Overview

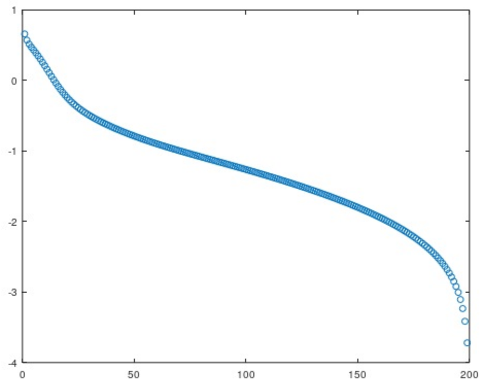
- Simultaneous iteration
- QR algorithm
- QR iteration with shifts

Algorithm

Given $\boldsymbol{A} \in \mathbb{R}^{m\times m}$
Choose $\boldsymbol{v}^{(0)}$
for $k = 1, 2, \ldots$
$\boldsymbol{v}^{(k)} = \boldsymbol{A}\boldsymbol{v}^{(k-1)}$
$\boldsymbol{v}^{(k)} = \boldsymbol{v}^{(k)} / \|\boldsymbol{v}^{(k)}\|$
$\lambda^{(k)} = [\boldsymbol{v}^{(k)}]^T \boldsymbol{A}\boldsymbol{v}^{(k)}$

$$e_k \cong a\left(\frac{\lambda_2}{\lambda_1}\right)^{2k}$$

$$\lg e_k = 2\lg\left|\frac{\lambda_2}{\lambda_1}\right| k + \lg a$$



Implementation

```
octave] function [lambda,iters]=power(A,v,maxiter)
          [m,n]=size(A); iters=zeros(maxiter,1);
          for k=1:maxiter
            v=A*v; v=v/norm(v); lambda=v'*A*v;
            iters(k)=lambda;
          end;
        end
```

```
octave] cd ~/courses/MATH662; load L16K; A=Expression1;
        m=size(A)(1); v=rand(m,1); v=v/norm(v); mxit=200;
        [lambda,iters]=power(A,v,mxit);
```

```
octave] lgerr = log10(abs(iters-lambda)); k=(1:mxit)'; clf;
        plot(k,lgerr,'o');
        rng=50:150; p=polyfit(k(rng),lgerr(rng),1);
```

```
octave] load L16L; L=Expression1; l21=L(2)/L(1);
        [lambda,L(1),2*log10(l21),p(1),cond(A)]
ans =

   2.9689e+01   2.9983e+01  -9.1927e-03  -5.8720e-03   6.6015e+16
```

```
octave]
```

- Power iteration: $\boldsymbol{v}^{(k)} = \boldsymbol{A}\boldsymbol{v}^{(k-1)}$, $\boldsymbol{A} \in \mathbb{R}^{m \times m}$, $\boldsymbol{A} = \boldsymbol{A}^T$
- Simultaneous iteration: Choose n initial vectors $\boldsymbol{V}^{(0)} = \left(\begin{array}{ccc} \boldsymbol{v}_1^{(0)} & \ldots & \boldsymbol{v}_n^{(0)} \end{array} \right)$,

$$\boldsymbol{V}^{(k)} = \boldsymbol{A}\boldsymbol{V}^{(k-1)} = \cdots = \boldsymbol{A}^k \boldsymbol{V}^{(0)}$$

- $\boldsymbol{A}$ symmetric is unitarily diagonalizable, $\boldsymbol{A}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{\Lambda}$, and since $\boldsymbol{Q}$ is a basis for $\mathbb{R}^m$

$$\boldsymbol{V}^{(0)} = \boldsymbol{Q}\boldsymbol{C}, \boldsymbol{V}^{(1)} = \boldsymbol{A}\boldsymbol{V}^{(0)} = \boldsymbol{A}\boldsymbol{Q}\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{C}, \boldsymbol{V}^{(2)} = \boldsymbol{A}\boldsymbol{V}^{(1)} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{C}$$

$$\boldsymbol{V}^{(k)} = \boldsymbol{A}^k \boldsymbol{V}^{(0)} = \boldsymbol{A}^k\boldsymbol{Q}\boldsymbol{C} = \boldsymbol{A}^k(\begin{array}{ccc} \boldsymbol{q}_1 & \ldots & \boldsymbol{q}_m \end{array})\boldsymbol{C} = (\begin{array}{ccc} \lambda_1^k\boldsymbol{q}_1 & \ldots & \lambda_m^k\boldsymbol{q}_m \end{array})\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{\Lambda}^k\boldsymbol{C}$$

- Assume $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > |\lambda_{n+1}| \geqslant \cdots \geqslant |\lambda_m|$. For large k, with $\boldsymbol{Q}_n = (\begin{array}{ccc} \boldsymbol{q}_1 & \ldots & \boldsymbol{q}_n \end{array})$,

$$\boldsymbol{V}^{(k)} \cong \boldsymbol{Q}_n \boldsymbol{\Lambda}_n^k \boldsymbol{C}, \text{with } \boldsymbol{Q}_n = (\begin{array}{ccc} \boldsymbol{q}_1 & \ldots & \boldsymbol{q}_n \end{array}), \boldsymbol{\Lambda}_n = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

- However condition number $\mu(\boldsymbol{V}^{(k)}) = |\lambda_1 / \lambda_n|^k$ also grows

- Avoid ill-conditioning ($\mu(\boldsymbol{V}^{(k)})$ large) by orthonormalization at each step

Algorithm

Choose $\boldsymbol{Q}_n^{(0)} \in \mathbb{R}^{m\times n}$, $(\boldsymbol{Q}_n^{(0)})^T \boldsymbol{Q}_n^{(0)} = \boldsymbol{I}_n$
for $k = 1, 2, \ldots$
  $\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{Q}_n^{(k-1)}$
  $\boldsymbol{Q}_n^{(k)} \boldsymbol{R}^{(k)} = \boldsymbol{Z}$

- Note that $\mathcal{R}(\boldsymbol{Q}_n^{(k)}) = \mathcal{R}(\boldsymbol{A}^k \boldsymbol{Q}_n^{(0)})$ (same column space), but $\mu(\boldsymbol{Q}_n^{(k)}) = 1$
- Subspace iteration converges quadratically (from Rayleigh quotient computation)
- Choose $\boldsymbol{Q}_n^{(0)} = \boldsymbol{I}$ and reorganize computation to obtain the $\boldsymbol{QR}$ algorithm

Subspace iteration	$\boldsymbol{QR}$ iteration
$\boldsymbol{Q}_m^{(0)} = \boldsymbol{I}$	$\boldsymbol{A}^{(0)} = \boldsymbol{A}$
$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{Q}_m^{(k-1)}$	$\boldsymbol{Q}^{(k)}\boldsymbol{R}^{(k)} = \boldsymbol{A}^{(k-1)}$, $\boldsymbol{A}^k = \boldsymbol{Q}_m^{(k)} \boldsymbol{R}^{(k)}$,
$\boldsymbol{Q}_m^{(k)}\boldsymbol{R}^{(k)} = \boldsymbol{Z}$	$\boldsymbol{A}^{(k)} = \boldsymbol{R}^{(k)}\boldsymbol{Q}^{(k)}$
$\boldsymbol{A}^{(k)} = [\boldsymbol{Q}_m^{(k)}]^T \boldsymbol{A}\boldsymbol{Q}_m^{(k)}$	$\boldsymbol{Q}_m^{(k)} = \boldsymbol{Q}^{(1)} \ldots \boldsymbol{Q}^{(k)}$

Subspace iteration	QR iteration
$Q_m^{(0)} = I$	$A^{(0)} = A$
$Z = AQ_m^{(k-1)}$	$Q^{(k)}R^{(k)} = A^{(k-1)}$, $A^k = Q_m^{(k)} R^{(k)}$,
$Q_m^{(k)}R^{(k)} = Z$	$A^{(k)} = R^{(k)}Q^{(k)}$
$A^{(k)} = [Q_m^{(k)}]^T A Q_m^{(k)}$	$Q_m^{(k)} = Q^{(1)} \ldots Q^{(k)}$

- Sequence of Rayleigh quotient matrices:

	Subspace iteration	QR iteration
	$Q_m^{(1)}R^{(1)} = A$,	$Q^{(1)}R^{(1)} = A^{(0)} = A \Rightarrow R^{(1)} = [Q^{(1)}]^T A^{(0)}$,
	$Q_m^{(1)} =$	$Q^{(1)}$
$A^{(1)} =$	$[Q_m^{(1)}]^T A Q_m^{(1)} =$	$R^{(1)}Q^{(1)} = [Q^{(1)}]^T A^{(0)} Q^{(1)}$ ✓
	$Q_m^{(2)}R^{(2)} = AQ_m^{(1)}$,	$Q^{(2)}R^{(2)} = A^{(1)} \Rightarrow R^{(2)} = [Q^{(2)}]^T A^{(1)}$,
	$Q_m^{(2)} =$	$Q^{(1)}Q^{(2)}$
$A^{(2)} =$	$[Q_m^{(2)}]^T A Q_m^{(2)} =$	$R^{(2)}Q^{(2)} = [Q^{(2)}]^T A^{(1)} Q^{(2)}$ ✓

Algorithm

Given $\boldsymbol{A} \in \mathbb{R}^{m \times m}$

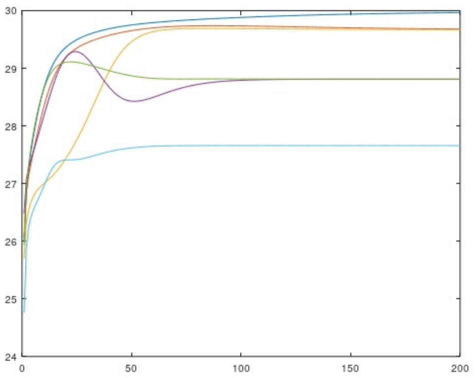
Choose $\boldsymbol{Q}_n^{(0)} \in \mathbb{R}^{m \times n}$

for $k = 1, 2, \ldots$

$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{Q}_n^{(k-1)}$

$\boldsymbol{Q}_n^{(k)}\boldsymbol{R}^{(k)} = \boldsymbol{Z}$

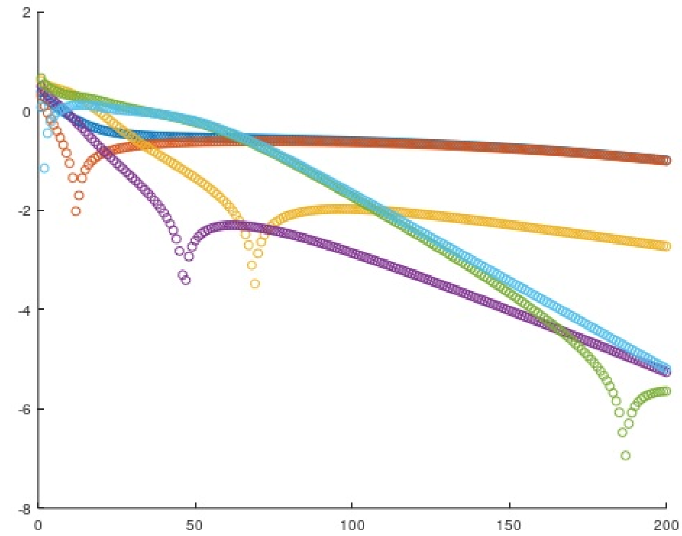
$\boldsymbol{A}^{(k)} = \left[\boldsymbol{Q}_n^{(k)}\right]^T \boldsymbol{A}\boldsymbol{Q}_n^{(k)}$



Implementation

```
octave] function iters=subsp(A,Q,maxiter)
          [m,n]=size(Q); iters=zeros(maxiter,n);
          for k=1:maxiter
            Z=A*Q; [Q,R]=qr(Z,0); Ak=Q'*A*Q;
            iters(k,:)=diag(Ak);
          end;
        end
octave] cd ~/courses/MATH662; load L16K; A=Expression1;
        m=size(A)(1); n=6; [Q,R]=qr(randn(m,n),0); mxit=200;
        iters=subsp(A,Q,mxit);
octave] k=(1:mxit)'; clf; plot(k,iters(:,1)); hold on;
        for j=1:n
          plot(k,iters(:,j));
        end;
octave] load L16L; L=Expression1; [iters(mxit,1:n);L(1:n)']
ans =

   29.880   29.768   29.669   28.811   28.811   27.659
   29.983   29.667   29.667   28.811   28.811   27.659

octave]
```

```
octave] k=(1:mxit)'; clf; hold on; rng=10:50;
        conv = zeros(n,3);
        for j=1:n
         lgerr(:,j) = log10(abs(iters(:,j)-L(j)));
         plot(k,lgerr(:,j),'o');
         p=polyfit(k(rng),lgerr(rng,j),1);
         pth=2*log10(L(j+1)/L(j));
         conv(j,:)=[pth p(1) abs(pth/p(1))];
        end;
        conv
```




```
conv =

  -9.1927e-03  -9.0161e-03   1.0196e+00
  -7.7146e-16   1.9221e-02   4.0137e-14
  -2.5442e-02  -4.4005e-02   5.7815e-01
  -9.6433e-17  -7.3820e-02   1.3063e-15
  -3.5445e-02  -1.3756e-02   2.5766e+00
  -2.8930e-16  -7.9476e-03   3.6400e-14
```

Algorithm

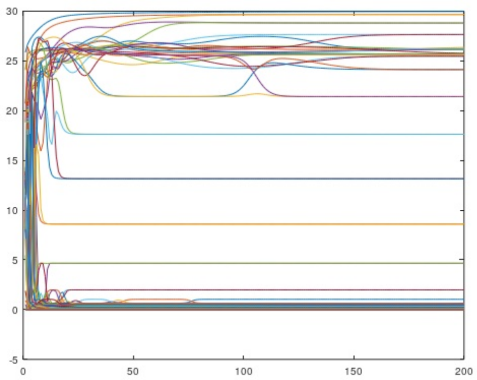
Given $\boldsymbol{A} \in \mathbb{R}^{m \times m}$

$\boldsymbol{A}^{(0)} = \boldsymbol{A}$

for $k = 1, 2, \ldots$

$\boldsymbol{Q}^{(k)} \boldsymbol{R}^{(k)} = \boldsymbol{A}^{(k-1)}$

$\boldsymbol{A}^{(k)} = \boldsymbol{R}^{(k)} \boldsymbol{Q}^{(k)}$



Implementation

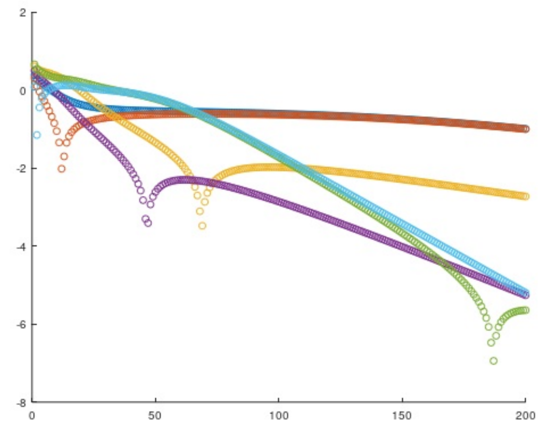
```
octave] function iters=QR(A,maxiter)
          m=size(A)(1); iters=zeros(maxiter,m); Ak=A;
          for k=1:maxiter
            [Q,R]=qr(Ak); Ak=R*Q;
            iters(k,:)=diag(Ak);
          end;
        end
octave] cd ~/courses/MATH662; load L16K; A=Expression1;
        mxit=200; iters=QR(A,mxit);
octave] k=(1:mxit)'; clf; plot(k,iters(:,1)); hold on; n=6;
        for j=1:3
          plot(k,iters(:,j));
        end;
octave] load L16L; L=Expression1; [iters(mxit,1:n);L(1:n)']
ans =

   29.974   29.667   29.676   28.811   28.811   27.659
   29.983   29.667   29.667   28.811   28.811   27.659

octave]
```

```
octave] k=(1:mxit)'; clf; hold on; rng=10:50;
        conv = zeros(n,3);
        for j=1:n
         lgerr(:,j) = log10(abs(iters(:,j)-L(j)));
         plot(k,lgerr(:,j),'o');
         p=polyfit(k(rng),lgerr(rng,j),1);
         pth=2*log10(L(j+1)/L(j));
         conv(j,:)=[pth p(1) abs(pth/p(1))];
        end;
        conv
```

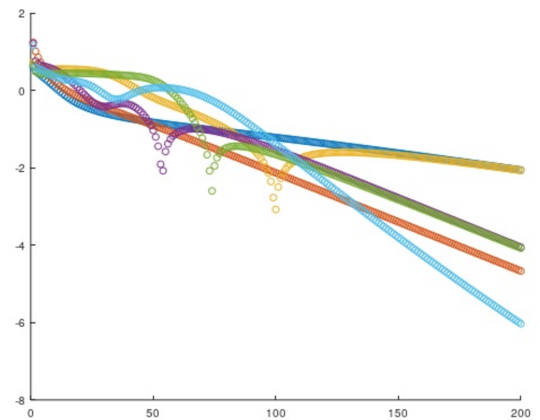
```
conv =

  -9.1927e-03  -2.0365e-02   4.5140e-01
  -7.7146e-16  -2.6153e-02   2.9497e-14
  -2.5442e-02  -1.8797e-02   1.3535e+00
  -9.6433e-17  -2.9572e-02   3.2609e-15
  -3.5445e-02  -3.5104e-03   1.0097e+01
  -2.8930e-16  -1.0764e-02   2.6878e-14
```

```
octave]
```

Subspace iteration



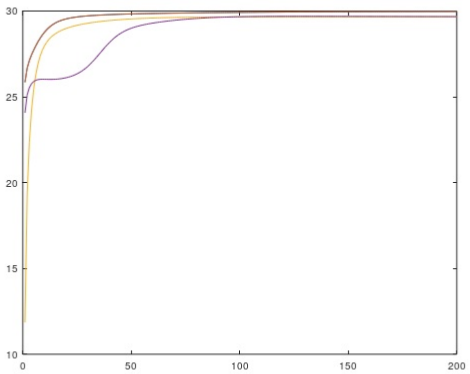
QR

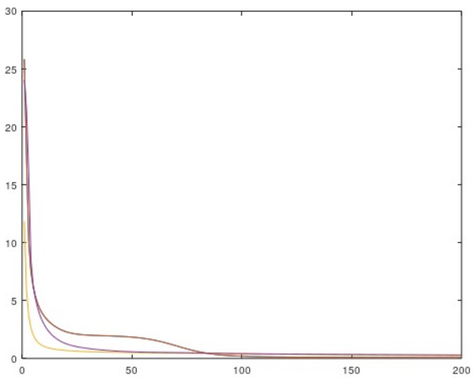


- Subspace iteration is the stabilized, multidimensional version of power iteration
- QR is full space iteration
- Construct the full-space, stabilized version of Rayleigh quotient iteration
- Choose $Q_n^{(0)} = I$ and reorganize computation to obtain the QR algorithm

QR iteration with shifts	QR iteration
$(Q^{(0)})^T A^{(0)} Q^{(0)} = A$	$A^{(0)} = A$
$Q^{(k)} R^{(k)} = A^{(k-1)} - \mu^{(k)} I$	$Q^{(k)} R^{(k)} = A^{(k-1)}$
$A^{(k)} = R^{(k)} Q^{(k)} + \mu^{(k)} I$	$A^{(k)} = R^{(k)} Q^{(k)}$
Deflate if some $A_{j,j+1}^{(k)} < \varepsilon$	
$\begin{pmatrix} A_1 & \\ & A_2 \end{pmatrix} = A^{(k)}$	
$\mathrm{QR}(A_1), \mathrm{QR}(A_2)$	

,





Implementation

```
octave] function iters=QR(A,maxiter)
          m=size(A)(1); iters=zeros(maxiter,m);
          Ak=A; muk=0;
          for k=1:maxiter
            [Q,R]=qr(Ak-muk*eye(m));
            Ak=R*Q+muk*eye(m); muk=max(diag(Ak));
            iters(k,:)=diag(Ak)+muk;
          end;
        end
octave] cd ~/courses/MATH662; load L16K; A=Expression1;
        mxit=200; iters=QR(A,mxit);
octave] k=(1:mxit)'; clf; plot(k,iters(:,1)); hold on; n=6;
        for j=1:3
          plot(k,iters(:,j));
        end;
octave] load L16L; L=Expression1; [iters(mxit,1:n);L(1:n)']
ans =

   29.974   29.667   29.676   28.811   28.811   27.659
   29.983   29.667   29.667   28.811   28.811   27.659

octave]
```