Singular value decomposition and its implementations

Wen Zhang
Anastasios Arvanitis
Asif Al-Rasheed

Outlines

- Implementation of indirect method
  - Matlab code
  - Testing results
- Implementation of direct method
  - Matlab code
  - Testing results
  - Combined method
- Comparisons of different method

Implementation of Indirect method

```
function [u,d,v]=SVDecom(A)
[m,n]=size(A); sinflag=0;
if (m>n)
   [u,d]=eig(A*A');  u=GSO(u); u=fliplr(u);
   d1(1:n,n+1:m)=zeros(n,m-n);
   dd=fliplr(diag(d.^(.5))')';
   d1(1:n,1:n)=diag(dd(1:n)); d=d1;
   for i=1:n
      if (d(i,i)~=0)
         v(:,i)=A'*u(:,i)/d(i,i);
      else
         sinflag=1;   v(:,i)=ones(n,1);
      end
   end
   v=GSO(v);
   if (sinflag==1)
       v=GSO(v);
   end
   d=d';
end
```

AA^T approach

```
if (m<n||m==n)
   [v,d]=eig(A'*A);
    v=GSO(v); v=fliplr(v);
   d1=zeros(m,n);
   dd=fliplr(diag(d.^(.5))');
   d1(1:m,1:m)=diag(dd(1:m));
   d=d1;
   for i=1:m
      if(d(i,i)~=0)
         u(:,i)=A*v(:,i)/d(i,i);
      else
         sinflag=1;
         u(:,i)=ones(m,1);
      end
   end
   u=GSO(u);
   if (sinflag==1)
      u=GSO(u);
   end
end
```

$A^T A$ approach

The process of GSO

```
function  Q=GSO(A)
[n,m]=size(A); Q1(:,1)=A(:,1);
for j=2:m
   Q1(:,j)=A(:,j);
   for i=1:j-1
      x=Q1(:,i); a=A(:,j); qnorm=x'*x;
      if (qnorm~=0)
        Q1(:,j)=Q1(:,j)-(a'*x)/qnorm*x;
      end
   end
end
Q(:,1)=Q1(:,1);
```

Double Orthogonalize

```
for j=2:m
   Q(:,j)=Q1(:,j);
   for i=1:j-1
      x=Q(:,i); a=Q1(:,j); qnorm=x'*x;
      if(qnorm~=0)
         Q(:,j)=Q(:,j)-(a'*x)/qnorm*x;
      end
   end
end
```

Normalize

```
for i=1:m
   qnorm=Q(:,i);x=(qnorm'*qnorm);
   if(x~=0)
      Q(:,i)=Q(:,i)/(x^(0.5));
   end
end
```

Some test results of Indirect method

```
>> A=rand(150,40);
>> [u,d,v]=SVDecom (A);
>> norm(u*d*v'-A,1)
ans =  4.3643e-013
>> norm(u*d*v'-A,inf)
ans =  4.6653e-013
>> norm(u*u'-eye(150),1)
ans =  6.6027e-015
>> norm(u'*u-eye(150),1)
ans =  5.7560e-015
```

```
>> norm(v'*v-eye(40),1)
ans =  1.8991e-015

>> norm(v*v'-eye(40),1)
ans =  2.4568e-015

>> [u1,d1,v1]=svd(A);

>> norm(d1-d,inf)

ans =  1.1546e-014
```

Implementation of Direct method

```
function [u,b,v]=BiDiag(A)
[m,n]=size(A); n1=min(m,n);
u1=Householder(A(:,1));b=u1*A; A1=b';
a=A1(2:n,1); v2=Householder(a);
v1(2:n,2:n)=v2';v1(1,1)=1; b=b*v1;
u=u1;v=v1;
for i=2:n1-2
    a=b(i:m,i);     clear u1;
   u1(1:i-1,1:i-1)=eye(i-1);
    u1(i:m,i:m)=Householder(a);
    b=u1*b; A1=b'; a1=A1(i+1:n,i);
    v2=Householder(a1);
    clear v1; v1(1:i,1:i)=eye(i);
    v1(i+1:n,i+1:n)=v2';    b=b*v1; u=u*u1;
    v=v*v1;
end
a=b(n1-1:m,n1-1);clear u1;
u1(1:n1-2,1:n1-2)=eye(n1-2);
u1(n1-1:m,n1-1:m)=Householder(a);
b=u1*b; u=u*u1;
```

First, transform to bidiagonal matrix:

$$A = UBV^T$$

Get a sequence of u1 and v1 by Householder & identity matrices

$$
\begin{bmatrix}
b_1 & b_2 & & & & O \\
 & b_3 & b_4 & & & \\
 & & & \ddots & & \\
 & & & & b_{2n-3} & b_{2n-2} \\
 & O & & & & b_{2n-1} \\
 & & & O & & \\
 & & & & &
\end{bmatrix}
=
\begin{bmatrix}
\hat{B} \\
O_{(m-n)\times n}
\end{bmatrix}
$$

Using Shuffle matrix P:

$$
B_1 = P^T \begin{bmatrix} O & \hat{B}^T \\ \hat{B} & O \end{bmatrix} P
$$

$$
B_1 =
\begin{bmatrix}
0 & b_1 & 0 & & \cdots & O & \\
b_1 & 0 & b_2 & & & & \\
0 & b_2 & 0 & & & & \\
\vdots & & \ddots & \ddots & \ddots & & \\
 & & & & & 0 & b_{2n-1} \\
 & O & & & & b_{2n-1} & 0
\end{bmatrix}
$$

About the Shuffle matrix

The Shuffle matrix P is defined by:

$$P = [e_1 \quad e_{n+1} \quad e_2 \quad e_{n+2} \quad \cdots \quad e_n \quad e_{2n}]$$

```
function p=GenerateShuff(m)
for i=1:2:2*m-1
   p(:,i)=geneVector(2*m,(i+1)/2);
end
for i=2:2:2*m
   p(:,i)=geneVector(2*m,i/2+m);
end

function p=geneVector(m,n)
p(m)=0;p=p';
p(n)=1;
```

```
>> GenerateShuff(3)
ans =

   1   0   0   0   0   0
   0   0   1   0   0   0
   0   0   0   0   1   0
   0   1   0   0   0   0
   0   0   0   1   0   0
   0   0   0   0   0   1
```

Get the SVD of bidiagonal matrix

```
function [u,d,v]=SVDUpBidiag(B)
[m,n]=size(B);
B1=B(1:n,1:n);
C(1:n,n+1:n+n)=B1';
C(n+1:2*n,1:n)=B1;
p=GenerateShuff(n);
c1=p'*C*p;
[x,d]=eig(c1);
x1=fliplr(x);
d1=fliplr(flipud(d));
d=d1(1:n,1:n);
x1=x1(:,1:n);
x1=x1*2^(.5);
for i=1:n
   v(i,:)=x1(2*i-1,:);
   u(i,:)=x1(2*i,:);
end
d(n+1:m,1:n)=zeros(m-n,n);
u(n+1:m,n+1:m)=eye(m-n);
```

Get eigen pairs of

$$\begin{bmatrix} 0 & b_1 & 0 & & \cdots & \\ b_1 & 0 & b_2 & & & O \\ 0 & b_2 & 0 & & & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & & 0 & b_{2n-1} \\ & O & & & b_{2n-1} & 0 \end{bmatrix}$$

Via the relation:

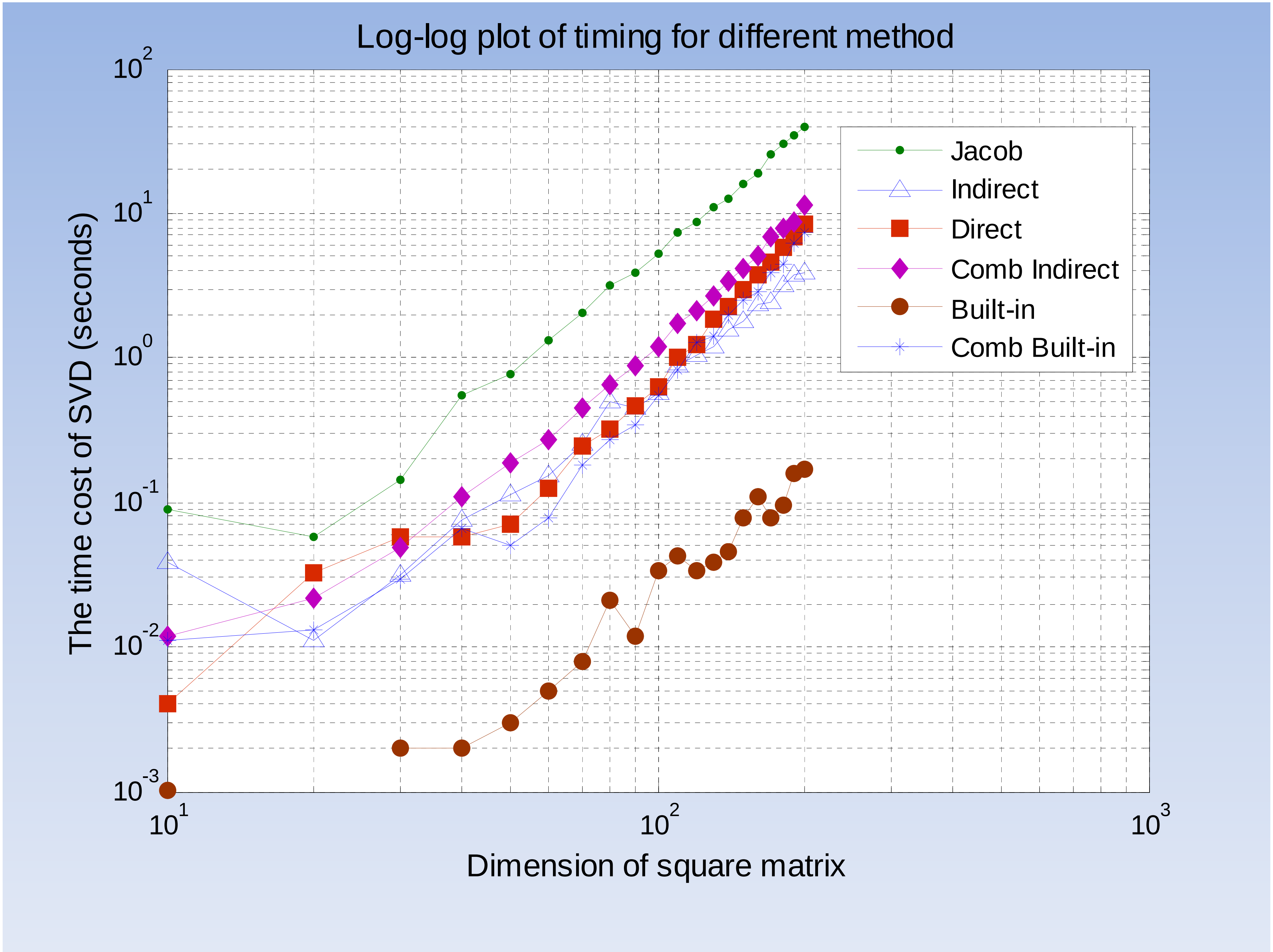
$$h_i^{\pm} = \frac{1}{\sqrt{2}}(v_{i1}, \pm u_{i1}, v_{i2}, \pm u_{i2}, \ldots, v_{in}, \pm u_{in})^T$$

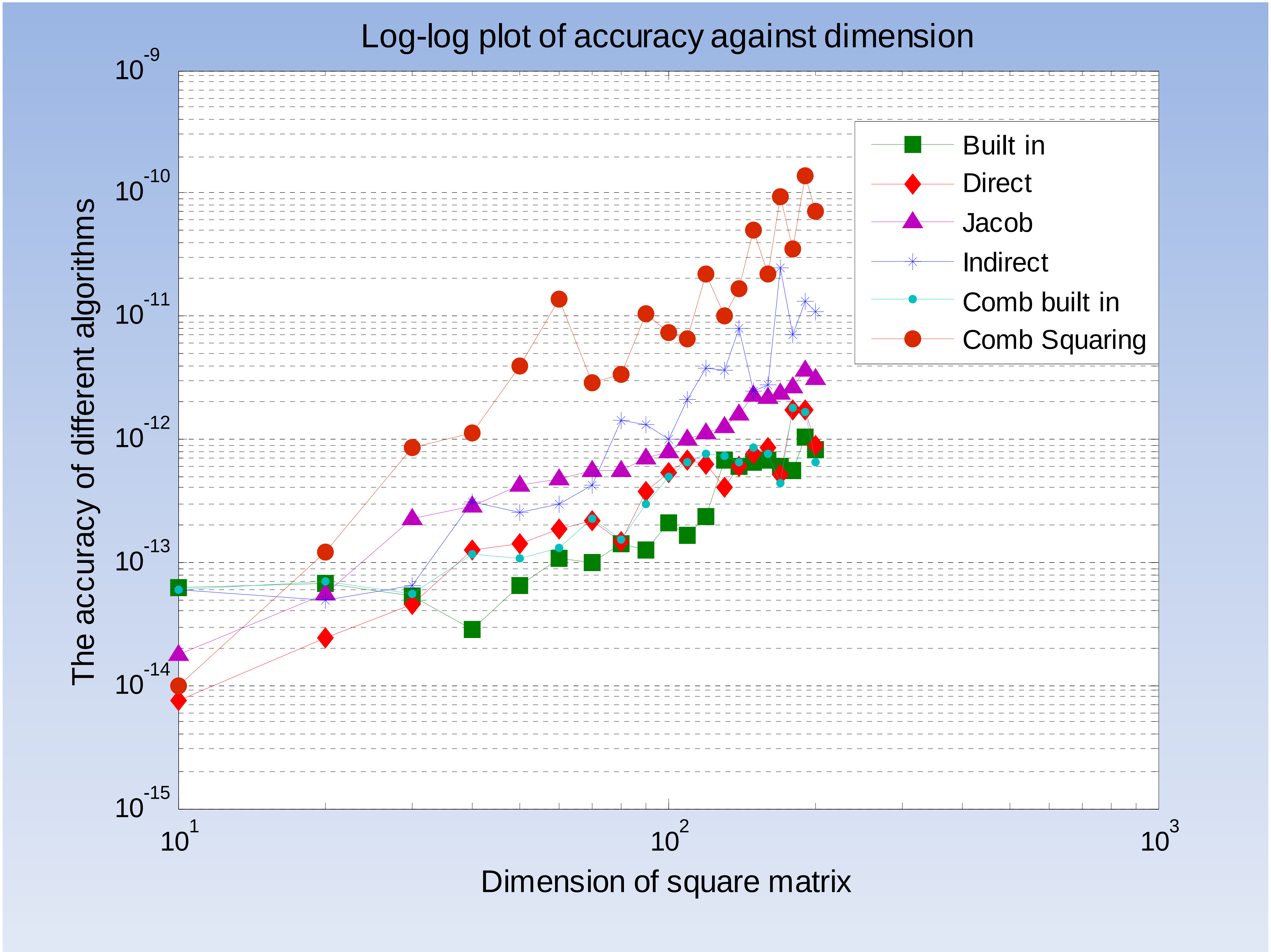
to get SVD of $\widehat{B}$

Combined method

- Transform the original matrix to the bidiagonal matrix, then to get SVD of bidiagonal:

  - Use built-in function
  - Use indirect method
  - Use Francis algorithm (cont')


Log-log plot of timing for different method
The time cost of SVD (seconds)
Dimension of square matrix
Jacob
Indirect
Direct
Comb Indirect
Built-in
Comb Built-in
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{1}
10^{2}
10^{3}



Log-log plot of accuracy against dimension
The accuracy of different algorithms
Dimension of square matrix
10^-9
10^-10
10^-11
10^-12
10^-13
10^-14
10^-15
10^1
10^2
10^3
Built in
Direct
Jacob
Indirect
Comb built in
Comb Squaring

Thank you!