# Smooth Factorizations in Dynamical Systems

Andrew Binder

July 29, 2009

# Standard Eigenvalue Problem

## Definition

The standard eigenvalue problem is of the form

$$Ax = \lambda x$$

where $A$ is a matrix, $\lambda$ is an eigenvalue, and $x$ is the corresponding eigenvector. The eigenvalues must satisfy the characteristic equation

$$\det(A - \lambda I) = 0.$$

# Nonlinear Eigenvalue Problem

## Definition

The nonlinear eigenproblem is a generalization of the standard eigenvalue problem. The nonlinear problem is of the form

$$A(\lambda)x = 0 \;\; or \;\; y^* A(\lambda) = 0$$

where $A(\lambda)$ is a matrix whose entries are functions dependent on the value $\lambda$, $\lambda$ is the nonlinear eigenvalue, and $x$ and $y^*$ are the right and left nonlinear eigenvectors respectively. If $A(\lambda) = B - \lambda I$, the problem reduces to the standard eigenvalue problem. The nonlinear eigenvalues must be the solutions of the characteristic equation

$$\det A(\lambda) = 0.$$

# Quadratic Eigenproblem

## Example

Quadratic Eigenproblem:

$$A_2\lambda^2 + A_1\lambda + A_0 = 0$$

# Quadratic Eigenproblem

## Example

Quadratic Eigenproblem:

$$A_2\lambda^2 + A_1\lambda + A_0 = 0$$

Applications:

- Structural Dynamics

# Quadratic Eigenproblem

## Example

Quadratic Eigenproblem:

$$A_2\lambda^2 + A_1\lambda + A_0 = 0$$

Applications:

- Structural Dynamics
- Vibrational Problems

# Quadratic Eigenproblem

## Example

Quadratic Eigenproblem:

$$A_2\lambda^2 + A_1\lambda + A_0 = 0$$

Applications:

- Structural Dynamics
- Vibrational Problems
- Fluid Dynamics

# Matrix Decomposition

### Definition

Matrix decomposition is the factorization of a matrix into the product of new matrices.

# Matrix Decomposition

**Definition**

Matrix decomposition is the factorization of a matrix into the product of new matrices.

- QR Decomposition.
- LU Decomposition.

# Rank Revealing LU Matrix

## Rank Deficient A

$$\begin{aligned} P_1 A P_2 &= LU \\ &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix} \end{aligned}$$

# Rank Revealing LU Matrix

## Rank Deficient A

$$
\begin{aligned}
P_1 A P_2 &= LU \\
&= \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

## Example

$$
A = \begin{bmatrix} 2 & 4 & 5 \\ 2 & 4 & 3 \\ 3 & 6 & 1 \end{bmatrix}
$$

# Rank Revealing LU Matrix

## Rank Deficient A

$$
\begin{aligned}
P_1 A P_2 &= LU \\
&= \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

## Example

$$
P_1 A P_2 = \begin{bmatrix} 6 & 1 & 3 \\ 4 & 5 & 2 \\ 4 & 3 & 2 \end{bmatrix}
$$

# Rank Revealing LU Matrix

## Rank Deficient A

$$\begin{aligned} P_1 A P_2 &= LU \\ &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & 0 \end{bmatrix} \end{aligned}$$

## Example

$$LU = \begin{bmatrix} 1 & 0 & 0 \\ \frac{2}{3} & 1 & 0 \\ \frac{2}{3} & \frac{7}{13} & 1 \end{bmatrix} \begin{bmatrix} 6 & 1 & 3 \\ 0 & 4\frac{1}{3} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

# Rank Revealing QR Matrix

## Rank Deficient A

$$
\begin{aligned}
AP &= QR \\
&= Q\begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}
\end{aligned}
$$

# Determining Nonlinear Eigenvalues Through Minimization

## Goal

- Find a $\lambda$ so that $\det A(\lambda) = 0$

# Determining Nonlinear Eigenvalues Through Minimization

## Goal

- Find a $\lambda$ so that $\det A(\lambda) = 0$

## Plan

- Guess the nonlinear eigenvalue
- Perform rank revealing decomposition
- Minimize lower right block
- Repeat steps using new guess until eigenvalue is found

# Newton's Minimization Technique

## Problem

- $f(x) = 0$

- $f(\lambda) = ||U_{22}(\lambda)||_F^2 \approx ||U_{22}(\lambda_0) + U_{22}'(\lambda_0)(\lambda - \lambda_0)||_F^2 = 0$

## Iterative Method

- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

# Newton's Minimization Technique

## Problem

- $f'(x) = 0$

- $f'(\lambda) = \frac{d}{d\lambda}||U_{22}(\lambda)||_F^2 \approx \frac{d}{d\lambda}||U_{22}(\lambda_0) + U'_{22}(\lambda_0)(\lambda - \lambda_0)||_F^2 = 0$

## Iterative Method

- $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$

# Smooth Decomposition of a Nonsingular Matrix

## Lemma

*All full column rank matrices $A(\lambda) \in C^k$ with nonsingular leading principle submatrices have a unique $L(\lambda)U(\lambda) \in C^k$ decomposition.*

## Proof.

- Assume $A(\lambda) = L(\lambda)U(\lambda)$
- Determine entries of $L(\lambda)$ and $U(\lambda)$ $\square$

# Smooth Decomposition of a Matrix Nonsingular at a Point

## Theorem

*Let $A(\lambda) \in C^k$ such that $A(\lambda_0)$ is nonsingular. Assume there's a permutation matrix $P$ such that $PA(\lambda_0) = L_0 U_0$, where $L_0$ is unit lower triangular and $U_0$ is upper triangular. Then, there is a neighborhood $N(\lambda_0)$ such that*

$$PA(\lambda) = L(\lambda)U(\lambda) \; \forall \; \lambda \in N(\lambda_0),$$

*with $L(\lambda_0) = L_0$, $U(\lambda_0) = U_0$; $L(\lambda), U(\lambda) \in C^k$, $L(\lambda)$ unit lower triangular matrix, and $U(\lambda)$ upper triangular.*

## Proof.

- Locally perturb $A(\lambda_0)$ using Taylor's Theorem
- Create lower triangular matrices so that the perturbation becomes upper triangular. □

# Smooth Decomposition of a General Matrix

## Theorem

*Let $A(\lambda) \in C^k$ be a $n \times n$ matrix such that $A(\lambda_0)$ has a column rank of $n - m$, $m \leq n - 1$. Assume there are permutation matrices $P_1$, $P_2$ such that $P_1 A(\lambda_0) P_2 = L_0 U_0$, where $L_0$ is a block unit lower triangular matrix and $U_0$ is a block upper triangular matrix. Then, there is a neighborhood $N(\lambda_0)$ such that*

$$P_1 A(\lambda) P_2 = L(\lambda) U(\lambda) \; \forall \; \lambda \in N(\lambda_0),$$

*with $L(\lambda_0) = L_0$, $U(\lambda_0) = U_0$; $L(\lambda)$, $U(\lambda) \in C^k$, $L(\lambda)$ a block unit lower triangular matrix, $U(\lambda)$ a block upper triangular matrix.*

# LU Algorithm for Computation of Nonlinear Eigenvalues

*Step 1*: Given an initial approximation $\lambda_0$ to $\lambda_*$
*Step 2*: Compute

$$A(\lambda_i) \text{ and } A'(\lambda_i),\ \ i = 0, 1, \cdots$$

*Step 3*: Compute the *LU* decomposition with complete column pivoting of $A(\lambda_i)$:

$$P_1 A(\lambda_i) P_2 = L(\lambda_i) U(\lambda_i)$$

# LU Algorithm for Computation of Nonlinear Eigenvalues

*Step 4*: Compute

$$U'_{2,2}(\lambda_i) = (L_i^{-1} P_1 A'(\lambda_i) P_2)_{2,2} - (L_i^{-1} P_1 A'(\lambda_i) P_2)_{2,1} ({U_{1,1}^{(i)}}^{-1} U_{1,2}^{(i)})$$

*Step 5*: Compute

$$\lambda_{i+1} = \lambda_i - \frac{(\text{col } U'_{2,2}(\lambda_i))^H \cdot \text{col } U_{2,2}(\lambda_i)}{||U'_{2,2}(\lambda_i)||_F^2}.$$

*Step 6*: If the desired accuracy is attained, stop the iteration. Otherwise, repeat steps 2-6.

# QR Algorithm for Computation of Nonlinear Eigenvalues

Step 3: Compute the $LU$ decomposition with complete column pivoting of $A(\lambda_i)$:

$$A(\lambda_i)P = Q(\lambda_i)R(\lambda_i)$$

Step 5: Compute

$$\lambda_{i+1} = \lambda_i - \frac{(\mathrm{col}\ R'_{2,2}(\lambda_i))^H \cdot \mathrm{col}\ R_{2,2}(\lambda_i)}{||R'_{2,2}(\lambda_i)||_F^2}.$$

# Theory of Numerical Rank Determination

## Property

Let $AP = QR$ be a rank revealing decomposition. Then, the diagonals of $R$ have the property that

$$|r_{1,1}| \geq \cdots \geq |r_{t,t}| >> |r_{t+1,t+1}| \geq \cdots \geq |r_{n,n}|.$$

# Theory of Numerical Rank Determination

**Property**

Let $AP = QR$ be a rank revealing decomposition. Then, the diagonals of $R$ have the property that

$$|r_{1,1}| \geq \cdots \geq |r_{t,t}| >> |r_{t+1,t+1}| \geq \cdots \geq |r_{n,n}|.$$

$$|r_{t+1,t+1}| \leq \epsilon |r_{1,1}| \leq |r_{t,t}|$$

# Theory of Numerical Rank Determination

## Property

Let $AP = QR$ be a rank revealing decomposition. Then, the diagonals of $R$ have the property that

$$|r_{1,1}| \geq \cdots \geq |r_{t,t}| >> |r_{t+1,t+1}| \geq \cdots \geq |r_{n,n}|.$$

$$\frac{|r_{t+1,t+1}|}{|r_{1,1}|} \leq \epsilon \leq \frac{|r_{t,t}|}{|r_{1,1}|}.$$

# 4 x 4 Time Comparison

Table: Time [ms] Comparison of 4 x 4 Algorithm Performance

| Nonlinear Matrix | LU Average | QR Average | Ratio of Averages (QR / LU) |
|---|---|---|---|
| Q | 4.369 | 16.141 | 3.695 |
| Q, E | 4.445 | 15.943 | 3.587 |
| Q, S | 4.472 | 16.088 | 3.597 |
| Q, E, S | 4.568 | 16.332 | 3.575 |

# 10 x 10 Time Comparison

Table: Time [ms] Comparison of 10 x 10 Algorithm Performance

| Nonlinear Matrix | LU Average | QR Average | Ratio of Averages (QR / LU) |
|---|---|---|---|
| Q | 9.632 | 44.052 | 4.574 |
| Q, E | 9.829 | 44.035 | 4.480 |
| Q, S | 10.088 | 44.643 | 4.425 |
| Q, E, S | 10.166 | 46.169 | 4.541 |

# 100 x 100 Time Comparison

Table: Time [ms] Comparison of 100 x 100 Algorithm Performance

| Nonlinear Matrix | LU Average | QR Average | Ratio of Averages (QR / LU) |
|---|---|---|---|
| Q | 391.154 | 1696.066 | 4.336 |
| Q, E | 362.494 | 1630.445 | 4.498 |
| Q, S | 393.234 | 1634.839 | 4.157 |
| Q, E, S | 389.039 | 1650.813 | 4.243 |

# Iteration Comparison

Table: Average Iteration Comparison of 10 x 10 Algorithm Performance

| Nonlinear Matrix | LU | | QR | |
|---|---|---|---|---|
| | Number | Time/Iter [ms] | Number | Time/Iter [ms] |
| Q | 4.40 | 2.19 | 4.28 | 10.29 |
| Q, E | 4.44 | 2.21 | 4.26 | 10.32 |
| Q, S | 4.51 | 2.24 | 4.29 | 10.42 |
| Q, E, S | 4.38 | 2.32 | 4.27 | 10.81 |

# Newton Steffensen Method

## Cubic Convergence Iterative Formula

Applying Steffensen's acceleration method to Newton's root finding method generates an iterative formula with cubic convergence. Let $f(x_*) = 0$ and let $x_0$ be sufficiently close to $x_*$, then the successive iterative approximations are determined by

$$x_{n+1} = x_n - \frac{f^2(x_n)}{f'(x_n)(f(x_n) - f(x_n^*))}$$

where

$$x_n^* = x_n - \frac{f(x_n)}{f'(x_n)}.$$

# Newton Steffensen Method

$$x_{n+1} = x_n - \frac{f'^2(x_n)}{f''(x_n)(f'(x_n) - f'(x_n^*))}$$

where

$$x_n^* = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

- $f'(\lambda) = (\mathrm{col}\ U'_{2,2}(\lambda_i))^H \cdot \mathrm{col}\ U_{2,2}(\lambda_i)$
- $f''(\lambda) = ||U'_{2,2}(\lambda_i)||_F^2$

# Cubic Time Comparison

Table: Time [ms] Comparison of 10 x 10 Cubic Convergence Algorithm Performance

| Nonlinear Matrix | LU Average | QR Average | Ratio of Averages (QR / LU) |
|---|---|---|---|
| Q | 12.323 | 57.906 | 4.699 |
| Q, E | 12.124 | 57.504 | 4.743 |
| Q, S | 13.311 | 62.117 | 4.667 |
| Q, E, S | 12.422 | 59.012 | 4.751 |

# Cubic Iteration Comparison

Table: Average Iteration Comparison of 10 x 10 Cubic Convergence Algorithm Performance

| Nonlinear Matrix | LU | | QR | |
|---|---|---|---|---|
| | Number | Time/Iter [ms] | Number | Time/Iter [ms] |
| Q | 3.26 | 3.775 | 3.13 | 18.480 |
| Q, E | 3.19 | 3.799 | 3.11 | 18.511 |
| Q, S | 3.48 | 3.823 | 3.23 | 19.052 |
| Q, E, S | 3.18 | 3.884 | 3.12 | 18.826 |

# Cost

**Lines where the most time was spent**

| Line Number | Code | Calls | Total Time | % Time | Time Plot |
|---|---|---|---|---|---|
| 59 | `[Q, R, P] = apqr(Ak);` | 5033 | 43.345 s | 95.4% | |
| 82 | `dR22 = Knt'*QAP*Knt - Knt'*QAP...` | 5033 | 0.859 s | 1.9% | |
| 76 | `QAP = Q'*dAk*P;` | 5033 | 0.223 s | 0.5% | |
| 92 | `guess = guess - coldR22'*colR2...` | 5033 | 0.158 s | 0.3% | |
| 50 | `Ak = Mnot + Mone*guess + Mtwo*...` | 5033 | 0.118 s | 0.3% | |
| All other lines | | | 0.724 s | 1.6% | |
| Totals | | | 45.426 s | 100% | |

# Super Quadratic Convergence

## Goal

- Approximate $f'(x_n^*)$ using previously calculated values.

# Super Quadratic Convergence

## Goal

- Approximate $f'(x_n^*)$ using previously calculated values.

- Add another term in the Taylor's Series expansion approximation.

# Super Quadratic Convergence

**Goal**

- Approximate $f'(x_n^*)$ using previously calculated values.

- Add another term in the Taylor's Series expansion approximation.
- Solve for $f'(x_n^*)$.

# Vibrating Train Tracks

- Vibrating rail track resting on sleepers (lateral supports)

# Vibrating Train Tracks

- Vibrating rail track resting on sleepers (lateral supports)
- Initially modeled as a partial differential equation

# Vibrating Train Tracks

- Vibrating rail track resting on sleepers (lateral supports)
- Initially modeled as a partial differential equation
- Discretized and turned into a quadratic eigenvalue problem with 10 x 10 matrices

# Vibrating Train Tracks

- Vibrating rail track resting on sleepers (lateral supports)
- Initially modeled as a partial differential equation
- Discretized and turned into a quadratic eigenvalue problem with 10 x 10 matrices
- Eigenvalues are explicitly known. There exist multiple eigenvalues.

# Vibrating Train Tracks

- Vibrating rail track resting on sleepers (lateral supports)
- Initially modeled as a partial differential equation
- Discretized and turned into a quadratic eigenvalue problem with 10 x 10 matrices
- Eigenvalues are explicitly known. There exist multiple eigenvalues.
- Algorithm was successful within an error tolerance of $10^{-15}$

# Vibrating Train Tracks

$$U = \begin{bmatrix} 0.66 & 0 & 0 & 0 & -0.25 & 0.31 & 0.31 & -0.25 & 0 & 0 \\ 0 & 0.66 & 0 & 0 & 0.31 & -0.25 & 0 & 0 & 0.31 & -0.25 \\ 0 & 0 & 0.66 & -0.25 & 0 & 0 & 0.31 & 0 & -0.25 & 0.31 \\ 0 & 0 & 0 & 0.57 & 0 & 0 & -0.13 & 0.31 & 0.22 & 0.12 \\ 0 & 0 & 0 & 0 & 0.42 & -0.01 & 0.12 & 0.22 & -0.15 & 0.12 \\ 0 & 0 & 0 & 0 & 0 & 0.42 & -0.14 & 0.13 & 0.11 & 0.22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.25 & -0.08 & 0.25 & -0.08 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.22 & 0 & 0.22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

# Next Step

- Analyze the super quadratic algorithm
- Take advantage of matrix structure such as symmetry
- Determine all eigenvalues in a region
- MATLAB polynomial nonlinear eigenvalue solver