Annales Univ. Sci. Budapest., Sect. Comp. 36 (2012) 23–39

COMPARING THE COMPUTATION OF CHEBYSHEV POLYNOMIALS IN COMPUTER ALGEBRA SYSTEMS

Sándor Czirbusz (Budapest, Hungary)

Communicated by Antal Járai

(Received November 22, 2011; accepted January 20, 2012)

Abstract. In this article we compare the efficiency of the computer algebra systems Maple and Sage, using as benchmark the calculation of Chebyshev polynomials with various methods. In most tests, Maple performed better, but Sage is also capable of doing the calculations. We conclude that Sage, although still inferior to Maple in functionality and performance, has by now become a reasonable open-source alternative of commercial computer algebra systems.

1. Introduction

Nowadays the teaching of Computer Algebra in some form is an organic part of higher education. We can select from a range of CA Systems, and it is a hard decision, which one is the best for our objectives.

In 1999 appeared Wester's reputed article: "A Critique of The Mathematical Abilities of CA Systems" [1]. Since then one can multiply the version numbers by three; some systems have disappeared and there are many new ones. In this article we performed tests on timing production of Chebyshev

Key words and phrases: Computer algebra, Chebyshev polynomials.
2010 Mathematics Subject Classification: 68W30.
The Research is supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

polynomials generated in different ways. These two systems are : Maple and Sage. The idea of this paper originated in the article of W. Koepf: "Efficient Computation of Chebyshev Polynomials" [1]. The source of all these tests can be found on my home page (`http://compalg.inf.elte.hu/ czirbusz/`).

Maple is one of the best known erudite CA Systems, at present the current version is 15. In the tests version 11 was used. Sage is a relatively new free system, based on Maxima, many abilities rewritten in Python. The system integrates a large number of free mathematical software products, see [9].

In both systems some timing functions are available, they were used in these tests. The results are measured in seconds, the Maple function returns three digits, the Sage only two digits precision. All results were measured on three computers, the used configurations are listed below:

Fujitsu Notebook: Fusjitsu Siemens Amilo PI2530, 2x Intel(R) Core(TM) 2 Duo CPU 1.50GHz with Ubuntu 10.10 32 bit operating system, Linux 2.6.35-30-generic (i686) kernel, 2059716 kB total memory, swap 71892 kB.

Lenovo Notebook: Lenovo ThinkPad T510, 4 x Intel(R) Core(TM) i7 CPU M 620 2.67GHz with Ubuntu 11.04 64 bit operating system, Linux 2.6.38-8-generic (x86_64) kernel, 1978924 kB total memory, swap 5660 MB.

Desktop Computer: AuthenticAMD, AMD Athlon(tm) 64 X2 Dual Core Processor 3800, 1999.827 MHz with 512 KB cache with Ubuntu Linux, the Ubuntu 11.04 64 bit operating system, 2.6.38-8-generic kernel, 998 MB total memory, 2922 MB swap.

None of Microsoft operation systems were used because on Windows Sage runs only through cygwin.

2. The Chebyshev polynomials

"Chebyshev polynomials are everywhere dense in numerical analysis." This remark illustrates the distinguished role of Chebyshev polynomials in numerical mathematics. They are very important in approximation theory. The relative simplicity of their definitions is easily adapted to symbolic computations and teaching computer algebra, see [2]. Hence they are especially suitable to illustrate the exact arithmetic performance of CA Systems.

2.1. The definition

Chebyshev polynomials of the first kind are orthogonal on the interval $(-1, 1)$ with respect to the weight function $1/\sqrt{1-x^2}$. We can define them in many ways, the two simplest are the trigonometrical one

$$T_n(\cos x) = \cos nx\,, \tag{2.1}$$

and the recursive one

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad T_0(x) = 1, \quad T_1(x) = x\,. \tag{2.2}$$

All the coefficients of these polynomials are integers. In particular $T_n(1) = 1$ for all n, and $T_n(0) = 0$ for odd n, $T_n(0) = (-1)^{n/2}$ if n is even.

2.2. Other properties

There are many interesting and unique properties of these polynomials, which can be found in several textbooks or internet home pages, for example [5, 6, 7, 8, 10, 11, 12]. In the following sections we use these properties without proof or further reference.

3. The tests

In the column headers of resulting tables the following abbreviations were used: “Mp” denotes Maple column, “SN” is Sage Notebook, “SS” is Sage Shell and “Mx” denotes Maxima.

3.1. The built-in capabilities

Maple and Sage have built-in functions to generate Chebyshev–polynomials. In older versions of Maple they were in the `orthpoly` package, but now it is a ”toplevel” function: `ChebyshevT`. The Sage solution is very simple: it calls the Maxima `chebyshev_t` function via an interface. Maple uses polynomials in a non-expanded form. Therefore the `time(expand(ChebyshevT(n,x)))` command was applied. The Sage method is different. First a polynomial ring should be defined then time can be measured as follows:

```
R = PolynomialRing(QQ, 'x')
x = R.gen()
time p = chebyshev_T(n,x)
```

Here the assignment statement is necessary, otherwise in notebook mode Sage prints the expanded polynomial, which is very impractical for large n. Since Sage uses the Maxima function, I tested the Maxima timing as well. The results are in Table 1.

	Built-in capabilities							
n	*Acer notebook*				*Lenovo notebook*			
	Mp	**SN**	**SS**	**Mx**	**Mp**	**SN**	**SS**	**Mx**
10	0	0.03	0.01	0	0	0.02	0.01	0
100	0	0.13	0.08	0.010	0	0.01	0.03	0
500	0.004	1.15	0.38	0.220	0	0.12	0.28	0.100
1000	0.008	3.56	3.54	0.690	0	0.61	0.75	0.360
5000	0.052	*	*	19.10	0.030	175.23	29.74	10.32
10000	0.164	*	*	84.88	0.060	2230.78	309.74	52.90
100000	2.376	*	*	**	0.850	*	*	**
* Does not response within two hours								
** Maxima run out from memory								

	Built-in capabilities			
n	*Desktop PC*			
	Mp	**SN**	**SS**	**Mx**
10	0	0.01	0.01	0
100	0	0.02	0.03	0
500	0.009	0.22	0.28	0.100
1000	0.010	1.36	0.75	0.360
5000	0.069	199.98	29.74	10.32
10000	0.189	*	309.74	52.90
100000	2.889	*	*	**
* Does not response within two hours				
** Maxima run out from memory				

Table 1. Built-in capabilities – computing the polynomials symbolically

It is surprising that Sage failed in the last three rows. This indicates that the interface between Sage and Maxima has a bottleneck.

As we mentioned above $T_n(1) = 1$ for all n. Examining this simple fact we get an astounding result. While Maple and Maxima computes the result in

0 second, Sage uses relatively notable time and on 32 bit system it crashes for $n \geq 5000$. The Sage source is as follows:

```
_init()
return sage_eval(maxima.eval('chebyshev_t(%s,x)'%ZZ(n)) .
  locals={'x':x})
```

Maxima makes the symbolic computation, Sage calls it without further examination of parameters. The situation is the same when instead of exact arithmetic we made floating-point computation with `evalf()` in Maple and `n()` in Sage.

3.2. Method of determinants

We can compute the Chebyshev polynomials of the firts kind as the determinant of the following matrix:

$$\begin{pmatrix} x & -1 & 0 & 0 & \dots & 0 \\ -1 & 2x & -1 & 0 & \dots & 0 \\ 0 & -1 & 2x & -1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 2x & -1 \\ 0 & 0 & \dots & 0 & -1 & 2x \end{pmatrix}. \tag{3.1}$$

It is evident that the result of the expansion of the determinant is the recurrence relation (2.2) within the elements of the polynomial sequence. Naturally, this method is very ineffective, but we get a little insight into the array–handling of computer algebra systems. There was a change in the matrix manipulation of Maple since version 9. Now Maple prefers the `LinearAlgebra` package, which is much more effective then the old `linalg` package. The Maple source is as follows:

```
with(LinearAlgebra);
ChebyshevTDet := proc (n, x)
  return Determinant(Matrix(n, (i,j)-> piecewise(i = 1 and
    j = 1, x,  i = j, 2*x, abs(i-j) = 1, -1, 0)))
end proc
```

The code with the `linalg` package is very similar. The same functionality in Sage is:

```
R1 = PolynomialRing(QQ, 'x')
def ChebyshevTDet(n):
R2 = MatrixSpace(R1,n,sparse=True)
T = R2.matrix(2*x)
T[0,0] = x;
if n > 1:
    T[0,1] = -1; T[n-1,n-2]= -1
    for i in xrange(1,n-1):
        T[i,i-1] = -1; T[i,i+1]= -1
return T.determinant()
```

This code shows the algebraic nature of Sage. See the timing results in Table 2.

	Determinant method			
n	*Acer notebook*			
	Mp*	**Mp****	**SN**	**SS**
10	0.036	0.036	0.03	0.03
50	4.780	5.096	5.30	5.05
100	71.912	76.116	85.08	80.20
150	369.087	424.386	492.58	435.02
200	1244.753	1578.302	1781.65	1660.02
500	***	***	***	***
1000	***	***	***	***
* Maple with LinearAlgebra package				
** Maple with linalg package				
*** Does not response within 2 hours				

	Determinant method			
n	*Lenovo notebook*			
	Mp*	**Mp****	**SN**	**SS**
10	0.00	0.030	0.02	0.03
50	0.129	1.640	2.23	2.33
100	0.600	19.930	31.11	32.90
150	1.230	94.900	166,57	170.34
200	2.390	293.369	594.64	569.60
500	15.820	***	***	***
1000	104.900	***	***	***
* Maple with LinearAlgebra package				
** Maple with linalg package				
*** Does not response within 2 hours				

	Determinant method			
n	*Desktop PC*			
	Mp*	**Mp****	**SN**	**SS**
10	0.029	0.030	0.02	0
50	3.459	3.460	3.70	0
100	47.30	50.049	55.97	0.100
150	239.570	279.510	292.30	0.360
200	784.119	930.850	1004.22	
500	***	***	***	10.32
1000	***	***	***	52.90
* Maple with LinearAlgebra package				
** Maple with linalg package				
*** Does not response within 2 hours				

Table 2. Method of determinants

It can be seen that the difference between the notebook and interactive shell interface of Sage is not significant, so hereafter we will omit this part of the tests. It is apparent that the `LinearAlgebra` package is very effective. We note that the performance of Sage is not too bad. In fact it is similar to that of the `linalg` package.

3.3. Generating function

The generating function of Chebyshev polynomials is

$$F(z) = \frac{1}{2}\left(\frac{1-z^2}{1-2xz+z^2}+1\right). \tag{3.2}$$

From this we can generate Chebyshev polynomials by Taylor series expansions. The results are in Table 3.

By the old test in [1] Maple failed from $n = 300$. Now we made it up to $n = 500$ without problem, and without deleting the remember table of the differential operator. Sage is systematically slower by a factor of three to six. This is because Sage uses Maxima for calculus and according to the previous tests this interface is very slow. The method is very impractical, and for $n \geq 1000$ Maple caused system crash.

n	Generating function					
	Acer notebook		*Lenovo notebook*		*Desktop PC*	
	Maple	**Sage**	**Maple**	**Sage**	**Maple**	**Sage**
10	0.040	0.09	0.020	0.06	0.020	0.07
50	0.612	1.80	0.310	0.89	0.659	1.27
100	2.400	7.12	1.019	3.51	2.559	5.10
200	9.588	28.38	4.219	13.89	10.310	20.55
400	39.542	113.24	16.559	55.61	42.829	81.48
500	61.251	177.27	26.129	86.77	68.360	128.27
700	121.039	350.95	52.019	350.95	140.340	252.32

Table 3. Generating function

3.4. Rodrigues formulas

We may represent the Chebyshev polynomials of first kind with Rodrigues formulas

$$T_n(x) = \frac{(-2)^n n!}{(2n)!}\sqrt{1-x^2}\,\frac{d^n}{dx^n}(1-x^2)^{n-\frac{1}{2}}\,. \tag{3.3}$$

The code in Maple is straightforward. In Sage we must pay attention to the `diff(f,x,n)` function in which it is not always clear that n is a repetition parameter or the second variable of a partial derivation. See the timing results in Table 5.

We note that there exist similar Rodrigues formulas for other kinds of orthogonal polynomials.

n	Maple	Sage
10	0	0,09
50	0,007	0,42
100	0,024	1,32
200	0,088	6,43
300	0,156	17,45
400	0,784	39,66
500	1,184	78,53

Table 4. Rodrigues formula

	Generating function					
n	*Acer notebook*		*Lenovo notebook*		*Desktop PC*	
	Maple	**Sage**	**Maple**	**Sage**	**Maple**	**Sage**
10	0	0.09	0	0.06	0	0.07
50	0.007	0.42	0	0.21	0	0.31
100	0.024	1.32	0.010	0.61	0.030	1.16
200	0.088	6.43	0.049	2.49	0.200	6.49
400	0.784	39.66	0.469	15.59	0.950	48.95
500	1.184	78.53	0.680	29.57	1.430	98.71
1000	*	**	3.860	241.34	**	**
5000	*	**	*	**	*	**
* System freezes						
** No response within 2 hours						

Table 5. Generating function

3.5. Matrix powers

This method is based on [3] and [4]. The Chebyshev polynomials may be represented by

$$\begin{pmatrix} T_n(x) \\ T_{n-1}(x) \end{pmatrix} = \begin{pmatrix} 2x & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} T_{n-1}(x) \\ T_{n-2}(x) \end{pmatrix} = \cdots = \begin{pmatrix} 2x & -1 \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} x \\ 1 \end{pmatrix}. \tag{3.4}$$

The matrix operations are different in the `LinearAlgebra` and the `linalg` packages of Maple, and the evaluation strategy is different as well. Both systems use fast matrix powering, we do not need any tricky codes, the programs in essence are identical in Maple and Sage. The code with `linalg` package:

```
T0 := matrix([[2*x, -1], [1, 0]]);
T1 := matrix([[x], [1]]);
T := (n, x) -> evalm(‘&*‘(T0^n, T1));
```

The timing results are in Table 6.

Firstly, we note that the running time is less than in the previous methods. Secondly, the result of the `LinearAlgebra` shows a little fluctuation. Probably this is the sampling error of time function. Additionally, we can see that `linalg` is significantly slower. Sage is surprisingly good, especially for larger n-s it is better than `linalg`. It should be noted that Maple uses non-expanded form for polynomials in matrix cells, on the other hand Sage expands the same elements. Further we note that for large n Maple is not able to display the matrix while Sage truncates the screen output and saves it to an external file.

n	Matrix Powering								
	Acer notebook			*Lenovo notebook*			*Desktop PC*		
	Mp*	**Mp****	**Sage**	**Mp***	**Mp****	**Sage**	**Mp***	**Mp****	**Sage**
10	0.003	0.003	0.02	0	0	0.02	0	0	0
50	0.003	0.004	0.03	0	0	0.00	0	0	0.02
100	0.004	0.011	0.05	0	0	0.02	0	0.009	0.03
200	0.008	0.016	0.05	0	0.009	0.04	0	0.009	0.03
500	0.007	0.060	0.08	0	0.030	0.04	0	0.019	0.05
1000	0.007	0.212	0.08	0	0.100	0.03	0	0.189	0.05
2000	0.008	0.732	0.10	0	0.370	0.04	0	0.690	0.06
5000	***	***	***	0	****	0.03	0.010	****	****
* Maple with LinearAlgebra package									
** Maple with linalg package									
***Not tested									
**** length of output exceeds limit of 1000000									

Table 6. Matrix powering

3.6. Per definitionem

3.6.1. A trigonometric joke

From (2.1) it follows that

$$T_n(x) = \cos(n \arccos x)\,. \tag{3.5}$$

From this expression Maple computes the Chebyshev polynomials for $n = 1 \ldots 99$ in 0 second. Virtually, it seems that this method works for larger n-s, but Maple is not able to expand the expression. Sage works in a similar manner and the expression is the expanded Chebyshev polynomial for large n-s.

3.6.2. The recursive formula

This formula is a simple programming exercise, essentially identical in most programming languages. (Except the symbolic handling of variable x.) This is true for the iterative version as well. In general the iterative version should be significantly more effective than the recursive version. But Maple has a very good solution for recursive programs, this is the remember table. When the remember option is defined into a procedure then after every execution this procedure inserts an entry to the remember table that records the return values. When the procedure is used with the same parameter values then the results are simply retrieved from here. There does not exist similar mechanism in Sage. The results of these tests presented in Tables 7 and 8.

n	The recursive relation in Maple					
	Acer notebook			*Lenovo notebook*		
	Mp*	**Mp****	**iterative**	**Mp***	**Mp****	**iterative**
10	0	0	0.003	0	0	0
50	0	***	***	0	***	0
100	0	***	***	0	***	0
500	0.016	***	***	0.009	***	0.009
1000	0.044	***	***	0.030	***	0.050
10000	1.852	***	***	0.0830	***	5.029
20000	34.146	***	***	****	***	27.670
* Maple with remember table						
** Maple without remember table						
*** No response within 2 hours						
**** Too many levels of recursion						

n	The recursive relation in Maple		
	Desktop PC		
	Mp*	**Mp****	**iterative**
10	0	0	0
50	0	***	0
100	0	***	0
500	0.009	***	0.010
1000	0.0040	***	0.039
10000	2.830	***	20.269
20000	****	***	111.430

Table 7. The recursive relation in Maple

n	The recursive relation in Sage					
	Acer notebook		*Lenovo notebook*		*Desktop PC*	
	recursive	**iterative**	**recursive**	**iterative**	**recursive**	**iterative**
10	0	0.01	0.01	0	0.01	0.03
50	*	0.06	**	0.02	**	0.04
100	*	0.13	-	0.05	-	0.13
500	*	2.62	-	1.53	-	2.10
1000	*	9.98	-	6.69	-	7.68
10000	*	*	-	**	-	**
20000	*	*	-	-	-	-
* No response within 2 hours						
** Javascript error: the name “Cheby” is not defined						

Table 8. The recursive relation in Sage

As it was noted Maple works very fast with the remember table. The recursive version without remember table caused system crash. In case of Sage only the iterative program gives a reasonable solution, but the running time grows very quickly with n. It is a surprise that the recursive version basically does not work.

3.7. An algebraically closed formula

There is a simple formula to compute the Chebyshev polynomials:

$$\frac{1}{2}\left(x+\sqrt{x^2-1}\right)^n+\frac{1}{2}\left(x-\sqrt{x^2-1}\right)^n.$$

Simply calling this expression with some n is a zero–time operation, thus the test was executed in Maple with the *expand* function, and on Sage with *simplify* and *expand*.

3.8. Differential equations

The Chebyshev polynomials are the uniqe solutions of the differential equations

$$(1-x^2)T_n''(x)-xT_n'(x)+n^2T_n(x)=0 \tag{3.6}$$

with initial conditions

$$\begin{aligned} T_n'(0) &= 0, \text{ if } n \text{ is even }, (-1)^{\frac{n-1}{2}} \text{ otherwise} \\ T_n(0) &= 0, \text{ if } n \text{ is odd}, (-1)^{\frac{n}{2}} \text{ otherwise}. \end{aligned} \tag{3.7}$$

Maple has a very impressive knowledge in the area of differential equations. It has a very powerful algorithmic background, the `dsolve()` function solves this equation in 0.04 sec if $n = 10000$. But a closer examination of solutions shows that it is a compound of trigonometric functions, and fairly hard to convert to polynomial form. Using the "series" option of "dsolve" function, the solution is slower by orders of magnitude. The solution is a truncated formal power series, which one must convert. Some Maple code follows:

```
In := proc (n)
  if `mod`(n, 2) = 1 then return
    [0, n*(-1)^((1/2)*n-1/2)]
  else
    return [(-1)^((1/2)*n), 0]
```

```
  end if
end proc;
n:=500;
Order :=n;
dsolve({eq, f(0) = In(n)[1],
            (D(f))(0) = In(n)[2]}, f(x), 'series')
```

Sage is very poor in this challenge. It uses Maxima, and solves only first and second order linear equations and some other special equations. In this case the program gets lost in the labyrinth of compound trigonometric and hyperbolic functions.

Certainly there is an alternate way to solve this equation. Replacing x with $\cos t$ the above equation becomes

$$f''(t) + n^2 f(t) = 0.$$

In Maple it is easy to deduce this form with the **dchange** statement of PDETools package, there is no similar mechanism in Sage. The solution is simple in both systems, but there remains the task of converting to polynomial form.

3.9. Series representation

One of best known series representation of Chebyshev polynomials follows:

$$T_n(x) = \frac{n}{2} \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^k \frac{(n-k-1)!}{k!(n-2k)!} (2x)^{n-2k}. \tag{3.8}$$

Both systems have an elegant `sum` function. But for large n-s the direct summing is not effective due to the calls of the factorial function. However, by using the recursive connection between the above coefficients we can make an effective recursive program in Maple. Since there is no remember table in Sage, for $n = 2000$ we get the error: "maximum recursion depth exceeded". The timing results are in Table 9.

MpS	Series representation							
	Acer notebook				*Lenovo notebook*			
	MpS	**MpP**	**SS**	**SP**	**MpS**	**MpP**	**SS**	**SP**
10	0	0	0.02	0.01	0	0	0.03	0.02
50	0	0.004	0.03	0.13	0	0	0.02	0.01
100	0.004	0	0.05	0.51	0	0	0.04	0.01
500	0.024	0.008	0.92	12.50	0	0	0.39	0.07
1000	0.100	0.024	6.78	51.41	0.030	0.020	2.50	0.16
2000	*	0.168	51.76	*	*	0.060	18.17	0.44
5000	-	0.804	930.32	-	-	0.419	296.30	2.49
10000	-	*	**	-	-	1.730	2145.61	7.48
20000	-	-	-	-	-	8.370	**	29.71
MpS = Maple with built-in Sum, MpP = Maple with with our program								
SS = Sage with built-in sum, SP = Sage with our program								
* system hangs								
** no respond within 2 hours								
*** exception runtime error								

MpS	Series representation			
	Desktop PC			
	MpS	**MpP**	**SS**	**SP**
10	0	0	0.02	0
50	0	0	0.02	0.01
100	0	0	0.06	0.02
500	0.010	0.010	1.70	0.33
1000	0.060	0.049	10.35	1.19
2000	**	0.150	74.16	4.71
5000	-	1.100	1071.28	28.89
10000	-	6.890	**	116.22
20000	-	43.010	-	**

Table 9. Series representation

3.10. Divide and conquer

There is an elegant relation between two arbitrary Chebyshev polynomials:

$$2T_n(x)T_m(x) = T_{n+m}(x) + T_{n-m}. \tag{3.9}$$

Applying this formula first with $m = n$, then $m = n - 1$, we get by far the

most effective procedure to make Chebyshev polynomials. This is true for both systems. The Sage code is as follows:

```
def T(n,x):
        if n == 0:
            return 1
        elif n == 1:
            return x
        elif Mod(n,2) == 0:
            return 2*T(n//2,x)^2-1
        else:
            return 2*T((n-1)/2,x)*T((n+1)/2,x)-x
```

In the resulting table only the Sage times are represented because the Maple running time is always zero.

Divide & Conquer			
n	*Acer*	*Lenovo*	*Desktop*
100	0.01	0	0
500	0.01	0	0.01
1000	0.03	0.02	0.01
5000	0.05	0.03	0.03
10000	0.05	0.01	0.03
10^6	0.28	0.12	0.21
10^9	9.81	3.93	7.57
10^{12}	200.68	81.72	156.95

Table 10. Divide and conquer

3.10.1. Some notes

Both systems use the non-expanded form. (In Sage the computation used the Symbolic Ring.) For large n-s both systems may calculate the degree of polynomials very fast, but expanding the polynomials is impossible.

4. Summary notes

These tests were a little unfair with Sage. Maple has very impressive knowledge in the area of mathematical analysis. It was written and compiled in C language, although the Java visualization is not very successful. The Sage developers are primarily algebraists, mainly using Maxima as a black-box for calculus. It causes serious problems, as we have seen in case of the built-in capabilities. As we can see beside "extreme" conditions (say large n) Maple is far better. It follows that for research purposes Maple is more appropriate, especially in calculus. However, we experienced system crash several times in case of Maple. For education purposes I prefer Sage. On the one hand it is free, and on the other hand in contempt of some "child's illnesses", the Web based interface is modern, and usually is the winner for students.

References

[1] **Wester, M.J.,** *Computer Algebra Systems - A Practical Guide*, John Wiley & Sons Ltd., 1999.

[2] **Geddes, K.O. and S.R. Czapor and G. Labahn,** *Algorithm for Computer Algebra*, Kluwer Academic Publisher, Boston, MA, 1992.

[3] **Fateman, R.,** *Lookup Tables, Recurrences and Complexity*, Proceedings of ISSAC'89", ACM Press.

[4] **Miller, J.C.P. and D.J.S. Brown,** An algorithm for evaluation of remote terms in a linear recurrence sequence, *The Computer Journal*, **9** (1966), 188–190.

[5] **Spanier, J. and K.B. Oldham,** *An Atlas of Functions*, Hemisphere Publishing Corporation, Washington, DC, 1987.

[6] **Mason J.C. and D.C. Handscomb,** *Chebyshev polynomials*, Chappman & Hall/CRC, Boca Raton, FL, 2003.

[7] **Culham, J.R.,** *Advanced Differential Equations And Special Functions*, `http://www.mhtl.uwaterloo.ca/courses/me755/`

[8] **Rivlin, T.J.,** *The Chebyshev Polynomials*, Pure and applied mathematics, John Wiley & Sons Ltd., New York, 1974.

[9] Sage Standard Packages, `http://sagemath.org/packages/standard/`

[10] **Weisstein, E.W.,** *Chebyshev Polynomials of the First Kind*, http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html
[11] Wikipedia, *Chebyshev polynomials*, http://en.wikipedia.org/wiki/Chebyshev_polynomials
[12] **Suetin, P.K.,** *Classical Orthogonal Polynomials*, Nauka, Moscow, 1976 (in Russian).

S. Czirbusz
Department of Computer Algebra
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest, Pázmány P. sétány 1/C
Hungary
czirbusz@compalg.inf.elte.hu