Je Matlab vhodné výpočetní prostředí? Úvod do IT++

Václav Šmídl,
smidl@utia.cas.cz

June 25, 2008

Outline

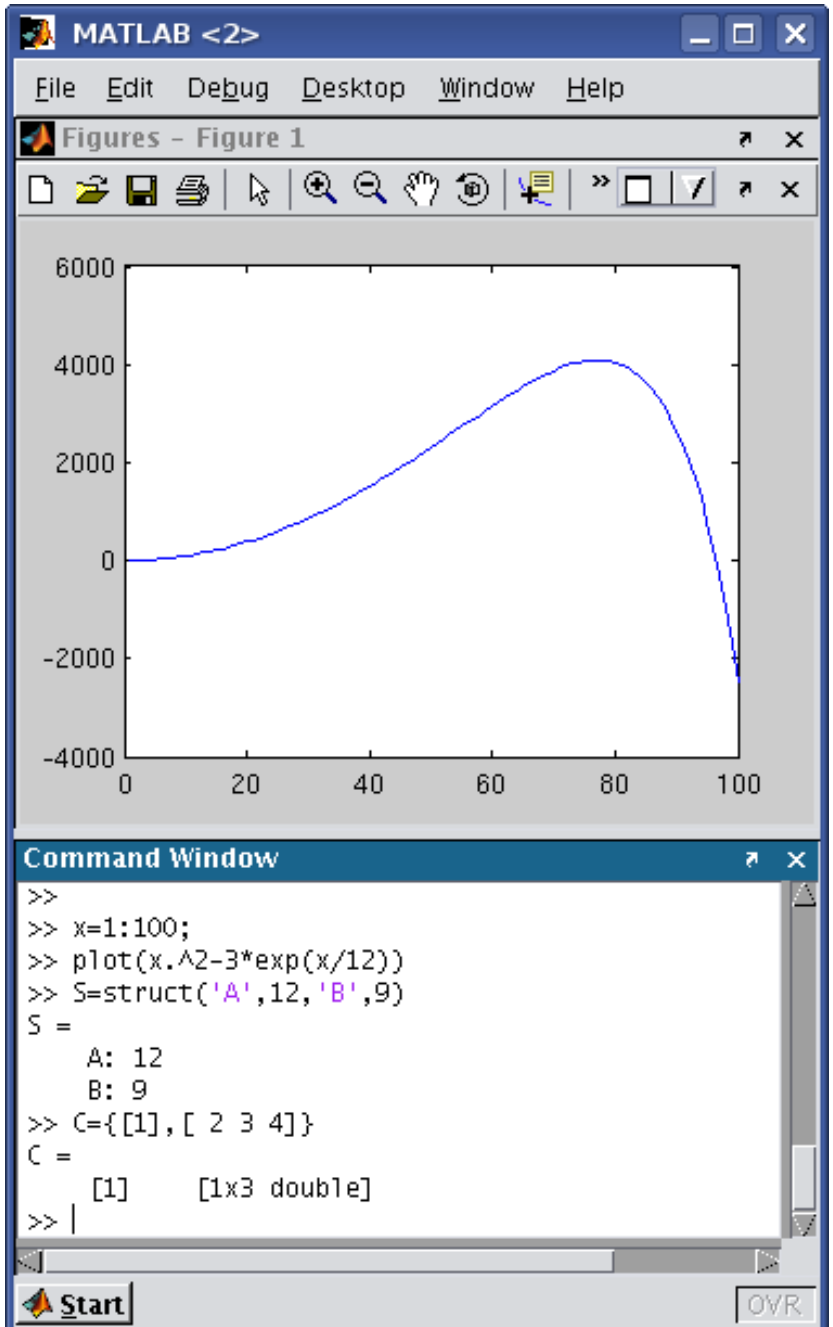
Matlab

Výhody:

- jednoduchost a uživatelská přístupnost
- interaktivní prostředí
- automatická inicializace proměnných
- snadné vytváření struktur
- zobrazovací funkce

Cena?

- licence
- výkon
- ještě něco?



Projekt Mix3k

Knihovna na podporu distribuovaného dynamického rozhodování.

1. **Podpora pro širokou škálu dynamických modelů;** discrétní a spojité proměnné, které můžou být korelovány v čase.
2. **Lego koncepce;** základní stavební bločky se mohou kombinovat do složitějších struktur.
   - Založeno na objektově-orientované analýze.
3. **Návrh strategie řízení**
   - v každém časovém okamžiku se provádí nový návrh řídící strategie.
   - V každém časovém okamžiku je možné komunikovat s okolím a měnit komunikační strategii.
4. **Uživatelské prostředí;** použití nástroje by mělo být snadné i pro neprogramátory.
   - Podpora zadávání experimentů v GUI.

Struktura M3k

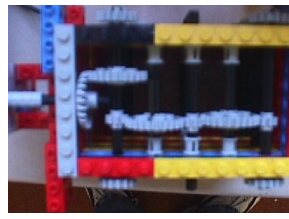
experiment

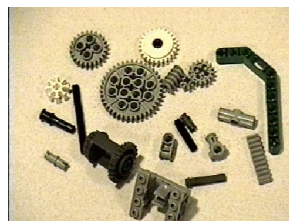
participant	environment

world model	communication	aims

likelihood (model)	posterior	Strategy	Data S.

Random Var	functions	matrix decomp.	projections





Implementační požadavky

1. Efektivní a stabilní numerické výpočty,
2. Přenosné mezi platformami,
3. Podpora OO programování,
4. Možnost využití stávajícího kódu (Matlab, ANSI C),
5. Ekonomicky dostupné,
6. Uživatelsky přívětivé.

Implementační požadavky

1. Efektivní a stabilní numerické výpočty,
2. Přenosné mezi platformami,
3. Podpora OO programování,
4. Možnost využití stávajícího kódu (Matlab, ANSI C),
5. Ekonomicky dostupné,
6. Uživatelsky přívětivé.

		+	−
1.	Matlab/Mex/ANSI C	1,2,**4**,6	3,5
2.	C++	1,2,3,5	**4**,6
3.	Java	2,3,5	1,4,6

Implementační požadavky

1. Efektivní a stabilní numerické výpočty,
2. Přenosné mezi platformami,
3. Podpora OO programování,
4. **Možnost využití stávajícího kódu (Matlab, ANSI C),**
5. Ekonomicky dostupné,
6. Uživatelsky přívětivé.

		+	−
1.	Matlab/Mex/ANSI C	1,2,**4**,6	3,5
2.	C++	1,2,3,5	**4**,6
3.	Java	2,3,5	1,4,6

Objekty v Matlabu

Imitace objektově orientovaných vlastností v Matlabu:

- Objekt je struktura s jednoznačným identifikátorem
- Metody objektů jsou generické funkce typu
  `>> Objekt = metoda(Objekt, parametry...)`
  která se automaticky převede na metodu
  `>> Objekt = trida_metoda(Objekt, parametry...)`
- Kontrola typů je ošetřena programátorem, dědičnost též.

Předpoklad:

- Udržování této struktury je menší zlo než přejít na "cizí jazyk"

První problémy

Ukazatel není Matlabem podporován

- **Řešení:** ukazatel je integer, který je indexem do globálního cellu.

copy-on-write předávání parametrů, prakticky znemožňuje inplace operace, které jsou kritické pro objektový styl programování.

- **Řešení:** inplace operace se budou dělat pomocí mex-funkcí (i když je to v dokumentaci zakázáno)
- mxCallMatlab() zajistí spolupráci mezi mexy a .m soubory.

Výsledky po 3 letech

- 637 m-, 233 mex- souborů
- Doba běhu experimentu zjobroom
  - *.m 50.26s 12x pomalejší
  - *.mex 4.14s
- Vastní nástroje:
  - generování dokumentace,
  - sledování konsistence .m a .c souborů

Výsledky po 3 letech

- 637 m-, 233 mex- souborů
- Doba běhu experimentu zjobroom
  - *.m 50.26s 12x pomalejší
  - *.mex 4.14s
- Vastní nástroje:
  - generování dokumentace,
  - sledování konsistence .m a .c souborů

!!! Z .mex souboru lze volat .m soubor, který už **nesmí** volat ten samý .mex.

  - Nemůžeme kombinovat .m a .mex implementace.
  - Kvůli rychlosti pouze .mex.

Mex file: example

```
  Facs    = (mxArray*)in[0];
  nchn    = mxGetN(Facs);
  Xychns  = mxCreateDoubleMatrix(1,nchn,0); /* Xychns allocated */
  ychns   = mxGetPr(Xychns);
  nychns   = LEN(Xychns);
  before  = mxCalloc(nchn*nchn,sizeof(int)); /* befoer allocated */
  after   = mxCalloc(nchn*nchn,sizeof(double)); /* after allocated

/* get ychns */
  for (fac=0; fac<nchn; fac++)     {
      Fac   = mxGetCell(Facs,fac);
      ychn = (int)mxGetScalar( mxGetFieldN(Fac, Fac_ychn) );
      ychns[fac] = (double)ychn;
    }

/* inspect component, set before, after */
  for (fac=0; fac<nchn; fac++)     {
      mxArray *Xstr;
      double *str;

      Fac   = mxGetCell(Facs,fac);
      ychn = (int)mxGetScalar( mxGetFieldN(Fac, Fac_ychn) );
      Xstr = mxGetFieldN(Fac, Fac_str);
      str   = mxGetPr( Xstr);
    }
```

Hledání alternativy

Matlab je v podstatě uživatelské rozhraní pro LAPACK a BLAS.

Python: scientific python, numerical python.

- interaktivní výpočetní prostředí,
- objektově orientované,
- bindings pro C a další jazyky.

C++: s nadstavbou pro lapack (lapack++, newmat)

- standardní překladače namnoha platformách,
- možnost přímého použití C kódu,
- objekty, šablony, knihovny.

IT++: knihovna inspirovaná Matlabem

IT++ vlastnosti

- Mapování BLAS na operátory (C = A*B;)
- Knihovna funkcí LAPACK (Ch = chol(A);)
- Ukládání a načítání dat do souboru `*.it`:

```
it_file itf ( "struct_test.it" );
itf << Name ( "exec_times" ) <<exec_times;
```

- Čtení v Matlabu:

```
>> itload("struct_test.it");
```

- Podpora pro použití IT++ v .mexu.
- Objekty pro zpracování signálu, statistické funkce...
- Open-source, vynikající podpora.

Lapack a BLAS v IT++ I

IT++ mapuje funkce LAPACKu a BLASu na C++ operátory ve stylu Matlabu:

```
a+b                          // a+b
a-b                          // a-b
elem_mult(a,b)               // a.*b  (elementwise product)
a*b                          // a.'*b (inner product)
conj(a)*b                    // a'*b (inner product)
outer_product(a,b)           // a*b.' (outer product)
elem_div(a,b)                // a./b
A+B;                         // A+B;
A-B;                         // A-B;
A*B;                         // A*B;
elem_mul(A,B)                // A.*B
elem_div(A,B)                // A./B
A\b;                         // ls_solve_od(A,b);
```

Lapack a BLAS v IT++ II

```
a.length()                          // length(a)
a(0)                                // a(1)
a(1)                                // a(2)
a(k-1)                              // a(k)
a(k-1)=x;  or a.set(k-1,x);         // a(k)=x;
a.left(k)                           // a(1:k)
a.right(k)                          // a(end-k+1:end)
a.mid(k,l)                          // a(k:k+l-1)
a.del(k);                           // a=[a(1:k-1); a(k+1:end)];
concat(a,b)                         // [a; b] (or [a.' b.'].')
a.clear();                          // a=zeros(size(a));  (or a=com
to_cmat(a)                          // complex(a)   (assuming a rea
```

Lapack a BLAS v IT++ III

```
zeros(n,n)                      // zeros(n,n)
zeros_c(n,n)                    // complex(zeros(n,n))
eye(n)                          // eye(n)
eye_c(n)                        // complex(eye(n))
linspace(alpha,beta,n)          // linspace(alpha,beta,n)
```

Hardcoded initializations:

```
mat X="1.1 1.2; 2.1; 2.2";      // X=[1.1 1.2; 2.1 2.2];
ivec a="1 2 3 4 5";             // a=[1; 2; 3; 4; 5]; (or a=[1
ivec a="1:-3:-8";               // a=1:-3:-8;
```

FOR cyklus pro násobení matic

```
function C=matmult(A,B)


for i=1:size(A,1)
   for j=1:size(A,2)
      sum=0;
      for k=1:size(A,2)
         sum=sum+A(i,k)*B(k,j);
      end
      C(i,j)=sum;
   end
end
```

```
mat matmul ( mat &A, mat &B ) {
 mat C ( A.rows(),B.cols() );
 int i,j,k;
 double sum;

 for (i=0; i<A.rows(); i++) {
    for (j=0; j<A.cols(); j++) {
       sum=0.0;
       for (k=0; k<A.cols(); k++) {
          sum+=A(i,k)*B(k,j);
       }
       C(i,j)=sum;
    }
 }
 return C;
}
```

- rozdíly : deklarace, číslování od 0, možnost psát do A a B

Přepis Matlabovského kódu do C++ je triviální

```
et =zeros(1,4);et2 =zeros(1,4);

idn = 1;
vn = [5,50,200,500]

for n= vn
    A=rand(n);B=rand(n);C=rand(n);

    tic;
    for i=1:10
        C=matmult(A,B);
    end
    et(idn) = toc;

    tic;
    for i=1:10
        C=A*B;
    end
    et2(idn) = toc;

    idn =idn+1;
end
```

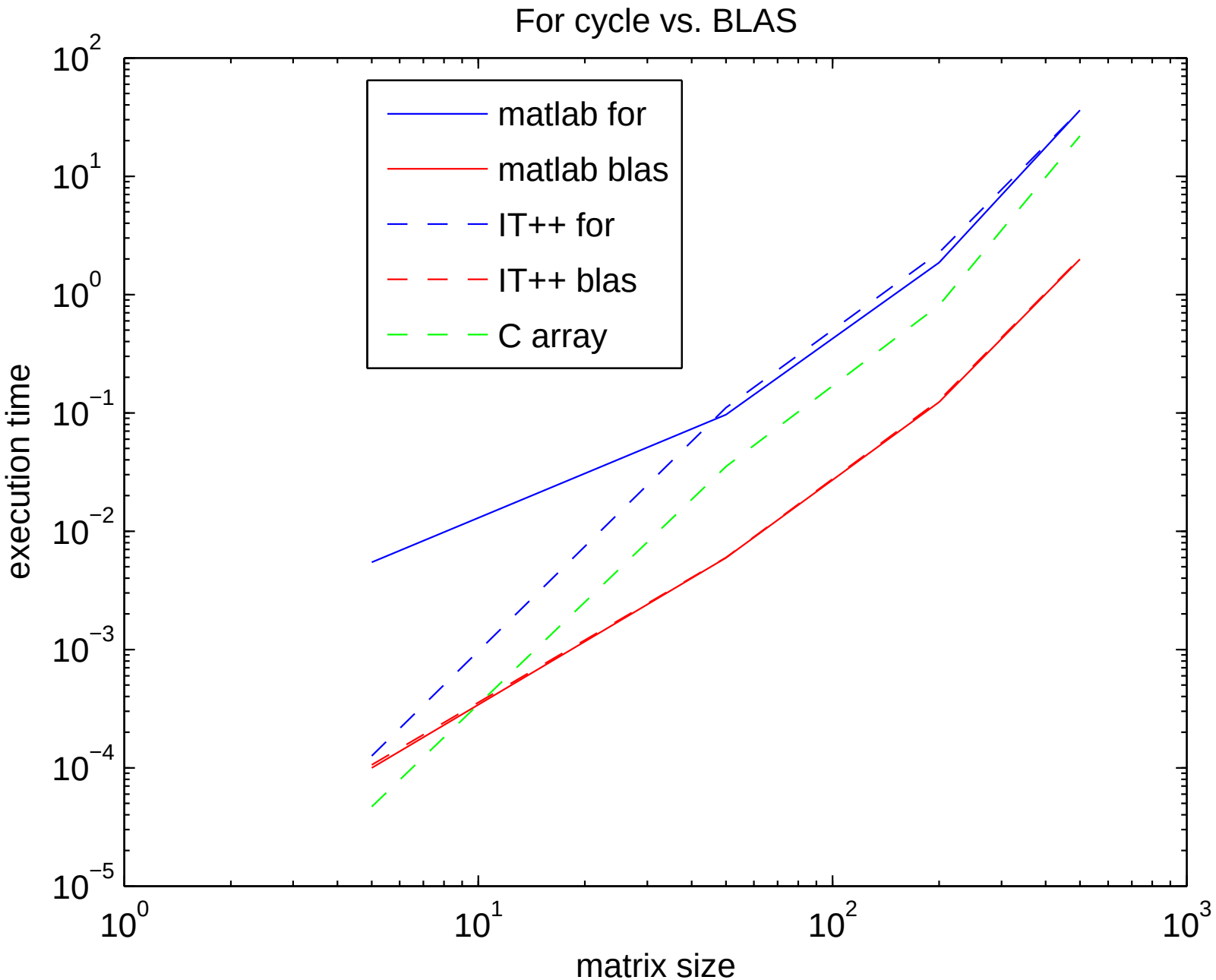
```
vec ex ( 4 );vec ex_b ( 4 );
mat A; mat B; mat C;
int n;
vec vn="5 50 200 500";
Real_Timer tt;

for ( int i=0;i<vn.length();i++ ) {
    n = vn ( i );
    A = randu(n,n);B = randu(n,n);C

    tt.tic();
    for ( int i=0;i<10;i++ ) {
       C = matmul ( A,B );
    }
    ex ( i ) =tt.toc();

    tt.tic();
    for ( int i=0;i<10;i++ ) {
        C = A*B;
    }
    ex_b ( i ) =tt.toc();
}
```

Test rychlosti: BLAS je 20x rychlejší



Test FOR cyklu pro struktury

```
et = zeros(1,3);

S=struct('A',2,'B',3);



iind=1;

vn = [1000 10000 100000];

for n= vn
   S.A = 0; tic
   for i=1:n
      S.A = S.A + 1;
   end
   et(iind) = toc;
   iind = iind+1;
end
et
```

```
vec ex =zeros( 3 );

typedef struct
{double A;double B;} St;

Real_Timer tt;
int i,j;
St S;
vec n = "1000 10000 100000";

for ( i=0;i<n.length();i++ ) {
   S.A=0; tt.tic();
   for ( j=0;j<n(i);j++ ) {
      S.A+=1;
   }
   ex ( i ) =tt.toc();
}
cout << ex <<endl;
```

Test FOR cyklu pro struktury

```
et = zeros(1,3);

S=struct('A',2,'B',3);



iind=1;

vn = [1000 10000 100000];

for n= vn
   S.A = 0; tic
   for i=1:n
      S.A = S.A + 1;
   end
   et(iind) = toc;
   iind = iind+1;
end
et
```

```
vec ex =zeros( 3 );

typedef struct
{double A;double B;} St;

Real_Timer tt;
int i,j;
St S;
vec n = "1000 10000 100000";

for ( i=0;i<n.length();i++ ) {
   S.A=0; tt.tic();
   for ( j=0;j<n(i);j++ ) {
      S.A+=1;
   }
   ex ( i ) =tt.toc();
}
cout << ex <<endl;
```

Matlab: [4e-5 3e-4 3e-3]

IT++: [4e-5 3e-4 3e-3]

Test FOR cyklu pro struktury volané funkcí

```
for n= vn
    S.A = 0; tic
    for i=1:n
        S=aa(S);
    end
    et(iind) = toc;
    iind = iind+1;
end
et

function S=aa(S)
S.A = S.A + 1;
```

```
void aa(St &S){S.A+=1;}

for ( i=0;i<n.length();i++ ) {
    S.A=0; tt.tic();
    for ( j=0;j<n(i);j++ ) {
        S.A+=1;
    }
    ex ( i ) =tt.toc();
}
cout << ex <<endl;
```

Test FOR cyklu pro struktury volané funkcí

```
for n= vn
    S.A = 0; tic
    for i=1:n
        S=aa(S);
    end
    et(iind) = toc;
    iind = iind+1;
end
et

function S=aa(S)
S.A = S.A + 1;
```

```
void aa(St &S){S.A+=1;}

for ( i=0;i<n.length();i++ ) {
    S.A=0; tt.tic();
    for ( j=0;j<n(i);j++ ) {
        S.A+=1;
    }
    ex ( i ) =tt.toc();
}
cout << ex <<endl;
```

Matlab: [3e-2 3e-1 1e0] IT++: [5e-5 4e-4 4e-3]

- Rozdíl téměř o 3 řády.
- Ve skutečnosti jen konstantní ztráta.

Test FOR cyklu s Cellem

```
C = {[2],[3]}

vn = [1000 10000 100000];
iind=1;
for n= vn
   tic
   for i=1:n
      C{1} = C{1} + 1;
   end
   et(iind) = toc;
   iind = iind+1;
end
```

```
Array<vec> S="{[2␣]␣[3␣]}␣";

vec n = "1000␣10000␣100000";

for ( i=0;i<n.length();i++ ) {
   tt.tic();
   S(1)(1)=0;
   for ( j=0;j<n(i);j++ ) S(1)+=1;
   exec_times ( i ) =tt.toc();
}
```

Test FOR cyklu s Cellem

```
C = {[2],[3]}

vn = [1000 10000 100000];
iind=1;
for n= vn
   tic
   for i=1:n
      C{1} = C{1} + 1;
   end
   et(iind) = toc;
   iind = iind+1;
end
```

```
Array<vec> S="{[2␣]␣[3␣]}␣";

vec n = "1000␣10000␣100000";

for ( i=0;i<n.length();i++ ) {
   tt.tic();
   S(1)(1)=0;
   for ( j=0;j<n(i);j++ ) S(1)+=1;
   exec_times ( i ) =tt.toc();
}
```

Matlab: [5e-2 2e-1 7e-1] IT++: [1e-4 1e-3 1e-2]

- Rozdíl necelé 2 řády...

Použití IT++ v Matlabu pomocí Mex

```
#include <itpp/itmex.h>
#include "../../bdm/estim/arx.h"
using namespace itpp;

void mexFunction ( int n_output, mxArray *output[], int n_input, c
   // Convert input variables to IT++ format
   mat Data = mxArray2mat ( input[0] );
   // ------------------- Start of routine -----------------------
   ARX Ar (eye(Data.rows(), 3.0 );

   for (int i=0;i<Data.cols();i++ ) { Ar.bayes ( Data.get_col ( i
   vec th = Ar._epdf().mean();
   // -------------------- End of routine -------------------------
   output[0] = mxCreateDoubleMatrix ( 1,th.length(), mxREAL );
   vec2mxArray ( th, output[0] );

   if ( n_output>1 ) { double ll=Ar._tll();
      output[1] = mxCreateDoubleMatrix ( 1,1, mxREAL );
      double2mxArray ( ll, output[1] );
   }
}
```

Paralelizace pomocí OpenMP

Paralelizace pomocí direktiv překladače.
FOR cyklus

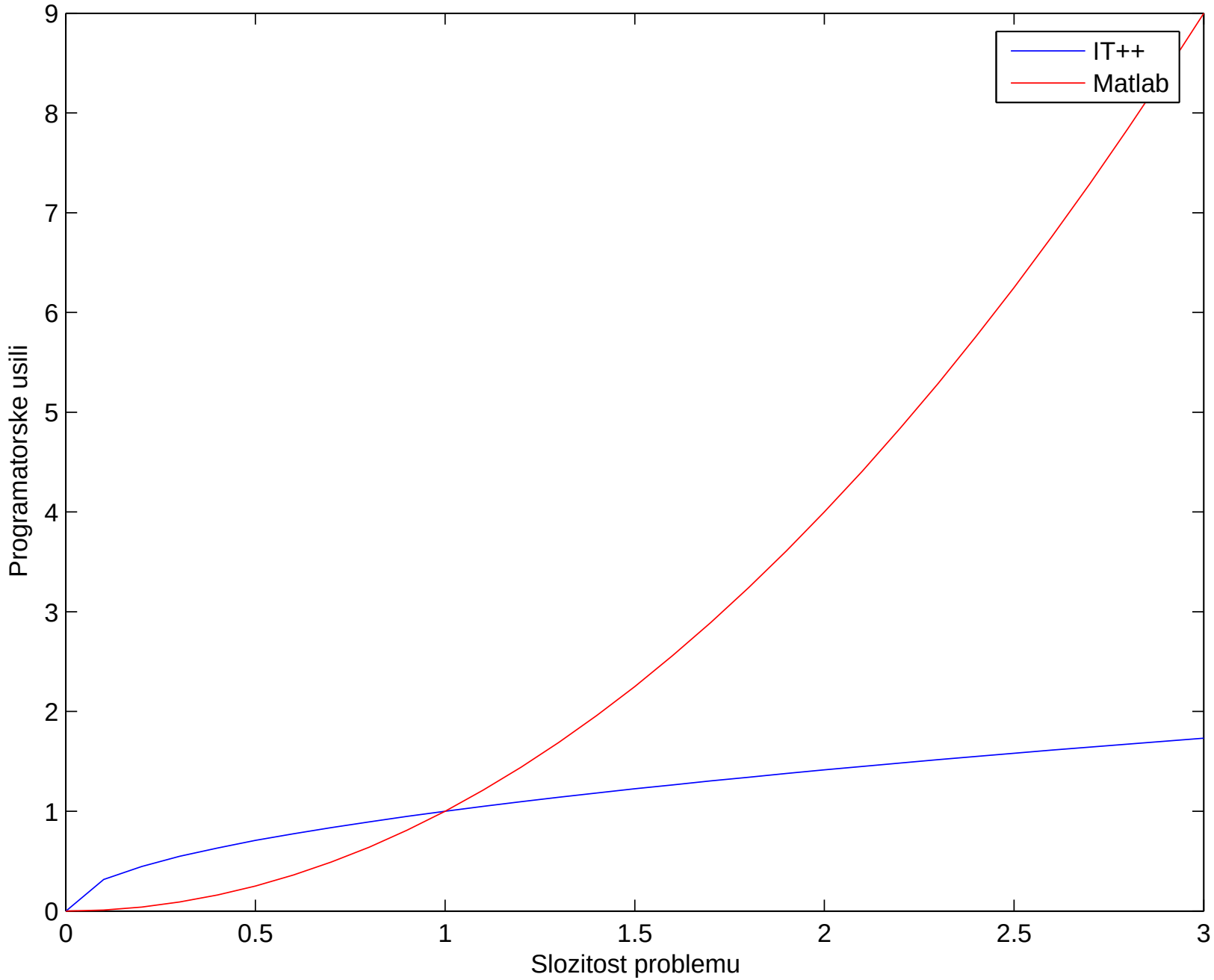
```
#pragma omp parallel for
 for ( i=0;i<n;i++ ) {
         Bms[i]->condition ( _samples ( i ) );
         Bms[i]->bayes ( dt );
         lls ( i ) = Bms[i]->_ll(); // lls above is also in propos
         if ( lls ( i ) >mlls ) mlls=lls ( i ); //find maximum lik
 }
```

Chráněné části

```
vec sample() const {
   vec smp ( rv.count() );
   #pragma omp critical
   UniRNG.sample_vector ( rv.count(),smp );
   return low+elem_mult(distance,smp);
}
```

Zrychlení na 16ti procesorovém stroji 10x-12x.

Závěr: osobní dojem



Závěr

Matlab je vhodný na:

- akademické projekty s malým množstvím vzájemně volaných funkcí,
- rychlé vyzkoušení izolovaného algoritmu,
- interaktivní práce: rychlý algoritmus, heuristické vyhodnocení.

C++ (IT++) je výhodnější pro:

- rozsáhlejší projekty s bohatší datovou strukturou,
- aplikačně orientované projekty,
- výpočetně náročné programy s omezenou interakcí

Přepis kódu Matlabu do IT++ je výrazně snazší než do mex.