פירוק LU

קודם: שקפים 1-4 על הלוח (הדוגמא גם על שקף 4-5)

דוגמא (משיעור שעבר):

$$\begin{array}{ccccc} A & X & & B \\ \begin{pmatrix} 0.0003 & 3 \\ 1 & 1 \end{pmatrix} & \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & = & \begin{pmatrix} 2.0001 \\ 1 \end{pmatrix} \end{array}$$

הפירוק (עם החלפת השורות):

$$\begin{array}{ccccc} L & & U & & A' \\ \begin{pmatrix} 1 & 0 \\ .0003 & 1 \end{pmatrix} & \cdot & \begin{pmatrix} 1 & 1 \\ 0 & 2.9997 \end{pmatrix} & = & \begin{pmatrix} 1 & 1 \\ 0.0003 & 3 \end{pmatrix} \end{array}$$

שלב 1:

$$\begin{array}{ccccc} L & & Y & & B' \\ \begin{pmatrix} 1 & 0 \\ .0003 & 1 \end{pmatrix} & \cdot & \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} & = & \begin{pmatrix} 1 \\ 2.0001 \end{pmatrix} \end{array}$$

שלב 2:

$$\begin{array}{ccccc} U & & X & & Y \\ \begin{pmatrix} 1 & 1 \\ 0 & 2.9997 \end{pmatrix} & \cdot & \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & = & \begin{pmatrix} 1 \\ 1.9998 \end{pmatrix} \end{array}$$

פירוק LU

void ludcmp(float **a, int n, int *indx, float *d)

Given a matrix a[1..n][1..n], this routine replaces it by the *LU* decomposition of a rowwise permutation of itself. a and n are input. a is output, arranged as in equation (2.3.14) above; indx[1..n] is an output vector that records the row permutation effected by the partial pivoting; d is output as ±1 depending on whether the number of row interchanges was even or odd, respectively. This routine is used in combination with lubksb to solve linear equations or invert a matrix.

$$a[1\ldots n][1\ldots n] \Rightarrow \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} \\ \alpha_{21} & \beta_{22} & \beta_{23} \\ \alpha_{31} & \alpha_{32} & \beta_{33} \end{pmatrix}$$

מחליף את a, אחרי הרצה:

$$\mathrm{indx}[1\ldots n]$$

שומר את החלפות השורות:

$$d = \pm 1$$

שומר אם היה מספר זוגי (+1) או אי-זוגי (-1) של החלפות שורות:

פירוק LU

void lubksb(float **a, int n, int *indx, float b[])
Solves the set of n linear equations *A·X* = *B*. Here a[1..n][1..n] is input, not as the matrix *A* but rather as its *LU* decomposition, determined by the routine ludcmp. indx[1..n] is input as the permutation vector returned by ludcmp. b[1..n] is input as the right-hand side vector *B*, and returns with the solution vector *X*. a, n, and indx are not modified by this routine and can be left in place for successive calls with different right-hand sides b. This routine takes into account the possibility that b will begin with many zero elements, so it is efficient for use in matrix inversion.

$$\mathrm{b}[1...n] : \mathrm{B} \rightarrow \mathrm{X}$$ **מחליף את b אחרי הרצה:**

אופן השימוש:

```
float **a,*b,d;
int n,*indx;
...
ludcmp(a,n,indx,&d);
lubksb(a,n,indx,b);
```


עכשיו: שקפים 5-11 על הלוח

פירוק SVD

void svdcmp(float **a, int m, int n, float w[], float **v)
Given a matrix a[1..m][1..n], this routine computes its singular value decomposition, $A = U \cdot W \cdot V^T$. The matrix U replaces a on output. The diagonal matrix of singular values W is output as a vector w[1..n]. The matrix V (not the transpose V^T) is output as v[1..n][1..n].

$$a[1...m][1...n] : A \Rightarrow U$$ **מחליף את a אחרי הרצה:**

$$w[1...n] : \mathrm{W} = \begin{pmatrix} w_1 & & 0 \\ & w_2 & \\ & & \ddots \\ 0 & & & w_n \end{pmatrix} \qquad v[1...n][1...n] : V$$

אז, מאפסים את ה- w_j הקטנים, ואחרי זה קוראים (אפשר עם סדרה של b) ל:

void svbksb(float **u, float w[], float **v, int m, int n, float b[], float x[])
Solves $A \cdot X = B$ for a vector X, where A is specified by the arrays u[1..m][1..n], w[1..n], v[1..n][1..n] as returned by svdcmp. m and n are the dimensions of a, and will be equal for square matrices. b[1..m] is the input right-hand side. x[1..n] is the output solution vector. No input quantities are destroyed, so the routine may be called sequentially with different b's.