IZ INFORMATIKE

Savjeti za korištenje matematičkih programskih paketa

Goranka Nogo[1]

1. Uvod

Povod pisanju ovoga članka je svakodnevno susretanje s neefikasnim korištenjem matematičkih programskih paketa (*MATLAB*, *Mathematica*, …) te nekritičkim prihvaćanjem rezultata dobivenih uporabom nekog od njih.

Primjeri navedeni u članku izvođeni su i komentirani sa studentima 4. i 5. godine diplomskih studija na PMF-Matematičkom odsjeku Sveučilišta u Zagrebu, kolegij Matematički softver. Riječ je o klasičnim pogreškama koje imaju tendenciju ponavljanja. Korišteni programski sustav u ovom članku je *MATLAB* 7.11.0. Ovisno o računalu, dobivena vremena mogu se razlikovati od ovdje navedenih u članku, no odnosi između vremena/rezultata ne ovise bitno o korištenom računalu.

2. Neefikasnost

Klasičan primjer neefikasnog korištenja matematičkih programskih paketa je kada, u nepoznavanju naredbe, sami pokušavamo implementirati u softver već ugrađenu funkciju. Studenti, u pravilu, problem pokušavaju riješiti korištenjem petlji, najčešće petlje `for`. U nastavku navodimo jedan takav tipični primjer. Jednostavnosti radi, ograničit ćemo se na elementarni matrični račun.

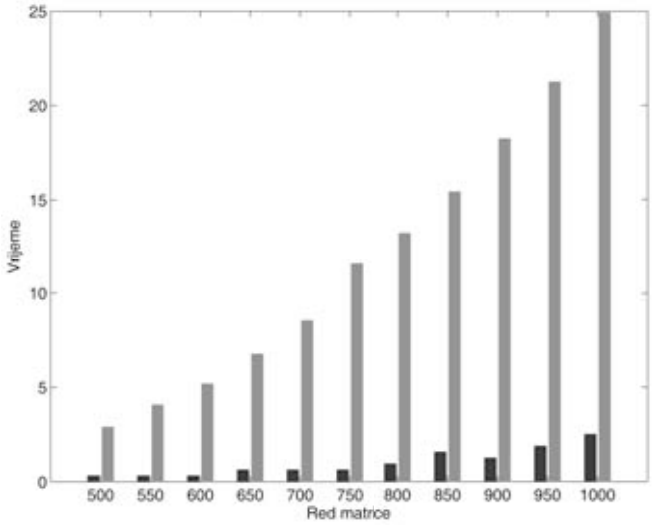
Primjer 1. Množenje matrica

Neka su A i B kvadratne matrice reda n. Njihov umnožak pomoću programskog sustava *MATLAB* možemo izračunati gotovom naredbom `C = A * B;`. Drugi je način da sami implementiramo funkciju (F) za računanje umnoška dviju matrica koristeći tri standardne ugniježdene petlje:

```
for i = 1 : n
    for j = 1 : n
      C(i,j) = 0;
      for k = 1 : n
        C(i,j) = C(i,j) + A(i,k) * B(k,j);
      end
    end
end
```

[1] Goranka Nogo, PMF-Matematički odsjek, Zagreb

Na Slici 1. grafički su prikazana vremena potrebna za računanje umnoška kvadratnih matrica reda $n = 500, 550, \ldots, 1000$. Viši stupci predstavljaju vremena izvršavanja (u sekundama) funkcije *F*, a niži vremena potrebna *MATLAB*-u pomnožena, vidljivosti radi, s 10. Elementi matrica su generirani pseudoslučajno pomoću funkcije rand, dok je za mjerenje vremena izvršavanja korištena funkcija `cputime`.

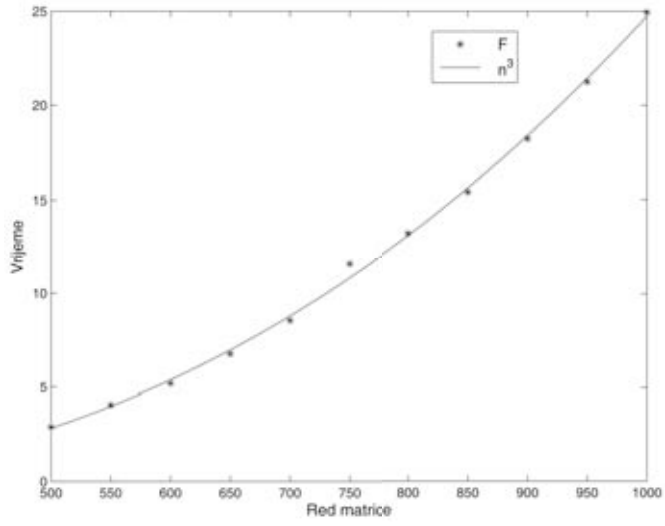


Slika 1.

Primijetimo da je naša implementacija funkcijom *F* bitno sporija od *MATLAB*-ove.

Zadatak 1. (vezan uz složenost množenja kvadratnih matrica reda *n*). Koja bi elementarna funkcija dobro aproksimirala vremena izvršavanja funkcije *F*?

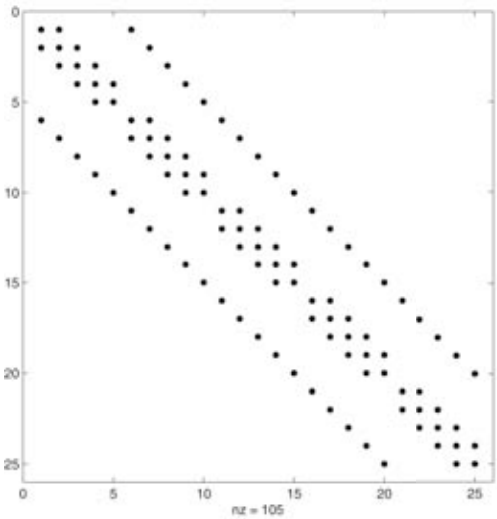
Na Slici 2. prikazana je aproksimacija vremena izvršavanja funkcije *F* polinomom trećeg stupnja.



Slika 2.

Primjer 2. Rad s rijetko popunjenim matricama

Za matricu A kažemo da je rijetko popunjena ako je većina njezinih elemenata jednaka nuli, a elementi koji nisu jednaki nuli obično su pravilno raspoređeni po matrici. Ako u komandnom prozoru napišemo `help gallery`, dobit ćemo popis rijetkih matrica, a grafički prikaz strukture rijetkih matrica (elementi koji su jednaki nuli nisu označeni) možemo dobiti pomoću naredbe `A = gallery('neumann', 25 ); spy(A)`.



Slika 3.

Broj na dnu grafičkog prikaza na Slici 3. označava koliko je elemenata matrice različito od nule.

U *MATLAB*-u je matrica pohranjena kao rijetka ako se pamte samo elementi različiti od nule te njihova pozicija. Time se, za rijetko popunjene matrice, štedi znatna količina memorije. Matrica je pohranjena kao puna ako su joj pohranjeni svi elementi. Konverziju iz rijetkog načina pohrane u puni obavlja naredba `full`. Na primjer,

```
A = full(gallery( 'neumann', 9 )) % Neumannova matrica reda 9
A =
     4    -2     0    -2     0     0     0     0     0
    -1     4    -1     0    -2     0     0     0     0
     0    -2     4     0     0    -2     0     0     0
    -1     0     0     4    -2     0    -1     0     0
     0    -1     0    -1     4    -1     0    -1     0
     0     0    -1     0    -2     4     0     0    -1
     0     0     0    -2     0     0     4    -2     0
     0     0     0     0    -2     0    -1     4    -1
     0     0     0     0     0    -2     0    -2     4
```

```
B = full(gallery( 'poisson', 3 ))  % Poissonova matrica reda 9
B =
     4    -1     0    -1     0     0     0     0     0
    -1     4    -1     0    -1     0     0     0     0
     0    -1     4     0     0    -1     0     0     0
    -1     0     0     4    -1     0    -1     0     0
     0    -1     0    -1     4    -1     0    -1     0
     0     0    -1     0    -1     4     0     0    -1
     0     0     0    -1     0     0     4    -1     0
     0     0     0     0    -1     0    -1     4    -1
     0     0     0     0     0    -1     0    -1     4
```

Mnogi gotovi algoritmi u *MATLAB*-u vješto koriste činjenicu da je matrica rijetka te na taj način postižu kraće vrijeme izvršavanja. Rezultat izvođenja skripte `test1` su vremena potrebna za umnožak dviju rijetkih, odnosno odgovarajućih verzija punih matrica.

```
function test1
    display( 'Mnozenje Neumannove i Poissonove matrice reda 1600' );
    A = gallery( 'neumann', 1600 );
    B = gallery( 'poisson', 40 );
    tic;
        A = A*B;
    toc;

    A = full( gallery( 'neumann', 1600 ) );
    B = full( gallery( 'poisson', 40 ) );
    tic;
        A = A*B;
    toc;
end
```

Dobivena vremena su:

```
      Elapsed time is 0.000695 seconds.
      Elapsed time is 0.443004 seconds.
```

Koristeći naredbu `sparse`, svaku matricu možemo pohraniti kao rijetku.

Zadatak 2. Za pseudoslučajno generirane matrice *A* i *B* reda 1000 (npr. `A = rand(1000)`) usporedite vremena izvršavanja naredbi `A*B` i `sparse(A)*sparse(B)`.

3. Računalo je izračunalo…

Prilikom izvršavanja nekog koda studenti često obrate pažnju samo na dobiveni rezultat, ignorirajući pritom na zaslonu ispisane poruke.

Primjer 3. Računanje inverzne matrice (kada to možemo izbjeći)

U numeričkoj matematici često je potrebno, za dane vektore y i z, i zadanu regularnu matricu A, izračunati vektor $x = y - A^{-1}z$. Naravno, x uvijek možemo izračunati direktno, invertirajući matricu A.

Označimo rezultat s $x1$: `x1 = y - inv(A)*z`. Drugi način je da, umjesto računanja inverza matrice A, rješavamo sustav linearnih jednadžbi $A(y - x) = z$. Općenito, u *MATLAB*-u matričnu jednadžbu oblika $CX = D$ možemo riješiti, bez računanja inverza, koristeći lijevo matrično dijeljenje – operator `\: X = C\D`.

Rješenje našeg sustava označimo s $x2$: `x2 = -A\z + y`.

Neka su x i y pseudoslučajni vektori generirani s `rand(n,1)` te neka je A pseudoslučajna kvadratna matrica reda n: `A = rand(n)`. Rezultat izvršavanja funkcije `test2`

```
function test2( n )
  A = rand(n);
  y = rand(n,1);
  z = rand(n,1);
  tic
  x1 = y - inv(A)*z;
  toc
  tic
  x2 = -A\z+y;
  toc
end
```

za $n = 10000$ je

```
      Elapsed time is 137.995938 seconds.
      Elapsed time is 45.294000 seconds.
```

Napomena. Računanje inverzne matrice treba, ako je to moguće, zamijeniti rješavanjem odgovarajućeg sustava linearnih jednadžbi.

Zadatak 3. U funkciju `test2` dodajte naredbu

```
fprintf ( 1, 'norma = %f\n',max(abs(x1-x2)) );
```

Što će se ispisati kao rezultat poziva `test2(10000)`?

Primjer 4. Raširenо je mišljenje da je svaki rezultat koji računalo „izbaci" točan. No, dobiveno i točno rješenje mogu se jako razlikovati. Promotrimo sljedeći kod.

```
function test3
A = [1 2 3;
     4 5 6;
     7 8 9];
b = A*[1;2;3];
% tocno rjesenje: [1;2;3]
y = A\b;
z = inv(A)*b;
end
```

Rezultat izvođenja skripte `test3`:

```
y = 2 0 4
z = 32 0 0
```

Zadatak 4. Zašto se točno rješenje razlikuje od dobivenih „rješenja" *y* i *z*? Možete li reći koje je od dobivenih „rješenja" točnije?

Uputa. Obratite pažnju na poruke dobivene prilikom izvršavanja. Izračunajte determinantu matrice *A*: `det(A)`.

Napomena. Rad s matricama koje su gotovo singularne treba, ako je to ikako moguće, izbjegavati.

4. Zaključak

Matematički programski sustavi nam, između ostaloga, olakšavaju numeričko računanje. Zbog efikasnosti, preporuka je da se, osim u slučajevima kada je riječ o naprednom korisniku, koriste gotove naredbe. No, uvijek trebamo imati na umu da dobivenim rezultatima ne trebamo slijepo vjerovati. Oni mogu biti pogrešni iz raznih razloga. Više informacija o vrstama pogrešaka možete naći na adresi
http://web.math.hr/~singer/num_mat/NM_0910/num_mat1.pdf.