The following is the definition of the residual.

$$\boldsymbol{r = b - A\hat{x}.}$$

Remark 0.1. The residual itself does not reveal much. Suppose we calculate $r = b - Ax$. Now solve for $kAx = kb$ and the residual required to solve that is k times as great. This is why we define the relative residual:

$$\frac{\|r\|}{\|A\| \cdot \|\hat{x}\|}$$

We can obtain a bound on the relative forward error required to solve $Ax = b$ in terms of r.

$$\|\Delta \boldsymbol{x}\| = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\| = \|\boldsymbol{A}^{-1}(\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{b})\| = \| - \boldsymbol{A}^{-1}\boldsymbol{r}\| \leq \|\boldsymbol{A}^{-1}\| \cdot \|\boldsymbol{r}\|.$$

Dividing both sides by $\|\hat{\boldsymbol{x}}\|$ and using the definition of $\text{cond}(\boldsymbol{A})$, we then have

$$\frac{\|\Delta \boldsymbol{x}\|}{\|\hat{\boldsymbol{x}}\|} \leq \text{cond}(\boldsymbol{A}) \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\| \cdot \|\hat{\boldsymbol{x}}\|}.$$

Remark 0.2. This bound tells us that if the residual is small and the matrix and well conditioned, then the relative error is low.

Example 2.8 Small Residual. Consider the linear system

$$\boldsymbol{Ax} = \begin{bmatrix} 0.913 & 0.659 \\ 0.457 & 0.330 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.254 \\ 0.127 \end{bmatrix} = \boldsymbol{b},$$

whose matrix we saw in Example 2.7. Consider two approximate solutions

$$\hat{\boldsymbol{x}}_1 = \begin{bmatrix} 0.6391 \\ -0.5 \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{x}}_2 = \begin{bmatrix} 0.999 \\ -1.001 \end{bmatrix}.$$

The norms of their respective residuals are

$$\|\boldsymbol{r}_1\|_1 = 7.0 \times 10^{-5} \quad \text{and} \quad \|\boldsymbol{r}_2\|_1 = 2.4 \times 10^{-2}.$$

So which is the better solution? We are tempted to say $\hat{\boldsymbol{x}}_1$ because of its much smaller residual. But the exact solution to this system is $\boldsymbol{x} = [1, \ -1]^T$, as is easily confirmed, so $\hat{\boldsymbol{x}}_2$ is actually much more accurate than $\hat{\boldsymbol{x}}_1$. The reason for this surprising behavior is that the matrix $\boldsymbol{A}$ is ill-conditioned, as we saw in Example 2.7, and because of its large condition number, a small residual does not imply a small error in the solution. To see how $\hat{\boldsymbol{x}}_1$ was obtained, see Example 2.17.

Demo: Vanilla Gaussian Elimination

What do we get by doing Gaussian Elimination?

> *Row Echelon Form.*

How is that different from being upper triangular?

> Zeros allowed on and above the diagonal.

What if we do not just eliminate downward but also upward?

> That's called *Gauss-Jordan elimination*. Turns out to be computationally inefficient. We won't look at it.

Remark 0.3. Also note that a matrix is in row echelon form if the first non-zero entry of each row (what was the pivot during gaussian elimination) is to the first of the first non-zero entry of any preceding row; moreover, entries in rows above the pivot (but in the same column) must be 0.

Elimination Matrices

What does this matrix do?

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ -\frac{1}{2} & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

- Add $(-1/2)\times$ the first row to the third row.
- One elementary step in Gaussian elimination
- Matrices like this are called *Elimination Matrices*

Remark 0.4. If we add k to the identity matrix at entry i, j, and left multiply the resultant matrix C by some matrix of interest A, then the result is to take the j th row of A multiply it by k and then add it to i. We can undo this process by using the same matrix but, in place of k, using $-k$. This second matrix is the inverse to the elimination matrix C.

Elimination Matrices

What does this matrix do?

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ -\frac{1}{2} & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{pmatrix}$$

- Add $(-1/2)\times$ the first row to the third row.
- One elementary step in Gaussian elimination
- Matrices like this are called *Elimination Matrices*

Remark 0.5. Suppose that we multiply A by an elimination matrix M_1, then by M_2 up to M_l, where M_l is the last matrix required to turn A into Row Echelon Form. Eventually, we will have

$$(M_l \dots M_1)A = U \implies A = (M_l \dots M_1)^{-1}U$$

At first glance, this is okay, because it turns out that left multiplication of an elimination matrix X by Y such that X has a non-zero off diagonal at column i and Y has a non-zero off diagonal at column j where $i < j$ results in an elimination matrix that just merges X and Y [1]

For whatever reason, pivoting foils this attempt:

No, very much not:

$$A = \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix}.$$

Q: Is this a problem with the process or with the entire *idea* of LU?

$$\begin{bmatrix} 1 & \\ \ell_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ & u_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 1 \end{bmatrix} \rightarrow \quad u_{11} = 0$$

$$\underbrace{u_{11} \cdot \ell_{21}}_{0} + 1 \cdot 0 = 2$$

It turns out to be that A doesn't *have* an LU factorization.

[1] Note that merging also takes place if we multiply two elimination matrices that have their off diagonal non-zero entry in the same column as each other.

The solution is to repeatedly apply permutations to A (in the form of permutation matrices) so that the pivot is the largest element in terms of absolute value in its column.

Thus, we now have

$$(M_l P_l \dots M_1 P_1)A = U \implies A = (M_l P_l \dots M_1 P_1)^{-1} U$$

However, what should be L above is not always left triangular. It can be shown that a factorization of $(M_l P_l \dots M_1 P_1)^{-1}$ does, however, give us a lower triangular system.

Sort out what LU with pivoting looks like. Have: $M_3 P_3 M_2 P_2 M_1 P_1 A = U$.

Define: $L_3 := M_3$
Define $L_2 := P_3 M_2 P_3^{-1}$
Define $L_1 := P_3 P_2 M_1 P_2^{-1} P_3^{-1}$

$$\begin{aligned}&(L_3 L_2 L_1)(P_3 P_2 P_1)\\ =&M_3(P_3 M_2 P_3^{-1})(P_3 P_2 M_1 P_2^{-1} P_3^{-1})P_3 P_2 P_1\\ =&M_3 P_3 M_2 P_2 M_1 P_1 \quad (!)\end{aligned}$$

$$\underbrace{P_3 P_2 P_1}_{P} A = \underbrace{L_1^{-1} L_2^{-1} L_3^{-1}}_{L} U.$$

$L_1, \dots, L_3$ are still lower triangular!

Q: Outline the solve process with pivoted LU

Changing Condition Numbers

Once we have a matrix A in a linear system $A\mathbf{x} = \mathbf{b}$, are we stuck with its condition number? Or could we improve it?

Diagonal scaling is a simple strategy that sometimes helps.

- Row-wise: $DA\mathbf{x} = D\mathbf{b}$
- Column-wise: $AD\hat{\mathbf{x}} = \mathbf{b}$
  Different $\hat{\mathbf{x}}$: Recover $\mathbf{x} = D\hat{\mathbf{x}}$.

What is this called as a general concept?

Preconditioning

- Left preconditioning: $MA\mathbf{x} = M\mathbf{b}$
- Right preconditioning: $AM\hat{\mathbf{x}} = \mathbf{b}$
  Different $\hat{\mathbf{x}}$: Recover $\mathbf{x} = M\hat{\mathbf{x}}$.

Remark 0.6. Suppose that D above satisfies $k(D) \approx 1$. Then

$$k(DA) = \|DA\|\|(DA)^{-1}\| \leq \|D\|\|A\|\|A^{-1}\|\|D^{-1}\| \leq k(A)$$

so that the condition number of $K(DA)$ is no greater than the condition number of A.

Assuming that D is invertible, then the set of x satisfying $Ax = b$ is precisely the set of x satisfying $Ax = b$. Left multiplication by D of A is called, understandably, left preconditioning and scales A in a row-wise manner; right multiplication by D of A is called right preconditioning.

Remark 0.7.

Computational Cost

What is the computational cost of multiplying two $n \times n$ matrices?

$O(n^3)$

What is the computational cost of carrying out LU factorization on an $n \times n$ matrix?

Recall

$$M_3 P_3 M_2 P_2 M_1 P_1 A = U \ldots$$

so $O(n^4)$?!!!

Fortunately not: Multiplications with permuation matrices and elimination matrices only cost $O(n^2)$.

So overall cost of LU is just $O(n^3)$.

Demo: Complexity of Mat-Mat multiplication and LU

Multiplication by a permutation matrix is only an n operation, since it involves switching rows. Multiplication by an elimination matrix simply involves scaling one row and mulitplying it by another, and this process is done at most n times for any one elimination matrix (making it $O(n^2)$ as well). Since these transformations are applied at most n times, the process of getting a matrix into LU form is only $O(n^3)$.

Remark 0.8.

LU on Blocks: The Schur Complement

Given a matrix

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

can we do 'block LU' to get a *block triangular matrix*?

Multiply the top row by $-CA^{-1}$, add to second row, gives:

$$\begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix}.$$

$D - CA^{-1}B$ is called the Schur complement. Block pivoting is also possible if needed.

Not sure why this is significant.

Remark 0.9. Unresolved

Example 2.15 Small Pivots. Using finite-precision arithmetic, we must avoid not only zero pivots but also *small* pivots in order to prevent unacceptable error growth, as shown in the following example. Let

$$\boldsymbol{A} = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix},$$

where ϵ is a positive number smaller than the unit roundoff ϵ_{mach} in a given floating-point system. If we do not interchange rows, then the pivot is ϵ and the resulting

multiplier is $-1/\epsilon$, so that we get the elimination matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ -1/\epsilon & 1 \end{bmatrix},$$

and hence

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$$

in floating-point arithmetic. But then

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} \neq \boldsymbol{A}.$$

Using a small pivot, and a correspondingly large multiplier, has caused an unrecoverable loss of information in the transformed matrix. If we interchange rows, on the other hand, then the pivot is 1 and the resulting multiplier is $-\epsilon$, so that we get the elimination matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ -\epsilon & 1 \end{bmatrix},$$

and hence

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{U} = \begin{bmatrix} 1 & 1 \\ 0 & 1 - \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

in floating-point arithmetic. We therefore have

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix},$$

Remark 0.10. Notice that if we already have an LU factorization, then computing a rank 1 update is just an $O(n^2)$ operation.

Changing matrices

Seen: LU cheap to re-solve if RHS changes. (Able to keep the expensive bit, the LU factorization) What if the *matrix* changes?

Special cases allow something to be done (a so-called *rank-one update*):

$$\hat{A} = A + \mathbf{u}\mathbf{v}^T$$

The Sherman-Morrison formula gives us

$$(A + \mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1 + \mathbf{v}^T A^{-1}\mathbf{u}}.$$

Proof: Multiply the above by $\hat{A}$ get the identity.
FYI: There is a rank-k analog called the Sherman-Morrison-Woodbury formula.

Demo: Sherman-Morrison

For

$$\left(A + uv^T\right)^{-1} b = A^{-1}b - \frac{\left(A^{-1}u\right) v^T A^{-1} b}{1 + v^T A^{-1} u}$$

And $A^{-1}x$ for any x is an $O(n^2)$ operation. The only other operation in this formula is to compute a dot product.