Složenost algoritama
8. predavanje

Saša Singer

singer@math.hr

web.math.hr/~singer

PMF – Matematički odjel, Zagreb

Množenje matrica

Množenje matrica

Problem: Zadan je prirodni broj $n \in \mathbb{N}$ i 3 matrice A, B i C, reda n. Treba izračunati izraz

$$C := C + A * B.$$

Akumulacija ("nazbrajavanje") produkta $A * B$ u matrici C

- standardni je oblik BLAS–3 rutine `xGEMM` za množenje matrica,

tj. baš ova operacija se često koristi u praksi.

Usput, to će opet

- "prevariti" optimizaciju compilera,

kod višestrukog ponavljanja eksperimenta.

Množenje matrica — formula

"Matematička" realizacija matrične operacije

$$C := C + A * B$$

po elementima je trivijalna:

$$c_{ij} := c_{ij} + \sum_{k=1}^{n} a_{ik} \cdot b_{kj},$$

za sve indekse

$$i = 1, \ldots, n, \quad j = 1, \ldots, n.$$

Dakle, "programski" — treba "zavrtiti" tri petlje.

Množenje matrica — potprogram

```
      subroutine mulijk (lda, n, a, b, c)
c
c   Matrix multiply
c   C(n, n) = C(n, n) + A(n, n) * B(n, n).
c
      implicit none
c
      integer lda, n
      double precision a(lda, lda), b(lda, lda),
     $                 c(lda, lda)
c
      integer i, j, k, nn
```

Množenje matrica — potprogram (nastavak)

```
c
c    IJK loop, inner
c
      nn = n
      do 30, i = 1, nn
         do 20, j = 1, nn
            do 10, k = 1, nn
               c(i, j) = c(i, j) + a(i, k) * b(k, j)
10          continue
20       continue
30    continue
c
      return
      end
```

Permutacija petlji

Ovu varijantu algoritma zovemo `ijk` — opet po poretku (indeksa) petlji, izvana prema unutra.

Sve tri petlje možemo permutirati, tj. napisati ih u bilo kojem poretku. Na taj način dobivamo ukupno 6 varijanti algoritma, koje zovemo leksikografskim redom:

- `ijk`,
- `ikj`,
- `jik`,
- `jki`,
- `kij`,
- `kji`.

Broj operacija

U svakom prolazu kroz unutarnju petlju imamo dvije operacije:

- množenje matričnih elemenata $a_{ik} \cdot b_{kj}$,
- zbrajanje tog produkta s c_{ij}.

Sve tri petlje imaju (svaka) točno n prolaza.

Ukupan broj operacija u svim varijantama algoritma je:

$$F(n) = 2n^3.$$

Broj ponavljanja $N(n)$ izabran je tako da dobijemo približno konstantno trajanje "okolne" petlje (s ponavljanjem) kojoj mjerimo vrijeme, sve dok $N(n)$ ne padne na 1, za $n = 450$.

Boje na grafovima

Legenda za čitanje grafova:

- petlja `ijk` — zeleno, rang 3;
- petlja `ikj` — narančasta, rang 5;
- petlja `jik` — žuta, rang 4;
- petlja `jki` — ljubičasta, rang 1;
- petlja `kij` — crveno, rang 6;
- petlja `kji` — plavo, rang 2.

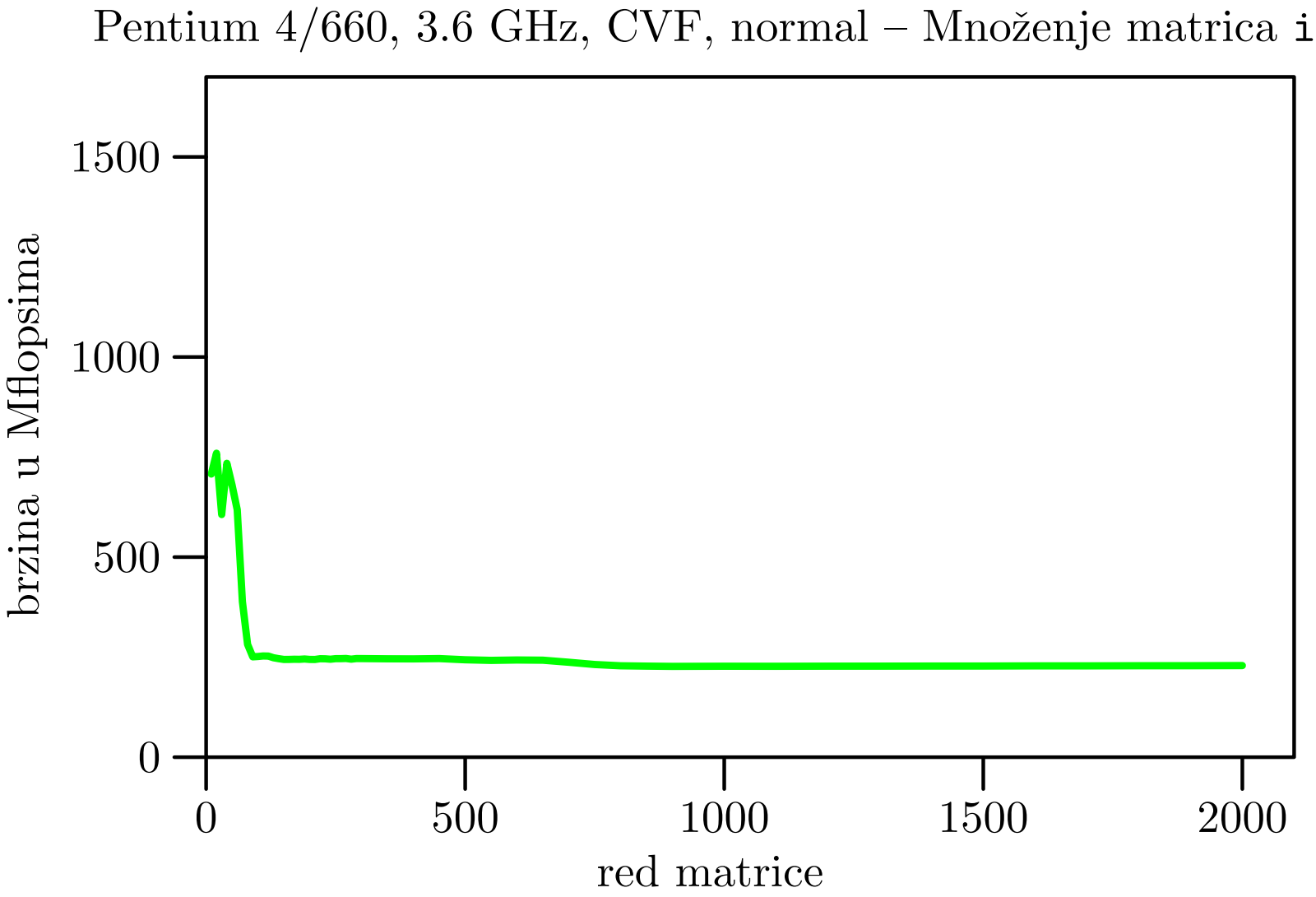
BabyBlue, CVF, normal

Compaq Visual Fortran:

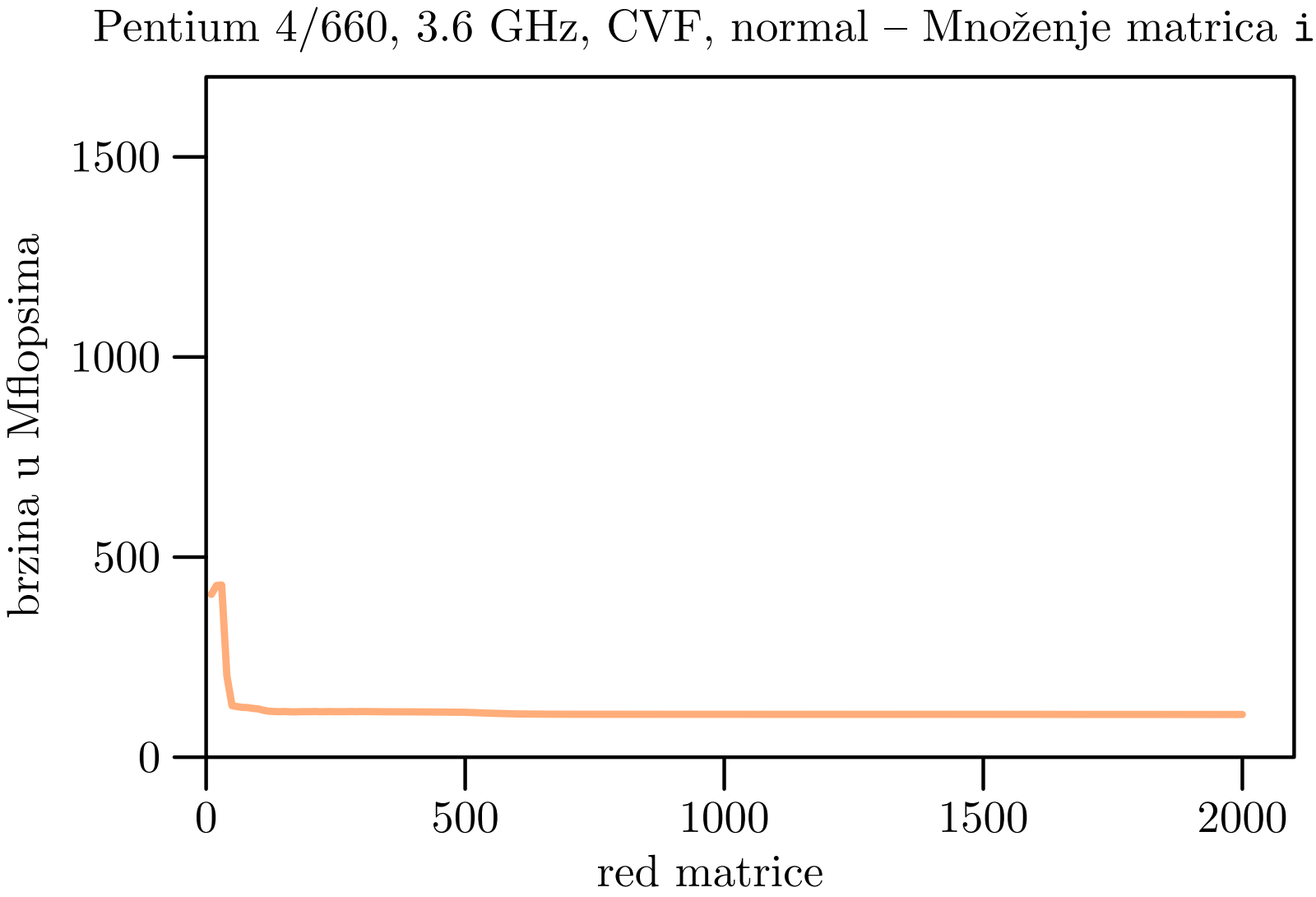
- normalna optimizacija:
  - prvo 6 pojedinačnih slika, leksikografskim redom, po petljama,
  - a zatim, zajednički graf za svih 6 petlji.
- fast optimizacija:
  - permutira petlje,

  tako da svih 6 petlji daje gotovo istu brzinu.
- Usporedba:
  - najbrže petlje `jki` u fast optimizaciji i
  - MKL-ovog algoritma `DGEMM`.

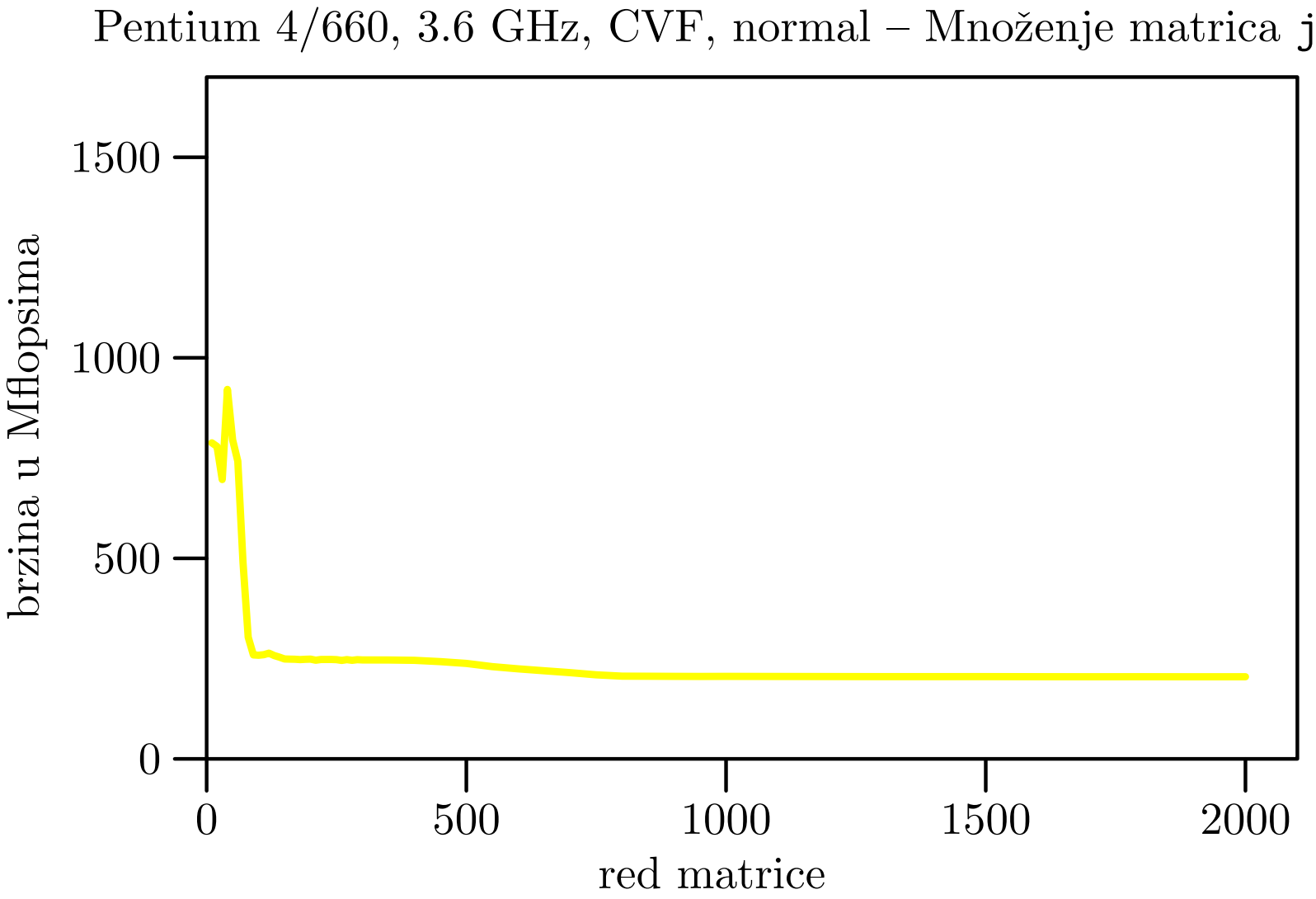
BabyBlue, CVF, normal — `ijk`



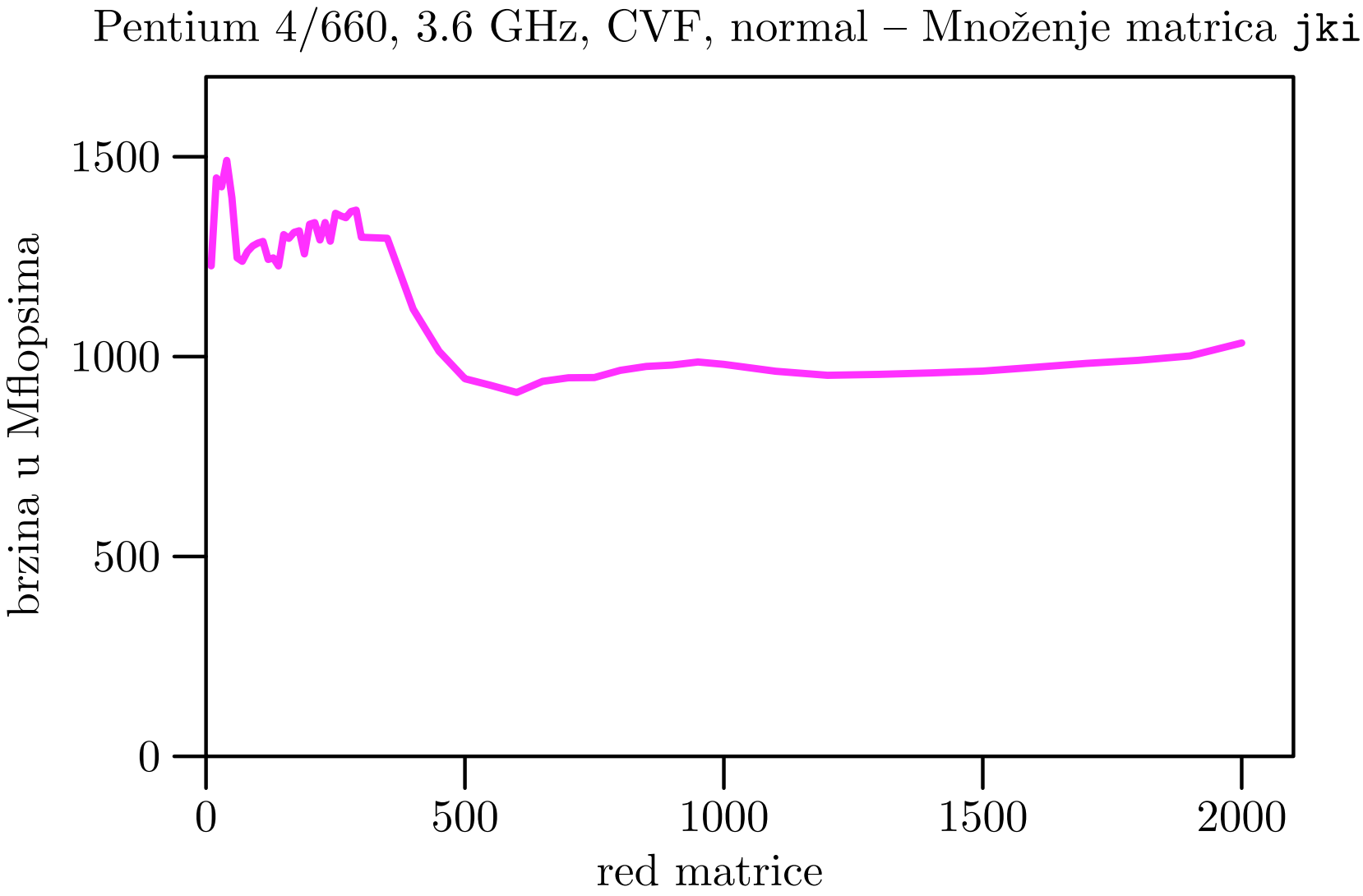
BabyBlue, CVF, normal — ikj



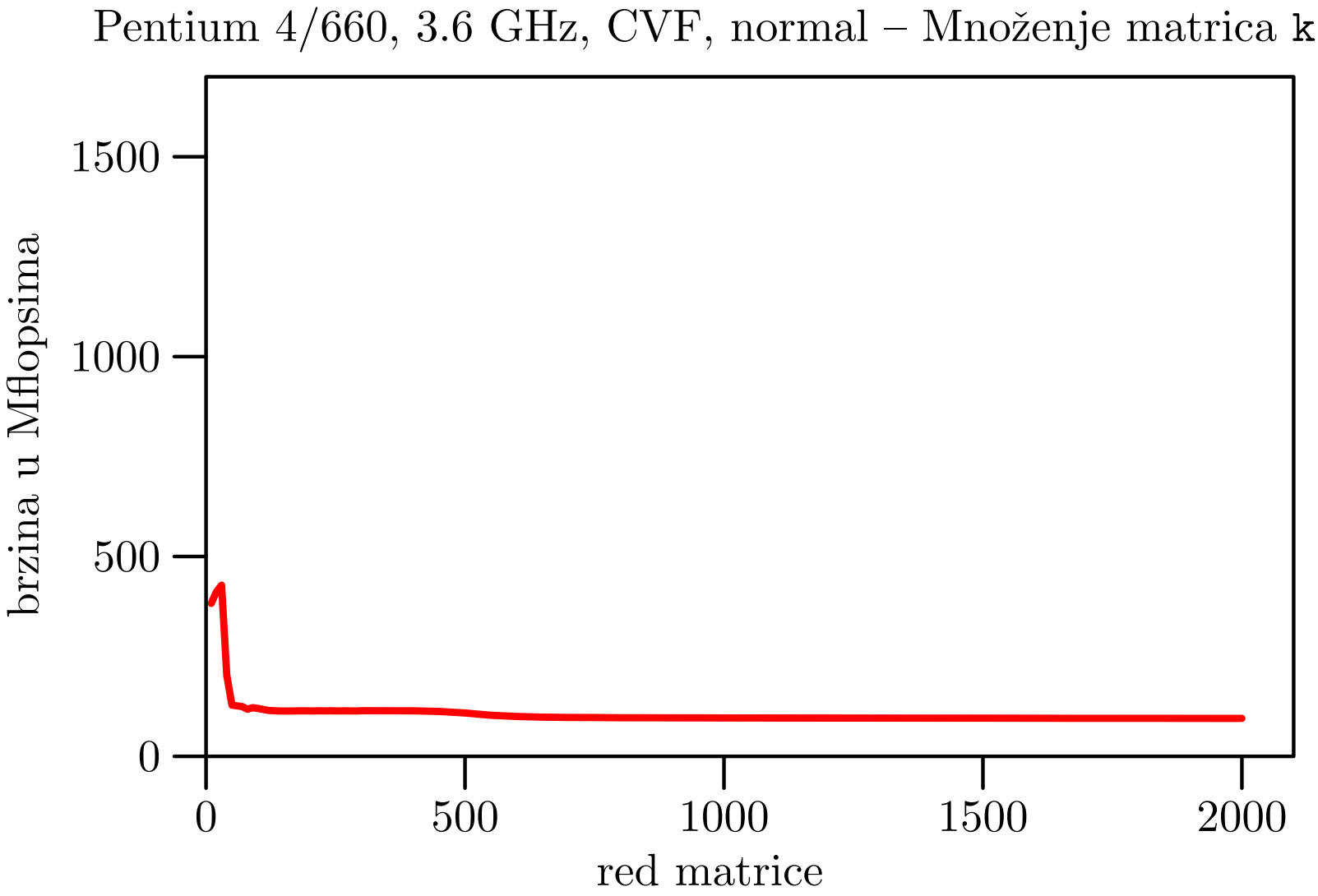
BabyBlue, CVF, normal — jik



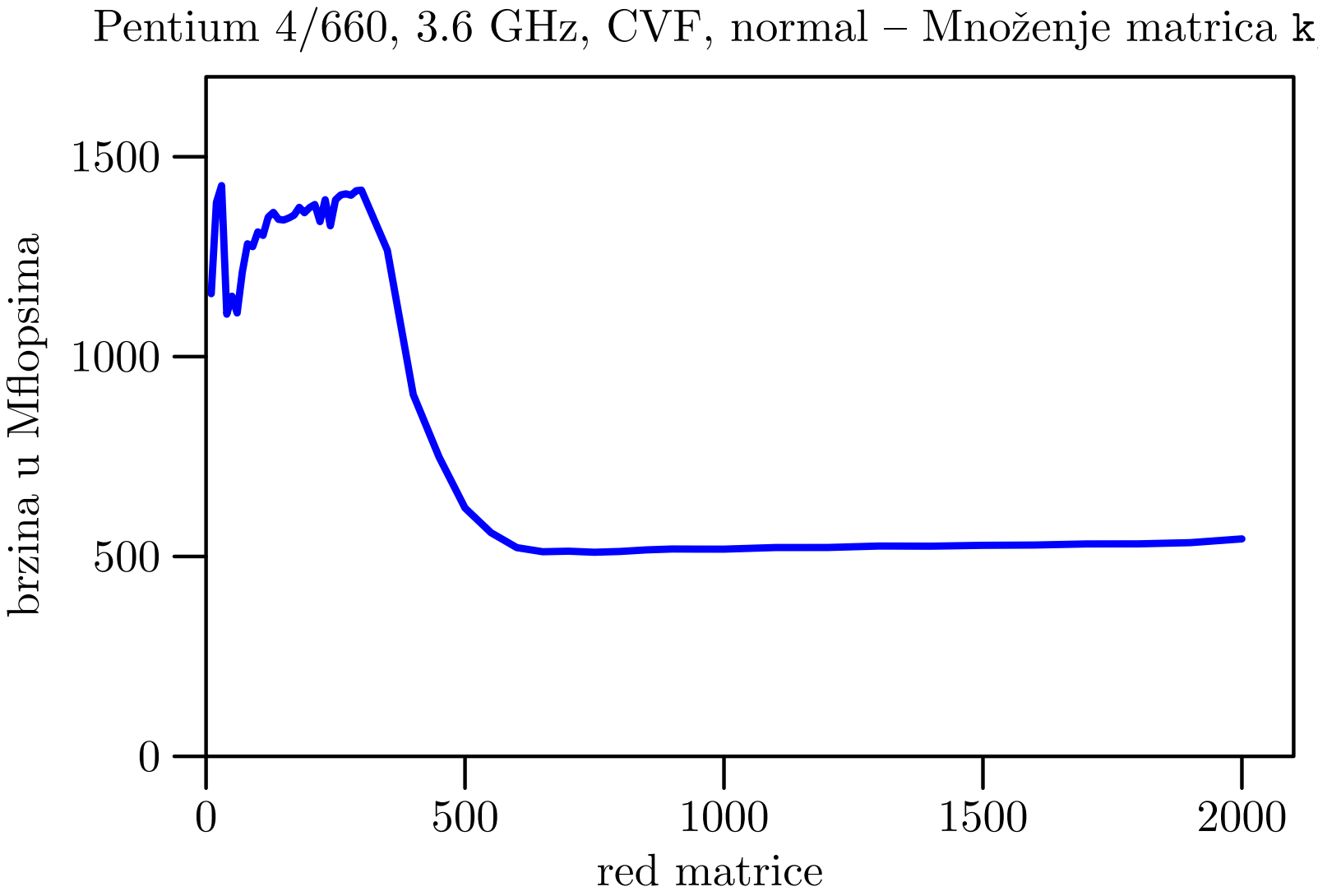
BabyBlue, CVF, normal — `jki`



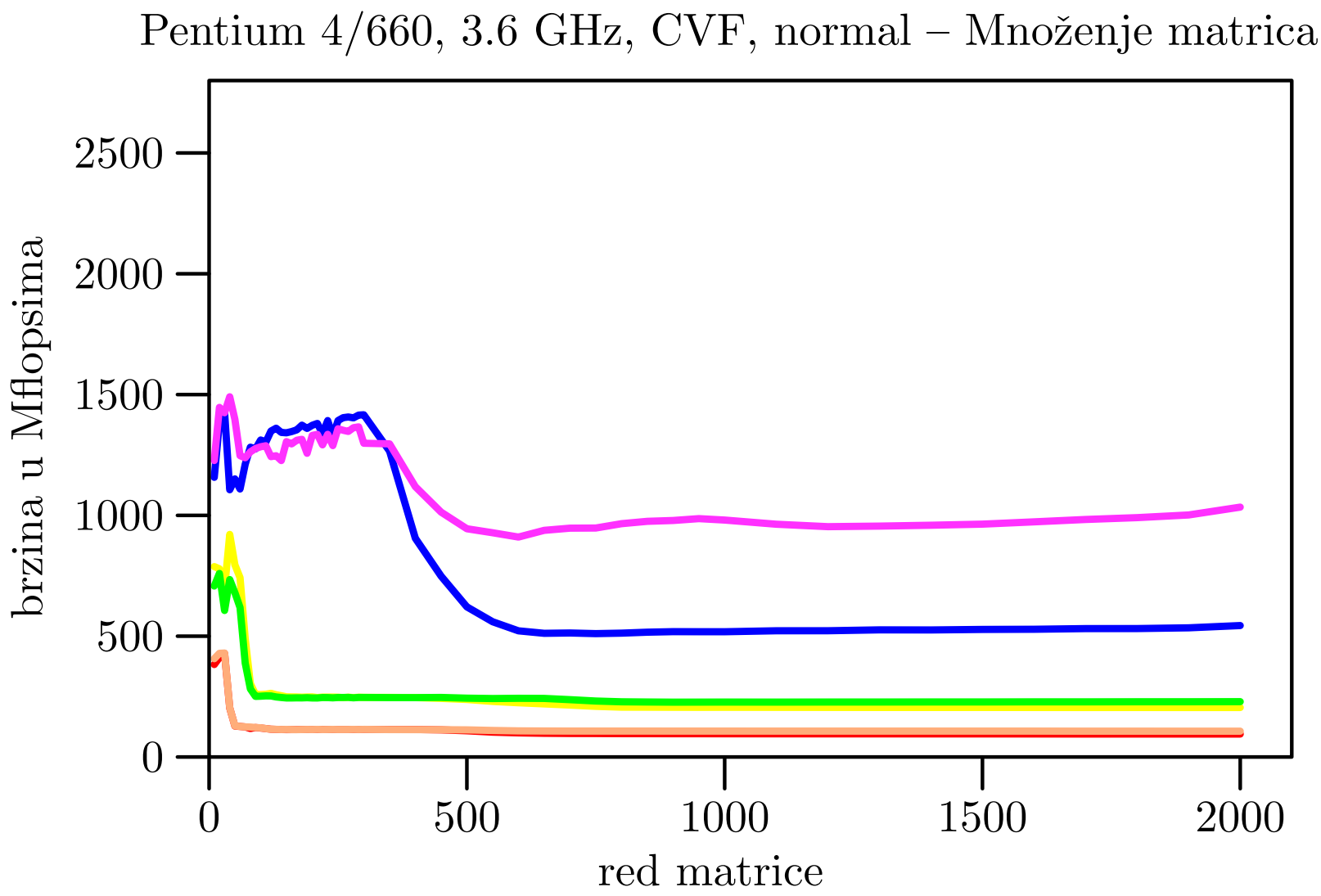
BabyBlue, CVF, normal — `kij`



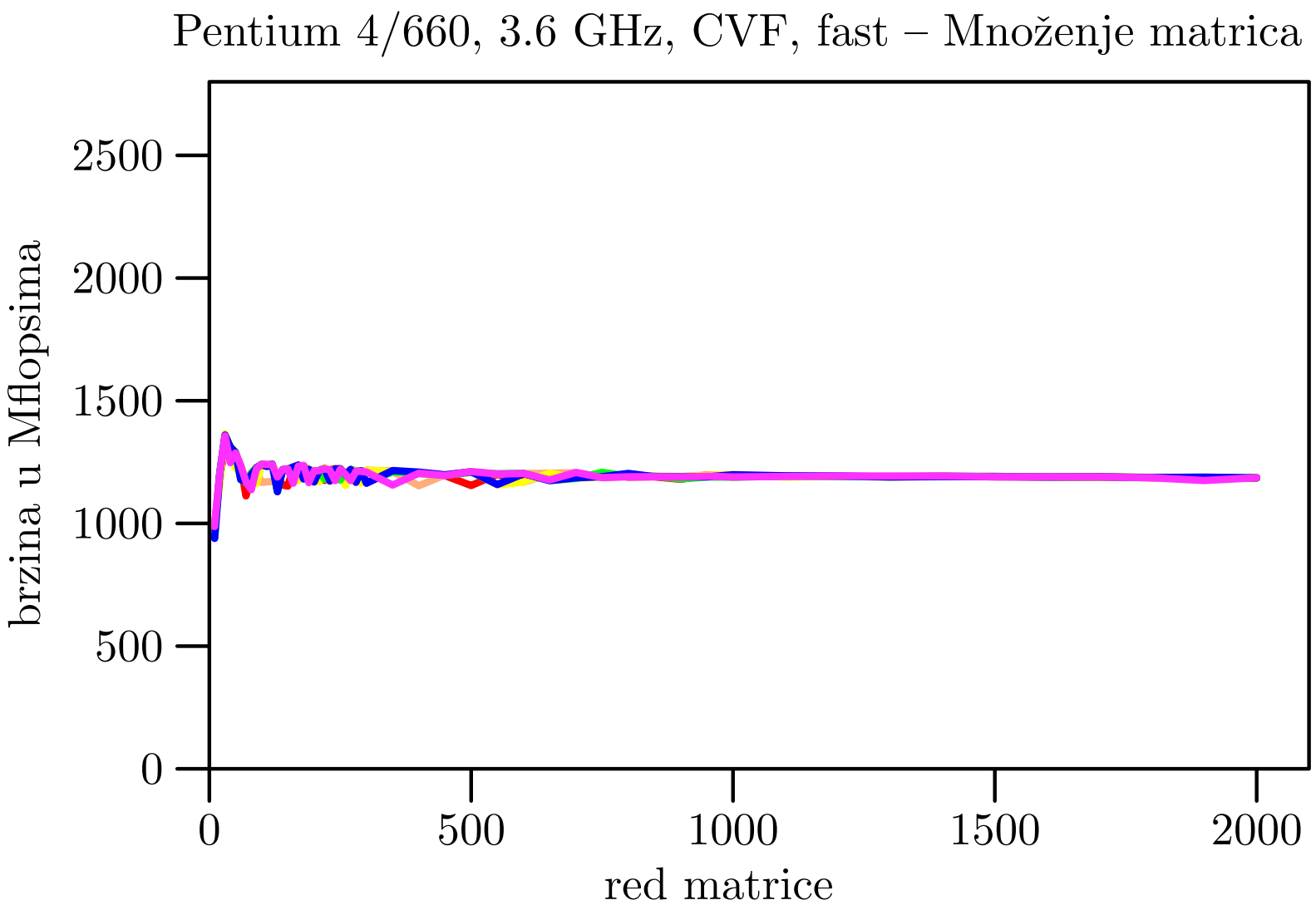
BabyBlue, CVF, normal — `kji`



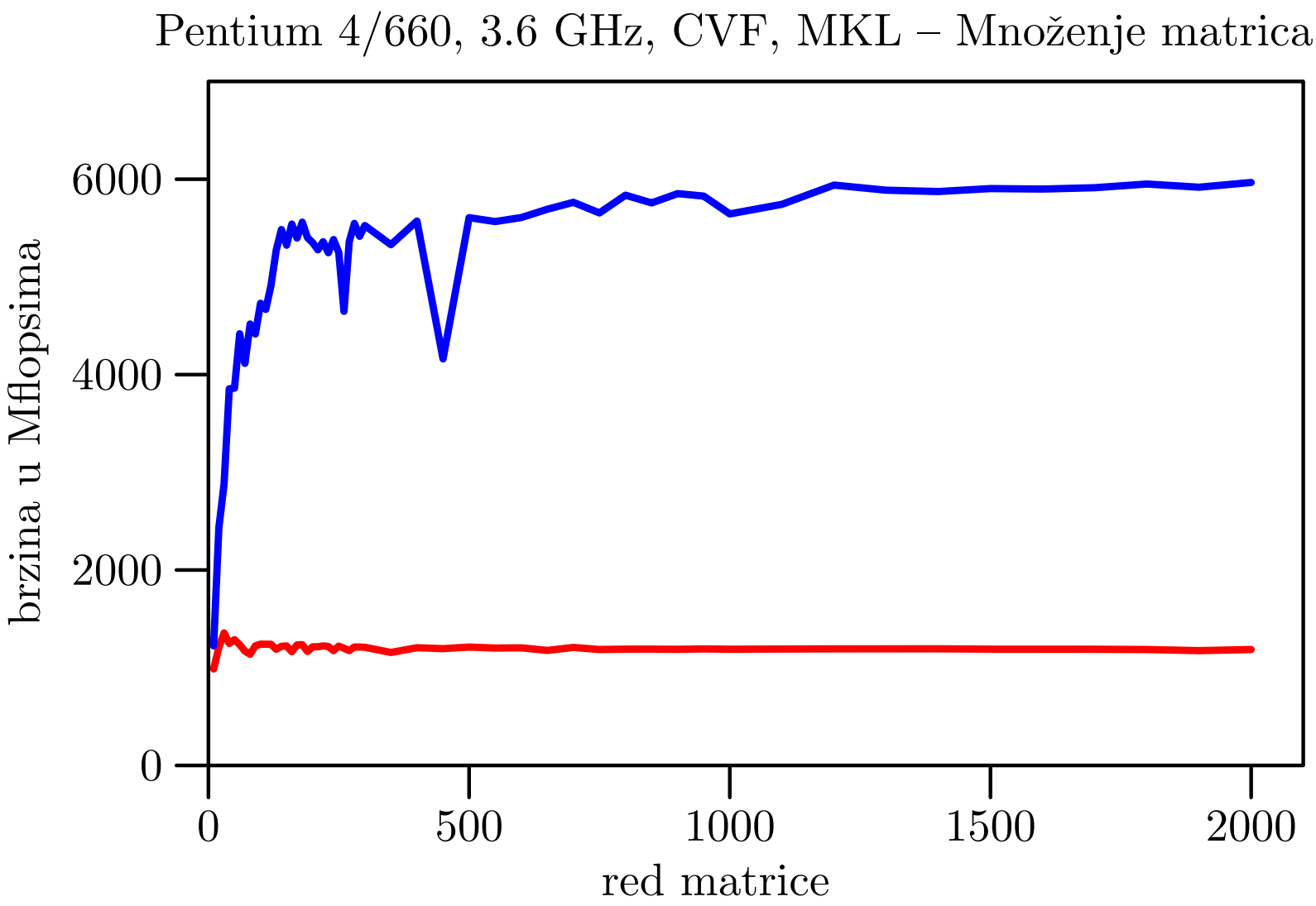
BabyBlue, CVF, normal



BabyBlue, CVF, fast



BabyBlue, CVF, fast — najbrži i MKL



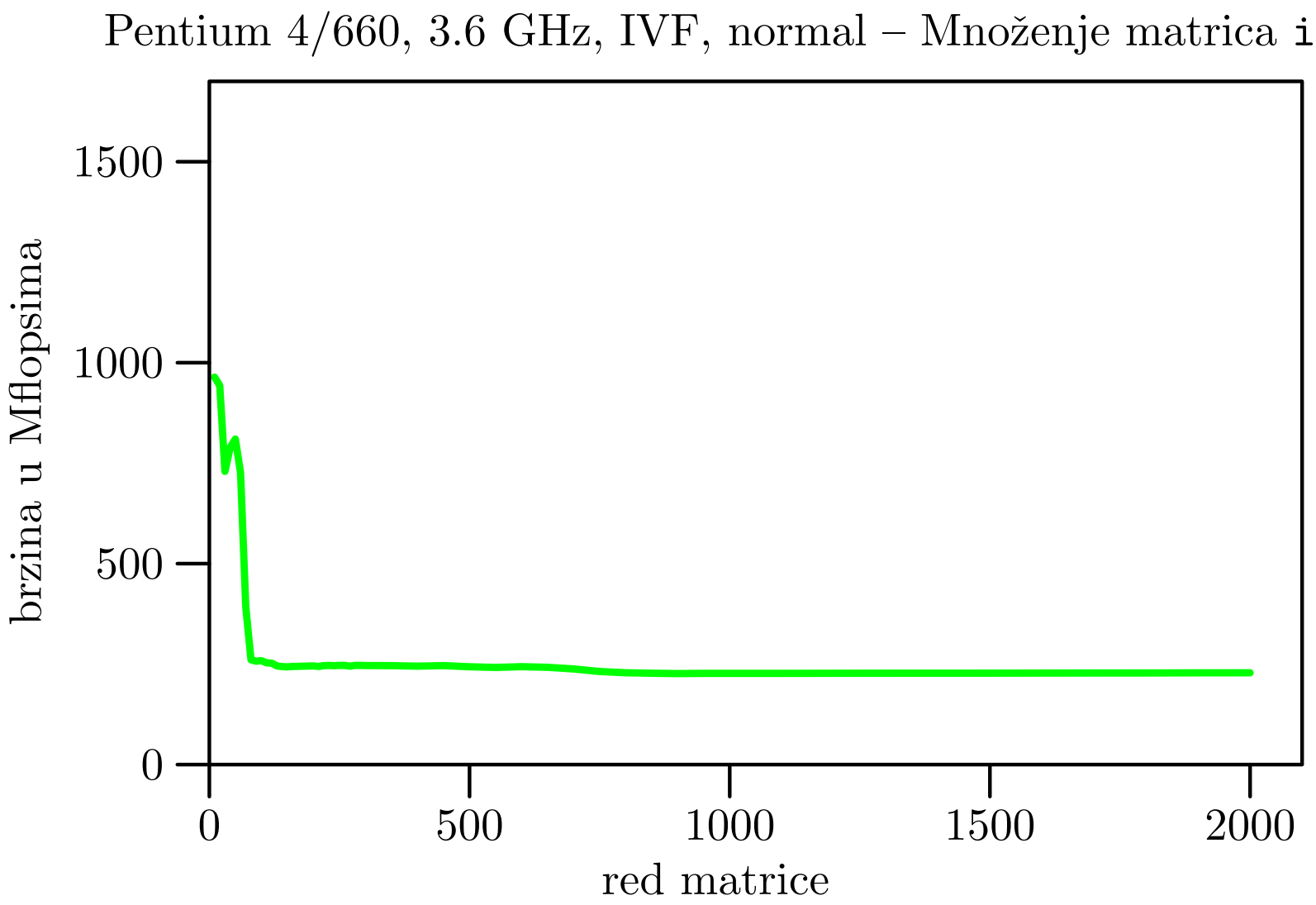
BabyBlue, IVF, normal

Intel Visual Fortran:

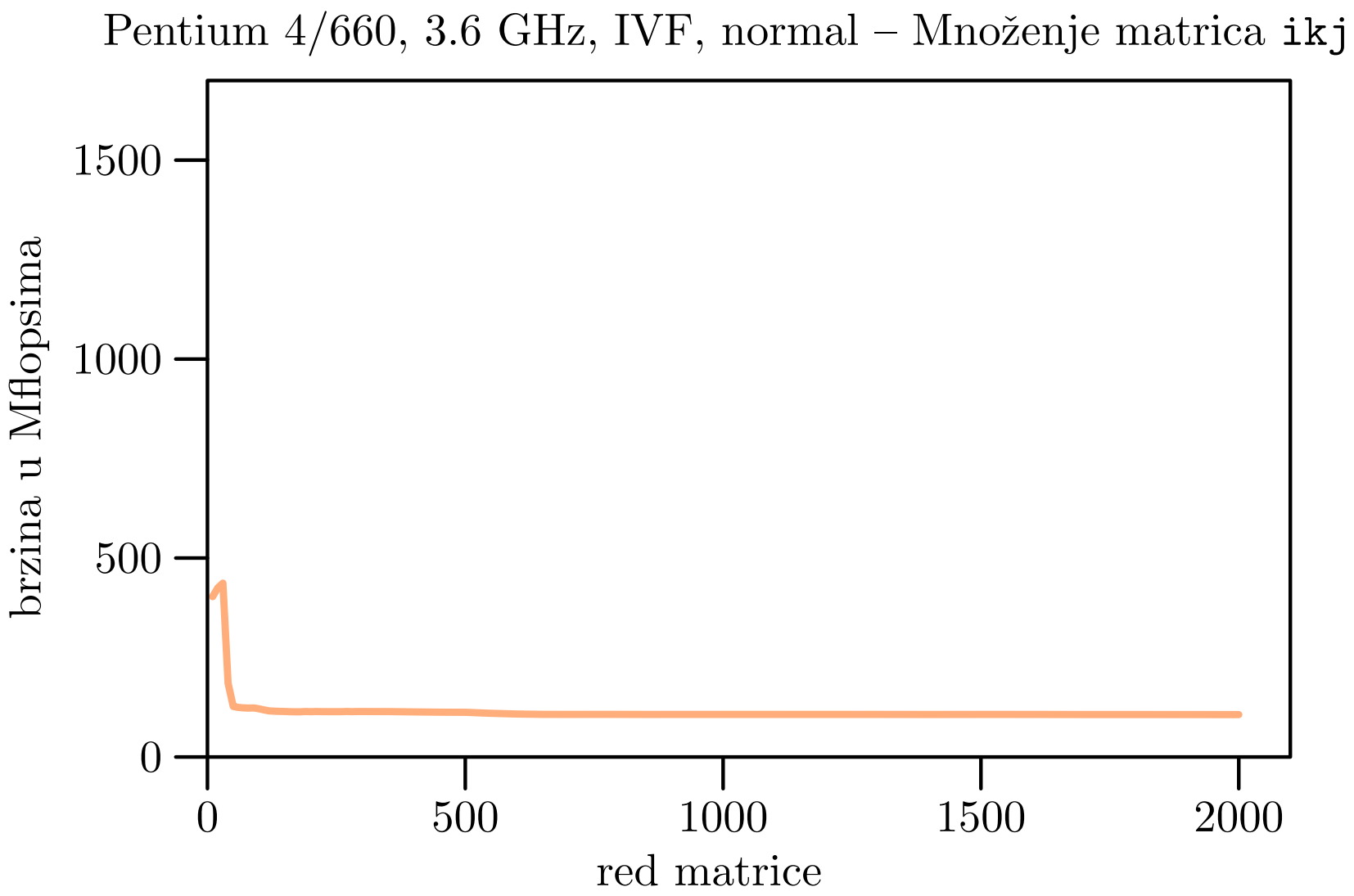
- normalna optimizacija:
  - prvo 6 pojedinačnih slika, leksikografskim redom, po petljama,
  - a zatim, zajednički graf za svih 6 petlji.
- fast optimizacija:
  - permutira petlje,

  tako da imamo 3 para petlji s gotovo istom brzinom.
- Usporedba:
  - najbrže petlje `jki` u fast optimizaciji i
  - MKL-ovog algoritma `DGEMM`.

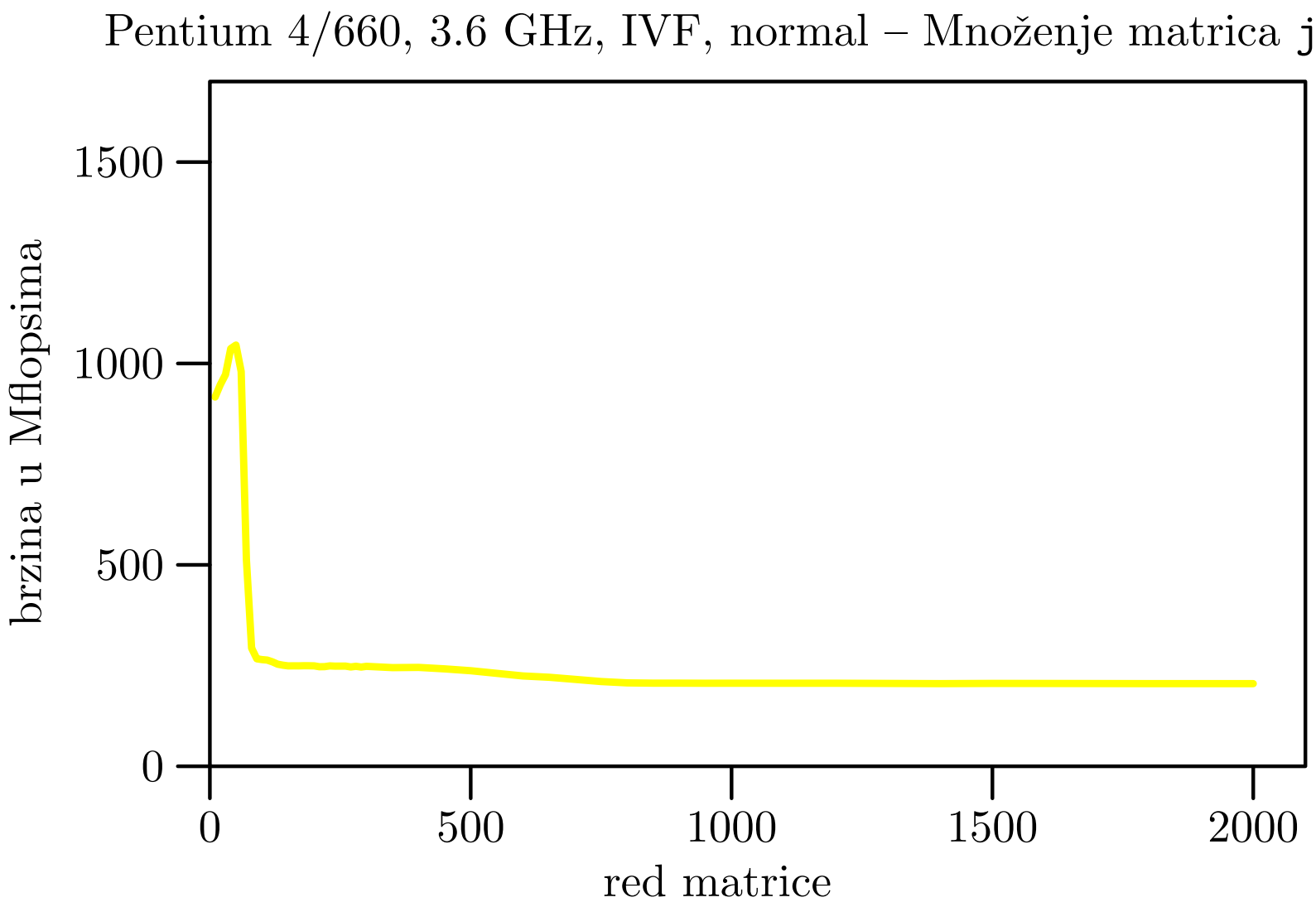
BabyBlue, IVF, normal — `ijk`



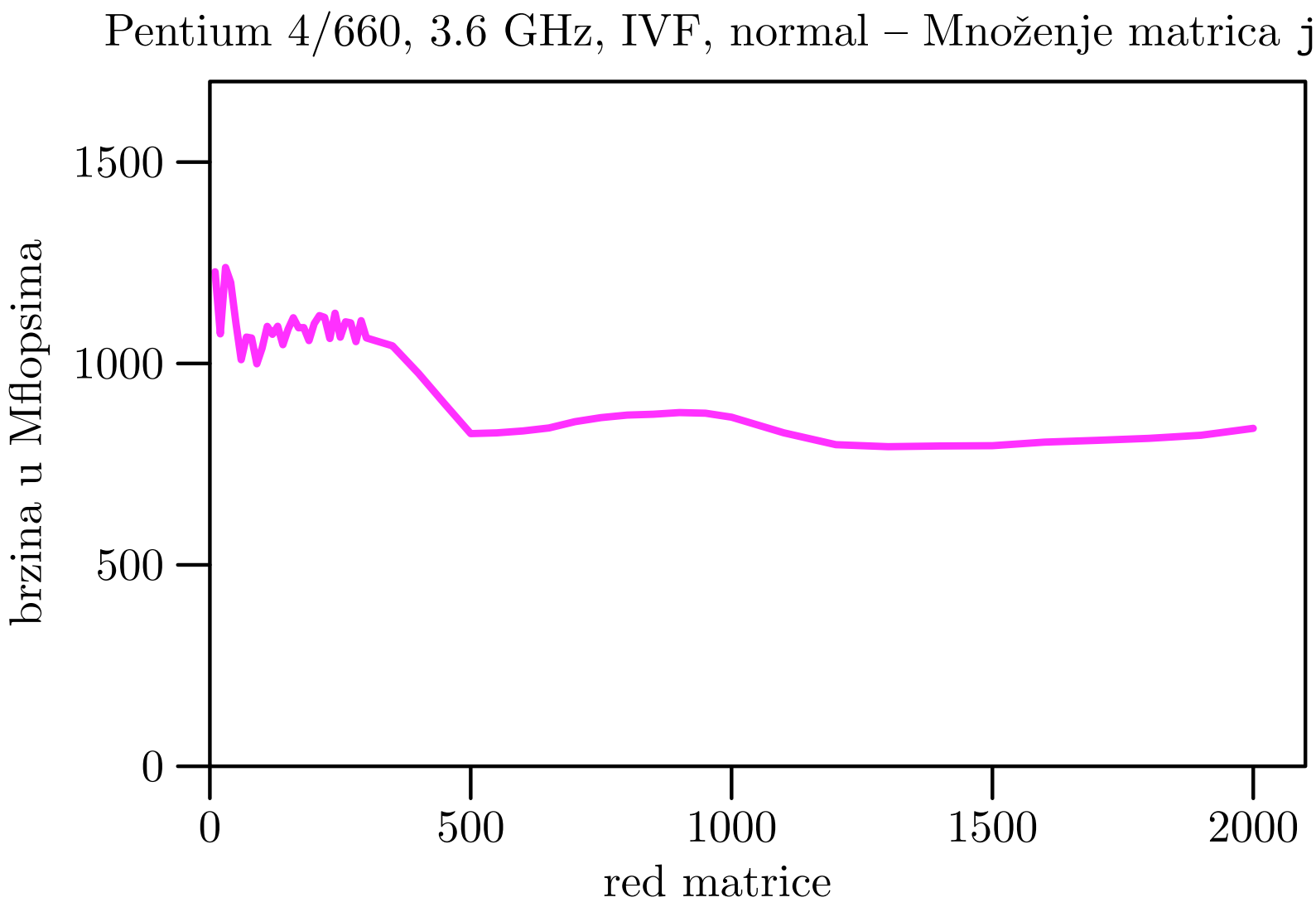
BabyBlue, IVF, normal — `ikj`



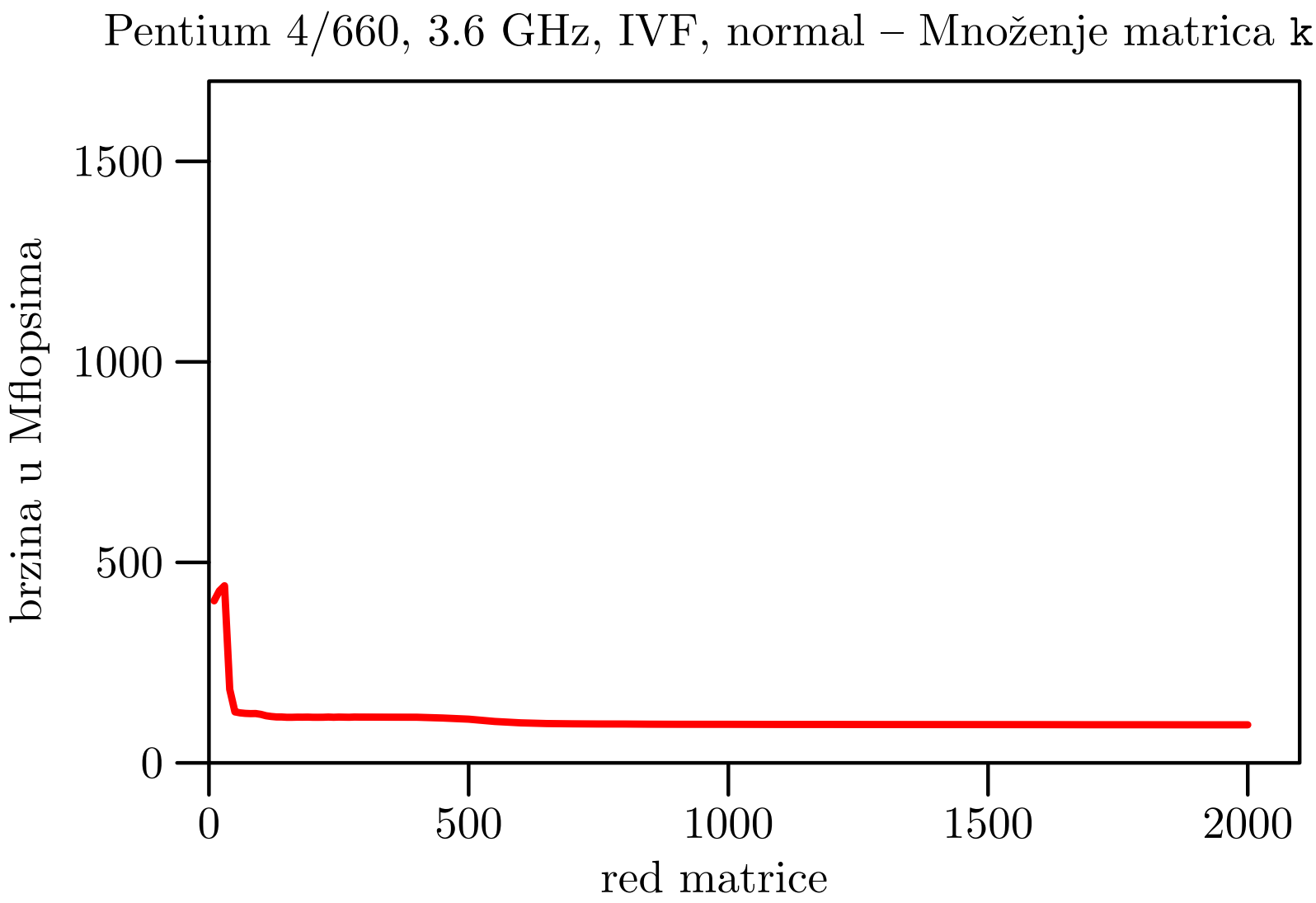
BabyBlue, IVF, normal — jik



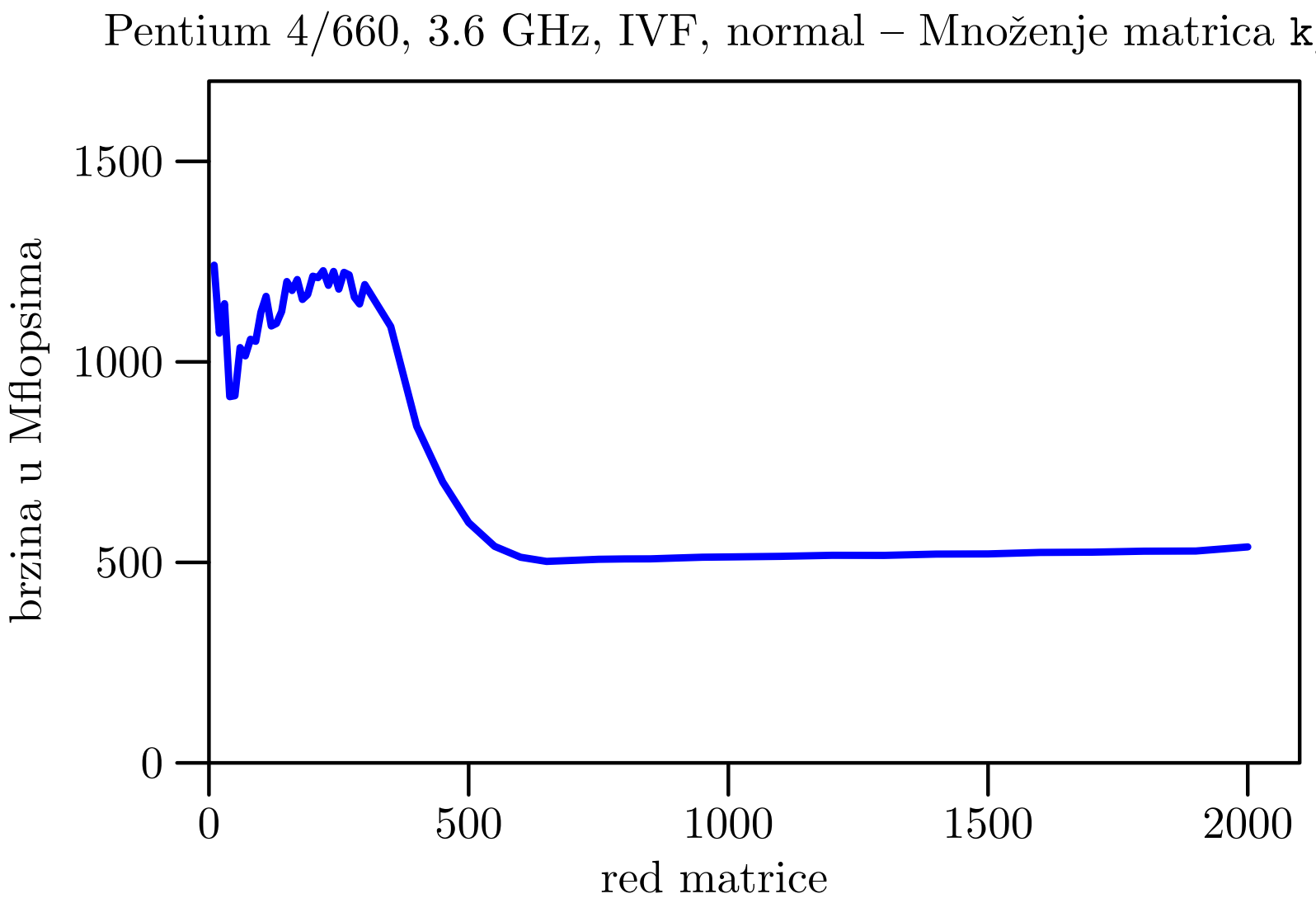
BabyBlue, IVF, normal — jki



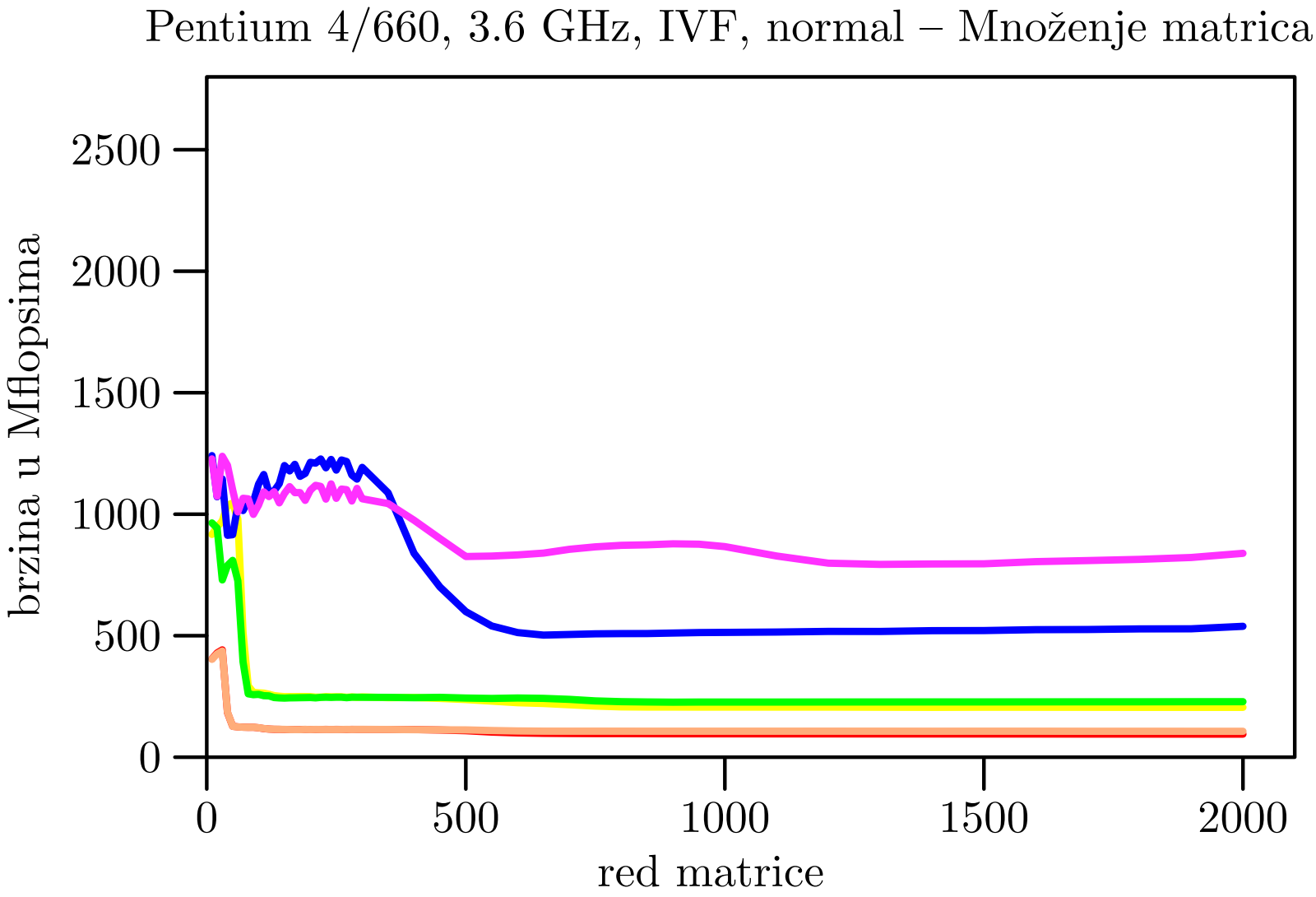
BabyBlue, IVF, normal — `kij`



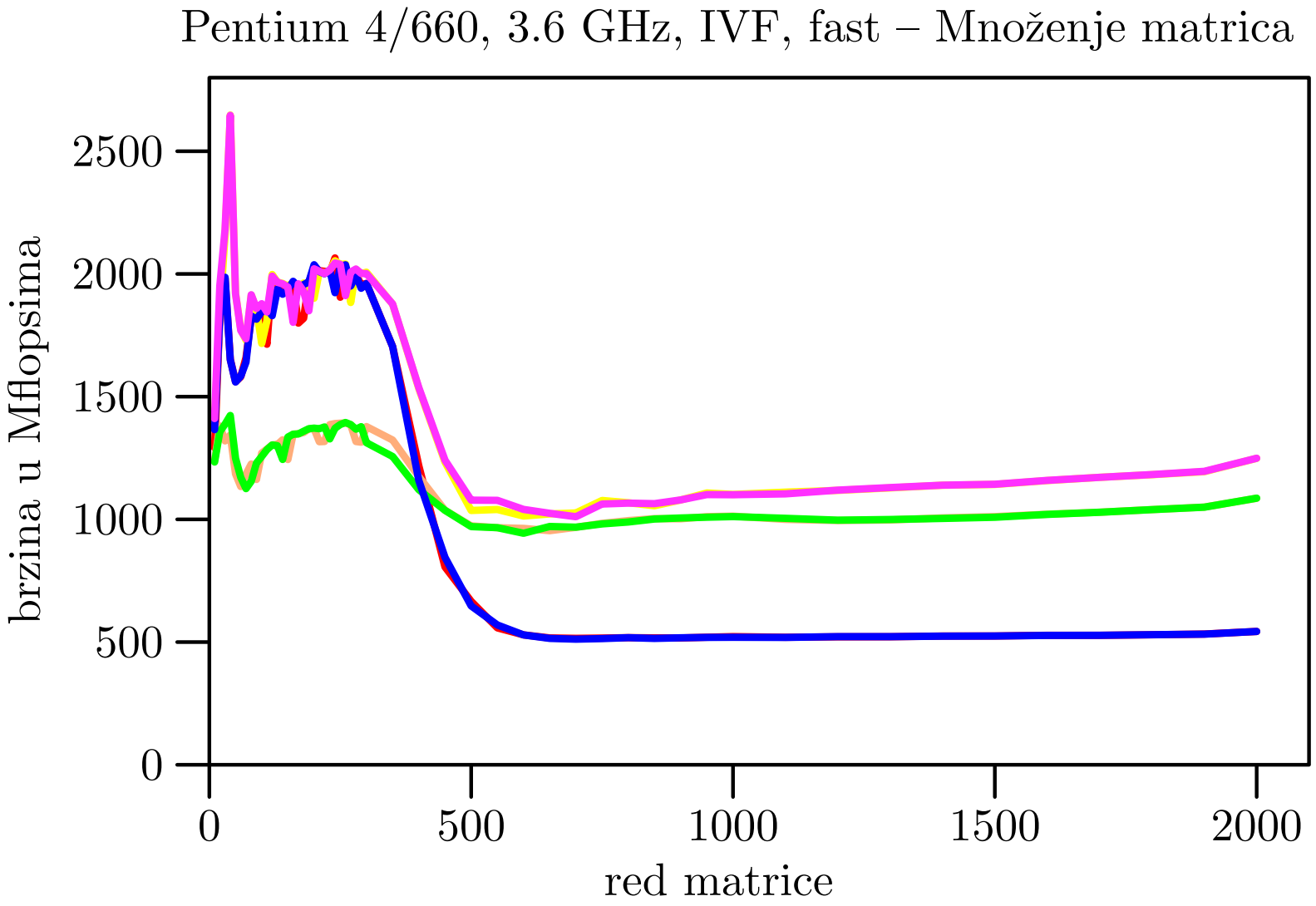
BabyBlue, IVF, normal — `kji`



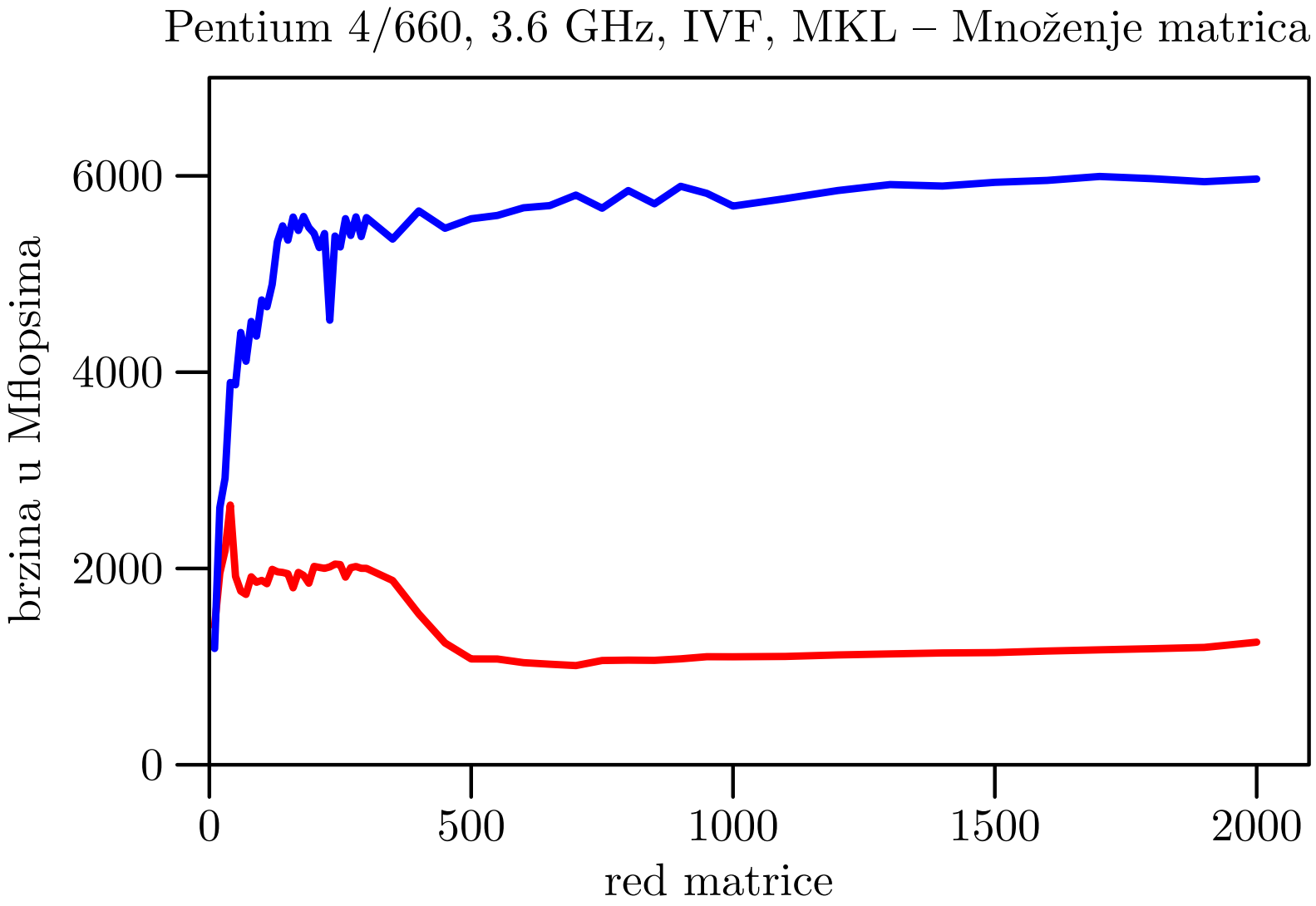
BabyBlue, IVF, normal



BabyBlue, IVF, fast



BabyBlue, IVF, fast — najbrži i MKL



Tablica brzina za velike n

Usporedba brzina (u Mflops) samo na BabyBlue:

- po petljama (uključivo i MKL),
- za normal i fast opcije kod oba compilera.

Petlja	normal CVF	normal IVF	fast CVF	fast IVF
`ijk`	229.0	228.4	1186.3	1086.4
`ikj`	106.7	106.5	1186.5	1086.0
`jik`	204.8	205.1	1185.3	1248.7
`jki`	1034.2	839.0	1186.4	1249.1
`kij`	95.0	94.9	1185.1	543.3
`kji`	544.0	538.6	1185.7	543.3
MKL	5945.7	5967.3	5966.5	5967.6

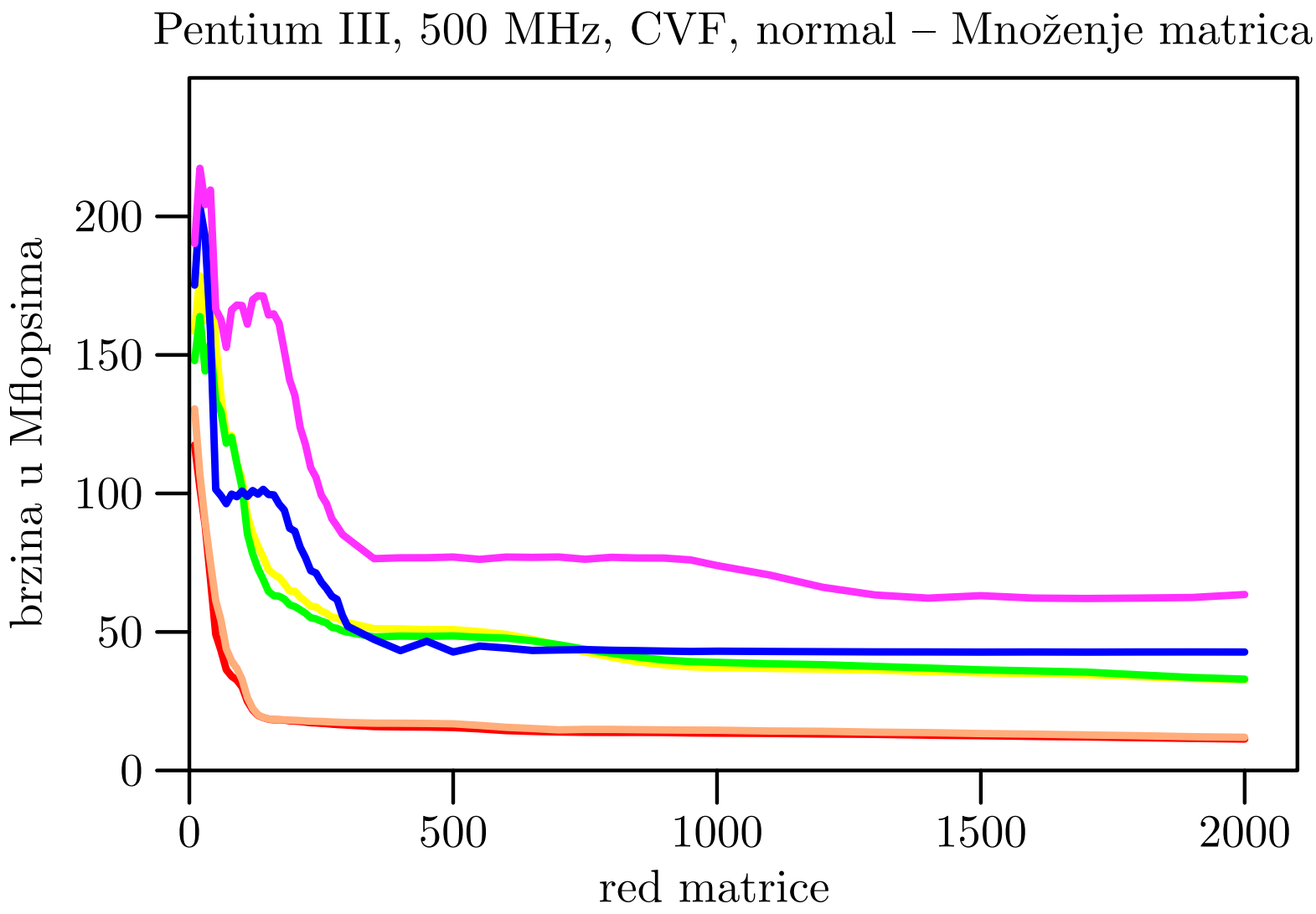
Ostala računala

Vrlo slično ponašanje brzina za petlje vidi se i na ostalim računalima.

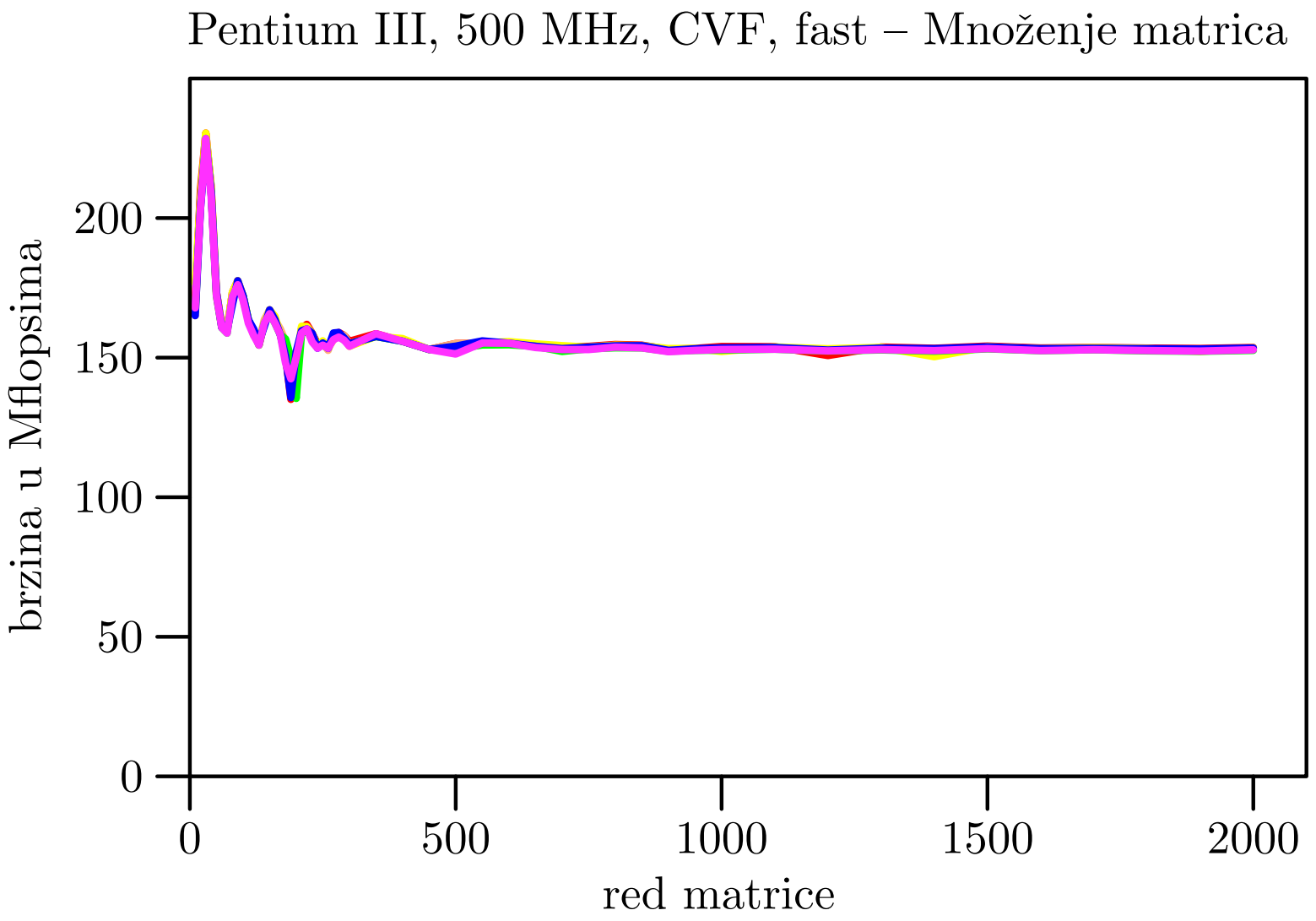
Grafovi su "skraćeni" tako da sadrže redom:

- usporedbu brzina svih 6 petlji za normal i fast opcije compilera (samo CVF),
- usporedbu najbrže fast petlje MKL-a.

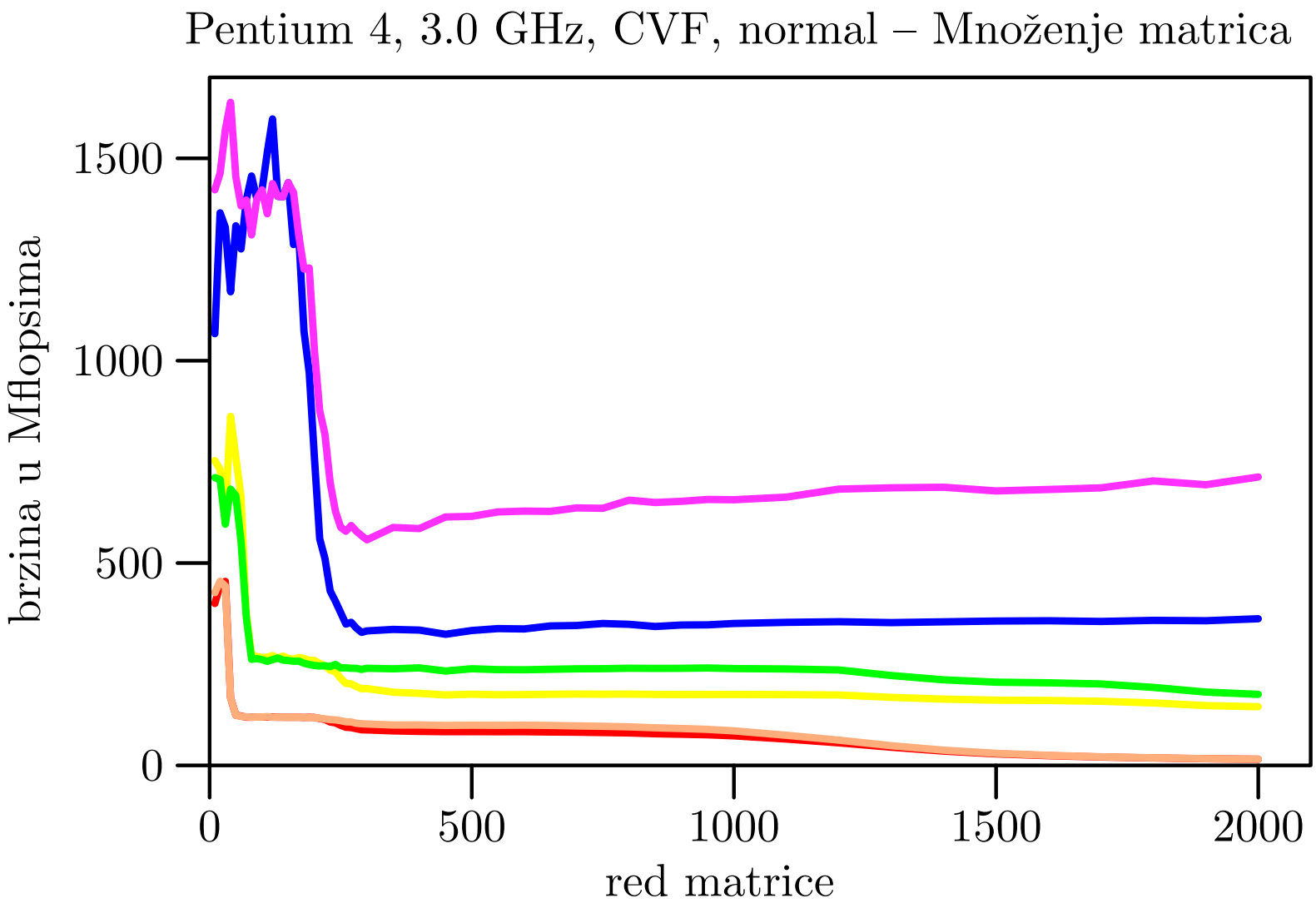
Klamath5, CVF, normal



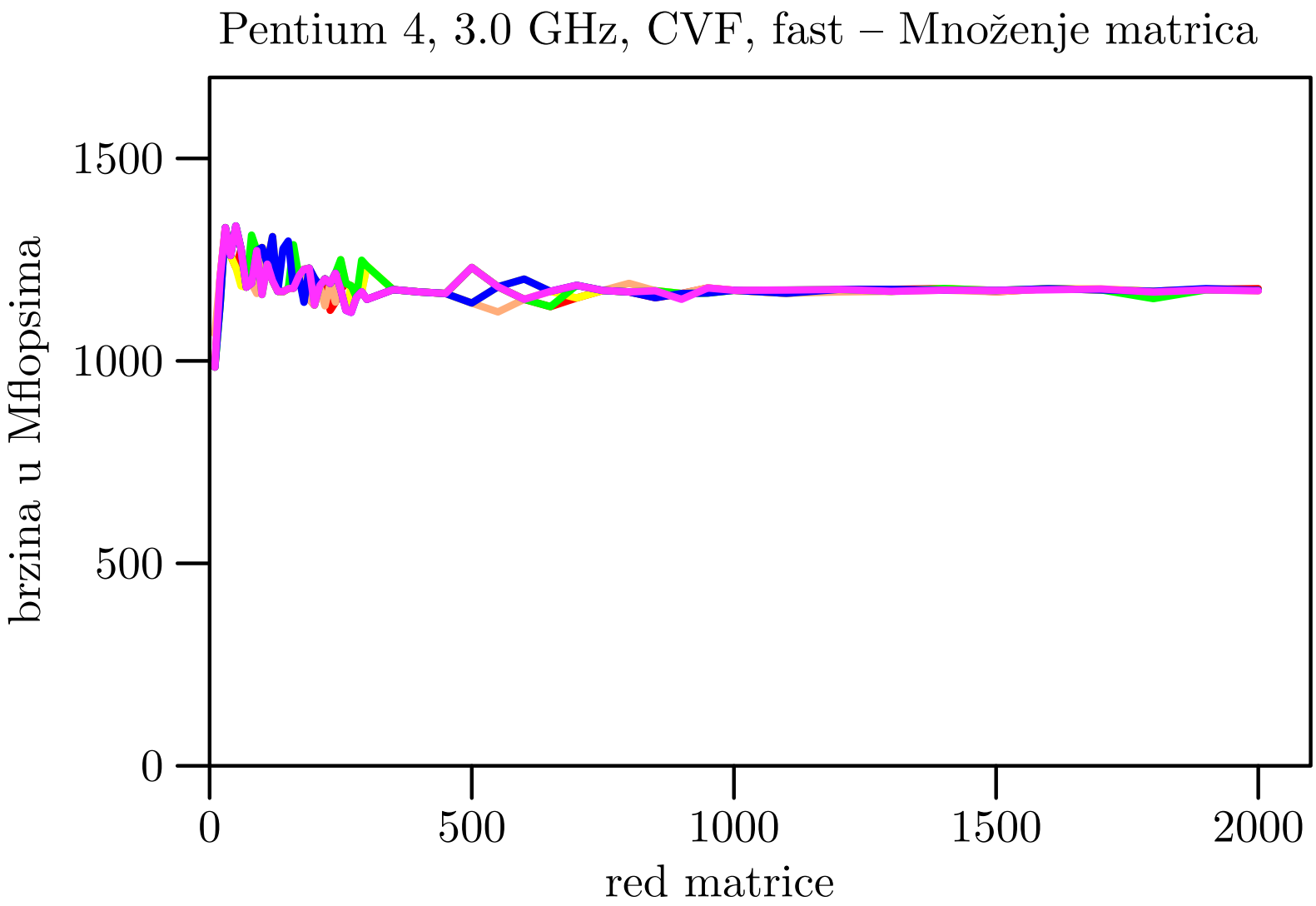
Klamath5, CVF, fast



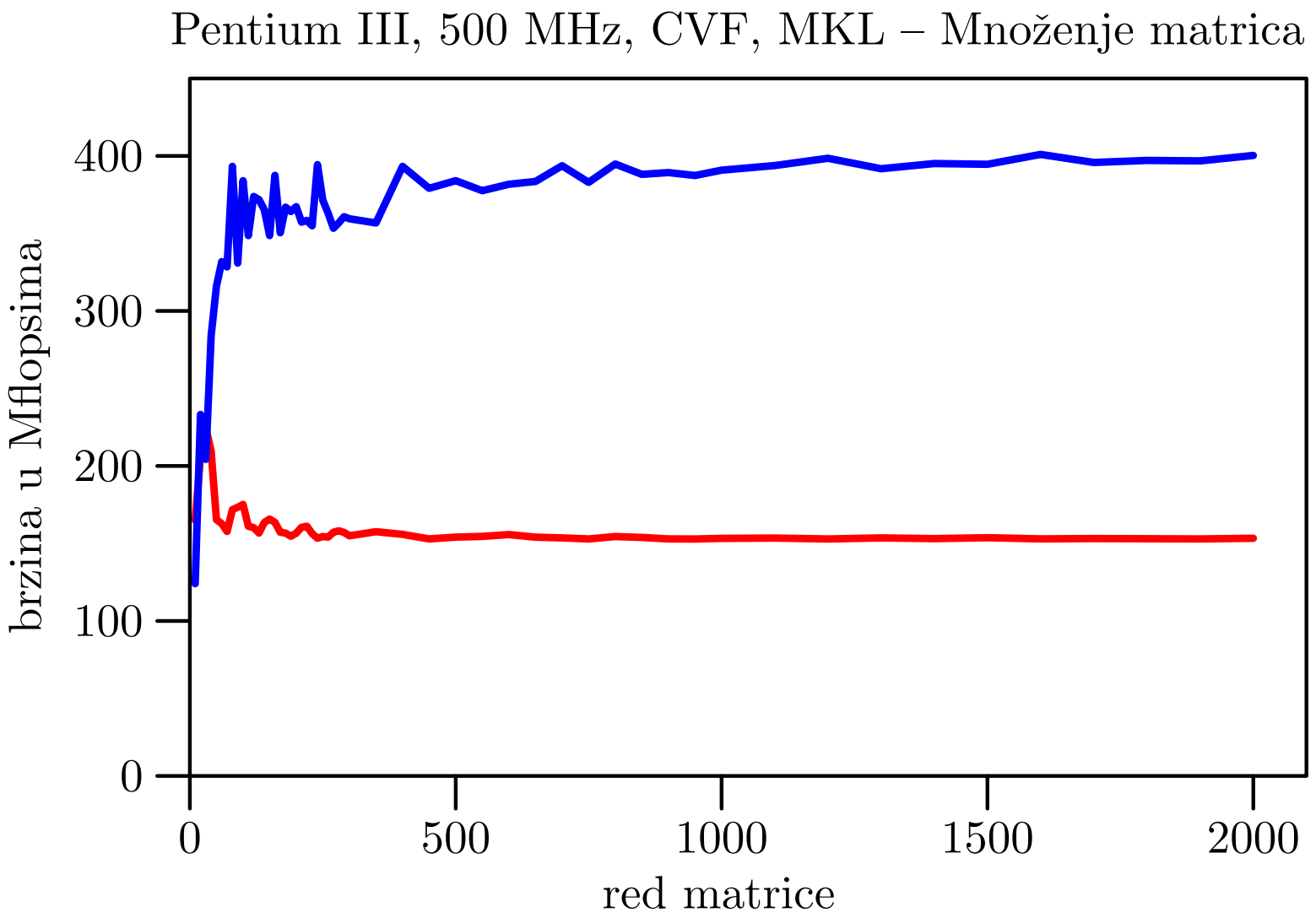
Veliki, CVF, normal



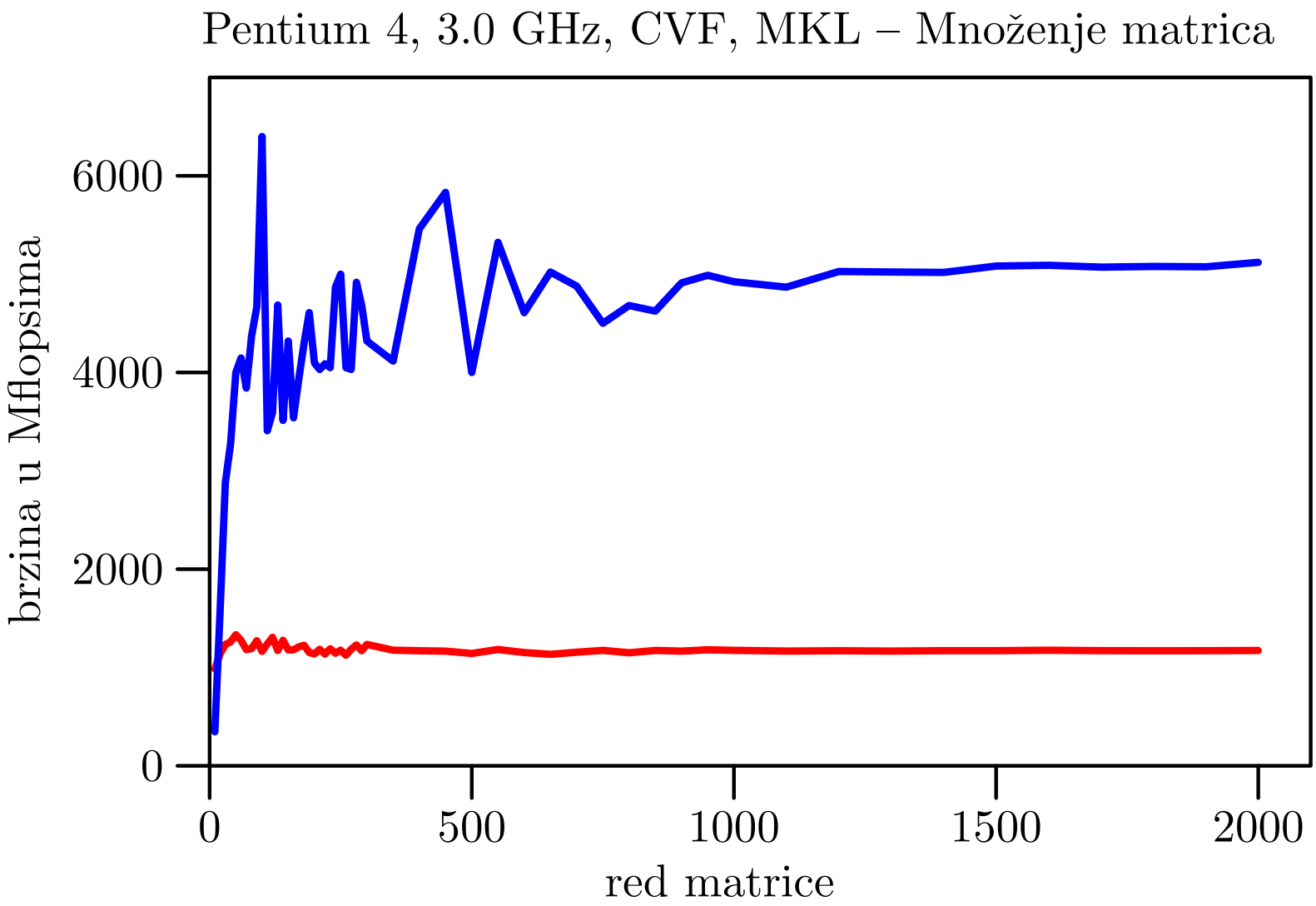
Veliki, CVF, fast



Klamath5, CVF, fast — najbrži i MKL



Veliki, CVF, fast — najbrži i MKL



Komentar rezultata

Kod množenja matrica, za razliku od zbrajanja,

- svaki ulazni podatak koristimo puno puta, (preciznije, točno n puta).

Zato brzina cache memorije može doći do izražaja, pa možemo dobiti

- bitno veće brzine nego kod zbrajanja.

Cache memorija je "glavni krivac" za:

- razlike u brzinama između raznih varijanti, i
- povećanu brzinu za male n-ove.

Ponavljanje eksperimenta ima ulogu samo za vrlo male redove n.

Komentar rezultata (nastavak)

Brže su one varijante koje

- učestalije koriste iste podatke, dok su oni još u cacheu.

Dokaz: "Blokovskom" realizacijom algoritma

- za velike n možemo postići gotovo iste brzine kao i za male n (tj. spriječiti pad brzine).

Ovo, naravno, ide samo onda kad

- za velike n dobijemo pad brzine.

U protivnom, compiler se "već pobrinuo" da optimalno iskoristi cache.

Primjer za IVF da to radi za normal, pa čak i za fast opciju.

Blokovsko množenje matrica

Blokovsko množenje matrica — primjer

IVF s normal opcijom za `jik` petlju daje brzine:

- 1050 MFlops za $n \leq 50$,
- 205 MFlops za velike n.

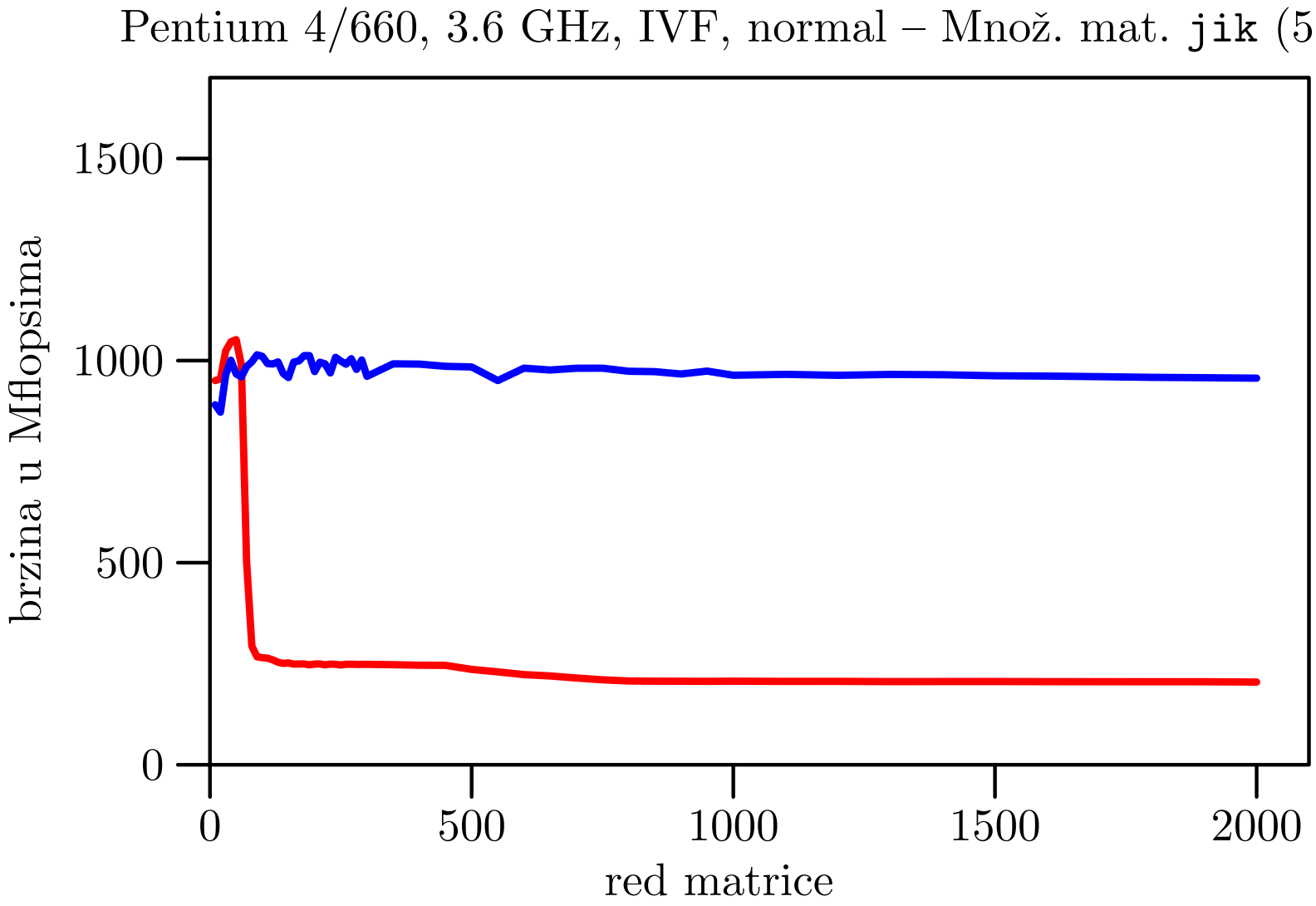
IVF s normal opcijom za `jki` petlju daje brzine:

- 1100 MFlops za $n \leq 300$,
- 840 MFlops za velike n.

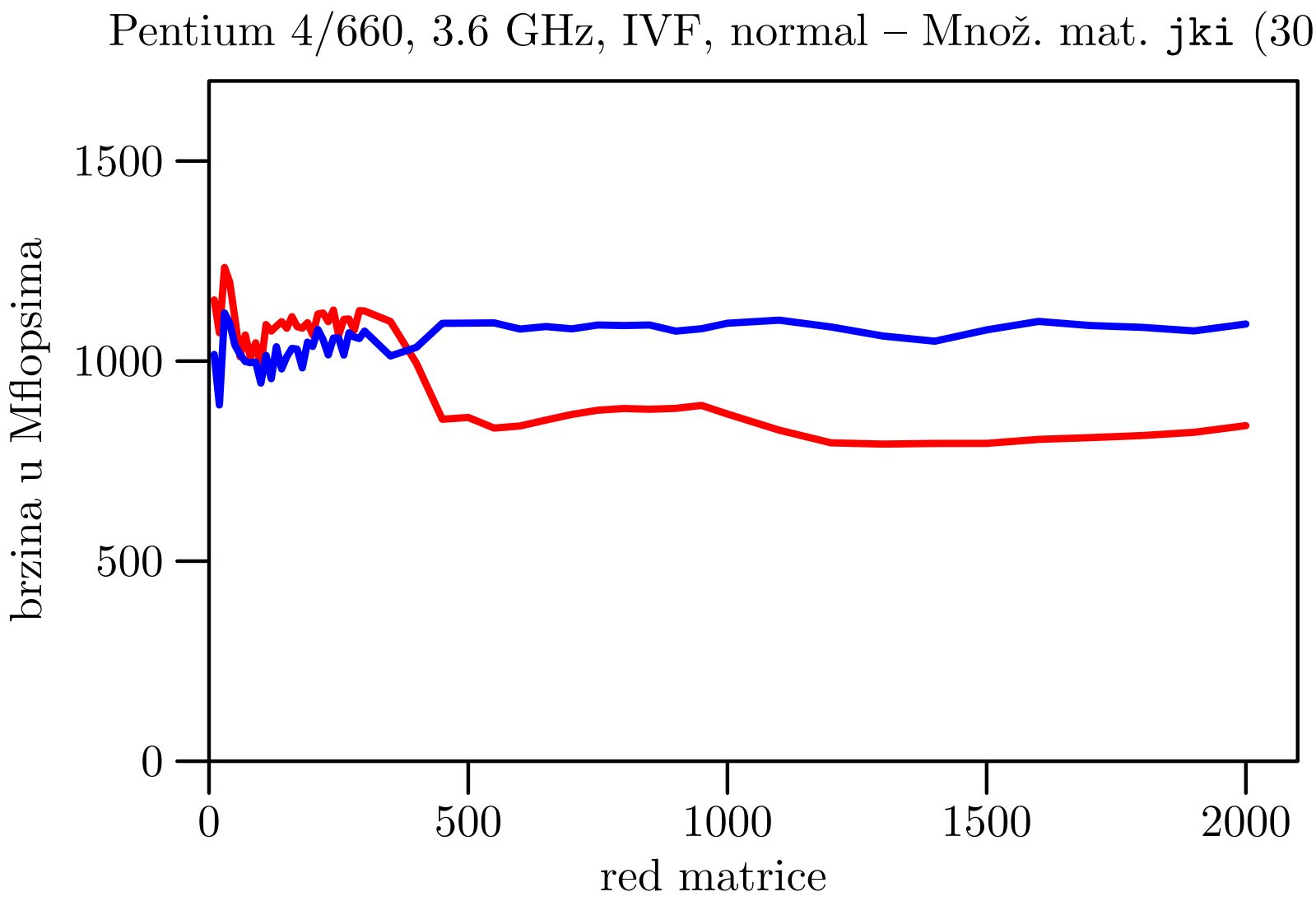
IVF s fast opcijom za `jki` petlju daje brzine:

- 2000 MFlops za $n \leq 300$,
- 1250 MFlops za velike n.

BabyBlue, IVF, normal — jik običnI i blok (50)



BabyBlue, IVF, normal — jki običnі i blok (300)



BabyBlue, IVF, fast — jki običnI i blok (300)