НЕКИ АСПЕКТИ СИМБОЛИЧКОГ РАЧУНА – ПРИМЕНА MAPLE-А У НАСТАВИ МАТЕМАТИКЕ

Бранко Малешевић Синиша Јешић
Наташа Бабачев Ивана Јововић

Електротехнички факултет - Београд

200 година Универзитета у Београду
МАТЕМАТИКА ДАНАС, настава, примене и рачунарство

Сава центар, Београд 13.–14. IX 2008.

На Електротехничком факултету, Универзитета у Београду, студенти поред обавезних математичких предмета могу да бирају и изборне математичке предмете. Један од њих је практикум

Рачунарски алати у математици - http://pm3.etf.bg.ac.yu/

Практикум обезбеђује могућност студентима, да после стеченог основног математичког образовања кроз предмете Математика 1 и 2, слушају курс на коме ће научити како да практично користе математички софтверски пакет Maple.

Курс Рачунарски алати у математици се састоји од четири дела:

1. увод у Maple, изабрана поглавља из Алгебре;
2. изабрана поглавља из Математичке анализе 1;
3. изабрана поглавља из Математичке анализе 2;
4. остале могућности програмског пакета Maple.

Практикум има за циљ да студентима друге године (одсека ОС, ОФ, ОГ) олакша праћење градива предмета Математика 3. У оквиру предмета Математика 3 предвиђено је да студент може освојити део поена израдом домаћег задатка применом математичког софтвера. У ту сврху знање стечено на овом практикуму свакако може да буде од користи јер обезбеђује добро познавање једног математичког софтвера.

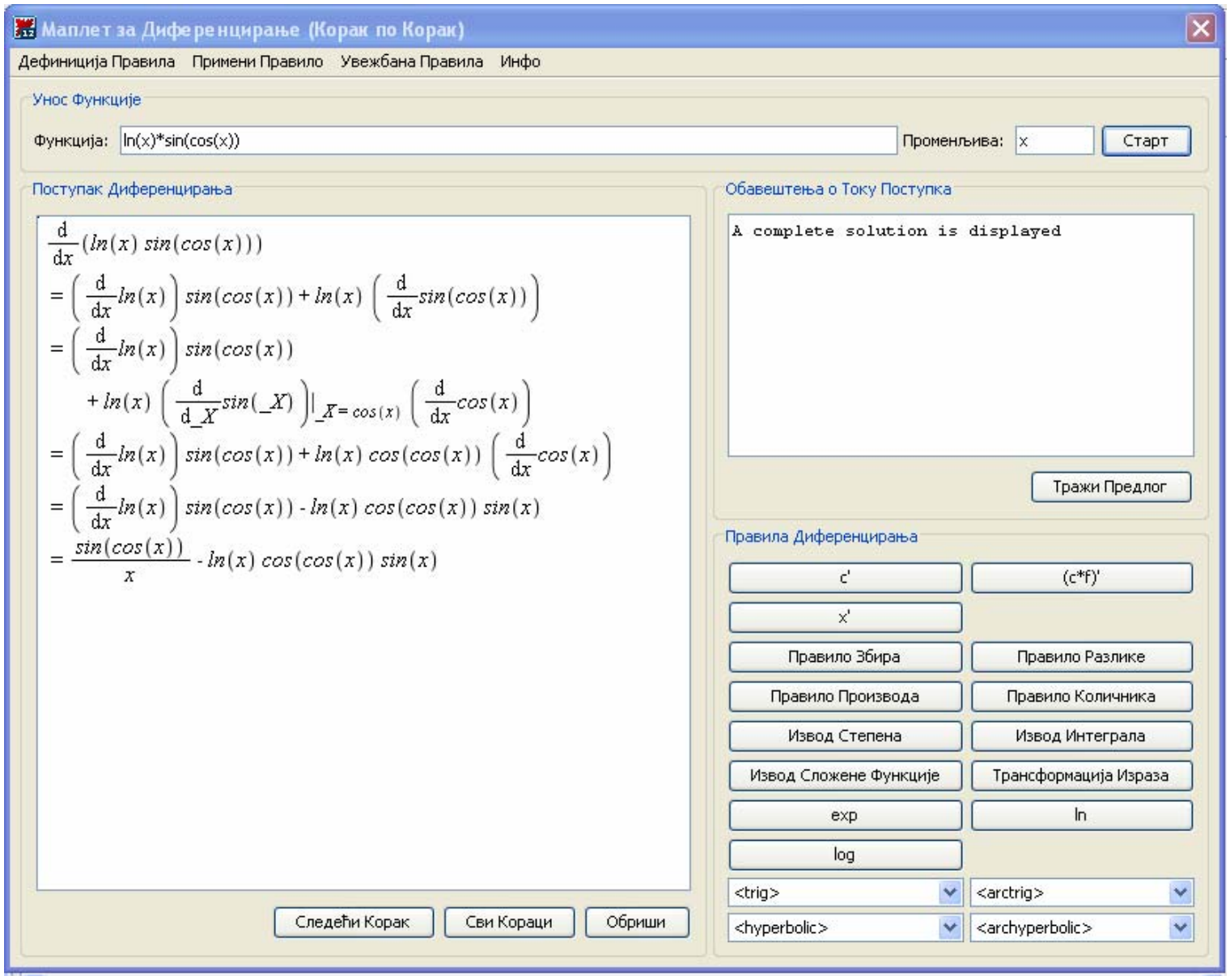
За студенте треће године одсека рачунарска техника и информатика (ИР) у курсу, поред 1. и 2. дела, од посебног интереса је и 4. део где се разматрају остале могућности програмског пакета Maple: повезивање Matlab-а са Maple-ом; као и разне Интернет, Јава и Маплет апликације.

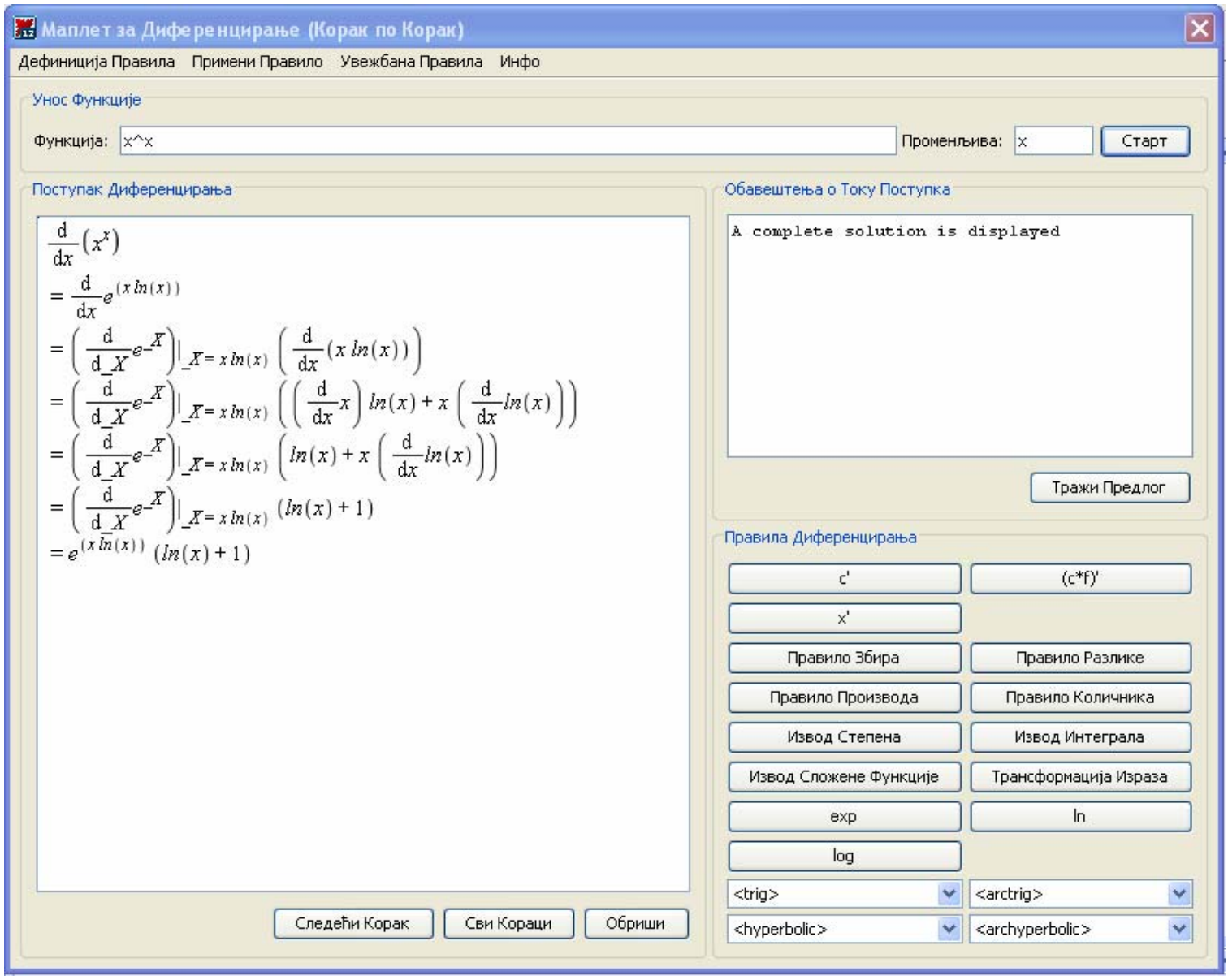
У оквиру ове презентације представићемо неке аспекте симболичког рачуна и примене Maple-а у настави математике кроз следеће теме:

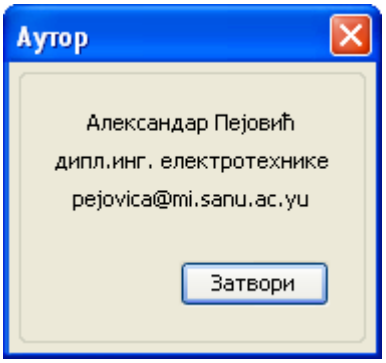
1. Маплет за диференцирање (корак по корак);
2. Maple, C и асемблер - повезивање и компарације;
3. Процедура за налажење к-тог степена матрице;
4. Примена Maple-а на комплексну анализу.

Maple програми* које приказујемо у наредном делу презентације илуструју део тема које се разматрају у овом практикуму.

* Прва два програма су студентски радови формулисани и рађени под руководством Б. Малешевића


Маплет за Диференцирање (Корак по Корак)
Дефиниција Правила
Примени Правило
Увежбана Правила
Инфо
Унос Функције
Функција:
ln(x)*sin(cos(x))
Променљива:
x
Старт
Поступак Диференцирања
Обавештења о Току Поступка
A complete solution is displayed
Тражи Предлог
Правила Диференцирања
c'
(c*f)'
x'
Правило Збира
Правило Разлике
Правило Производа
Правило Количника
Извод Степена
Извод Интеграла
Извод Сложене Функције
Трансформација Израза
exp
ln
log
<trig>
<arctrig>
<hyperbolic>
<archyperbolic>
Следећи Корак
Сви Кораци
Обриши


Маплет за Диференцирање (Корак по Корак)
Дефиниција Правила
Примени Правило
Увежбана Правила
Инфо
Унос Функције
Функција:
x^x
Променљива:
x
Старт
Поступак Диференцирања
$\frac{d}{dx}(x^x)$
$= \frac{d}{dx} e^{(x\,ln(x))}$
$= \left(\frac{d}{d\_X} e^{\_X}\right)|_{\_X = x\,ln(x)} \left(\frac{d}{dx}(x\,ln(x))\right)$
$= \left(\frac{d}{d\_X} e^{\_X}\right)|_{\_X = x\,ln(x)} \left(\left(\frac{d}{dx}x\right) ln(x) + x\left(\frac{d}{dx} ln(x)\right)\right)$
$= \left(\frac{d}{d\_X} e^{\_X}\right)|_{\_X = x\,ln(x)} \left(ln(x) + x\left(\frac{d}{dx} ln(x)\right)\right)$
$= \left(\frac{d}{d\_X} e^{\_X}\right)|_{\_X = x\,ln(x)} (ln(x) + 1)$
$= e^{(x\,ln(x))}\,(ln(x) + 1)$
Следећи Корак
Сви Кораци
Обриши
Обавештења о Току Поступка
A complete solution is displayed
Тражи Предлог
Правила Диференцирања
c'
(c*f)'
x'
Правило Збира
Правило Разлике
Правило Производа
Правило Количника
Извод Степена
Извод Интеграла
Извод Сложене Функције
Трансформација Израза
exp
ln
log
<trig>
<arctrig>
<hyperbolic>
<archyperbolic>


Аутор
Александар Пејовић
дипл.инг. електротехнике
pejovica@mi.sanu.ac.yu
Затвори

Application Demonstration

Maplesoft
www.maplesoft.com

Maple, C and Assembly Language – Performance Comparison

Milorad Pop-Tošić, Igor Skender

Department of Computer Engineering

School of Electrical Engineering, University of Belgrade, Serbia

poptosic@gmail.com, igor.skender@gmail.com

Abstract

We show how to utilize Maple external calling mechanism to speed up function execution by coding them in C and assembly language. Techniques were demonstrated on Jones' algorithm for finding the n-th prime number.

Introduction

In this article, it will be shown how to utilize Maple external calling mechanism in order to solve real problems faster, by calling external functions written in C and assembly language. Maple `define_external` function was used to call routines written in C and assembly language MASM, from DLL libraries [3,4]. These techniques were demonstrated on the example of Jones' algorithm, an algorithm for finding the n-th prime number [1, 2]. Furthermore, we made measurements and comparison of execution time for different implementations of the algorithm: assembly language, C language, and Maple procedures.

Jones' algorithm

Jones' algorithm for finding the n-th prime number states [1, 2]:

Let $P(n)$ be the n-th prime number. Formula $P(n)$, which generates the n-th prime number for given n, is given as:

$$P_n = \sum_{i=0}^{n^2} \left(1 \dot{-} \left(\left(\sum_{j=0}^{i} r\left((j \dot{-} 1)! ^2, j\right)\right) \dot{-} n\right)\right)$$

where $r(a, b)$ is a function that returns the remainder of division of number a by number b, where $r(a, 0) \equiv a$ and $\dot{-}$ is a function defined as:
*$a \dot{-} b = a - b$ **if** $a \geq b$ **else** $a \dot{-} b = 0$*

Function P returns the array of prime numbers $P(1)=2, P(2)=3, P(3)=5,...$
There is a function in Maple, **ithprime(n)**, which returns values of P from a database. Our goal is to illustrate what can be achieved by connecting Maple to C and assembly language, on the example of Jones' algorithm for finding the function P.

The code for the procedure in Maple for finding P function using Jones' algorithm is given below:

```
> restart:
> M := proc (x::integer, y::integer)
>   if x<y then 0;
>   else x-y;
>   fi;
> end:
> JonesM := proc (n::integer)
>   local m,s,i,p,j,f,k;
>   m := n*n;
>   s := 1;
>   for i from 1 to m do
>           p:=0;
>           for j from 1 to i do
>                   f := 1;
>                   for k from 1 to (j-1) do
>                     f := f*k*k;
>                     f := f mod j;
>                   od;
>                   p := p+f;
>           od;
>           s := s+ M(1, M(p,n));
>   od;
> RETURN (s);
> end:
```

The code in C for finding P function using Jones' algorithm is given below:

Jones.c

```
#define M(x,y) ((x)>(y) ? ((x)-(y)):(0))

int i,j,k,n,m,p,f,s;

int Jones(int n) {
             for ( m=n*n, s=i=1; i<=m; i++) {
                                  for ( p=0, j=1; j<=i; j++) {
                                     for ( f=k=1; k<j; k++) { f=f*k*k; f%=j; }
```

```
                                    p+=f;
                                }
                                s+=M(1, M(p,n));
            }
            return s;
}
```

In order to call this code from within Maple it should, first, be compiled into a DLL (Dynamic Linking Library). A compiler from Microsoft Visual Studio .NET was chosen at this place. We compile the code by issuing the following command:

```
> cl.exe -LD -Gy -Gz JonesC.c -link /export:Jones
```

It is essential to include keyword /export: followed by the name of the function to be exported and used from within Maple. Note that any other C compiler can be used here as long as it produces a DLL with stdcall calling convention, and exports symbol Jones.

From this point onwards, compiled DLL library JonesC.dll, is connected to Maple by using **`define_external`** function, as follows:

```
> JonesC:=define_external(
>   'Jones',
>   'n'::integer[4],
>   'RETURN'::integer[4],
>   'LIB'="./JonesC.dll"
>   ):
```

It should be noted that Maple can also call functions from UNIX .so libraries, which have similar function as DLL libraries in Windows.

From this point onwards, the call **`JonesC()`** from within Maple appears to be exactly the same as a call to a built-in Maple function, although the function Jones from the DLL library gets called.

The procedure in assembly language for finding *P* function using Jones' algorithm is given below [7]:

JonesASM.asm

```
.586
.model flat, stdcall

.code
LibMain proc h:DWORD, r:DWORD, u:DWORD
        mov eax, 1
        ret
LibMain Endp
                              ; s=ax   j=bx   k=cx  p=si  i=di
Jones proc n:DWORD
    LOCAL m:DWORD
    LOCAL s:DWORD
    push ebx
    push ecx
    push edx
    mov eax, n
```

```
    mul eax
    mov m, eax
    mov eax, 1
    mov s,  eax
    mov edi, eax
loop1:
    cmp edi, m
    jg  l0
    mov ebx, 1
    xor esi, esi
loop2:
    cmp ebx, edi
    jg  l1
    mov eax, 1
    mov ecx, eax
loop3:
    cmp ecx, ebx
    jge l2
    mul ecx
    mul ecx
    div ebx
    mov eax, edx
    inc ecx
    jmp loop3
l2:
      add esi, eax
      inc ebx
      jmp loop2
l1:
  cmp esi, n
  jg  skip
  inc s
skip:
    inc edi
    jmp loop1
l0:
    pop edx
    pop ecx
    pop ebx

    mov eax, s
ret

Jones endp

End LibMain
```

Apart form this file, one more is needed. It lists functions, which are to be exported from the DLL, which is, in this case, the function Jones.

JonesASM.def

```
LIBRARY JonesASM
EXPORTS Jones
```

Compiling and linking the DLL library, using MASM is done by issuing:

```
  > \masm32\bin\ml /c /coff JonesASM.asm
  > \masm32\bin\Link /SUBSYSTEM:WINDOWS /DLL /DEF:JonesASM.def JonesASM.obj
```

Generated library JonesASM.dll is connected to Maple, as follows:

```
> JonesASM:=define_external(
    'Jones',
    'n'::integer[4],
    'RETURN'::integer[4],
    'LIB'="./JonesASM.dll"
    ):
```

The three shown implementations of function Jones are called from within Maple by issuing the following commands, respectively:

```
> n := 30;
```

$$n := 30 \tag{3.1}$$

```
> JonesM(n);          # Call to Maple procedure
> JonesC(n);          # Call to function in C DLL
> JonesASM(n);        # Call to function in ASM DLL
```

$$113$$
$$113$$
$$113 \tag{3.2}$$

Conclusion

Measurements and comparison of execution time for Jones' algorithm were made for all three presented implementations. To accomplish that, we used Maple **time()** function which returns total processor time used for executing expression. We used this function to calculate execution time for the three solutions, for $n \in \{1..50\}$, by issuing the following commands:

```
> time(JonesM(n));    # Maple procedure execution
```

$$115.874 \tag{4.1}$$

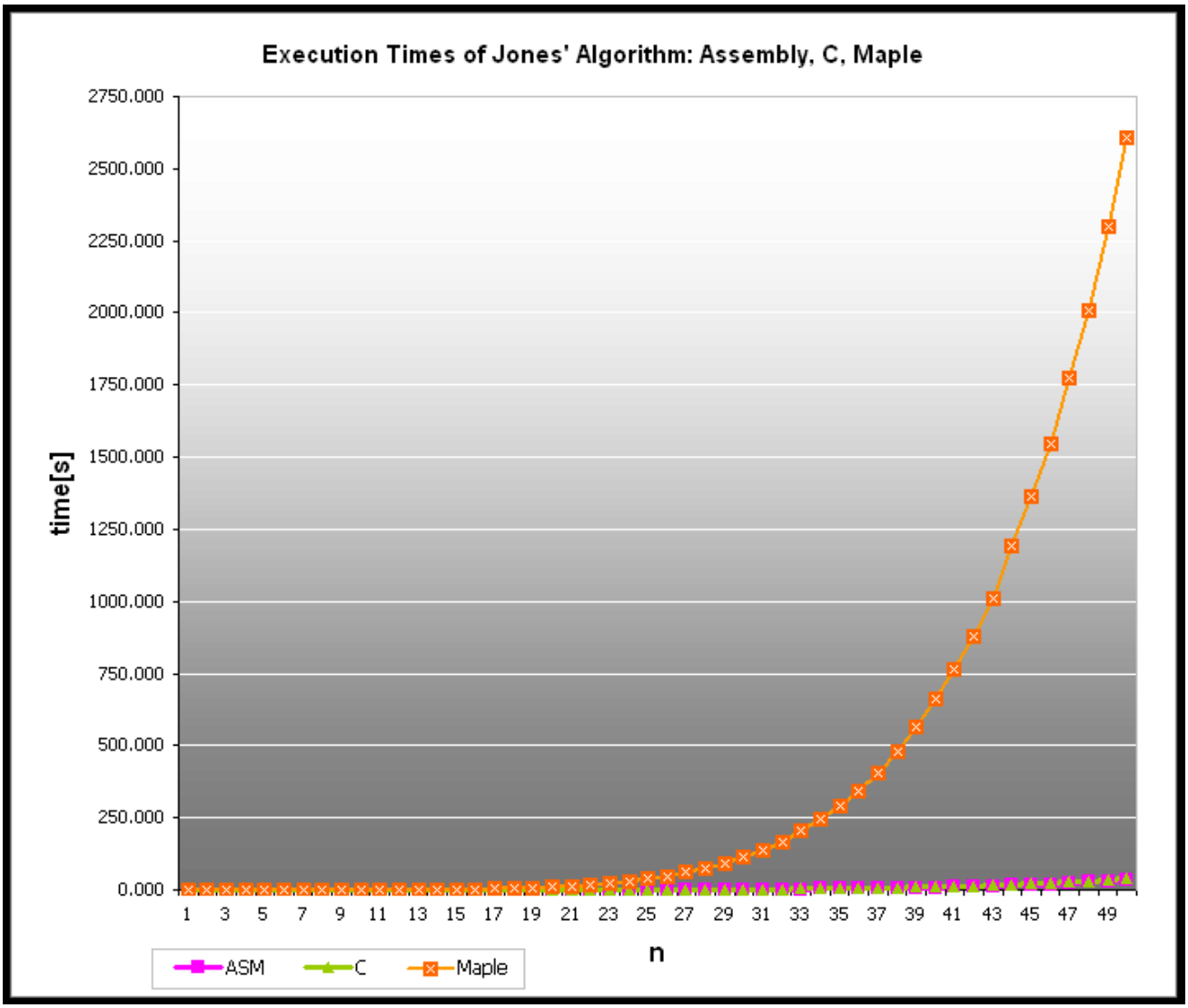
```
> time(JonesC(n));    # C function execution time
```

$$1.907 \tag{4.2}$$

```
> time(JonesASM(n));  # ASM function execution time
```

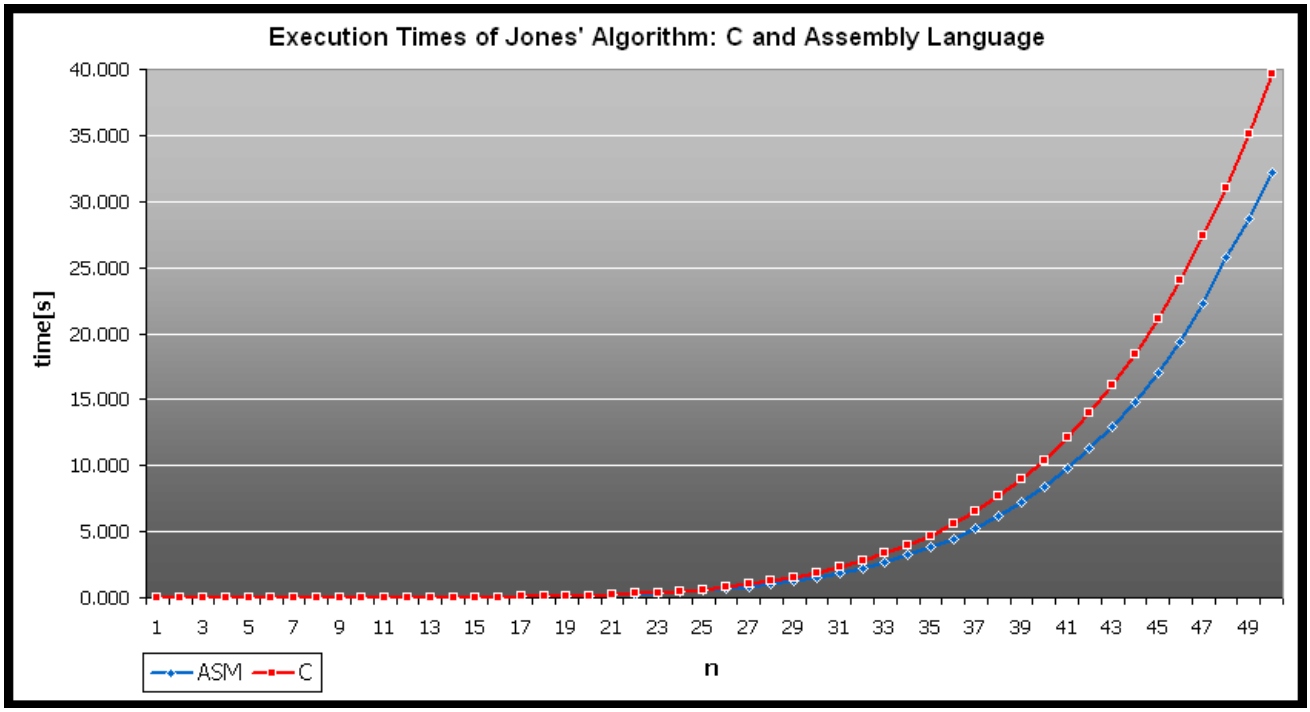
$$1.514 \tag{4.3}$$

Based on measured values, the chart that shows execution times for three presented implementations was created.



It can be observed from this chart that C and assembly language solutions have considerably better performance, compared to Maple procedures. For $n = 50$ procedure in Maple completes in about half an hour, while the same result, by applying C and assembly language solutions, is computed in the matter of seconds. The function that describes the time of execution of Maple procedures rises more sharply, so the differences are even more stressed for larger n.
From what is said can be concluded that Maple procedures are rather slow solution for problems which contain large number of iterations, primarily because Maple code is interpreted, and not compiled. In such cases, it is much more efficient to program in C, or even assembly language.

The chart that follows shows the comparison of execution time for procedures written in assembly language and C. We can observe performance advantage of assembly language over C, which becomes more stressed, as n gets larger. For instance, assembly language implementation is about 25 % faster for $n = 50$. For that reason, putting in more effort in producing assembly language code, especially for loops repeating billion times or more. For loops repeating couple of million times, there is a minor difference between assembly language and C in terms of execution time, so it is simpler to write such a function in C.



Acknowledgement

This article is based on a semester work in course Practicum of Computer Tools in Mathematics, which is taught in the 5th semester of Computer Engineering program on the Faculty of Electrical Engineering, University of Belgrade. We wish to thank assistant professor Dr Branko Malešević for acquainting us with this topic.

References

[1] James P. Jones: Formula for the nth prime number,
Canadian Mathematical Bulletin 18, (1975), pp. 433--434.

[2] James P. Jones; Daihachiro Sato; Hideo Wada; Douglas Wiens:
Diophantine Representation of the Set of Prime Numbers,
The American Mathematical Monthly Vol. 83, No. 6 (Jun., 1976), pp. 449-464

[3] Aleksandrs Mihailovs: Writing DLL in Assembly Language for External Calling in Maple - Technical Report,
Department of Mathematics, Tennessee Technological University
TR No. 2004-5, July 2004, http://www.math.tntech.edu/techreports/TR_2004_5.pdf

[4] Aleksandrs Mihailovs: Writing DLL in Assembly Language for External Calling in Maple,
http://www.maplesoft.com/applications/app_center_view.aspx?AID=1295&CID=9&SCID=63

[5] Michael Monagan: Programming in Maple: The Basics,
Institut für Wissenschaftliches Rechnen ETH-Zentrum, CH-8092 Zürich, Switzerland

[6] David A. Patterson, John L. Henessy: Computer Organization and Design: The Hardware/Software Interface,

Morgan Kaufmann; 3rd edition

[7] Branko Malešević: Examples for the special course – Algorithms in C, (according to the MSc course of Department of Algebra and Mathematical Logic, Faculty of Mathematics, Belgrade 1995).

[8] Veljko Milutinović: The Best Method for Presentation of Research Results,
Department of Computer Engineering, School of Electrical Engineering, University of Belgrade

A procedure for finding the k-th power of a matrix

Branko Malesevic
Faculty of Electrical Engineering
University of Belgrade, Serbia
malesh@EUnet.yu

Ivana Jovovic
Faculty of Electrical Engineering
University of Belgrade, Serbia
ivana121@EUnet.yu

Introduction

This worksheet demonstrates the use of Maple in Linear Algebra.
We give a new procedure (**PowerMatrix**) in Maple for finding the k-th power of n-by-n square matrix A, in a symbolic form, for any positive integer k, k>=n. The algorithm is based on an application of Cayley-Hamilton theorem. We used the fact that the entries of the matrix A^k satisfy the same recurrence relation which is determined by the characteristic polynomial of the matrix A (see [1]). The order of these recurrences is n-d, where d is the lowest degree of the characteristic polynomial of the matrix A.
For non-singular matrices the procedure can be extended for k not only a positive integer.

Initialization

```
> restart:
  with(LinearAlgebra):
```

Procedure Definition

PowerMatrix

Input data are a square matrix **A** and a parameter **k**. Elements of the matrix **A** can be numbers and/or parameters. The parameter **k** can take numeric value or be a symbol. The output data is the k-th power of the matrix. The procedure PowerMatrix is as powerful as the procedure **rsolve**.

```
> PowerMatrix:=proc(A::Matrix,k)
  local i,j,m,r,q,n,d,f,P,F,C;
  P:=x->CharacteristicPolynomial(A,x);
  n:=degree(P(x),x);
```

```
d:=ldegree(P(x),x);
F:=(i,j)->rsolve({sum(coeff(P(x),x,m)*f(m+q),m=0..n)=0,seq
(f(r)=(A^r)[i,j],r=d+1..n)},f); C:=q->Matrix(n,n,F);
if type(k,integer) then return(simplify(A^k)) elif
(Determinant(A)=0 and not type(k,numeric)) then printf
("The %a-th power of the matrix for %a>=%d:",k,k,n) elif
(Determinant(A)=0 and type(k,numeric)) then return
(simplify(A^k)) fi; return(simplify(subs(q=k,C(q))));
end:
```

Examples

Example 1.

```
> A:=Matrix([[4,-2,2],[-5,7,-5],[-6,6,-4]]);
```

$$A := \begin{bmatrix} 4 & -2 & 2 \\ -5 & 7 & -5 \\ -6 & 6 & -4 \end{bmatrix} \tag{3.1.1}$$

```
> PowerMatrix(A,k);
```

$$\begin{bmatrix} 2\,3^k - 2^k & -2\,3^k + 2^{1+k} & 2\,3^k - 2^{1+k} \\ 5\,2^k - 5\,3^k & -4\,2^k + 5\,3^k & 5\,2^k - 5\,3^k \\ -6\,3^k + 6\,2^k & 6\,3^k - 6\,2^k & -6\,3^k + 7\,2^k \end{bmatrix} \tag{3.1.2}$$

```
> Determinant(A);
```

$$12 \tag{3.1.3}$$

```
> B:=A^(-1);
```

$$B := \begin{bmatrix} \frac{1}{6} & \frac{1}{3} & -\frac{1}{3} \\ \frac{5}{6} & -\frac{1}{3} & \frac{5}{6} \\ 1 & -1 & \frac{3}{2} \end{bmatrix} \tag{3.1.4}$$

```
> PowerMatrix(B,k);
```

$$\begin{bmatrix} -2^{-k} + 2\,3^{-k} & -2\,3^{-k} + 2^{1-k} & 2\,3^{-k} - 2^{1-k} \\ 5\,2^{-k} - 5\,3^{-k} & 5\,3^{-k} - 4\,2^{-k} & 5\,2^{-k} - 5\,3^{-k} \\ 6\,2^{-k} - 6\,3^{-k} & 6\,3^{-k} - 6\,2^{-k} & -6\,3^{-k} + 7\,2^{-k} \end{bmatrix} \tag{3.1.5}$$

Example 2.

```
> A:=Matrix([[1-p,p],[p,1-p]]);
```

$$A := \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \tag{3.2.1}$$

```
> PowerMatrix(A,k);
```

$$\begin{bmatrix} \frac{1}{2}+\frac{1}{2}(1-2p)^k & -\frac{1}{2}(1-2p)^k+\frac{1}{2} \\ -\frac{1}{2}(1-2p)^k+\frac{1}{2} & \frac{1}{2}+\frac{1}{2}(1-2p)^k \end{bmatrix} \tag{3.2.2}$$

The example is from [4], page 272, exercise 19.

Example 3.

```
> A:=Matrix([[a,b,c],[d,e,f],[g,h,i]]);
```

$$A := \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \tag{3.3.1}$$

```
> PowerMatrix(A,k)[1,1];
```

$$\sum_{\_R=RootOf\left(-1+(gbf+hdc-gce-hfa-idb+iea)\_Z^3+(gc+hf-ie-ia+db-ea)\_Z^2+(i+e+a)\_Z\right)} \left(\left(1-\_R\,i-\_R\,e-\_R^2\,hf+\_R^2\,ie\right)\left(\frac{1}{\_R}\right)^k\right) \Big/ \left(\left(3\,\_R^2\,hdc-3\,\_R^2\,egc+3\,\_R^2\,gbf+3\,\_R^2\,iea-3\,\_R^2\,hfa+i+e+a+2\,\_R\,gc+2\,\_R\,hf-2\,\_R\,ie-2\,\_R\,ia+2\,\_R\,db-2\,\_R\,ea-3\,\_R^2\,idb\right)\_R\right) \tag{3.3.2}$$

Warning! In this example **MatrixPower** and **MatrixFunction** procedures cannot be done in real-time.

```
> # MatrixPower(A,k)[1,1];
> # MatrixFunction(A,v^k,v)[1,1];
```

Example 4.

```
> A:=Matrix([[0,0,1,0,1],[1,0,0,0,1],[0,0,0,1,1],[0,1,0,0,1],
  [1,1,1,1,0]]);
```

$$A := \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (3.4.1)$$

```
> PowerMatrix(A,k)[1,5];
```

$$\frac{1}{17}\sqrt{17}\left(\left(\frac{1}{2}+\frac{1}{2}\sqrt{17}\right)^k-\left(\frac{1}{2}-\frac{1}{2}\sqrt{17}\right)^k\right) \quad (3.4.2)$$

Replace '**:**' with '**;**' and see result!

```
> MatrixPower(A,m)[1,5]:
> simplify(MatrixPower(A,m)[1,5]):
> assume(m::integer): simplify(MatrixPower(A,m)[1,5]):
```

The example is from [3], page 101.

Example 5. and Example 6.

Pay attention what happens for singular matrices.

Example 5.

```
> A:=Matrix([[0,2,1,3],[0,0,-2,4],[0,0,0,5],[0,0,0,0]]);
```

$$A := \begin{bmatrix} 0 & 2 & 1 & 3 \\ 0 & 0 & -2 & 4 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.5.1.1)$$

```
> PowerMatrix(A,2);
```

$$\begin{bmatrix} 0 & 0 & -4 & 13 \\ 0 & 0 & 0 & -10 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.5.1.2)$$

```
> PowerMatrix(A,3);
```

(3.5.1.3)

$$\begin{bmatrix} 0 & 0 & 0 & -20 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.5.1.3)$$

```
> PowerMatrix(A,k);
```

The k-th power of the matrix for k>=4:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.5.1.4)$$

```
> MatrixPower(A,k);
```

$$\mathit{LinearAlgebra:-LA\_Main:-MatrixPower}\left(\begin{bmatrix} 0 & 2 & 1 & 3 \\ 0 & 0 & -2 & 4 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 \end{bmatrix}, k, \mathit{outputoptions}\right) \quad (3.5.1.5)$$

$$= [\,]$$

```
> MatrixFunction(A,v^k,v);
```

```
Error, (in LinearAlgebra:-LA_Main:-MatrixFunction) could not compute
finite interpolating value by evaluation of v^k*k/v at eigenvalue 0
which has multiplicity greater than one in the minimal polynomial
```

The example is from [2], page 151, exercise 23.

Example 6.

```
> A:=Matrix([[1,1,1,0],[1,1,1,-1],[0,0,-1,1],[0,0,1,-1]]);
```

$$A := \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (3.5.2.1)$$

```
> PowerMatrix(A,k);
```

The k-th power of the matrix for k>=4:

$$\begin{bmatrix} 2^{-1+k} & 2^{-1+k} & \frac{1}{16}2^k\left((-1)^{1+k}+5\right) & \frac{1}{16}2^k\left((-1)^k-1\right) \\ 2^{-1+k} & 2^{-1+k} & \frac{5}{16}2^k\left((-1)^{1+k}+1\right) & \frac{1}{16}2^k\left(-1+5\,(-1)^k\right) \\ 0 & 0 & (-1)^k 2^{-1+k} & (-1)^{1+k}2^{-1+k} \\ 0 & 0 & (-1)^{1+k}2^{-1+k} & (-1)^k 2^{-1+k} \end{bmatrix} \quad \textbf{(3.5.2.2)}$$

```
> MatrixPower(A,k);
```

$$\textit{LinearAlgebra:-LA\_Main:-MatrixPower}\left(\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}, k, \textit{outputoptions}\right) \quad \textbf{(3.5.2.3)}$$

$$= [\,]$$

```
> MatrixFunction(A,v^k,v);
Error, (in LinearAlgebra:-LA_Main:-MatrixFunction) could not compute
finite interpolating value by evaluation of v^k*k/v at eigenvalue 0
which has multiplicity greater than one in the minimal polynomial
```

References

[1] Branko Malesevic. *Some combinatorial aspects of the composition of a set of functions.* NSJOM., 2006 (36), 3-9.
http://www.im.ns.ac.yu/NSJOM/Papers/36_1/NSJOM_36_1_003_009.pdf or http://arxiv.org/abs/math.CO/0409287 .
[2] John B. Johnston, G. Baley Price, Fred S. Van Vleck. *Linear Equations and Matrices.* Addison-Wesley, 1966.
[3] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra* . SIAM, 2001.
[4] Robert Messer. *Linear Algebra Gateway to Mathematics.* New York: Harper-Collins College Publisher, 1993.

Conclusions

This procedure has an educational character. It is an interesting demonstration for finding the k-th power of a matrix in a symbolic form. Sometimes, it gives solutions in the better form than the existing procedure **MatrixPower** (see example 4.). See also example 5. and example 6., where we consider singular matrices. In these cases the procedure **MatrixPower** does not give a solution. The procedure **PowerMatrix** calculates the k-th power of any singular matrix. In some examples it is possible to get a solution in the better form with using the procedure **allvalues** (see example 3.).

Acknowledgment: Research partially supported by MNTRS, Serbia, Grant No. 144020.

Thank you for evaluating this Maple application sample www.maplesoft.com

Комплексне функције

Диференцијабилност комплексних функција

Испитати да ли постоји регуларна функција *f(z)=u(x,y)+iv(x,y)* чији је имагинарни део *v(x,y)=e^x(xsiny+ycosy)* и ако постоји одредити је у затвореном облику.
Решење: Да би постојала тражена функција она мора бити хармонијска:

```
> restart;
```

```
> v(x,y):=(exp(1)^x)*(x*sin(y)+y*cos(y));
```

$$v(x,y) := (\mathrm{e})^{x}\,(x\sin(y)+y\cos(y))$$

```
> a:=diff(v(x,y),x);
```

$$a := (\mathrm{e})^{x}\,(x\sin(y)+y\cos(y)) + (\mathrm{e})^{x}\sin(y)$$

```
> A:=diff(v(x,y),x,x);
```

$$A := (\mathrm{e})^{x}\,(x\sin(y)+y\cos(y)) + 2\,(\mathrm{e})^{x}\sin(y)$$

```
> b:=diff(v(x,y),y);
```

$$b := (\mathrm{e})^{x}\,(x\cos(y)+\cos(y)-y\sin(y))$$

```
> B:=diff(v(x,y),y,y);
```

$$B := (\mathrm{e})^{x}\,(-x\sin(y)-2\sin(y)-y\cos(y))$$

Ако важи *A+B=0* то значи да постоји регуларна функција са задатим имагинарним делом.

```
> Diff(u(x,y),x)=Diff(v(x,y),y);#CR-uslov
```

$$\frac{\partial}{\partial x}u(x,y) = \frac{\partial}{\partial y}\left((\mathrm{e})^{x}\,(x\sin(y)+y\cos(y))\right)$$

```
> u(x,y):=Int(Diff(v(x,y),y),x);
```

$$u(x,y) := \int \frac{\partial}{\partial y}\left((\mathrm{e})^{x}\,(x\sin(y)+y\cos(y))\right)\,dx$$

```
> u(x,y):=int(diff(v(x,y),y),x)+F(y);
```

$$u(x,y) := -(e)^{x}(y\sin(y) - x\cos(y)) + F(y)$$

```
> Diff(u(x,y),y)=-Diff(v(x,y),x);#CR-uslov
```

$$\frac{\partial}{\partial y}\left(-(e)^{x}(y\sin(y) - x\cos(y)) + F(y)\right) = -\left(\frac{\partial}{\partial x}\left((e)^{x}(x\sin(y) + y\cos(y))\right)\right)$$

```
> solve(diff(u(x,y),y)+diff(v(x,y),x)=0,diff(F(y),y));
```

$$0$$

То значи да је *F(y)=C=const*.

```
> u(x,y):=int(diff(v(x,y),y),x)+C;
```

$$u(x,y) := -(e)^{x}(y\sin(y) - x\cos(y)) + C$$

```
> f:=u(x,y)+I*v(x,y);#dobijena je tražena funkcija u
  razdvojenom obliku
```

$$f := -(e)^{x}(y\sin(y) - x\cos(y)) + C + I(e)^{x}(x\sin(y) + y\cos(y))$$

```
> F:=(x,y)->(-y*sin(y)+x*cos(y))*(exp(1))^x+C+I*(exp
  (1))^x*(x*sin(y)+y*cos(y));
```

$$F := (x,y) \to (-y\sin(y) + x\cos(y))(e)^{x} + C + I(e)^{x}(x\sin(y) + y\cos(y))$$

```
> F(z,0);
```

$$z(e)^{z} + C$$

Комплексна интеграција

Помоћу остатка израчунати вредност интеграла по контури |z-1|=2:

```
> restart;
> Int(1/(z^4+4),z=C..``);
```

$$\int_{C} \frac{1}{z^4+4}\,dz$$

```
> f := z -> 1/(z^4 + 4):
  `f(z) ` = f(z);
```

$$f(z) = \frac{1}{z^4+4}$$

Нађимо сада сингуларитете од f(z):

```
> Zn := sort([solve(denom(f(z))=0, z)]):
  `За функцију f(z) ` = f(z);
```

```
`сингуларитети су:`;
z1 := subs(z=Zn[1],z): z[1] = z1;
z2 := subs(z=Zn[2],z): z[2] = z2;
z3 := subs(z=Zn[3],z): z[3] = z3;
z4 := subs(z=Zn[4],z): z[4] = z4;
```

$$\textit{За функцију } f(z) = \frac{1}{z^4+4}$$

сингуларитети су:

$$z_1 = 1 + \mathrm{I}$$

$$z_2 = 1 - \mathrm{I}$$

$$z_3 = -1 + \mathrm{I}$$

$$z_4 = -1 - \mathrm{I}$$

Треба сада пронаћи који су сингуларитети унутар круга |z-1|<2:

```
> print(abs(z[1]-1),`< 2 `,abs(z1-1)<2, evalb(evalf
  (abs(z1-1))<2));
  print(abs(z[2]-1),`< 2 `,abs(z2-1)<2, evalb(evalf
  (abs(z2-1))<2));
  print(abs(z[3]-1),`< 2 `,abs(z3-1)<2, evalb(evalf
  (abs(z3-1))<2));
  print(abs(z[4]-1),`< 2 `,abs(z4-1)<2, evalb(evalf
  (abs(z4-1))<2));
```

$$|z_1 - 1|, \ < 2 \ , 1 < 2, \ true$$

$$|z_2 - 1|, \ < 2 \ , 1 < 2, \ true$$

$$|z_3 - 1|, \ < 2 \ , \sqrt{5} < 2, false$$

$$|z_4 - 1|, \ < 2 \ , \sqrt{5} < 2, false$$

Израчунајмо ресидууме у тачкама z2 и z4

```
> r1 := residue(f(z), z=z2): `Res[f`,z1,`] ` = r1;
  r2 := residue(f(z), z=z4): `Res[f`,z2,`] ` = r2;
```

$$Res[f, 1 + \mathrm{I},] = -\frac{1}{16} + \frac{1}{16}\,\mathrm{I}$$

$$Res[f, 1 - \mathrm{I},] = \frac{1}{16} + \frac{1}{16}\,\mathrm{I}$$

Вредност интеграла је

```
> val := 2*Pi*(r1 + r2):
  Int(f(z),z=C..``) = val;
```

$$\int_C e^z \, dz = 2\,\pi\,(r1 + r2)$$

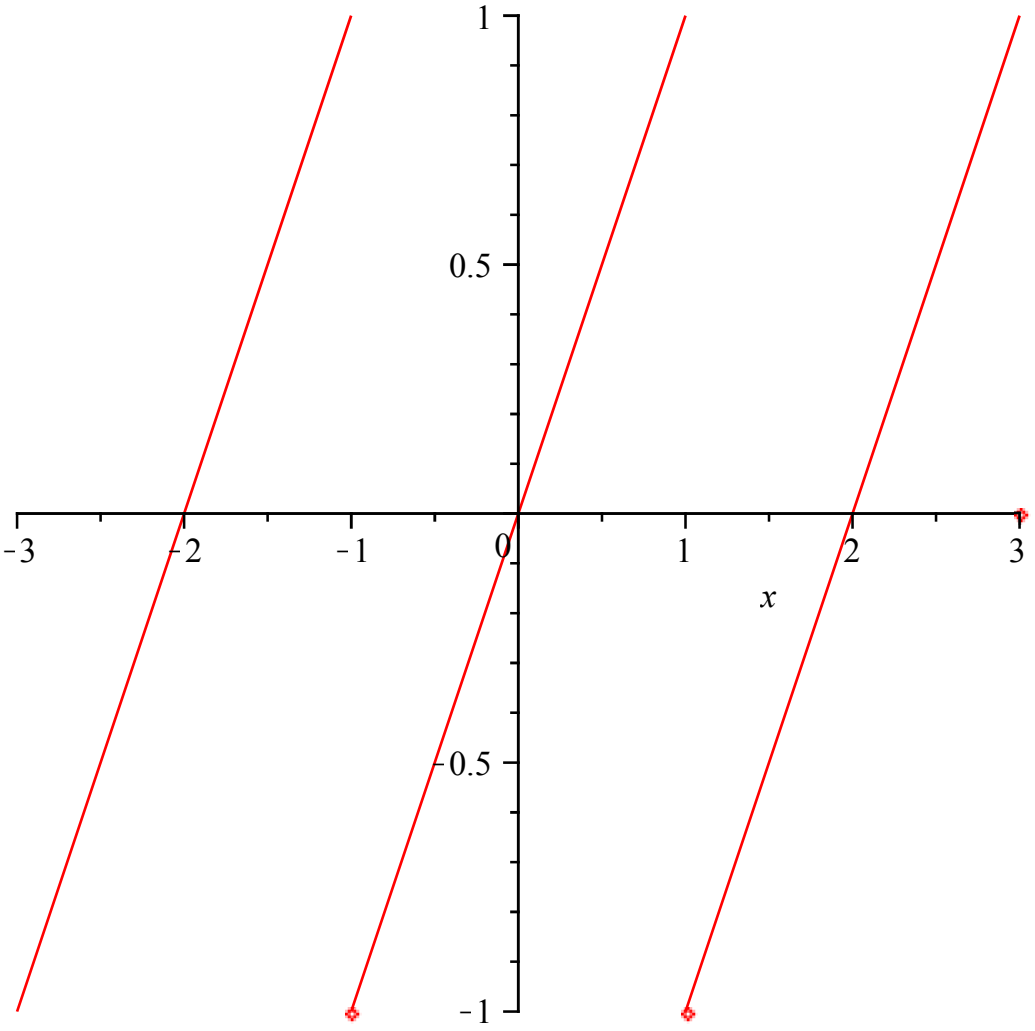
Развој функције у Fourier-ов ред

```
> restart;
```

```
> f:=piecewise(x<-1,x+2,x<1,x,x<3,x-2);
```

$$f := \begin{cases} x+2 & x < -1 \\ x & x < 1 \\ x-2 & x < 3 \end{cases}$$

```
> plot(f,x=-3..3,discont=true);
```



Пошто је функција непарна рачунамо само коефицијент

b_n

b_n

```
> bn:=Int(x*sin(n*Pi*x),x=-1..1);
```

$$bn := \int_{-1}^{1} x \sin(n \pi x) \, dx$$

```
> bn:=int(x*sin(n*Pi*x),x=-1..1);
```

$$bn := -\frac{2\,(-\sin(n\pi) + \cos(n\pi)\, n\pi)}{n^2 \pi^2}$$

Првих 8 b_n коефицијената:

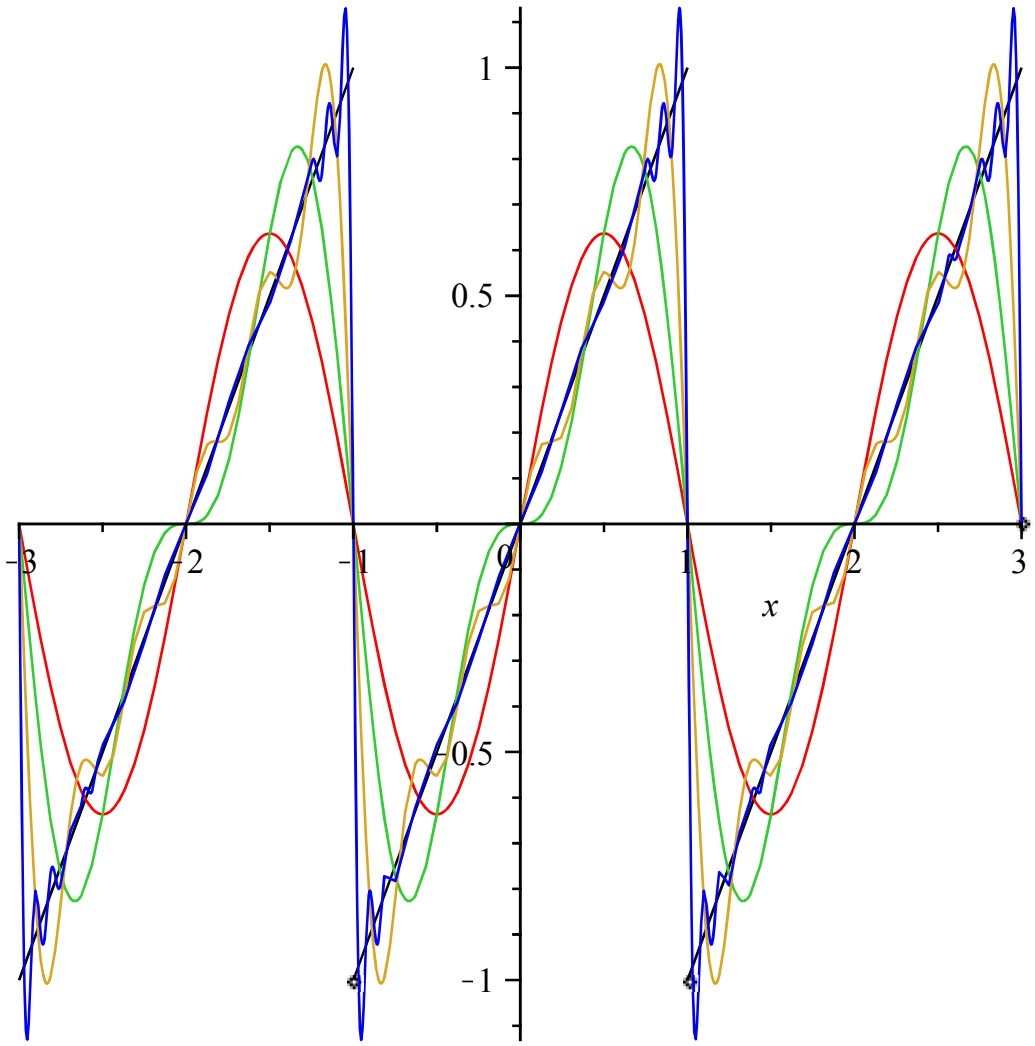
b_n

```
> seq(bn, n=1..8);
```

$$\frac{2}{\pi}, -\frac{1}{\pi}, \frac{2}{3\pi}, -\frac{1}{2\pi}, \frac{2}{5\pi}, -\frac{1}{3\pi}, \frac{2}{7\pi}, -\frac{1}{4\pi}$$

```
> with(plots):
> F:= plot(f, x = -3..3, discont=true, color=black):

> S1 := sum(bn*sin(n*Pi*x), n = 1..1):
  S2 := sum(bn*sin(n*Pi*x), n = 1..2):
  S5 := sum(bn*sin(n*Pi*x), n = 1..5):
  S20 := sum(bn*sin(n*Pi*x), n = 1..20):
  Fplot := plot({S1,S2,S5,S20}, x = -3..3):
  display({F,Fplot});
```



>

Решавање диференцијалних једначина применом Laplace-ове трансформације

Решити једначину са задатим почетним условима:

$y'''+y''-6y'=\sin 4t$;

$y(0)=2$, $y'(0)=0$, $y''(0)=-1$

```
> restart;with(inttrans):
> jednacina:=diff(y(t),t$3) + diff(y(t), t$2) - 6*diff
  (y(t), t)= sin(4*t);
```

$$jednacina := \frac{d^3}{dt^3} y(t) + \frac{d^2}{dt^2} y(t) - 6 \left(\frac{d}{dt} y(t) \right) = \sin(4\,t)$$

```
> korak1:= laplace(jednacina, t,s); #racunamo Laplace-
  a
```

$$korak1 := s^3\, laplace(y(t), t, s) - \mathrm{D}^{(2)}(y)(0) - s\,\mathrm{D}(y)(0) - s^2 y(0) + s^2\, laplace(y(t), t, s) - \mathrm{D}(y)(0) - s\,y(0) - 6\,s\, laplace(y(t), t, s) + 6\,y(0) = \frac{4}{s^2+16}$$

```
> korak11:= subs({y(0) = 2, D(y)(0) =0, (D@@2)(y)(0)=
  -1}, korak1); #ubacujemo uslove iz zadatka
```

$$korak11 := s^3\, laplace(y(t), t, s) + 13 - 2\,s^2 + s^2\, laplace(y(t), t, s) - 2\,s - 6\,s\, laplace(y(t), t, s) = \frac{4}{s^2+16}$$

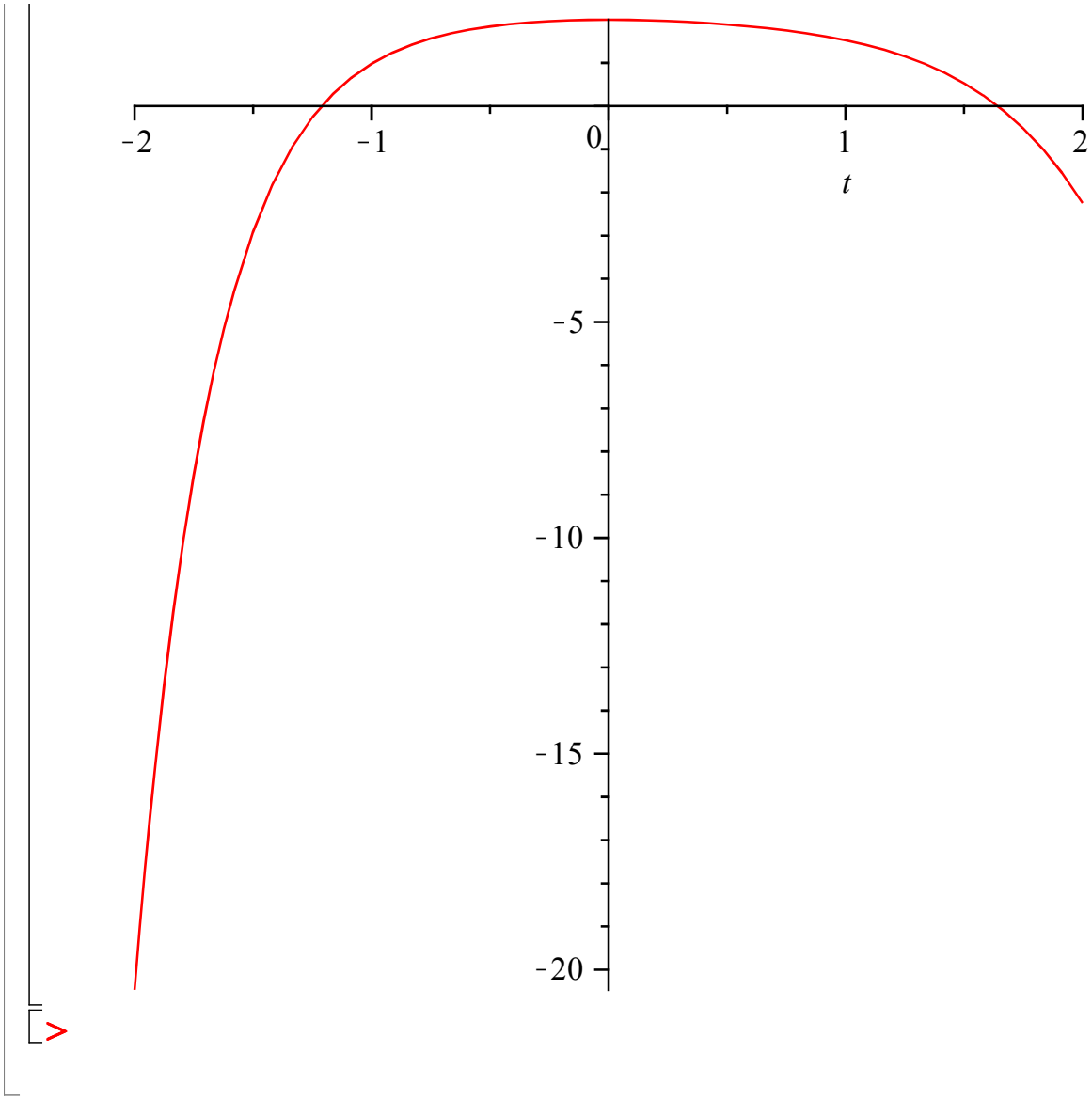
```
> korak2:=simplify(solve(korak11, laplace(y(t), t,s)))
  ; #racunamo resenje za Y(s)
```

$$korak2 := \frac{19\,s^2 - 204 + 2\,s^4 + 2\,s^3 + 32\,s}{s\left(s^4 + 10\,s^2 + s^3 + 16\,s - 96\right)}$$

```
> resenje:=invlaplace(korak2, s,t); #i, konacno
  trazimo fju f(t) pomocu inverznog Laplace-a
```

$$resenje := -\frac{2}{25}\,e^{2t} + \frac{11}{1000}\cos(4\,t) - \frac{1}{500}\sin(4\,t) - \frac{7}{125}\,e^{-3t} + \frac{17}{8}$$

```
> plot(resenje, t = -2..2); #kako to izgleda u
  Dekartovom koordinatnom sistemu.
```


-2
-1
0
1
2
t
-5
-10
-15
-20