Unavoidable Error

Table of Contents

Gwen Lofman, ISC3222, 27 October 2016

Roundoff Error

Perform Gaussian elimination on

```
A = [ 1 1      1
      1 1.0001 1
      1 2      2 ];
b = [ 1
      2
      1 ];
```

Without pivoting. Use three-figure floating point arithmetic during backward substitution. This means that we can keep all the digits during the calculation, but keep only three digits after the decimal point in the last step of backward substitution.

```
% Create a matrix that inclues the values of b
A1 = [A,b];

x1_   = A\b;

% Make sure the diagonal is 1
A1(1,:) = A1(1,:)  ./ A1(1,1);

% Cancel the other digits
A1(2,:) = A1(2,:) - (A1(1,:).* A1(2,1));

% Make sure the diagonal is 1
A1(2,:) = A1(2,:) ./ A1(2,2);

% Cancel the other digits
A1(3,:) = A1(3,:) - A1(1,:) .* A1(3,1);
A1(3,:) = A1(3,:) - A1(2,:) .* A1(3,2);

% Make sure the diagonal is 1
A1(3,:) = A1(3,:) ./ A1(3,3);

% Perform back-substitution
round(A1,3,'decimals');
x1 = zeros(3,1);
x1(3) = A1(3,4);
x1(2) = A1(2,4) - (x1(3) * A1(2,3));
```

```
x1(1) = A1(1,4) - (x1(2) * A1(1,2)) - (x1(3) * A1(1,3));
```

*Now conduct pivoting by interchanging equations 2 and 3. *

```
% Create a matrix that includes the values of b
A2 = [A,b];

% Perform pivoting
a2 = A2(2,:);
a3 = A2(3,:);
A2(2,:) = a3;
A2(3,:) = a2;

% Calculate with standard method
x2_   = A2(:,1:3)\A2(:,4);

% Begin gaussian elimination

% Make sure the diagonal is 1
A2(1,:) = A2(1,:)  ./ A2(1,1);

% Cancel the other digits
A2(2,:) = A2(2,:) - (A2(1,:).* A2(2,1));

% Make sure the diagonal is 1
A2(2,:) = A2(2,:) ./ A2(2,2);

% Cancel the other digits
A2(3,:) = A2(3,:) - A2(1,:) .* A2(3,1);
A2(3,:) = A2(3,:) - A2(2,:) .* A2(3,2);

% Perform back-substitution
round(A2,3,'decimals');
x2    = zeros(3,1);
x2(3) = A2(3,4);
x2(2) = A2(2,4) - (x2(3) * A2(2,3));
x2(1) = A2(1,4) - (x2(2) * A2(1,2)) - (x2(3) * A2(1,3));
```

What conclusion can you draw from this exercise?

```
Eabs1 = abs(x1 - x1);
Erel1 = Eabs1./x1_;

Eabs2 = abs(x2 - x2_);
Erel2 = Eabs2./x2_;

fprintf('   no pivot    abserr      relerr    |   pivot     abserr
 relerr\n')
for i=1:3
    fprintf(' %10.3f  %10.3e  %10.3e |%8.3f  %10.3e  %10.3e
\n',x1(i),Eabs1(i),Erel1(i),x2(i),Eabs2(i),Erel2(i))
end
```

```
   no pivot    abserr      relerr    |   pivot     abserr      relerr
      1.000  0.000e+00   0.000e+00 |   1.000  0.000e+00   0.000e+00
```

```
 10000.000   0.000e+00   0.000e+00 |  -1.000   1.000e+04   1.000e+00
-10000.000   0.000e+00  -0.000e+00 |   1.000   1.000e+04  -1.000e+00
```

Moving the elements has a great effect on the answer of the exercise; making sure that the elements involved have the proper positions in the matrix ensures that operations performed over the elements have the necessary precision to prevent catastrophic cancellation errors.

Relative Convergance Criteria

The absolute and relative convergence criteria in Equation (5.12) of the textbook are written one way, but example 5.6 has `while abs(r-rold)/rold>delta & it<maxit` Is this an error? Should the while statement have a < sign instead of a > sign? Why?

This is not an error; the convergence criteria determines whether a calculation has converged, and thus must return true when convergence happens. However, in the while statement, the expression must return true as long as we want the computation to keep on happening; thus, the while statement shows the desired behavior, as once the convergence criteira is met, it will return false, telling the while loop to terminate.

Rewriting the statement as `while ~(abs(r-rold)/rold<delta) && it<maxit` may make this desired behavior more clear to the user.

Published with MATLAB® R2016a