Lösung durch Umformungen (1)

- Die Berechnung von Determinanten hat eine sehr hohe Komplexität und ist deshalb nur für kleine Systeme geeignet. Deshalb programmieren wir eine zweite Lösungsmöglichkeit mit Hilfe von Äquivalenzumformungen (Gauß-Jordan-Algorithmus).
- Zuerst führen wir x und y in einer Matrix zusammen:

```
k = np.arange(12, step=1.0).reshape(3, 4)
k[:3, :3] = x
k[:, 3] = y
```

- Die Matrix k hat folgende Form:

Durch Äquivalenzumformungen sollen die roten Einträge auf 0 und die orangen Einträge auf 1 gesetzt werden. Dann kann man die Lösung in der rechten Spalte ablesen.

Lösung durch Umformungen (2)

- Die ersten Umformungen lauten:

  k[0] = k[0] / k[0, 0]
  k[1] = k[1] - k[1, 0] * k[0]
  k[2] = k[2] - k[2, 0] * k[0]

- Weiter geht‘s mit:

  k[1] = k[1] / k[1, 1]
  k[0] = k[0] - k[0, 1] * k[1]
  k[2] = k[2] - k[2, 1] * k[1]

  und

  k[2] = k[2] / k[2, 2]
  k[0] = k[0] - k[0, 2] * k[2]
  k[1] = k[1] - k[1, 2] * k[2]

- Die Lösung ist dann: k[:, 3]

Danach sieht die Matrix k folgendermaßen aus:

$$\begin{bmatrix} 1 & \text{☹} & \text{☹} & \text{☺} \\ 0 & \text{😐} & \text{☹} & \text{☺} \\ 0 & \text{☹} & \text{😐} & \text{☺} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & \text{☹} & \text{☺} \\ 0 & 1 & \text{☹} & \text{☺} \\ 0 & 0 & \text{😐} & \text{☺} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & \text{☺} \\ 0 & 1 & 0 & \text{☺} \\ 0 & 0 & 1 & \text{☺} \end{bmatrix}$$

Lösung durch Umformungen (3)

Aufgaben

1. Schreiben Sie eine Funktion

   def gauss(x, y):

<u>Parameter</u>:	array x : Koeffizientenmatrix
	array y : Konstantenvektor
<u>Rückgabewert</u>:	array : Lösungsvektor

   die das Gleichungssystem mit Umformungen löst. Die Funktion soll wieder unabhängig von der Anzahl der Gleichungen sein. Außerdem soll wieder geprüft werden, ob die Anzahl der Zeilen von x und die Anzahl der Spalten von x und die Anzahl der Elemente von y gleich sind. Zusätzlich soll, falls ein Diagonalelement k[i, i] gleich 0 ist, np.array([None]) zurückgegeben werden.

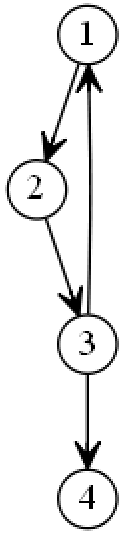
Lösung durch Umformungen (4)

Aufgaben

2. Gerichtete Graphen werden sehr häufig in der Informatik für verschiedene Fragestellungen verwendet. Ein solcher Graph besteht aus Knoten und gerichteten Kanten.

   Beispiel: Der nebenstehende Graph hat 4 Knoten, die mit den Zahlen 1 bis 4 beschriftet sind. Er hat ebenfalls 4 Kanten, die als Pfeile dargestellt sind.

   In einem Programm kann ein gerichteter Graph durch seine Adjazenzmatrix A gespeichert werden. In der i-ten Zeile und der j-ten Spalte wird eine 1 eingetragen, falls es eine Kante von Knoten i zu Knoten j gibt. Sonst steht überall 0. Beispiel:



$$\begin{array}{ccc}
 & \begin{matrix} & 1 \to 2 & & \end{matrix} & \\
\begin{matrix} \\ \\ 3 \to 1 \\ \\ \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} & \begin{matrix} \\ 2 \to 3 \\ 3 \to 4 \\ \\ \end{matrix}
\end{array}$$

Lösung durch Umformungen (5)

Aufgaben

Oft möchte man herausfinden, ob es einen Weg längs der gerichteten Kanten von einem Knoten i zu einen anderen Knoten j gibt. Das geht, indem man

$$W = A + A^2 + A^3 + \ldots + A^n$$

berechnet, wobei A^r die Matrix A r-mal mit sich selbst multipliziert ist, d.h. $A^r = A \cdot A \cdot \ldots \cdot A$ (r-mal). n ist dabei die Anzahl der Knoten. Steht in der i-ten Zeile und der j-ten Spalte von W eine 0, gibt es keinen Weg von i nach j, sonst gibt es einen.

Schreiben Sie ein Skript, das die Matrix W zum Graphen auf der letzten Folie berechnet. Zur Matrixmultiplikation gibt es den Operator @, z.B. B = C @ D.