

Numeryczna algebra liniowa w ANSI C

Co zrobić, gdy musimy *policzyć wartości własne* (oraz opcjonalnie — także *wektory własne*) dużej macierzy A ?

$$„ \mathbf{A} \mathbf{v} = \lambda \mathbf{v} ”$$

1) **Biblioteka GSL** (*GNU Scientific Library*):

Uniwersalne narzędzie do różnorodnych problemów numerycznych (całkowanie, rozwiązywanie równań różniczkowych, obliczanie wartości funkcji specjalnych, minimalizacja funkcji, ...)

[*Zawarta w wielu dystrybucjach Linuxa; także część Cygwin-a*]

zob. <https://www.gnu.org/software/gsl/>

2) **LAPACK:** Linear Algebra PACKage —

<http://www.netlib.org/lapack/explore-html/index.html>

Thursday, June 25, 2009

gsl vs lapack performance

I had some doubts about the LU routines in the gsl library (GNU Scientific Library). See <http://yetanothermathprogrammingconsultant.blogspot.com/2009/06/gsl-gnu-scientific-library.html>. Here I try a quick experiment by inverting a square $n \times n$ matrix. As test matrix I used the Pei matrix (<http://portal.acm.org/citation.cfm?id=368975>). Here are the results:

library	gsl	lapack
routine	gsl_linalg_LU_deco mp + gsl_linalg_LU_invert	dgesv
compiler	cygwin gcc	Lahey If95
n=100	0.047 seconds	0.024 seconds
n=1000	6.735 seconds	2.017 seconds
n=2000	1:01 minutes	17.257 seconds

[<http://yetanothermathprogrammingconsultant.blogspot.com/2009/06/gsl-vs-lapack-performance.html>]

Przykład: ZGEEV — oblicza wartości własne (opcjonalnie, także *lewo- i prawostronne wektory własne*) macierzy kwadratowej zespolonej.

◆ zgeev()

```
subroutine zgeev ( character                JOBVL,  
                 character                JOBVR,  
                 integer                  N,  
                 complex*16, dimension( lda, * ) A,  
                 integer                  LDA,  
                 complex*16, dimension( * ) W,  
                 complex*16, dimension( ldvl, * ) VL,  
                 integer                  LDVL,  
                 complex*16, dimension( ldvr, * ) VR,  
                 integer                  LDVR,  
                 complex*16, dimension( * ) WORK,  
                 integer                  LWORK,  
                 double precision, dimension( * ) RWORK,  
                 integer                  INFO  
                 )
```

```

/* Sposob podlaczenia LAPACKA w ANSI C, zob R.H. Landau,
http://physics.oregonstate.edu/~landaur/nacphy/lapack/
cprog.html
*/
#include <stdio.h>

#define size 3      /* dimension of matrix in THIS example */
struct complex {double re; double im;};      /* a complex
number */

/* Deklaracja f.zewnetrznej - mozna pominac w ANSI C (!) */
extern void zgeev_(char * jobvl, char * jobvr, int * n,
        struct complex * A, int * lda,
        struct complex * w,
        struct complex * vl, int * ldvl,
        struct complex * vr, int * ldvr,
        struct complex * work, int * lwork,
        double * rwork, int * info);

```

```

/* Eigenvalues of a generix complex matrix with ZGEEV */
int main()
{
    /*          ( 1  -i  0 )    */
    /*   Input matrix:   A = ( i  -1  0 )    */
    /*          ( 0   0  7 )    */
    struct complex A[3][3] = {
        { 1.0, 0.0, 0.0, -1.0, 0.0, 0.0 },
        { 0.0, 1.0, -1.0, 0.0, 0.0, 0.0 },
        { 0.0, 0.0, 0.0, 0.0, 7.0, 0.0 },
    };

    struct complex b[3], DUMMY[1], WORK[6];
    struct complex AT[size*size];    /* ==> input for ZGEEV */
    int i, j, ok, c1, c2, c3;
    char c4;

```

```
for (i=0; i<size; i++) /* to call a f77 routine from C */
{
    /* we have to transform A ... */
    for (j=0; j<size; j++) {
        AT[j + size*i].re = A[j][i].re;
        AT[j + size*i].im = A[j][i].im;
    }
}

c1 = size; /* ... and put all numbers, etc., */
c2 = 2*size; /* we want to pass */
c3 = 1; /* to the routine in variables */
c4 = 'N';
```

```

/* to find solution using LAPACK routine ZGEEV, all      */
/* the arguments have to be pointers and you have to add */
/* an underscore to the routine name                    */

    zgeev_(&c4, &c4, &c1, AT, &c1, b, DUMMY, &c3, DUMMY, &c3,
        WORK, &c2, (double *)WORK, &ok);

/*
parameters, in the order as they appear,
in the function call:
no left eigenvectors, no right eigenvectors,
order of input matrix A, input matrix A,
leading dimension of A, array for eigenvalues,
array for left eigenvalue, leading dimension of DUMMY,
array for right eigenvalues, leading dimension of DUMMY,
workspace array dim >= 2*(order of A), dimension of WORK
workspace array dim = 2*(order of A), return value
*/

```

```

if (ok==0)                /* print eigenvalues */
    for (i=0; i<size; i++) {
        printf("%lf\t%lf\n", b[i].re, b[i].im);
    }
else
    printf("An error occured\n");

/* The correct output is:

    1.414214      0.000000
   -1.414214      0.000000
    7.000000      0.000000
*/
return 0;
}

```


Uwagi o LAPACK-u:

Bardziej *eleganckie* wywołania (*oryginalnych!*) funkcji lapackowskich są możliwe po zainstalowaniu *wrapperów* do języka C; zob.:

<http://www.netlib.org/lapack/lapwrapc/>

http://www.netlib.org/lapack/lapwrapc/presentation_wrapper_C.pdf

Funkcje biblioteki Lapack, napisane w Fortranie 77 i dostarczone w postaci skompilowanej biblioteki dla konkretnej maszyny, mogą być wywoływane także z poziomu rozmaitych środowisk obliczeniowych, jak np. *Mathematica*:

[https://reference.wolfram.com/language/tutorial/](https://reference.wolfram.com/language/tutorial/LinearAlgebraMatrixComputations.html)

[LinearAlgebraMatrixComputations.html](https://reference.wolfram.com/language/tutorial/LinearAlgebraMatrixComputations.html)

<http://library.wolfram.com/infocenter/Conferences/4886/>