KSN — III FK — teoria 2

Rozwiązywanie UARL metodami bezpośrednimi (2)

Nadal zajmujemy się rozwiązywaniem układów algebraicznych równań liniowych $\mathbf{Ax} = \mathbf{b}$ metodami numerycznymi.

Tym razem posłużymy się metodą *rozkładu LU*, w której macierz $\mathbf{A}$ zastępuje się iloczynem macierzy $\mathbf{LU}$, gdzie $\mathbf{L}$ jest macierzą trójkątną dolną z jedynkami na diagonali, a $\mathbf{U}$ — macierzą trójkątną górną. Wyznaczenia rozwiązania $\mathbf{x}$ tą metodą sprowadza się do rozwiązania dwu układów z macierzami trójkątnymi: $\mathbf{Ly} = \mathbf{b}$ i $\mathbf{Ux} = \mathbf{y}$.

W bibliotece *Numerical Recipes* rozkładu LU macierzy dokonuje procedura `LUDCMP`, zaś postępowanie odwrotne realizuje procedura `LUBKSB`:

```
REAL a(np,np),b(np,np),x(np),p
INTEGER i,j,indx(np)
...
DO i=1,np
x(i)=b(i,1)
ENDDO
...
CALL LUDCMP(a,n,np,indx,p)
CALL LUBKSB(a,n,np,indx,x)
```

Wówczas, podobnie jak w procedurze `GAUSSJ` wektor wyrazów wolnych $\mathbf{b}$ jest nadpisywany rozwiązaniem $\mathbf{x}$.

Dekompozycja LU może też posłużyć do szybkiego odwracania macierzy, tj. znajdowania macierzy $\mathbf{Y} = \mathbf{A}^{-1}$, takiej, że $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{1}$:

```
INTEGER np,indx(np)
REAL a(np,np),y(np,np)

DO i=1,n
  DO j=1,n
    y(i,j)=0.
  ENDDO
  y(i,i)=1.
ENDDO

CALL LUDCMP(a,n,np,indx,p)

DO j=1,n
  CALL LUBKSB(a,n,np,indx,y(1,j))
ENDDO
```

Po tej operacji elementy macierzy `y` będą zawierały elementy macierzy odwrotnej do macierzy `a`.

Zawarte w bibliotece *GNU Scientific Library (GSL)* funkcje służące do rozwiązywania układów równań liniowych operują na obiektach `gsl_vector` i `gsl_matrix` (lub `gsl_vector_complex` i `gsl_matrix_complex` w przypadku obliczeń w zbiorze liczb zespolonych – należy wtedy użyć zespolonych odpowiedników opisanych poniżej funkcji, szczegóły w dokumentacji biblioteki).
Rozdział dokumentacji *GSL* pt. *Vectors and Matrices* opisuje sposoby tworzenia takich obiektów i przeprowadzania na nich wielu godnych uwagi operacji.

Rozkład LU macierzy $\mathbf{A}$ realizowany jest za pomocą funkcji

```
int gsl_linalg_LU_decomp (gsl_matrix * A, gsl_permutation * p, int * signum),
```

której użycie skutkuje zapisaniem macierzy $\mathbf{U}$ nad i na diagonali macierzy $\mathbf{A}$; poniżej diagonali umieszczana jest macierz $\mathbf{L}$ (oczywiście jej przekątnej, zawierającej same jedynki, nie trzeba nigdzie zapisywać).
Permutację `p` tworzy się funkcją `gsl_permutation * gsl_permutation_alloc (size_t n)`; `p` wraz z `signum` zawierają zakodowaną macierz permutacji dla wykonanej dekompozycji.

Natomiast funkcja

```
int gsl_linalg_LU_solve (const gsl_matrix * LU, const gsl_permutation * p,
                         const gsl_vector * b, gsl_vector * x)
```

rozwiązuje układ $\mathbf{Ax} = \mathbf{b}$ wykorzystując rozkład LU macierzy $\mathbf{A}$ na (`LU, p`), wyznaczone wcześniej za pomocą `gsl_linalg_LU_decomp`. Wynik umieszczany jest w `x`.

Krzysztof Malarz & Maciej Wołoszyn, Kraków, 21 października 2004