Adequacy in Solution of Simultaneous Linear Equations

University of South Florida

United States of America

kaw@eng.usf.edu

NOTE: This worksheet demonstrates the use of Maple to illustrate three different techniques in finding the condition number of a coefficient matrix $[A]_{nxn}$.

```
> restart;
```

Introduction

The condition number allows one to quantify the accuracy in solution of [A] [X] = [C], where $[A]_{nxn}$ is an invertible square matrix, $[X]_{nx1}$ is the solution vector, and $[C]_{nx1}$ is the right hand side array. Multiply the condition number by machine epsilon, and compare the result to 0.5×10^{-m} to find out at least how many m significant digits are correct in solution.

$$0.5\text{x}10^{-m} < Cond(A) \cdot (machine\ \ \varepsilon)$$

$$0.5\ x\ 10^{(-m)} < Cond(A)\ machine\ \varepsilon \qquad (1.1)$$

To learn more about the relationship between condition number and the adequacy of solutions, click here.

The following simulation uses three different techniques to determine the condition number of coefficient matrix $[A]_{nxn}$.

Section 1: Input Data

Below are the input parameters to begin the simulation. This is the only section that requires user input. Once the values are entered, Maple will calculate the condition number using an exact method as well as two approximate methods.

Input Parameters:
A = nxn invertible square matrix
n = number of equations
Bits_used_for_mantissa = number of bits used for mantissa in floating point representation

```
> restart;
> A:=Matrix([[8,12,1,7],[1,2,3.001,6],[57,9,1,4],[1.047,108,98,5]]
  );
  n:=4;
  Bits_used_for_mantissa:=23;
```

$$A := \begin{bmatrix} 8 & 12 & 1 & 7 \\ 1 & 2 & 3.001 & 6 \\ 57 & 9 & 1 & 4 \\ 1.047 & 108 & 98 & 5 \end{bmatrix}$$

$$n := 4$$

$$Bits\_used\_for\_mantissa := 23 \qquad (2.1)$$

Section 2: Calculating the Condition Number

In this section, three distinct methods are used to calculate the condition number of coefficient matrix [A]. Each has its own advantages while utilizing theorems that relate the norm of a matrix to the conditioning of the matrix. Complete details for finding the norm of a matrix and its relationship to the conditioning of a matrix are given here.

Method 1: Finding the exact value of the Condition Number of a matrix

The following method finds the exact condition number of a square matrix. The exact formula is given by

$$Cond(A) = \|A\| * \|A^{-1}\| \qquad (3.1.A)$$

Once the condition number is calculated, it can then be used to solve for *m*, the number of significant digits that one can trust in solution.
Please note that, although this is the most direct method it may not be practical in its computational time for higher order matrices because this method requires calculation of the inverse of coefficient matrix [A]. The problem in finding the inverse lies in solving *n* sets of *n* equations which can be computationally intensive for large coefficient matrices.

Calculating the Condition number:

```
> with(LinearAlgebra):
  #Calculating [A^-1].
  Ainverse:=MatrixInverse(A):
> #Calculating ||A||.
```

```
NA:=Norm(A,infinity):
> #Calculating ||A^-1||.
NAInv:=Norm(Ainverse,infinity):
> #Calculating the condition number of [A] knowing that Cond(A)=||A||*||A^-1||.
Cond_A:=NA*NAInv;
```

$$Cond\_A := 46.03604065 \quad (3.1.1)$$

```
>
```

Calculating machine epsilon:

```
> Machine_epsilon:=evalf(2^(-Bits_used_for_mantissa));
```

$$Machine\_epsilon := 1.192092896\ 10^{-7} \quad (3.1.2)$$

Calculating the number of significant digits that one can trust in solution:

```
> #Using Equation (1.1).
sig_digits:=floor(solve(0.5*10^(-m)=Machine_epsilon*Cond_A,m))
:
Trust_digits:=max(0,sig_digits);
```

$$Trust\_digits := 4 \quad (3.1.3)$$

Method 2: Finding an approximate value of the condition number

The following numerical method finds the condition number of [A] using the inequality

$$Cond(A) \geq \frac{||\Delta X|| / ||X+\Delta X||}{||\Delta C|| / ||C||} \quad (3.2.A)$$

However, the value (X + ΔX) is equivalent to X'. The inequality then becomes

$$Cond(A) \geq \frac{||\Delta X|| / ||X'||}{||\Delta C|| / ||C||} \quad (3.2.B)$$

where ||ΔX|| / ||X'|| is the relative change in the norm of the solution vector and ||ΔC|| / ||C|| is the relative change in the norm of the right hand side vector. The ratio between these two values quantifies the conditioning of a system of equations, demonstrating the accuracy in solution. That is, for any small change made in the right hand side array, the resulting change in the solution vector will govern how accurate the system is and therefore how many significant digits one can trust in the solution of a system of simultaneous linear equations.

In this method, the condition number is calculated using equation (3.2.B) by first conducting the following steps:

1) A right hand side vector [C] is chosen such that the solution vector equals 1. (i.e. [X] = [1,1,..., 1]).

2) A new, unbiased right hand side vector [C'] is then generated by Maple. This is done by adding a random positive or negative value to each element of the old right hand side vector.

3) The new right hand side vector [C'] can then be used to calculate a new solution vector [X'].

By creating a small relative change in the right hand side array (i.e., $||\Delta C||/||C|| < 1$), the magnitude of the condition number will be largely influenced by the relative error in the solution vector, demonstrating how accurate the solution actually is.

NOTE: Each time the worksheet is executed, Maple will generate a new condition number. The user should run the worksheet several times to see if the estimate of the condition number approaches the exact value given in Method 1. The greatest of the generated values will be the most accurate approximation and will never exceed the true condition number due to the above inequality, Eq.(3.2.B).

Parameter names:
RHS = old right hand side array [RHS]
RHS1 = new right hand side array [RHS']
X = old solution vector [X]
X1 = new solution vector [X']

```
> #Defining RHS as a vector.
  RHS:=Vector(1..n):
  #Defining RHS1 as a vector.
  RHS1:=Vector(1..n):
  #Defining X as a vector.
  X:=Vector(1..n):
  #Defining X1 as a vector.
  X1:=Vector(1..n):
```

```
> #Step 1: Assigning values to [RHS] so that [X] equals [1,..,1].
  for i from 1 by 1 to n do
     RHS[i]:=add(A[i,j],j=1..n);
  end do:
  #Solving for [X].
  X:=LinearSolve(A,RHS);
```

$$X := \begin{bmatrix} 1. \\ 1. \\ 0.99999999999999988 \\ 1.00000000000000022 \end{bmatrix} \qquad (3.2.1)$$

```
> #Step 2: Generating the new, unbiased right hand side vector [RHS'] by adding a random ±
```

value between -0.001*RHS[i] and +0.001*RHS[i] to create a small ΔC value.

```
for i from 1 by 1 to n do
      Randomize():
      #Randomizing the sign of the value added to each element of new right hand side
vector.
      roll:=rand(1..2):
      sign_val:=roll():
      Randomize():
      RHS1[i]:=RHS[i]+(-1)^sign_val*0.001*RHS[i]*rand()/1E12:
   end do:
print('RHS1'=RHS1);
```

$$RHS1 = \begin{bmatrix} 27.98543338 \\ 12.00126911 \\ 70.98685772 \\ 212.2256756 \end{bmatrix} \tag{3.2.2}$$

> **#Step 3:** Calculating the new solution vector [X'].

```
X1:=LinearSolve(A,RHS1);
```

$$X1 := \begin{bmatrix} 0.999917419536820006 \\ 0.999225632902688288 \\ 1.00273101881558913 \\ 0.998950772865370262 \end{bmatrix} \tag{3.2.3}$$

Notice the small difference between the values of the old and new right hand side vectors. Now look at the new solution vector [X1]. Do the values deviate much from the old solution vector [X] ? If not, then any small changes that are made in the right hand side array do not affect the accuracy of the solution vector, and the solution can therefore be trusted. Otherwise, the system of equations is ill-conditioned. Below, the condition number of [A] is calculated by determining the values required by Equation (3.2.B).

> #Calculating the relative change in the norm of the solution vector.

```
Num_Cond:=Norm(X1-X,infinity)/Norm(X1,infinity):
#Calculating the relative change in the norm of the right hand side array.
Den_Cond:=Norm(RHS1-RHS,infinity)/Norm(RHS,infinity):
#Calculating the condition number, using Equation (3.2.B).
Cond_AA[k]:=Num_Cond/Den_Cond;
```

$$Cond\_AA_k := 3.232266237 \tag{3.2.4}$$

Method 3: A second technique in approximating the condition number

The following numerical technique is simpler than Method 2 as it requires fewer calculations. This method utilizes the theorem

$$\text{Cond}(A) \geq \frac{\|A\| \, \|X\|}{\|C\|} \qquad (3.3.A)$$

The proof is as follows:

Let
$[X] = [A^{-1}] \, [C]$

Applying norm properties, this equation becomes
$\|X\| \leq \|A^{-1}\| \, \|C\|$

Multiplying $\|A\|$ to both sides gives
$\|A\| \, \|X\| \leq \text{Cond}(A) \, \|C\|$

by the definition of condition number.

And division by $\|C\|$ results in the final inequality.

In this technique, however, the right hand side values of [C] matrix are chosen to equal [±1, ±1, ±1] with signs generated randomly. This will result in $\|C\| = 1$, minimizing the number of calculations required to solve for Cond(*A*). Therefore, the calculation is reduced to:

$$\text{Cond}(A) \geq \|A\| \, \|X\|. \qquad (3.3.B)$$

Again, the user should re-execute the worksheet to see if the greatest approximate condition number approaches the exact condition number defined in Method 1.

```
> #Randomly choosing a (+1) or (-1) value for each element of the right hand side array.
  for i from 1 by 1 to n do
        Randomize():
        roll:=rand(1..2):
        sign_val:=roll():
        Randomize():
        RHS[i]:=(-1)^sign_val:
        end do:
  #Solving for the solution vector [X].
  X:=LinearSolve(A,RHS):
  #Calculating the condition number using Equation (3.3.B).
```

```
Cond_AAA:=Norm(A,infinity)*Norm(X,infinity);
```

```
>
```

$$Cond\_AAA := 41.01967884 \tag{3.3.1}$$

References

[1] Autar Kaw, *Holistic Numerical Methods Institute, http://numericalmethods.eng.usf.edu/mws,* See
System of Equations
What is Machine Epsilon?
Adequacy of Solutions

Conclusion

Maple helped us apply our knowledge of norms and condition numbers to quantify the accuracy in a solution for a system of simultaneous linear equations.

Question 1: Choose a coefficient matrix [A] that is well-conditioned. For example,
Matrix([[10, −7, 0], [−3, 2.099, 6], [5, −1, 5]])

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{bmatrix} \tag{5.1}$$

See how the condition number of the matrix affects the number of significant digits that one can trust in solution.

Question 2: Choose a coefficient matrix [A] that is ill-conditioned. For example,
Matrix([[1, 2, 3], [1, 2, 3.001], [5, 61, 128]])

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3.001 \\ 5 & 61 & 128 \end{bmatrix} \tag{5.2}$$

See how the condition number of the matrix affects the number of significant digits that one can trust in solution.

Question 3: Choose the number of bits used for the mantissa in single and double precision. See how these numbers affect the number of significant digits that one can trust in the solution.