Solving large scale eigenvalue problems

Lecture 3, March 7, 2018: Newton methods

http://people.inf.ethz.ch/arbenz/ewp/

Peter Arbenz
Computer Science Department, ETH Zürich
E-mail: arbenz@inf.ethz.ch

Survey of today's lecture

- Linear and nonlinear eigenvalue problems
- Eigenvalues as zeros of the determinant function
- Hyman's method for Hessenberg matrices
- Algorithmic differentiation
- Newton iterations
- Successive linear approximations

Linear and nonlinear eigenvalue problems

- Linear eigenvalue problems

  Find values $\lambda \in \mathbb{C}$ such that $A - \lambda I$ is singular.
  Or equivalently:
  Find values $\lambda \in \mathbb{C}$ such that there is a nonzero (nontrivial) $\boldsymbol{x}$ such that

$$(A - \lambda I)\boldsymbol{x} = \mathbf{0} \quad \Longleftrightarrow \quad A\boldsymbol{x} = \lambda\boldsymbol{x}.$$

Linear and nonlinear eigenvalue problems (cont.)

- Nonlinear eigenvalue problems

  More general: Find $\lambda \in \mathbb{C}$ such that $A(\lambda)\boldsymbol{x} = \mathbf{0}$ where $A(\lambda)$ is a matrix the elements of which depend on λ.

  Examples: $A(\lambda) = \sum_{k=0}^{d} \lambda^k A_k$;
  $d = 1$: $A(\lambda) = A_0 - \lambda A_1$, $A_0 = A$, $A_1 = I$.

Linear and nonlinear eigenvalue problems (cont.)

- Matrix polynomials

  Matrix polynomials can be linearized.

  Example: $A\mathbf{x} + \lambda K\mathbf{x} + \lambda^2 M\mathbf{x}$.

  We can generate equivalent eigenvalue problems that are linear but have the size doubled: With $\mathbf{y} = \lambda\mathbf{x}$ we get

$$\begin{pmatrix} A & O \\ O & I \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} -K & -M \\ I & O \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

  or

$$\begin{pmatrix} A & K \\ O & I \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} O & -M \\ I & O \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}.$$

  Many other linearizations exist.

  (C.f. transformation of high order to first order ODE's.)

Numerical example

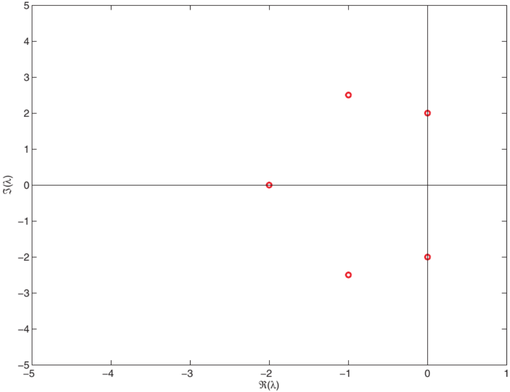
- Example: The matrix

$$A = \begin{pmatrix} -0.9880 & 1.8000 & -0.8793 & -0.5977 & -0.7819 \\ -1.9417 & -0.5835 & -0.1846 & -0.7250 & 1.0422 \\ 0.6003 & -0.0287 & -0.5446 & -2.0667 & -0.3961 \\ 0.8222 & 1.4453 & 1.3369 & -0.6069 & 0.8043 \\ -0.4187 & -0.2939 & 1.4814 & -0.2119 & -1.2771 \end{pmatrix}$$

has eigenvalues given approximately by $\lambda_1 = -2$, $\lambda_2 = -1 + 2.5\imath$, $\lambda_3 = -1 - 2.5\imath$, $\lambda_4 = 2\imath$, and $\lambda_5 = -2\imath$.

It is known that closed form formulas for the roots of a polynomial do not generally exist if the polynomial is of degree 5 or higher. Thus we cannot expect to be able to solve the eigenvalue problem in a finite procedure.

Numerical example (cont.)



Eigenvalues in $\mathbb{C}$. For real matrices, the complex eigenvalues come in pairs. If λ is an eigenvalue, then so is $\bar{\lambda}$.

Zeros of determinant

Find values $\lambda \in \mathbb{C}$ such that $A - \lambda I$ is singular.

Equivalent:

Find values $\lambda \in \mathbb{C}$ such that

$$\det A(\lambda) = 0. \tag{1}$$

Apply zero finder to eq. (1).

Questions:

1. What zero finder?
2. How to compute $f(\lambda) = \det A(\lambda)$?
3. How to compute $f'(\lambda) = \frac{d}{d\lambda} \det A(\lambda)$?

Gaussian elimination with partial pivoting (GEPP)

Let the factorization

$$P(\lambda)A(\lambda) = L(\lambda)U(\lambda)$$

be obtained by GEPP.

P: permutation matrix,
L: lower unit triangular matrix,
U: upper triangular matrix.

$$\det P(\lambda) \cdot \det A(\lambda) = \det L(\lambda) \cdot \det U(\lambda).$$

$$\pm 1 \cdot \det A(\lambda) = 1 \cdot \prod_{i=1}^{n} u_{ii}(\lambda).$$

Newton iteration

Need the derivative $f'(\lambda)$ of $f(\lambda) = \det A(\lambda)$.

$$
\begin{aligned}
f'(\lambda) &= \pm 1 \cdot \sum_{i=1}^{n} u_{ii}'(\lambda) \prod_{j \neq i}^{n} u_{jj}(\lambda) \\
&= \pm 1 \cdot \sum_{i=1}^{n} \frac{u_{ii}'(\lambda)}{u_{ii}(\lambda)} \prod_{j=1}^{n} u_{jj}(\lambda) = \sum_{i=1}^{n} \frac{u_{ii}'(\lambda)}{u_{ii}(\lambda)} f(\lambda).
\end{aligned}
$$

How do we compute the u_{ii}'?

Possibility: algorithmic differentiation

See: Arbenz & Gander: Solving Nonlinear Eigenvalue Problems by Algorithmic Differentiation. Computing 36, 205 – 215 (1986).

Algorithmic differentiation

Example: Horner scheme to evaluate polynomial

$$f(z) = \sum_{i=1}^{n} c_i z^i.$$

$$p_0(z) = c_0 + z\,(c_1 + z\,(c_2 + \cdots + z\,(c_n)))$$

by the recurrence

$$\begin{aligned} p_n &:= c_n, \\ p_i &:= z\,p_{i+1} + c_i, \qquad i = n-1, n-2, \ldots, 0 \\ f(z) &:= p_0. \end{aligned}$$

Consider the p_i as functions (polynomials) in z.

Algorithmic differentiation (cont.)

$$
\begin{aligned}
dp_n &:= 0, \quad p_n := c_n, \\
dp_i &:= p_{i+1} + z\, dp_{i+1}, \quad p_i := z\, p_{i+1} + c_i, \quad i = n-1, n-2, \ldots, 0, \\
f'(z) &:= dp_0, \quad f(z) := p_0.
\end{aligned}
$$

Can proceed in a similar fashion for computing $\det A(\lambda)$.

Need to be able to compute derivatives a'_{ij}. Then, derive each single assignment in the algorithm of Gaussian elimination.

Discussion

We restrict ourselves to the standard eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$, i.e., $A(\lambda) = A - \lambda I$.

Then $A'(\lambda) = -I$.

In the Newton method we have to compute the determinant for possibly many values λ.

Computing the determinant costs $\frac{2}{3}n^3$ flops (floating point operations).

Can we do better?

Idea: Transform A by a similarity transformation to Hessenberg form.

Hessenberg matrices

Definition

A matrix H is a Hessenberg matrix if its elements below the lower off-diagonal are zero,

$$h_{ij} = 0, \qquad i > j + 1.$$

Any matrix A can be transformed into a Hessenberg matrix by a sequence of elementary Householder transformations, for details see QR algorithm.
Let $S^*AS = H$, where S is unitary. Then

$$A\mathbf{x} = \lambda\mathbf{x} \Longleftrightarrow H\mathbf{y} = \lambda\mathbf{y}, \qquad \mathbf{x} = S\mathbf{y}.$$

We assume that H is unreduced, i.e., $h_{i+1,i} \neq 0$ for all i.

Hessenberg matrices (cont.)

Let λ be an eigenvalue of H and

$$(H - \lambda I)\mathbf{x} = \mathbf{0}, \tag{2}$$

i.e., $\mathbf{x}$ is an eigenvector of H associated with the eigenvalue λ.

Then $x_n \neq 0$. (Proof by contradiction.)

W.l.o.g., we can set $x_n = 1$.
If λ is an eigenvalue then there are x_i, $1 \leq i < n$, such that

$$\begin{pmatrix} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Hessenberg matrices (cont.)

If λ is not an eigenvalue then we determine the x_i such that

$$\left(\begin{array}{ccc|c} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ \hline h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{array}\right)\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} * \\ 0 \\ 0 \\ 0 \end{pmatrix}. \qquad (*)$$

Determine the $n-1$ numbers $x_{n-1}, x_{n-2}, \ldots, x_1$ by the equations n down to 2 of the equation above

$$x_i = \frac{-1}{h_{i+1,i}}\left((h_{i+1,i+1}-\lambda)\, x_{i+1} + h_{i+1,i+2}\, x_{i+2} + \cdots + h_{i+1,n} \underbrace{x_n}_{1}\right).$$

The first equation gives

$$(h_{1,1}-\lambda)\, x_1 + h_{1,2}\, x_2 + \cdots + h_{1,n}\, x_n = c \cdot f(\lambda). \qquad (3)$$

Hessenberg matrices (cont.)

We can consider the x_i as functions of λ, in fact, $x_i \in \mathbb{P}_{n-i}$.

Therefore, we can algorithmically differentiate the x_i' to get $f'(\lambda)$.

For $i = n-1, \ldots, 1$ we have

$$x_i' = \frac{-1}{h_{i+1,i}}\left(-x_{i+1} + (h_{i+1,i+1} - \lambda)\, x_{i+1}' + h_{i+1,i+2}\, x_{i+2}' + \cdots + h_{i+1,n-1} x_{n-1}'\right).$$

Finally,

$$c \cdot f'(\lambda) = -x_1 + (h_{1,1} - \lambda)\, x_1' + h_{1,2}\, x_2' + \cdots + h_{1,n-1} x_{n-1}'.$$

Hessenberg matrices (matrix form)

In matrix form $(*)$ reads

$$(H - \lambda I)\begin{pmatrix}\mathbf{x}(\lambda)\\ 1\end{pmatrix} = \begin{bmatrix}\mathbf{h}(\lambda) & h_{1n}\\ R(\lambda) & \mathbf{k}(\lambda)\end{bmatrix}\begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix} = \begin{pmatrix}p(\lambda)\\ \mathbf{0}\end{pmatrix}.$$

Computing p:

$$R(\lambda)\mathbf{x}(\lambda) + \mathbf{k}(\lambda) = \mathbf{0} \quad \Longrightarrow \quad \mathbf{x}(\lambda) = -R(\lambda)^{-1}\mathbf{k}(\lambda),$$
$$p(\lambda) = \mathbf{h}(\lambda)\mathbf{x}(\lambda) + h_{1n}.$$

Hessenberg matrices (matrix form) (cont.)

Computing $q = p'$:

$$R'(\lambda)\mathbf{x}(\lambda)+R(\lambda)\mathbf{x}'(\lambda) = -\mathbf{k}'(\lambda) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad R'(\lambda) = \begin{bmatrix} 0 & -1 & & \\ & 0 & \ddots & \\ & & \ddots & -1 \\ & & & 0 \end{bmatrix}.$$

$$\mathbf{x}'(\lambda) = R(\lambda)^{-1}[-\mathbf{k}'(\lambda) - R'(\lambda)\mathbf{x} = R(\lambda)^{-1} \begin{pmatrix} x_2 \\ \vdots \\ x_{n-1} \\ 1 \end{pmatrix}$$

$$q(\lambda) = \mathbf{h}'(\lambda)\mathbf{x}(\lambda) + \mathbf{h}(\lambda)\mathbf{x}'(\lambda), \quad \mathbf{h}'(\lambda) = [-1, 0, \dots, 0].$$

Hyman's algorithm

We have shown that we can compute $f(\lambda) = \det(H(\lambda))$ and its derivative $f'(\lambda)$ of a Hessenberg matrix H in $\mathcal{O}(n^2)$ operations. Apply Newton iteration:

Choose initial guess λ_0.

While not converged,

$$\lambda_{k+1} = \lambda_k - \frac{f(\lambda_k)}{f'(\lambda_k)}, \qquad k = 0, 1, \ldots$$

Note: Higher order deriatives of f can be computed in an analogous fashion. Higher order zero finders are then applicable (e.g. Laguerre's zero finder).

Computing multiple zeros

If we have found a zero z of $f(x) = 0$ and want to compute another one, we want to avoid recomputing the already found z.

We can explicitely deflate the zero by defining a new function

$$f_1(x) := \frac{f(x)}{x - z},$$

and apply our method of choice to f_1. This procedure can in particular be done with polynomials. The coefficients of f_1 are however very sensitive to inaccuracies in z.

We can proceed similarly for multiple zeros $z_1, \ldots, z_m$.

Explicit deflation is not recommended and often not feasible since f is not given explicitely.

Computing multiple zeros (cont.)

For the reciprocal Newton correction for f_1 we get

$$\frac{f_1'(x)}{f_1(x)} = \frac{\frac{f'(x)}{x-z} - \frac{f(x)}{(x-z)^2}}{\frac{f(x)}{x-z}} = \frac{f'(x)}{f(x)} - \frac{1}{x-z}.$$

Then a Newton correction becomes

$$\boxed{x^{(k+1)} = x_k - \frac{1}{\frac{f'(x_k)}{f(x_k)} - \frac{1}{x_k - z}}}$$

and similarly for multiple zeros $z_1, \ldots, z_m$.

The above procedure is called implicit deflation. f is not modified.

In this way errors in z are not propagated to f_1

Inverse Iteration

We consider again the nonlinear eigenvalue problem

$$\begin{aligned} A(\lambda)\,\mathbf{x} &= \mathbf{0}, \\ \mathbf{c}^T\mathbf{x} &= 1, \end{aligned} \tag{4}$$

where $\mathbf{c}$ is some given vector.

For the *linear* eigenvalue problem we have $A(\lambda) = \lambda I - A$.

Solving (4) is equivalent with finding a zero of the nonlinear function $\mathbf{f}(\mathbf{x}, \lambda)$,

$$\mathbf{f}(\mathbf{x}, \lambda) = \begin{pmatrix} A(\lambda)\,\mathbf{x} \\ \mathbf{c}^T\mathbf{x} - 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}. \tag{5}$$

Inverse Iteration (cont.)

To apply Newton's zero finding method we need the Jacobian of $\boldsymbol{f}$,

$$J(\boldsymbol{x},\lambda) \equiv \frac{\partial \boldsymbol{f}(\boldsymbol{x},\lambda)}{\partial(\boldsymbol{x},\lambda)} = \begin{pmatrix} A(\lambda) & A'(\lambda)\boldsymbol{x} \\ \boldsymbol{c}^T & 0 \end{pmatrix}. \tag{6}$$

Then a step of Newton's iteration is given by

$$\begin{pmatrix} \boldsymbol{x}_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_k \\ \lambda_k \end{pmatrix} - J(\boldsymbol{x}_k,\lambda_k)^{-1}\boldsymbol{f}(\boldsymbol{x}_k,\lambda_k), \tag{7}$$

or, with the abbreviations $A_k := A(\lambda_k)$ and $A'_k := A'(\lambda_k)$,

$$\begin{pmatrix} A_k & A'_k\,\boldsymbol{x}_k \\ \boldsymbol{c}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \\ \lambda_{k+1} - \lambda_k \end{pmatrix} = \begin{pmatrix} -A_k\,\boldsymbol{x}_k \\ 1 - \boldsymbol{c}^T\boldsymbol{x}_k \end{pmatrix}. \tag{8}$$

Inverse Iteration (cont.)

If $\boldsymbol{x}_k$ is normalized, $\boldsymbol{c}^T\boldsymbol{x}_k = 1$, then second equation in (8) yields

$$\boldsymbol{c}^T(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = 0 \quad \Longleftrightarrow \quad \boldsymbol{c}^T\boldsymbol{x}_{k+1} = 1. \tag{9}$$

First equation in (8) gives

$$\begin{aligned} A_k\,(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) + (\lambda_{k+1} - \lambda_k)\,A'_k\,\boldsymbol{x}_k &= -A_k\,\boldsymbol{x}_k \\ \Longleftrightarrow \quad A_k\,\boldsymbol{x}_{k+1} &= -(\lambda_{k+1} - \lambda_k)\,A'_k\,\boldsymbol{x}_k. \end{aligned}$$

Introduce auxiliary vector $\boldsymbol{u}_{k+1}$:

$$A_k\,\boldsymbol{u}_{k+1} = A'_k\,\boldsymbol{x}_k. \tag{10}$$

$\boldsymbol{u}_{k+1}$ points in the desired direction; it just needs to be normalized.

Inverse Iteration (cont.)

Normalizing $\boldsymbol{u}_{k+1}$ gives

$$1 = \boldsymbol{c}^T \boldsymbol{x}_{k+1} = -(\lambda_{k+1} - \lambda_k)\, \boldsymbol{c}^T \boldsymbol{u}_{k+1}, \tag{11}$$

or

$$\lambda_{k+1} = \lambda_k - \frac{1}{\boldsymbol{c}^T \boldsymbol{u}_{k+1}}. \tag{12}$$

Algorithm: Newton iteration for solving (5)

1: Choose a starting vector $\mathbf{x}_0 \in \mathbb{R}^n$ with $\boldsymbol{c}^T \boldsymbol{x}_0 = 1$. $k := 0$.
2: **repeat**
3: Solve $A(\lambda_k)\, \boldsymbol{u}_{k+1} := A'(\lambda_k)\, \boldsymbol{x}_k$ for $\boldsymbol{u}_{k+1}$; (10)
4: $\mu_k := \boldsymbol{c}^T \boldsymbol{u}_{k+1}$;
5: $\boldsymbol{x}_{k+1} := \boldsymbol{u}_{k+1}/\mu_k$; (Normalize $\boldsymbol{u}_{k+1}$)
6: $\lambda_{k+1} := \lambda_k - 1/\mu_k$; (12)
7: $k := k + 1$;
8: **until** some convergence criterion is satisfied

Note: • For linear eigenvalue problemswe have $A'(\lambda)\boldsymbol{x} = \boldsymbol{x}$.
• In above algorithm: In each iteration step a linear system has to be solved.

Successive linear approximations

$$A(\lambda)\mathbf{x} \approx (A(\lambda_k) - \vartheta A'(\lambda_k))\mathbf{x} = \mathbf{0}, \qquad \lambda = \lambda_k - \vartheta.$$

This suggests the method of successive linear problems.

Start with approximation λ_1 of an eigenvalue of $A(\lambda)$.
for $k = 1, 2, \ldots$ **do**
    Solve the linear eigenvalue problem $A(\lambda)\mathbf{u} = \vartheta A'(\lambda)\mathbf{u}$.
    Choose an eigenvalue ϑ smallest in modulus.
    $\lambda_{k+1} := \lambda_k - \vartheta;$
end for

Remark: If A is twice continuously differentiable, and λ is an eigenvalue of problem (1) such that $A'(\lambda)$ is singular and 0 is an algebraically simple eigenvalue of $A'(\lambda)^{-1}A(\lambda)$, then the method in Algorithm 3 converges quadratically towards λ.

Discussion

- Methods of today can be used to compute a few eigenvalues of small and/or dense matrices.
- Methods require a factorization of a matrix in each iteration step.
  This may lead to excessive flop counts.
- Hyman's method is designed for Hessenberg matrices.
  Transformation of large sparse matrices to Hessenberg form leads to dense matrices. So, it not suited for large sparse matrices.

References

[1] H. Voss: *Iterative projection methods for large-scale nonlinear eigenvalue problems*, TU Hamburg-Harburg, TR 2010. Available from https://www.mat.tu-harburg.de/ins/forschung/rep/rep147.pdf.

[2] A. Ruhe: *Algorithms for the nonlinear eigenvalue problem*. SIAM J. Numer. Anal. 10, 674–689, 1973.

[3] F. Tisseur and K. Meerbergen: *The quadratic eigenvalue problem*. SIAM Rev. 43, 235–286, 2001.

Exercise 3

http://people.inf.ethz.ch/arbenz/ewp/Exercises/exercise03.pdf