SOLVING EQUATIONS

Quantitative Economics 2024

Piotr Żoch

October 30, 2024

INTRODUCTION

- Systems of linear equations.
- Nonlinear equations (to be added)

LINEAR SYSTEMS OF EQUATIONS

- One of the most common problems in scientific computation: solve

$$\mathbf{Ax} = \mathbf{b},$$

  for $\mathbf{x}$, where $\mathbf{A}$ is a square matrix and $\mathbf{b}$ is a vector.
- Seems like an easy problem, but it will teach us many things.
- Multiple specialized libraries for numerical linear algebra: LAPACK, BLAS, IMKL...

DIRECT METHODS

- Elementary operations:
  - mutliply a row by a scalar,
  - add a scalar multiple of a row to another row,
  - interchange two rows.
- Solve $\mathbf{Ax} = \mathbf{b}$ by using elementary row operations on the augmented matrix $[\mathbf{A} \mid \mathbf{b}]$.
- Transform $\mathbf{A}$ into a reduced row echelon form.

DIRECT METHODS

- Two step procedure:
  - Forward elimination.

$$\left[\begin{array}{cccc|c} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{array}\right]$$

  - Backward elimination.

$$\left[\begin{array}{cccc|c} * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & * & * & & * \\ & * & * & & * \\ & & * & & * \\ & & & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & * & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \end{array}\right] \to \left[\begin{array}{cccc|c} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \end{array}\right]$$

FLOATING POINT NUMBERS

- Forward elimination: to deal with the first column we need n^2 operations, for the second $n^2 - 1$, for the third $n^2 - 2$ and so on.
- Backward elimination: to deal with the last column we need n operations, for the second to last $n - 1$, for the third to last $n - 2$ and so on.
- Forward elimination is $\mathcal{O}\left(n^3\right)$, backward elimination is $\mathcal{O}\left(n^2\right)$.

TRIANGULAR SYSTEMS

- Lower triangular system:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

  can be solved using forward elimination, starting from the top

$$y_i = b_i - \sum_{j=1}^{i-1} l_{ij} y_j.$$

UPPER TRIANGULAR SYSTEMS

- Upper triangular system:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

  can be solved using backward elimination, starting from the bottom

$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right).$$

- Suppose we can write

$$\mathbf{A} = \mathbf{L}\mathbf{U}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix.

- We have

$$\mathbf{L}\left(\mathbf{U}\mathbf{x}\right) = \mathbf{b} \rightarrow \mathbf{L}\mathbf{y} = \mathbf{b}$$

which we can solve for $\mathbf{y}$ using forward elimination.

- We then solve

$$\mathbf{U}\mathbf{x} = \mathbf{y}$$

for $\mathbf{x}$ using backward elimination.

- This is known as Gaussian elimination.
  1. Compute $\mathbf{L}$ and $\mathbf{U}$.
  2. Solve $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$ using forward elimination.
  3. Solve $\mathbf{Ux} = \mathbf{y}$ for $\mathbf{x}$ using backward elimination.
- Step 1. is known as LU decomposition.
- Nice thing: we can keep $\mathbf{L}$ and $\mathbf{U}$ and recycle them for different $\mathbf{b}$.
- How to perform the decomposition?

MATRIX MULTIPLICATION BY OUTER PRODUCTS

- Write the columns of $\mathbf{A}$ as $\mathbf{a}_1, \ldots, \mathbf{a}_n$.
- Write the rows of $\mathbf{B}$ as $\mathbf{b}_1^\top, \ldots, \mathbf{b}_n^\top$.
- We have

$$\mathbf{AB} = \sum_{k=1}^{n} \mathbf{a}_k \mathbf{b}_k^\top.$$

- Useful: for triangular matrices $\mathbf{L}$, $\mathbf{U}$ only the first outer product contributes to the first row and the first column of $\mathbf{LU}$

$$\mathbf{e}_1^\top \sum_{k=1}^{n} \mathbf{l}_k \mathbf{u}_k^\top = l_{11} \mathbf{u}_1^\top, \quad \left(\sum_{k=1}^{n} \mathbf{l}_k \mathbf{u}_k^\top \right) \mathbf{e}_1 = u_{11} \mathbf{l}_1.$$

LU FACTORIZATION WITHOUT PIVOTING

Algorithm LU Factorization without Pivoting

Require: Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$

for $j = 1$ to n **do**
  for $i = j + 1$ to n **do**
    $a_{ij} = \frac{a_{ij}}{a_{jj}}$
    for $k = j + 1$ to n **do**
      $a_{ik} = a_{ik} - a_{ij} a_{jk}$
    end for
  end for
end for

- This algorithm uses **A** matrix to store **L** and **U**.
- Problem when $a_{jj} = 0$ at any step -

SOLVING THE SYSTEM

- We need

$$\sum_{j=1}^{n}\sum_{i=j+1}^{n}\left(1+\sum_{k=j+1}^{n}2\right)=\frac{2}{3}n^3-\frac{1}{2}n^2-\frac{1}{6}n$$

  operations to perform the factorization.
- We then need

$$\sum_{i=1}^{n}\left(2+\sum_{j=1}^{i-1}2\right)=n^2+n$$

  operations for forward and backward substitution each.
- LU decomposition is the most costly step.

SOLVING THE SYSTEM

- We need

$$\sum_{j=1}^{n} \sum_{i=j+1}^{n} \left(1 + \sum_{k=j+1}^{n} 2\right) = \frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$$

  operations to perform the factorization.

- We then need

$$\sum_{i=1}^{n} \left(2 + \sum_{j=1}^{i-1} 2\right) = n^2 + n$$

  operations for forward and backward substitution each.

- LU decomposition is the most costly step.

SOLVING THE SYSTEM

- In practice we use a similar method, but with pivoting
- PLU factorization:

$$\tilde{\mathbf{A}} = \mathbf{L}\mathbf{U},$$

  where $\tilde{\mathbf{A}}$ is a matrix $\mathbf{A}$ with its rows permuted.
- It works if and only if $\mathbf{A}$ is non-singular.
- Asymptotically uses the same number of operations and LU without pivoting.

NORMS

- A vector norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ that satisfies:
  1. $\|\mathbf{x}\| \geq 0$,
  2. $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$,
  3. $\|a\mathbf{x}\| = |a| \, \|\mathbf{x}\|$,
  4. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$,

  for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $a \in \mathbb{R}$.
- Common vector norms: $\ell_1, \ell_2, \ell_\infty$:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|, \quad \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}, \quad \|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

NORMS

- For matrices we have matrix norms.
- A Frobenius norm is

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}.$$

- Imagine representing a matrix as a vector with columns stacked on top of each other.
- An induced matrix norm is

$$\|\mathbf{A}\|_p = \max_{\|\mathbf{x}\|_p=1} \|\mathbf{A}\mathbf{x}\|_p .$$

- In Julia: `norm(A)` is the Frobenius norm, `opnorm(A,p)` is the induced norm.

NORMS

- We have
  1. $\|\mathbf{Ax}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\|$,
  2. $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$,
  3. for a square matrix, $\left\|\mathbf{A}^k\right\| \leq \|\mathbf{A}\|^k$ for any integer $k \geq 0$.
- Two common matrix norm are the 1-norm and the ∞-norm:

$$\|\mathbf{A}\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|, \quad \|\mathbf{A}\|_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}|.$$

CONDITIONING OF LINEAR SYSTEMS

- Consider the perturbed system

$$\mathbf{A}\left(\mathbf{x}+\mathbf{h}\right)=\mathbf{b}+\mathbf{d}.$$

- The condition number is the relative change in the solution divided by the relative change in the data:

$$\kappa=\frac{\|\mathbf{h}\|\,/\,\|\mathbf{x}\|}{\|\mathbf{d}\|\,/\,\|\mathbf{b}\|}=\frac{\|\mathbf{h}\|\,\|\mathbf{b}\|}{\|\mathbf{d}\|\,\|\mathbf{x}\|}.$$

- Note that $\mathbf{h}=\mathbf{A}^{-1}\mathbf{d}$ so

$$\|\mathbf{h}\|\leq\left\|\mathbf{A}^{-1}\right\|\,\|\mathbf{d}\|\,.$$

CONDITIONING OF LINEAR SYSTEMS

- Use $\|\mathbf{h}\| \leq \left\|\mathbf{A}^{-1}\right\| \|\mathbf{d}\|$ to write

$$\frac{\|\mathbf{h}\| \|\mathbf{b}\|}{\|\mathbf{d}\| \|\mathbf{x}\|} \leq \frac{\left\|\mathbf{A}^{-1}\right\| \|\mathbf{d}\| \|\mathbf{A}\| \|\mathbf{x}\|}{\|\mathbf{d}\| \|\mathbf{x}\|} = \left\|\mathbf{A}^{-1}\right\| \|\mathbf{A}\| .$$

- We can prove that inequality is tight.
- The matrix condition number of an invertible square matrix $\mathbf{A}$ is

$$\kappa(\mathbf{A}) = \left\|\mathbf{A}^{-1}\right\| \|\mathbf{A}\| .$$

CONDITIONING OF LINEAR SYSTEMS

- If $\mathbf{A}\left(\mathbf{x}+\Delta\mathbf{x}\right)=\mathbf{b}+\Delta\mathbf{b}$ then

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|}\leq\kappa\left(\mathbf{A}\right)\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}.$$

- The condition number is a measure of how sensitive the solution is to changes in the data.
- We can derive a similar result for perturbed $\mathbf{A}$.
- The condition number is at least equal to 1.
- A condition number of 10^k means that we lose k digits of precision.

CONDITIONING OF LINEAR SYSTEMS

- Suppose that we compute a "solution" $\tilde{\mathbf{x}}$ to the system $\mathbf{A}\mathbf{x} = \mathbf{b}$.
- We would like to compare $\tilde{\mathbf{x}}$ to the true solution $\mathbf{x}$ - but we do not know $\mathbf{x}$.
- We can calculate the residual

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$$

.

- We have

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa\left(\mathbf{A}\right)\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

ITERATIVE METHODS

- We saw that Gaussian elimination is $\mathcal{O}(n^3)$.
- This is prohibitive for large n (unless a matrix has a special structure).
- But matrix-vector multiplication is $\mathcal{O}(n^2)$.
- In some cases we can apply repeated matrix-vector multiplication to solve the system.
- We call these iterative methods.

JACOBI METHOD

- Suppose we want to solve $\mathbf{Ax} = \mathbf{b}$.
- This can be written as

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad \text{or} \quad x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j \right).$$

- Start from some initial $\mathbf{x}^{(0)}$ and iterate:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right)$$

until $\mathbf{x}^{(k+1)}$ is close enough to $\mathbf{x}^{(k)}$.

GAUSS-SEIDEL METHOD

- We can also write

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

- The Gauss-Seidel method uses the newest values of x_j:

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

ITERATIVE METHODS

- Rewrite $\mathbf{A} = \mathbf{P} - \mathbf{N}$ so that the system is

$$\mathbf{P}\mathbf{x} = \mathbf{N}\mathbf{x} + \mathbf{b}.$$

- An iterative method is

$$\mathbf{P}\mathbf{x}^{(k+1)} = \left(\mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}\right).$$

- $\mathbf{P}$ is called a preconditioner
- Jacobi and Gauss-Seidel differ in the choice of $\mathbf{P}$.
- Since $\mathbf{x}^{k+1} = \mathbf{P}^{-1}\left(\mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}\right)$ a good preconditioner should be easy to invert.

- Why do we call $\mathbf{P}$ a preconditioner?
- Suppose we have a system $\mathbf{Ax} = \mathbf{b}$.
- The condition number is $\kappa(\mathbf{A})$.
- Suppose we have a preconditioned system $\mathbf{P}^{-1}\mathbf{Ax} = \mathbf{P}^{-1}\mathbf{b}$.
- The condition number is $\kappa\left(\mathbf{P}^{-1}\mathbf{A}\right)$.
- If $\mathbf{P} \approx \mathbf{A}^{-}1$, then $\kappa\left(\mathbf{P}^{-1}\mathbf{A}\right) \approx 1$.

ITERATIVE METHODS

- Iterative methods do not always converge.
- They are guaranteed to converge if **A** is diagonally dominant:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|.$$

- There are better methods than Jacobi and Gauss-Seidel - the idea is to choose **P** so that the convergence is faster.