A Summary of Methods of Solution for Linear Systems

Nate DeMaagd, Kurt O'Hearn

MTH 499-02

April 23, 2013

Outline

Outline

- Types of methods

Outline

- Types of methods
- Direct methods and examples

Outline

- Types of methods
- Direct methods and examples
- Indirect methods and examples

Recall: For the linear system $Ax = b$ with $A_{m \times n}$:

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Overview of Types of Methods for Solving Linear Systems

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result

Recall: For the linear system $Ax = b$ with $A_{m \times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
    - Execute a predetermined number of computations to produce a result
    - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods

Recall: For the linear system $Ax = b$ with $A_{m \times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods
  - Generate a sequence of intermediate results which (hopefully) produce the desired final result

Recall: For the linear system $Ax = b$ with $A_{m \times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods
  - Generate a sequence of intermediate results which (hopefully) produce the desired final result
  - Methods:

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods
  - Generate a sequence of intermediate results which (hopefully) produce the desired final result
  - Methods:
    - General: Richardson, Jacobi, Gauss-Seidel, SOR

Recall: For the linear system $Ax = b$ with $A_{m \times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods
  - Generate a sequence of intermediate results which (hopefully) produce the desired final result
  - Methods:
    - General: Richardson, Jacobi, Gauss-Seidel, SOR
    - Symmetric, positive definite: steepest descent, conjugate gradient

Recall: For the linear system $Ax = b$ with $A_{m\times n}$:

System Type	Possible Number of Solutions
Square ($m = n$)	None, Unique
Overdetermined ($m > n$)	None, Unique
Underdetermined ($m < n$)	None, Infinite

Types of Methods

- Direct methods
  - Execute a predetermined number of computations to produce a result
  - Methods: compute A^{-1}, transform A using factorizations/pivoting
- Indirect methods
  - Generate a sequence of intermediate results which (hopefully) produce the desired final result
  - Methods:
    - General: Richardson, Jacobi, Gauss-Seidel, SOR
    - Symmetric, positive definite: steepest descent, conjugate gradient

Note: methods can be general or exploit certain matrix characteristics

Compute A^{-1}

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Gaussian Elimination

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Gaussian Elimination

- Solve by reducing the coefficient matrix to triangular form using elementary row operations

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Gaussian Elimination

- Solve by reducing the coefficient matrix to triangular form using elementary row operations
- Result can then by easily solved using forward/backward substitution

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Gaussian Elimination

- Solve by reducing the coefficient matrix to triangular form using elementary row operations
- Result can then by easily solved using forward/backward substitution
- Possible row operations:
  - Interchange
  - Scaling
  - Addition

Compute A^{-1}

- Most likely too difficult but not always (e.g., diagonal)

Gaussian Elimination

- Solve by reducing the coefficient matrix to triangular form using elementary row operations
- Result can then by easily solved using forward/backward substitution
- Possible row operations:
  - Interchange
  - Scaling
  - Addition
- Recall: GE is equivalent to computing LU factorization

Pivoting Methods

Pivoting Methods

- Types of pivoting:
  - Partial/complete: pertains to the amount of the matrix examined to find where to pivot

Pivoting Methods

- Types of pivoting:
  - Partial/complete: pertains to the amount of the matrix examined to find where to pivot
    - Partial: analyze one row
    - Complete: analyze entire submatrix

Pivoting Methods

- Types of pivoting:
  - Partial/complete: pertains to the amount of the matrix examined to find where to pivot
    - Partial: analyze one row
    - Complete: analyze entire submatrix
  - Unscaled/scaled: pertains to how the potential pivot locations are compared

Pivoting Methods

- Types of pivoting:
  - Partial/complete: pertains to the amount of the matrix examined to find where to pivot
    - Partial: analyze one row
    - Complete: analyze entire submatrix
  - Unscaled/scaled: pertains to how the potential pivot locations are compared
    - Unscaled: compare absolute value of entries directly
    - Scaled: compare ratios of pivot entries to maximum entries in magnitude in each row

$$A_1 = \begin{bmatrix} 1 & 3 & 6 \\ 2 & 1 & 1 \\ 1 & 3 & 3 \end{bmatrix}$$

$$p = \begin{bmatrix} 1, & 2, & 3 \end{bmatrix}$$

```
function GE_PP(n, (a_ij), (p_i))
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            z ← a_{p_i k} / a_{p_k k}
            a_{p_i k} ← 0
            for j = k + 1 to n do
                a_{p_i j} ← a_{p_i j} − z a_{p_k j}
    return (a_ij)
```

$$\max(|1|, |2|, |1|) = 2$$

Direct Methods: Pivoting Examples

$$A_1 = \begin{bmatrix} 1 & 3 & 6 \\ 2 & 1 & 1 \\ 1 & 3 & 3 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

function GE_PP$(n, (a_{ij}), (p_i))$
  for $k = 1$ **to** $n - 1$ **do**
    for $i = k + 1$ **to** n **do**
      $z \leftarrow a_{p_i k} / a_{p_k k}$
      $a_{p_i k} \leftarrow 0$
      for $j = k + 1$ **to** n **do**
        $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
  return (a_{ij})

$$\max(|1|, |2|, |1|) = 2$$

$$R_1 \leftarrow R_1 - \frac{1}{2} R_2$$

$$R_3 \leftarrow R_3 - \frac{1}{2} R_2$$

$$R_2 \leftarrow \frac{1}{2} R_2$$

$$A_2 = \begin{bmatrix} 0 & 2.5 & 5.5 \\ 1 & 0.5 & 0.5 \\ 0 & 2.5 & 2.5 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

```
function GE_PP(n, (a_ij), (p_i))
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            z ← a_{p_i k} / a_{p_k k}
            a_{p_i k} ← 0
            for j = k + 1 to n do
                a_{p_i j} ← a_{p_i j} − z a_{p_k j}
    return (a_ij)
```

$$\max(|1|, |2|, |1|) = 2$$

$$R_1 \leftarrow R_1 - \frac{1}{2} R_2$$

$$R_3 \leftarrow R_3 - \frac{1}{2} R_2$$

$$R_2 \leftarrow \frac{1}{2} R_2$$

$$A_2 = \begin{bmatrix} 0 & 2.5 & 5.5 \\ 1 & 0.5 & 0.5 \\ 0 & 2.5 & 2.5 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

function GE_PP$(n, (a_{ij}), (p_i))$
  for $k = 1$ **to** $n - 1$ **do**
    for $i = k + 1$ **to** n **do**
      $z \leftarrow a_{p_i k} / a_{p_k k}$
      $a_{p_i k} \leftarrow 0$
      for $j = k + 1$ **to** n **do**
        $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
  return (a_{ij})

$$\max(|2.5|, |2.5|) = 2.5$$

$$A_2 = \begin{bmatrix} 0 & 2.5 & 5.5 \\ 1 & 0.5 & 0.5 \\ 0 & 2.5 & 2.5 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

```
function GE_PP(n, (a_ij), (p_i))
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            z ← a_{p_i k} / a_{p_k k}
            a_{p_i k} ← 0
            for j = k + 1 to n do
                a_{p_i j} ← a_{p_i j} − z a_{p_k j}
    return (a_ij)
```

$$\max(|2.5|, |2.5|) = 2.5$$

$$R_3 \leftarrow R_3 - R_2$$

$$R_1 \leftarrow \frac{2}{5} R_1$$

Direct Methods: Pivoting Examples

$$A_3 = \begin{bmatrix} 0 & 1 & 2.2 \\ 1 & 0.5 & -0.6 \\ 0 & 0 & -3 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

function GE_PP$(n, (a_{ij}), (p_i))$
  for $k = 1$ **to** $n - 1$ **do**
    for $i = k + 1$ **to** n **do**
      $z \leftarrow a_{p_i k} / a_{p_k k}$
      $a_{p_i k} \leftarrow 0$
      for $j = k + 1$ **to** n **do**
        $a_{p_i j} \leftarrow a_{p_i j} - z a_{p_k j}$
  return (a_{ij})

$$\max(|2.5|, |2.5|) = 2.5$$

$$R_3 \leftarrow R_3 - R_2$$

$$R_1 \leftarrow \frac{2}{5} R_1$$

$$A_3 = \begin{bmatrix} 0 & 1 & 2.2 \\ 1 & 0.5 & -0.6 \\ 0 & 0 & -3 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

```
function GE_PP(n, (a_ij), (p_i))
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            z ← a_{p_i k} / a_{p_k k}
            a_{p_i k} ← 0
            for j = k + 1 to n do
                a_{p_i j} ← a_{p_i j} − z a_{p_k j}
    return (a_ij)
```

$$R_3 \leftarrow -\frac{1}{3} R_3$$

$$A_4 = \begin{bmatrix} 0 & 1 & 2.2 \\ 1 & 0.5 & \text{-}0.6 \\ 0 & 0 & 1 \end{bmatrix}$$

$$p = \begin{bmatrix} 2, & 1, & 3 \end{bmatrix}$$

```
function GE_PP(n, (a_ij), (p_i))
    for k = 1 to n − 1 do
        for i = k + 1 to n do
            z ← a_{p_i k} / a_{p_k k}
            a_{p_i k} ← 0
            for j = k + 1 to n do
                a_{p_i j} ← a_{p_i j} − z a_{p_k j}
    return (a_ij)
```

$$R_3 \leftarrow -\frac{1}{3} R_3$$

Recall: General Conditions for Iterative Method Convergence

Recall: General Conditions for Iterative Method Convergence

Theorem

For the linear system $Ax = b$ *with* A *invertible, define the iteration formula*

$$x^{(k)} = Gx^{(k-1)} + c.$$

The sequence $\left\{x^{(k)}\right\}$ *will converge to* $(I - G)^{-1}c$ *provided that* $\rho(G) < 1$.

Recall: General Conditions for Iterative Method Convergence

Theorem

For the linear system $Ax = b$ with A invertible, define the iteration formula

$$x^{(k)} = Gx^{(k-1)} + c.$$

The sequence $\left\{x^{(k)}\right\}$ will converge to $(I - G)^{-1}c$ provided that $\rho(G) < 1$.

Methods

- Richardson
- Jacobi
- Gauss-Seidel
- SOR
- Steepest Descent
- Conjugate Gradient

Method

- Richardson

System Stipulations

- A invertible

Iteration Formula

- $x^{(k)} = (I - A)x^{(k-1)} + b = x^{(k-1)} + r^{(k-1)}$

Convergence Criteria

- $\rho(I - A) < 1$

- Consider matrix $A = \begin{bmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{bmatrix}$ with solution $b = \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$.

- Consider matrix $A = \begin{bmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{bmatrix}$ with solution $b = \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$.
- Algorithm is $x^{(k+1)} = (I - \omega A)x^{(k)} + \omega b^{(k)}$ for some scalar $\omega \neq 0$ such that $|r| < 1$.

- Consider matrix $A = \begin{bmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{bmatrix}$ with solution $b = \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$.
- Algorithm is $x^{(k+1)} = (I - \omega A)x^{(k)} + \omega b^{(k)}$ for some scalar $\omega \neq 0$ such that $|r| < 1$.
- Let $x^{(0)} = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}^T$ and $\omega = \dfrac{1}{6}$.

- Consider matrix $A = \begin{bmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{bmatrix}$ with solution $b = \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$.
- Algorithm is $x^{(k+1)} = (I - \omega A)x^{(k)} + \omega b^{(k)}$ for some scalar $\omega \neq 0$ such that $|r| < 1$.
- Let $x^{(0)} = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}^T$ and $\omega = \frac{1}{6}$.
-

$$x^{(1)} = \begin{bmatrix} 0 & -1/6 & -1/6 \\ -1/3 & 1/3 & 0 \\ -1/6 & -1/3 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} + \frac{1}{6} \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$$
$$= \begin{bmatrix} 4/3 & 0 & 0 \end{bmatrix}^T$$

- Consider matrix $A = \begin{bmatrix} 6 & 1 & 1 \\ 2 & 4 & 0 \\ 1 & 2 & 6 \end{bmatrix}$ with solution $b = \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix}$.
- Algorithm is $x^{(k+1)} = (I - \omega A)x^{(k)} + \omega b^{(k)}$ for some scalar $\omega \neq 0$ such that $|r| < 1$.
- Let $x^{(0)} = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}^T$ and $\omega = \frac{1}{6}$.
- $$\begin{aligned} x^{(1)} &= \begin{bmatrix} 0 & -1/6 & -1/6 \\ -1/3 & 1/3 & 0 \\ -1/6 & -1/3 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} + \frac{1}{6} \begin{bmatrix} 12 \\ 0 \\ 6 \end{bmatrix} \\ &= \begin{bmatrix} 4/3 & 0 & 0 \end{bmatrix}^T \end{aligned}$$
- After 12 iterations, $x^{(12)} \approx \begin{bmatrix} 2 & -1 & 1 \end{bmatrix}^T$.

Method

- Jacobi

System Stipulations

- A invertible
- A diagonally dominant

Iteration Formula

- $x^{(k)} = (I - D^{-1}A)x^{(k-1)} + D^{-1}b$
- $D = a_{ii}$

Convergence Criteria

- $\rho(I - D^{-1}A) < 1$

Method

- Gauss-Seidel

System Stipulations

- A invertible
- A diagonally dominant

Iteration Formula

- $x^{(k)} = (I - L^{-1}A)x^{(k-1)} + L^{-1}b$
- $L = a_{ij}, i \geq j$

Convergence Criteria

- $\rho(I - L^{-1}A) < 1$

Method

- SOR

System Stipulations

- Complex system
- A: positive definite, Hermitian

Iteration Formula

- $x^{(k+1)} = (I - (\alpha D - C)^{-1} A)x^{(k)} + (\alpha D - C)^{(-1)} b$
- D: positive definite, Hermitian
- $C : C + C^{*} = D - A$
- $\alpha \in \mathbb{R} \ni \alpha > \frac{1}{2}$

Convergence Criteria

- $\rho(I - (\alpha D - C)^{-1} A) < 1$

Typical values

- $D = a_{ii}, C = -a_{ij} \ni i \geq j$

Method

- Steepest Descent

System Stipulations

- A invertible
- A symmetric, positive definite

Iteration Formula

- $x^{(k)} = x^{(k-1)} + t_{k-1}v^{(k-1)}$
- $t_{k-1} = \frac{\langle v^{(k-1)}, b - Ax^{(k-1)}\rangle}{\langle v^{(k-1)}, Av^{(k-1)}\rangle}$
- $v^{(k-1)} = -\frac{d}{dx}q(x^{(k-1)})$
- $q(x) = \langle x, Ax\rangle - 2\langle x, b\rangle$

Indirect Methods: Conjugate Gradient

Method

- Conjugate Gradient

System Stipulations

- A invertible
- A symmetric, positive definite

Iteration Formula

- $x^{(k)} = x^{(k-1)} + t_{k-1} v^{(k-1)}$
- $t_{k-1} = \frac{\langle v^{(k-1)}, b - Ax^{(k-1)} \rangle}{\langle v^{(k-1)}, Av^{(k-1)} \rangle}$
- $\left\{ v^{(0)}, \ldots, v^{(n)} \right\}$ are A-orthogonal
- $v^{(k-1)} = -\frac{d}{dx} q(x^{(k-1)})$
- $q(x) = \langle x, Ax \rangle - 2\langle x, b \rangle$