Solving $A\bar{x} = \bar{b}$: Iterative methods

- Recap : We have looked at direct methods

  - Gaussian elimination
  - LU decomposition $\rightarrow A^{-1} \rightarrow \bar{x} = A^{-1}\bar{b}$

  In principle, exact methods.

- Alternative approach : Iterative methods.

  - Iterate closer and closer to true solution (but not exact)
  - Often faster, or with smaller mem footprint, than direct methods when matrices are large.

- Iterative scheme : Need to think about req. for the iterative procedure to converge.

- Methods :

  - Jacobi method (not to be confused with Jacobi's rotation method for $A\bar{x} = \lambda\bar{x}$)
  - Gauss-Seidel
  - Over-relaxation methods

- Checking convergence (when to stop iterations)

- Typically, monitor the <u>relative change in vector norm</u>

$$\varepsilon = \left| \frac{|\bar{x}^{(m+1)}|_l - |\bar{x}^{(m)}|_l}{|\bar{x}^{(m)}|_l} \right|$$

- Intuitively: stop when $\bar{x}^{(m+1)}$ is almost identical to $\bar{x}^{(m)}$, i.e. when it doesn't change anymore.

- Can use different norms:

$$|\bar{x}|_l \equiv \left[\sum_{i=1}^{N} |x_i|^l \right]^{1/l}$$

  - $l=1$: $|\bar{x}|_1 = |x_1| + |x_2| + .. + |x_N|$ (sum of abs. vals)
  - $l=2$: $|\bar{x}|_2 = \sqrt{x_1^2 + x_2^2 + .. + x_N^2}$ (Eucl. length)
  - $l=\infty$: $|\bar{x}|_\infty = \max_i (x_i)$ (Max element of $\bar{x}$)

Intuition:

$$0.9^{1000} + 1.2^{1000} + 1.19^{1000} + 0.7^{1000} \approx 1.2^{1000}$$

Formal:

$$\lim_{l \to \infty} \left[x_1^l + x_2^l + .. + x_N^l \right]^{1/l} = \left[\max(x_i)^l \right]^{1/l}$$

$$= \max(x_i)$$

- Alternative way to monitor convergence:

Monitor <u>residual</u> $\bar{r}$:

$$\bar{r} = A\bar{x}^{(n+1)} - \vec{b}$$

Note:

$$\begin{aligned}\bar{r} &= A\bar{x}^{(n+1)} - \bar{b} \\ &= A\bar{x}^{(n+1)} - A\bar{x} \\ &= A(\bar{x}^{(n+1)} - \bar{x}) \\ &= A\,\underbrace{\bar{e}^{(n+1)}}_{\text{error vector}}\end{aligned}$$

- So would check $\dfrac{|\vec{r}|_2}{|\bar{b}_2|}$

at every iteration.

- Pro: This directly monitors how far we are from a vector that satisfies $A\bar{x} = \bar{b}$., not just relative change from $\bar{x}^{(n)}$ to $\bar{x}^{(n+1)}$

- Con: More expensive computationally, since it requires another matrix-vector multiplication.

The Jacobi method (for iterative solution of $A\bar{x} = \bar{b}$)

Summary:

$$\left[\begin{array}{l} \circ \text{ Decompose } A: \quad A = L + D + U \\ \circ \text{ Iterate to find } \bar{x}: \quad \bar{x}^{(m+1)} = -D^{-1}\left[L+U\right]\bar{x}^{(m)} + D^{-1}\bar{b} \\ \circ \text{ Start from some } \underline{\text{guess}} \quad \bar{x}^{(0)} \end{array}\right]$$

- <u>Always</u> converges if A is <u>diagonally dominant</u>, i.e.

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{for all rows } i$$

or A is <u>positive definite</u>, i.e. all eigenvalues > 0.

(<u>Can</u> converge in other cases too.)

- Rearrange A as sum $L + D + U$:

$$\underbrace{\begin{bmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} & & & \\ \bullet & & 0 & \\ \bullet & \bullet & & \\ \bullet & \bullet & \bullet & \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} \bullet & & & \\ & \bullet & & 0 \\ & 0 & \bullet & \\ & & & \bullet \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} & \bullet & \bullet & \bullet \\ & & \bullet & \bullet \\ & 0 & & \bullet \\ & & & \end{bmatrix}}_{U}$$

- Note: Unrelated to LU decomposition, which was a <u>product</u>

$$A = LU$$

- $A\bar{x} = (L + D + U)\bar{x} = \bar{b}$

$$\Rightarrow \quad D\bar{x} = -(L+U)\bar{x} + \bar{b}$$

$$\Rightarrow \quad \bar{x} = -D^{-1}(L+U)\bar{x} + D^{-1}\bar{b}$$

- Treat this as <u>iterative equation</u>, with $\bar{x}^{(m+1)}$ on the left-hand side, and $\bar{x}^{(m)}$ on the right-hand side.

- $T = D^{-1}[L+U]$ is called the "update matrix".

- Trivial to get D^{-1} since D is diagonal:

$$D^{-1} = \operatorname{diag}\left(\frac{1}{d_{11}}, \frac{1}{d_{22}}, \dots, \frac{1}{d_{NN}}\right) = \begin{bmatrix} \frac{1}{d_{11}} & & & \\ & \frac{1}{d_{22}} & & 0 \\ & 0 & \ddots & \\ & & & \frac{1}{d_{NN}} \end{bmatrix}$$

- 4×4 example: $\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$

  - $x_1^{(m+1)} = \left[b_1 - a_{12}x_2^{(m)} - a_{13}x_3^{(m)} - a_{14}x_4^{(m)}\right] / a_{11}$

  - $x_2^{(m+1)} = \left[b_2 - a_{21}x_1^{(m)} - a_{23}x_3^{(m)} - a_{24}x_4^{(m)}\right] / a_{22}$

  - $x_3^{(m+1)} = \left[b_3 - a_{31}x_1^{(m)} - a_{32}x_2^{(m)} - a_{34}x_4^{(m)}\right] / a_{33}$

  - $x_4^{(m+1)} = \left[b_4 - a_{41}x_1^{(m)} - a_{42}x_2^{(m)} - a_{43}x_3^{(m)}\right] / a_{44}$

- In general: $x_i^{(m+1)} = \left[b_i - \sum\limits_{\substack{j=1 \\ j\neq i}}^{N} a_{ij}x_j^{(m)}\right] / a_{ii}$

Gauss – Seidel method

- Consider 4x4 example for the Jacobi method, but now use the (m+1) versions of already computed vector elements :

$$x_1^{(m+1)} = \left[b_1 - a_{12} x_2^{(m)} - a_{13} x_3^{(m)} - a_{14}^{(m)} \right] / a_{11}$$

$$x_2^{(m+1)} = \left[b_2 - a_{21} \underline{x_1^{(m+1)}} - a_{23} x_3^{(m)} - a_{24} x_4^{(m)} \right] / a_{22}$$

$$x_3^{(m+1)} = \left[b_3 - a_{31} \underline{x_1^{(m+1)}} - a_{32} \underline{x_2^{(m+1)}} - a_{34} x_4^{(m)} \right] / a_{33}$$

$$x_4^{(m+1)} = \left[b_4 - a_{41} \underline{x_1^{(m+1)}} - a_{42} \underline{x_2^{(m+1)}} - a_{43} \underline{x_3^{(m+1)}} \right] / a_{44}$$

- We're doing forward subst.
- On matrix form we're rewriting $A\bar{x} = (L + D + U)\bar{x} = \bar{b}$ as

$$D\bar{x} = -L\bar{x} - U\bar{x} + \bar{b}$$

and turning it into an iterative equation as

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

$$\left[\Leftrightarrow \quad \bar{x}^{(m+1)} = -(L+D)^{-1} \cdot U\bar{x}^{(m)} + (L+D)^{-1}\bar{b} \right.$$

but this just looks confusing...]

- Component form :

$$x_i^{(m+1)} = \left[-\sum_{j=1}^{i-1} a_{ij} x_j^{(m+1)} - \sum_{j=i+1}^{N} a_{ij} x_j^{(m)} + b_i \right] / a_{ii}$$

- Must be computed in order
- Typically better conv. than Jacobi's method

Successive over-relaxation (SOR)

- Modified version of Gauss-Seidel, with better convergence
- But: has a free parameter ω that must be chosen/tuned for the specific problem.

- Schematically:

$$[\text{new component}] = [\text{old component}] + [\underset{\omega}{\text{weight}}]\left[\begin{array}{c}\text{change new-old from}\\ \text{Gauss-Seidel algo.}\end{array}\right]$$

- For $\omega = 1$ we get back Gauss-Seidel
- Can be proven: For $\omega \in (1, 2]$ we have <u>better</u> convergence than G-S, but optimal choice is problem specific.

- For $\omega > 2$, SOR fails. ("Wobbles" out of control?)

- Component formulation:

$$x_i^{(m+1)} = x_i^{(m)} + \frac{\omega}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{N} a_{ij}x_j^{(m)} + b_i - a_{ii}x_i^{(m)}\right]$$

$i = 1, 2, \ldots, N$
(evaluated in this order)

- Matrix formulation (confusing-looking)

$$\bar{x}^{(m+1)} = \left[\omega L + D\right]^{-1} \cdot \left[-\left[\omega U + (\omega - 1)D\right]\bar{x}^{(m)} + \omega\bar{b}\right]$$

- Why are iterative methods often useful ?

  - Each iteration only requires matrix-vector multiplication, which has complexity

$$\mathcal{O}(N^1) < \mathcal{O}(N^k) < \mathcal{O}(N^2)$$

If A is diagonal   if A is dense matrix

  - If the method requires M iterations for convergence, we have a total cost of

$$\mathcal{O}(N^k M) \text{ operations}$$

which is typically much less than the typical $\mathcal{O}(N^3)$ cost of direct methods (assuming number of iterations M is smaller than matrix size N)

- Second advantage: Can (in practice) often get more <u>accurate</u> results than direct (exact) methods, because iterative methods are less susceptible to round-off errors.