Computer-Assisted Problem-Solving / Numerical Methods

Iterative Methods for $A\vec{x} = \vec{b}$

version: March 4th, 2020

Legend: Method, Theory, Example, Advanced, Appendix

Theory

Iterative Methods for $A\vec{x} = \vec{b}$

Important part of numerical mathematics.
Still vigorously developed !

Example: $x^{(m+1)} = Mx^{(m)} + c$

M determined by A, c determined by A and b

When M,c constant $\rightarrow$ stationary method

Start vector $x^{(0)}$, and then $x^{(1)}$,$x^{(2)}$,...,$x^{(\infty)}$

Continue until residual $\|r^{(m)}\| := \|Ax^{(m)} - b\|$ small

Stationary methods (point-wise methods):
- **relatively simple (analysis, implementation)**
- **Jacobi, Gauss-Seidel, SOR, JOR, SLOR, ...**

Non-stationary methods (vector-wise methods):
- **more complicated, but often more effective**
- **CG, PCG, MICCG, MILU, GMRES, BiCG, BiCG-STAB, etc.**

Method

Stationary Methods

Derivation and analysis through partitioning A

Partition method 1: $A = N - P$

$$Ax = b \Longrightarrow Nx = b + Px \Longrightarrow Nx^{(m+1)} = b + Px^{(m)}$$

$$\Longrightarrow x^{(m+1)} = N^{-1}b + N^{-1}Px^{(m)}$$

Partition method 2: $A = L + D + R$

Example:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix} + \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix} + \begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

Remark:

partition methods often related, e.g.

$N = D$, $P = -(L + R)$ **(Jacobi)**

$N = D + L$, $P = -R$ **(Gauss-Seidel)**

Method

Jacobi Method

Also known as 'Gauss-Jacobi Method' or 'Method of Simultaneous Changes'

$Ax = b$, $A = N - P \Longrightarrow$

$Nx^{(m+1)} = b + Px^{(m)} \Longrightarrow x^{(m+1)} = N^{-1}b + N^{-1}Px^{(m)}$

$A = L + D + R$, **via** $N = D$, $P = -(L + R) \Longrightarrow$

$$x^{(m+1)} = D^{-1}b - D^{-1}(L + R)x^{(m)}$$

Iteration rule:

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(m)}\}, \quad i = 1 \cdots n$$

Algorithm:

1 start at iteration $m = 0$ **with initial guess** $\vec{\mathbf{x}}_0$

$\Longrightarrow$ **2. for** $i = 1 \ldots N$

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(m)}\}$$

3 if converged $\Longrightarrow$ **STOP**

4 $m = m + 1$ **and goto (2)** $\Longrightarrow$

Number of operations: $\#\text{iterations} * 2N^2 \Longrightarrow \mathcal{O}(N^2)$

Stop criteria:

1) $\|x^{(m+1)} - x^{(m)}\| < \epsilon$ **3)** $\|r^{(m)}\| < \epsilon$, $r^{(m)} = Ax^{(m)} - b$

2) $\frac{\|x^{(m+1)} - x^{(m)}\|}{\|x^{(m+1)}\|} < \epsilon$ **4)** $\frac{\|r^{(m)}\|}{\|x^{(m)}\|} < \epsilon$ $r^{(m)}$ **is residual**

Example
$$\begin{pmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ -5 \\ 14 \end{pmatrix}$$

Start $x^{(0)} = (0\ 0\ 0)^T$ **Exact solution** $x = (1\ 1\ 1)^T$

Exact Error in iteration m**:** $\|e^{(m)}\|_\infty = \|x - x^{(m)}\|_\infty$

Error Ratio $= \|e^{(m+1)}\|_\infty / \|e^{(m)}\|_\infty$

m	x_1	x_2	x_3	$\|e^{(m)}\|_\infty$	**Ratio**
0	**0**	**0**	**0**	**1**	
1	**1.4**	**0.5**	**1.4**	**0.5**	**0.5**
2	**1.11**	**1.20**	**1.11**	**0.2**	**0.4**
3	**0.929**	**1.055**	**0.929**	**0.071**	**0.36**
4	**0.9906**	**0.9645**	**0.9906**	**0.0355**	**0.50**
5	**1.01159**	**0.9953**	**1.01159**	**0.01159**	**0.33**
6	**1.000251**	**1.005795**	**1.000251**	**0.005795**	**0.50**

In 6 iterations: (approx.) 2 digits accuracy
(see Appendix A)

A **is diagonally dominant**

Linear convergence with rate $\alpha \approx 0.5$

Estimate of rate follows from
"proof of convergence" Jacobi Method

Theorem:

A **diagonally dominant** $\Longrightarrow$
Jacobi converges with "linear rate"
(independent of start vector $x^{(0)}$**)**

Definitions:

1) Error in iteration m**:** $e^{(m)} := x - x^{(m)}$

2) Linear convergence:

$\|e^{(m+1)}\|_\infty = \alpha \|e^{(m)}\|_\infty$**, with** $|\alpha| < 1$

Proof of convergence Jacobi:

$$x_i = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j\}, \quad i = 1 \cdots n$$

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(m)}\}, \quad i = 1 \cdots n$$

Combine (subtract) $\Longrightarrow$

$$e_i^{(m+1)} = x_i - x_i^{(m+1)} = -\frac{1}{a_{ii}}\{\sum_{j=1, j\neq i}^{n} a_{ij}(x_j - x_j^{(m)})\}$$
$$= -\sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(m)} \quad i = 1 \cdots n$$

Estimate $\Longrightarrow$

$$|e_i^{(m+1)}| \leq \sum_{j=1, j\neq i}^{n} |\frac{a_{ij}}{a_{ii}}| \ \|e^{(m)}\|_\infty$$

Define $$\mu := \max_{1\leq i\leq n} \sum_{j=1, j\neq i}^{n} |\frac{a_{ij}}{a_{ii}}|$$

As a result

$$|e_i^{(m+1)}| \leq \mu \|e^{(m)}\|_\infty$$

Right-hand side independent of i $\Longrightarrow$

$$\|e^{(m+1)}\|_\infty \leq \mu \|e^{(m)}\|_\infty$$

and hence convergence if $\mu < 1$, with rate μ

(Strictly) Diagonally dominant $\Longrightarrow \mu < 1$, since

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1 \cdots n \Longrightarrow \mu < 1$$

Example:

$$A = \begin{pmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{pmatrix}$$

$$\mu := \max_{1 \leq i \leq n} \sum_{j=1, j\neq i}^{n} |\frac{a_{ij}}{a_{ii}}| = \frac{5}{10} \Longrightarrow$$

Convergence rate Jacobi (in this example) 0.5

Remarks:

rate μ is conservative (often somewhat faster)

if $a_{ii} = 0 \Longrightarrow$ apply re-ordering of matrix

Method

Gauss-Seidel Method

Method of Successive Changes

$Ax = b,\ A = N - P \Longrightarrow$

$Nx^{(m+1)} = b + Px^{(m)} \Longrightarrow x^{(m+1)} = N^{-1}b + N^{-1}Px^{(m)}$

$N = D + L,\ P = -R \Longrightarrow$

$$x^{(m+1)} = (D + L)^{-1}b - (D + L)^{-1}Rx^{(m)}$$

This is correct, but more convenient is

$$(D + L)x^{(m+1)} = b - Rx^{(m)} \Longrightarrow$$

$$x^{(m+1)} = D^{-1}b - D^{-1}(Lx^{(m+1)} + Rx^{(m)})$$

Iteration rule:

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\},\quad i = 1 \cdots n$$

Use calculated $x_i^{(m+1)}$ **directly for** $x_j^{(m+1)}$ $j \geq i+1$

Number of operations: $\#\text{iterations} * 2N^2 \Longrightarrow \mathcal{O}(N^2)$

Stop criteria: same as for Jacobi Method

Algorithm:

1) start at iteration $m = 0$ with initial guess $\vec{\mathbf{x}}_0$

$\Longrightarrow$ **2) for** $i = 1 \ldots N$

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)}\}$$

3) if converged $\Longrightarrow$ STOP

4) $m = m + 1$ **and goto (2)** $\Longrightarrow$

Same example as for Jacobi:

$$\begin{pmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ -5 \\ 14 \end{pmatrix}$$

Start vector $x^{(0)} = (0\ 0\ 0)^T$ **Solution** $x = (1\ 1\ 1)^T$

m	x_1	x_2	x_3	$\|e^{(m)}\|_\infty$	**Ratio**
0	**0**	**0**	**0**	**1**	
1	**1.40000**	**0.78**	**1.026**	**4.00E-1**	**0.40**
2	**1.06340**	**1.02048**	**0.98752**	**6.34E-2**	**0.16**
3	**0.99951**	**0.99528**	**1.00191**	**4.90E-3**	**0.08**
4	**1.00123**	**1.00082**	**0.99963**	**1.23E-3**	**0.25**
5	**0.99979**	**0.99985**	**1.00007**	**2.08E-4**	**0.17**

Linear convergence with rate $\alpha < 0.5$

Faster than Jacobi

Notice: A **is diagonally dominant**

Theorem:

A **diagonally dominant** $\Longrightarrow$

1) Gauss-Seidel linearly convergent
(independent of start vector $x^{(0)}$**)**

2) Gauss-Seidel converges faster than Jacobi

Estimate of convergence rate follows from proof of convergence of Gauss-Seidel Method (see Appendix B)

Method

SOR Method

Successive-Over-Relaxation Method
Gauss-Seidel with acceleration

Iteration rule Gauss-Seidel:

$$\hat{x_i}^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\}, \quad i = 1 \cdots n$$

Extrapolation:

$$x_i^{(m+1)} = \omega \hat{x_i}^{(m+1)} + (1-\omega)x_i^{(m)}$$

Under-relaxation $0 < \omega < 1$
Over-relaxation $1 < \omega < \infty$
Gauss-Seidel $\omega = 1$

Algorithm:

1) start at iteration $m = 0$ **with initial guess** $\vec{\mathbf{x}}_0$

$\Longrightarrow$ **2) for** $i = 1 \ldots N$

$$\begin{aligned} \hat{x_i}^{(m+1)} &= \frac{1}{a_{ii}}\{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\} \\ \mathbf{x}_i^{m+1} &= \omega * \hat{\mathbf{x}}_i + (1-\omega) * \mathbf{x}_i^m \end{aligned}$$

3) if converged $\Longrightarrow$ **STOP**

4) $m = m + 1$ **and goto (2)** $\Longrightarrow$

Number of operations: $\#\text{iterations} * (2N^2 + 4N)$ $\Longrightarrow \mathcal{O}(N^2)$

Advantage: $\#\text{iterations}$ **much smaller!**

Number of operations often $\mathcal{O}(N)$ **or** $\mathcal{O}(N^{3/2})$ **depending on sparsity of matrix**
(avoid use of full A **!)**

Partition method for Gauss-Seidel:

$Ax = b$**,** $A = N - P$ **met** $N = D + L$ **en** $P = -R$ $\Longrightarrow Nx^{(m+1)} = b + Px^{(m)} \Longrightarrow$

$$x^{(m+1)} = D^{-1}b - D^{-1}(Lx^{(m+1)} + Rx^{(m)})$$

Partition for SOR via back calculation

$$x^{(m+1)} = \omega\{D^{-1}b - D^{-1}(Lx^{(m+1)} + Rx^{(m)})\} + (1-\omega)x^{(m)}$$

$$(I + \omega D^{-1}L)x^{(m+1)} = \omega D^{-1}b - \omega D^{-1}Rx^{(m)} + (1-\omega)x^{(m)}$$

$$\begin{aligned} N &= I + \omega D^{-1}L \\ P &= -\omega D^{-1}R + (1-\omega)I \end{aligned}$$

Convergence behaviour SOR: $e^{(m+1)} = M_\omega e^{(m)}$**, with iteration matrix**

$$M_\omega = N^{-1}P = (I + \omega D^{-1}L)^{-1}\{(1-\omega)I - \omega D^{-1}R\}$$

For optimal convergence:
$r_\sigma(M_\omega)$ **minimal (and** < 1**)**
With this enormous gain can be achieved !

Example: $$\begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ 30 \\ -24 \end{pmatrix}$$

Start vector $x^{(0)} = (1; 1; 1)^T$

Exact solution $x = (3; 4; -5)^T$

Gauss-Seidel			
m	x_1	x_2	x_3
0	**1**	**1**	**1**
1	**5.25**	**3.8125**	**-5.046875**
2	**3.1406250**	**3.8828125**	**-5.0292969**
3	**3.0878906**	**3.9267578**	**-5.0183105**
4	**3.0549316**	**3.9542236**	**-5.0114441**
5	**3.0343323**	**3.9713898**	**-5.0071526**
6	**3.0214577**	**3.9821186**	**-5.0044703**
7	**3.0134110**	**3.9888241**	**-5.0027940**

SOR ($\omega = 1.25$)			
m	x_1	x_2	x_3
0	**1**	**1**	**1**
1	**6.3125**	**3.5195313**	**-6.6501465**
2	**2.6223145**	**3.9585266**	**-4.6004238**
3	**3.1333027**	**4.0102646**	**-5.0966863**
4	**2.9570512**	**4.0074838**	**-4.9734897**
5	**3.0037211**	**4.0029250**	**-5.0057135**
6	**2.9963276**	**4.0009262**	**-4.9982822**
7	**3.0000498**	**4.0002586**	**-5.0003486**

SOR: 14 iterations for $x_i - x_i^{(m)} < 1.0E{-}7$**,** $i = 1 \cdots 3$

Gauss-Seidel: 34 iterations

Advanced

Conjugate Gradient Method (CG)

Notation: inner product $(x, y) = x^T y$

Definitions:
(1) A **positive definite:** $(Ax, x) > 0$ **if** $x \neq 0$
(2) x **and** y **conjugate:** $x^T A y = (x, Ay) = 0$

Theorem:
A **symmetric** $\Longrightarrow (Ax, y) = (x, Ay)$

Theorem:
If A **symmetric and positive definite** $\Longrightarrow$
Solving $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$ **is equivalent to minimizing**

$$J[\vec{\mathbf{y}}] := \frac{1}{2}(\vec{\mathbf{y}}, A\vec{\mathbf{y}}) - (\vec{\mathbf{y}}, \vec{\mathbf{b}})$$

At iteration k**: approximation** $\vec{\mathbf{x}}_k$

Problem: find search direction $\vec{\mathbf{z}}_k$ **such that new sol.** $\vec{\mathbf{x}}_{k+1} = \vec{\mathbf{x}}_k + \alpha_k \vec{\mathbf{z}}_k$ **minimizes** $J[\vec{\mathbf{x}}_k + \alpha_k \vec{\mathbf{z}}_k]$

When search direction $\vec{\mathbf{z}}_k$ **known:**
Minimum $J[\vec{\mathbf{y}}]$ **found when**
(J **quadr. function of** α**; A symm. and pos.def.):**

$$\frac{\partial}{\partial \alpha} J[\vec{\mathbf{x}}_k + \alpha_k \vec{\mathbf{z}}_k] = 0 \Longrightarrow \alpha_k = \frac{(\vec{\mathbf{z}}_k, \vec{\mathbf{r}}_k)}{(\vec{\mathbf{z}}_k, A\vec{\mathbf{z}}_k)}$$

with $\vec{\mathbf{r}}_k = \vec{\mathbf{b}} - A\vec{\mathbf{x}}_k$ **residual at iteration** k

How to find search direction $\vec{\mathbf{z}}_k$?
Different approaches $\Longrightarrow$ different CG variants

Example: method of steepest descent
Multi-Variable calculus:
Greatest decrease of $J[\vec{\mathbf{y}}]$ for direction $-\nabla J[\vec{\mathbf{y}}]$

$$-\nabla J[\vec{\mathbf{y}}] = -\frac{1}{2}\nabla(\vec{\mathbf{y}}, A\vec{\mathbf{y}}) + \nabla(\vec{\mathbf{y}}, \vec{\mathbf{b}}) = -A\vec{\mathbf{y}} + \vec{\mathbf{b}}$$

Evaluated in $\vec{\mathbf{y}} = \vec{\mathbf{x}}_k \quad \Longrightarrow \quad \vec{\mathbf{z}}_k = -\nabla J[\vec{\mathbf{x}}_k] = \vec{\mathbf{r}}_k$

Algorithm: Steepest Descent (for $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$)

1. **start at iteration $k = 0$ with initial guess $\vec{\mathbf{x}}_0$**
2. **compute initial residual $\vec{\mathbf{r}}_0 = \vec{\mathbf{b}} - A\vec{\mathbf{x}}_0$**

$\Longrightarrow$ 3. **compute α_k from** $\alpha_k = \dfrac{(\vec{\mathbf{r}}_k, \vec{\mathbf{r}}_k)}{(\vec{\mathbf{r}}_k, A\vec{\mathbf{r}}_k)}$

4. **update solution $\vec{\mathbf{x}}_{k+1} = \vec{\mathbf{x}}_k + \alpha_k \vec{\mathbf{r}}_k$**
5. **if converged $\Longrightarrow$ STOP**
6. **$k = k + 1$ and goto (3) $\Longrightarrow$**

Theorem:
convergence Steepest Descent in $\leq n$ iterations
(in case of exact arithmetic)

Example: $\begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ 30 \\ -24 \end{pmatrix}$

Startvector $x^{(0)} = (0; 0; 0)^T$

Exact solution $x = (3; 4; -5)^T$

Steepest Descent			
k	x_1	x_2	x_3
0	**0**	**0**	**0**
1	**6**	**0**	**0**
2	$\frac{6}{7}$	$\frac{48}{7}$	**0**
3	**3**	**4**	**-5**

Ready in exactly 3 iterations!

Remarks:

- **Kind of direct method**
- **Computation of** α_k **most expensive**
- **Iterations very different from SOR: (vector-wise approach)**
- **Irregular convergence behaviour**
- **Impractical: search directions often not optimal (e.g. because of round-off)**
- **Impractical for large matrices: approx.** n **iterations, with e.g.** $n = 10^5$ **or** 10^6

Advanced

CG Methods & Preconditioning

Speed-up of steepest descent method

More complicated linear algebra

Orthogonal search directions $\vec{\mathbf{z}}_k$:
$(\vec{\mathbf{z}}_i, A\vec{\mathbf{z}}_j) = 0$ if $i \neq j$ ($\vec{\mathbf{z}}_i$ and $\vec{\mathbf{z}}_j$ conjugate) $\Longrightarrow$
Conjugate Gradient (CG) methods

Krylov (linear-) subspaces:
$K^n(A; \vec{\mathbf{r}}_0) = \text{span}\{\vec{\mathbf{r}}_0, A\vec{\mathbf{r}}_0, A^2\vec{\mathbf{r}}_0, \ldots A^{n-1}\vec{\mathbf{r}}_0\} \Longrightarrow$
Krylov subspace methods

Search directions depend on previous search directions

Computational effort CG comparable with SOR

Further speed-up by means of preconditioning:
$A\vec{\mathbf{x}} = \vec{\mathbf{b}} \Longrightarrow CA\vec{\mathbf{x}} = C\,\vec{\mathbf{b}}$

Preconditioner (matrix) $C \approx A^{\text{inv}}$, but ...
C should be easy to compute and store
Equations with C should be 'easy' to solve

Preconditioning $\Longrightarrow$
Large reduction of computational effort !

"It's the preconditioning that counts"

Algorithm: Preconditioned CG (for $A\vec{\mathbf{x}} = \vec{\mathbf{b}}$)

0. start at iteration $k = 0$ with initial guess $\vec{\mathbf{x}}_0$

1. compute initial residual $\vec{\mathbf{r}}_0 = \vec{\mathbf{b}} - A\vec{\mathbf{x}}_0$

2. compute (initial) auxiliary variables:
$\vec{\mathbf{w}}_0 = C\vec{\mathbf{r}}_0 \qquad \alpha = (\vec{\mathbf{w}}_0, \vec{\mathbf{w}}_0)$

3. compute initial search direction $\vec{\mathbf{z}}_0 = C\vec{\mathbf{w}}_0$

4. $k = 1$

$\Longrightarrow$ **5. compute auxiliary vector $\vec{\mathbf{u}}_k = A\vec{\mathbf{z}}_k$**

6. compute auxiliary variable $t = \alpha/(\vec{\mathbf{u}}_k, \vec{\mathbf{z}}_k)$

7. update solution and residual vectors

$$\begin{aligned} \vec{\mathbf{x}}_{k+1} &= \vec{\mathbf{x}}_k + t\,\vec{\mathbf{z}}_k \\ \vec{\mathbf{r}}_{k+1} &= \vec{\mathbf{r}}_k - t\,\vec{\mathbf{u}}_k \\ \vec{\mathbf{w}}_{k+1} &= C\vec{\mathbf{r}}_{k+1} \end{aligned}$$

8. compute auxiliary variable $\beta = (\vec{\mathbf{w}}_{k+1}, \vec{\mathbf{w}}_{k+1})$

9. update search direction

$$\vec{\mathbf{z}}_{k+1} = C\vec{\mathbf{w}}_{k+1} + (\beta/\alpha)\vec{\mathbf{z}}_k$$

10. $\alpha = \beta$

11. check convergence: $\|\vec{\mathbf{z}}_{k+1}\| < \epsilon$, $\|\vec{\mathbf{r}}_{k+1}\| < \epsilon$
if converged $\Longrightarrow$ STOP

12. $k = k + 1$ and goto (5) $\Longrightarrow$

Example 1: $\begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 24 \\ 30 \\ -24 \end{pmatrix}$

CG without preconditioner $(C = I)$

Startvector $x^{(0)} = (0; 0; 0)^T$

Exact solution $x = (3; 4; -5)^T$

CG			
k	x_1	x_2	x_3
0	**0**	**0**	**0**
1	**3.525773196**	**4.407216495**	**-3.525773196**
2	**2.858011121**	**4.148971939**	**-4.954222164**
3	**2.999999998**	**4.000000002**	**-4.999999998**

Ready in (exactly) 3 iterations!

Example 2:
$$\begin{pmatrix} 0.2 & 0.1 & 1 & 1 & 0 \\ 0.1 & 4 & -1 & 1 & -1 \\ 1 & -1 & 60 & 0 & -2 \\ 1 & 1 & 0 & 8 & 4 \\ 0 & -1 & -2 & 4 & 700 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Initial vector $x^{(0)} = (0; 0; 0; 0; 0)^T$

Exact solution

$x = (7.859713; 0.4229264; -0.07359224; -0.5406430; 0.01062616)^T$

A is symmetric, positive definite,
but ill-conditioned: $k(A)_\infty = 13961.71 >> 1$

Compare 5 methods, stop tolerance $\epsilon = 0.01$

method	**iterations**	$\|x - x^{(m)}\|_\infty$
Jacobi	**49**	**0.00305834**
Gauss-Seidel	**15**	**0.02445559**
SOR ($\omega = 1.25$)	**7**	**0.00818607**
CG ($C = I$)	**5**	**0.00629785**
PCG ($C = D^{-1}$)	**4**	**0.00009312**

Notice: Iterations of Jacobi, GS and SOR different from iterations with CG and PCG

Preconditioning:
PCG better than CG: preconditioning helps
Diagonal matrix D^{-1} easy to handle

Equations with preconditioner C in algorithm should be easy to solve:

$\vec{\mathbf{w}}_{k+1} = C\vec{\mathbf{r}}_{k+1}$, with $C \approx A^{\text{inv}} \Longrightarrow A\vec{\mathbf{w}}_{k+1} \approx \vec{\mathbf{r}}_{k+1}$

solve $\vec{\mathbf{w}}_{k+1}$ to a good approximation

For example: $C = D^{-1} \approx A^{\text{inv}}$,
when A strongly diagonally dominant

Various methods:

1) ILU method:
Incomplete LU factorization $A = LU$

2) ICCG method:
Incomplete Choleski factorization $A = LL^T$
(ILU for symmetric matrices)

3) Modified ILU and ICCG (MILU, MICCG):
modification w.r.t. diagonal

4) BiCG-STAB method:
Bi-Conjugate Gradient Stabilized
- extension of (Preconditioned) CG method
- combination of two CG-like iterations
- more stable convergence behaviour
- can be combined with any preconditioner

Appendix

A: Spectral radius & Convergence

For the error in Jacobi:

$$e_i^{(m+1)} = x_i - x_i^{(m+1)} = -\sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(m)} \quad i = 1 \cdots n$$

This can be written as $e^{(m+1)} = Me^{(m)}$, with

$$M := -\begin{pmatrix} 0 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & \dots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \dots & \dots & 0 \end{pmatrix}$$

Induction $\Longrightarrow$ $e^{(m)} = M^m e^{(0)}$
For $e^{(m)} \to 0$ $(m \to \infty)$ we must have $M^m \to 0$

General splitting $A = N - P$:

$Ax = b \Longrightarrow Nx = b + Px \Longrightarrow x = N^{-1}b + N^{-1}Px$ **(1)**

Iterative method: $x^{(m+1)} = N^{-1}b + N^{-1}Px^{(m)}$ **(2)**

Combination (1) and (2) $\Longrightarrow$

$$e^{(m+1)} = x - x^{(m+1)} = N^{-1}Pe^{(m)}$$

Hence, in $e^{(m+1)} = Me^{(m)}$ we replace $M = N^{-1}P$

Theorem from Linear Algebra:
1) $r_\sigma(M) < 1$ **(spectral radius smaller than 1) necessary and sufficient for** $M^m \to 0$

2) $r_\sigma(M)$ **smaller** $\Longrightarrow$ **convergence speed** $\uparrow$

Theorem:
Stationary Methods convergent $\Longleftrightarrow r_\sigma(M) < 1$

Hence: $r_\sigma(N^{-1}P) < 1$
necessary and sufficient for convergence
Computation via $det(N^{-1}P - \lambda I) = 0$**,**
or generally easier via $det(P - \lambda N) = 0$

Theorem:

$\|M\| < 1$ **for arbitrary norm** $\Longrightarrow$

stationary method convergent

(sufficient, not necessary !)

Reason: $$r_\sigma(M) = \inf_{\|\ \|} \|M\|$$

(spectral radius smaller than every other norm)

Remark:

condition $\mu < 1$ **for Jacobi** $\Longleftrightarrow \|M\|_\infty < 1$

$$\mu := \max_{1 \le i \le n} \sum_{j=1, j \ne i}^{n} |\frac{a_{ij}}{a_{ii}}|$$

Example (see Jacobi): $\begin{pmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ -5 \\ 14 \end{pmatrix}$

Jacobi method $\Longrightarrow$

$$M_j = -D^{-1}(L+R) = \begin{pmatrix} 0 & -0.3 & -0.1 \\ 0.2 & 0 & 0.3 \\ -0.1 & -0.3 & 0 \end{pmatrix}$$

For this M_j **we have :**

$\|M_j\|_\infty = 0.5, \quad \|M_j\|_1 = 0.6, \quad r_\sigma(M_j) \approx 0.39$ **(all** < 1**)**

Definitions (after m **iterations** $e^{(m)} = M^m e^{(0)}$**):**

- (total) convergence factor $\|M^m\|$

- average convergence factor $\|M^m\|^{1/m}$

- average conv. speed $R_m = -\frac{1}{m} \log \|M^m\|$

- asympt. conv. speed $R_\infty = \lim_{m\to\infty} R_m$

$= .. = -\log\{r_\sigma(M)\}$

Gain ν **digits in** m **iterations:**

$\|e^{(m)}\| = \|M^m\| \|e^{(0)}\| = 10^{-\nu} \|e^{(0)}\| \Longrightarrow -^{10}\log \|M^m\| = \nu$

Average gain per iter.: $\frac{\nu}{m} = -\frac{1}{m}{}^{10}\log \|M^m\| = R_m$

For the example:

$$M_j^6 = \begin{pmatrix} -0.0008 & 0.0018 & -0.0013 \\ -0.0014 & -0.0027 & -0.0017 \\ -0.0008 & 0.0018 & -0.0013 \end{pmatrix}$$

Which gives:

$\|M_j^6\|_\infty = 0.0058$**,** $\|M_j^6\|_\infty^{1/6} = 0.42$**,** $R_6 = 0.37$**,** $R_\infty = 0.41$

6 iterations $\Longrightarrow 6 * 0.4 \approx 2$ **gain in digits**

Appendix

B: Convergence of Gauss-Seidel

Theorem:
A diagonally dominant $\Longrightarrow$
1) Gauss-Seidel linearly convergent
(independent of start vector $x^{(0)}$)
2) Gauss-Seidel converges faster than Jacobi

Proof (convergence Gauss-Seidel):

$$x_i = \frac{1}{a_{ii}}\{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\}, \quad i = 1 \cdots n$$

$$x_i^{(m+1)} = \frac{1}{a_{ii}}\{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\}, \; i = 1...n$$

Combination (via splitting Σ) $\Longrightarrow$

$$\begin{aligned} e_i^{(m+1)} &= x_i - x_i^{(m+1)} \\ &= -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} e_j^{(m+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} e_j^{(m)} \quad i = 1 \cdots n \end{aligned}$$

Now define $\alpha_i := \sum_{j=1}^{i-1} |\frac{a_{ij}}{a_{ii}}| \quad \beta_i := \sum_{j=i+1}^{n} |\frac{a_{ij}}{a_{ii}}|$

with $\alpha_1 = \beta_n = 0$

This yields for μ (see Jacobi)

$$\mu = \max_{1\leq i\leq n} \sum_{j=1, j\neq i}^{n} |\frac{a_{ij}}{a_{ii}}| = \max_{1\leq i\leq n} (\alpha_i + \beta_i)$$

Estimation $\Longrightarrow$

$$|e_i^{(m+1)}| \leq \alpha_i \|e^{(m+1)}\|_\infty + \beta_i \|e^{(m)}\|_\infty \quad i = 1 \cdots n$$

Suppose k is index with $\|e^{(m+1)}\|_\infty = |e_k^{(m+1)}|$

This gives

$$\|e^{(m+1)}\|_\infty \leq \alpha_k \|e^{(m+1)}\|_\infty + \beta_k \|e^{(m)}\|_\infty \Longrightarrow$$

$$\|e^{(m+1)}\|_\infty \leq \frac{\beta_k}{1-\alpha_k} \|e^{(m)}\|_\infty \Longrightarrow$$

$$\|e^{(m+1)}\|_\infty \leq \eta \|e^{(m)}\|_\infty, \quad \textbf{with } \eta := \max_{1 \leq i \leq n} \frac{\beta_i}{1-\alpha_i}$$

Hence, there is linear convergence if $\eta < 1$

Computations give:

$$\forall i \quad (\alpha_i + \beta_i) - \frac{\beta_i}{1-\alpha_i} = \frac{(\alpha_i + \beta_i)(1-\alpha_i) - \beta_i}{1-\alpha_i} =$$
$$\frac{\alpha_i \{1 - (\alpha_i + \beta_i)\}}{1-\alpha_i} \geq \frac{\alpha_i}{1-\alpha_i}(1-\mu) \geq 0$$

with $\mu < 1$ and $0 \leq \alpha_i \leq 1$ used in the final step (both follow from diagonal dominance of A)

Choose i for which η is reached:

$$(\alpha_i + \beta_i) - \eta \geq 0$$

With $\mu = \max_{1 \leq i \leq n} (\alpha_i + \beta_i)$

we obtain $\mu - \eta \geq (\alpha_i + \beta_i) - \eta \geq 0,$

such that finally $\eta \leq \mu < 1$

Hence:

linear convergence ($\eta < 1$)

and faster than Jacobi ($\eta \leq \mu$)

For the example:

$$A = \begin{pmatrix} 10 & 3 & 1 \\ 2 & -10 & 3 \\ 1 & 3 & 10 \end{pmatrix} \Longrightarrow \mu := \max_{1\le i\le n} \sum_{j=1, j\neq i}^{n} |\frac{a_{ij}}{a_{ii}}| = \frac{5}{10}$$

$$\begin{aligned} \alpha_i &= \sum_{j=1}^{i-1} |\frac{a_{ij}}{a_{ii}}| \quad \alpha_1 = 0, \alpha_2 = \frac{2}{10}, \alpha_3 = \frac{4}{10} \\ \beta_i &= \sum_{j=i+1}^{n} |\frac{a_{ij}}{a_{ii}}| \quad \beta_1 = \frac{4}{10}, \beta_2 = \frac{3}{10}, \beta_3 = 0 \\ \eta &= \max_{1\le i\le n} \frac{\beta_i}{1-\alpha_i} = \frac{4}{10} \end{aligned}$$

$\Longrightarrow$ **Convergence rate Gauss-Seidel** 0.4

Remark:

for PDEs often $\eta = \mu = 1$**, but still convergence**

Appendix

C: Convergence Theorems

Theorem 1:

$x^{(m+1)} = Mx^{(m)} + c$ **convergent** $\iff r_\sigma(M) < 1$

Theorem 2:

$\|M\| < 1$ **for arbitrary matrix norm** $\Longrightarrow$

a) $x^{(m+1)} = Mx^{(m)} + c$ **convergent**

b) $\|x - x^{(m)}\| \leq \|M\|^m \, \|x - x^{(0)}\|$

c) $\|x - x^{(m)}\| \leq \frac{\|M\|^m}{1-\|M\|} \, \|x^{(1)} - x^{(0)}\|$

Theorem 3:

A **strict diagonally dominant** $\Longrightarrow$
Jacobi and Gauss-Seidel convergent

Theorem 4: $A = N - P$

a) A **and** N **symmetric and positive definite,** $2N - A$ **pos. definite** $\Longrightarrow$ **convergence**

b) A **symmetric and positive definite,** $N + N^T - A$ **positive definite** $\Longrightarrow$ **convergence**

c) Jacobi convergent if A **and** $2D - A$ **symmetric and positive definite**

d) Gauss-Seidel convergent if A **symmetric and positive definite**

Theorem 5: $\|x - x^{(m)}\| \approx r_\sigma^m(M)\|x - x^{(0)}\|$
(since $r_\sigma(M) \leq \|M\|$ **for arbitrary matrix norm)**

Theorem 6 (Stein-Rosenberg):
If $a_{ij} \leq 0$ $(i \neq j)$ **and** $a_{ii} > 0 \Longrightarrow$
exactly one statement below holds:

a) $0 \leq r_\sigma(M_g) < r_\sigma(M_j) < 1$

b) $1 < r_\sigma(M_j) < r_\sigma(M_g)$

c) $r_\sigma(M_j) = r_\sigma(M_g) = 0$

d) $r_\sigma(M_j) = r_\sigma(M_g) = 1$

Theorem 7 (Kahan):
If $a_{ii} \neq 0 \Longrightarrow r_\sigma(M_\omega) \geq |\omega - 1| \Longrightarrow$
SOR only converges if $0 < \omega < 2$
(necessary, not sufficient)

Theorem 8 (Ostrowski-Reich):
If A **symmetric and positive definite,**
and $0 < \omega < 2 \Longrightarrow$ **SOR convergent**

Theorem 9:
If A **symmetric, pos. definite and tri-diagonal**
Then: a) $r_\sigma(M_g) = r_\sigma^2(M_j) < 1$
b) SOR optimal: $\omega_{\text{opt}} = \dfrac{2}{1 + \sqrt{1 - r_\sigma^2(M_j)}}$

Theorem 10:
If: A **hermitian** $(A^* = A)$ **with positive diagonal**
Then: G.S. convergent $\Longleftrightarrow$ **A positive definite**

Example 1:
(see SOR example) $A = \begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$

A symmetric, positive definite and tri-diagonal

$$M_j = -D^{-1}(L+R) = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 0 & -3 & 0 \\ -3 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{3}{4} & 0 \\ -\frac{3}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix}$$

$$M_j - \lambda I = \begin{pmatrix} -\lambda & -\frac{3}{4} & 0 \\ -\frac{3}{4} & -\lambda & \frac{1}{4} \\ 0 & \frac{1}{4} & -\lambda \end{pmatrix} \Longrightarrow det(M_j - \lambda I) = 0 \Longrightarrow$$

$\lambda = 0$ **or** $\lambda = \pm\sqrt{(}5/8) \Longrightarrow r_\sigma(M_j) = \sqrt{(}5/8) \Longrightarrow$

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \frac{5}{8}}} \approx 1.24$$

Example 2: $\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$

Gauss-Seidel:
$M = -(D+L)^{-1}R = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{4} & -\frac{1}{4} \\ 0 & \frac{1}{8} & \frac{3}{8} \end{pmatrix}$

$\|M\|_\infty = 1$, $\|M\|_1 = 9/8$, $\|M\|_F = 0.884 < 1$

Theorem 2 $\Longrightarrow$ Gauss-Seidel convergent, although $\|M\|_\infty \geq 1$ and $\|M\|_1 \geq 1$

Notice: condition is sufficient, not necessary