Examples for Lecture 3

Outline

1. Polynomial fitting
2. LU decomposition
3. Sensitivity of linear systems

Polynomial fitting

The national government carries out a census of its population every few years.

Our goal is to predict the population of a country (e.g. China) in between the census yeawrs, or to estimate future population, one approach is oto use **interpolation**.

Let us achieve this goal using polynomial functions. Specifically, we assume the population and the year has the following relation

$$f(x) = c_0 + c_1 t + c_2 t^2 + \ldots c_{n-1} t^{n-1}$$

This function is nonlinear in x but linear in c_i.

We can write a linear system explicitly .

$$\begin{bmatrix} 1 & t_1 & \dots & t_1^{n-2} & t_1^{n-1} \\ 1 & t_2 & & t_2^{n-2} & t_2^{n-1} \\ \vdots & & \vdots & & \vdots \\ 1 & t_n & \dots & t_n^{n-2} & t_n^{n-1} \end{bmatrix} c = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Or writen as $Vc = y$ for short. The above type of matrix is called a *Vandermonde* matrtix.

LU decomposition

Here is the system that "broke" LU factorization for us.

```
A = [ 2 0 4 3; -2 0 2 -13 ; 1 15 2 -4.5 ; -4 5 -7 -10 ];
b = [ 4; 40; 29; 9 ];
```

When we use the built-in `|lu|` function with three outputs, we get the elements of the `PLU` factorization.

```
[L,U,P] = lu(A)
```

We can solve this as before by incorporating the permutation.


```
x = backsub( U, forwardsub(L,P*b) )
```

However, if we use just two outputs with |lu|, we get $\mathbf{P}^T\mathbf{L}$ as the first result.

```
[PtL,U] = lu(A)
```

MATLAB has engineered the backslash so that systems with triangular or permuted triangular structure are solved with the appropriate style of triangular substitution.

```
x = U \ (PtL\b)
```

The pivoted factorization and triangular substitutions are done silently and automatically when backslash is called on the original matrix.

```
x = A\b
```

Sensitivity of linear systems

We want to show how the error propagates in the ill-conditioned problems.

$$\frac{\|\Delta x\|}{\|x\|} \simeq \kappa(A) \frac{\|\Delta A\|}{\|A\|}$$

`H = hilb(n)` returns the Hilbert matrix of order n. The Hilbert matrix is a notable example of a poorly conditioned matrix. The elements of Hilbert matrices are given by $H(i,j) = 1/(i + j - 1)$.

```
A = hilb(7);
kappa = cond(A)

kappa =
    4.7537e+08
```

Next we engineer a linear system problem to which we know the exact answer.

```
x_exact = (1:7)';

b = A*x_exact;
```


Now we perturb the data randomly but with norm 10^{-12}.


```
randn('state',333);      % reproducible results

dA = randn(size(A)); dA = 1e-12*(dA/norm(dA));
```

We solve the perturbed problem using built-in pivoted LU and see how the solution was changed.

```
x = (A+dA) \ b;

dx = x - x_exact;
```


Here is the relative error in the solution.


```
rel_error = norm(dx) / norm(x_exact)
```

And here are upper bounds predicted using the condition number of the original matrix.

```
A_bound = kappa * 1e-12/norm(A)
```

Even if we don't make any manual perturbations to the data, machine epsilon does when we solve the linear system numerically.

```
x = A\b;

rel_error = norm(x - x_exact) / norm(x_exact)

rounding_bound = kappa*eps
```

Now we choose an even more poorly conditioned matrix from this family.

```
A = hilb(14);

kappa = cond(A)
```


Before we compute the solution, note that κ exceeds |1/eps|. In principle we might end up with an answer that is completely wrong.


```
rounding_bound = kappa*eps
```

MATLAB will notice the large condition number and warn us not to expect much from the result.

```
x_exact = (1:14)';

b = A*x_exact; x = A\b;
```


In fact the error does exceed 100%.


```
relative_error = norm(x_exact - x) / norm(x_exact)
```