Bi7740: Scientific computing

Systems of linear equations

Vlad Popovici
popovici@iba.muni.cz

Institute of Biostatistics and Analyses
Masaryk University, Brno

Backward substituion

```
1  function x = bksolve(A, b)
2  % Solves Ax = b for x, assuming that A is a
3  % nonsingular upper triangular matrix.
4
5    n = length(b);
6    x = b;
7
8    % add tests for A(i,i) close to 0!
9    x(n) = b(n) / A(n,n);
10   for i = n-1:-1:1
11     x(i) = (b(i) - A(i,i+1:n)*x(i+1:n)) / A(i,i);
12   end
13 return
```

Try:

```
>> A = [2,2,3,4;0,5,6,7;0,0,8,9;0,0,0,10];
>> b = [20,34,25,10]';
>> bksolve(A,b)

ans =

      2
      3
      2
      1
```

Forward substituion

```
1  function x = fwsolve(A, b)
2  % Solves Ax = b for x, assuming that A is a
3  % nonsingular lower triangular matrix.
4
5    n = length(b);
6    x = b;
7
8    % add tests for A(i,i) close to 0!
9    x(1) = b(1) / A(1,1);
10   for i = 2:n
11     x(i) = (b(i) - A(i,1:i-1)*x(1:i-1)) / A(i,i);
12   end
13
14 return
```

LU decomposition

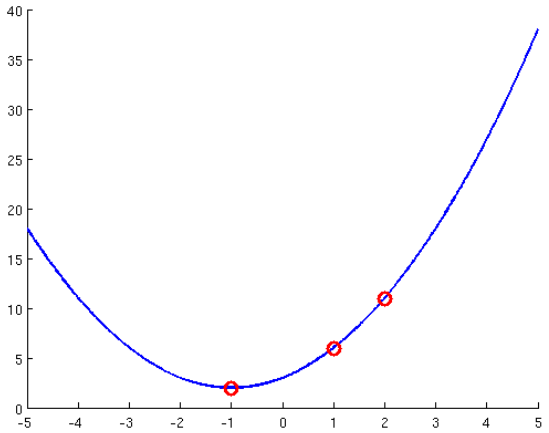
```
1 for e = logspace(-2,-18,9)
2   A = [e 1; 1 1]; b = [1+e; 2];
3   L = [ 1 0; A(2,1)/A(1,1) 1];
4   U = [A(1,1) A(1,2) ; 0 A(2,2)-L(2,1)*A(1,2)];
5   y(1) = b(1); y(2) = b(2) - L(2,1)*y(1);
6   x(2) = y(2)/U(2,2); x(1) = (y(1) - ...
        U(1,2)*x(2))/U(1,1);
7 fprintf(' %5.0e   %20.15f   %20.15f\n', e, x(1), x(2))
8 end
```

```
 Delta            x(1)                       x(2)
------------------------------------------------------
 1e-02     1.000000000000001      1.000000000000000
 1e-04     0.999999999999890      1.000000000000000
 1e-06     1.000000000028756      1.000000000000000
 1e-08     0.999999993922529      1.000000000000000
 1e-10     1.000000082740371      1.000000000000000
 1e-12     0.999866855977416      1.000000000000000
 1e-14     0.999200722162641      1.000000000000000
 1e-16     2.220446049250313      1.000000000000000
 1e-18     0.000000000000000      1.000000000000000
```

Polynomial interpolation

```
1  function a = polyinterp(x, y)
2  % Returns the length(x) coefficients of a polynomial
3  % of degree length(x)-1 interpolating the points (x,y).
4
5    n = length(x);
6    if length(y) ≠ n
7      stop('x and y must be of equal length')
8    end
9    V = zeros(n, n);
10   for k = 1:n
11       V(:,k) = x .^ (n-k);
12   end
13   a = V \ y;                    % try different methods here
14
15 return
```

- MATLAB: `vander` returns the Vandermonde matrix for `x`
- try:
  - `polyinterp([-1, 1, 2]', [2,6,11]')` → should return `[1 2 3]'` for $p(x) = x^2 + 2x + 3$



  - `polyinterp([1, 2, 3]', [3, 5, 7]')` → what happens?