An Example of Roundoff

Mth 341 Linear Algebra
Spring 2000
Bent Petersen


```
> restart;with(linalg):
Warning, new definition for norm
Warning, new definition for trace
```

Let's row reduce the following matrix "by hand" so to speak:

```
> B:=matrix(3,3,[-.13,.10,.02,.05,-.16,.07,.08,.06,-.09]);
```

$$B := \begin{bmatrix} -.13 & .10 & .02 \\ .05 & -.16 & .07 \\ .08 & .06 & -.09 \end{bmatrix}$$

```
> addrow(B,1,2,1);addrow(%,2,3,1);mulrow(%,1,-1/.13);B1:=addrow(%
  ,1,2,.08);
```

$$\begin{bmatrix} -.13 & .10 & .02 \\ -.08 & -.06 & .09 \\ .08 & .06 & -.09 \end{bmatrix}$$

$$\begin{bmatrix} -.13 & .10 & .02 \\ -.08 & -.06 & .09 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ -.08 & -.06 & .09 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B1 := \begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ 0 & -.1215384615 & .07769230770 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> B2:=mulrow(B1,2,1/B1[2,2]);B3:=addrow(%,2,1,-B2[1,2]);
>
```

$$B2 := \begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ 0 & 1.000000000 & -.6392405066 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B3 := \begin{bmatrix} 1.000000000 & 0 & -.6455696204 \\ 0 & 1.000000000 & -.6392405066 \\ 0 & 0 & 0 \end{bmatrix}$$

It is clear that the matrix B has rank 2. Now let's let Maple do the row reduction directly:

```
> rref(B);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This time it looks as if B has rank 3. Which is it? Let's do the row reduction again by hand but this time we normalize our pivot in column 1 first

```
> C1:=mulrow(B,1,1/B[1,1]);
```

$$C1 := \begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ .05 & -.16 & .07 \\ .08 & .06 & -.09 \end{bmatrix}$$

```
> addrow(C1,1,2,-C1[2,1]); C2:=addrow(%,1,3,-C1[3,1]);
```

$$\begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ 0 & -.1215384615 & .07769230769 \\ .08 & .06 & -.09 \end{bmatrix}$$

$$C2 := \begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ 0 & -.1215384615 & .07769230769 \\ 0 & .1215384615 & -.07769230770 \end{bmatrix}$$

```
> C3:=mulrow(C2,2,1/C2[2,2]); addrow(C3,2,1,-C3[1,2]);
  C4:=addrow(%,2,3,-C3[3,2]);
```

$$C3 := \begin{bmatrix} 1.000000000 & -.7692307692 & -.1538461538 \\ 0 & 1.000000000 & -.6392405065 \\ 0 & .1215384615 & -.07769230770 \end{bmatrix}$$

$$\begin{bmatrix} 1.000000000 & 0 & -.6455696203 \\ 0 & 1.000000000 & -.6392405065 \\ 0 & .1215384615 & -.07769230770 \end{bmatrix}$$

$$C4 := \begin{bmatrix} 1.000000000 & 0 & -.6455696203 \\ 0 & 1.000000000 & -.6392405065 \\ 0 & 0 & -.1\ 10^{-10} \end{bmatrix}$$

Notice in the third column, third row, we have a very small number which arises from roundoff in our calculations. Since it is not zero it becomes a bogus pivot. In a sense we are doing the calculations too accurately. This bogus pivot is very much smaller than the presumed 2 place accuracy in our original data. If we do the calculations with just 2 or 3 decimal places of accuracy we are less likely to be led astray.

In Maple we can set the accuracy for decimal calculations by setting the quantitiy "Digits." The default is 10. Let's try a few other values.

```
> Digits:=2: rref(B);
```

$$\begin{bmatrix} 1 & 0 & -.65 \\ 0 & 1 & -.65 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> Digits:=3: rref(B);
```

$$\begin{bmatrix} 1 & 0 & -.644 \\ 0 & 1 & -.637 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> Digits:=4: rref(B);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Digits:=19: rref(B);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Digits:=20: rref(B);
```

$$\begin{bmatrix} 1 & 0 & -.64556962025316455697 \\ 0 & 1 & -.63924050632911392406 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> Digits:=160: rref(B);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Digits:=10: # Reset to default
```

We do have a problem here! Let's multiply B by 100 and convert the entries to integers. If we row reduce the resulting matrix Maple will use exact rational arithmetic:

```
> BB:=matrix(3,3,[-13,10,2,5,-16,7,8,6,-9]); rref(BB);
```

$$BB := \begin{bmatrix} -13 & 10 & 2 \\ 5 & -16 & 7 \\ 8 & 6 & -9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & \frac{-51}{79} \\ 0 & 1 & \frac{-101}{158} \\ 0 & 0 & 0 \end{bmatrix}$$

Thus if B is actually given exactly by the original expression then the rank is 2. But if the entries in B consist of experimental data which is not known exactly the situation is less clear.

Here's a little routine to compute the rank of B, in floating point, using a prescribed accuracy of n decimal digits.

```
> rnk:=proc(A,n)
>   local r,t;
>   t:=Digits; Digits:=n;
>   if rref(A)[3,3] = 1 then r:=3
>   else r:=2; fi;
>   Digits:=t;
> r;
> end:
```

Remark: This is not a good algorithm for computing the rank.

Let's look at the sequence of results that we get by considering decimal precisions from 2 to 240 decimal digits.

```
> seq(rnk(B,n),n=2..240);
```

2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 2, 2, 3,
3, 3, 3, 2, 2, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3,
2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 2, 2, 3,
3, 3, 3, 2, 2, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3,
2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3, 2, 2, 3,
3, 3, 3, 2, 2, 3, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 2, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3,
2, 2, 3, 3, 3

One thing is abundantly clear - extra precision does not cast any light on the problem. Our sequence of rank estimates looks almost random! The calculation of rref(B) is inherently badly posed. To see this let's perturb a bit the exact version of BB = 100B and row reduce using exact arithmetic.

First recall BB

```
> evalm(BB); rref(BB);
```

$$\begin{bmatrix} -13 & 10 & 2 \\ 5 & -16 & 7 \\ 8 & 6 & -9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & \frac{-51}{79} \\ 0 & 1 & \frac{-101}{158} \\ 0 & 0 & 0 \end{bmatrix}$$

Now lets perturb one of the diagonal entries in BB by a very tiny little bit

```
> CC:=evalm(BB+diag(1,0,0)/10^50);
```

$$CC := \begin{bmatrix} \frac{-129999999999999999999999999999999999999999999999999999}{100000000000000000000000000000000000000000000000000} & 10 & 2 \\ 5 & -16 & 7 \\ 8 & 6 & -9 \end{bmatrix}$$

```
> rref(CC);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Recall this is an <u>exact</u> calculation. At this point you may wonder is the rank of B two or three? If we view B as an exact matrix then the rank is 2, otherwise we should probably view it as having rank 3 if our interest is primarily in the rank (any tiny pertubation of B is almost certain to have exact rank 3). On the other hand we have to admit that the rank 3 result arises from promoting a very small pivot to 1, and that in an applied problem it might make more sense to set it to 0. From a practical real-world point of view then we should regard the rank as 2.

Solving a linear system throws some light on our dilema:

```
> linsolve(B,vector([0,0,1]));
```

$$[-.6455696203\ 10^{11}, -.6392405065\ 10^{11}, -.1\ 10^{12}]$$

```
> linsolve(BB,vector([0,0,100]));
```

Note if we view B as exact (i.e., the second case) we have no solution for the given right hand side. In the first case the solution is found using floating point arithmetic and is much larger than seems reasonable. Altering the precision has a big effect on the solution. For example:

```
> Digits:=20: linsolve(B,vector([0,0,1]));
```

$$[-.64556962025316455695\ 10^{21}, -.63924050632911392405\ 10^{21}, -.1\ 10^{22}]$$

These "solutions" (which are unique) clearly depend upon noise in the floating point calculations. From the point of view of solving a linear system with coefficient matrix B we should think of B as having rank 2 and the example system above, with inhomogeneous term [0,0,1], as being inconsistent, that is, as having no solution.

The matrix B was carefully chosen, on the brink so to speak, to yield a badly conditioned problem. You might think that in real world problems an experimentally obtained matrix is likely to have maximal rank and lead to better behaved problems. Normally that would be almost true (maximal rank does not suffice), but in the presence of symmetries, exactly conserved quantities, or relations between the elements of the matrix, one may easily end up with very badly conditioned systems. Solutions to real world problems should probably be accompanied by estimates of the error in the data, and estimates of the influence of such errors, together with roundoff and other errors, on the solution. Even powerful systems such as Maple do not allow you to dispense with careful thought!

Note in our example the matrix B has column sums 0. You can imagine that B describes some physical system with a conservation law that requires the column sums be exactly 0. Then all physically meaningful matrices B have rank at most 2 and we would feel no reservation about killing an exceptionally small pivot that threatens to yield rank 3.