Journal of Physical Sciences, Vol. 10, 2006, 18 – 29

A Modified DETGTRI Algorithm with Applications

Moawwad E.A.El-Mikkawy
Department of Mathematics, Faculty of Science, Mansoura University, Mansoura, 35516, EGYPT e-mail: mikkawy@yahoo.com

Received 30 December, 2006 ; accepted 30 December, 2006

ABSTRACT

In the current article we present a modified version of the DETGTRI algorithm in [1]. Based on the modified algorithm , an efficient and reliable Maple procedure is written to compute any nth order tri-diagonal determinant. As an application, we show that all orthogonal polynomials can be computed efficiently using this procedure.

Keywords : Orthogonal Polynomials , Determinants, Recurrence Relations, Maple.

1. Introduction

We begin this section by considering the general tri-diagonal determinant T of order n of the form :

$$T = \begin{vmatrix} d_1 & a_1 & 0 & \cdots & \cdots & 0 \\ b_2 & d_2 & a_2 & \ddots & & \vdots \\ 0 & b_3 & d_3 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & a_{n-1} \\ 0 & \cdots & \cdots & 0 & b_n & d_n \end{vmatrix} \tag{1.1}$$

These types of determinants and their corresponding matrices play a fundamental role in many areas of science and engineering. In [1] the author developed an algorithm called

DETGTRI to compute nth order tri-diagonal determinants in linear time. Based on the DETGTRI algorithm we obtain

$$T = \begin{vmatrix} d_1 & b_2a_1 & 0 & \cdots & \cdots & 0 \\ 1 & d_2 & b_3a_2 & \ddots & & \vdots \\ 0 & 1 & d_3 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & b_na_{n-1} \\ 0 & \cdots & \cdots & 0 & 1 & d_n \end{vmatrix} \qquad (1.2)$$

From (1.2), we see that any nth order general tri-diagonal determinant of the form (1.1) can be stored in at most 2n memory locations by using only two vectors. These vectors are a = $(a_1, a_2, ..., a_n)$ and d = $(d_1, d_2, ..., d_n)$. This is always a good habit in computation in order to save memory space.

The form (1.2) is very useful in many situations. For example , this form enables us to represent the Fibonacci numbers F_n in the forms :

$$F_n = \begin{vmatrix} 1 & -1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & -1 & \ddots & & \vdots \\ 0 & 1 & 1 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & -1 \\ 0 & \cdots & \cdots & 0 & 1 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & 1 & \ddots & & \vdots \\ 0 & -1 & 1 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{vmatrix}, \; n = 1,2, \ldots. \qquad (1.3)$$

instead of the form

$$F_n = \begin{vmatrix} 1 & 2 & 0 & \cdots & \cdots & 0 \\ -\frac{1}{2} & 1 & 3 & \ddots & & \vdots \\ 0 & -\frac{1}{3} & 1 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & n \\ 0 & \cdots & \cdots & 0 & -\frac{1}{n} & 1 \end{vmatrix}, \; n = 1,2, \ldots. \qquad (1.4)$$

which is given in [2].

The form (1.2) can also be used to prove some facts directly. For example, we can use it to prove that the following nth order determinant

$$A = \begin{vmatrix} k & x & 0 & \cdots & \cdots & 0 \\ \frac{1}{x} & k & x & \ddots & & \vdots \\ 0 & \frac{1}{x} & k & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & x \\ 0 & \cdots & \cdots & 0 & \frac{1}{x} & k \end{vmatrix} \tag{1.5}$$

does not depend on x (see also [3] , p.105) .
Perhaps, more interesting is the fact the form (1.2) helps us to reformulate the DETGTRI algorithm in a more economical form as follows:

The Modified DETGTRI Algorithm
To compute the nth order tri-diagonal determinant of the form (1.1) , we may proceed as follows:
Step 1: Use the recurrence relation

$$c_i = \begin{cases} d_1 & \text{if i = 1} \\ d_i - \dfrac{a_{i-1}}{c_{i-1}} & \text{if i = 2, 3, ..., n} \end{cases} \tag{1.6}$$

to compute the simplest rational forms of the n components of the vector $c = (c_1, c_2, ..., c_n)$.

If $c_i = 0$ for any $i \le n$, set $c_i = m$ (μ is just a symbolic name) and continue to compute $c_{i+1}, c_{i+2}, ..., c_n$ in terms of by using (1.6).

Step 2 : The simplest rational form of the product $\prod_{r=1}^{n} c_r$ (this product is a polynomial in μ) evaluated at $\mu = 0$ is equal to the determinant of the matrix T in (1.1).

It is well known that [4, 5] for all n, any set of orthogonal polynomials $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ satisfies a three-term recurrence relation of the form :

$$\phi_k(x) = (A_k x + B_k)\,\phi_{k-1}(x) - C_{k-1}\,\phi_{k-2}(x) \tag{1.7}$$

where $\phi_{-1}(x) = 0$, $\phi_0(x) = 1$ and A_k, B_k and C_k are real constants and does not depend on x.

The Chebyshev polynomials of the first kind $T_n(x)$ arises as a solution to the differential equation

$$(1-x^2)y'' - xy' + n^2 y = 0 \tag{1.8}$$

and those of the second kind as a solution to

$$(1-x^2)y'' - 3xy' + n(n+2)y = 0 \tag{1.9}$$

The Chebyshev polynomials of the first kind , are defined by the recurrence relation

$$T_0(x) = 1,\quad T_1(x) = x,\quad T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x),\ n = 2,3,\ldots \tag{1.10}$$

The Chebyshev polynomials of the second kind, $U_n(x)$ are defined by the recurrence relation

$$U_0(x) = 1,\quad U_1(x) = 2x,\quad U_n(x) = 2xU_{n-1}(x) - U_{n-2}(x),\ n = 2,3,\ldots \tag{1.11}$$

It is known that [6,7] , the Chebyshev polynomials $T_n(x)$ and $U_n(x)$ are characterized by tri-diagonal determinants as follow :

$$T_0(x) = 1,\ T_k(x) = \begin{vmatrix} x & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 2x & 1 & \ddots & & \vdots \\ 0 & 1 & 2x & 1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 2x \end{vmatrix},\ k = 1,2,\ldots,n \tag{1.12}$$

For the Chebyshev polynomials of the second kind $U_n(x)$, we have

$$U_0(x)=1,\ U_k(x)=\begin{vmatrix} 2x & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 2x & 1 & \ddots & & \vdots \\ 0 & 1 & 2x & 1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 2x \end{vmatrix},\ k=1,2,\ldots,n \qquad (1.13)$$

Therefore it is an interesting question to ask whether any classical orthogonal polynomial can be represented by a determinant. One of the main objectives of the current paper is to answer this question. The main result is presented in section 2.

2. Main Results

By using the recurrence relation (1.7) together with the modified DETGTRI algorithm, which is mainly based on a two-term recurrence, we get

$$\phi_0(x)=1,\ \phi_k(x)=\begin{vmatrix} A_1x+B_1 & C_1 & 0 & \cdots & \cdots & 0 \\ 1 & A_2x+B_2 & C_2 & \ddots & & \vdots \\ 0 & 1 & A_3x+B_3 & C_3 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & C_{n-1} \\ 0 & \cdots & \cdots & 0 & 1 & A_nx+B_n \end{vmatrix} \qquad (2.1)$$

(k=1,2,…., n)

showing directly that any orthogonal polynomial can be characterized by a tri-diagonal determinant as long as its recurrence relation of the form (1.7) is available. This is the answer to the question raised in this paper.

As examples, consider the Legendre polynomials $P_n(x)$ which satisfy the three-term recurrence

$$P_0(x)=1,\ P_k(x)=\frac{(2k-1)}{k}x\ P_{k-1}(x)-\frac{(k-1)}{k}P_{k-2}(x)\ ,\ k=1,2,\ldots,n \tag{2.2}$$

Consequently from (2.1) we get

$$P_0(x)=1\ ,P_k(x)=\begin{vmatrix} x & \frac{1}{2} & 0 & \cdots & \cdots & 0 \\ 1 & \frac{3}{2}x & \frac{2}{3} & \ddots & & \vdots \\ 0 & 1 & \frac{5}{3}x & \frac{3}{4} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & \frac{k-1}{k} \\ 0 & \cdots & \cdots & 0 & 1 & \frac{(2k-1)}{k}x \end{vmatrix},\ k=1,2,\ldots,n \tag{2.3}$$

For the Laguerre and Hermite polynomials we have respectively the recurrence relations

$$L_0(x)=1,\ L_1(x)=1-x,\ L_n(x)=(2n-1-x)L_{n-1}(x)-(n-1)^2L_{n-2}(x)\ ,\ n=2,3,.. \tag{2.4}$$

and

$$H_0(x)=1,\ H_1(x)=1-x,\ H_n(x)=2xH_{n-1}(x)-2(n-1)H_{n-2}(x) \tag{2.5}$$

Hence using (2.1) gives the characterization

$$L_0(x)=1\ ,L_k(x)=\begin{vmatrix} 1-x & 1^2 & 0 & \cdots & \cdots & 0 \\ 1 & 3-x & 2^2 & \ddots & & \vdots \\ 0 & 1 & 5-x & 3^2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & (k-1)^2 \\ 0 & \cdots & \cdots & 0 & 1 & 2k-1-x \end{vmatrix},\ k=1,2,\ldots,n \tag{2.6}$$

and

$$H_0(x)=1, H_k(x)=\begin{vmatrix} 2x & 2 & 0 & \cdots & \cdots & 0 \\ 1 & 2x & 4 & \ddots & & \vdots \\ 0 & 1 & 2x & 6 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & \ddots & \ddots & 2k-2 \\ 0 & \cdots & \cdots & 0 & 1 & 2x \end{vmatrix}, \quad k=1,2,\ldots,n \tag{2.7}$$

For the sake of completeness, we implemented the modified DETGTRI algorithm of this paper in order to compute any orthogonal polynomial by using tri-diagonal determinants .

A Maple Procedure to Compute Orthogonal Polynomials and Tri-diagonal Determinants

```
> # A Maple procedure to compute orthogonal polynomials.
> # Written by : Prof. Dr. Moawwad E.A.El-Mikkawy.
> # Date : 01 / 09 / 2006.
> # Here 2 vectors b and d only are used.
> # Note that the procedure reuses d to store c.
> # The procedure may be used to compute any tridiagonal determinant
> # in which all b[i] = 1. This form is actually easy to obtain.
> # All you need to do to obtain this form is to replace each a[i] by b[i+1]*a[i],i=1(1)n-1 in the original determinant.
> restart:
> with(linalg,vector):
> tridet := proc(n::posint,d::vector,a::vector)
      local i,r ; global V;
       # Components of c by using d.
      if d[1] = 0 then d[1] := mu ; fi:
      for i from 2 to n do
         d[i] := simplify(d[i] - a[i-1]/d[i-1]);
         if d[i] = 0 then d[i] := mu ; fi:
      od:
      # To compute T = det (A) which is the required polynomial.
      V := subs (mu = 0,simplify(product(d[r],r=1..n))):
       eval (expand(V)):
 end:
```

```
> # Call no.1 . To compute the Legendre polynomial P(7,x)
> mu :='mu': n:=7: d:=array(1..n): a:=array(1..n):

> if n =1 then a[1] := 0 else a[1]:= 1/2 ;fi:d[1]:=x:
> for i from 2 to n do
     d[i]:= (2*i-1)*x/i;
     a[i]:= i/(i+1);
  od:

> P||n||x := tridet(n,d,a);
```

$$P7x := \frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x$$

```
> # Check
> simplify(1/(2^n*(n!))*diff((x^2-1)^n,x$n));
```

$$\frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x$$

```
> # Another way to check via the orthopoly package.
>  orthopoly[P](n,x);
```

$$\frac{429}{16}x^7 - \frac{693}{16}x^5 + \frac{315}{16}x^3 - \frac{35}{16}x$$

```
> # Excellent agreement. OK.
> # Call no.2 . To compute the Chebyshev polynomial T(8,x)
> mu :='mu':n:= 8: d:= array(1..n): a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:=1 ;fi:d[1]:=x:
> for i from 2 to n do d[i] := 2*x ; a[i] := 1 ; od:
```

$$T8x := 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$

```
> T||n||x := tridet(n,d,a);
```

$$128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$

```
> # Check using the orthopoly package.
> orthopoly [T](n,x);

> # Excellent agreement. OK.
> # Call no.3 . To compute the Chebyshev polynomial U(8,x)
> mu :='mu':n:= 8: d:= array(1..n):a:= array(1..n):
```

```
> if n =1 then a[1] := 0 else a[1]:= 1 ;fi: d[1]:=2*x:
> for i from 2 to n do d[i] := 2*x ; a[i] := 1 ; od:

> U||n||x := tridet(n,d,a);
```

$$U8x := 256x^8 - 448x^6 + 240x^4 - 40x^2 + 1$$

```
> # Check using the orthopoly package.
> orthopoly [U](n,x);
```

$$256x^8 - 448x^6 + 240x^4 - 40x^2 + 1$$

```
> # Excellent agreement. OK.
> # Call no.4 . To compute the Laguerre polynomial L(9,x)
> mu :='mu': n:=9: d:= array(1..n) : a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:= 1 ;fi: d[1]:=1-x:
> for i from 2 to n do d[i] := -x+2*i-1; a[i] := i^2 ;od:

> L||n||x := tridet(n,d,a);
```

$$L9x := 81x^8 - x^9 - 2592x^7 + 42336x^6 - 381024x^5 + 381024x^4 - 5080320x^3 + 6531840x^2 - 3265920x + 362880$$

```
> # Check
> simplify(exp(x)*diff(x^n*exp(-x),x$n));
```

$$81x^8 - x^9 - 2592x^7 + 42336x^6 - 381024x^5 + 1905120x^3 + 6531840x^2 - 3265920x + 362880$$

```
> # Another way to check via the orthopoly package.
> # L(n,x) in orthopoly package should be multiplied by n!.
> n!*(orthopoly [L] (n,x));
```

$$81x^8 - x^9 - 2592x^7 + 42336x^6 - 381024x^5 + 1905120x^4 - 5080320x^3 + 6531840x^2 - 3265920x + 362880$$

```
> # Excellent agreement. OK.
> # Call no.5 . To compute the Hermite polynomial H(10,x)using arrays.
> mu :='mu': n:=10: d:=array(1..n): a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:= 2 ;fi: d[1]:= 2*x:
> for i from 2 to n do d[i]:= 2*x; a[i]:= 2*i; od:

> H||n||x := tridet(n,d,a);
```

$$H10x := 1024x^{10} - 23040x^8 + 161280x^6 - 403200x^4 + 302499x^2 - 30240$$

> # Check using the orthopoly package.

> orthopoly [H](n,x);

$$1024x^{10} - 23040x^8 + 161280x^6 - 4032400x^2 - 30240$$

> # Excellent agreement. OK.
> # Sylevester determinant.
> # Call no.6 . To compute Sylevester determinant of order 10.
> mu :='mu': n:=10: d:=array(1..n): a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:= m-1 ;fi: d[1]:= x:
> for i from 2 to n do d[i]:= x; a[i]:= (m-i)*i; od:

> factor(tridet(n,d,a));

$$893025 + 53550x^4m^2 - 100800x^2m^3 + 23625m^4 - 7560x^6m + 897750m^2 + x^{10}$$
$$-2255040x^2m - 217350m^3 + 630x^6m^2 - 284130x^4m - 3150x^4m^3 + 4725x^2m^4$$
$$-45x^8m - 945m^5 + 749070x^2m^2 + 2250621x^2 - 159610m + 463490x^4$$
$$+21378x^6 + 285x^8$$

> collect(%,x);

$$x^{10} + (-45m + 285)x^8 + (630m^2 - 7560m + 21378)x^6$$
$$+(-284130m + 53550m^2 + 4634490 - 3150m^3)x^4$$
$$+(-100800m^3 - 2255040m + 749070m^2 + 4725m^4 + 2250621)x^2$$
$$+893025 - 945m^5 + 23625m^4 - 1596105m + 897750m^2 - 217350m^3$$

> Sylvdet:= factor(eval(%,m=n));

$$Sylvdet := (x - 1)(x+9)(x-3)(x+7)(x-5)(x+5)(x-7)(x+3)(x-9)(x+1)$$

> # Check
> product(x-2*j+n-1,j=0..n-1);

$$(x - 1)(x+9)(x-3)(x+7)(x-5)(x+5)(x-7)(x+3)(x-9)(x+1)$$

> # Excellent agreement. OK.

```
> # Call no.7 . To compute a tri-daiagonal determinant of order 10.
> # tri-daiagonal determinant in Ref.[8].
> mu :='mu': n:=10: d:=array(1..n): a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:= -1/((alpha*beta)^2) ;fi: d[1]:= -(alpha+beta)/((alpha*beta)^2):
> for i from 2 to n do d[i]:= (alpha+beta)/(alpha*beta); a[i]:=1/(alpha*beta); od:

> detval:= simplify(tridet(n,d,a));
```

$$detval := -\frac{\alpha^{10}+\alpha^{8}\beta^{2}+\alpha^{6}\beta^{4}+\alpha^{4}\beta^{6}+\alpha^{2}\beta^{8}+\alpha^{9}\beta+\alpha^{7}\beta^{3}+\alpha^{5}\beta^{5}+\alpha^{3}\beta^{7}+\alpha\beta^{9}+\beta^{10}}{\alpha^{11}\beta^{11}}$$

```
> # Check
> print(sum(alpha^(n-j)*beta^j,j=0..n)/((alpha*beta)^(n+1)));
```

$$\frac{\alpha^{10}+\alpha^{8}\beta^{2}+\alpha^{6}\beta^{4}+\alpha^{4}\beta^{6}+\alpha^{2}\beta^{8}+\alpha^{9}\beta+\alpha^{7}\beta^{3}+\alpha^{5}\beta^{5}+\alpha^{3}\beta^{7}+\alpha\beta^{9}+\beta^{10}}{\alpha^{11}\beta^{11}}$$

```
> # Excellent agreement. OK.
>
> # Call no.8 . To compute a tri-daiagonal determinant of order 10.
> # tri-daiagonal determinant in Ref.[8].
> mu :='mu': n:=10: d:=array(1..n): a:= array(1..n):

> if n =1 then a[1] := 0 else a[1]:= 2/ (alpha * beta) ;fi: d[1]:= (alpha+beta)/(alpha*beta):
> for i from 2 to n do d[i]:= (alpha+beta)/(alpha*beta); a[i]:=1/(alpha*beta); od:

> detval:= simplify(tridet(n,d,a));
```

$$detval := \frac{\alpha^{10}+\beta^{10}}{\alpha^{10}\beta^{10}}$$

```
> # Check
> print((alpha^n + beta^n)/((alpha*beta)^n));
```

$$\frac{\alpha^{10}+\beta^{10}}{\alpha^{10}\beta^{10}}$$

```
> # Excellent agreement. OK.
>
> # Call no.9. To compute Fibonacci numbers F15.
> mu :='mu': n:=15: d:=array(1..n): a:= array(1..n):
```

$$F15 := 987$$

```
> if n =1 then a[1] := 0 else a[1]:= -1 ;fi: d[1]:= 1:
> for i from 2 to n do d[i]:= 1; a[i]:= -1; od:
                                                    987
> F||n := simplify(tridet(n,d,a));

> # Check
> combinat[fibonacci](n+1);

> # Excellent agreement. OK.
```

REFERENCES

1. M.E.A.El-Mikkawy, A fast algorithm for evaluating nth order tri-diagonal determinants , J. Comput. Appl. Math. , Vol. 166 , pp. 581-584 , 2004.
2. R.Witula,D.Stota, On Computing the determinants and inverses of some special type of tridiagonal and constant diagonals matrices, Appli. Math. Comput. , doi : 10.1016 / j.amc.2006.11.112 , 2007.
3. F. Zhang, Matrix Theory, Springer, New York, Berlin, Heidelberg, 1999.
4. M.Abramowitz and I.A. Stegun (Eds.), Handbook of Mathematical Functions, National Bureau of Standards, Dover, New York, 1974.
5. http://en.wikipedia.org/wiki/Orthogonal_polynomials
6. http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html
7. http://mathworld.wolfram.com/ChebyshevPolynomialoftheSecondKind.html
8. E. Kilic, D.Tasci, Factorization and representations of the backward second-order linear recurrences, J. Comput. Appl. Math., pp. 182-197, 2007.