Improvement to Hessenberg Reduction

Shankar, Yang, Hao

What is Hessenberg Matrix

A special square matrix has zero entries below the first subdiagonal or above the first superdiagonal.

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 3 & 4 & 1 & 7 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 0 & 0 \\ 5 & 2 & 3 & 0 \\ 3 & 4 & 3 & 7 \\ 5 & 6 & 1 & 1 \end{bmatrix}$$

Why Hessenberg Matrix

Less computational efforts required for triangular matrix

Not convenient to convert to triangular directly then Hessenberg is a good transient form

Real life application

Earthquake modeling
Weather forecasting
Real world matrices are huge.
Dense solves are inefficient

Ways to convert to Hessenberg matrix

Householder Transform

Givens Rotation

Variants of the above (block forms etc)

Improvement of Hessenberg Reduction

Method 1: Householder Reflection

Setp1: Form Householder Matrix P_1 P_2

Setp2: Householder Reflection $A=P_1AP_1^{-1}$ Note: P_1 is orthogonal matrix, $P_1=P_1^{-1}$

A: 4 by 4 random matrix

A

0.2815	0.1386	0.5038	0.4494
0.7311	0.5882	0.4896	0.9635
0.1378	0.3662	0.8770	0.0423
0.8367	0.8068	0.3531	0.9730

Original Matrix

$A= P_2P_1AP_1P_2$

0.2815	-0.4884	-0.4152	0.2532
1.1196	1.7764	0.4547	-0.1854
0.0000	0.1629	0.8220	0.1283
0.0000	-0.0000	-0.0017	-0.1603

Final Matrix

Target :Zero these entries

P1

1.0000	0	0	0
0	-0.6530	-0.1230	-0.7473
0	-0.1230	0.9908	-0.0556
0	-0.7473	-0.0556	0.6621

P2

1.0000	0	0	0
0	1.0000	0	0
0	0	-0.9802	0.1980
0	0	0.1980	0.9802

Improvement of Hessenberg Reduction

How to form Householder Matrix

Initial Operation

$$\alpha = -\text{sgn}(a_{21})\sqrt{\sum_{j=2}^{n} a_{j1}^2};$$

$$r = \sqrt{\frac{1}{2}(\alpha^2 - a_{21}\alpha)};$$

From α and r, construct vector v:

$$v^{(1)} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix},$$

where $v_1 = 0;\ v_2 = \frac{a_{21} - \alpha}{2r}$, and

$v_k = \frac{a_{k1}}{2r}$ for each k=3,4 ..n

Then compute:

$$P^1 = I - 2v^{(1)}(v^{(1)})^t$$

$$A^{(1)} = P^1 A P^1$$

Loop it

$$\alpha = -\text{sgn}(a_{k+1,k})\sqrt{\sum_{j=k+1}^{n} a_{jk}^2};$$

$$r = \sqrt{\frac{1}{2}(\alpha^2 - a_{k+1,k}\alpha)};$$

$$v_1^k = v_2^k = .. = v_k^k = 0;$$

$$v_{k+1}^k = \frac{a_{k+1,k}^k - \alpha}{2r}$$

$v_j^k = \frac{a_{jk}^k}{2r}$ for j=k+2; k+3,..., n

$$P^k = I - 2v^{(k)}(v^{(k)})^t$$

$$A^{(k+1)} = P^k A^{(k)} P^k$$

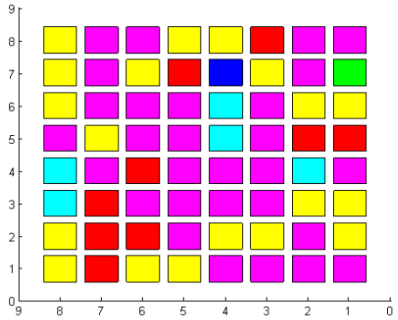
Code

```
clc
clear
n=8;
A=rand(n,n);

for k=1:n-2
    [P,A]=Tridiag(A,k)
end

function [P,A]=Tridiag(A,k)
% function of tridiagnolization by householder method
% A: comeback tridiagnoal matrix
n=length(A);
aerfa=-1*sign(A(k+1,k))*sqrt(sum(A((k+1:n),k).^2));
r=sqrt(0.5*(aerfa^2-A(k+1,k)*aerfa));
v(1:k)=0;
v(k+1)=(A(k+1,k)-aerfa)/2/r;
for i=k+2:n
    v(i)=A(i,k)/2/r;
end
v=v';
P=eye(n,n)-2*v*v';
A=P*A*P;
end
```

Improvement of Hessenberg Reduction

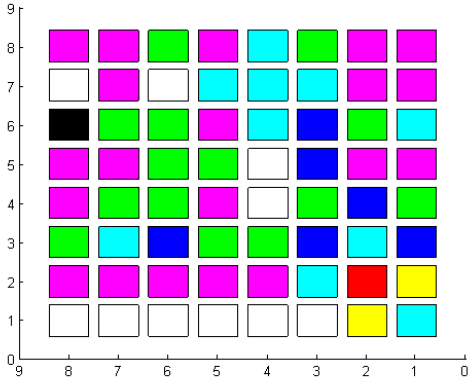
Example

A 8x8 Random Square Matrix =

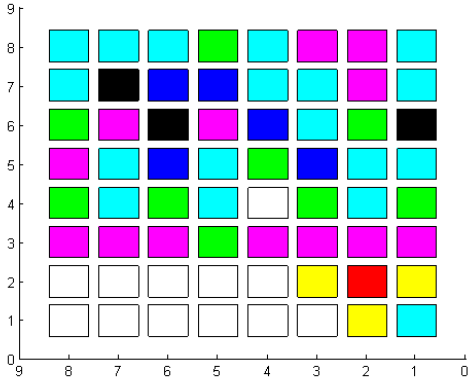


Householder Transform

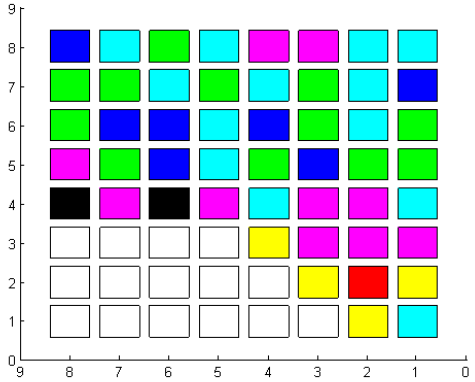
Loop1: $A_1=P_1AP_1$



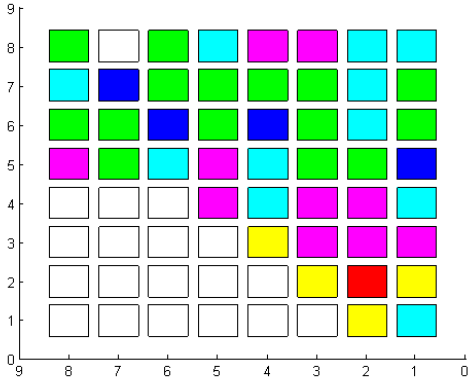
Loop2: $A_2=P_2A_1P_2$



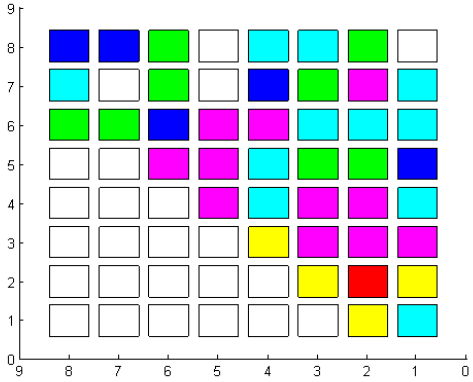
Loop3: $A_3=P_3A_2P_3$



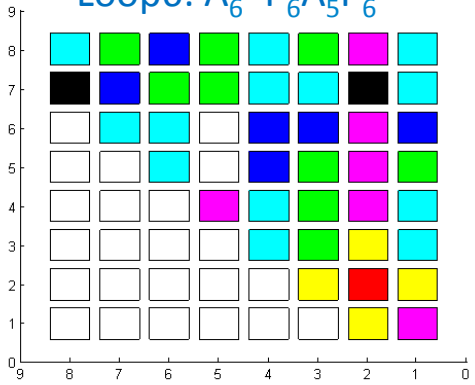
Loop4: $A_4=P_4A_3P_4$



Loop5: $A_5=P_5A_4P_5$



Loop6: $A_6=P_6A_5P_6$



Improvement of Hessenberg Reduction

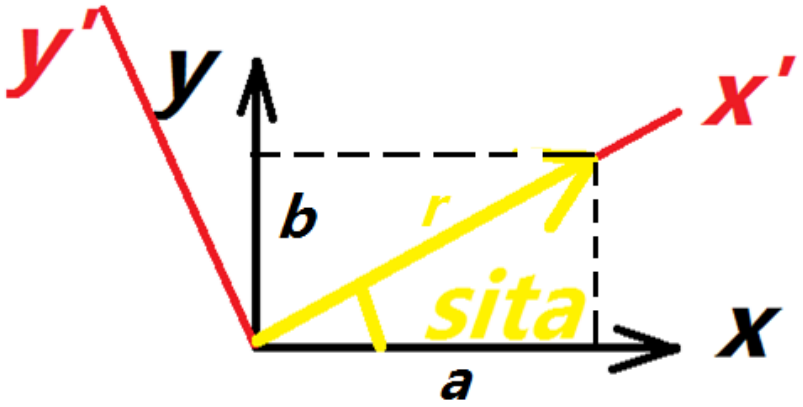
Method 2: Givens Rotation

General Idea: Coordinate Transform, xy to x'y', zeros one of axis component

Suppose vector v=(a,b) in xy coordinate and rotation matrix G between xy and x'y'

$$Gv = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

Two Degree Coordinate



Expand to N Degree Coordinate

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

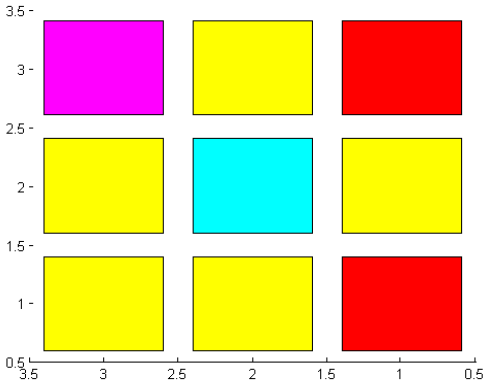
$$g_{k\,k} = 1 \quad \text{for } k \neq i,\ j$$
$$g_{i\,i} = c$$
$$g_{j\,j} = c$$
$$g_{j\,i} = -s$$
$$g_{i\,j} = s \quad \text{for } i > j$$

$$r \leftarrow \sqrt{a^2 + b^2}$$
$$c \leftarrow a/r$$
$$s \leftarrow -b/r.$$

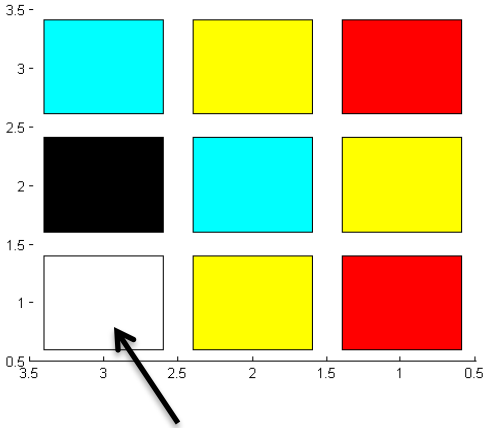
Improvement of Hessenberg Reduction

Example

Original Matrix



Resulted Matrix



Zero this entry

Code by Matlab

```
clc
clear
A=[6 5 6 ; 5 1 4 ; 5 4 3 ];
n=length(A);
for k=1:n-2
for i=k+2:n
    if A(i,k)~=0
        G=givens(A,i,k)
        A=G*A
    end

end
end
```

```
function G=givens(A,r,c)
%comeback gives rotation matrix
% A=[6 5 0; 5 1 4; 0 4 3];
% r=2;c=1;
n=size(A);
if r>=c
    a=A(c,c);
    b=A(r,c);
else
    a=A(c,c);
    b=A(r,c);
end

rt=sqrt(a^2+b^2);
cs=a/rt;
ss=-b/rt;
G(r,r)=cs;
G(c,c)=cs;
    if r>=c
    G(r,c)=ss;
    G(c,r)=-ss;
    else
    G(r,c)=-ss;
    G(c,r)=ss;
    end
   for i=1:n
       if i~=r && i~=c
           G(i,i)=1;
       end
   end

end
```

Improvement of Hessenberg Reduction

Implemetion1: Hessenburg Reduction by using Householder Matrix

General Householder Reflection

A  P_1AP_1  Full Again

Solution: Hessenburg Reduction

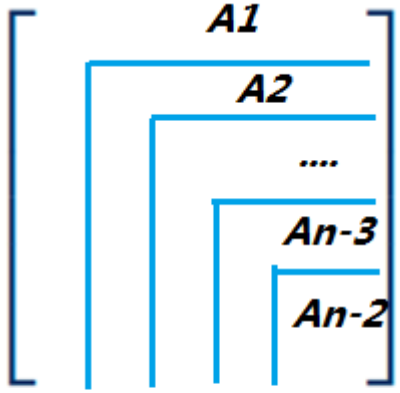
Form the Householder matrix from A (n-1) by (n-1) change the householder matrix as:

First Transform Matrix $P_1 = \begin{bmatrix} 1 & 0 \\ 0 & \hat{P}_1 \end{bmatrix}$

First Transform $P_1 A P_1^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \hat{P}_1 \end{bmatrix} A \begin{bmatrix} 1 & 0 \\ 0 & \hat{P}_1 \end{bmatrix}^{-1}$

Iteration

A1=A (n-1,n-1)
A2=A (n-2,n-2)
⋮
An-2



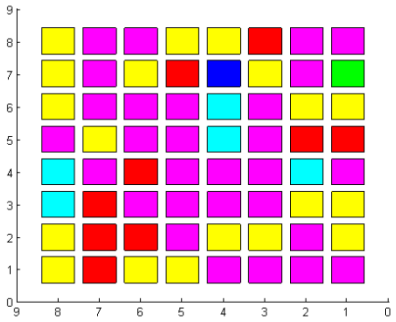
Improvement of Hessenberg Reduction

Hessenburg Reduction by Matlab

```
clc
clear
% A=[4 1 -2 2; 1 2 0 1; -2 0 3 -2; 2 1 -2 -1];
A=rand(8,8);
m=length(A);
for k=1:m-2
    P=Householder_matrix(A(k:end,k:end)); %call for the house holder
    if k==1
        PP=P;
    else
    PP=[eye(k-1) zeros(k-1,m-k+1);zeros(m-k+1,k-1) P];
    end
    A=PP*A*PP;
end
R=A

function P=Householder_matrix(A)
% come back the householder matrix P
k=1;
n=length(A);
aerfa=-1*sign(A(k+1,k))*sqrt(sum(A((k+1:n),k).^2));
r=sqrt(0.5*(aerfa^2-A(k+1,k)*aerfa));
v(1:k)=0;
v(k+1)=(A(k+1,k)-aerfa)/2/r;
for i=k+2:n
    v(i)=A(i,k)/2/r;
end
v=v';
P=eye(n,n)-2*v*v';
% A=P*A*P;
end
```

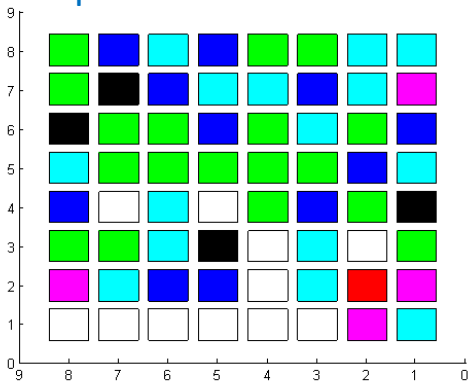
Improvement of Hessenberg Reduction

A 8x8 Random Square Matrix =

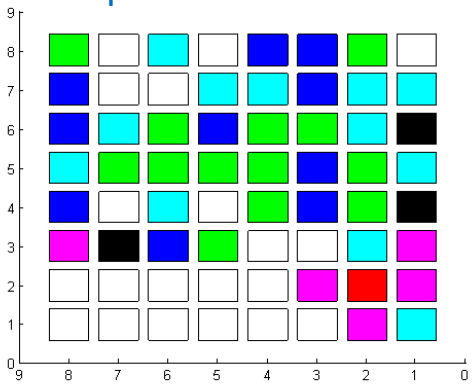


Hessenburg Reduction

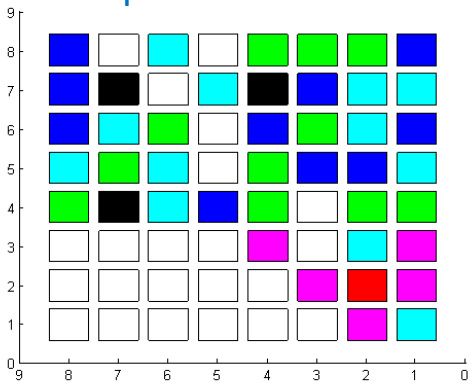
Loop1



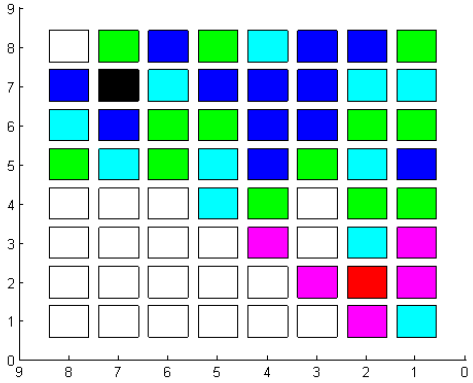
Loop2



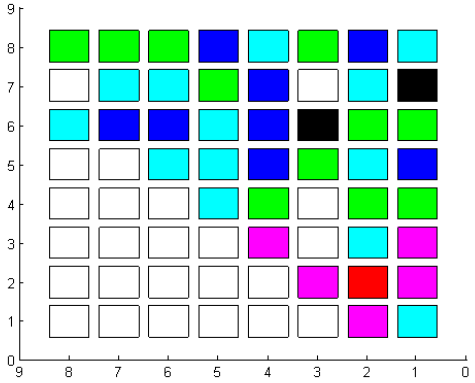
Loop3



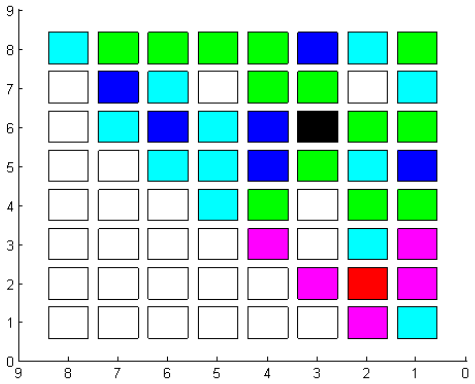
Loop4



Loop5



Loop6



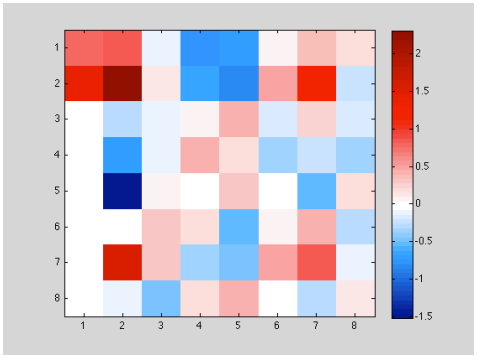
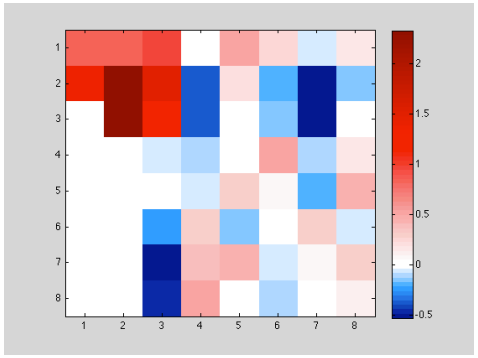
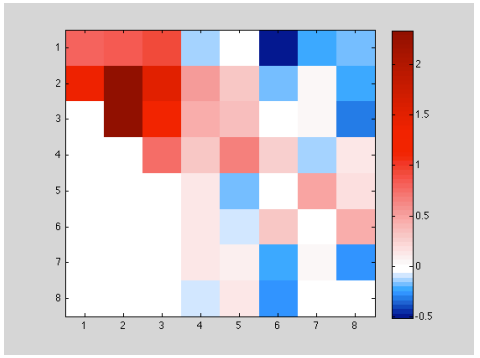
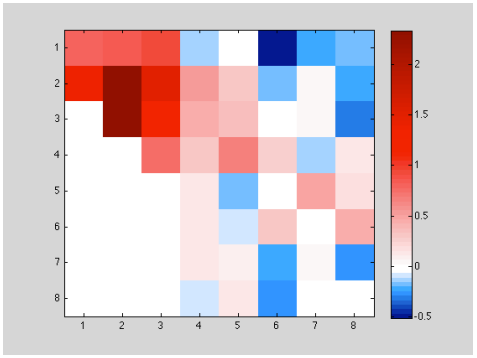
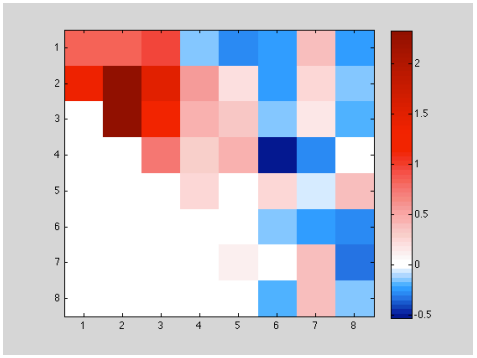
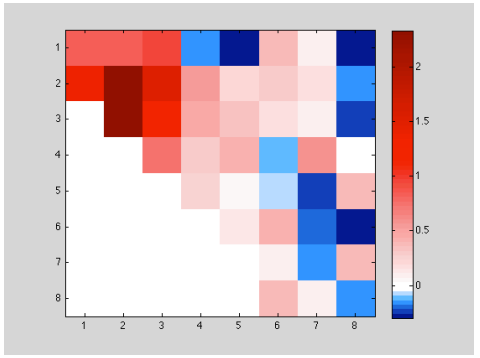
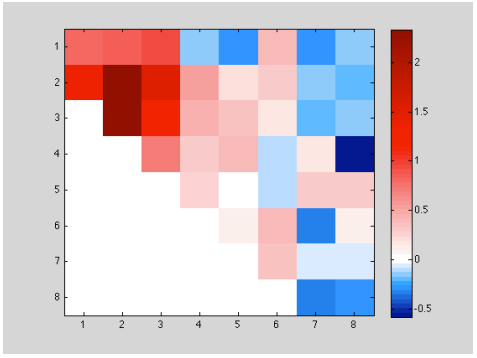
Givens Rotation

```
function [g]=givens(x,j,i)
% Function of Givens Rotation

% x: Input matrix
% i: Row affected by the zeroing operation
% j: Row to be zeroed (column 1)
% G: Givens rotation matrix
g=eye(length(x)); %Initialize givens matrix
xi=x(i,1); %Identify the ordinate pair over which the rotation happens
xj=x(j,1);
r=sqrt(xi^2+xj^2); %Find length of vector r from origin to this point
%Populate rotation matrix with the necessary elements
cost=xi/r;
sint=xj/r;
g(i,i)=cost;
g(i,j)=-sint;
g(j,i)=sint;
g(j,j)=cost;
end
```

Hessenberg Reduction through Givens Rotation

```
function [R]=hessen1(A)
% Hessenberg Reduction by using Givens Method
count=1;
n=size(A);
G=eye(size(A)); %Gives rotation matrix accumulator
R=A; %Copy A into R
for j=1:n-2 %Outer loop (determines columns being zeroed out)
    for i=n:-1:j+2 %Inner loop (successively zeroes jth column)
        giv=givens(R(j:n, j:n), i-j+1, i-j);
        giv=blkdiag(eye(j-1), giv); %Resize rotator to full size
        G=giv*G; %Accumulate G which give a Q in the end
        %Perform similarity transform
        R=giv'*R;
        R=R*giv;
        count=count+1;
    end
end
end
```

Result

Performance Improvement Criteria

- CPU
  - Cache Locality
  - Memory Bandwidth
  - Memory bound
  - Computational Efficiency (Number of calcs)
  - Size of data
- GPU
  - Large latency overhead
  - Memory Bandwidth
  - Memory bound
  - Parallelization Algorithms
  - GPU Targetting

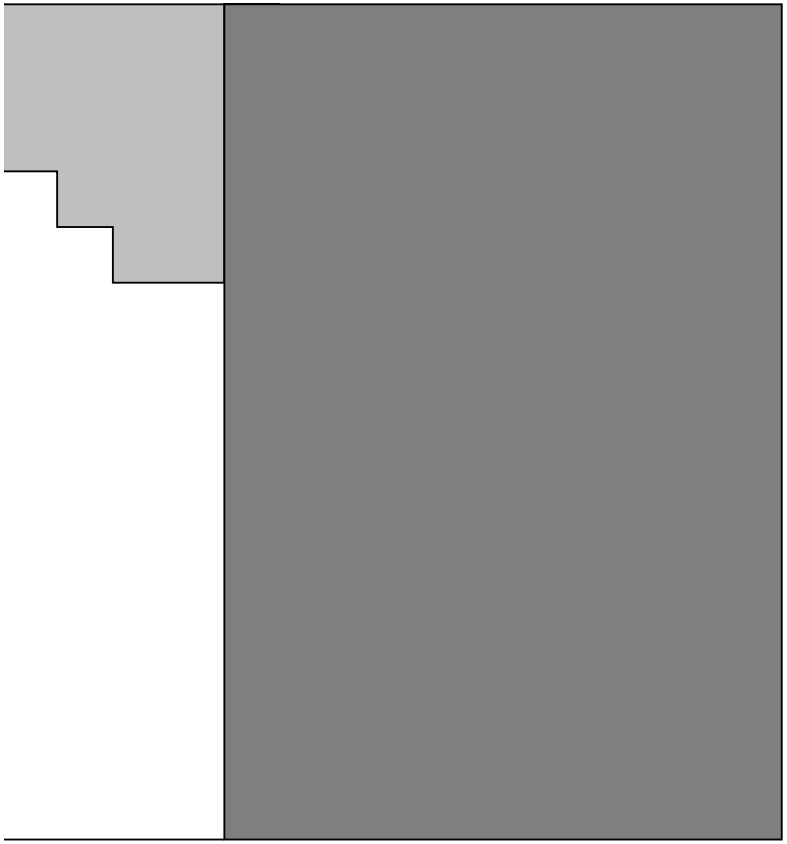
Blocked Operations (SIMD)

- Operate on large chunks of data
- Provides cache locality
- No pipeline bubbles
- Streaming extensions (SSEx) on modern processors target these operations
- Order of efficiency (ascending)
  - Vector-Vector (Level 1)
  - Vector-Matrix (Level 2)
  - Matrix-Matrix (Level 3)

Parallel Operations

- Modern CPUs can operate on two/more sets of data simultaneously and independently
- Still share memory, so cache locality still important
- Important: Algorithm needs to be rewritten to find independent operations without dependencies on previous values
- Synchronization very important
- GPU perfect for this, massively parallel processing pipeline (‘stream processors’)
- ASIC(Application Specific Ics) and FPGAs (Field Programmable Gate Arrays) are perfect for this

Block Hessenberg

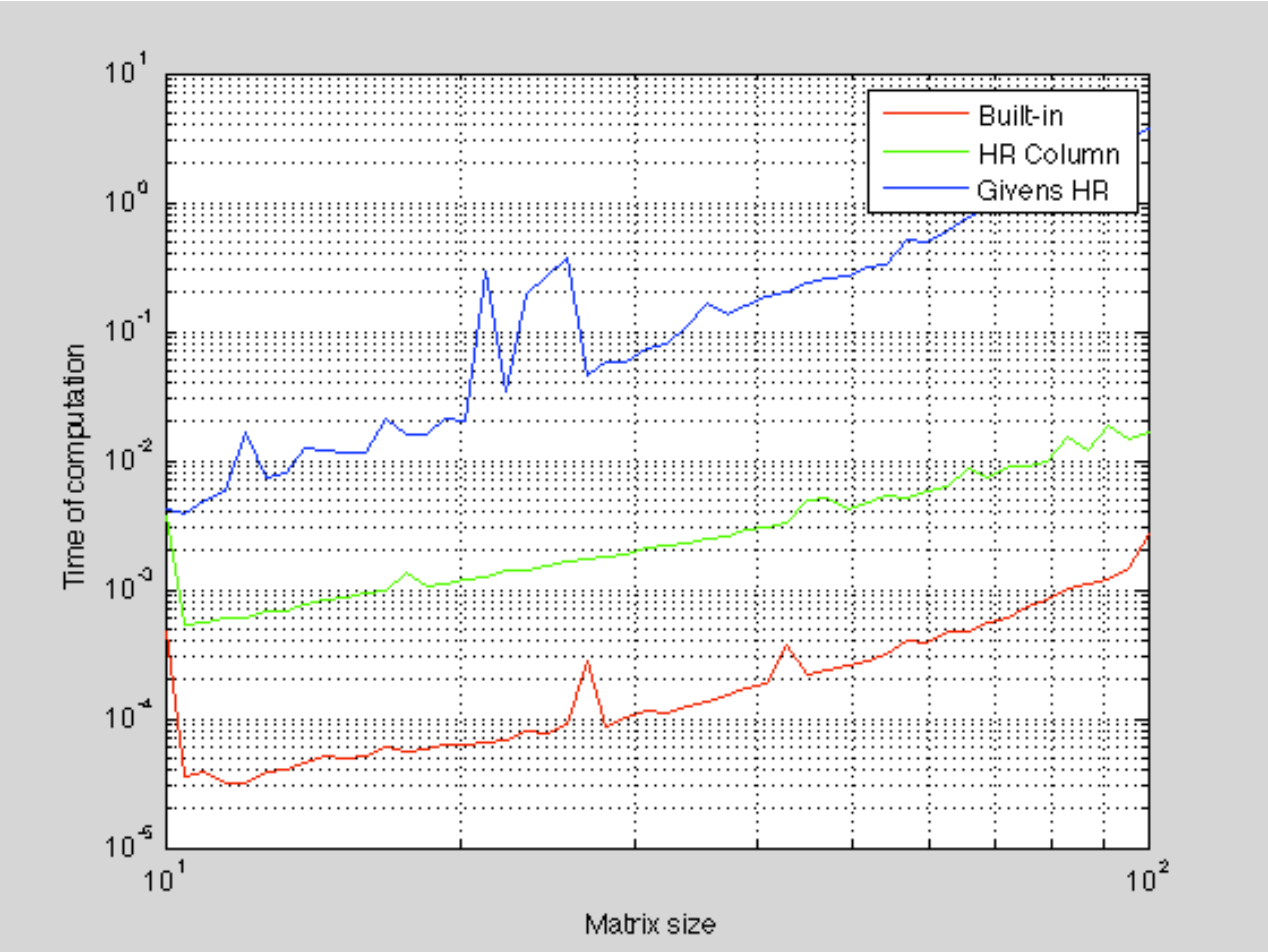


$\tilde{A}^{(k)}(i,j) = 0$

$\tilde{A}^{(k)}(i,j) = A^{(k)}(i,$

$\tilde{A}^{(k)}(i,j) = A(i,j)$

Results


Built-in
HR Column
Givens HR
Time of computation
Matrix size
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{1}
10^{2}

Results

norm1 =

0

norm2 =

6.013108476912430e-13

norm3 =

66.823468331017068

eig_norm =

0

eig_norm1 =

4.965725351883417e-14

eig_norm2 =

5.924617829737880e-14

Conclusion

- Hessenberg Reduction reduces complexity of dense eigen value solvers
- Column householder and Givens rotation
- Study of computing architecture tells us how to improve performance
- Blocking (SIMD) and parallelism
- Blocking algorithm fastest