NA02 Mnozenje matrica

Ivan Slapničar

11. listopada 2018.

1 Množenje matrica

Matrice možemo množiti na **tri različita načina**:

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}1&2&0\\4&3&2\\1&-1&1\end{bmatrix}=\begin{bmatrix}(1\cdot1+2\cdot4+3\cdot1) & 1\cdot2+2\cdot3+3\cdot(-1)) & (1\cdot0+2\cdot2+3\cdot1)\\(4\cdot1+5\cdot4+6\cdot1) & (4\cdot2+5\cdot3+6\cdot(-1)) & (4\cdot0+5\cdot2+6\cdot1)\\(7\cdot1+8\cdot5+9\cdot1) & (7\cdot2+8\cdot3+9\cdot(-1)) & (7\cdot0+8\cdot2+9\cdot1)\end{bmatrix}$$

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}1&2&0\\4&3&2\\1&-1&1\end{bmatrix}=\begin{bmatrix}1\\4\\7\end{bmatrix}\begin{bmatrix}1&2&0\end{bmatrix}+\begin{bmatrix}2\\5\\8\end{bmatrix}\begin{bmatrix}4&3&2\end{bmatrix}+\begin{bmatrix}3\\6\\9\end{bmatrix}\begin{bmatrix}1&-1&1\end{bmatrix}$$

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\begin{bmatrix}1&2&0\\4&3&2\\1&-1&1\end{bmatrix}=\begin{bmatrix}1\begin{bmatrix}1\\4\\7\end{bmatrix}+4\begin{bmatrix}2\\5\\8\end{bmatrix}+1\begin{bmatrix}3\\6\\9\end{bmatrix} & 2\begin{bmatrix}1\\4\\7\end{bmatrix}+3\begin{bmatrix}2\\5\\8\end{bmatrix}+(-1)\begin{bmatrix}3\\6\\9\end{bmatrix} & 0\begin{bmatrix}1\\4\\7\end{bmatrix}+2\begin{bmatrix}2\\5\\8\end{bmatrix}+1\begin{bmatrix}3\\6\\9\end{bmatrix}\end{bmatrix}$$

Formule se razlikuju u načinu prisutupa memoriji pa stoga i po brzini te u pogreškama zaokruživanja.

In [1]:
```
n=5
A=ones(5,5)
B=ones(5,5)
C=zeros(n,n)
for i=1:n
    for j=1:n
        for k=1:n
            C[i,j]=C[i,j]+A[i,k]*A[k,j]
        end
    end
end
C
```

Out[1]:
```
5×5 Array{Float64,2}:
 5.0  5.0  5.0  5.0  5.0
```

```
 5.0  5.0  5.0  5.0  5.0
 5.0  5.0  5.0  5.0  5.0
 5.0  5.0  5.0  5.0  5.0
 5.0  5.0  5.0  5.0  5.0
```

In [2]:
```julia
# Ili, krace: _dot
for i=1:n
    for j=1:n
        @show C[i,j]=(A[i,:]⋅B[:,j])[]
    end
end
```

```
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
C[i, j] = (A[i, :] ⋅ B[:, j])[] = 5.0
```

In [3]:
```julia
C=zeros(n,n)
for k=1:n
    for j=1:n
        for i=1:n
            C[i,j]=C[i,j]+A[i,k]*A[k,j]
            # @show i,j,C[i,j]
        end
    end
end
```

```
In [4]: # Ili, krace: _syrk
        C=zeros(n,n)
        for i=1:n
            @show C=C+A[:,i]*A[i,:]'
        end
```

```
C = C + A[:, i] * (A[i, :])' = [1.0 1.0 1.0 1.0 1.0; 1.0 1.0 1.0 1.0 1.0;
    1.0 1.0 1.0 1.0 1.0; 1.0 1.0 1.0 1.0 1.0; 1.0 1.0 1.0 1.0 1.0]
C = C + A[:, i] * (A[i, :])' = [2.0 2.0 2.0 2.0 2.0; 2.0 2.0 2.0 2.0 2.0;
    2.0 2.0 2.0 2.0 2.0; 2.0 2.0 2.0 2.0 2.0; 2.0 2.0 2.0 2.0 2.0]
C = C + A[:, i] * (A[i, :])' = [3.0 3.0 3.0 3.0 3.0; 3.0 3.0 3.0 3.0 3.0;
    3.0 3.0 3.0 3.0 3.0; 3.0 3.0 3.0 3.0 3.0; 3.0 3.0 3.0 3.0 3.0]
C = C + A[:, i] * (A[i, :])' = [4.0 4.0 4.0 4.0 4.0; 4.0 4.0 4.0 4.0 4.0;
    4.0 4.0 4.0 4.0 4.0; 4.0 4.0 4.0 4.0 4.0; 4.0 4.0 4.0 4.0 4.0]
C = C + A[:, i] * (A[i, :])' = [5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0;
    5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0]
```

```
In [5]: C=zeros(n,n)
        for j=1:n
            for k=1:n
                for i=1:n
                    C[i,j]=C[i,j]+A[i,k]*A[k,j]
                    # @show i,j,C[i,j]
                end
            end
        end
```

```
In [6]: # _axpy
        C=zeros(n,n)
        for i=1:n
            for k=1:n
                C[:,i]=C[:,i]+A[:,k]*B[k,i]
            end
            @show C
        end
```

```
C = [5.0 0.0 0.0 0.0 0.0; 5.0 0.0 0.0 0.0 0.0; 5.0 0.0 0.0 0.0 0.0;
     5.0 0.0 0.0 0.0 0.0; 5.0 0.0 0.0 0.0 0.0]
C = [5.0 5.0 0.0 0.0 0.0; 5.0 5.0 0.0 0.0 0.0; 5.0 5.0 0.0 0.0 0.0;
     5.0 5.0 0.0 0.0 0.0; 5.0 5.0 0.0 0.0 0.0]
C = [5.0 5.0 5.0 0.0 0.0; 5.0 5.0 5.0 0.0 0.0; 5.0 5.0 5.0 0.0 0.0;
     5.0 5.0 5.0 0.0 0.0; 5.0 5.0 5.0 0.0 0.0]
C = [5.0 5.0 5.0 5.0 0.0; 5.0 5.0 5.0 5.0 0.0; 5.0 5.0 5.0 5.0 0.0;
     5.0 5.0 5.0 5.0 0.0; 5.0 5.0 5.0 5.0 0.0]
C = [5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0;
     5.0 5.0 5.0 5.0 5.0; 5.0 5.0 5.0 5.0 5.0]
```

1.1 Basic Linear Algrbea Subroutines - BLAS

Na primjer, `ddot.f` - uočite *loop unrolling*:

```
DOUBLE PRECISION FUNCTION DDOT(N,DX,INCX,DY,INCY)
*     .. Scalar Arguments ..
      INTEGER INCX,INCY,N
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION DX(*),DY(*)
*     ..
*
*  Purpose
*  =======
*
*     forms the dot product of two vectors.
*     uses unrolled loops for increments equal to one.
*     jack dongarra, linpack, 3/11/78.
*     modified 12/3/93, array(1) declarations changed to array(*)
*
*
*     .. Local Scalars ..
      DOUBLE PRECISION DTEMP
      INTEGER I,IX,IY,M,MP1
*     ..
*     .. Intrinsic Functions ..
      INTRINSIC MOD
*     ..
      DDOT = 0.0d0
      DTEMP = 0.0d0
      IF (N.LE.0) RETURN
      IF (INCX.EQ.1 .AND. INCY.EQ.1) GO TO 20
*
*        code for unequal increments or equal increments
*          not equal to 1
*
      IX = 1
      IY = 1
      IF (INCX.LT.0) IX = (-N+1)*INCX + 1
      IF (INCY.LT.0) IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
          DTEMP = DTEMP + DX(IX)*DY(IY)
          IX = IX + INCX
          IY = IY + INCY
   10 CONTINUE
      DDOT = DTEMP
      RETURN
*
```

```
*        code for both increments equal to 1
*
*
*        clean-up loop
*
   20 M = MOD(N,5)
      IF (M.EQ.0) GO TO 40
      DO 30 I = 1,M
          DTEMP = DTEMP + DX(I)*DY(I)
   30 CONTINUE
      IF (N.LT.5) GO TO 60
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
          DTEMP = DTEMP + DX(I)*DY(I) + DX(I+1)*DY(I+1) +
     +            DX(I+2)*DY(I+2) + DX(I+3)*DY(I+3) + DX(I+4)*DY(I+4)
   50 CONTINUE
   60 DDOT = DTEMP
      RETURN
      END
```

1.2 Brzina računanja

In [7]:
```
function AB{T}(A::Array{T},B::Array{T})
    n=maximum(size(A))
    C=zeros(n,n)
    # C=[zeros(A[1,1]) for i=1:n, j=1:n]
    for i=1:n
        for j=1:n
            for k=1:n
                C[i,j]=C[i,j]+(A[i,k]*B[k,j])
            end
        end
    end
    C
end
```

Out[7]:
```
AB (generic function with 1 method)
```

In [8]:
```
srand(123)
n=512
A=rand(n,n)
B=rand(n,n)
```

Out[8]:
```
512×512 Array{Float64,2}:
 0.403342   0.194433   0.368166  …  0.897631   0.635233   0.54833
 0.582401   0.540058   0.968354     0.928777   0.585136   0.897064
 0.0895888  0.019672   0.460148     0.84592    0.540231   0.49852
```

```
 0.109843   0.350876    0.895157      0.805366   0.569382   0.225873
 0.77105    0.12302     0.336079      0.506111   0.267254   0.20065
 0.705956   0.809889    0.705933   …  0.308255   0.0268284  0.869825
 0.28384    0.51511     0.22677       0.477558   0.948534   0.45572
 0.527531   0.466912    0.183345      0.451524   0.91217    0.726608
 0.207026   0.760507    0.100759      0.347039   0.532559   0.43716
 0.655316   0.00615315  0.0437554     0.84412    0.325017   0.436636
 0.858271   0.495363    0.628733   …  0.576524   0.770021   0.773433
 0.380046   0.953783    0.727147      0.955753   0.359923   0.98021
 0.261233   0.332958    0.64573       0.778991   0.915038   0.504806
 ⋮                                 ⋱             ⋮
 0.297016   0.625424    0.682813   …  0.898927   0.825086   0.337797
 0.736339   0.863817    0.51978       0.025043   0.146582   0.0397381
 0.279265   0.645635    0.444262      0.638976   0.887484   0.124973
 0.736236   0.262118    0.036378      0.489839   0.766442   0.718518
 0.546335   0.307645    0.900528      0.889106   0.506884   0.576916
 0.252813   0.0126003   0.491304   …  0.431563   0.232152   0.791677
 0.223265   0.206073    0.77377       0.781308   0.878107   0.866325
 0.707541   0.516783    0.52801       0.0034571  0.769619   0.572177
 0.0741184  0.679187    0.127014      0.35834    0.444379   0.939392
 0.657141   0.816249    0.317785      0.438778   0.642568   0.570322
 0.996618   0.707525    0.491185   …  0.103464   0.911435   0.581906
 0.26521    0.434631    0.936293      0.731415   0.645221   0.509805
```

In [11]:
```
# Izvedite 2 puta, releventno je drugo mjerenje.
@time C=A*B
```

```
  0.024089 seconds (6 allocations: 2.000 MiB)
```

Out[11]:
```
512×512 Array{Float64,2}:
 123.337  124.249  131.322  127.233  …  133.223  135.108  124.309  132.097
 133.376  134.8    137.887  133.887     138.991  141.678  132.803  140.949
 126.961  129.731  133.623  133.527     133.546  137.939  135.245  135.957
 126.108  123.805  130.514  134.326     132.437  133.251  128.107  126.73
 130.729  129.568  136.093  134.189     138.33   136.799  135.126  135.724
 126.205  124.428  132.415  130.515  …  132.152  135.271  129.957  132.848
 120.917  124.84   129.601  125.556     129.19   127.631  125.792  123.917
 125.053  127.381  130.673  129.123     132.479  131.542  124.371  125.556
 133.749  133.756  139.0    138.217     134.56   138.714  138.2    136.309
 127.146  132.717  134.176  127.841     135.178  129.61   129.31   130.388
 126.794  126.378  126.967  130.196  …  131.32   131.26   128.144  130.846
 123.789  127.18   128.614  129.805     131.113  129.648  126.747  125.307
 124.961  125.775  134.394  130.152     134.337  131.297  131.058  129.366
   ⋮                                 ⋱                      ⋮
 131.484  131.195  134.791  135.718  …  136.421  136.722  134.437  134.316
 131.213  128.729  136.668  129.596     133.15   138.446  131.258  135.624
```

```
113.597  116.385  120.277  121.159      123.104  118.596  123.459  121.393
131.939  130.103  132.737  137.077      133.701  138.763  134.861  133.103
127.102  132.749  136.247  132.839      136.606  136.292  133.537  130.643
123.59   121.411  126.328  123.083  ...  126.67   126.208  122.048  125.833
130.109  131.142  137.36   134.332      134.815  135.174  133.47   134.408
125.029  123.201  131.51   130.954      134.798  133.658  130.067  132.203
125.66   127.162  132.862  133.201      132.145  132.777  131.97   130.675
128.1    129.915  133.9    134.478      135.938  136.686  129.873  133.115
127.718  133.966  135.553  135.696  ...  134.991  140.197  133.096  136.787
130.27   126.665  135.02   133.442      135.579  136.147  130.095  134.022
```

In [12]:
```
operacija_u_sekundi=(2*n^3)/0.024
```

Out[12]:
```
1.1184810666666666e10
```

In [14]:
```
# Izvedite 2 puta, releventno je drugo mjerenje. Izvodjenje našeg programa
# je značajno sporije!!
@time AB(A,B);
```

```
  2.911200 seconds (10 allocations: 2.000 MiB)
```

1.2.1 Blok varijanta

Da bi ubrzali naš program, potrebno je računati s blok-matricama.

In [15]:
```
function AB{T}(A::Array{T},B::Array{T})
    n=maximum(size(A))
    # C=zeros(n,n)
    C=[zeros(A[1,1]) for i=1:n, j=1:n]
    for i=1:n
        for j=1:n
            for k=1:n
                C[i,j]=C[i,j]+(A[i,k]*B[k,j])[]
            end
        end
    end
    C
end
```

Out[15]:
```
AB (generic function with 1 method)
```

In [16]:
```
# Probajte k,l=32,16 i k,l=64,8. Izvedite dva puta.
k,l=64,8
Ab=[rand(k,k) for i=1:l, j=1:l]
Bb=[rand(k,k) for i=1:l, j=1:l];
```

```
In [18]: # Ovo je bitno brze
         @time C=AB(Ab,Bb);
```

```
0.068175 seconds (2.18 k allocations: 34.084 MiB, 13.10% gc time)
```

1.3 Točnost računanja

1.3.1 Osnovne računske operacije

Za operacije $\odot \in \{+,*,/\}$ vrijedi

$$fl(a\odot b)=(1+\varepsilon_\odot)(a\odot b), \qquad |\varepsilon_\odot|\leq \varepsilon.$$

1.3.2 Zbrajanje

Ako potpuno točno zbrojimo dva broja koji imaju (male) pogreške iz prethodnih računanja, jednakost

$$a(1+\varepsilon_a)+b(1+\varepsilon_b)=(1+\varepsilon_{ab})(a+b)$$

daje

$$\varepsilon_{ab}=\frac{a\varepsilon_a+b\varepsilon_a}{a+b},$$

pa je

$$|\varepsilon_{ab}|\leq \varepsilon\,\frac{|a|+|b|}{|a+b|}.$$

Ako su a i b veliki brojevi različitog predznaka i ako je $a-b$ malo, pogreška može biti velika (*katastrofalno kraćenje*).

1.3.3 Skalarni produkt

Za vektore x i y iz $\mathbb{R}^n$, rekurzivna primjena prethodne formule daje apsloutna pogrešku

$$|fl(x\cdot y)-x\cdot y|\leq O(n\varepsilon)|x|\cdot|y| \tag{1}$$

i relativnu pogrešku

$$\frac{|fl(x\cdot y)-x\cdot y|}{|x\cdot y|}\leq O(n\varepsilon)\frac{|x|\cdot|y|}{|x\cdot y|}$$

Ukoliko su vektori x i y skoro okomiti, relativna pogreška može biti velika.

1.3.4 Množenje matrica

Iz formule (1) za matrice A i B reda n slijedi

$$|fl(A\cdot B)-A\cdot B|\leq O(n\varepsilon)|A|\cdot|B|.$$

```
In [19]: n=1_000_000
         x=rand(n)
         y=rand(n)-0.5;
```

```
In [20]: d=x⋅y
```

```
Out[20]: -166.36263639921918
```

```
In [21]: ab=abs.(x)⋅abs.(y)
```

```
Out[21]: 125090.9108607984
```

```
In [22]: # provjerimo rješenje pomocu `BigFloat` brojeva (70tak tocnih znamenki)
         xbig=map(BigFloat,x)
         ybig=map(BigFloat,y)
         ybig[1]
```

```
Out[22]: 1.229958355596845187562848877860233187675476074218750000000000000
      00000000000000000e-01
```

```
In [23]: dbig=xbig⋅ybig
```

```
Out[23]: -1.6636263639922141026363744061680900247921371146060882518113416
      38137370368966117e+02
```

```
In [24]: abserr=map(Float64,abs(d-dbig))
```

```
Out[24]: 2.2261501747522156e-12
```

```
In [25]: relerr=map(Float64,abserr/d)
```

```
Out[25]: -1.3381310989867578e-14
```

```
In [26]: n=256
         A=rand(n,n)
         B=rand(n,n).-0.5;
```

```
In [27]: Ab=map(BigFloat,A)
         Bb=map(BigFloat,B);
```

```
In [28]: C=A*B
         # ovo traje malo duze
         Cb=Ab*Bb
         abserr=abs.(C-map(Float64,Cb))
```

```
Out[28]: 256×256 Array{Float64,2}:
 6.66134e-16  0.0          0.0          …  1.77636e-15  1.77636e-15
 1.77636e-15  2.66454e-15  1.55431e-15     1.66533e-16  8.88178e-16
 1.11022e-16  3.55271e-15  1.77636e-15     8.88178e-16  1.77636e-15
 8.88178e-16  2.66454e-15  8.88178e-16     1.77636e-15  4.44089e-15
 6.66134e-16  3.55271e-15  8.88178e-16     8.88178e-16  1.11022e-15
 1.11022e-15  1.33227e-15  4.44089e-16  …  9.4369e-16   1.11022e-15
 2.22045e-15  4.44089e-15  4.44089e-16     2.22045e-16  2.32453e-16
 2.22045e-16  0.0          7.77156e-16     8.88178e-16  0.0
 8.88178e-16  1.77636e-15  1.77636e-15     0.0          8.88178e-16
 1.33227e-15  4.44089e-16  2.22045e-15     4.44089e-16  5.55112e-16
 8.88178e-16  5.32907e-15  2.66454e-15  …  6.66134e-16  8.88178e-16
 4.44089e-16  8.88178e-16  8.88178e-16     4.44089e-16  2.88658e-15
 1.55431e-15  3.55271e-15  4.44089e-16     8.88178e-16  1.77636e-15
 ⋮                                      ⋱               ⋮
 0.0          8.88178e-16  2.22045e-16     2.22045e-16  1.33227e-15
 4.44089e-16  1.77636e-15  8.88178e-16  …  4.44089e-16  1.22125e-15
 4.44089e-16  1.77636e-15  3.55271e-15     0.0          3.55271e-15
 8.88178e-16  2.66454e-15  1.77636e-15     4.44089e-16  2.22045e-15
 8.88178e-16  4.44089e-16  4.44089e-16     6.66134e-16  1.97065e-15
 2.22045e-16  3.55271e-15  2.22045e-15     1.11022e-16  1.77636e-15
 0.0          1.77636e-15  8.88178e-16  …  8.88178e-16  3.10862e-15
 1.77636e-15  2.66454e-15  0.0             2.22045e-16  4.44089e-16
 2.66454e-15  2.66454e-15  5.55112e-16     2.27596e-15  1.249e-15
 8.88178e-16  3.55271e-15  4.44089e-16     1.11022e-15  1.33227e-15
 2.66454e-15  2.66454e-15  1.77636e-15     1.11022e-16  0.0
 8.88178e-16  1.33227e-15  3.33067e-16  …  0.0          5.13478e-16
```

```
In [29]: abs.(A)*abs.(B)*n*eps()
```

```
Out[29]: 256×256 Array{Float64,2}:
 1.83639e-12  1.83419e-12  1.78526e-12  …  1.77334e-12  1.67548e-12
 1.84036e-12  1.89081e-12  1.70531e-12     1.68096e-12  1.65554e-12
 1.94663e-12  2.01423e-12  1.90719e-12     1.87747e-12  1.8151e-12
 1.9607e-12   1.98346e-12  1.89684e-12     1.82503e-12  1.81072e-12
 1.70046e-12  1.74589e-12  1.6326e-12      1.71198e-12  1.56358e-12
 1.96286e-12  1.94125e-12  1.87781e-12  …  1.84714e-12  1.69475e-12
 1.8108e-12   1.86127e-12  1.7464e-12      1.76974e-12  1.64591e-12
 1.94416e-12  2.00862e-12  1.92649e-12     1.94244e-12  1.89081e-12
 1.95015e-12  1.9416e-12   1.87136e-12     1.87483e-12  1.75342e-12
 1.77933e-12  1.70724e-12  1.67142e-12     1.75837e-12  1.56263e-12
 1.80551e-12  1.87718e-12  1.76786e-12  …  1.74094e-12  1.66777e-12
```

1.81314e-12	1.77318e-12	1.67584e-12		1.6883e-12	1.5647e-12
1.78267e-12	1.85467e-12	1.67084e-12		1.73509e-12	1.72991e-12
$\vdots$			$\ddots$		$\vdots$
1.8914e-12	1.85778e-12	1.71498e-12		1.79509e-12	1.76e-12
1.84988e-12	1.88118e-12	1.75533e-12	...	1.77237e-12	1.75369e-12
1.87798e-12	1.92275e-12	1.87157e-12		1.88175e-12	1.80263e-12
1.90133e-12	1.90932e-12	1.84729e-12		1.82367e-12	1.70088e-12
2.01442e-12	1.98482e-12	1.88667e-12		1.92884e-12	1.81926e-12
1.80943e-12	1.81526e-12	1.74822e-12		1.76786e-12	1.62152e-12
1.90969e-12	1.84804e-12	1.72465e-12	...	1.80407e-12	1.71811e-12
1.83971e-12	1.81031e-12	1.6655e-12		1.77403e-12	1.67088e-12
1.76758e-12	1.80629e-12	1.70321e-12		1.79813e-12	1.74381e-12
1.96305e-12	1.93885e-12	1.83691e-12		1.9976e-12	1.83174e-12
1.87235e-12	1.87717e-12	1.80247e-12		1.84804e-12	1.75175e-12
1.82492e-12	1.88004e-12	1.77832e-12	...	1.80628e-12	1.65533e-12

Ovdje nije lako razabrati relativnu pogrešku pa nam treba pogodnija mjera. U sljedećeoj bilježnici objasnit ćemo *matrične i vektorske norme*.