Oplossingen NLA 4

Nico Vervliet, *KU Leuven* *25 mei 2018*

1 QR methode

Opgave 1

```
% A1
L1 = diag(1:4);
P1 = orth(rand(4));
A1 = P1 * L1 * P1';

% A2
L2 = L1;
P2 = orth(rand(4))*diag(logspace(0, 2, 4))*orth(rand(4))';
A2 = (P2*L2)/P2;

% A3
L3 = diag([1 2 3 3]);
L3(3,4) = 1;
P3 = P2;
A3 = (P3*L3)/P3;
```

Met matlab krijgen we de volgende eigenwaarden:

```
ew1 = eig(A1)'
ew2 = eig(A2)'
ew3 = eig(A3)'
```

```
ew1 = eig(A1)'

ew1 =

    4.0000    1.0000    3.0000    2.0000
ew2 = eig(A2)'
```

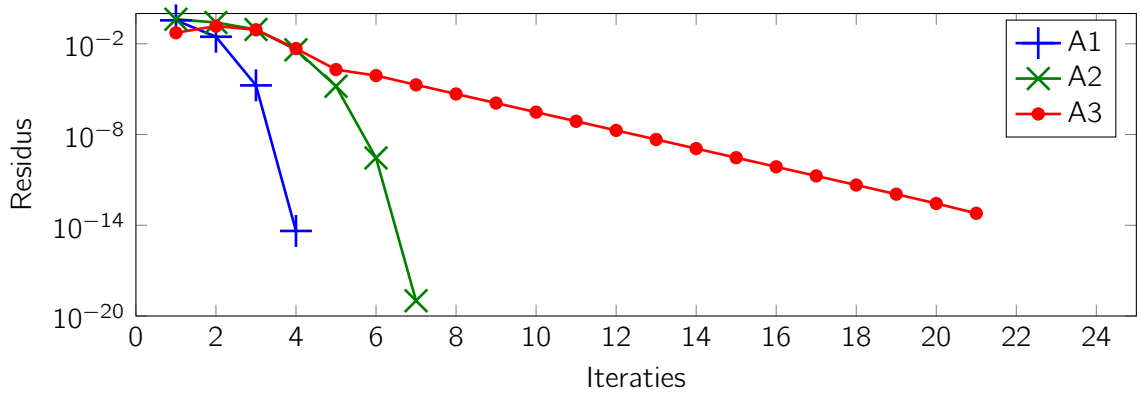
```
ew2 =

    1.0000    2.0000    3.0000    4.0000
ew3 = eig(A3)'

ew3 =

    1.0000    2.0000    3.0000    3.0000
```

We berekenen de eigenwaarden nu met de `nlaqr` methode:

```
[e1, res1] = nlaqr(A1, true);
[e2, res2] = nlaqr(A2, true);
[e3, res3] = nlaqr(A3, true);
e1
e2
e3
```

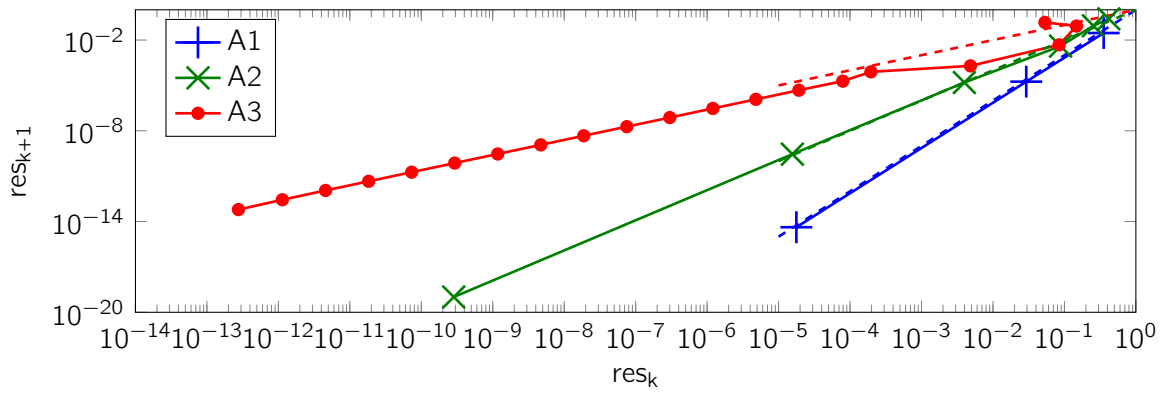
```
e1 =

    3.0000
e2 =

    1.0000
e3 =

    3.0000
```

```
n1 = length(res1); n2 = length(res2); n3 = length(res3);
semilogy(1:n1, res1, '+-', 1:n2, res2, 'x-', 1:n3, res3, '.-')
xlabel('Iteraties');
ylabel('Residus');
legend('A1','A2','A3');
```



De plot geeft de fout in elke iteratiestap. We zien dat voor A_3 de convergentie het traagst is en eerder lineair is. Voor de andere twee matrices is de convergentie snel, maar het is moeilijk om effectief te zien hoe snel. Daarom maken we een tweede plot die de residu uitzet ten opzichte van de vorige residu.

```
loglog(res1(1:n1-1), res1(2:n1), '+-', ...
      res2(1:n2-1), res2(2:n2), 'x-', ...
      res3(1:n3-1), res3(2:n3), '.-');
hold on
t = [1e-5, 1];
plot(t, t.^3, '--', t, t.^2, '--', t, t, '--');
hold off
legend('A1','A2','A3', 'Location', 'northwest');
xlabel('res_k');
ylabel('res_{k+1}');
```



We hebben ook de lijnen voor t, t^2 en t^3 uitgezet om gemakkelijk mee te vergelijken. We zien dat de methode voor A_1 kubisch convergeert, voor A_2 kwadratisch en voor A_3 lineair.

De redenen hiervoor:

- A_1 is symmetrisch
- A_2 is niet-symmetrisch, maar ook niet defectief
- A_3 is defectief

```
[v, e] = eig(A3)
threes = find(abs(diag(e)-3) < 1e-6);
rank(v(:,threes), 1e-6)
```

```
v =

   1.2739e-01   1.7762e-01  -1.7762e-01   1.0762e-01
   1.2600e-02  -4.6590e-01   4.6590e-01  -3.6652e-01
   7.0796e-01   6.4037e-01  -6.4037e-01  -6.9773e-01
  -6.9456e-01  -5.8423e-01   5.8423e-01   6.0601e-01


e =

   1.0000e+00            0            0            0
             0   3.0000e+00            0            0
             0            0   3.0000e+00            0
             0            0            0   2.0000e+00
ans =

     1
```

2 Eigenwaarden bepalen met de Arnoldi iteratie

Opgave 3

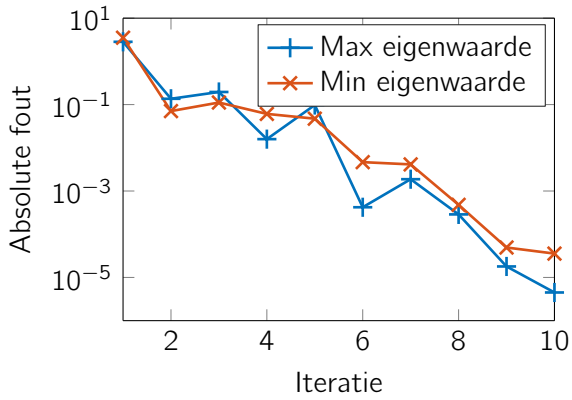
De volgende code berekent het verschil tussen grootste (kleinste) eigenwaarde en de grootste (kleinste) Ritz waarde.

```
function [H, Q, maxd, mind] = arnoldiew(A, b, N)
% function [H, Q, maxd, mind] = arnoldiew(A, b, N)
%
% maximum en minimum eigenwaarden met Arnoldi iteratie
% maxd(:) absolute waarde van verschil tussen
%         grootste eigenwaarde en grootste Ritz waarde
% mind(:) absolute waarde van verschil tussen
```

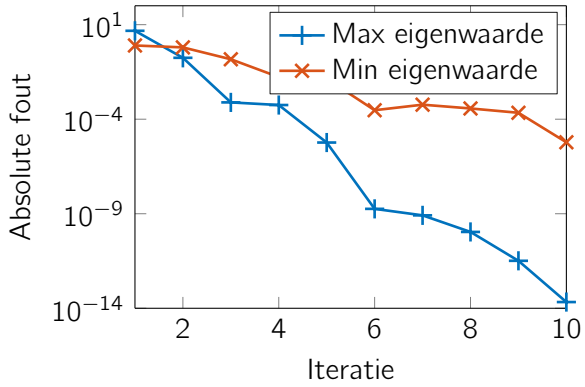
```
%         kleinste eigenwaarde en kleinste Ritz waarde

% Eigenwaarden van volle matrix
ewA = eig(full(A));
maxewA = max(ewA);
minewA = min(ewA);

% preallocation
n = size(A,1);
Q = zeros(n,N+1);
H = zeros(N+1,N);

Q(:,1) = b(:)/norm(b);
for n = 1:N
    v = A*Q(:,n);
    for j = 1:n
        H(j,n) = Q(:,j)'*v;
        v = v - H(j,n)*Q(:,j);
    end
    H(n+1,n) = norm(v);
    if H(n+1,n) <= 0, break; end
    Q(:,n+1) = v/H(n+1,n);

    % bereken Ritz waarden en fouten
    ewH = eig(full(H(1:n,1:n)));
    maxewH = max(ewH);
    minewH = min(ewH);
    maxd(n) = abs(maxewA - maxewH);
    mind(n) = abs(minewA - minewH);
end
```

```
m = 10;
L = eigint(4,5,m);
L(1) = 8;
%L(2) = 2;
[A,V] = willglv(L);
b = rand(m,1);

[H, Q, maxd, mind] = arnoldiew(A, b, m);

t = 1:m;
semilogy(t, maxd, '+-', t, mind,'x-');
xlabel('Iteratie');
ylabel('Absolute fout');
```

```
legend('Max eigenwaarde','Min eigenwaarde')
```



Figuur 1: Fouten voor eigenwaarden in $(4, 5)$.



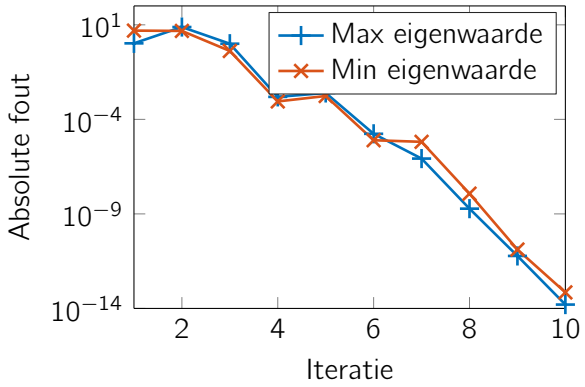
Figuur 2: Fouten voor eigenwaarden in $(4, 5)$ en 8.

De resultaten voor de drie gevallen worden getoond in Figuren 1, 2 en 3. Extreme eigenwaarden worden sneller goed benaderd.

Opgave 4

We gebruiken de volgende routine om de Ritz waarden te berekenen:

```
function [H, Q, rw] = arnoldiritz(A, b, N)
% function [H, Q, rw] = arnoldiritz(A, b, N)
%
% rw: Ritswaarden

rw = nan(N,N);
```

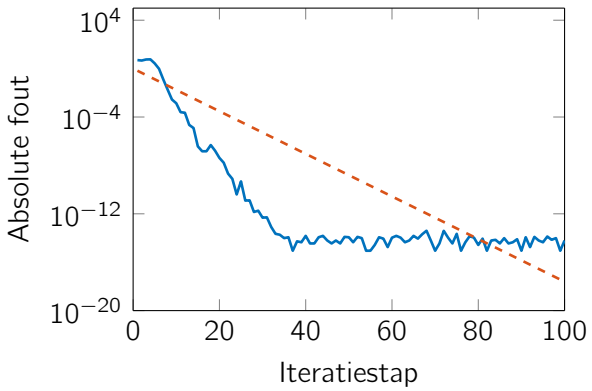


Figuur 3: Fouten voor eigenwaarden in $(4, 5)$, 8 en 2.

```
% preallocation
n = size(A,1);
Q = zeros(n,N+1);
H = zeros(N+1,N);

Q(:,1) = b(:)/norm(b);
for n = 1:N
    v = A*Q(:,n);
    for j = 1:n
        H(j,n) = Q(:,j)'*v;
        v = v - H(j,n)*Q(:,j);
    end
    H(n+1,n) = norm(v);
    if H(n+1,n) <= 0, break; end
    Q(:,n+1) = v/H(n+1,n);

    % Bereken Ritz waarden
    ewH = eig(full(H(1:n,1:n)));
    rw(1:n,n) = ewH;
end
```

```
N = 100;
A = sprand(1000,1000, 0.01);
b = randn(size(A,2),1);
eigs(A)
[H, Q, rw] = arnoldiritz(A,b,N);
rwmax = real(max(rw));
eigmax = real(max(eigs(A)));

clf;
```

```
semilogy(abs(rwmax-eigmax))
hold all
n = 1:N;
semilogy(n,(2/3).^n,'--');
xlabel('Iteratiestap');
ylabel('Absolute fout');
```



Figuur 4: Convergentie van Ritz waarden.

De convergentie van de Ritz waarden wordt getoond in Figuur 4. De waarden convergeren sneller dan de voorspelde convergentie met een factor $\left(\frac{2}{3}\right)^n$ met n de iteratiestap. De snellere convergentie is te verklaren door de geïsoleerde eigenwaarde (plot hiervoor de eigenwaarden van `A`.)