New Algorithms for Solving Nonlinear Eigenvalue Problems

Walter Gander

ETH and HKBU

The 5th International Conference on
Matrix Methods in Mathematics and applications

Moscow, Russia, August 19-23, 2019

Nonlinear EV-Problem

small

- Let $A : \lambda \mapsto \mathbb{C}^{n\times n}$ analytic on open set $\{\lambda\} \subset \mathbb{C}$

  Find λ such that $\det(A(\lambda)) = 0$

- Example from Tisseur and Meerbergen SIAM. Rev. 2001

$$A(\lambda) = \begin{pmatrix} \lambda+1 & 6\lambda^2-6\lambda & 0 \\ 2\lambda & 6\lambda^2-7\lambda+1 & 0 \\ 0 & 0 & \lambda^2+1 \end{pmatrix} = \lambda^2 M + \lambda C + K$$

- $M = \begin{pmatrix} 0 & 6 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ singular, $C = \begin{pmatrix} -1 & -6 & 0 \\ 2 & -7 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $K = I$

- $\det(A(\lambda)) = \det(M)\lambda^6 + \cdots$ has only degree 5
  $= -6\lambda^5 + 11\lambda^4 - 12\lambda^3 + 12\lambda^2 - 6\lambda + 1$

- $\Rightarrow$ quadr. EV-Problem has 5 finite + one infinite eigenvalue

Computing Determinants by Gaussian Elimination:

$A = LU \implies \det(A) = \det(L)\det(U) = u_{11}u_{22}\cdots u_{nn}$

```
function f=det1(A)
% DET1  compute determinant by LU-decomposition
n=length(A);
f=1;
for j=1:n
  [amax,kmax]=max(abs(A(j:n,j)));          % partial pivoting
  if amax==0                               % matrix is singular
    f=0; return
  end
  kmax=kmax+j-1;
  if kmax~=j                               % interchanging rows
    h=A(j,:); A(j,:)=A(kmax,:); A(kmax,:)=h;
    f=-f;
  end
  f=f*A(j,j);                              % mult diag elements
  A(j+1:n,j)=A(j+1:n,j)/A(j,j);            % elimination step
  A(j+1:n,j+1:n)=A(j+1:n,j+1:n)-A(j+1:n,j)*A(j,j+1:n);
end
```

Better Compute the Logarithm of Determinant

```
function logf=det2(A)
% DET2  compute logarithm of determinant by LU-decomposition
n=length(A);
logf=0;
for j=1:n
  [amax,kmax]=max(abs(A(j:n,j)));          % partial pivoting
  if amax==0                               % matrix is singular
    logf=-Inf; return
  end
  kmax=kmax+j-1;
  if kmax~=j                               % interchanging rows
    h=A(j,:); A(j,:)=A(kmax,:); A(kmax,:)=h;
    logf=logf+1i*pi;
  end
  logf=logf+log(A(j,j));                   % add log
  A(j+1:n,j)=A(j+1:n,j)/A(j,j);            % elimination step
  A(j+1:n,j+1:n)=A(j+1:n,j+1:n)-A(j+1:n,j)*A(j,j+1:n);
end
```

Applying Newton for $f(\lambda) = \det(A(\lambda)) = 0$

- For Newton $\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)}$ we need the derivative $f'(\lambda)$
- Explicit expression for $f'(\lambda)$: formula of Jacobi

$$f'(\lambda) = \det A(\lambda)\ \operatorname{trace}(A^{-1}(\lambda)A'(\lambda)) \implies \frac{f(\lambda)}{f'(\lambda)} = \frac{1}{\operatorname{trace}(A^{-1}(\lambda)A'(\lambda))}$$

- Note: derivative of $\log f(\lambda)$

$$\frac{d}{d\lambda}\log(f(\lambda)) = \frac{f'(\lambda)}{f(\lambda)}$$

  is the inverse Newton correction!

- Using algorithmic differentiation, we compute directly the ratio

$$\texttt{ffp} = \frac{f(\lambda_k)}{f'(\lambda_k)}$$

Algorithmic Differentiation

- Consider the polynomial
  $f(x) = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_{n-1} x + a_n$
- To evaluate y $= f(x)$, use Horner's scheme and compute
  yp $= f'(x)$ by algorithmic differentiation

```
function y=horner(a,x);
n=length(a);
y=0;
for i=1:n
  y=y*x+a(i);
end
```

```
function [y,yp]=horner(a,x);
n=length(a);
yp=0; y=0;
for i=1:n
  yp=yp*x+y;
  y=y*x+a(i);
end
```

- By differentiating every statement we get yp $= f'(x)$
- Same algorithm as computing the second row of Horner's scheme

Simplifications

- For Newton's iteration we need only the ratio $\frac{f}{f'}$
- Thus if we compute the derivative

$$\texttt{logfp} = \frac{d}{d\lambda}\log(f) = \frac{f'(\lambda)}{f(\lambda)}$$

  we get

$$\frac{f}{f'} = \frac{1}{\texttt{logfp}}$$

- No need to compute $\texttt{logf} = \log(\det(A(\lambda)))$
- Furthermore since we compute the ratio, no sign changes necessary anymore when interchanging rows.

Computing Newton's correction

```
function ffp=deta(A,Ap)
% DETA computes Newton correction ffp=f/f'
n=length(A); logfp=0;
for j=1:n
   [amax,kmax]= max(abs(A(j:n,j)));
   if amax == 0,ffp=0; return, end
   kmax=kmax+j-1;
   if kmax ~= j
      h=Ap(kmax,:); Ap(kmax,:)=Ap(j,:); Ap(j,:)=h;
      h=A(j,:); A(j,:)=A(kmax,:); A(kmax,:)=h;
   end
   logfp=logfp + Ap(j,j)/A(j,j);  % logf=logf+log(A(j,j));
   Ap(j+1:n,j)=(Ap(j+1:n,j)*A(j,j)-A(j+1:n,j)*Ap(j,j))/A(j,j)^2;
   A(j+1:n,j)=A(j+1:n,j)/A(j,j);
   Ap(j+1:n,j+1:n)=Ap(j+1:n,j+1:n) - Ap(j+1:n,j)*A(j,j+1:n)- ...
                     A(j+1:n,j)*Ap(j,j+1:n);
   A(j+1:n,j+1:n)=A(j+1:n,j+1:n) - A(j+1:n,j)*A(j,j+1:n);
end
ffp=1/logfp;
```

Avoid Recomputing Zeros

- Suppress already computed: $\lambda_1, \ldots, \lambda_k$

$$\text{Define} \quad f_k(\lambda) := \frac{f(\lambda)}{(\lambda - \lambda_1) \cdots (\lambda - \lambda_k)}$$

$$f_k'(\lambda) = \frac{f'(\lambda)}{(\lambda - \lambda_1) \cdots (\lambda - \lambda_k)} - \frac{f(\lambda)}{(\lambda - \lambda_1) \cdots (\lambda - \lambda_k)} \sum_{i=1}^{k} \frac{1}{\lambda - \lambda_i}$$

- Newton-Maehly iteration

$$\lambda_{new} = \lambda - \frac{f_k(\lambda)}{f_k'(\lambda)} = \lambda - \frac{f(\lambda)}{f'(\lambda)} \frac{1}{1 - \frac{f(\lambda)}{f'(\lambda)} \sum_{i=1}^{k} \frac{1}{\lambda - \lambda_i}}$$

- Need only Newton-correction $\frac{f(\lambda)}{f'(\lambda)}$ to suppress computed zeros

Quadratic Eigenvalue Problem

- Example from Tisseur and Meerbergen:
  Quadratic Eigenvalue Problem, SIAM. Rev. 2001.
- Eigenvalues of mass-spring system
- $\det Q(\lambda) = 0$ where $Q(\lambda) = \lambda^2 M + \lambda C + K$
  $M = I$
  $C = \tau\,\mathrm{tridiag}(-1, 3, -1)$
  $K = \kappa\,\mathrm{tridiag}(-1, 3, -1)$
- $\kappa = 5$, $\tau = 3$, nonoverdamped
- $n = 50$, 100 Eigenvalues

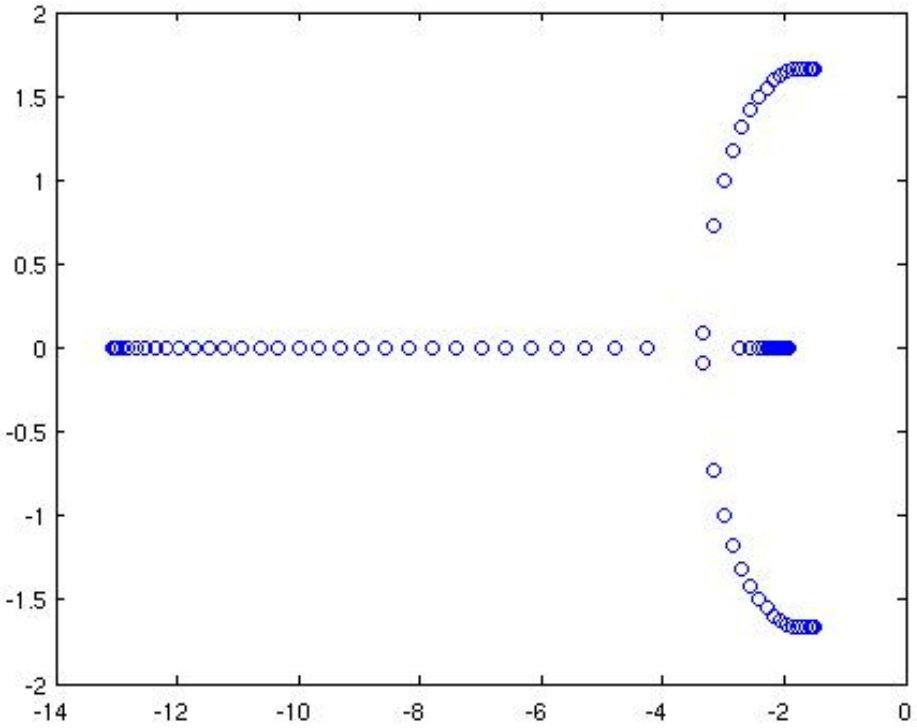
Program

TN1V.m

```
n=50, tau=3, kappa=5,  % nonoverdamped
e=-ones(n-1,1);
C=(diag(e,-1)+diag(e,1)+3*eye(n)); K=kappa*C; C=tau*C;
lam=-0.5+0.1*1i; lamb=[];           % start
for k=1:2*n
  ffp=1;
  while abs(ffp)>1e-14
    Qp=2*lam*eye(n)+C; Q=lam*(lam*eye(n)+ C)+K;
    ffp=deta(Q,Qp);
    s=sum(1./(lam-lamb(1:k-1)));
    lam=lam-ffp/(1-ffp*s);          % Newton step
  end
  lamb(k)=lam;
  lam=lam*(1+0.01*1i);              % start for next eigenvalue
end
plot(real(lamb),imag(lamb),'o')
```

Results

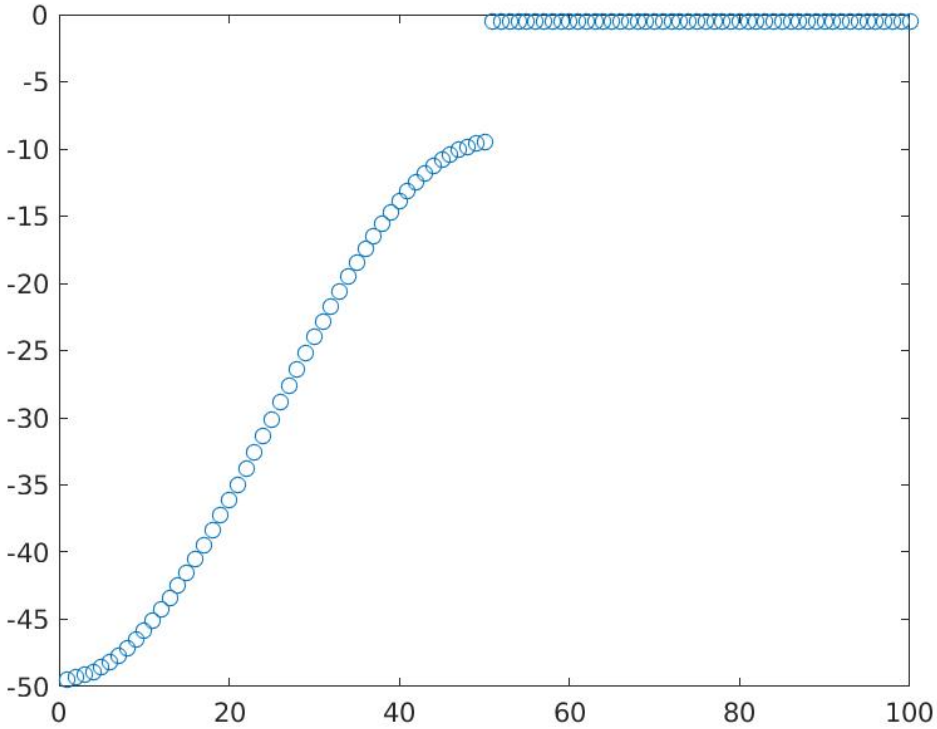
- tau=3, kappa=5
  nonoverdamped
- $n = 50 \implies 100$ eigenvalues (real and complex conjugate)
- made no use of tridiagonal structure of the matrix $Q(\lambda)$



Second Example

TN2V.m

- tau=10, kappa=5
  overdamped
- $n = 50$
  $\Longrightarrow$ 100 real eigenvalues
- made no use of tridiagonal structure of the matrix $Q(\lambda)$



Cubic Eigenvalue Problem (Arbenz/Gander, 1986)

AGNV.m

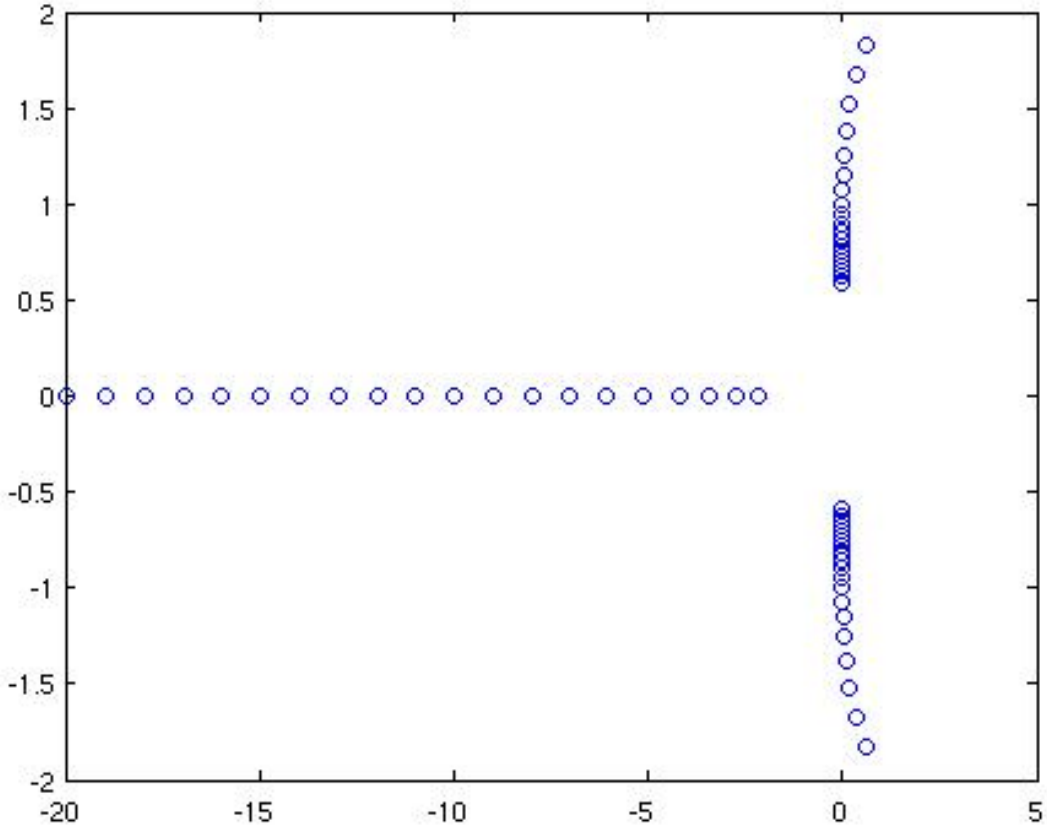
$$Q(\lambda) = \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0$$

$$A_0 = \begin{pmatrix} 8 & 1 & & \\ 1 & 8 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 8 \end{pmatrix}$$

$$A_2 = \mathrm{diag}(1, 2, \ldots, n)$$

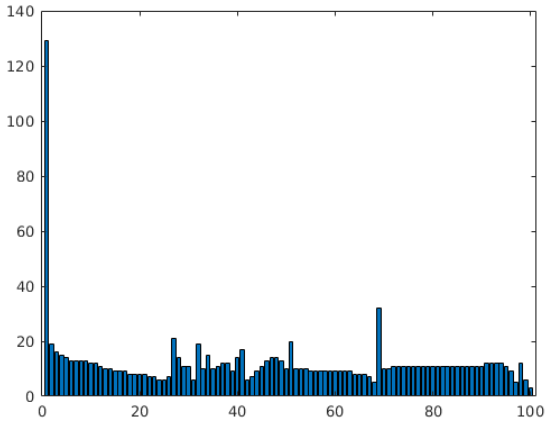
$$A_1 = A_3 = I$$

$n = 20 \Rightarrow 60$ Eigenvalues



Improving Convergence

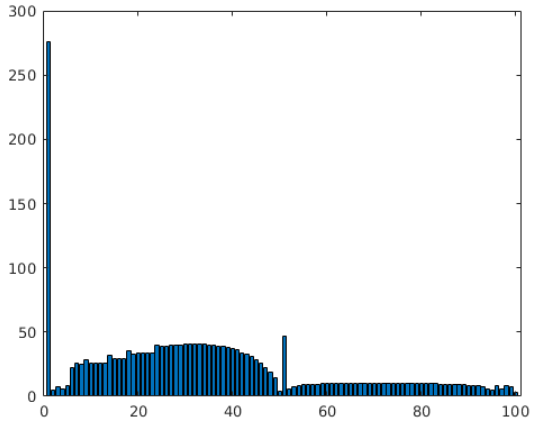
- Newton converges locally quadratically to simple zeros
- Global convergence may be slow
- # iterations and mean for one EV for the 3 examples



nonoverdamped

mean = 12.0

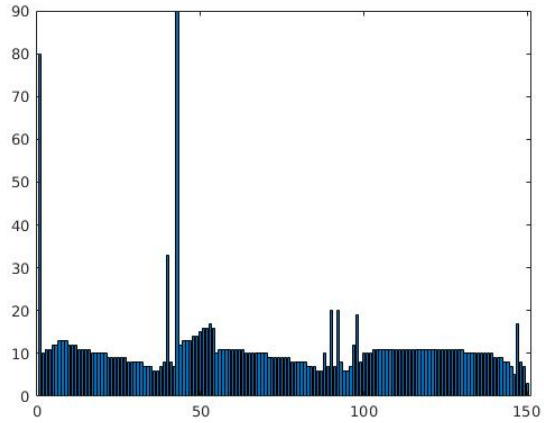
$n = 50$



overdamped

mean = 22.3

$n = 50$



cubic EVP

mean= 11.3

$n = 50$

Newton and Halley

- Newton's iteration replaces f at x_k by a linear function g such that $f(x_k) = g(x_k)$, $f'(x_k) = g'(x_k)$:

$$g(x) = f(x_k) + f'(x_k)(x - x_k)$$

and x_{k+1} is the zero of g

- Halley's iteration (cubic convergence)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \frac{1}{1 - \frac{1}{2} \frac{f(x_k) f''(x_k)}{f'(x_k)^2}}.$$

replaces f locally by a hyperbolic function

$$g(x) = \frac{a}{x + b} + c$$

such that $f(x_k) = g(x_k)$, $f'(x_k) = g'(x_k)$ and $f''(x_k) = g''(x_k)$ and x_{k+1} is the zero of g

Implementing Halley's Iteration

- We need the second derivative of the determinant
- More precisely, we need $t(x) = \dfrac{f(x)f''(x)}{f'^2(x)}$
- The derivative of Newton's correction is

$$\frac{d}{dx}\left(\frac{f}{f'}\right) = \frac{f'^2 - ff''}{f'^2} = 1 - \frac{ff''}{f'^2}$$

- Thus

$$\Longrightarrow \quad \frac{ff''}{f'^2} = 1 - \frac{d}{dx}\left(\frac{f}{f'}\right)$$

- $\Longrightarrow$ we only need the derivative `dffp` of the variable `ffp` in our function `deta` to get

$$t = 1 - \text{ dffp}$$

Second Derivative by Algorithmic Differentiation

```
function [ffp,dffp] = det2p(A,Ap,App)
% DET2P computes Newton correction ffp = f/f' and its derivative 1-ff''/f'^2
n=length(A);
logfpp=0;                                % logfpp = derivative of log(f)'
logfp=0;                                 % log(f)'
for k=1:n
  [amax,kmax]=max(abs(A(k:n,k)));        % partial pivoting
  if amax==0                             % matrix singular
    ffp=0; dffp=0;return
  end
  kmax=kmax+k-1;
  if kmax~=k                             % interchange rows
    h=App(k,:); App(k,:)=App(kmax,:); App(kmax,:)=h;
    h=Ap(k,:); Ap(k,:)=Ap(kmax,:); Ap(kmax,:)=h;
    h=A(k,:); A(k,:)=A(kmax,:); A(kmax,:)=h;
  end
  logfpp=logfpp+(A(k,k)*App(k,k)-Ap(k,k)^2)/A(k,k)^2;
  logfp=logfp+Ap(k,k)/A(k,k);
```

```
  App(k+1:n,k)=(A(k,k)*App(k+1:n,k)-Ap(k+1:n,k)*Ap(k,k))/A(k,k)^2-...
       (Ap(k+1:n,k)*Ap(k,k)/A(k,k)^2+ ...
       A(k+1:n,k)*App(k,k)/A(k,k)^2-...
       2* A(k+1:n,k)*Ap(k,k)^2/A(k,k)^3);

  Ap(k+1:n,k)=Ap(k+1:n,k)/A(k,k)-A(k+1:n,k)*Ap(k,k)/A(k,k)^2;
  A(k+1:n,k)=A(k+1:n,k)/A(k,k);          % elimination step

  App(k+1:n,k+1:n)=App(k+1:n,k+1:n) -...
      (App(k+1:n,k)*A(k,k+1:n) + Ap(k+1:n,k)*Ap(k,k+1:n) )  - ...
      (Ap(k+1:n,k)*Ap(k,k+1:n) + A(k+1:n,k)*App(k,k+1:n));

  Ap(k+1:n,k+1:n)=Ap(k+1:n,k+1:n) - Ap(k+1:n,k)*A(k,k+1:n)-...
      A(k+1:n,k)*Ap(k,k+1:n);
  A(k+1:n,k+1:n)=A(k+1:n,k+1:n) - A(k+1:n,k)*A(k,k+1:n);
end
dffp=-logfpp/logfp^2;
ffp=1/logfp;
```

Suppressing Already Computed Zeros

$\pi = (\lambda - \lambda_1) \cdots (\lambda - \lambda_k)$, $\pi' = \pi\sigma$, $\sigma = \sum_{j=1}^{k} \frac{1}{\lambda - \lambda_j}$, $\sigma' = -\sum_{j=1}^{k} \frac{1}{(\lambda - \lambda_j)^2}$

- $f_k(\lambda) = \dfrac{f(\lambda)}{\pi}$, $f_k' = \dfrac{f' - \sigma f}{\pi}$, $\qquad \dfrac{f_k}{f_k'} = \dfrac{f}{f'} \dfrac{1}{1 - \frac{f}{f'}\sigma}$

- $f_k'' = \dfrac{f'' - \sigma' f - \sigma f' - \sigma f' + \sigma^2 f}{\pi}$

$$\Longrightarrow \frac{f_k''}{f_k'} = \frac{f'' - \sigma' f - 2\sigma f' + \sigma^2 f}{f' - \sigma f} = \frac{\frac{f''}{f'} - \sigma' \frac{f}{f'} - 2\sigma + \sigma^2 \frac{f}{f'}}{1 - \sigma \frac{f}{f'}}$$

- Multiply and rearrange

$$\frac{f_k f_k''}{{f_k'}^2} = \frac{\frac{f f''}{f'^2} + (\sigma^2 - \sigma') \left(\frac{f}{f'}\right)^2 - 2\sigma \frac{f}{f'}}{\left(1 - \sigma \frac{f}{f'}\right)^2}$$

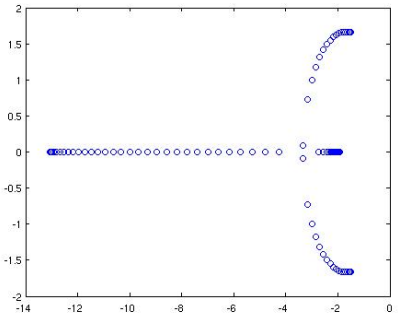
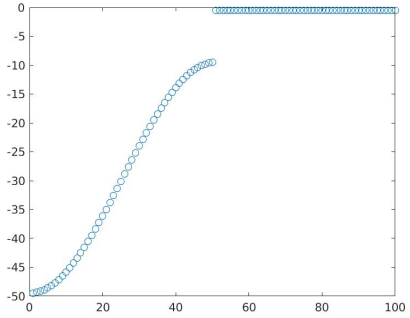
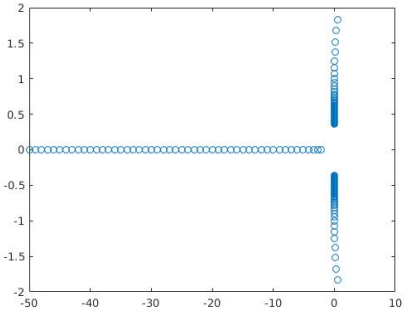
"Halley-Maehly" Iteration

1. Compute $\dfrac{f_k}{f_k'} = \dfrac{f}{f'} \dfrac{1}{1 - \dfrac{f}{f'}\sigma}$

2. and also $\dfrac{f_k f_k''}{{f_k'}^2} = \dfrac{\dfrac{f f''}{{f'}^2} + (\sigma^2 - \sigma') \left(\dfrac{f}{f'}\right)^2 - 2\sigma \dfrac{f}{f'}}{\left(1 - \sigma \dfrac{f}{f'}\right)^2}$

3. Iterate $\lambda_{\text{new}} = \lambda - \dfrac{f_k}{f_k'} \dfrac{1}{1 - \dfrac{1}{2} \dfrac{f_k f_k''}{{f_k'}^2}}$

Comparison: Newton $\leftrightarrow$ Halley

n	50	50	50
# Eigvals	100	100	150
Newton			
max It	128	275	90
mean It	11.4	20.9	11.3
Halley			
max It	67	140	46
mean It	7	12.1	7.1

Laguerre's Method for Zeros of Polynomials

- f a polynom of degree n is approximated by $g(\lambda) = a(\lambda - x_1)(\lambda - x_2)^{n-1}$
- determine a, x_1, x_2 by $f(\lambda_k) = g(\lambda_k)$, $f'(\lambda_k) = g'(\lambda_k)$, $f''(\lambda_k) = g''(\lambda_k)$
- solve $g(\lambda) = 0$ to get λ_{k+1} (Laguerre's Method)

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)} \frac{n}{1 + \sqrt{(n-1)^2 - n(n-1)\frac{f(\lambda_k)f''(\lambda_k)}{f'(\lambda_k)^2}}}$$

- for $n \to \infty$ we get Ostrowski's method:

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)} \frac{1}{\sqrt{1 - \frac{f(\lambda_k)f''(\lambda_k)}{f'(\lambda_k)^2}}}$$

- As Halley, both methods need only $\frac{f(\lambda_k)}{f'(\lambda_k)}$ and $\frac{f(\lambda_k)f''(\lambda_k)}{f'(\lambda_k)^2}$

Third Order Iteration Methods

- THEOREM (WG 1985). *If s simple zero of f, G any function with $G(0) = 1$, $G'(0) = \frac{1}{2}$ and $|G''(0)| < \infty$, then*

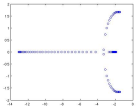
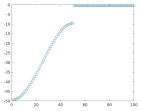
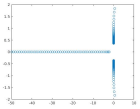
$$x_{\text{new}} = x - \frac{f(x)}{f'(x)}\, G(t(x)), \quad t(x) = \frac{f(x)f''(x)}{f'(x)^2}$$

  converges at least cubically to s.

- Halley's formula: $G(t) = \frac{1}{1-\frac{1}{2}t} = 1 + \frac{1}{2}t + \frac{1}{4}t^2 + \frac{1}{8}t^3 + \ldots$
- Euler's formula: $G(t) = \frac{2}{1+\sqrt{1-2t}} = 1 + \frac{1}{2}t + \frac{1}{2}t^2 + \frac{5}{8}t^3 + \ldots$
- Quadratic inverse interpolation: $G(t) = 1 + \frac{1}{2}t$
- Ostrowski's square root iteration: $G(t) = \frac{1}{\sqrt{1-t}} = 1 + \frac{1}{2}t + \frac{3}{8}t^2 + \ldots$
- Laguerre: $G(t) = \frac{n}{1+\sqrt{(n-1)^2 - n(n-1)t}} = 1 + \frac{1}{2}t + \frac{1}{8}\,\frac{3\,n-2}{n-1}t^2 + \ldots$
- Hansen-Patrick family: $G(t) = \frac{\alpha+1}{\alpha+\sqrt{1-(\alpha+1)t}} = 1 + \frac{1}{2}t + \frac{\alpha+3}{8}t^2 + \ldots$

Halley $\leftrightarrow$ Laguerre, Ostrowski

AllMethods1,2,3

n	50	50	50
# of eigenvalues	100	100	150
Halley			
max iterations	67	140	46
mean iterations	7	12.1	7.1
Laguerre			
max iterations	18	36	16
mean iterations	5.3	6.6	5.2
Ostrowski			
max iterations	23	43	18
mean iterations	5.5	7.1	5.2

The determinant for a quadratic or cubic EVP is a polynom.

$\Longrightarrow$ Laguerre's iteration is well suited

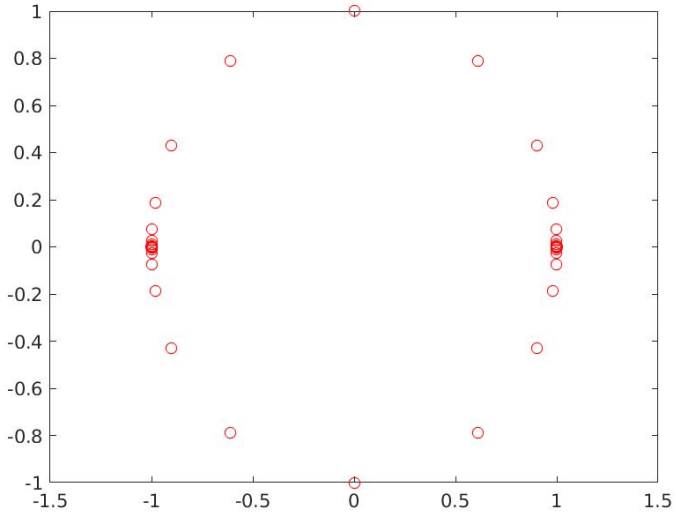
Resources for Nonlinear EV Problems

- Françoise Tisseur's talk at ICIAM 2011:
  A Review of Nonlinear Eigenvalue Problems.
  `http://www.ma.man.ac.uk/~ftisseur/talks`
- T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, NLEVP: A Collection of Nonlinear Eigenvalue Problems, MIMS EPrint 2011.116, December 2011.
- T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, NLEVP: A Collection of Nonlinear Eigenvalue Problems. Users' Guide, MIMS EPrint 2010.117, December 2011.
- NLEVP: A Collection of Nonlinear Eigenvalue Problems
  `https://tinyurl.com/y6pywbv7`

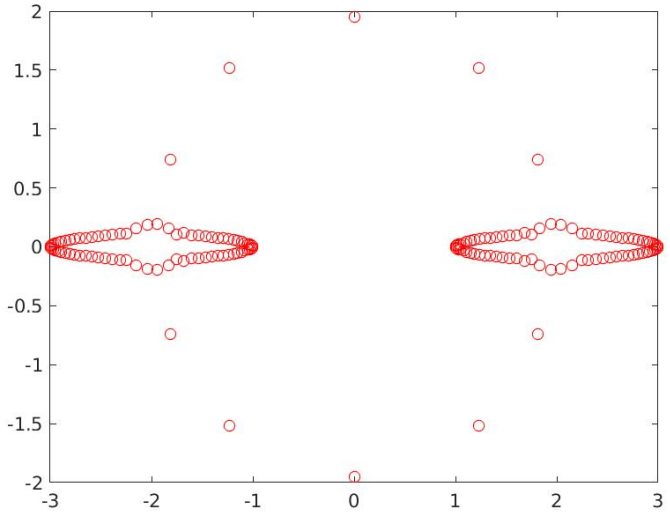
Dense Matrices from NLEVP

- $n \leq 30$
  - gen-tantipal2(16)
  - hadeler(30)
  - metal-strip
  - omnicam
- $n > 30$
  - gen-tpal2(64)
  - sign1(81), sign2(81)
  - wiresaw1(200)
  - wiresaw2(100)

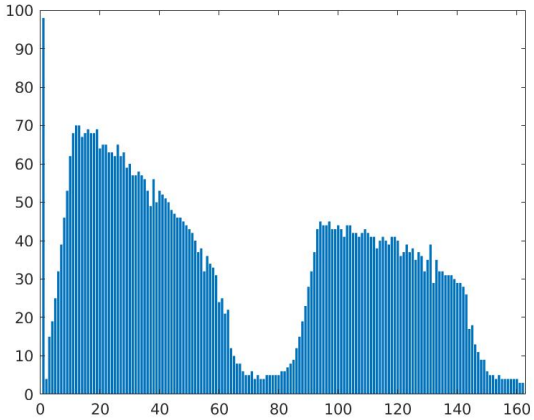
`sign`: dense $n = 81$



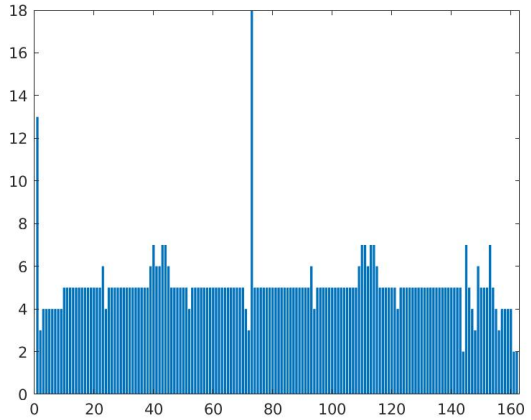
solved with Laguerre



`SIGN1` 63.92 seconds

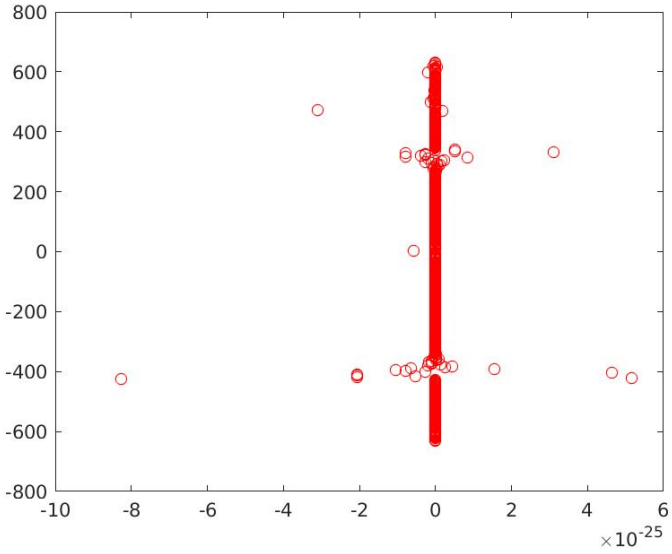


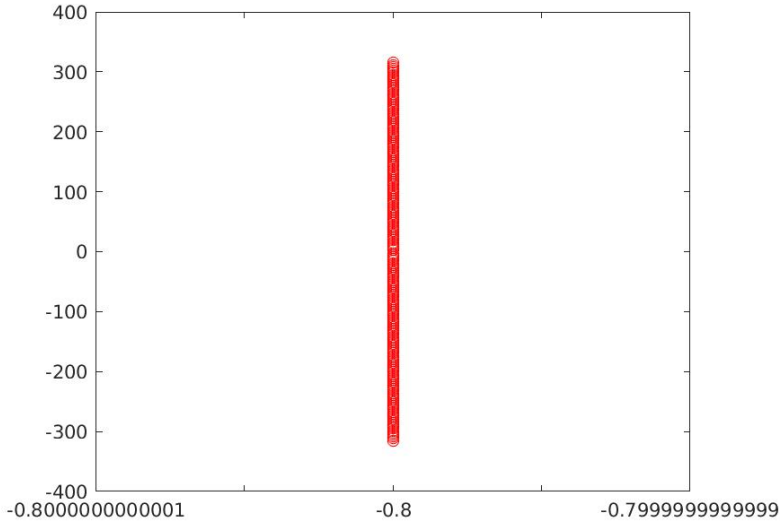
`SIGN2` 10.82 seconds



`wiresaw`: dense

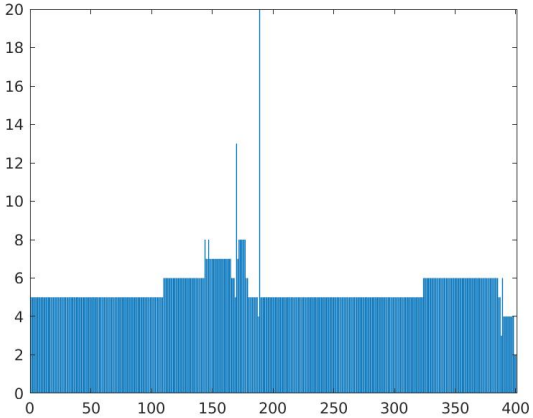
solved with Laguerre

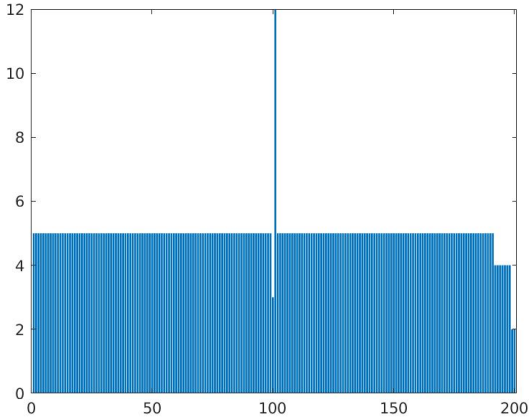




$n = 200$, `WIRESAW1` 191.02 seconds

$n = 100$, `WIRESAW2` 15.10 seconds





Non-polynomial Problem of Xiaoping Cheng

$$A(\lambda) = \lambda I - \begin{pmatrix} -5 & 1 \\ 2 & -6 \end{pmatrix} - \begin{pmatrix} -2 & 1 \\ 4 & -1 \end{pmatrix} e^{-\tau\lambda}, \qquad \tau = 1$$

solve with Ostrowski iteration

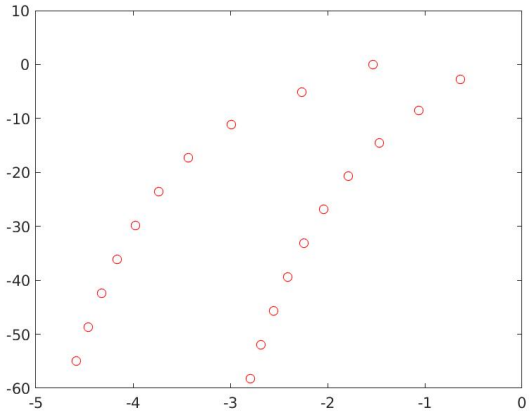
chen.m

$N = 20$

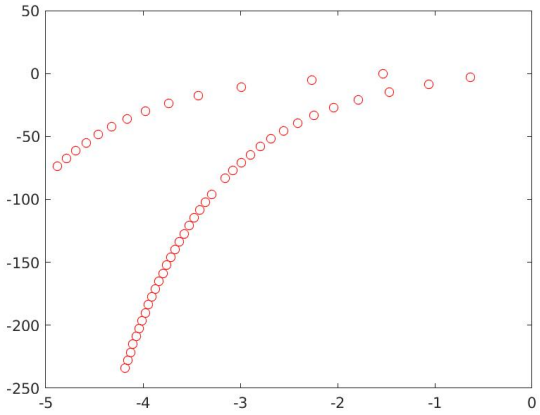
```
-1.5359 + 0.0000i
-0.6355 - 2.7175i
-2.2674 - 5.0693i
-1.0580 - 8.4500i
-2.9902 -11.1010i
-1.4690 -14.4908i
-3.4301 -17.3212i
-1.7878 -20.6596i
-3.7377 -23.5801i
-2.0374 -26.8800i
-3.9734 -29.8516i
-2.2400 -33.1245i
-4.1642 -36.1284i
-2.4096 -39.3817i
-4.3244 -42.4078i
-2.5553 -45.6465i
-4.4625 -48.6886i
-2.6828 -51.9161i
-4.5839 -54.9703i
-2.7960 -58.1887i
```

can choose # of EV:

$N = 20$



$N = 50$



only one real eigenvalue $\lambda_1 = -1.535876071474386$

Non-polynomial Problem from Tisseur's NLEP

$A(\lambda) = -\lambda I + A_0 + A_1 e^{-\lambda} \in \mathbb{R}^{3\times 3}$ "*. . . characteristic equation of a time-delay system with a single delay and constant coefficents. The problem has a double non-semisimple eigenvalue* $\lambda = 3\pi i$"

solve with Ostrowski iteration

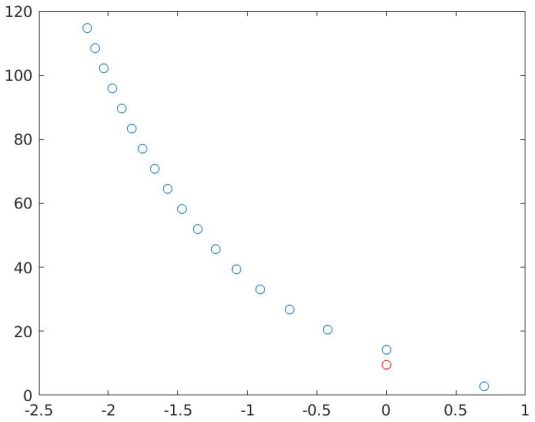
TimeDelay.m

$N = 20$

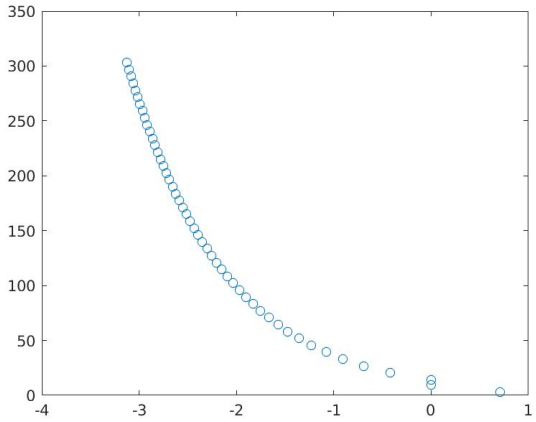
```
1.0e+02 *
 0.0071 + 0.0274i
-0.0000 + 0.0942i
-0.0000 + 0.0942i
 0.0000 + 0.1414i
-0.0042 + 0.2049i
-0.0069 + 0.2676i
-0.0090 + 0.3303i
-0.0108 + 0.3930i
-0.0123 + 0.4558i
-0.0136 + 0.5186i
-0.0147 + 0.5814i
-0.0157 + 0.6442i
-0.0167 + 0.7070i
-0.0175 + 0.7698i
-0.0183 + 0.8326i
-0.0190 + 0.8954i
-0.0197 + 0.9582i
-0.0203 + 1.0211i
-0.0209 + 1.0839i
-0.0215 + 1.1467i
```

can choose # of EV:

$N = 20$



$N = 50$

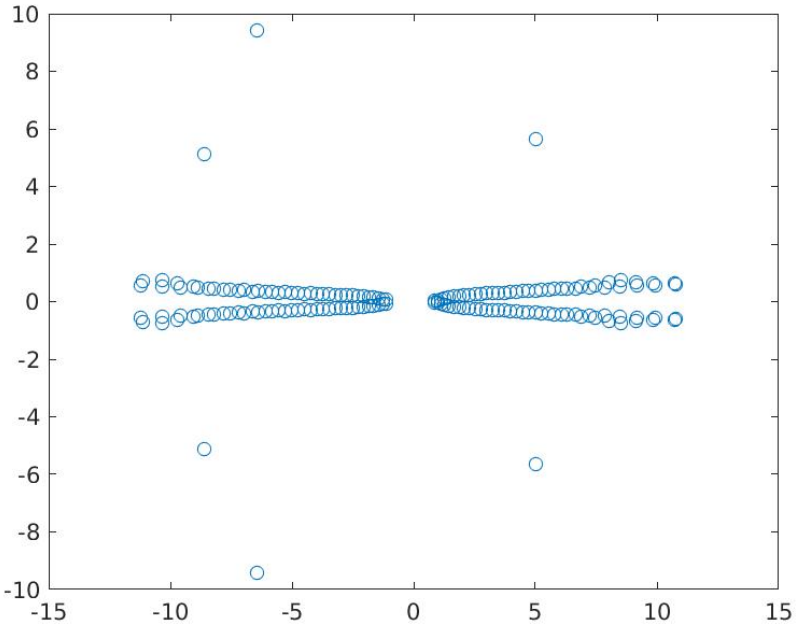


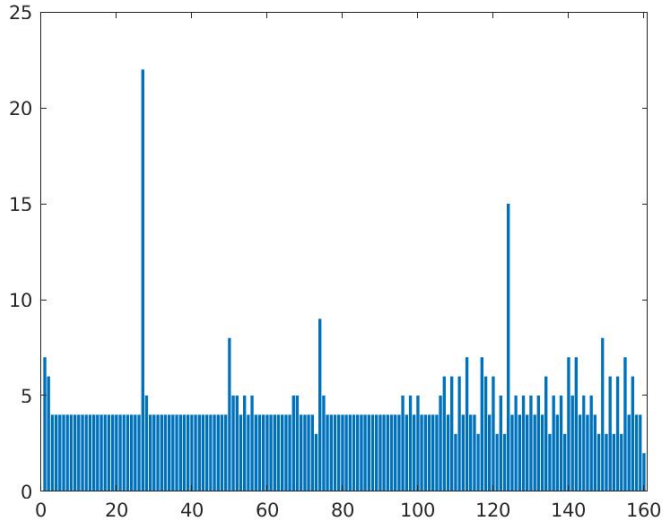
$$\lambda_2 = -0.000000005149180 + 9.424777943433675i$$
$$\lambda_3 = -0.000000007679198 + 9.424777969836999i$$
$$\lambda_4 = 0.000000000000001 + 14.137166941154069i$$

Dirac Example from NLEP

The spectrum of this matrix polynomial is the second order spectrum of the Dirac operator with an electric Coulombic potential, relative to the subspace generated by the Hermite functions of odd order.

dirac, QEV-problem, dense matrix, $n = 80$, $2n = 160$ eigenvalues





Solved with Ostrowski (8 sec.)

dirac.m

Many Matrices in NLEP have Band Structure

Examples:

- QEP Eigenvalues of mass-spring system: tridiagonal
- Cubic EVP Gander-Arbenz: tridiagonal
- Damped Beam: 7 diagonals, $p = q = 3$
- pdde-stability: 33 diagonals $p = q = 16$, n=225
- shaft: 7 diagonals, $p = q = 3$, $n = 400$
- acoustic-wave-2d: 13 diagonals $q = p = 6$ $n = 30$

Gaussian Elimination for Banded Matrices

- $A \in \mathbb{R}^{n\times n}$, q lower and p upper diagonals, stored as $B \in \mathbb{R}^{n\times(p+1+q)}$
- Append q zeros columns to B for partial pivoting

$$A = \begin{bmatrix} x & x & 0 & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & 0 & 0 \\ 0 & x & x & x & x & 0 & 0 & 0 \\ 0 & 0 & x & x & x & x & 0 & 0 \\ 0 & 0 & 0 & x & x & x & x & 0 \\ 0 & 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & 0 & 0 & x & x & x \end{bmatrix} \quad B = \left[\begin{array}{cccc|cc} 0 & 0 & x & x & 0 & 0 \\ 0 & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & x & 0 & 0 \\ x & x & x & 0 & 0 & 0 \end{array}\right]$$

- Use `StoreBandMatrix.m` and `EliminationBandMatrix.m` from our book W. Gander, M. J. Gander, F.Kwok, *Scientific Computing*, Springer 2014

Newton-Correction for Banded Matrices

```
function ffp=detaband(p,q,B,Bp);
% DETABAND computes Newton-correction for a banded matrix
n=length(B); logfp=0;
Bp=[Bp,zeros(n,q)]; B=[B,zeros(n,q)];   % augment B with q columns
normb=norm(B,1);
for j=1:n
  maximum=0;                            % search pivot
  for k=j:min(j+q,n)
    if abs(B(k,j-k+q+1))>maximum,
      kmax=k; maximum=abs(B(k,j-k+q+1));
    end
  end
  if maximum<1e-14*normb;               % only small pivots
    ffp=0; return                       % consider det=0
  end
  if j~=kmax                            % interchange rows
    index1=j-kmax+q+1:min(n,j+2*q+p-kmax+1);
    index2=q+1:min(n,2*q+p+1);
    h=Bp(kmax,index1); Bp(kmax,index1)=Bp(j,index2); Bp(j,index2)=h;
    h=B(kmax,index1); B(kmax,index1)=B(j,index2); B(j,index2)=h;
  end
```

```
  logfp=logfp+Bp(j,q+1)/B(j,q+1);
  for k=j+1:min(n,j+q)                    % elimination step
    Bp(k,j-k+q+1)=(B(j,q+1)*Bp(k,j-k+q+1)-B(k,j-k+q+1)*Bp(j,q+1))/B(j,q+1)^2;;
    B(k,j-k+q+1)=B(k,j-k+q+1)/B(j,q+1);
  end
  for k=j+1:min(n,j+q)
    for l=j+1:min(n,j+p+q)
      Bp(k,l-k+q+1)=Bp(k,l-k+q+1)-Bp(k,j-k+q+1)*B(j,l-j+q+1)-...
      B(k,j-k+q+1)*Bp(j,l-j+q+1);
      B(k,l-k+q+1)=B(k,l-k+q+1)-B(k,j-k+q+1)*B(j,l-j+q+1);
    end
  end
end
ffp=1/logfp;
```

Newton-Correction and Derivative for Banded Matrices

```
function [ffp,dffp]=det2pband(p,q,B,Bp,Bpp);
% DETABAND computes Newton-correction and derivative for a banded matrix
n=length(B);logfpp=0; logfp=0;
Bpp=[Bpp,zeros(n,q)];
Bp=[Bp,zeros(n,q)]; B=[B,zeros(n,q)];   % augment B with q columns
normb=norm(B,1);
for j=1:n
  maximum=0; kmax=j;                    % search pivot
  for k=j:min(j+q,n)
    if abs(B(k,j-k+q+1))>maximum,
      kmax=k; maximum=abs(B(k,j-k+q+1));
    end
  end
  if maximum<1e-14*normb;               % only small pivots
    ffp=0; dffp=0; return               % consider det=0
  end
  if j~=kmax                            % interchange rows
    ind1=j-kmax+q+1:min(n,j+2*q+p-kmax+1);
    ind2=q+1:min(n,2*q+p+1);
    h=Bpp(kmax,ind1); Bpp(kmax,ind1)=Bpp(j,ind2); Bpp(j,ind2)=h;
    h=Bp(kmax,ind1); Bp(kmax,ind1)=Bp(j,ind2); Bp(j,ind2)=h;
    h=B(kmax,ind1); B(kmax,ind1)=B(j,ind2); B(j,ind2)=h;
```

```
  end
  logfpp=logfpp+(B(j,q+1)*Bpp(j,q+1)-Bp(j,q+1)^2)/B(j,q+1)^2;
  logfp=logfp+Bp(j,q+1)/B(j,q+1);
  for k=j+1:min(n,j+q)                    % elimination step
    ind3=j-k+q+1;
    Bpp(k,ind3)=(Bpp(k,ind3)*B(j,q+1)-Bp(j,q+1)*Bp(k,ind3))/B(j,q+1)^2 ...
                 -(Bp(k,ind3)*Bp(j,q+1)+B(k,ind3)*Bpp(j,q+1))/B(j,q+1)^2 ...
               +2*Bp(j,q+1)^2*B(k,ind3)/B(j,q+1)^3;
    Bp(k,ind3)=(B(j,q+1)*Bp(k,ind3)-B(k,ind3)*Bp(j,q+1))/B(j,q+1)^2;;
    B(k,ind3)=B(k,ind3)/B(j,q+1);
  end
  for k=j+1:min(n,j+q)
    for l=j+1:min(n,j+p+q)
      ind4=l-k+q+1; ind5=j-k+q+1; ind6=l-j+q+1;
      Bpp(k,ind4)=Bpp(k,ind4)-Bpp(k,ind5)*B(j,ind6)-Bp(k,ind5)*Bp(j,ind6)...
                  -Bp(k,ind5)*Bp(j,ind6)-B(k,ind5)*Bpp(j,ind6);
      Bp(k,ind4)=Bp(k,ind4)-Bp(k,ind5)*B(j,ind6)-B(k,ind5)*Bp(j,ind6);
      B(k,ind4)=B(k,ind4)-B(k,ind5)*B(j,ind6);
    end
  end
end
dffp=-logfpp/logfp^2; ffp=1/logfp;
```

Comparison Full $\leftrightarrow$ Band Algorithm

- `Beamsensitivity`: 7 diagonals, $n = 200$, 400 eigenvalues, using Laguerre, measured with MATLAB's `tic,toc`,
  - Full-matrix algorithm: 83.85 seconds
  - Band-matrix algorithm: 10.39 seconds
- `Nonoverdamped`: tridiagonal, $n = 500$, 1000 eigenvalues
  - too large for full matrix algorithm
  - Band with Newton: 151.97 seconds
  - Band with Halley: 52.71 seconds

Damped Beam Example

- Quote from N. Higham et al., 2008:

  The standard approach to the numerical solution of the QEP is to convert the quadratic $Q(\lambda) = \lambda^2 M + \lambda D + K$ into a linear polynomial $L(\lambda) = \lambda X + Y$ of twice the dimension of Q but with the same spectrum. The resulting generalized eigenproblem $L(\lambda)z = 0$ is usually solved by the QZ algorithm for small- to medium-size problems or by a Krylov method for large sparse problems.

  A common choice of L in practice is the first companion form, given by

$$C_1(\lambda) = \lambda \begin{pmatrix} M & 0 \\ 0 & I \end{pmatrix} + \begin{pmatrix} D & K \\ -I & 0 \end{pmatrix}$$

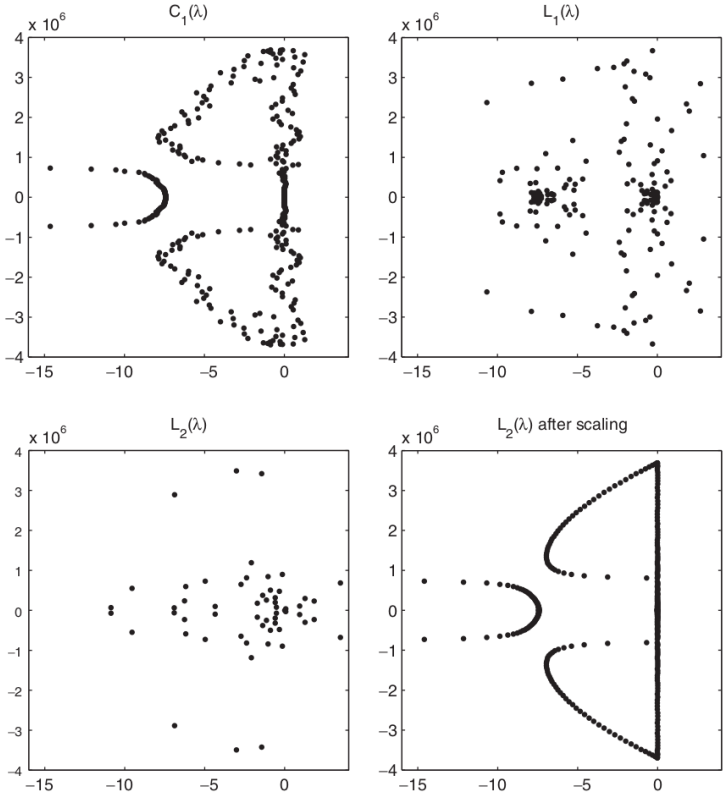
  When K and M, respectively, are nonsingular the two pencils

$$L_1(\lambda) = \lambda \begin{pmatrix} M & 0 \\ 0 & -K \end{pmatrix} + \begin{pmatrix} D & K \\ K & 0 \end{pmatrix}, \quad L_2(\lambda) = \lambda \begin{pmatrix} 0 & M \\ M & D \end{pmatrix} + \begin{pmatrix} -M & 0 \\ 0 & K \end{pmatrix}$$

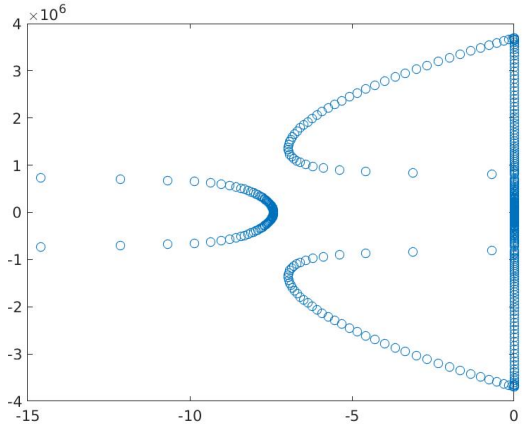
  are other possible linearizations

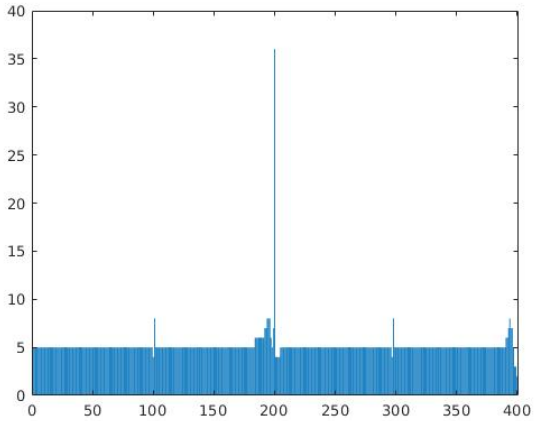
- Using these transformations, Higham et al. noticed poor results without scalings

Damped Beam (cont.)
Results without/with Scaling



Higham et al. (2008)

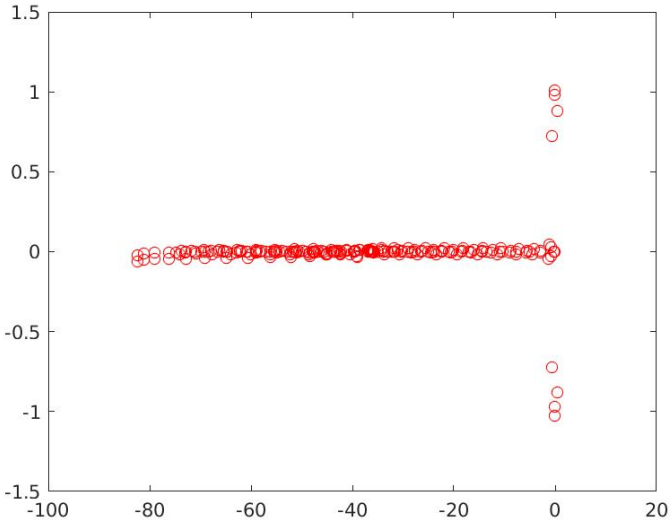




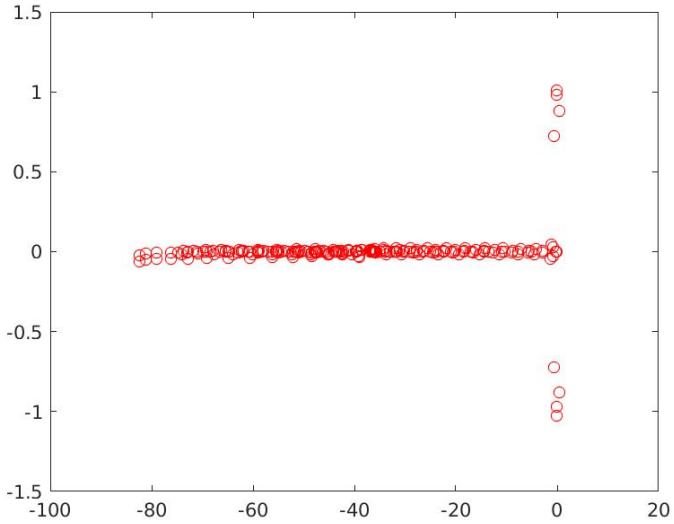
banded 7 diagonals, Laguerre's iteration, no need to scale.
$n = 200$, 400 EV

`BeamSensitivityBand`

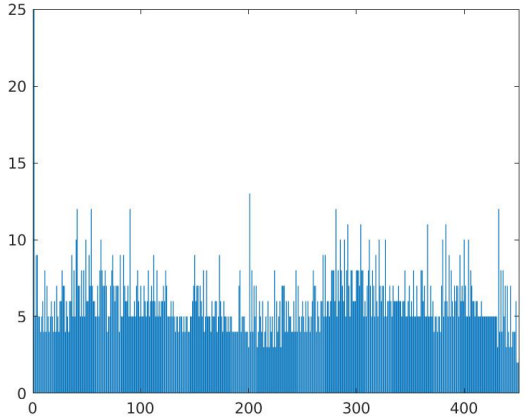
`pdde-stability`, $n = 225$



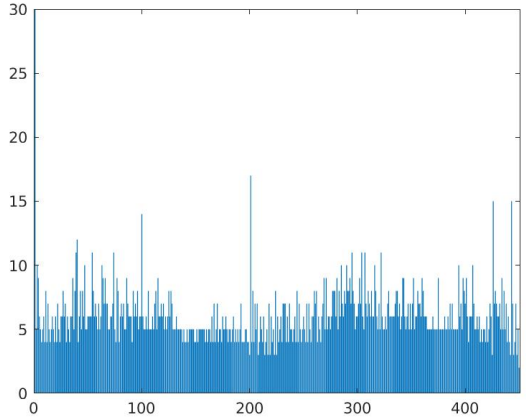
banded $q = p = 16$



Laguerre 497.49 sec



Ostrowski 354.35 sec



Summary

- By computing determinants with Gaussian Elimination we can solve accurately nolinear EVP
- To solve $f(\lambda) = \det A(\lambda) = 0$ using Newton or a third order method we only need the quotients

$$\frac{f(\lambda)}{f'(\lambda)} \quad \text{and} \quad t(\lambda) = \frac{f(\lambda)f''(\lambda)}{f'(\lambda)^2}$$

- They are obtained by algorithmic differentiation of Gaussian elimination
- We extended suppression of computed eigenvalues to third order methods
- Many problems from Tisser's NLEP collection have band-structure. For the solution we adapted Gaussian elimination with partial pivoting for band matrices.

References

- P. Arbenz and W. Gander, Solving nonlinear Eigenvalue Problems by Algorithmic Differentiation, Computing 36, 205-215, 1986.
- W. Gander, On Halley's Iteration Method, The American Mathematical Monthly, Vol. 92, No. 2, February 1985.
- W. Gander, Zeros of Determinants of λ-Matrices, In: Matrix Methods: Theory, Algorithms and Applications, Dedicated to the Memory of Gene Golub. Vadim Olshevsky & Eugene Tyrtyshnikov eds. World Scientific Publishers, 2010
- H. J. Maehly, Zur iterativen Auflösung algebraischer Gleichungen, ZAMP (Zeitschrift für angewandte Mathematik und Physik), (1954), pp. 260–263.
- F. Tisseur and K. Meerbergen, The Quadratic Eigenvalue Problem, SIAM. Rev., 43, pp. 234–286, 2001.
- Güttel, Stefan and Tisseur, Françoise, The Nonlinear Eigenvalue Problem, 2017 MIMS EPrint: 2017.7
- Nicholas J. Higham, D. Steven Mackey, Françoise Tisseur, and Seamus D. Garvey, Scaling, sensitivity and stability in the numerical solution of quadratic eigenvalue problems, Int. J. Numer. Meth. Engng 2008; 73:344–360