Gauss-Elimination

October 26, 2018

1 Gauss-Elimination

Wir erzeugen zunächst ein zufälliges lineares Gleichungssystem $Ax = b$ und importieren die benötigten Bibliotheken. Der Einfachheit halber verzichten wir auf die Ausgabe von x und geben immer nur die Matrizen und Vektoren aus.

```
In [1]: import numpy as np
        import time
```

```
In [2]: N=3
        np.random.seed(1)
        A=np.random.uniform(-1,1,[N,N])
        # b=np.random.uniform(-1,1,[N,1])
        x=np.ones([N,1])
        b=np.matmul(A,x)
        A_orig=A
        b_orig=b
        x_orig=x
        print(A)
        print(b)
        print(x)
```

```
[[-0.16595599  0.44064899 -0.99977125]
 [-0.39533485 -0.70648822 -0.81532281]
 [-0.62747958 -0.30887855 -0.20646505]]
[[-0.72507825]
 [-1.91714588]
 [-1.14282317]]
[[1.]
 [1.]
 [1.]]
```

Wir implementieren direkt Spaltenpivotsuche. Wir suchen also in der 1. Spalte das Betragsmaximum und vertauschen dann in der Matrix und in der rechten Seite die entsprechende Zeile nach oben.

```
In [3]: i=0
        i_max=np.argmax(abs(A[i:N,i]))
        (A[i,:],A[i_max,:])=(A[i_max,:],A[i,:].copy())
        (b[i],b[i_max])=(b[i_max,:],b[i,:].copy())
        print(A)
        print(b)
```

```
[[-0.62747958 -0.30887855 -0.20646505]
 [-0.39533485 -0.70648822 -0.81532281]
 [-0.16595599  0.44064899 -0.99977125]]
[[-1.14282317]
 [-1.91714588]
 [-0.72507825]]
```

Damit steht jetzt das betragsmaximale Element der ersten Spalte links oben. Falls es Null ist, tue nichts, ansonsten eliminiere unterhalb der Hauptdiagonalen.

```
In [4]: if (abs(A[i,i])>0):
            for j in range(i+1,N):
                lji=-A[j,i]/A[i,i]
                A[j,:]=A[j,:]+lji*A[i,:]
            b[j]=b[j]+lji*b[i]
        print(A)
```

```
[[-6.27479577e-01 -3.08878546e-01 -2.06465052e-01]
 [-5.55111512e-17 -5.11883550e-01 -6.85242352e-01]
 [ 0.00000000e+00  5.22341279e-01 -9.45165310e-01]]
```

Ok, damit haben wir in der ersten Spalte unterhalb der Hauptdiagonalen eliminiert. Wir fassen das zusammen für alle Spalten und machen auch gleich noch das Rückwärtseinsetzen. Pivotisierung ist optional. Falls nicht pivotisiert wird, aber auf der Hauptdiagonalen eine Null erscheint, wird abgebrochen.

```
In [5]: def gauss_elim(A,b,pivot=1):
            N=A.shape[1]
            A=A.copy()
            b=b.copy()
            for i in range(0,N-1):
                if (pivot>0):
                    i_max=np.argmax(abs(A[i:N,i]))
                    (A[i,:],A[i_max,:])=(A[i_max,:],A[i,:].copy())
                    (b[i],b[i_max])=(b[i_max,:],b[i,:].copy())
                else:
                    if (A[i,i]==0):
                        raise ValueError('A very bad thing happened.')
                if (abs(A[i,i])>0):
                    for j in range(i+1,N):
```

```
                lji=-A[j,i]/A[i,i]
                A[j,:]=A[j,:]+lji*A[i,:]
                b[j]=b[j]+lji*b[i]
        x=np.zeros([N,1])
        for i in range(N-1,-1,-1):
            for j in range (i+1,N):
                b[i]=b[i]-A[i,j]*x[j]
            x[i]=b[i]/A[i,i]
        return x
```

Wir testen zunächst an unserem Beispiel.

```
In [6]: A=A_orig
        b=b_orig
        x=gauss_elim(A,b,0)
        print(np.linalg.norm(np.matmul(A,x)-b))
```

```
5.551115123125783e-16
```

Geht. Nun etwas größer, und wir messen die Zeit.

```
In [7]: N=1024
        A=np.random.uniform(-1,1,[N,N])
        x=np.random.uniform(-1,1,[N,1])
        b=np.matmul(A,x)
        t1=time.time()
        x=gauss_elim(A,b,0)
        print(time.time()-t1,' Sekunden')
        print('Fehler: ',np.linalg.norm(np.matmul(A,x)-b))
```

```
17.128010749816895  Sekunden
Fehler:  5.621837509468689e-09
```

Wir verdoppeln die Matrixgröße. Wir erwarten eine Komplexität von $n^3/3$, d.h. bei Verdopplung der Matrixgröße sollte sich die Laufzeit ungefähr verachtfachen.

```
In [8]: N=N*2
        A=np.random.uniform(-1,1,[N,N])
        x=np.random.uniform(-1,1,[N,1])
        # x=np.ones([N,1])
        b=np.matmul(A,x)
        t1=time.time()
        x=gauss_elim(A,b,0)
        print(time.time()-t1,' Sekunden')
        print('Fehler: ',np.linalg.norm(np.matmul(A,x)-b))
```

```
79.22900485992432  Sekunden
Fehler:  1.679814473098973e-08
```

Hier ist dies noch nicht der Fall, abhängig von der CPU bekommen wir hier eher einen Faktor 5. Bei größeren Matrizen sollte der Faktor näher an 8 liegen.

Abschließend schauen wir auf das Beispiel zur Spaltenpivotsuche aus der Vorlesung. Wir wählen die Matrix noch etwas extremer und führen Gauss ohne Spaltenpivotsuche durch.

```
In [9]: A=np.array([[1e-20,1],[1,1]])
        x=np.ones([2,1])
        print(x)
        b=np.matmul(A,x)
        x1=gauss_elim(A,b,0)
        print(x1)

[[1.]
 [1.]]
[[0.]
 [1.]]
```

Ok, gleiches Ergebnis wie in der Vorlesung: Ohne Spaltenpivotsuche ist der Fehler groß (100% in der ersten Komponente von x). Jetzt mit Spaltenpivotsuche:

```
In [10]: x2=gauss_elim(A,b,1)
         print(x2)

[[1.]
 [1.]]
```

Wie erwartet - der Fehler ist 0, wie in der Vorlesung.
Es gilt sicher: Ohne Spaltenpivotsuche ist die Gauss-Elimination nicht stabil.
Mit Spaltenpivotsuche ist die Gauss-Elimination stabil (nur an diesem Beispiel gezeigt).