Highly Accurate Numerical Linear Algebra

Froilán M. Dopico

Department of Mathematics and ICMAT, Universidad Carlos III de Madrid, Spain

XXIII CEDYA / XIII CMA
Universitat Jaume I
Castelló de la Plana (Spain). September 9th-13th, 2013

Outline

Outline

What is Numerical Linear Algebra? (I)

Numerical Linear Algebra is a part of Numerical Analysis that develops efficient and stable algorithms to solve

- Systems of linear equations:

  $$A\,x = b\,,$$

  where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$;

- Least squares problems:

  $$\min_x \|A\,x - b\|_2\,,$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^{m\times 1}$;

- Matrix eigenvalue/vector problems:

  $$A\,x = \lambda\,x\,,$$

  where $A \in \mathbb{R}^{n\times n}$;

- Matrix singular value problems:

  $$A^T\,A\,x = \sigma^2\,x\,,$$

  where $A \in \mathbb{R}^{m\times n}$.

What is Numerical Linear Algebra? (I)

Numerical Linear Algebra is a part of Numerical Analysis that develops efficient and stable algorithms to solve

- Systems of linear equations:

  $$A\,x = b\,,$$

  where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$;

- Least squares problems:

  $$\min_x \|A\,x - b\|_2\,,$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^{m\times 1}$;

- Matrix eigenvalue/vector problems:

  $$A\,x = \lambda\,x\,,$$

  where $A \in \mathbb{R}^{n\times n}$;

- Matrix singular value problems:

  $$A^T\,A\,x = \sigma^2\,x\,,$$

  where $A \in \mathbb{R}^{m\times n}$.

What is Numerical Linear Algebra? (I)

Numerical Linear Algebra is a part of Numerical Analysis that develops efficient and stable algorithms to solve

- Systems of linear equations:

  $$A\,x = b\,,$$

  where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$;

- Least squares problems:

  $$\min_x \|A\,x - b\|_2\,,$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^{m\times 1}$;

- Matrix eigenvalue/vector problems:

  $$A\,x = \lambda\,x\,,$$

  where $A \in \mathbb{R}^{n\times n}$;

- Matrix singular value problems:

  $$A^T\,A\,x = \sigma^2\,x\,,$$

  where $A \in \mathbb{R}^{m\times n}$.

What is Numerical Linear Algebra? (I)

Numerical Linear Algebra is a part of Numerical Analysis that develops efficient and stable algorithms to solve

- Systems of linear equations:

  $$A\,x = b\,,$$

  where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$;

- Least squares problems:

  $$\min_x \|A\,x - b\|_2\,,$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^{m\times 1}$;

- Matrix eigenvalue/vector problems:

  $$A\,x = \lambda\,x\,,$$

  where $A \in \mathbb{R}^{n\times n}$;

- Matrix singular value problems:

  $$A^T\,A\,x = \sigma^2\,x\,,$$

  where $A \in \mathbb{R}^{m\times n}$.

What is Numerical Linear Algebra? (I)

Numerical Linear Algebra is a part of Numerical Analysis that develops efficient and stable algorithms to solve

- Systems of linear equations:

  $$A\,x = b\,,$$

  where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$;

- Least squares problems:

  $$\min_x \|A\,x - b\|_2\,,$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^{m\times 1}$;

- Matrix eigenvalue/vector problems:

  $$A\,x = \lambda\,x\,,$$

  where $A \in \mathbb{R}^{n\times n}$;

- Matrix singular value problems:

  $$A^T\,A\,x = \sigma^2\,x\,,$$

  where $A \in \mathbb{R}^{m\times n}$.

What is Numerical Linear Algebra? (II)

The problems in previous slide are the **classic problems** of Numerical Linear Algebra, **but nowadays many other problems are considered** in Numerical Linear Algebra. Among them:

- Efficient and stable algorithms for polynomial eigenvalue problems:

$$\left(A_k \lambda^k + A_{k-1} \lambda^{k-1} + \cdots + A_1 \lambda + A_0 \right) x = 0 \, ,$$

  where $A_i \in \mathbb{R}^{n \times n}$, for $i = 0, 1, \ldots, k$;

- Efficient and stable algorithms for more general nonlinear eigenvalue problems:

$$F(\lambda) \, x = 0,$$

  where $F : \mathbb{C} \to \mathbb{C}^{n \times n}$.

- Linear and nonlinear matrix equations (Sylvester, Lyapunov, Riccati,...)
- Matrix nearness problems, Matrix optimization problems, *Tensor computations* (or numerical MULTIlinear algebra),...

What is Numerical Linear Algebra? (II)

The problems in previous slide are the **classic problems** of Numerical Linear Algebra, **but nowadays many other problems are considered** in Numerical Linear Algebra. Among them:

- Efficient and stable algorithms for polynomial eigenvalue problems:

$$\left(A_k \lambda^k + A_{k-1} \lambda^{k-1} + \cdots + A_1 \lambda + A_0 \right) x = 0 \,,$$

  where $A_i \in \mathbb{R}^{n \times n}$, for $i = 0, 1, \ldots, k$;

- Efficient and stable algorithms for more general nonlinear eigenvalue problems:

$$F(\lambda)\, x = 0,$$

  where $F : \mathbb{C} \to \mathbb{C}^{n \times n}$.

- Linear and nonlinear matrix equations (Sylvester, Lyapunov, Riccati,...)
- Matrix nearness problems, Matrix optimization problems, *Tensor computations* (or numerical MULTIlinear algebra),...

The problems in previous slide are the **classic problems** of Numerical Linear Algebra, **but nowadays many other problems are considered** in Numerical Linear Algebra. Among them:

- Efficient and stable algorithms for polynomial eigenvalue problems:

$$\left(A_k \lambda^k + A_{k-1} \lambda^{k-1} + \cdots + A_1 \lambda + A_0 \right) x = 0 \,,$$

  where $A_i \in \mathbb{R}^{n \times n}$, for $i = 0, 1, \ldots, k$;

- Efficient and stable algorithms for more general nonlinear eigenvalue problems:

$$F(\lambda)\, x = 0,$$

  where $F : \mathbb{C} \to \mathbb{C}^{n \times n}$.

- Linear and nonlinear matrix equations (Sylvester, Lyapunov, Riccati,...)

- Matrix nearness problems, Matrix optimization problems, *Tensor computations* (or numerical MULTIlinear algebra),...

The problems in previous slide are the **classic problems** of Numerical Linear Algebra, **but nowadays many other problems are considered** in Numerical Linear Algebra. Among them:

- Efficient and stable algorithms for polynomial eigenvalue problems:

$$\left(A_k \, \lambda^k + A_{k-1} \, \lambda^{k-1} + \cdots + A_1 \, \lambda + A_0 \right) x = 0 \, ,$$

  where $A_i \in \mathbb{R}^{n \times n}$, for $i = 0, 1, \ldots, k$;

- Efficient and stable algorithms for more general nonlinear eigenvalue problems:

$$F(\lambda) \, x = 0,$$

  where $F : \mathbb{C} \to \mathbb{C}^{n \times n}$.

- Linear and nonlinear matrix equations (Sylvester, Lyapunov, Riccati,...)
- Matrix nearness problems, Matrix optimization problems, *Tensor computations* (or numerical MULTIlinear algebra),...

What is Numerical Linear Algebra? (II)

The problems in previous slide are the **classic problems** of Numerical Linear Algebra, **but nowadays many other problems are considered** in Numerical Linear Algebra. Among them:

- Efficient and stable algorithms for polynomial eigenvalue problems:

$$\left(A_k \lambda^k + A_{k-1} \lambda^{k-1} + \cdots + A_1 \lambda + A_0 \right) x = 0 \,,$$

  where $A_i \in \mathbb{R}^{n \times n}$, for $i = 0, 1, \ldots, k$;

- Efficient and stable algorithms for more general nonlinear eigenvalue problems:

$$F(\lambda)\, x = 0,$$

  where $F : \mathbb{C} \to \mathbb{C}^{n \times n}$.

- Linear and nonlinear matrix equations (Sylvester, Lyapunov, Riccati,...)
- Matrix nearness problems, Matrix optimization problems, *Tensor computations* (or numerical MULTIlinear algebra),...

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as direct algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of dense matrices are termed direct, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as **direct** algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of **dense** matrices are termed **direct**, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as **direct** algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of **dense** matrices are termed **direct**, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as **direct** algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of **dense** matrices are termed **direct**, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as **direct** algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of **dense** matrices are termed **direct**, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (I)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

- These algorithms are also known as **direct** algorithms since they terminate after an essentially fixed number of operations.
- In the jargon of Numerical Linear Algebra even the algorithms for eigenvalues of **dense** matrices are termed **direct**, since they deliver the maximum accuracy allowed by the computer in $O(n^3)$ operations.
- What are dense matrices? Those that are not represented in terms of a number of parameters much smaller than n^2.
- The cost $O(n^3)$ implies that these algorithms can be used only for matrices of moderate size: $n \lesssim 20000$.

Dense versus Sparse Numerical Linear Algebra (II)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

(b) Algorithms for $n \times n$ matrices with n very large ($n \gtrsim 10^5$) and that can be represented in terms of a number of parameters $\ll O(n^2)$, with computational cost $\ll O(n^3)$ operations.

- Very often the matrices considered by these algorithms are **sparse**, in the sense that have many zero entries,
- but they may not have zero entries at all and just to allow an sparse representation as, for instance, **low rank matrices**.
- Very often these algorithms are **iterative**.

Dense versus Sparse Numerical Linear Algebra (II)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

(b) Algorithms for $n \times n$ matrices with n very large ($n \gtrsim 10^5$) and that can be represented in terms of a number of parameters $\ll O(n^2)$, with computational cost $\ll O(n^3)$ operations.

- Very often the matrices considered by these algorithms are **sparse**, in the sense that have many zero entries,
- but they may not have zero entries at all and just to allow an sparse representation as, for instance, **low rank matrices**.
- Very often these algorithms are **iterative**.

Dense versus Sparse Numerical Linear Algebra (II)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

(b) Algorithms for $n \times n$ matrices with n very large ($n \gtrsim 10^5$) and that can be represented in terms of a number of parameters $\ll O(n^2)$, with computational cost $\ll O(n^3)$ operations.

- Very often the matrices considered by these algorithms are **sparse**, in the sense that have many zero entries,
- but they may not have zero entries at all and just to allow an sparse representation as, for instance, **low rank matrices**.
- Very often these algorithms are **iterative**.

Dense versus Sparse Numerical Linear Algebra (II)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

(b) Algorithms for $n \times n$ matrices with n very large ($n \gtrsim 10^5$) and that can be represented in terms of a number of parameters $\ll O(n^2)$, with computational cost $\ll O(n^3)$ operations.

- Very often the matrices considered by these algorithms are **sparse**, in the sense that have many zero entries,
- but they may not have zero entries at all and just to allow an sparse representation as, for instance, **low rank matrices**.
- Very often these algorithms are **iterative**.

Dense versus Sparse Numerical Linear Algebra (II)

Traditionally the algorithms of Numerical Linear Algebra are divided into two classes:

(a) Algorithms for **dense** $n \times n$ matrices with computational cost of $O(n^3)$ arithmetic operations.

(b) Algorithms for $n \times n$ matrices with n very large ($n \gtrsim 10^5$) and that can be represented in terms of a number of parameters $\ll O(n^2)$, with computational cost $\ll O(n^3)$ operations.

- Very often the matrices considered by these algorithms are **sparse**, in the sense that have many zero entries,
- but they may not have zero entries at all and just to allow an sparse representation as, for instance, **low rank matrices**.
- Very often these algorithms are **iterative**.

Dense versus Sparse Numerical Linear Algebra (III)

(a) Examples of **dense/direct** algorithms:

- Gaussian elimination for linear systems of equations.
- Householder-QR for least squares problems.
- Francis-QR for eigenvalues and eigenvectors.

(b) Examples of **sparse/iterative** algorithms:

- Conjugate gradient and GMRES for systems of equations.
- Multigrid for systems of equations.
- Lanczos and Arnoldi for eigenvalues and eigenvectors.
- Jacobi-Davidson for eigenvalues and eigenvectors.

Dense versus Sparse Numerical Linear Algebra (III)

(a) Examples of **dense/direct** algorithms:

- Gaussian elimination for linear systems of equations.
- Householder-QR for least squares problems.
- Francis-QR for eigenvalues and eigenvectors.

(b) Examples of **sparse/iterative** algorithms:

- Conjugate gradient and GMRES for systems of equations.
- Multigrid for systems of equations.
- Lanczos and Arnoldi for eigenvalues and eigenvectors.
- Jacobi-Davidson for eigenvalues and eigenvectors.

“Accurate” versus “Standard” Numerical Linear Algebra

- We see that there is a widely accepted division between **dense/direct** and **sparse/iterative** Numerical Linear Algebra.

- In this talk, I pretend first to establish a much less known division between **accurate** and **standard** Numerical Linear Algebra.

In order to understand this division, we need before to revise a few key concepts of rounding error analysis in Numerical Linear Algebra.

- We see that there is a widely accepted division between **dense/direct** and **sparse/iterative** Numerical Linear Algebra.
- In this talk, I pretend first to establish a much less known division between **accurate** and **standard** Numerical Linear Algebra.

In order to understand this division, we need before to revise a few key concepts of rounding error analysis in Numerical Linear Algebra.

- We see that there is a widely accepted division between **dense/direct** and **sparse/iterative** Numerical Linear Algebra.
- In this talk, I pretend first to establish a much less known division between **accurate** and **standard** Numerical Linear Algebra.

In order to understand this division, we need before to revise a few key concepts of rounding error analysis in Numerical Linear Algebra.

Outline

Numbers and arithmetic in current computers (I)

- Computers can only represent a finite subset of the real numbers, which is called the set of **floating point** numbers, denoted by $\mathbb{F}$. This fact produces errors.
- $\mathbb{F}$ is not closed under basic arithmetic operations $(+, -, \times, /)$, but when they are performed on a computer, they must give another number of $\mathbb{F}$. This fact produces further errors.
- These two facts are encapsulated into the **axioms of rounding error analysis.**

Numbers and arithmetic in current computers (I)

- Computers can only represent a finite subset of the real numbers, which is called the set of **floating point** numbers, denoted by $\mathbb{F}$. This fact produces errors.
- $\mathbb{F}$ is not closed under basic arithmetic operations $(+, -, \times, /)$, but when they are performed on a computer, they must give another number of $\mathbb{F}$. This fact produces further errors.
- These two facts are encapsulated into the **axioms of rounding error analysis.**

Numbers and arithmetic in current computers (I)

- Computers can only represent a finite subset of the real numbers, which is called the set of **floating point** numbers, denoted by $\mathbb{F}$. This fact produces errors.
- $\mathbb{F}$ is not closed under basic arithmetic operations $(+, -, \times, /)$, but when they are performed on a computer, they must give another number of $\mathbb{F}$. This fact produces further errors.
- These two facts are encapsulated into the **axioms of rounding error analysis.**

Numbers and arithmetic in current computers (II)

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u},$$

where $\mathbf{u}$ is the **unit roundoff of the computer!!!** ($\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$ in IEEE double precision).

Axiom 2. Arithmetic

If $x, y \in \mathbb{F}$ and $\mathbf{op} \in \{+, -, \times, /\}$, then

$$\text{computed}(x\,\mathbf{op}\,y) = (x\,\mathbf{op}\,y)\,(1+\alpha), \qquad |\alpha| \leq \mathbf{u},$$

where $(x\,\mathbf{op}\,y)$ is the exact result, that may not be in $\mathbb{F}$.

In plain words: the relative error committed in rounding a single number or in performing a single arithmetic operation on a computer is bounded by $\mathbf{u}$.

Numbers and arithmetic in current computers (II)

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u},$$

where $\mathbf{u}$ is the **unit roundoff of the computer!!!** ($\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$ in IEEE double precision).

Axiom 2. Arithmetic

If $x, y \in \mathbb{F}$ and $\mathbf{op} \in \{+, -, \times, /\}$, then

$$\text{computed}(x\,\mathbf{op}\,y) = (x\,\mathbf{op}\,y)\,(1+\alpha), \qquad |\alpha| \leq \mathbf{u},$$

where $(x\,\mathbf{op}\,y)$ is the exact result, that may not be in $\mathbb{F}$.

In plain words: the relative error committed in rounding a single number or in performing a single arithmetic operation on a computer is bounded by $\mathbf{u}$.

Numbers and arithmetic in current computers (II)

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u},$$

where $\mathbf{u}$ is the **unit roundoff of the computer!!!** ($\mathbf{u} = 2^{-53} \approx 1.11 \times 10^{-16}$ in IEEE double precision).

Axiom 2. Arithmetic

If $x, y \in \mathbb{F}$ and $\mathbf{op} \in \{+, -, \times, /\}$, then

$$\text{computed}(x\,\mathbf{op}\,y) = (x\,\mathbf{op}\,y)\,(1+\alpha), \qquad |\alpha| \leq \mathbf{u},$$

where $(x\,\mathbf{op}\,y)$ is the exact result, that may not be in $\mathbb{F}$.

In plain words: the relative error committed in rounding a single number or in performing a single arithmetic operation on a computer is bounded by $\boxed{\mathbf{u}}$.

Consequences:

- The unit roundoff $\mathbf{u}$ necessarily appears in any error bound for the output of any algorithm running on a computer.
- Relative rounding error bounds cannot be smaller than $\mathbf{u}$, since this is the bound for the error committed in just one arithmetic operation,
- that is, if y is the exact output of any algorithm and $\widehat{y}$ is the computed output when the algorithm runs in a computer, **the best error bound that can be expected is**

$$\frac{|y - \widehat{y}|}{|y|} \leq O(\mathbf{u}),$$

  with the constant in $O(\mathbf{u})$ roughly equal to the number of operations.
- **Standard algorithms** in Numerical Linear Algebra **do NOT guarantee** such error bounds, but are valid in general.
- **(Highly) accurate algorithms** in Numerical Linear Algebra **guarantee** such error bounds, but are NOT valid in general.

Consequences:

- The unit roundoff $\mathbf{u}$ necessarily appears in any error bound for the output of any algorithm running on a computer.
- Relative rounding error bounds cannot be smaller than $\mathbf{u}$, since this is the bound for the error committed in just one arithmetic operation,
- that is, if y is the exact output of any algorithm and $\widehat{y}$ is the computed output when the algorithm runs in a computer, **the best error bound that can be expected is**

$$\frac{|y - \widehat{y}|}{|y|} \leq O(\mathbf{u}),$$

  with the constant in $O(\mathbf{u})$ roughly equal to the number of operations.
- **Standard algorithms** in Numerical Linear Algebra **do NOT guarantee** such error bounds, but are valid in general.
- **(Highly) accurate algorithms** in Numerical Linear Algebra **guarantee** such error bounds, but are NOT valid in general.

First accuracy limit of an algorithm in Numerical Linear Algebra

Consequences:

- The unit roundoff $\mathbf{u}$ necessarily appears in any error bound for the output of any algorithm running on a computer.
- Relative rounding error bounds cannot be smaller than $\mathbf{u}$, since this is the bound for the error committed in just one arithmetic operation,
- that is, if y is the exact output of any algorithm and $\widehat{y}$ is the computed output when the algorithm runs in a computer, **the best error bound that can be expected is**

$$\frac{|y - \widehat{y}|}{|y|} \leq O(\mathbf{u}),$$

with the constant in $O(\mathbf{u})$ roughly equal to the number of operations.

- **Standard algorithms** in Numerical Linear Algebra **do NOT guarantee** such error bounds, but are valid in general.
- **(Highly) accurate algorithms** in Numerical Linear Algebra **guarantee** such error bounds, but are NOT valid in general.

Consequences:

- The unit roundoff $\mathbf{u}$ necessarily appears in any error bound for the output of any algorithm running on a computer.
- Relative rounding error bounds cannot be smaller than $\mathbf{u}$, since this is the bound for the error committed in just one arithmetic operation,
- that is, if y is the exact output of any algorithm and $\widehat{y}$ is the computed output when the algorithm runs in a computer, **the best error bound that can be expected is**

$$\frac{|y - \widehat{y}|}{|y|} \leq O(\mathbf{u}),$$

with the constant in $O(\mathbf{u})$ roughly equal to the number of operations.

- **Standard algorithms** in Numerical Linear Algebra **do NOT guarantee** such error bounds, but are valid in general.
- **(Highly) accurate algorithms** in Numerical Linear Algebra **guarantee** such error bounds, but are NOT valid in general.

Consequences:

- The unit roundoff $\mathbf{u}$ necessarily appears in any error bound for the output of any algorithm running on a computer.
- Relative rounding error bounds cannot be smaller than $\mathbf{u}$, since this is the bound for the error committed in just one arithmetic operation,
- that is, if y is the exact output of any algorithm and $\widehat{y}$ is the computed output when the algorithm runs in a computer, **the best error bound that can be expected is**

  $$\frac{|y-\widehat{y}|}{|y|} \leq O(\mathbf{u}),$$

  with the constant in $O(\mathbf{u})$ roughly equal to the number of operations.
- **Standard algorithms** in Numerical Linear Algebra **do NOT guarantee** such error bounds, but are valid in general.
- **(Highly) accurate algorithms** in Numerical Linear Algebra **guarantee** such error bounds, but are NOT valid in general.

Why standard algorithms are not accurate? Second accuracy limit

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Why standard algorithms are not accurate? Second accuracy limit

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Why standard algorithms are not accurate? Second accuracy limit

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Why standard algorithms are not accurate? Second accuracy limit

Axiom 1. Rounding

If $x \in \mathbb{R}$ lies in the range of $\mathbb{F}$, then x is approximated by a number $fl(x) \in \mathbb{F}$ such that

$$fl(x) = x\,(1+\delta), \qquad |\delta| \leq \mathbf{u}\,,$$

where $\mathbf{u}$ is the **unit roundoff of the computer**.

- The input data x of any algorithm are rounded, i.e., approximated by nearby numbers such that

$$\frac{|x - fl(x)|}{|x|} \leq O(\mathbf{u})\,,$$

- This imposes a more subtle **second accuracy limit**:
- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- Nearby in the relative sense $O(\mathbf{u})$.
- The algorithms that satisfy this are called **backward stable**.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward error analysis

- **The idea of backward error analysis is to attach the errors to the input data of an algorithm**, instead to the output result.
- The first backward error analysis was performed by **Wallace Givens** (1910-1993) **in 1954**,
- but **James Wilkinson** (1919-1986) is widely considered as the founder of backward error analysis, since he developed backward error analyses of many important algorithms in Numerical Linear Algebra.
- Traditionally, it is said that, from the point of view of rounding errors, **"being BACKWARD STABLE is the best we can hope for an algorithm in Numerical Linear Algebra"**.
- **Many famous and useful algorithms are NOT backward stable.**
- Backward error analysis is often subtle and difficult.
- The best possible illustration of backward error analysis is via examples.

Backward errors in Gaussian Elimination for linear systems

Theorem (Wilkinson, 1961)

Let $A \in \mathbb{R}^{n\times n}$ *be* **any nonsingular matrix**, *let* $b \in \mathbb{R}^n$, *and let*

$$\boxed{\widehat{x}}$$

be the approximate solution of

$$\boxed{Ax = b}$$

computed by **Gaussian Elimination with partial pivoting in a computer with unit roundoff** $\mathbf{u}$.

Then

$$\boxed{(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),}$$

i.e., **the computed solution is the exact solution of a nearby linear system.**

Backward errors of Householder-QR for least squares problems

Theorem (Stewart, 1973)

Let $A \in \mathbb{R}^{m\times n}$, $m \geq n$, let $b \in \mathbb{R}^{m\times 1}$, and let $\widehat{x}$ *be the approximate solution of*

$$\min_x \|b - Ax\|_2$$

computed via **the QR factorization implemented with the Householder algorithm in a computer with unit-roundoff $\mathbf{u}$**.

Then, $\widehat{x}$ **is the exact solution of the least squares problem**

$$\min_x \|(b + \Delta b) \; - \; (A + \Delta A)x\|_2,$$

where

$$\frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u}) \quad \textit{and} \quad \frac{\|\Delta b\|_2}{\|b\|_2} \leq O(\mathbf{u})\,.$$

Theorem

Let $A \in \mathbb{R}^{n \times n}$ *and let*

$$\widehat{\lambda}_1 \,,\, \widehat{\lambda}_2 \,, \ldots , \widehat{\lambda}_n$$

be the approximate eigenvalues of A *computed via* **the Francis-QR algorithm in a computer with unit-roundoff** $\mathbf{u}$.

Then,

$$\widehat{\lambda}_1 \,,\, \widehat{\lambda}_2 \,, \ldots , \widehat{\lambda}_n$$

are the exact eigenvalues of

$$(A + \Delta A)\,, \quad \textit{where} \ \frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u})\,.$$

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:

    1. Multiplication of two matrices.
    2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
    3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:
  1. Multiplication of two matrices.
  2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
  3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:
  1. Multiplication of two matrices.
  2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
  3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:
  1. Multiplication of two matrices.
  2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
  3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:
  1. Multiplication of two matrices.
  2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
  3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

Some conclusions on backward error analysis

- There is no attempt of comparing the exact output with the computed output.
- In fact, the exact outputs do not appear in the statements of the theorems.
- **Most algorithms** in Numerical Linear Algebra **are NOT backward stable**.
- Backward stable algorithms should be considered GEMS of Numerical Analysis.
- Very famous and useful algorithms that are **NOT** backward stable are:
  1. Multiplication of two matrices.
  2. Krylov iterative methods: conjugate gradient, GMRES, Lanczos, Arnoldi,...
  3. Direct methods for solving Sylvester and Lyapunov matrix equations as the Bartels-Stewart algorithm...

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

Why standard algorithms are not accurate?

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

Why standard algorithms are not accurate?

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

Why standard algorithms are not accurate?

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

Why standard algorithms are not accurate?

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

Why standard algorithms are not accurate?

- **"The best that can be expected from an algorithm is to compute outputs that are exact for nearby inputs."**
- The algorithms that satisfy this are called **backward stable**,
- and are great, but
- if tiny perturbations of the inputs may produce huge variations of the outputs, then
- **backward stable algorithms may produce huge errors in the outputs**.
- Errors in the outputs are called **forward errors**,
- and to evaluate them in backward stable algorithms requires the use of **perturbation theory**.

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u})\, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \le O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u}) \, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \qquad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u}) \, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

From backward to forward errors (I): Linear Systems

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u}) \, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u}) \, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

From backward to forward errors (I): Linear Systems

The approximate solution $\widehat{x}$ of $Ax = b$ computed by **GEPP** satisfies

$$(A + \Delta A)\widehat{x} = b, \quad \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq O(\mathbf{u}),$$

Bounding the difference between the **exact solution**, x, and the **computed solution**, $\widehat{x}$, becomes a mathematical problem of **perturbation theory**.

Theorem (Wilkinson, 1963)

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \|A\|_\infty \, \|A^{-1}\|_\infty \, \frac{\|\Delta A\|_\infty}{\|A\|_\infty} \lesssim O(\mathbf{u}) \, \kappa_\infty(A)$$

Definition (The (very famous!!!) condition number of a matrix)

$$\kappa_\infty(A) := \|A\|_\infty \, \|A^{-1}\|_\infty$$

Several other norms are used for condition numbers.

From backward to forward errors (II): Least Squares Problems

The approximate solution $\widehat{x}$ of $\min_z \|b - Az\|_2$ computed by **Householder-QR is the exact solution of**

$$\min_z \|(b + \Delta b) \; - \; (A + \Delta A) z\|_2,$$

$$\frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u}) \quad \text{and} \quad \frac{\|\Delta b\|_2}{\|b\|_2} \leq O(\mathbf{u}) \, .$$

Theorem (Wedin, 1973)

If x is the exact solution of $\min_z \|b - Az\|_2$, then

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim 2\, \kappa_2(A) \frac{\|\Delta A\|_2}{\|A\|_2} + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} \frac{\|\Delta b\|_2}{\|b\|_2} + \kappa_2(A)^2 \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \frac{\|\Delta A\|_2}{\|A\|_2}$$

Theorem (Forward errors for least squares problems)

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim O(\mathbf{u}) \left(2\, \kappa_2(A) + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} + \kappa_2(A)^2 \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \right)$$

From backward to forward errors (II): Least Squares Problems

The approximate solution $\widehat{x}$ of $\min_z \|b - Az\|_2$ computed by **Householder -QR** **is the exact solution of**

$$\min_z \|(b + \Delta b) \; - \; (A + \Delta A)z\|_2,$$

$$\frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u}) \quad \text{and} \quad \frac{\|\Delta b\|_2}{\|b\|_2} \leq O(\mathbf{u}) \, .$$

Theorem (Wedin, 1973)

If x is the exact solution of $\min_z \|b - Az\|_2$, then

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim 2\, \kappa_2(A) \, \frac{\|\Delta A\|_2}{\|A\|_2} + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} \, \frac{\|\Delta b\|_2}{\|b\|_2} + \kappa_2(A)^2 \, \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \, \frac{\|\Delta A\|_2}{\|A\|_2}$$

Theorem (Forward errors for least squares problems)

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim O(\mathbf{u}) \left(2\, \kappa_2(A) + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} + \kappa_2(A)^2 \, \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \right)$$

From backward to forward errors (II): Least Squares Problems

The approximate solution $\widehat{x}$ of $\min_z \|b - Az\|_2$ computed by **Householder-QR** **is the exact solution of**

$$\min_z \|(b + \Delta b) - (A + \Delta A) z\|_2,$$

$$\frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u}) \quad \text{and} \quad \frac{\|\Delta b\|_2}{\|b\|_2} \leq O(\mathbf{u}) \,.$$

Theorem (Wedin, 1973)

If x is the exact solution of $\min_z \|b - Az\|_2$, then

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim 2\, \kappa_2(A) \frac{\|\Delta A\|_2}{\|A\|_2} + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} \frac{\|\Delta b\|_2}{\|b\|_2} + \kappa_2(A)^2 \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \frac{\|\Delta A\|_2}{\|A\|_2}$$

Theorem (Forward errors for least squares problems)

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \lesssim O(\mathbf{u}) \left(2\, \kappa_2(A) + \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x\|_2} + \kappa_2(A)^2 \frac{\|Ax - b\|_2}{\|A\|_2 \, \|x\|_2} \right)$$

From backward to forward errors (III): Eigenvalues

The approximate eigenvalues $\widehat{\lambda}_1 \geq \cdots \geq \widehat{\lambda}_n$ of $A = A^T \in \mathbb{R}^{n \times n}$ computed by the **Francis-QR algorithm** **are the exact eigenvalues of**

$$(A + \Delta A)\,, \quad \text{where} \quad \frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u})\,.$$

Theorem (Weyl, 1912)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i |\widehat{\lambda}_i - \lambda_i| \leq \|\Delta A\|_2$$

Theorem (Forward errors)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i \left| \frac{\widehat{\lambda}_i - \lambda_i}{\lambda_i} \right| \leq O(\mathbf{u})\, \kappa(A)$$

From backward to forward errors (III): Eigenvalues

The approximate eigenvalues $\widehat{\lambda}_1 \geq \cdots \geq \widehat{\lambda}_n$ of $A = A^T \in \mathbb{R}^{n\times n}$ computed by the **Francis-QR algorithm** **are the exact eigenvalues of**

$$(A + \Delta A)\,, \quad \text{where} \quad \frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u})\,.$$

Theorem (Weyl, 1912)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i |\widehat{\lambda}_i - \lambda_i| \leq \|\Delta A\|_2$$

Theorem (Forward errors)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i \left| \frac{\widehat{\lambda}_i - \lambda_i}{\lambda_i} \right| \leq O(\mathbf{u})\, \kappa(A)$$

From backward to forward errors (III): Eigenvalues

The approximate eigenvalues $\widehat{\lambda}_1 \geq \cdots \geq \widehat{\lambda}_n$ of $A = A^T \in \mathbb{R}^{n\times n}$ computed by the **Francis-QR algorithm** **are the exact eigenvalues of**

$$(A + \Delta A)\,, \quad \text{where } \frac{\|\Delta A\|_2}{\|A\|_2} \leq O(\mathbf{u})\,.$$

Theorem (Weyl, 1912)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i |\widehat{\lambda}_i - \lambda_i| \leq \|\Delta A\|_2$$

Theorem (Forward errors)

If $\lambda_1 \geq \cdots \geq \lambda_n$ are the exact eigenvalues of A, then

$$\max_i \left| \frac{\widehat{\lambda}_i - \lambda_i}{\lambda_i} \right| \leq O(\mathbf{u})\,\kappa(A)$$

Conclusion: Backward stable algorithms are NOT accurate

Key Result (Backward stable algorithms are not accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n \times n}$ for different purposes satisfy

$$\textbf{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Therefore, **backward stable algorithms produce huge errors for ill-conditioned matrices**, i.e., for matrices such that

$$\kappa(A) := \|A\|\,\|A^{-1}\| \gg 1$$

Conclusion: Backward stable algorithms are NOT accurate

Key Result (Backward stable algorithms are not accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n\times n}$ for different purposes satisfy

$$\textbf{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Therefore, **backward stable algorithms produce huge errors for ill-conditioned matrices**, i.e., for matrices such that

$$\kappa(A) := \|A\|\,\|A^{-1}\| \gg 1$$

Famous example: eigenvalues of Hilbert matrix (I)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \leq i,j \leq 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0$.
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.77970862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$

- **Extremely ill-conditioned matrices arise often in practice: Vandermonde, Cauchy, scaled-matrices...**
- **We need to do something MUCH better!!**

Famous example: eigenvalues of Hilbert matrix (I)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \le i,j \le 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0.$
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.779700862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$

- **Extremely ill-conditioned matrices arise often in practice: Vandermonde, Cauchy, scaled-matrices...**
- **We need to do something MUCH better!!**

Famous example: eigenvalues of Hilbert matrix (I)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \leq i,j \leq 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0.$
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.779700862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$

- Extremely ill-conditioned matrices arise often in practice: Vandermonde, Cauchy, scaled-matrices...
- We need to do something MUCH better!!

Famous example: eigenvalues of Hilbert matrix (I)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \leq i,j \leq 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0.$
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.779700862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$

- **Extremely ill-conditioned matrices arise often in practice: Vandermonde, Cauchy, scaled-matrices...**
- **We need to do something MUCH better!!**

Famous example: eigenvalues of Hilbert matrix (I)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \leq i,j \leq 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0.$
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.779700862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$

- **Extremely ill-conditioned matrices arise often in practice: Vandermonde, Cauchy, scaled-matrices...**
- **We need to do something MUCH better!!**

Famous example: eigenvalues of Hilbert matrix (II)

$$H = [h_{ij}]; \qquad h_{ij} := \frac{1}{i+j-1}, \qquad 1 \leq i,j \leq 100$$

- $\lambda_1 > \lambda_2 > \cdots > \lambda_{100} > 0$.
- $\kappa(H) \approx 3.8 \cdot 10^{150}$

	λ_{100}
EXACT	$5.779700862834802 \cdot 10^{-151}$
MATLAB (eig)	$-1.216072660266760 \cdot 10^{-19}$
Jacobi	$-2.488943645649488 \cdot 10^{-17}$
Implicit Jacobi	$5.779700862834813 \cdot 10^{-151}$

Outline

Key Result (Backward stable algorithms are NOT accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n\times n}$ for different purposes satisfy

$$\textbf{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Definition ("Highly accurate algorithms" or "accurate algorithms")

These are algorithms that provide

$$\textbf{Relative errors} \leq O(\mathbf{u})$$

- **for certain particular classes of structured matrices**
- with roughly the same computational cost of $O(n^3)$ operations than standard algorithms.

Fundamental Fact on Accurate Algorithms

They are valid only for particular classes of structured matrices.

Key Result (Backward stable algorithms are NOT accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n\times n}$ for different purposes satisfy

$$\text{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Definition ("Highly accurate algorithms" or "accurate algorithms")

These are algorithms that provide

$$\text{Relative errors} \leq O(\mathbf{u})$$

- for certain particular classes of structured matrices
- with roughly the same computational cost of $O(n^3)$ operations than standard algorithms.

Fundamental Fact on Accurate Algorithms

They are valid only for particular classes of structured matrices.

Highly accurate algorithms

Key Result (Backward stable algorithms are NOT accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n \times n}$ for different purposes satisfy

$$\textbf{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Definition ("Highly accurate algorithms" or "accurate algorithms")

These are algorithms that provide

$$\textbf{Relative errors} \leq O(\mathbf{u})$$

- **for certain particular classes of structured matrices**
- with roughly the same computational cost of $O(n^3)$ operations than standard algorithms.

Fundamental Fact on Accurate Algorithms

They are valid only for particular classes of structured matrices.

Key Result (Backward stable algorithms are NOT accurate)

Backward stable algorithms applied to a matrix $A \in \mathbb{R}^{n\times n}$ for different purposes satisfy

$$\textbf{Relative errors} \geq O(\mathbf{u})\ \kappa(A)$$

Definition ("Highly accurate algorithms" or "accurate algorithms")

These are algorithms that provide

$$\textbf{Relative errors} \leq O(\mathbf{u})$$

- **for certain particular classes of structured matrices**
- with roughly the same computational cost of $O(n^3)$ operations than standard algorithms.

Fundamental Fact on Accurate Algorithms

They are valid only for particular classes of structured matrices.

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

General comments on accurate algorithms

- Some of them exist since the first days of Numerical Linear Algebra: Powell and Reid (1969), Björck and Pereyra (1970),...
- Intensive development and rigorous error analyses of these algorithms start in late 1980's-early 1990's: Barlow, Higham, Demmel, Kahan, Stewart, Veselić,...
- It continues in the 1990's: Dhillon, Drmač, Eisenstat, Fernando, Hari, Ipsen, Gu, Li, Parlett, Slapničar,...
- and in the 2000-10's: Alonso, Barreras, Boros, Castro-González, Ceballos, Delgado, D., Kailath, Koev, Marco, Martínez, Molera, Moro, Olshesky, Peña, Peláez, Ye,...
- **New structured results on matrix perturbation theory have been needed to perform the error analyses.**
- Many different structured algorithms, many different error analyses,....
- **I will focus in the rest of the talk on an approach that unifies many results on accurate algorithms and is based on...**

Outline

Summary of the rest of the talk (I)

- **Given the factors** of a rank revealing decomposition (RRD)

$$A = XDY,$$

where X and Y are well-conditioned, and D is diagonal and non-singular (and, so, it inherits the potential ill-conditioning of A).

- We present briefly **ACCURATE** algorithms developed by
  1. Demmel, Gu, Eisenstat, Slapničar, Veselić, and Drmač, Lin. Alg. Appl., 1999.
  2. D., Koev, and Molera, Numer. Math., 2009.
  3. D. and Molera, IMA J. Numer. Anal., 2012.
  4. Castro-González, Ceballos, D., Molera, SIAM J. Matrix Anal. Appl., 2013.

Summary of the rest of the talk (I)

- **Given the factors** of a rank revealing decomposition (RRD)

$$A = XDY,$$

where X and Y are well-conditioned, and D is diagonal and non-singular (and, so, it inherits the potential ill-conditioning of A).

- We present briefly **ACCURATE** algorithms developed by
    1. Demmel, Gu, Eisenstat, Slapničar, Veselić, and Drmač, Lin. Alg. Appl., 1999.
    2. D., Koev, and Molera, Numer. Math., 2009.
    3. D. and Molera, IMA J. Numer. Anal., 2012.
    4. Castro-González, Ceballos, D., Molera, SIAM J. Matrix Anal. Appl., 2013.

Summary of the rest of the talk (II)

- These **ACCURATE** algorithms allow us to compute (in the same order)
  - Singular Value Decomposition (SVD) of $A = XDY$,
  - Eigenvalues and eigenvectors of symmetric $A = XDX^T$,
  - Solution of linear system $(XDY)x = b$,
  - Solution of least squares problem $\min_x \|b - (XDY)x\|_2$,
- with

$$\textbf{Relative Errors} \leq O(\mathbf{u})$$

  independently of $\kappa(A)$,

- but, more precisely, with

$$\textbf{Relative Errors} \leq O(\kappa\, \mathbf{u}),$$

  where κ is a relevant condition number for each problem with respect perturbations of the factors.

- κ is almost always of order $O(1)$ and

$$\kappa \ll \kappa(A) = \|A\|\, \|A^{-1}\|$$

Summary of the rest of the talk (II)

- These **ACCURATE** algorithms allow us to compute (in the same order)
  - Singular Value Decomposition (SVD) of $A = XDY$,
  - Eigenvalues and eigenvectors of symmetric $A = XDX^T$,
  - Solution of linear system $(XDY)x = b$,
  - Solution of least squares problem $\min_x \|b - (XDY)x\|_2$,
- with

$$\textbf{Relative Errors} \leq O(\mathbf{u})$$

independently of $\kappa(A)$,

- but, more precisely, with

$$\textbf{Relative Errors} \leq O(\kappa\, \mathbf{u}),$$

where κ is a relevant condition number for each problem with respect perturbations of the factors.

- κ is almost always of order $O(1)$ and

$$\kappa \ll \kappa(A) = \|A\|\,\|A^{-1}\|$$

Summary of the rest of the talk (II)

- These **ACCURATE** algorithms allow us to compute (in the same order)
  - Singular Value Decomposition (SVD) of $A = XDY$,
  - Eigenvalues and eigenvectors of symmetric $A = XDX^T$,
  - Solution of linear system $(XDY)x = b$,
  - Solution of least squares problem $\min_x \|b - (XDY)x\|_2$,
- with

$$\textbf{Relative Errors} \leq O(\mathbf{u})$$

independently of $\kappa(A)$,

- but, more precisely, with

$$\textbf{Relative Errors} \leq O(\kappa\, \mathbf{u}),$$

where κ is a relevant condition number for each problem with respect perturbations of the factors.

- κ is almost always of order $O(1)$ and

$$\kappa \ll \kappa(A) = \|A\| \, \|A^{-1}\|$$

Summary of the rest of the talk (II)

- These **ACCURATE** algorithms allow us to compute (in the same order)
  - Singular Value Decomposition (SVD) of $A = XDY$,
  - Eigenvalues and eigenvectors of symmetric $A = XDX^T$,
  - Solution of linear system $(XDY)x = b$,
  - Solution of least squares problem $\min_x \|b - (XDY)x\|_2$,
- with

$$\textbf{Relative Errors} \leq O(\mathbf{u})$$

independently of $\kappa(A)$,

- but, more precisely, with

$$\textbf{Relative Errors} \leq O(\kappa\, \mathbf{u}),$$

where κ is a relevant condition number for each problem with respect perturbations of the factors.

- κ is almost always of order $O(1)$ and

$$\kappa \ll \kappa(A) = \|A\| \, \|A^{-1}\|$$

Summary of the rest of the talk (III)

- **Most important NEW message of this talk:** From a RRD $A = XDY$ of a matrix, that may be arbitrarily ill-conditioned, it is possible to solve (almost) all basic problems of Numerical Linear Algebra with relative errors of $O(\mathbf{u})$ and with computational costs of the same order as traditional algorithms.
- Only the nonsymmetric eigenvalue problem is excluded from this approach.
- Therefore, for those classes of matrices for which RRDs can be accurately computed, it is possible to solve accurately (almost) all basic problems of Numerical Linear Algebra, independently of the magnitude of the traditional condition number of the matrix.
- **Key idea: Algorithms for RRDs never form the matrix.** They **work directly on the factors** and deal with the ill-conditioned diagonal factor D in a special and highly accurate way.

Summary of the rest of the talk (III)

- **Most important NEW message of this talk:** From a RRD $A = XDY$ of a matrix, that may be arbitrarily ill-conditioned, it is possible to solve (almost) all basic problems of Numerical Linear Algebra with relative errors of $O(\mathbf{u})$ and with computational costs of the same order as traditional algorithms.
- Only the nonsymmetric eigenvalue problem is excluded from this approach.
- Therefore, for those classes of matrices for which RRDs can be accurately computed, it is possible to solve accurately (almost) all basic problems of Numerical Linear Algebra, independently of the magnitude of the traditional condition number of the matrix.
- **Key idea: Algorithms for RRDs never form the matrix**. They **work directly on the factors** and deal with the ill-conditioned diagonal factor D in a special and highly accurate way.

Summary of the rest of the talk (III)

- **Most important NEW message of this talk:** From a RRD $A = XDY$ of a matrix, that may be arbitrarily ill-conditioned, it is possible to solve (almost) all basic problems of Numerical Linear Algebra with relative errors of $O(\mathbf{u})$ and with computational costs of the same order as traditional algorithms.
- Only the nonsymmetric eigenvalue problem is excluded from this approach.
- Therefore, for those classes of matrices for which RRDs can be accurately computed, it is possible to solve accurately (almost) all basic problems of Numerical Linear Algebra, independently of the magnitude of the traditional condition number of the matrix.
- **Key idea: Algorithms for RRDs never form the matrix.** They **work directly on the factors** and deal with the ill-conditioned diagonal factor D in a special and highly accurate way.

Summary of the rest of the talk (III)

- **Most important NEW message of this talk:** From a RRD $A = XDY$ of a matrix, that may be arbitrarily ill-conditioned, it is possible to solve (almost) all basic problems of Numerical Linear Algebra with relative errors of $O(\mathbf{u})$ and with computational costs of the same order as traditional algorithms.
- Only the nonsymmetric eigenvalue problem is excluded from this approach.
- Therefore, for those classes of matrices for which RRDs can be accurately computed, it is possible to solve accurately (almost) all basic problems of Numerical Linear Algebra, independently of the magnitude of the traditional condition number of the matrix.
- **Key idea: Algorithms for RRDs never form the matrix**. They **work directly on the factors** and deal with the ill-conditioned diagonal factor D in a special and highly accurate way.

Definition of RRD

Definition (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

An RRD of $A \in \mathbb{R}^{m\times n}$ is a factorization

$$A = XDY,$$

where $X \in \mathbb{R}^{m\times r}$, $D = \mathrm{diag}(d_1, d_2, \ldots, d_r) \in \mathbb{R}^{r\times r}$ is nonsingular, and $Y \in \mathbb{R}^{r\times n}$ are such that

- $\mathrm{rank}\, A = \mathrm{rank}\, X = \mathrm{rank}\, D = \mathrm{rank}\, Y = r$, and
- X and Y are well conditioned.

$$A = XDY = \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{bmatrix} \begin{bmatrix} \times & \\ & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

Definition (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

An RRD of $A \in \mathbb{R}^{m\times n}$ is a factorization

$$A = XDY,$$

where $X \in \mathbb{R}^{m\times r}$, $D = \mathrm{diag}(d_1, d_2, \ldots, d_r) \in \mathbb{R}^{r\times r}$ is nonsingular, and $Y \in \mathbb{R}^{r\times n}$ are such that

- $\mathrm{rank}\, A = \mathrm{rank}\, X = \mathrm{rank}\, D = \mathrm{rank}\, Y = r$, and
- X and Y are well conditioned.

$$A = XDY = \begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \\ \times & \times \end{bmatrix} \begin{bmatrix} \times & \\ & \times \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

Definition of accurate RRD

Definition (Demmel et al, LAA, 1999)

Let $\widehat{X} \in \mathbb{R}^{m\times r}$, $\widehat{D} = \mathrm{diag}(\widehat{d}_1, \widehat{d}_2, \ldots, \widehat{d}_r) \in \mathbb{R}^{r\times r}$, and $\widehat{Y} \in \mathbb{R}^{r\times n}$ be the factors of an RRD $A = XDY$ computed by a certain algorithm. We say that $\widehat{X}\widehat{D}\widehat{Y}$ has been accurately computed if

$$\frac{\|\widehat{X} - X\|_2}{\|X\|_2} = O(\mathbf{u}), \quad \frac{\|\widehat{Y} - Y\|_2}{\|Y\|_2} = O(\mathbf{u}), \quad \text{and}$$

$$\frac{|\widehat{d}_i - d_i|}{|d_i|} = O(\mathbf{u}), \quad i = 1 : r.$$

- This is the accuracy that we need to apply the algorithms of this talk to $\widehat{X}\widehat{D}\widehat{Y}$ and to perform accurate Numerical Linear Algebra on A.
- This accuracy can be obtained only for special types of matrices through highly structured implementations of Gaussian elimination with complete pivoting (GECP) (and for one class via QRCP).

Definition of accurate RRD

Definition (Demmel et al, LAA, 1999)

Let $\widehat{X} \in \mathbb{R}^{m \times r}$, $\widehat{D} = \mathrm{diag}(\widehat{d}_1, \widehat{d}_2, \ldots, \widehat{d}_r) \in \mathbb{R}^{r \times r}$, and $\widehat{Y} \in \mathbb{R}^{r \times n}$ be the factors of an RRD $A = XDY$ computed by a certain algorithm. We say that $\widehat{X}\widehat{D}\widehat{Y}$ has been accurately computed if

$$\frac{\|\widehat{X} - X\|_2}{\|X\|_2} = O(\mathbf{u}), \quad \frac{\|\widehat{Y} - Y\|_2}{\|Y\|_2} = O(\mathbf{u}), \quad \text{and}$$

$$\frac{|\widehat{d}_i - d_i|}{|d_i|} = O(\mathbf{u}), \quad i = 1:r.$$

- This is the accuracy that we need to apply the algorithms of this talk to $\widehat{X}\widehat{D}\widehat{Y}$ and to perform accurate Numerical Linear Algebra on A.
- This accuracy can be obtained only for special types of matrices through highly structured implementations of Gaussian elimination with complete pivoting (GECP) (and for one class via QRCP).

Definition of accurate RRD

Definition (Demmel et al, LAA, 1999)

Let $\widehat{X} \in \mathbb{R}^{m\times r}$, $\widehat{D} = \mathrm{diag}(\widehat{d}_1, \widehat{d}_2, \ldots, \widehat{d}_r) \in \mathbb{R}^{r\times r}$, and $\widehat{Y} \in \mathbb{R}^{r\times n}$ be the factors of an RRD $A = XDY$ computed by a certain algorithm. We say that $\widehat{X}\widehat{D}\widehat{Y}$ has been accurately computed if

$$\frac{\|\widehat{X} - X\|_2}{\|X\|_2} = O(\mathbf{u}), \quad \frac{\|\widehat{Y} - Y\|_2}{\|Y\|_2} = O(\mathbf{u}), \quad \text{and}$$

$$\frac{|\widehat{d}_i - d_i|}{|d_i|} = O(\mathbf{u}), \quad i = 1 : r.$$

- This is the accuracy that we need to apply the algorithms of this talk to $\widehat{X}\widehat{D}\widehat{Y}$ and to perform accurate Numerical Linear Algebra on A.
- This accuracy can be obtained only for special types of matrices through highly structured implementations of Gaussian elimination with complete pivoting (GECP) (and for one class via QRCP).

Matrices for which accurate RRDs can be computed

- Cauchy, Scaled-Cauchy, Vandermonde (DFT + GECP). [Demmel]
- Diagonally Dominant M-Matrices. [Demmel and Koev, Peña]
- Polynomial Vandermonde. [Demmel and Koev]
- Well Scalable Symmetric Positive Definite. [Demmel and Veselić]
- Some well Scalable Symmetric Indefinite. [Slapničar and Veselić]
- Scaled Diagonally Dominant. [Barlow and Demmel]
- Acyclic Matrices (include bidiagonal). [Demmel and Gragg]
- Diagonally Dominant. [Ye, D. and Koev]
- Totally Nonnegative. [D. and Koev]
- DSTU. [Demmel]
- Graded Matrices. [Demmel et al.] [Higham]
- Symmetric versions. [D. and Koev] [D., Molera, Ceballos] [Peláez and Moro]
- ....

Outline

Algorithm (D. & Molera, IMA J. Numer. Anal., 2012)

- Input: $A \in \mathbb{R}^{n\times n}$ (nonsingular), $b \in \mathbb{R}^n$
- Output: x solution of $Ax = b$

1. Compute an accurate RRD of $A = XDY$
2. Solve the three systems

$$
\begin{array}{ccc}
X\,s = b & \longrightarrow & s \\
D\,w = s & \longrightarrow & w \\
Y\,x = w & \longrightarrow & x
\end{array}
$$

- $X\,s = b$ and $Y\,x = w$ are solved by any backward stable method.
- $w_i = s_i/d_{ii}, \quad i = 1:n.$
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(n^3)$ flops.

The algorithm for Linear Systems

Algorithm (D. & Molera, IMA J. Numer. Anal., 2012)

- Input: $A \in \mathbb{R}^{n\times n}$ (nonsingular), $b \in \mathbb{R}^n$
- Output: x solution of $Ax = b$

1. Compute an accurate RRD of $A = XDY$
2. Solve the three systems

$$\begin{array}{rcl} X\, s = b & \longrightarrow & s \\ D\, w = s & \longrightarrow & w \\ Y\, x = w & \longrightarrow & x \end{array}$$

- $X\, s = b$ and $Y\, x = w$ are solved by any backward stable method.
- $w_i = s_i / d_{ii}, \quad i = 1 : n.$
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(n^3)$ flops.

The algorithm for Linear Systems

Algorithm (D. & Molera, IMA J. Numer. Anal., 2012)

- `Input:` $A \in \mathbb{R}^{n\times n}$ `(nonsingular),` $b \in \mathbb{R}^n$
- `Output:` x `solution of` $Ax = b$

1. `Compute an accurate RRD of` $A = XDY$
2. `Solve the three systems`

$$\begin{array}{ccc} X\,s = b & \longrightarrow & s \\ D\,w = s & \longrightarrow & w \\ Y\,x = w & \longrightarrow & x \end{array}$$

- $X\,s = b$ and $Y\,x = w$ are solved by any backward stable method.
- $w_i = s_i/d_{ii}, \quad i = 1:n$.
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(n^3)$ flops.

The algorithm for Linear Systems

Algorithm (D. & Molera, IMA J. Numer. Anal., 2012)

- Input: $A \in \mathbb{R}^{n\times n}$ (nonsingular), $b \in \mathbb{R}^n$
- Output: x solution of $Ax = b$

1. Compute an accurate RRD of $A = XDY$
2. Solve the three systems

$$\begin{array}{ccc} X\,s = b & \longrightarrow & s \\ D\,w = s & \longrightarrow & w \\ Y\,x = w & \longrightarrow & x \end{array}$$

- $X\,s = b$ and $Y\,x = w$ are solved by any backward stable method.
- $w_i = s_i / d_{ii}, \quad i = 1 : n$.
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(n^3)$ flops.

Algorithm (D. & Molera, IMA J. Numer. Anal., 2012)

- `Input:` $A \in \mathbb{R}^{n\times n}$ `(nonsingular),` $b \in \mathbb{R}^n$
- `Output:` x `solution of` $A\,x = b$

1. `Compute an accurate RRD of` $A = X D Y$
2. `Solve the three systems`

$$\begin{array}{rcc} X\,s = b & \longrightarrow & s \\ D\,w = s & \longrightarrow & w \\ Y\,x = w & \longrightarrow & x \end{array}$$

- $X\,s = b$ and $Y\,x = w$ are solved by any backward stable method.
- $w_i = s_i / d_{ii}, \quad i = 1:n$.
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(n^3)$ flops.

Theorem (D. & Molera, IMA J. Numer. Anal., 2012)

If $\widehat{x}$ is the solution of $Ax = b$ computed by the algorithm in previous slide ($A = XDY$), then

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \leq O(\mathbf{u}) \max\{\kappa_2(X)\,,\,\kappa_2(Y)\} \frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2}$$

To be compared with

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \leq O(\mathbf{u})\,\kappa_2(A) \quad ,$$

that holds for traditional algorithms as GEPP, GECP, QR,...

Theorem (D. & Molera, IMA J. Numer. Anal., 2012)

If $\widehat{x}$ is the solution of $Ax = b$ computed by the algorithm in previous slide ($A = XDY$), then

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \leq O(\mathbf{u}) \max\{\kappa_2(X)\,,\,\kappa_2(Y)\}\, \frac{\|A^{-1}\|_2 \|b\|_2}{\|x\|_2}$$

To be compared with

$$\frac{\|\widehat{x} - x\|_2}{\|x\|_2} \leq O(\mathbf{u})\, \kappa_2(A) \quad ,$$

that holds for traditional algorithms as GEPP, GECP, QR,...

$\left(\|A^{-1}\|_2\|b\|_2 \,/\, \|x\|_2\right)$ is "almost always" a moderate number (Chan & Foulser, 1988)

- $\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \kappa_2(A)$, but even more **"almost always"**
- $\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \ll \kappa_2(A)$, if A very ill-conditioned because,
- $\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \dfrac{1}{\cos\theta(u_n, b)}$,

  where u_n left-singular vector of A of smallest singular value.

Example in MATLAB:

```
>> V = vander(randn(20,1)); b=randn(20,1);
>> cond(V)= 7.1021e+11
>> x = V\b;
>> norm(inv(V))*norm(b)/norm(x) = 8.4317
```

$(\|A^{-1}\|_2\|b\|_2 \,/\, \|x\|_2)$ is "almost always" a moderate number (Chan & Foulser, 1988)

- $\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \kappa_2(A)$, but even more **"almost always"**
- $\boxed{\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \ll \kappa_2(A)}$, if A very ill-conditioned because,
- $\dfrac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \dfrac{1}{\cos\theta(u_n, b)}$,

  where u_n left-singular vector of A of smallest singular value.

Example in MATLAB:

```
>> V = vander(randn(20,1)); b=randn(20,1);
>> cond(V)= 7.1021e+11
>> x = V\b;
>> norm(inv(V))*norm(b)/norm(x) = 8.4317
```

$(\|A^{-1}\|_2\|b\|_2 \,/\, \|x\|_2)$ is "almost always" a moderate number (Chan & Foulser, 1988)

- $$\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \kappa_2(A),$$ but even more **"almost always"**
- $$\boxed{\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \ll \kappa_2(A)},$$ if A very ill-conditioned because,
- $$\boxed{\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \frac{1}{\cos\theta(u_n, b)}},$$
  where u_n left-singular vector of A of smallest singular value.

Example in MATLAB:

```
>> V = vander(randn(20,1)); b=randn(20,1);
>> cond(V)= 7.1021e+11
>> x = V\b;
>> norm(inv(V))*norm(b)/norm(x) = 8.4317
```

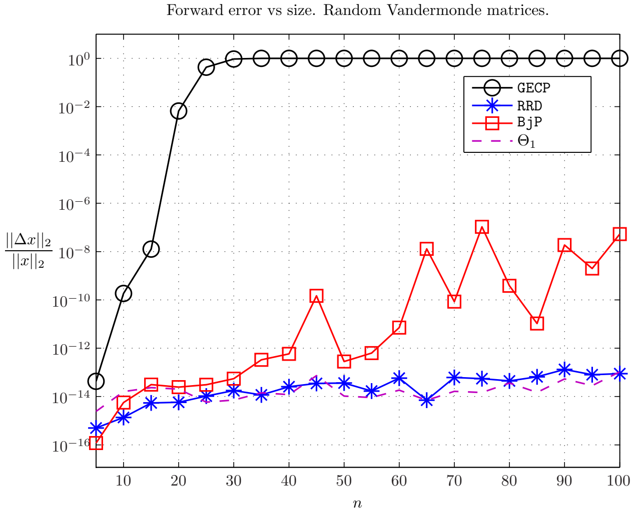
$(\|A^{-1}\|_2\|b\|_2 \,/\, \|x\|_2)$ is "almost always" a moderate number (Chan & Foulser, 1988)

- $$\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \kappa_2(A),$$ but even more **"almost always"**
- $$\boxed{\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \ll \kappa_2(A)},$$ if A very ill-conditioned because,
- $$\boxed{\frac{\|A^{-1}\|_2\|b\|_2}{\|x\|_2} \leq \frac{1}{\cos\theta(u_n, b)}},$$
  where u_n left-singular vector of A of smallest singular value.

Example in MATLAB:

```
>> V = vander(randn(20,1)); b=randn(20,1);
>> cond(V)= 7.1021e+11
>> x = V\b;
>> norm(inv(V))*norm(b)/norm(x) = 8.4317
```

Numerical tests for linear systems: Random Vandermonde Matrices. (computing RRD as Demmel, SIMAX, 1999)



Outline

The algorithm for Least Squares Problems

Algorithm (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

- Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
- Output: x_0 minimum 2-norm solution of $\min_{x\in\mathbb{R}^n} \|b - A\,x\|_2$

1. Compute an accurate RRD of $A = XDY$
2. Apply to XDY the following steps
   1. Compute the solution x_1 of $\min_{x\in\mathbb{R}^r} \|b - X\,x\|_2$ using Householder-QR.
   2. Solve the diagonal linear system, $D\,x_2 = x_1$.
   3. Compute the minimum 2-norm solution x_0 of the underdetermined linear system $Y x = x_2$ using Householder-QR on Y^*.

- $x_2(i) = x_1(i)/d_{ii}$, $\quad i = 1:r$.
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(m\,n^2)$ flops.

The algorithm for Least Squares Problems

Algorithm (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

- Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
- Output: x_0 minimum 2-norm solution of $\min_{x\in\mathbb{R}^n} \|b - A\,x\|_2$

1. Compute an accurate RRD of $A = XDY$
2. Apply to XDY the following steps
   1. Compute the solution x_1 of $\min_{x\in\mathbb{R}^r} \|b - X\,x\|_2$ using Householder-QR.
   2. Solve the diagonal linear system, $D\,x_2 = x_1$.
   3. Compute the minimum 2-norm solution x_0 of the underdetermined linear system $Y\,x = x_2$ using Householder-QR on Y^*.

- $x_2(i) = x_1(i)/d_{ii}, \quad i = 1 : r.$
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(m\,n^2)$ flops.

Algorithm (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

- Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$
- Output: x_0 minimum 2-norm solution of $\min_{x\in\mathbb{R}^n} \|b - A\,x\|_2$

1. Compute an accurate RRD of $A = XDY$
2. Apply to XDY the following steps
   1. Compute the solution x_1 of $\min_{x\in\mathbb{R}^r} \|b - X\,x\|_2$ using Householder-QR.
   2. Solve the diagonal linear system, $D\,x_2 = x_1$.
   3. Compute the minimum 2-norm solution x_0 of the underdetermined linear system $Y x = x_2$ using Householder-QR on Y^*.

- $x_2(i) = x_1(i)/d_{ii}, \quad i = 1 : r.$
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(m\,n^2)$ flops.

The algorithm for Least Squares Problems

Algorithm (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

- Input: $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$
- Output: x_0 minimum 2-norm solution of $\min_{x \in \mathbb{R}^n} \|b - A\,x\|_2$

1. Compute an accurate RRD of $A = XDY$
2. Apply to XDY the following steps
   1. Compute the solution x_1 of $\min_{x \in \mathbb{R}^r} \|b - X\,x\|_2$ using Householder-QR.
   2. Solve the diagonal linear system, $D\,x_2 = x_1$.
   3. Compute the minimum 2-norm solution x_0 of the underdetermined linear system $Y x = x_2$ using Householder-QR on Y^*.

- $x_2(i) = x_1(i)/d_{ii}, \quad i = 1 : r.$
- **Intuition:** the ill-conditioned linear system is solved very accurately.
- **Cost:** $O(m\,n^2)$ flops.

Theorem (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

If $\widehat{x}_0$ is the minimum 2-norm solution of $\min_{x \in \mathbb{R}^n} \|b - A\,x\|_2$ computed by the algorithm in previous slide ($A = XDY$) and x_0 is the exact one, then

$$\frac{\|\widehat{x}_0 - x_0\|_2}{\|x_0\|_2} \leq O(\mathbf{u}) \max\{\kappa_2(Y), \kappa_2(X)\} \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x_0\|_2},$$

$A^\dagger$ is the Moore-Penrose pseudo-inverse of A and $\kappa_2(Y) = \|Y\|_2 \|Y^\dagger\|_2$.

To be compared with

$$\frac{\|\widehat{x}_0 - x_0\|_2}{\|x_0\|_2} \leq O(\mathbf{u}) \left(\kappa_2(A) + \frac{\|A^\dagger\|_2 \|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2 \frac{\|b - Ax_0\|_2}{\|A\|_2 \, \|x_0\|_2} \right),$$

that holds for traditional algorithms as Householder-QR, SVD,...

Theorem (Castro, Ceballos, D., Molera, SIAM J. Matrix Anal., 2013)

If $\widehat{x}_0$ is the minimum 2-norm solution of $\min_{x\in\mathbb{R}^n} \|b - A\,x\|_2$ computed by the algorithm in previous slide ($A = XDY$) and x_0 is the exact one, then

$$\frac{\|\widehat{x}_0 - x_0\|_2}{\|x_0\|_2} \leq O(\mathbf{u}) \, \max\{\kappa_2(Y), \kappa_2(X)\} \, \frac{\|A^\dagger\|_2 \, \|b\|_2}{\|x_0\|_2} \, ,$$

$A^\dagger$ is the Moore-Penrose pseudo-inverse of A and $\kappa_2(Y) = \|Y\|_2 \|Y^\dagger\|_2$.

To be compared with

$$\frac{\|\widehat{x}_0 - x_0\|_2}{\|x_0\|_2} \leq O(\mathbf{u}) \left(\kappa_2(A) + \frac{\|A^\dagger\|_2 \|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2 \, \frac{\|b - Ax_0\|_2}{\|A\|_2 \, \|x_0\|_2} \right),$$

that holds for traditional algorithms as Householder-QR, SVD,...

Define

$$\kappa_{LS}(A,b) := \left(\kappa_2(A) + \frac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2\,\frac{\|b - Ax_0\|_2}{\|A\|_2\,\|x_0\|_2}\right),$$

- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \kappa_{LS}(A,b)$, but even more **"almost always"**
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \ll \kappa_{LS}(A,b)$, if A very ill-conditioned because,
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \dfrac{1}{\cos\theta(u_r,b)}$,

where u_r left-singular vector of A of smallest nonzero singular value.

$\left(\|A^\dagger\|_2\|b\|_2 \,/\, \|x_0\|_2\right)$ is "almost always" a moderate number

Define

$$\kappa_{LS}(A,b) := \left(\kappa_2(A) + \frac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2\,\frac{\|b-Ax_0\|_2}{\|A\|_2\,\|x_0\|_2}\right),$$

- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \kappa_{LS}(A,b),$ but even more **"almost always"**
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \ll \kappa_{LS}(A,b)$, if A very ill-conditioned because,
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \dfrac{1}{\cos\theta(u_r,b)}$,

where u_r left-singular vector of A of smallest nonzero singular value.

Define

$$\kappa_{LS}(A,b) := \left(\kappa_2(A) + \frac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2\,\frac{\|b - Ax_0\|_2}{\|A\|_2\,\|x_0\|_2}\right),$$

- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \kappa_{LS}(A,b)$, but even more **"almost always"**
- $\boxed{\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \ll \kappa_{LS}(A,b)}$, if A very ill-conditioned because,
- $\boxed{\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \dfrac{1}{\cos\theta(u_r,b)}}$,

where u_r left-singular vector of A of smallest nonzero singular value.

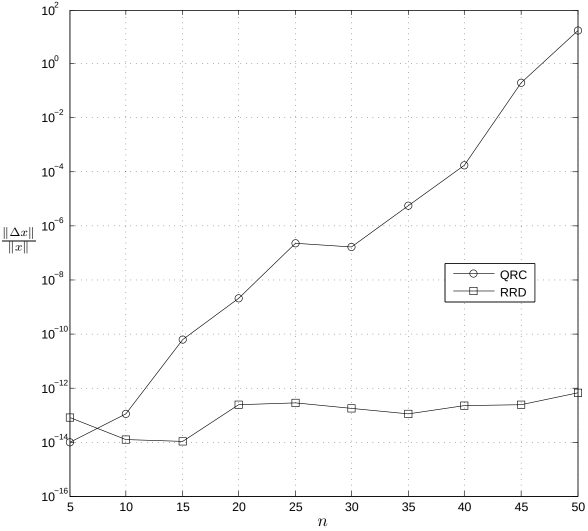
Define

$$\kappa_{LS}(A,b) := \left(\kappa_2(A) + \frac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} + \kappa_2(A)^2\,\frac{\|b - Ax_0\|_2}{\|A\|_2\,\|x_0\|_2}\right),$$

- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \kappa_{LS}(A,b)$, but even more **"almost always"**
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \ll \kappa_{LS}(A,b)$, if A very ill-conditioned because,
- $\dfrac{\|A^\dagger\|_2\|b\|_2}{\|x_0\|_2} \leq \dfrac{1}{\cos\theta(u_r,b)}$,

  where u_r left-singular vector of A of smallest nonzero singular value.

Numerical tests for Least Squares: $50 \times n$ graded matrices $S_1 B S_2$, with $\kappa_2(B) = 10$ and $\kappa_2(S_1) = \kappa_2(S_2) = 10^{(2:2:16)}$ (comp. RRD as Higham, 2000)



Outline

The algorithm for eigenvalues-vectors of symmetric matrices

Algorithm (IMPLICIT JACOBI. (D., Koev, Molera, Numer. Math., 2009))

- Input: $A = A^T \in \mathbb{R}^{n \times n}$
- Output: e-values, λ_i, and matrix of e-vectors, U, of A

1. Compute an accurate symmetric RRD of $A = XDX^T$
2. Apply **implicit Jacobi to** XDX^T, i.e.,

   $U = I_n$

   repeat

   for $i < j$

   compute a_{ii}, a_{ij}, a_{jj} of $A = XDX^T$

   compute Jacobi Rotation R s.t. $a_{ij} = 0$ by similarity

   $X = R^* X$

   $U = U R$

   endfor

   until convergence $\left(\frac{|a_{ij}|}{\sqrt{|a_{ii} a_{jj}|}} \leq \mathtt{tol} = O(\mathtt{u}) \quad \text{for all } i < j \right)$

   compute $\lambda_k = a_{kk}$ for $k = 1 : n$.

Comments on the IMPLICIT JACOBI algorithm

- The ill-conditioned matrix D **is never modified**.
- **Only the well-conditioned factor X is transformed** in the process.
- This is the reason why high relative accuracy is obtained.
- **Cost**: $O(n^3)$ flops.
- **Efficient implementation requires preconditioning** via QR factorization with column pivoting of $X\sqrt{|D|}$ (this was suggested by Drmač).

Comments on the IMPLICIT JACOBI algorithm

- The ill-conditioned matrix D **is never modified**.
- **Only the well-conditioned factor X is transformed** in the process.
- This is the reason why high relative accuracy is obtained.
- **Cost**: $O(n^3)$ flops.
- **Efficient implementation requires preconditioning** via QR factorization with column pivoting of $X\sqrt{|D|}$ (this was suggested by Drmač).

Comments on the IMPLICIT JACOBI algorithm

- The ill-conditioned matrix D **is never modified**.
- **Only the well-conditioned factor X is transformed** in the process.
- This is the reason why high relative accuracy is obtained.
- **Cost**: $O(n^3)$ flops.
- **Efficient implementation requires preconditioning** via QR factorization with column pivoting of $X\sqrt{|D|}$ (this was suggested by Drmač).

Comments on the IMPLICIT JACOBI algorithm

- The ill-conditioned matrix D **is never modified**.
- **Only the well-conditioned factor X is transformed** in the process.
- This is the reason why high relative accuracy is obtained.
- **Cost**: $O(n^3)$ flops.
- **Efficient implementation requires preconditioning** via QR factorization with column pivoting of $X\sqrt{|D|}$ (this was suggested by Drmač).

Comments on the IMPLICIT JACOBI algorithm

- The ill-conditioned matrix D **is never modified**.
- **Only the well-conditioned factor X is transformed** in the process.
- This is the reason why high relative accuracy is obtained.
- **Cost**: $O(n^3)$ flops.
- **Efficient implementation requires preconditioning** via QR factorization with column pivoting of $X\sqrt{|D|}$ (this was suggested by Drmač).

The errors in IMPLICIT JACOBI

Theorem (D., Koev, Molera, Numer. Math., 2009)

Let $\widehat{\lambda}_i$ be the eigenvalues of $A = XDX^T$ computed by the Implicit Jacobi Algorithm and λ_i the exact ones. Let $\widehat{v}_i$ and v_i be the corresponding eigenvectors. Then

$$\boxed{\frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq O(\mathbf{u})\,\kappa_2(X)} \quad \text{and} \quad \boxed{\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})\,\kappa(X)}{\min_{j\neq i} \left|\frac{\lambda_i - \lambda_j}{\lambda_i}\right|}} \quad \textit{for all} \quad i,$$

To be compared with

$$\boxed{\frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq O(\mathbf{u})\,\kappa_2(A)} \quad \text{and} \quad \boxed{\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})}{\frac{\min_{j\neq i}|\lambda_i - \lambda_j|}{\max_i |\lambda_i|}}} \quad \text{for all} \quad i,$$

that hold for traditional algorithms as QR, divide and conquer,...

The errors in IMPLICIT JACOBI

Theorem (D., Koev, Molera, Numer. Math., 2009)

Let $\widehat{\lambda}_i$ be the eigenvalues of $A = XDX^T$ computed by the Implicit Jacobi Algorithm and λ_i the exact ones. Let $\widehat{v}_i$ and v_i be the corresponding eigenvectors. Then

$$\boxed{\frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq O(\mathbf{u})\, \kappa_2(X)} \quad \text{and} \quad \boxed{\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})\, \kappa(X)}{\min_{j \neq i} \left| \frac{\lambda_i - \lambda_j}{\lambda_i} \right|}} \quad \textit{for all} \quad i,$$

To be compared with

$$\boxed{\frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq O(\mathbf{u})\, \kappa_2(A)} \quad \text{and} \quad \boxed{\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})}{\frac{\min_{j \neq i} |\lambda_i - \lambda_j|}{\max_i |\lambda_i|}}} \quad \text{for all} \quad i,$$

that hold for traditional algorithms as QR, divide and conquer,...

EXAMPLE: Symmetric INDEFINITE 100×100 Cauchy matrix A

$$a_{ij} = \frac{1}{x_i + x_j}, \quad \text{with} \quad \begin{cases} x_i = i - 0.5 \quad for \quad i = 1:99 \\ x_{100} = -99.5 \end{cases}$$

- $\kappa(A) = 3.5 \cdot 10^{147}$
- **Errors in RRD + Imp. Jacobi** compared to 200-decimal digits MATLAB's `eig` command

$$\max_i \frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} = 1.2 \cdot 10^{-13} \quad \text{and} \quad \max_i \|\widehat{v}_i - v_i\|_2 = 5.7 \cdot 10^{-14}$$

- **Errors in MATLAB's `eig` function**

$$\max_i \frac{|\widehat{\lambda}_i - \lambda_i|}{|\lambda_i|} = 1.84 \cdot 10^{132} \quad \text{and} \quad \max_i \|\widehat{v}_i - v_i\|_2 = 1.41$$

Outline

The algorithm for the Singular Value Decomposition (SVD)

Algorithm (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

- Input: $A \in \mathbb{R}^{m\times n}$
- Output: $U\Sigma V^T$ SVD of A

1. Compute an accurate RRD of $A = XDY$
2. Apply to XDY the following algorithm
    1. QR with column pivoting of $XD = QRP$ (so $A = QRPY$)
    2. Multiply to get $W = RPY$ (so $A = QW$)
    3. Compute SVD of $W = \bar{U}\Sigma V^*$ with **one-sided Jacobi** (so $A = Q\bar{U}\Sigma V^T$)
    4. Multiply $U = Q\bar{U}$ (so $A = U\Sigma V^T$)

- The one-sided Jacobi step can be performed with new fast and accurate Jacobi algorithm by Drmač and Veselić, SIMAX, 2008.
- **Cost:** $O(mn^2)$ flops.

The algorithm for the Singular Value Decomposition (SVD)

Algorithm (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

- `Input:` $A \in \mathbb{R}^{m \times n}$
- `Output:` $U \Sigma V^T$ `SVD of` A

1. `Compute an accurate RRD of` $A = XDY$
2. `Apply to` XDY `the following algorithm`
   1. `QR with column pivoting of` $XD = QRP$ `(so` $A = QRPY$`)`
   2. `Multiply to get` $W = RPY$ `(so` $A = QW$`)`
   3. `Compute SVD of` $W = \bar{U} \Sigma V^*$ `with` **one-sided Jacobi** `(so` $A = Q\bar{U}\Sigma V^T$`)`
   4. `Multiply` $U = Q\bar{U}$ `(so` $A = U\Sigma V^T$`)`

- The one-sided Jacobi step can be performed with new fast and accurate Jacobi algorithm by Drmač and Veselić, SIMAX, 2008.
- **Cost:** $O(mn^2)$ flops.

The algorithm for the Singular Value Decomposition (SVD)

Algorithm (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

- `Input:` $A \in \mathbb{R}^{m \times n}$
- `Output:` $U \Sigma V^T$ `SVD of` A

1. `Compute an accurate RRD of` $A = XDY$
2. `Apply to` XDY `the following algorithm`
    1. `QR with column pivoting of` $XD = QRP$ `(so` $A = QRPY$`)`
    2. `Multiply to get` $W = RPY$ `(so` $A = QW$`)`
    3. `Compute SVD of` $W = \bar{U} \Sigma V^*$ `with` **one-sided Jacobi** `(so` $A = Q\bar{U}\Sigma V^T$`)`
    4. `Multiply` $U = Q\bar{U}$ `(so` $A = U \Sigma V^T$`)`

- The one-sided Jacobi step can be performed with new fast and accurate Jacobi algorithm by Drmač and Veselić, SIMAX, 2008.
- **Cost:** $O(mn^2)$ flops.

Theorem (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

Let $\widehat{\sigma}_i$ be the singular values of $A = XDY$ computed by the algorithm in previous slide and σ_i the exact ones. Let $\widehat{u}_i$ and u_i be the corresponding left singular vectors and $\widehat{v}_i$ and v_i the right ones. Then

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(\mathbf{u}) \max\{\kappa_2(X), \kappa_2(Y)\} \quad \text{and}$$

$$\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u}) \max\{\kappa_2(X), \kappa_2(Y)\}}{\min_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \textit{for all} \quad i,$$

To be compared with

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(\mathbf{u}) \, \kappa_2(A) \quad \text{and} \quad \theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})}{\frac{\min_{j \neq i} |\sigma_i - \sigma_j|}{\max_i |\sigma_i|}} \quad \text{for all} \quad i,$$

that hold for traditional algorithms as QR, divide and conquer,...

The errors in accurate SVD

Theorem (Demmel, Gu, Eisenstat, Slapničar, Veselić, Drmač, LAA, 99)

Let $\widehat{\sigma}_i$ be the singular values of $A = XDY$ computed by the algorithm in previous slide and σ_i the exact ones. Let $\widehat{u}_i$ and u_i be the corresponding left singular vectors and $\widehat{v}_i$ and v_i the right ones. Then

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(\mathbf{u}) \max\{\kappa_2(X), \kappa_2(Y)\} \quad \text{and}$$

$$\theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u}) \max\{\kappa_2(X), \kappa_2(Y)\}}{\min_{j \neq i} \left| \frac{\sigma_i - \sigma_j}{\sigma_i} \right|} \quad \textit{for all} \quad i,$$

To be compared with

$$\frac{|\widehat{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq O(\mathbf{u})\, \kappa_2(A) \quad \text{and} \quad \theta(v_i, \widehat{v}_i) \leq \frac{O(\mathbf{u})}{\frac{\min_{j \neq i} |\sigma_i - \sigma_j|}{\max_i |\sigma_i|}} \quad \text{for all} \quad i,$$

that hold for traditional algorithms as QR, divide and conquer,...

Outline

Conclusions and open problems

- Rank-Revealing decompositions (RRDs) may be computed with high accuracy for many classes of structured matrices.
- This allows us to perform with high accuracy almost all basic tasks of Numerical Linear Algebra for these structured matrices.
- By contrast, standard algorithms may not produce a single digit of accuracy for these matrices.
- The only basic task that is not included in the RRD framework is the nonsymmetric eigenvalue problem.
- This problem will be the subject of future research.

Conclusions and open problems

- Rank-Revealing decompositions (RRDs) may be computed with high accuracy for many classes of structured matrices.
- This allows us to perform with high accuracy almost all basic tasks of Numerical Linear Algebra for these structured matrices.
- By contrast, standard algorithms may not produce a single digit of accuracy for these matrices.
- The only basic task that is not included in the RRD framework is the nonsymmetric eigenvalue problem.
- This problem will be the subject of future research.

- Rank-Revealing decompositions (RRDs) may be computed with high accuracy for many classes of structured matrices.
- This allows us to perform with high accuracy almost all basic tasks of Numerical Linear Algebra for these structured matrices.
- By contrast, standard algorithms may not produce a single digit of accuracy for these matrices.
- The only basic task that is not included in the RRD framework is the nonsymmetric eigenvalue problem.
- This problem will be the subject of future research.

- Rank-Revealing decompositions (RRDs) may be computed with high accuracy for many classes of structured matrices.
- This allows us to perform with high accuracy almost all basic tasks of Numerical Linear Algebra for these structured matrices.
- By contrast, standard algorithms may not produce a single digit of accuracy for these matrices.
- The only basic task that is not included in the RRD framework is the nonsymmetric eigenvalue problem.
- This problem will be the subject of future research.

Conclusions and open problems

- Rank-Revealing decompositions (RRDs) may be computed with high accuracy for many classes of structured matrices.
- This allows us to perform with high accuracy almost all basic tasks of Numerical Linear Algebra for these structured matrices.
- By contrast, standard algorithms may not produce a single digit of accuracy for these matrices.
- The only basic task that is not included in the RRD framework is the nonsymmetric eigenvalue problem.
- This problem will be the subject of future research.