Orthogonal Iteration

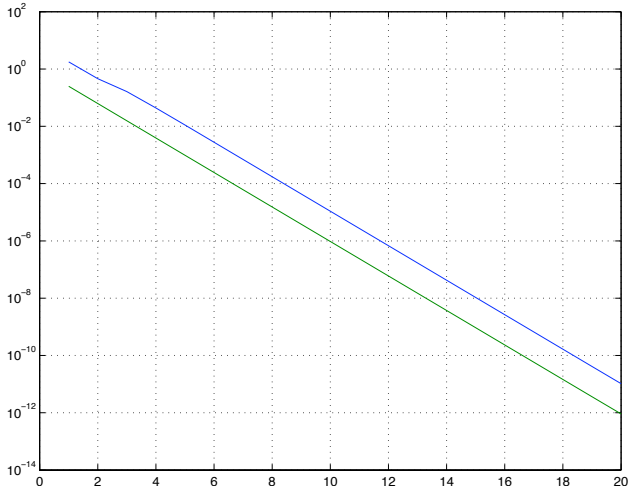
```
function Q = OrthogonalIteration(A,iter)

n = size(A,1);
Q = eye(n);

for k = 1:iter
    Y = A*Q;
    [Q,R] = qr(Y);
end
```

Example

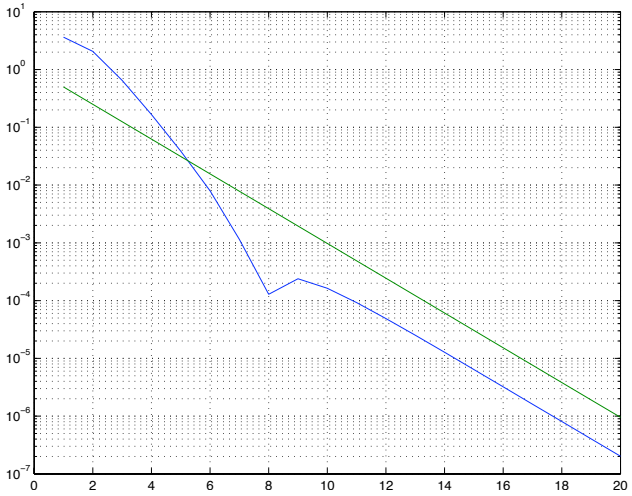
```
lambda = [1 2 4 8 32];
n = 5;
X = randn(n);
A = X*diag(lambda)*inv(X);
Q = OrthogonalIteration(A,iter);
Q'*A*Q


ans =

   32.0000    8.0920   24.8092   10.8339    7.4218
   -0.0000    8.0000    0.0750    1.6064    0.2155
    0.0000   -0.0000    4.0000   -3.4723   -0.2473
   -0.0000    0.0000    0.0000    2.0000    0.0182
   -0.0000   -0.0000   -0.0000   -0.0000    1.0000
```

Convergence of eigenvalues

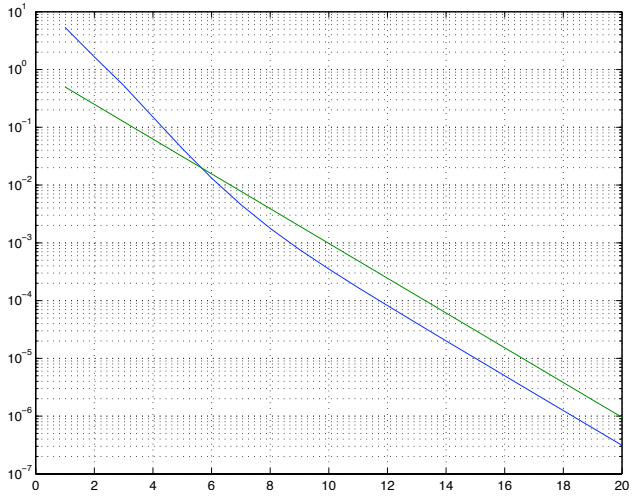
```
for  k = 1:n,
    if k == 1,
        ratio = abs(lambda(k+1)/lambda(k));
    elseif k == n,
        ratio = abs(lambda(k)/lambda(k-1));
    else
        ratio = max(abs(lambda(k+1)/lambda(k)),
                                abs(lambda(k+1)/lambda(k)));
    end

    error = abs(lambda_iter(k,:)-lambda(k));
    ix = 1:iter;
    figure; semilogy(ix,error,ix,ratio.^ix)
    grid on
end
```



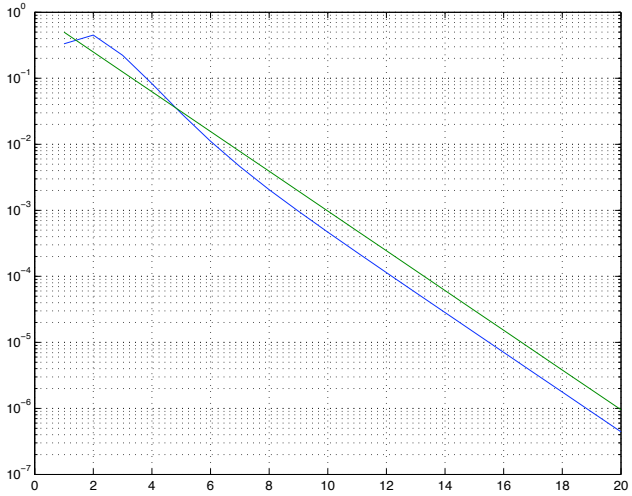
λ_1



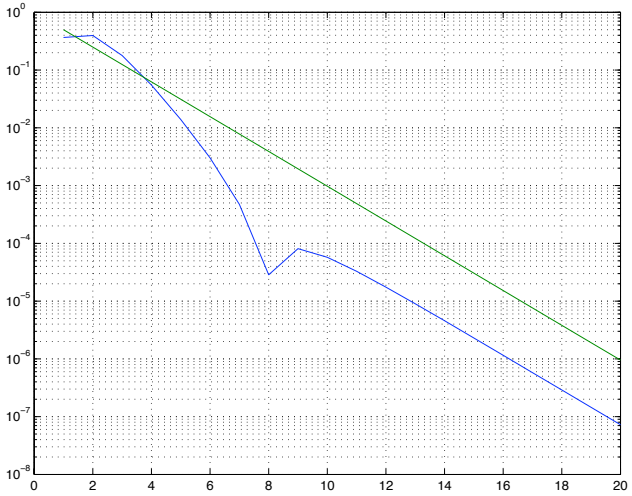
λ_2



λ_3



λ_4



λ_5

QR Iteration

```
function Schur = QRIteration(A,iter);

Schur = A;
for k = 1:iter,
    [Q,R] = qr(Schur);
    Schur = R*Q;
end
```

Orthogonal and QR iterations are the same!

```
Schur = QRIteration(A,iter)

Schur =

   32.0000    8.0920   24.8092   10.8339   -7.4218
   -0.0000    8.0000    0.0750    1.6064   -0.2155
    0.0000   -0.0000    4.0000   -3.4723    0.2473
   -0.0000    0.0000    0.0000    2.0000   -0.0182
   -0.0000    0.0000    0.0000    0.0000    1.0000
```

This is the same as before (except for a multiplication by -1)!

QR Iteration with shift

Implicit shift is here taken to be $\boldsymbol{A}_i(n,n)$ in the QR iteration

```
function Schur = QRIterationWithShift(A,iter);

Schur = A;
n = size(Schur,1);

for k = 1:iter,
    shift = Schur(n,n);
    [Q,R] = qr(Schur-shift*eye(n));
    Schur = R*Q + shift*eye(n);
end
```

Convergence

Same matrix as before: $A = X \,\mathrm{diag}(1, 2, 4, 8, 32)\, X^{-1}$

```
% Last row of A
A(5,:)

ans =

   -6.4337    7.6650  -18.2837    1.6667   21.5406


for k = 1:iter
Schur = QRIterationWithShift(A,k);
x(k,:)=Schur(5,:);
end
```

x =

-3.4388	5.9051	-3.5675	-0.7900	24.6298
-1.4488	2.7706	-3.2813	-2.3696	33.3064
0.0576	-0.1172	0.1036	0.1252	32.0947
-0.0002	0.0004	-0.0003	-0.0005	32.0003
0.0000	-0.0000	0.0000	0.0000	32.0000
-0.0000	0.0000	-0.0000	-0.0000	32.0000
0.0000	-0.0000	0.0000	0.0000	32.0000
-0.0000	0.0000	-0.0000	-0.0000	32.0000
0.0000	-0.0000	0.0000	0.0000	32.0000
-0.0000	0.0000	-0.0000	-0.0000	32.0000

Show quadratic convergence

```
format long, x(1:iter,5)

ans =

  24.62981360928721
  33.30641401742761
  32.09467193680406
  32.00034342913918
  32.00000000451928
  32.00000000000001
  32.00000000000001
  32.00000000000001
  32.00000000000001
  32.00000000000001
```

Eigenvalue computation

```
n = 5;
X = randn(n);
lambda = [1 1.5 3 5 10];
A = X*diag(lambda)*inv(X);

H = hess(A) % reduction to Hessenberg form

H =

    1.1658    1.0439   -0.8343   -2.0977   -1.6628
   -3.3894    7.9269   -6.0601  -14.7435   -8.8604
         0   -2.0500    8.5224    8.1747    3.9817
         0         0   -1.6334   -0.4954   -1.1119
         0         0         0    1.0397    3.3803

[Q,T] = schur(H) % Schur factorization (QR iteration with shifts)
```

Q =

```
    0.1256    0.1256   -0.4002    0.8608   -0.2596
    0.7996    0.5656    0.0854   -0.1171    0.1405
   -0.5803    0.7688    0.1412    0.1001    0.2054
    0.0888   -0.2217    0.7880    0.4779    0.3059
    0.0140   -0.1554   -0.4378    0.0829    0.8816
```

T =

```
   10.0000    1.5198   -8.7700  -12.9513  -13.9804
         0    5.0000    0.6462   -2.8755   -1.2923
         0         0    1.0000   -0.8835   -1.5588
         0         0         0    1.5000   -0.4206
         0         0         0         0    3.0000
```

Eigenvalue computation with a symmetric matrix

```
A = X'*X;
H = hess(A)

H =

    1.5623   -0.1974         0         0         0
   -0.1974    4.5136   -3.2166         0         0
         0   -3.2166    3.7755    1.3253         0
         0         0    1.3253    5.0432    4.2772
         0         0         0    4.2772    5.4968

[Q,T] = schur(H)

Q =

   -0.0613    0.7740    0.6296   -0.0251    0.0033
   -0.4338    0.3163   -0.4012    0.7293   -0.1378
   -0.5826    0.2506   -0.3911   -0.6280    0.2247
    0.5340    0.3339   -0.3579    0.1046    0.6814
   -0.4284   -0.3557    0.4019    0.2495    0.6828
```

T =

0.1657	0	0	0	0
0	1.4817	0	0	0
0	0	1.6881	0	0
0	0	0	7.2902	0
0	0	0	0	9.7658

Compare CPU times

```
% Problem size
n = 1000;

X = randn(n);                          % Nonsymmetric matrix
tic, D = eig(X); toc                   % Eigenvalues only
Elapsed time is 14.000721 seconds.

tic, [V,D] = eig(X); toc               % Eigenvalues and eigenvectors
Elapsed time is 42.917404 seconds.


Y = X+X';                              % Symmetric matrix
tic, D = eig(Y); toc                   % Eigenvalues only
Elapsed time is 1.959447 seconds.

tic, [V,D] = eig(Y); toc               % Eigenvalues and eigenvectors
Elapsed time is 9.417701 seconds.
```

```
n = 1000;                                % Problem size

X = randn(n);                            % Nonsymmetric matrix
tic, H = hess(X); toc                    % Hessenberg reduction
Elapsed time is 3.087571 seconds.

tic, T = schur(H); toc                   % Schur factorization
Elapsed time is 3.087571 seconds.


Y = X+X’;                                % Symmetric matrix
tic, H = hess(Y); toc                    % Hessenberg reduction
Elapsed time is 1.893651 seconds.

tic, T = schur(H); toc                   % Schur factorization
Elapsed time is 1.832920 seconds.
```