genau genommen ist L das inverse der Operationen. Das inverse einer unteren Dreiecksmatrix ist aber wieder eine Dreiecksmatrix.

Hat man nun diese Zerlegung gefunden ist das berechnen der Determinante leicht. Dazu muss man zunächst zeigen, dass die Determinante einer Dreiecksmatrix das Produkt der Diagonalelemente ist.

a) (1 Zusatzpunkt) Zeigen Sie dass für obere bzw untere Dreiecksmatritzen U bzw. L gilt

$$\det L = \prod_{i=1}^{n} l_{ii} \qquad\qquad \det U = \prod_{i=1}^{n} u_{ii}.$$

b) (1 Zusatzpunkt) Zeigen Sie nun wie man mit dem Determinantenproduktsatz aus der LU Zerlegung die Determinante einer Matrix erhält.

Hier sollen Sie den *Crout*-Algorithmus zur Bestimmung einer LU-Zerlegung implementieren. Bei diesem werden die Einträge besonders effizient bestimmt. Um den Crout-Algorithmus herzuleiten definieren wir

$$L = \begin{pmatrix} \alpha_{00} & 0 & \dots \\ \alpha_{10} & \alpha_{11} & \ddots \\ \vdots & \vdots & \ddots \end{pmatrix} \qquad R = \begin{pmatrix} \beta_{00} & \beta_{01} & \dots \\ 0 & \beta_{11} & \dots \\ \vdots & \ddots & \ddots \end{pmatrix} \qquad A = \begin{pmatrix} a_{00} & a_{01} & \dots \\ a_{10} & a_{11} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Das der LU-Zerlegung zu Grunde liegende Gleichungssystem lässt sich also schreiben als

$$\begin{pmatrix} \alpha_{00} & 0 & \dots & & \\ \alpha_{10} & \alpha_{11} & 0 & \dots & \\ \alpha_{20} & \alpha_{21} & \alpha_{11} & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} \beta_{00} & \beta_{01} & \beta_{02} & \dots \\ 0 & \beta_{11} & \beta_{12} & \dots \\ \vdots & 0 & \beta_{22} & \dots \\ \vdots & \vdots & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & \dots \\ a_{10} & a_{11} & a_{12} & \dots \\ a_{20} & a_{21} & a_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{1}$$

Führt man nun das Matritzenprodukt aus, entsteht für jeden Eintrag ein Gleichungssystem. Da jedoch mehr α und β als a existieren ist dieses Gleichungssystem überbestimmt. Man wählt nun einfach $\alpha_{ii} = 1$. Das so entstehende Gleichungssystem ist nun bestimmt[1].

Für Crouts Alorithmus bestimmt man spaltenweise zunächst die β und dann damit die α. Der Algorithmus geht folgendermaßen

1. Gehe die spalten nacheinander durch. Der Spaltenindex sei j. Beginne bei 0.

[1] Zumindest wenn die Matrix nicht singulär ist und man Pivoting betreibt. Später mehr dazu.

2. Berechne für alle Zeilen in der j-ten Spalte die β. Der Zeilenindex sei i. Die β erhält man für $i \leqslant j$ durch

$$\beta_{ij} = a_{ij} - \sum_{k=0}^{i-1} \alpha_{ik}\beta_{k,j}. \tag{2}$$

   Man fängt bei $i = 0$ an und geht immer eins weiter.

3. Da man jetzt die β hat kann man nun auch die α bestimmen. Die α bekommt man für $i > j$ durch

$$\alpha_{ij} = \frac{1}{\beta_{jj}}\left(a_{ij} - \sum_{k=0}^{j-1} \alpha_{ik}\beta_{kj}\right). \tag{3}$$

   Man fängt bei $i = j + 1$ an und geht immer eins weiter.

4. Mache in der nächster Spalte weiter.

Einige Anmerkungen zum Algorithmus sind noch angebracht. Die Gleichungen (3) und (2) erhält man indem man die entsprechenden Terme im Matritzenprodukt in Gleichung (1) nach der ensprechenden Größe auflöst. Das entscheidende ist, dass alle zur Berechnung benötigten α und β schon berechnet wurden. Man kann sich also so durch das Schema hangeln.

Der Crout-Algorithmus lässt sich besonders speichereffizient implementieren. Man muss sich dazu klar machen, dass a_{ij} nur für die Berechnung von β_{ij} (für $i \leqslant j$) bzw. α_{ij} (für $i > j$) benötigt wird. Man kann also das entsprechende Ergebnis an seine Stelle schreiben. Man bekommt so also als Endergebnis

$$\begin{pmatrix} \beta_{00} & \beta_{01} & \beta_{02} & \dots \\ \alpha_{10} & \beta_{11} & \beta_{12} & \dots \\ \alpha_{20} & \alpha_{21} & \beta_{22} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

c) (10 Zusatzpunkte) Implementieren Sie den Crout-Algorithmus und testen Sie ihn mit einer Zufallsmatrix. Multiplizieren Sie dazu L und U und vergewissern Sie sich, dass wieder A heraus kommt.

In Gleichung (3) teilt man durch β_{jj}. Hier kann es passieren auch bei einer nichtsingulären Matrix passieren dass dieser Eintrag 0 ist. Beim Gauss-Jordan Verfahren reagiert man darauf, indem man einfach zwei Zeilen im Gleichungssystem tauscht und so mit einem von Null verschiedenen Wert auf der Diagonale weiter macht. Hier ist dies etwas komplizierter.

Um zu sehen wie die Vertauschung hier ablaufen muss muss man die Gleichungen (2) für $i = j$ vergleichen mit Gleichung (3). Man sieht, dass Sie insgesamt bis auf den Unterschied, dass durch β_{jj} geteilt wurde, gleich aussehen. Multipliziert man also Gleichungen (3) mit β_{jj} und macht sich klar dass $\alpha_{jj} = 1$ ist sieht man, dass dann alle Gleichungen für $i \geqslant j$ bis auf die Zeilennummer gleich aussehen. Wenn man jetzt in der ursprünglichen Matrix A die Zeilen Tauscht entspricht dies also auch einem Zeilentausch in diesen Gleichungen. Auf diese Weise kann man verhindern, dass ein $\beta_{nn} = 0$ wird. Man tauscht also die Zeilen so dass der Eintrag mit dem größten $\beta_{jj}\alpha_{ij}$ auf der diagonalen steht. Diese Permutation muss man sich aber merken.

Diese Prozedur nennt man Pivoting. Nur mit diesem Pivoting ist der Algorithmus stabil und liefert sicher ein Ergebnis. Eine genauere Erklärung mit Beispiel finden Sie in den auch online unter www.nr.com erhältlichen “Numerical Recipes”. Andere bessere Erklärungen finden Sie auch in diversen Skripten im Internet. Wenn Sie Fragen haben können Sie auch im Büro des Assistenten vorbei kommen.

d) (10 Zusatzpunkte) Implementieren Sie den Crout-Algorithmus mit Pivoting

e) (3 Zusatzpunkte) Benutzen sie die LU-Zerlegung um die Determinante einer Matrix zu bestimmen.