These are the lecture notes for CSC349A Numerical Analysis taught by Rich Little. They roughly correspond to the material covered in each lecture in the classroom but the actual classroom presentation might deviate significantly from them depending on the flow of the course delivery. They are provide as a reference to the instructor as well as supporting material for students who miss the lectures. They are simply notes to support the lecture so the text is not detailed and they are not thoroughly checked. Use at your own risk. They are complimentary to the handouts. Many thanks to all the guidance and materials I received from Dale Olesky who has taught this course for many years and George Tzanetakis.

1 Gaussian Elimination with Partial Pivoting

The Naive Gaussian elimination algorithm will fail if any of the pivots $a_{11}, a_{22}^{(1)}, a_{33}^{(2)}, ...$ is equal to 0. Mathematically, it works provided this does not occur. Algorithmically, it breaks down when the pivots are even close to 0 because of floating-point arithmetic. The problem occurs in the multiplier, it becomes far larger than the other entries

1.1 Example 1

Consider the $n = 2$ linear system with augmented matrix with $k = 4$, $b = 10$, rounding, floating-point arithmetic.

$$\left[\begin{array}{cc|c} 0.003 & 59.14 & 59.17 \\ 5.291 & -6.13 & 46.78 \end{array}\right]$$

Note the the exact solution to this system is $x = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$. The multiplier is

$$m_{21} = fl(a_{21}/a_{11}) = fl(5.291/0.003) = fl(1763.666...) = 1764$$

Thus, forward elimination gives

$$\begin{aligned} a_{22} &= fl(-6.13 - fl(1764 * 59.14)) \\ &= fl(-6.13 - fl(104322.96)) \\ &= fl(-6.13 - 104300) \\ &= fl(-104306.13) \\ &= -104300 \end{aligned}$$

and

$$\begin{aligned} b_2 &= fl(46.78 - fl(1764 * 59.17)) \\ &= fl(46.78 - fl(104375.88)) \\ &= fl(46.78 - 104400) \\ &= fl(-104353.22) \\ &= -104400 \end{aligned}$$

The new augmented matirix is

$$\left[\begin{array}{rr|r} 0.003 & 59.14 & 59.17 \\ 0 & -104300 & -104400 \end{array}\right]$$

Finally, we do back substitution giving

$$\begin{aligned} x_2 &= fl(-104400/-104300) = fl(1.000958...) = 1.001 \\ x_1 &= fl\left(\frac{b_1 - a_{12}x_2}{a_{11}}\right) \\ &= fl\left(\frac{fl(59.17 - fl(59.14 * 1.001))}{0.003}\right) \\ &= fl\left(\frac{fl(59.17 - 59.20)}{0.003}\right) \\ &= fl\left(\frac{-0.03}{0.003}\right) \\ &= -10 \end{aligned}$$

Therefore, $\hat{x} = \begin{bmatrix} -10 \\ 1.001 \end{bmatrix}$.

Analysis of the above Example: The relative error in $\hat{x}_1$ is 200%. The source of the extremely inaccurate computed solution $\hat{x}$ is the **large magnitude of the multiplier**. Here, 1764 is much larger than the rest of the numbers in the system. This number is large because the pivot, $a_{11} = 0.003$, is much smaller than the other numbers in the system. Consequently, in the floating-point computations of $a_{22}^{(1)}$ and $b_2^{(1)}$, the numbers -6.13 and 46.78 are so small they are lost. The **partial pivoting strategy** is designed to avoid the selection of small pivots.

2 Partial Pivoting

At step k of forward elimination, where $1 \leq k \leq n-1$, choose the pivot to be the **largest entry in absolute value**, from

$$\begin{bmatrix} a_{kk} \\ a_{k+1,k} \\ a_{k+2,k} \\ \vdots \\ a_{n,k} \end{bmatrix}$$

If a_{pk} is the largest (that is, $|a_{pk}| = \max_{k \leq i \leq n} |a_{ik}|$), then switch row k with row p. Note that $|mult| \leq 1$ for all multipliers since the denominator is always the largest value. Note also that switching rows does not change the final solution. It is an elementary row operation of type 3.

Algorithm 1 pseudocode for partial pivoting

```
for $k = 1$ to $n - 1$ do
    $p = k$
    for $i = k + 1$ to $n$ do
        Find largest pivot
    end for
    if $p \neq k$ then
        for $j = k$ to $n$ do
            swap $a_{kj}$ and $a_{pj}$
        end for
        swap $b_k$ and $b_p$
    end if
    do forward elimination
end for
do back susbstitution
```

2.1 Example 2

Consider the $n = 2$ linear system with augmented matrix with $k = 4$, $b = 10$, rounding, floating-point arithmetic using partial pivoting.

$$\left[\begin{array}{cc|c} 0.003 & 59.14 & 59.17 \\ 5.291 & -6.13 & 46.78 \end{array}\right]$$

Note the the exact solution to this system is $x = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$. Because $|5.291| > |0.003|$, we swap rows (equations) 1 and 2, thus

$$\left[\begin{array}{cc|c} 5.291 & -6.13 & 46.78 \\ 0.003 & 59.14 & 59.17 \end{array}\right]$$

The multiplier is

$$m_{21} = fl(a_{21}/a_{11}) = fl(0.003/5.291) = fl(0.000567) = 0.000567$$

Thus, forward elimination gives

$$\begin{aligned} a_{22} &= fl(59.14 - fl(0.000567 * (-6.13))) \\ &= fl(59.14 - fl(-0.003475713)) \\ &= fl(59.14 + 0.003476) \\ &= fl(59.143476) \\ &= 59.14 \end{aligned}$$

and

$$\begin{aligned} b_2 &= fl(59.17 - fl(0.000567 * 46.78)) \\ &= fl(59.17 - fl(0.02652426)) \\ &= fl(59.17 - 0.02652) \\ &= fl(59.14348) \\ &= 59.14 \end{aligned}$$

The new augmented matirix is

$$\left[\begin{array}{rr|r} 5.291 & -6.13 & 46.78 \\ 0 & 59.14 & 59.14 \end{array}\right]$$

Finally, we do back substitution giving

$$\begin{aligned} x_2 &= fl(59.14/59.14) = 1 \\ x_1 &= fl\left(\frac{b_1 - a_{12}x_2}{a_{11}}\right) \\ &= fl\left(\frac{fl(46.78 - fl(-6.13 * 1))}{5.291}\right) \\ &= fl\left(\frac{fl(46.78 + 6.13)}{5.291}\right) \\ &= fl\left(\frac{52.91}{5.291}\right) \\ &= 10 \end{aligned}$$

Therefore, $\hat{x} = \begin{bmatrix} 10 \\ 1 \end{bmatrix}$ which is exactly correct.

3 Gaussian Elimination with Scaling

If the entries of maximum absolute value in different rows (equations) differ greatly, the computed solution (using floating point arithmetic and partial pivoting) can be very inaccurate.

3.1 Example 3

Using $k = 4$ precision, floating-point arithmetic with rounding, solve the following system by Gaussian Elimination with partial pivoting.

$$\left[\begin{array}{cc|c} 2 & 100,000 & 100,000 \\ 1 & 1 & 2 \end{array}\right]$$

Note the the exact solution to this system is $x = \begin{bmatrix} 1.00002 \\ 0.99998 \end{bmatrix}$. Because $|2| > |1|$, we leave the matrix as is.

The multiplier is

$$m_{21} = fl(a_{21}/a_{11}) = fl(1/2) = 0.5$$

Thus, forward elimination gives

$$\begin{aligned} a_{22} &= fl(1 - 0.5(100000)) \\ &= fl(1 - 50000) \\ &= fl(-49999) \\ &= -50000 \end{aligned}$$

and

$$\begin{aligned} b_2 &= fl(2 - 0.5(100000)) \\ &= fl(2 - 50000) \\ &= fl(-49998) \\ &= -50000 \end{aligned}$$

The new augmented matirix is

$$\left[\begin{array}{cc|c} 2 & 100,000 & 100,000 \\ 0 & -50000 & -50000 \end{array}\right]$$

Finally, we do back substitution giving

$$\begin{aligned} x_2 &= 1 \\ x_1 &= 0 \end{aligned}$$

Therefore, $\hat{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, which is 100% incorrect for x_1.

Scaling

We look at two ways of using **scaling** to solve this problem: (1) Equilibration and (2) Scaled Factors.

(1) Equilibration: On way of solving this problem is to multiply each row by a nonzero constant so that the largest entry in each row of A has magnitude of 1. That is, we add the third elemntary row operation. But, there is a problem with this form of scaling, it introduces another source of round-off error (more computations). Turns out there is another way to accomplish the stability we need without adding more computations to the system.

(2) Scaled Factors: Use the scaling factors to pick pivots but NOT actually scaling. Let $s_i = \max_{1 \leq j \leq n} |a_{ij}|$ for $i = 1, 2, \ldots, n$. Step $i = 1$: pivot is max of

$$\begin{bmatrix} |a_{11}/s_1| \\ |a_{21}/s_2| \\ \vdots \\ |a_{n1}/s_n| \end{bmatrix}$$

If $|a_{p1}/s_p|$ is the max then interchange rows 1 and p then do forward elimination step.

Step $i = 2$: pivot is max of

$$\begin{bmatrix} |a_{22}^{(1)}/s_2| \\ |a_{32}^{(1)}/s_3| \\ \vdots \\ |a_{n2}^{(1)}/s_n| \end{bmatrix}$$

If $|a_{q2}/s_q|$ is the max then interchange rows 2 and q then do forward elimination step. etc. Finish with back susbstitution as usual.

3.2 Example 4

Using $k = 4$ precision, floating-point arithmetic with rounding, solve the following system by Gaussian Elimination with partial pivoting and scaled factors.

$$\left[\begin{array}{cc|c} 2 & 100,000 & 100,000 \\ 1 & 1 & 2 \end{array}\right]$$

Recall, the the exact solution to this system is $x = \begin{bmatrix} 1.00002 \\ 0.99998 \end{bmatrix}$.

The scaling factors for this example are $s_1 = 100000$ and $s_2 = 1$. Because $|2/100000| < |1/1|$, we swap rows 1 and 2, resulting in new matrix

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 100,000 & 100,000 \end{array}\right]$$

The multiplier is

$$m_{21} = fl(a_{21}/a_{11}) = fl(2/1) = 2$$

Thus, forward elimination gives

$$\begin{aligned} a_{22} &= fl(100000 - 2(1)) \\ &= fl(99998) \\ &= 100000 \end{aligned}$$

and

$$\begin{aligned} b_2 &= fl(100000 - 2(2)) \\ &= fl(99996) \\ &= 100000 \end{aligned}$$

The new augmented matirix is

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & 100000 & 100000 \end{array}\right]$$

Finally, we do back substitution giving

$$\begin{aligned} x_2 &= 1 \\ x_1 &= (2 - (1)(1))/1 = 1 \end{aligned}$$

Therefore, $\hat{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, which is very close to the correct solution.

4 Determinant of A

The reduction of A to upper triangular form by **Naive Gaussian elimination** uses only the type 2 elementary row operation

$$E_i = E_i - factor \times E_j.$$

This row operation does not change the value of the determinant of A. That is, if no rows are interchanged then,

$$\det A = a_{11} a_{22}^{(1)} a_{33}^{(2)} \cdots a_{nn}^{(n-1)}$$

since the determinant of a triangular matrix is equal to the product of its diagonal entries.

However, if **Gaussian elimination with partial pivoting** is used, then each row interchange causes the determinant to change signs (that is, determinant is multiplied by -1.)

Thus, if m row interchanges are done during the reduction of A to upper triangular form, then

$$\det A = (-1)^m a_{11} a_{22}^{(1)} a_{33}^{(2)} \cdots a_{nn}^{(n-1)}$$

As a consequence, Gaussian elimination provides us with a simple method of calculating the determinant of a matrix.

5 Stability and Condition of Systems of Linear Equations

5.1 Stability of Algorithms for Solving $Ax = b$

Given a nonsingular matrix A, a vector b and some algorithm for computing the solution of $Ax = b$, let $\hat{x}$ denote the computed solution using this algorithm. The computation is said to be stable if there exist small perturbations E and e of A and b, respectively, such that $\hat{x}$ is close to the exact solution y of the perturbed linear system

$$(A+E)y = b+e$$

That is, the computed solution $\hat{x}$ is very close to the exact solution of some small perturbation of the given problem.

Known Results Gaussian elimination without pivoting may be unstable. In practice, Gaussian elimination with partial pivoting is almost always stable. A much more stable version of Gaussian elimination uses complete pivoting, which uses both row and column interchanges. However, as this algorithm is much more expensive to implement and since partial pivoting is almost always stable, complete pivoting is seldom used.

5.2 Condition of $Ax = b$

A given problem $Ax = b$ is ill-conditioned if its exact solution is very sensitive to small changes in the data $[A|b]$. That is, if there exist small perturbations E and e of A and b, respectively, such that $x = A^{-1}b$ is not close to the exact solution y of the perturbed linear system

$$(A+E)y = b+e,$$

then the linear system $Ax = b$ is ill-conditioned. If such perturbations E and e do not exist, then $Ax = b$ is well conditioned.

Example 2: Recall, the linear system $Hx = b$, with $H = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}$, $b = \begin{bmatrix} 11/6 \\ 13/12 \\ 47/60 \end{bmatrix}$ and solution $x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, is ill-conditioned.

We do this by perturbing H and b as follows. Let $(H+E) = \begin{bmatrix} 1 & 1/2 & 0.333 \\ 1/2 & 0.333 & 1/4 \\ 0.333 & 1/4 & 1/5 \end{bmatrix}$, $(b+e) = \begin{bmatrix} 1.83 \\ 1.08 \\ 0.783 \end{bmatrix}$, and solving to get $y = \begin{bmatrix} 1.0895... \\ 0.48796... \\ 1.4910.... \end{bmatrix}$. We see from this that y is about 50% incorrect for two of the three values, for small input perturbations.

Matrix Norms and Condition Number

The *norm* of a matrix (or vector) is a measure of the "size" of the matrix. We denote the norm of a matrix A by $\|A\|$. There exist a number of different ways of calculating a norm.

- $\|A\|_e = \sqrt{\sum\sum a_{ij}^2}$ is the Euclidean norm.
- $\|A\|_\infty = \max\limits_{1\leq i\leq n} \sum |a_{ij}|$ is the uniform norm, etc.

Turns out, for any of these forms of the norm, when solving for $Ax = b$,

$$\frac{\|x-y\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|e\|}{\|b\|}$$

The condition number of a matrix, A, is given by

$$cond[A] = \|A\|\|A^{-1}\|$$

Properties of the condition number.

- $cond[A] \geq 1$
- $cond[I] = 1$

As usual, the higher the condition number the more ill-conditioned it is, but the range of well-conditioned matrices is bigger. For example, if consider $k = 4$, $b = 10$, floating-point, then a condition number between 1 and 100 is considered well-conditioned.

Example 3: We verify Example 2 using the condition number, with uniform norms. Recall,

$$H = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix}$$

From this we get that,

$$\begin{aligned} \|H\|_\infty &= \max(1 + 1/2 + 1/3, 1/2 + 1/3 + 1/4, 1/3 + 1/4 + 1/5) \\ &= \max(1.833, 1.0833, 0.7533) = 1.833 \end{aligned}$$

Note that (from MATLAB),

$$H^{-1} = \begin{bmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{bmatrix}$$

So,

$$\begin{aligned} \|H^{-1}\|_\infty &= \max\left(9 + 36 + 30, 36 + 192 + 180, 30 + 180 + 180\right) \\ &= \max\left(75, 408, 390\right) = 408 \end{aligned}$$

Finally,

$$cond[H] = \|H\|\|H^{-1}\| = 1.833 \times 408 = 747.864$$

and thus H is ill-conditioned since the condition number is much larger than 100.