FORMAL KUVVET SERİLERİ İÇİN BİR SEMBOLİK ALGORİTMA

Göksal BİLGİCİ

YÜKSEK LİSANS TEZİ

MATEMATİK EĞİTİMİ

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Aralık 1999

ANKARA

Göksal BİLGİCİ tarafından hazırlanan "FORMAL KUVVET SERİLERİ İÇİN BİR SEMBOLİK ALGORİTMA" adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Prof. Dr. Şeref MİRASYEDİOĞLU

Tez Yöneticisi

Bu çalışma, jurimiz tarafından Matematik Eğitimi Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan : Prof. Dr. Aydın TİRYAKİ

Üye : Prof. Dr. Şeref MİRASYEDİOĞLU

Üye : Doç. Dr. Ziya ARGÜN

Üye :

Üye :

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

İÇİNDEKİLER

FORMAL KUVVET SERİLERİ İÇİN BİR SEMBOLİK ALGORİTMA

(Yüksek Lisans Tezi)
Göksal BİLGİCİ
GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Ekim- 1999

ÖZET

Bu tezde, KOEPF yöntemine dayalı, $\sum_{k=0}^{\infty} a_k (x - x_0)^k$ formunda kuvvet serilerinin sembolik hesabı için bir algoritmik yöntem araştırılmıştır. Geliştirilen algoritmanın MAPLE'da program kodu verilmiştir.

Bilim Kodu : 500.10.78
Anahtar Kelimeler : Kuvvet Serileri, Chebyshev polinomları, Maple
Sayfa Adedi : 46
Tez Yöneticisi : Prof. Dr. Şeref MİRASYEDİOĞLU

A SYMBOLİC ALGORİTHM FOR FORMAL POWER SERİES

(Master Thesis)

Göksal BİLGİCİ

GAZİ UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

December-1999

ABSTRACT

In this thesis. we investigated a symbolic algorithm to compute formal power series of the form $\sum_{k=0}^{\infty} a_k (x - x_0)^k$ using KOEPF's method. Then the algorithm has been implemented in MAPLE.

Science Code : 500.10.78
Key Words : Power series. Chebyshev polinominals. Maple
Page Number : 46
Adviser : Prof.Dr. Şeref MİRASYEDİOĞLU

TEŞEKKÜR

Bu tez de değerli tecrübelerini benden esirgemeyen hocam ve danışmanım Sayın Prof. Dr. Şeref MİRASYEDİOĞLU'na, her konuda yardım ve desteklerini gördüğüm Sayın Prof. Dr. Aydın TİRYAKİ ve Sayın Doç. Dr. Ziya ARGÜN'e, değerli hocalarım ve arkadaşlarım Arş.Gör. Onur KIYMAZ ve Arş.Gör. Tolga GÜYER'e teşekkürlerimi sunarım.

Göksal BİLGİCİ

KISALTMALAR

DD	: Diferensiyel Denklem
TE	: Tekrarlama Eşitliği
FKS	: Formal Kuvvet Serisi
FLS	: Formal Laurent Serisi
BCS	: Bilgisayar Cebir Sistemi

1. GİRİŞ

Sembolik hesaplama, matematiksel problemlerin bilgisayar ortamında analitik çözümlere yönelik işlem yapılabilmesini sağlayan yöntemleri kapsar. Bu yöntemlere yönelik günümüze dek çeşitli bilgisayar cebir sistemleri geliştirilmiştir. [1],[2],[6],[14],[15]. Ayrıca ilk kez kuvvet serilerinin hesabı için Koepf (1992) tarafından algoritmik bir yöntem verilmiştir.

Bu tezde, önce, kuvvet serilerinin hesabı için sembolik bir algoritma araştırılmıştır. Daha sonra, formal kuvvet serilerinin sembolik hesaplama tekniği ile çözümleri gerçekleştirilmiştir

Bu amaçla:

i. Hipergeometrik fonksiyonlar ve rasyonel fonksiyonlar incelenmiştir.

ii. Homojen lineer diferansiyel denklemlerin tekrarlama eşitliklerini (TE) elde etme yöntemleri araştırılmıştır.

iii. Wolfram KOEPF tarafından geliştirilen algoritmaya dayalı, formal kuvvet serilerinin hesabı için sembolik algoritma sunulmuştur ve MAPLE'da simülasyonu yapılmıştır.

Ele alınan konular, dört bölüm altında incelenmiştir.

Birinci bölüm giriştir.

İkinci bölümde, temel tanım ve teoremler sunulmuştur.

Üçüncü bölümde, algoritma ve uygulamaları sunulmuştur.

Dördüncü bölümde, ele alınan konuyla ilgili sonuçlar verilmiştir.

Ekler bölümünde ise kuvvet serilerinin hesabı için geliştirilen algoritmanın MAPLE program kodu verilmiştir.

2. TEMEL KAVRAMLAR

2.1 Temel Tanım ve Teoremler

2.1.1. Seriler

Tanım 2.1.1.1. f: $IN^+ \to IR$ şeklinde tanımlı fonksiyonlara dizi denir.

Tanım 2.1.1.2. (a_n) dizisi verilmiş olsun. Genel terimi

$$S_n=a_1+a_2+...+a_n$$

şeklinde tanımlanan (S_n) dizisini göz önüne alalım. $((a_n),(S_n))$ ikilisine seri adı verilir. a_n terimine serinin genel terimi, (S_n) dizisine de serinin kısmi toplamlar dizisi denir. Eğer (S_n) dizisi bir s sayısına yakınsak ise yani,

$$\lim S_n=s$$

ise $((a_n),(S_n))$ serisi yakınsak ve toplamı s'dir denir. Yakınsak olmayan seriye ıraksak seri adı verilir. Genel terimi a_n olan bir yakınsak serinin toplamı s ise bu

$$\sum_{n=1}^{\infty} a_n = s$$

şeklinde gösterilir. $\sum a_k$ sembolü yakınsak olan bir serinin toplamını göstermektedir. Fakat alışılmış olduğu için seri ister yakınsak ister ıraksak olsun, genel terimi a_n olan seri $\sum_{n=1}^{\infty} a_n$ ile gösterilir.

Teorem 2.1.1.3. $\sum_{n=1}^{\infty} a_n$ serisi yakınsak ise $\lim_{n\to\infty} a_n = 0$ dır. $\lim_{n\to\infty} a_n \neq 0$ ise seri ıraksaktır [4].

Tanım 2.1.1.4. $\sum_{n=1}^{\infty} a.k^{n-1} = a + ak + ak^2 + ... + ak^{n-1} + ...$

biçiminde verilen seriye geometrik seri denir. Serinin

$$S_n = a(1+k+k^2+...+k^{n-1})$$

n. kısmi toplamı,

$$1-k^n = (1-k)(1+k+...+k^{n-1})$$

özdeşliğinden $k\neq 1$ iken

$$S_n = \frac{a(1-k^n)}{1-k}$$

dir.

Teorem 2.1.1.5. $\sum_{n=1}^{\infty} a.k^{n-1}$ serisi

1. $|k|<1$ (yani $-1<k<1$) ise yakınsaktır ve toplamı

$$s = \frac{a}{1-k}$$

sayısıdır.

2. $|k|\geq 1$ ise ıraksaktır [4].

Teorem 2.1.1.6.

(a) $\sum_{n=1}^{\infty} a_n$ serisi yakınsak ve toplamı s ise, c bir sabit olmak üzere $\sum_{n=1}^{\infty} ca_n$ serisi de yakınsaktır ve toplamı c.s dir

(b) $c\neq 0$ olmak üzere,

$\sum_{n=1}^{\infty} a_n$ serisi ıraksak ise $\sum_{n=1}^{\infty} ca_n$ serisi de ıraksaktır [4].

Teorem 2.1.1.7. $\sum_{n=1}^{\infty} a_n = S_1$, $\sum_{n=1}^{\infty} b_n = S_2$ ise

$$\sum_{n=1}^{\infty}(a_n \pm b_n) = S_1 \pm S_2$$

dir [4].

Teorem 2.1.1.8.

(a) $\sum_{n=1}^{\infty} a_n$ serisi, terimleri pozitif olan bir seri, ayrıca $\sum_{n=1}^{\infty} b_n$ serisi de pozitif terimli ve yakınsak bir diğer seri olsun. $\forall n \in IN^+$ için $a_n \leq b_n$ ise $\sum_{n=1}^{\infty} a_n$ serisi de yakınsaktır.

(b) $\sum_{n=1}^{\infty} c_n$ pozitif terimli serisi ıraksak ve $\forall n \in IN^+$ için $a_n \geq c_n$ ise $\sum_{n=1}^{\infty} a_n$ serisi de ıraksaktır [4].

Teorem 2.1.1.9. $\sum_{n=1}^{\infty} a_n$, $\sum_{n=1}^{\infty} b_n$ pozitif terimli iki seri olsun.

(a) $\lim_{n\to\infty} \frac{a_n}{b_n} = c \quad (c \neq 0)$ ise ikisi de ıraksaktır veya ikisi de yakınsaktır.

(b) $\lim_{n\to\infty} \frac{a_n}{b_n} = 0$ ise $\sum_{n=1}^{\infty} b_n$ yakınsak ise $\sum_{n=1}^{\infty} a_n$ serisi de yakınsaktır.

(c) $\lim_{n\to\infty} \frac{a_n}{b_n} = \infty$ ise $\sum_{n=1}^{\infty} b_n$ ıraksak ise $\sum_{n=1}^{\infty} a_n$ serisi de ıraksaktır [4].

Teorem 2.1.1.10. p sabit bir sayı olmak üzere,

$$\sum_{n=1}^{\infty} \frac{1}{n^p} = \frac{1}{1^p} + \frac{1}{2^p} + \ldots + \frac{1}{n^p} + \ldots \quad \text{serisi,}$$

$p>1$ ise yakınsaktır,

$p \leq 1$ ise ıraksaktır [4].

Teorem 2.1.1.11. f fonksiyonu $x \geq 1$ için pozitif değerli, sürekli ve azalan bir fonksiyon olmak üzere

$$\sum_{n=1}^{\infty} f(n) = f(1) + f(2) + ... + f(n) + ...$$

serisi

$$\int_{1}^{\infty} f(x)dx$$

integrali yakınsak ise yakınsaktır.

$$\lim_{b \to \infty} \int_{1}^{b} f(x)dx = \infty$$

ise ıraksaktır [4].

Tanım 2.1.1.12. $\sum_{k=1}^{\infty} |a_k|$ serisi yakınsak ise $\sum_{k=1}^{\infty} a_k$ serisine mutlak yakınsak denir. [4]

Tanım 2.1.1.13. $\sum_{n=1}^{\infty} a_n$ serisinde, $\forall n \in \mathbb{N}^+$ için $S_n < N$ olacak şekilde bir $N \in \mathbb{R}$ varsa bu seriye üstten sınırlı denir.

Teorem 2.1.1.14. Pozitif terimli bir serinin kısmi toplamlar dizisi üstten sınırlı ise yakınsaktır [4].

Teorem 2.1.1.15. Terimleri sıfırdan farklı olan $\sum_{n=1}^{\infty} a_n$ serisi için,

$$\lim_{n \to \infty} \left| \frac{a_{n-1}}{a_n} \right| = \lambda$$

ise, $0 \leq \lambda < 1$ ise seri mutlak yakınsaktır, $\lambda > 1$ ya da $\lambda = \infty$ ise seri ıraksaktır. $\lambda = 1$ ise seri yakınsak da ıraksak da olabilir [4].

Bu teoreme oran (D'alambert kriteri)kriteri denir.

Teorem 2.1.1.16. $\sum_{n=1}^{\infty} a_n$ serisi için

$$\lim_{n\to\infty} \sqrt[n]{|a_n|} = \lambda$$

ise, $0 \le \lambda < 1$ ise seri mutlak yakınsaktır, $\lambda > 1$ ya da $\lambda = \infty$ ise seri ıraksaktır. $\lambda = 1$ ise seri yakınsak da ıraksak da olabilir [4].

Bu teoreme kök (Cauchy kök) kriteri denir.

Tanım 2.1.1.17. $\sum_{n=1}^{\infty} (-1)^n a_n$ biçimindeki bir seriye $(\forall a_n > 0)$ alterne seri denir. Bir alterne seride, $\forall n \in IN^+$ için

$$|a_{n+1}| < |a_n| \text{ ve } \lim_{n\to\infty} a_n = 0$$

ise bu seri yakınsaktır [4].

2.1.2 Kuvvet Serileri

Tanım 2.1.2.1. $\sum_{j=0}^{\infty} a_j (z - z_0)^j$ şeklindeki bir seri formu, bir kuvvet serisi olarak adlandırılır. Burada a_j sabitlerine kuvvet serisinin katsayıları denir.

Teorem 2.1.2.2. Her $\sum_{j=0}^{\infty} a_j (z - z_0)^j$ kuvvet serisi için aşağıdaki koşulları sağlayan bir $R \in [0, \infty)$ reel sayı vardır

(i) $|z - z_0| < R$ diski içinde seri yakınsaktır.

(ii) $|z - z_0| \le R' < R$ şeklindeki herhangi bir kapalı diskinde seri üniform yakınsaktır.

(iii) $|z - z_0| > R$ için ise seri ıraksaktır [7].

Bu R reel sayısına kuvvet serisinin yakınsaklık yarıçapı denir. R = 0 için kuvvet serisi sadece $z = z_0$ noktasında yakınsaktır. R = ∞ ise her $z \in C$ için kuvvet serileri yakınsaktır. $0 < R < \infty$ için $|z - z_0| = R$ çemberine yakınsaklık çemberi denir. Eğer z noktası bu çemberin üzerinde ise genel bir yakınsaklık ifadesi oluşturulamaz.

Lemma 2.1.2.3. $\sum_{j=0}^{\infty} a_j z^j$ kuvvet serisi r çapına sahip bir noktada yakınsak ise $|z| < r$ diskinin içindeki her noktada yakınsaktır [7].

Teorem 2.1.2.4. Bir kuvvet serisi, yakınsaklık çemberinin her noktasında analitik olan bir f fonksiyonuna denk gelir [7].

Teorem 2.1.2.5. $\sum_{j=0}^{\infty} a_j (z - z_0)^j$, z_0'ın bazı komşuluklarında f(z) noktasına yakınsıyorsa (yani yakınsaklık çemberinin yarıçapı sıfırdan farklı ise);

$$a_j = \frac{f^{(j)}(z_0)}{j!} \quad (j = 0, 1, 2, \ldots)$$

olur. Yani,

$$\sum_{j=0}^{\infty} a_j (z - z_0)^j .$$

serisi, z_0 civarında f(z) nin Taylor serisidir [7].

Teorem 2.1.2.6. $\sum_{n=0}^{\infty} c_n x^n$ serisi $x=x_0$ için yakınsak ise $|x| < |x_0|$ koşulunu sağlayan tüm x'ler için mutlak yakınsaktır. Aynı seri $x=x_0$ için ıraksak ise $|x| > |x_0|$ koşulunu sağlayan tüm x'ler için ıraksaktır [7].

Teorem 2.1.2.7. $\sum_{n=0}^{\infty} c_n x^n$ serisi R yarıçaplı aralıkta yakınsak ise, $\sum_{n=1}^{\infty} n c_n x^{n-1}$ serisi de R yarıçaplı aralıkta yakınsaktır [7].

Teorem 2.1.2.8. $\sum_{n=0}^{\infty} c_n x^n$ serisi [-d.d] aralığında yakınsak ve fonksiyonu F(x)= $\sum_{n=0}^{\infty} c_n x^n$ şeklinde tanımlanmış olsun. O zaman $F'(x)$. $\forall x(-d,d)$ için tanımlıdır ve

$$F'(x)=\sum_{n=1}^{\infty} n c_n x^{n-1}$$

dir [4].

2.1.3 Laurent Serileri

Teorem 2.1.3.1. f fonksiyonu bir R bölgesinde analitik ve Γ_0 ve Γ_1 iki eğri olsun. Bu durumda,

$$\int_{\Gamma_0} f(z)dz = \int_{\Gamma_1} f(z)dz \qquad (2.1)$$

dir [12].

z_0 kompleks düzlemde bir nokta ve f fonksiyonu $0 \le \alpha < \beta \le \infty$ ve f, A:$\alpha < |z - z_0| \le \beta$ halkasında analitik olsun. $\alpha<\rho<\beta$ için Γ_ρ, $z=z_0+\rho^{i\tau}$ $(0\le \tau \le 2\pi)$ çemberini göstersin.

z, A'nın bir noktası olsun.

$$g(t) = \begin{cases} \dfrac{f(t)-f(z)}{t-z}, & t \neq z, \\ f\ (z), & t = z, \end{cases}$$

şeklinde tanımlanan g fonksiyonu A'da analitiktir. Teorem 2.1.3.1'e göre

$$\int_{\Gamma} g(t)dt$$

integrali ρ'dan bağımsızdır. Özel olarak $\alpha < \rho_1 < |z - z_0| < \rho_2 < \beta$ ise $\Gamma_{\rho_1} =: \Gamma'$, $\Gamma_{\rho_2} =: \Gamma''$ alınırsa, bu durumda

$$\int_{\Gamma'} \frac{f(t)-f(z)}{t-z} dt = \int_{\Gamma''} \frac{f(t)-f(z)}{t-z} dt$$

veya f(z) bir sabit olduğundan,

$$f(z)\left\{ \int_{\Gamma'} \frac{1}{t-z} dt - \int_{\Gamma''} \frac{1}{t-z} dt \right\} = \int_{\Gamma'} \frac{f(t)}{t-z} dt - \int_{\Gamma''} \frac{f(t)}{t-z} dt$$

olarak yazılabilir. Ancak [12] den

$$\int_{\Gamma'} \frac{1}{t-z} dt = 0 \ , \ \int_{\Gamma''} \frac{1}{t-z} dt = 2\pi i$$

ve

$$f(z) = \frac{1}{2\pi i} \int_{\Gamma'} \frac{f(t)}{t-z} dt - \frac{1}{2\pi i} \int_{\Gamma''} \frac{f(t)}{t-z} dt \qquad (2.2)$$

olur. Bu da f'nin Γ' ve Γ'' çemberleri üzerindeki z noktasındaki değerlerinin bir gösterimidir ve $\rho_1 < |z - z_0| < \rho_2$ şeklindeki bütün z'ler için geçerlidir.

$t \in \Gamma''$ için $|t - z_0| > |z - z_0|$ dır. Bu nedenle h0z-z_0 alınırsa

$$\frac{1}{1-z} = \frac{1}{t-z_0-h} = \frac{1}{t-z_0} \frac{1}{1-h/(t-z_0)} = \sum_{n=0}^{\infty} \frac{h^n}{(t-z_0)^{n+1}}.$$

$t \in \Gamma'$ için, $|t - z_0| < |z - z_0|$ olur. Böylece

$$\frac{1}{1-z} = -\frac{1}{h-(t-z_0)} = -\frac{1}{h} \frac{1}{1-(t-z_0)/h} = -\sum_{n=0}^{\infty} \frac{(1-z_0)^n}{h^{n+1}}$$

$t \in \Gamma'$ ve $t \in \Gamma''$ için bu serilerin üniform yakınsaklığından dolayı, bunları integral(2.2) ile değiştirebiliriz ve terim terim integral alabiliriz.

$$a_n = \frac{1}{2\pi i}\int_{\Gamma}\frac{f(t)}{(t-z_0)^{n+1}}dt, \quad n = 0.1.2$$

$$a_{-n} = \frac{1}{2\pi i}\int_{\Gamma} f(t)(t-z_0)^n, \quad n = 1.2.....$$

Böylece, sonuç aşağıdaki şekilde basit bir formda yazılabilir:

$$f(z) = \sum_{-\infty}^{\infty} a_n h^n, \quad h=z-z_0 \tag{2.3}$$

Katsayılar için formül, $f(t)(t-z_0)^k$ fonksiyonlarının bütün k tamsayıları için A'da analitik olduğu gösterilerek sadeleştirilebilir. ρ, $\alpha<\rho<\beta$ şeklinde herhangi bir sayı olsun. Her $n\in Z$ için teorem 2.1.5 den katsayıların tanımladığı integraller Γ_ρ çemberi boyunca hesaplanabilir ve iki formül tek formül olarak birleştirilebilir:

$$a_n = \frac{1}{2\pi i}\int_{\Gamma_\rho}\frac{f(t)}{(t-z_0)^{n+1}}dt, \quad n = 0.\pm1.\pm2...., \tag{2.4}$$

α_1 ve β_1, $\alpha < \alpha_1 < \beta_1 < \beta$ şeklinde iki sayı olsun. (2.3) serisi $\alpha_1 \le |h| \le \beta_1$ için düzgün yakınsaktır. Seriyi iki kuvvet serisinin toplamı olarak yazabiliriz:

$$\sum_{n=-\infty}^{\infty} a_n h^n = \sum_{n=0}^{\infty} a_n h^n + \sum_{n=1}^{\infty} a_{-n} h^{-n}$$

Sağdaki ilk seri, β yakınsaklık yarıçapı ile h içinde bir kuvvet serisidir: bu nedenle $|h| \le \beta_1$ için üniform yakınsaktır. İkinci seri, $|h^{-1}| \le \alpha_1^{-1}$ için, α^{-1} yakınsaklık yarıçapıyla h^{-1} içinde üniform yakınsayan bir kuvvet serisidir.

Teorem 2.1.3.2. z_0 kompleks bir sayı, $0 \le \alpha < \beta \le \infty$ ve f fonksiyonu A: $\alpha < |z-z_0| < \beta$ halkası içinde analitik olsun. a_n katsayıları (2.4) şeklinde tanımlanmış ise her $z \in A$ için f fonksiyonu, (2.3) şeklinde bir seriyle gösterilebilir. Seri A'nın her kompakt alt kümesinde üniform yakınsaktır [12].

(2.3) serisi Laurent genişlemesi ya da Laurent serisi olarak adlandırılır.

Teorem 2.1.3.3. Verilen bir halkada bir analitik fonksiyonun Laurent serisi tekdir [12].

2.1.4. Chebyshev Polinomları

$n \in IN^+$, $x=\cos(\theta)$ ve $0 \le \theta \le \pi$ olsun.

$$T_n(x) = \text{Cosn}\theta \tag{2.5}$$

fonksiyonunu göz önüne alalım. $0 \le \arccos x \le \pi$ olmak üzere

$$T_n(x) = \cos n(\arccos x)) \tag{2.6}$$

şeklinde yazılabilir.

$$e^{i\theta} = \cos\theta + i\sin\theta$$

ve

$$e^{in\theta} = (\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta \tag{2.7}$$

olduğunu biliyoruz. Binom açılımından

$$(\cos\theta+i\sin\theta)^n=\cos^n\theta+\binom{n}{1}\cos^{n-1}\theta(i\sin\theta)+\binom{n}{2}\cos^{n-2}\theta(i^2\sin^2\theta)+\ldots+$$

$$+\binom{n}{n}(i\sin\theta)^n$$

yazılır ve

$$\cos n\theta = \cos^n\theta - \binom{n}{2}\cos^{n-2}\theta\sin^2\theta + \binom{n}{4}\cos^{n-4}\theta\sin^4\theta + \ldots + (-1)^{[n/2]}\binom{n}{2[n/2]}\cos^{n-2[n/2]}\theta\sin^{2[n/2]}\theta \tag{2.8}$$

elde edilir. Burada $\sin^2\theta = 1-\cos^2\theta$ trigonometrik özdeşliği kullanılırsa

$$\cos n\theta = \sum_{q=0}^{[n/2]}(-1)^q\binom{n}{2q}\cos^{n-2q}\theta\left(\sum_{k=0}^{q}(-1)^k\binom{q}{k}\cos^{2k}\theta\right) \qquad (2.9)$$

bulunur. (2.9) un sağ tarafı $x=\cos\theta$ için bir polinom olup, $T_n(x)$ fonksiyonu da bir polinomdur. Ayrıca (2.9) un sağ tarafı üçgen toplamdır. Eğer

$$A_q = (-1)^q\binom{n}{2q}\cos^{n-2q}\theta, \quad q = 0,\ldots,\left[\frac{n}{2}\right]$$

ve

$$B_{k,q} = (-1)^k\binom{q}{k}\cos^{2k}\theta, \quad k = 0,1,\ldots,q,$$

denirse

$$\begin{aligned}\cos n\theta = {} & A_0B_{0,0} \\ & +A_1B_{0,1}+A_1B_{1,1} \\ & +A_2B_{0,2}+A_2B_{1,2}A_2B_{2,2} \\ & + \\ & \vdots \\ & + \\ & +(A_{[n/2]-1}B_{0,[n/2]-1}A_{[n/2]}B_{1,[n/2]}) \\ & +A_{[n/2]}B_{0,[n/2]}; \end{aligned} \qquad (2.10)$$

bulunur. (ardışık diagonalleri eleyerek) (2.10) eşitliğinin sağ tarafı toplanırsa

$$\begin{aligned}\cos n\theta = {} & (A_0B_{0,0}+A_1B_{1,1}+\ldots+A_{[n/2]}B_{[n/2],[n/2]}) \\ & +(A_1B_{0,1}+A_2B_{1,2}+\ldots+ A_{[n/2]}B_{[n/2]-1,[n/2]}) \\ & -\end{aligned}$$

$$\vdots$$

$$+(A_{[n/2]-1}B_{0.[n/2]-1}+A_{[n/2]}B_{1.[n/2]})$$

$$+A_{[n/2]}B_{0.[n/2]}$$

ve burada A_q ve $B_{k.q}$ terimleri yerine yazılırsa

$$\cos n\theta = \sum_{k=0}^{[n/2]}\left((-1)^k \sum_{j=k}^{[n/2]}\binom{n}{2j}\binom{j}{k}\right)\cos^{n-2k}\theta \qquad (2.11)$$

elde edilir.

(2.11) eşitliği. $T_n(x)$ fonksiyonun n. dereceden bir polinom olduğunu gösterir. Böylece.

$$T_n(x) = t_0^{(n)} + t_1^{(n)}x + \ldots + t_n^{(n)}x^n \qquad (2.12)$$

yazılabilir. Burada

$$\begin{aligned} t_{n-(2k+1)}^{(n)} &= 0, \qquad k = 0,\ldots,\left[\frac{n-1}{2}\right] \\ t_{n-2k}^{(n)} &= (-1)^k \sum_{j=k}^{[n/2]}\binom{n}{2j}\binom{j}{k}, \quad k = 0,\ldots,\left[\frac{n}{2}\right] \end{aligned} \qquad (2.13)$$

dır. $T_n(x)$ fonksiyonuna n. Dereceden Chebyshev polinomu denir [6]. Özel olarak

$$T_0(x) = 1; \quad T_1(x) = x; \quad T_2(x) = 2x^2 - 1; \quad T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1 \qquad T_5(x) = 16x^5 - 20x^3 + 5x \qquad (2.14)$$

şeklindedir.

2.2. Hipergeometrik Tip Seriler

Formal kuvvet serilerini hesaplamak için vereceğimiz metodlarda, genelliği yitirmeksizin $x_0 = 0$ olarak alalım. Bu durumda FKS

$$\sum_{k=0}^{\infty} a_k x^k \quad a_k \in C \ (k \in IN_0) \tag{2.15}$$

şeklinde yazılır.

Bir F FKS `nin F' türevi

$$F'(x) = \sum_{k=0}^{\infty} k a_k x^{k-1} = \sum_{k=0}^{\infty} (k+1) a_{k+1} x^k \tag{2.16}$$

ve F FKS`nin integrali

$$\int F(x)dx = \sum_{k=0}^{\infty} \frac{a_k}{k+1} x^{k+1} = \sum_{k=1}^{\infty} \frac{a_{k-1}}{k} x^k \tag{2.17}$$

ile verilir.

FKS için, toplama, çarpma, bölme ve yer değiştirme gibi bütün cebirsel işlemler, yalnızca ilk N katsayıları değerlendirilirse ($N \in IN^-$), sonlu algoritmalarla yapılabilir. Bu algoritmalar belirli Bilgisayar Cebir Sistemleri (BCS) içinde uygulanabilir, örneğin MAPLE`da.

Eğer a_k katsayıları için açık formülle ilgileniliyorsa, çoğu cebirsel işlemler, sonlu bir algoritmayla çok zor yapılır.

Algoritmik olarak BCS`de FKS`nin kullanımını gerçekleştirmek ve bazı özel türlerin FKS`ni hesaplamak için bir yöntem verilecektir. Bu FKS tanım kümesi hesap yapmak için yeterince zengindir. Fakat diğer taraftan da bir BCS`de yerine getirilmesi sınırlı olabilir. Sınıfımız pek çok özel fonksiyon içerir, fakat fonksiyonların lineer kombinasyonlarının FKS`sini hesaplamak için verilecek yöntem yeterli değildir.

Bazı genel fonksiyonların formal kuvvet serileri cinsinden karşılıkları

$$\binom{\alpha}{k}=\begin{cases}1 & k=0\\ \dfrac{\alpha(\alpha-1)...(\alpha-k+1)}{k!} & k\in IN\end{cases}$$

binom katsayıları.

$$k!=\begin{cases}1 & k=0\\ 1.2...k & k\in IN^{*}\end{cases}$$

$$Bn=\begin{cases}1 & n=0\\ -\dfrac{1}{n+1}\sum_{k=0}^{n-1}\binom{n+1}{k}B_k & n\in IN\end{cases} \qquad (2.18)$$

Bernoulli sayılarını göstermek üzere

$$\sum_{k=0}^{\infty}\binom{\alpha}{k}x^k \leftrightarrow (1+x)^{\alpha}$$

binom serisi

$$\sum_{k=0}^{\infty}\frac{1}{k!}x^k \leftrightarrow \exp(x)$$

$$\sum_{k=0}^{\infty}\frac{1}{k}x^k \leftrightarrow -\ln(1-x)$$

$$\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)!}x^{2k+1} \leftrightarrow \sin(x)$$

$$\sum_{k=0}^{\infty}\frac{1}{(2k)!}x^{2k} \leftrightarrow \cos(x)$$

$$\sum_{k=0}^{\infty}\frac{1}{(2k+1)!}x^{2k+1} \leftrightarrow \sinh(x)$$

$$\sum_{k=0}^{\infty}\frac{(-1)}{(2k)!}x^{2k} \leftrightarrow \cosh(x)$$

$$\sum_{k=0}^{\infty}\frac{(-1)^{k-1}4^k(4^k-1)B_{2k}}{(2k)!}x^{2k-1}\leftrightarrow\tan(x)$$

$$\sum_{k=0}^{\infty}\frac{(-1)^k4^kB_{2k}}{(2k)!}x^{2k}\leftrightarrow x\,.\,\cot(x)$$

$$\sum_{k=0}^{\infty}\frac{4^k(4^k-1)B_{2k}}{(2k)!}x^{2k}\leftrightarrow\tanh(x)$$

$$\sum_{k=0}^{\infty}\frac{4^kB_{2k}}{(2k)!}x^{2k}\leftrightarrow x\,.\,\coth(x)$$

$$x+\sum_{k=1}^{\infty}\frac{1.3.5...(2k-1)}{2^kk!(2k+1)}x^{2k+1}\leftrightarrow\arcsin(x)$$

$$\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}x^{2k+1}\leftrightarrow\arctan(x)$$

$$x+\sum_{k=1}^{\infty}\frac{(-)^k1.3.5...(2k-1)}{2^kk!(2k+1)}x^{2k+1}\leftrightarrow\operatorname{arsinh}(x)$$

$$\sum_{k=0}^{\infty}\frac{1}{2k+1}x^{2k+1}\leftrightarrow\operatorname{artanh}(x)$$

şeklindedir. FKS için daha genel bir düzenlemede bulunmak için hipergeometrik serilerden söz edeceğiz.

Tanım 2.2.1.

$$(a)_k=\begin{cases}1 & k=0\\ a.(a+1)...(a+k-1) & k\in IN\end{cases}\qquad(2.19)$$

(2.19) Pochhammer sembolü (ya da shifted faktöryel) olmak üzere

$${}_pF_q\left(\begin{matrix}a_1\ a_2\ \dots\ a_p\\ b_1\ b_2\ \dots\ b_q\end{matrix}\middle|x\right)=\sum_{k=0}^{\infty}\frac{(a_1)_k.(a_2)_k\dots(a_p)_k}{(b_1)_k.(b_2)\dots(b_q)_k\,k!}x^k\qquad(2.20)$$

ifadesine (genelleştirilmiş) hipergeometrik seri gösterimi denir [7]. Burada $\frac{(a)_k}{k!} = \binom{a+k-1}{k}$ dir.

$\sum_{k=0}^{\infty} A_k x^k$ hipergeometrik serileri, A_k katsayıları, A_0=1 başlangıç koşulu altında

$$A_{k+1} = \frac{(k+a_1).(k+a_2)...(k+a_p)}{(k+b_1).(k+b_2)...(k+b_q)(k+1)} . A_k \quad (k \in IN) \qquad (2.21)$$

özel tekrarlama eşitliğinin (TE) (recurrence equation) tek çözümüdür [7]. Ayrıca $\frac{A_{k+1}}{A_k}$ k'da rasyoneldir.

θ , $x\frac{d}{dx}$ diferensiyel operatörü olmak üzere

$$\theta(\theta + b_1 - 1) \ldots (\theta + b_q - 1)f = x(\theta + a_1) \ldots (\theta + a_p)f \qquad (2.22)$$

diferensiyel denkleminin (DD) çözümü f fonksiyonu, ve f fonksiyonuna karşılık gelen hipergeometrik seri F yani

$$f \leftrightarrow F(x) = {}_pF_q\left(\begin{matrix} a_1\ a_2 \ldots a_p \\ b_1\ b_2 \ldots b_q \end{matrix}\middle| x\right)$$

olsun. Bu halde bu diferensiyel denklem

$$\sum_{k=0}^{Q}\sum_{j=0}^{Q} c_{jk} x^j f^{(k)} = 0 \qquad c_{jk} \in C \ (j, k = 0, \ldots, Q) \qquad (2.23)$$

şeklindedir (Q=max {p, q} + 1) ve bu denklem homojen lineer diferensiyel denklemdir. Bazı fonksiyonların kuvvet serilerinin hipergeometrik serilerle gösterimi

$$B_{\alpha}(x) = {}_1F_0(-\alpha|x).$$

$$\exp(x) = {}_0F_0(x).$$

$$\ln x = x.\ {}_2F_1\left(\begin{matrix}1 & 1\\ & 2\end{matrix}\middle| x\right).$$

$$\sin x = x.\ {}_0F_1\left(3/2\middle| -\frac{x^2}{4}\right).$$

$$\cos x = {}_0F_1\left(1/2\middle| -\frac{x^2}{4}\right).$$

$$\sinh(x) = x.\ {}_0F_1\left(3/2\middle| \frac{x^2}{4}\right).$$

$$\cosh(x) = {}_0F_1\left(1/2\middle| \frac{x^2}{4}\right).$$

$$\arcsin x = x.\ {}_2F_1\left(\begin{matrix}1/2 & 1/2\\ & 3/2\end{matrix}\middle| x^2\right).$$

$$\arctan x = x.\ {}_2F_1\left(\begin{matrix}1/2 & 1\\ & 3/2\end{matrix}\middle| -x^2\right).$$

$$\operatorname{arcsin\,h}(x) = x.\ {}_2F_1\left(\begin{matrix}1/2 & 1/2\\ & 3/2\end{matrix}\middle| x^2\right).$$

$$\operatorname{arctan\,h}(x) = x.\ {}_2F_1\left(\begin{matrix}1/2 & 1\\ & 3/2\end{matrix}\middle| x^2\right)$$

biçimindedir. $f(x^m)$ şeklindeki bir fonksiyona ve dolayısıyla $F(x^m)$ biçimindeki bir FKS`ne m-katlı simetrik denir. Çift fonksiyonlar 2-katlı simetrik ve tek fonksiyonlar ise 2-katlı simetrik fonksiyonlara dönüştürülebilirler.

Formal Laurent serilerinin;

$$F(x)= \sum_{k=k_0}^{\infty} a_k x^k \qquad (a_{k_0} \neq 0) \tag{2.24}$$

şeklinde gösterildiğini belirtmiştik.

Tanım 2.2.2. (2.24) deki gösterimiyle bir FLS` ye (hem de benzeri f fonksiyonuna) eğer yakınsama yarıçapı pozitif ve a_k katsayıları bir $m \in IN$, $A_k \in C$ $(k = k_0 + 1, k_0 + 2, \ldots, k_0 + m - 1)$, $A_{k_0} \in C\setminus\{0\}$ şeklinde ise ve bir R rasyonel fonksiyonu için

$$\begin{aligned} a_{k+m} &= R(k)a_k && k \geq k_0 \\ a_k &= A_k && k = k_0, k_0+1, \ldots, k_0+m-1 \end{aligned} \tag{2.25}$$

şeklindeki bir TE`ye uyuyorsa hipergeometrik tip denir. (2.25) tipi hipergeometrik tip olarak isimlendirilir. $m \in IN$ doğal sayısına simetri sayısı denir [8].

Sinx fonksiyonunun, m=1 olacak şekilde (2.25)`deki gibi bir TE`si olmadığından bir genelleştirilmiş hipergeometrik fonksiyon değildir. Fakat m=2 simetri sayısı için bir hipergeometrik tiptir. $e^{\arcsin x}$ fonksiyonu 2 simetri sayılı hipergeometrik tipte bir fonksiyon olmasına rağmen ne tek ne de çift fonksiyondur.

Bir f fonksiyonu orjinde hipergeometrik tip ise bir diğer $x_0 \neq 0$ noktasına göre hipergeometrik tip olması gerekmez yani $f(x-x_0)$ bir hipergeometrik tip olmayabilir. Bu duruma bir örnek, hipergeometrik tip olan $\frac{\sin x}{x}$

fonksiyonudur. Bununla birlikte $\frac{\sin(x - x_0)}{x - x_0}$ fonksiyonu $x_0 \neq 0$ için hipergeometrik tip değildir.

Eğer f fonksiyonu, m-katlı simetrik ise, m simetri sayılı bir (2.25) hipergeometrik gösterimi vardır, fakat böyle bir gösterim herhangi bir simetriyi garanti etmez. f fonksiyonu, j simetri sayılı bir hipergeometrik tip ise, tümevarım yoluyla

$$a_{k+jm} = R(k+j)R(k+j)...R(k+(m-)j)a_k$$

ve $R(k)R(k+j)R(k+2j)...R(k+(m-1)j)$ nin rasyonel olduğu gösterilebilir. Bundan dolayı f fonksiyonu, her m için mj simetri sayılı hipergeometrik tiptir. j=1 simetri sayılı her hipergeometrik tip fonksiyon herhangi bir m simetri sayısı için de hipergeometrik tiptir.

Bir FLS`nin türevi

$$F(x) = \sum_{k=k_0-1}^{\infty} (k+1)a_{k+1}x^k \qquad (2.26)$$

şeklindedir.

Lemma 2.2.3 F. (2.24) gösterimli hipergeometrik tipte bir FLS olsun. Bu durumda

(a) xF .

(b) F/x .

(c) $F(Ax)\,(A \in C)$.

(d) $F(x^n)\,(n \in IN)$.

(e) $\frac{F(x) \pm F(-x)}{2}$.

(f) F .

serileri de hipergeometrik tiptir.

Eğer F(x), m simetri sayısına sahip ise $F(x^n)$ n.m simetri sayısına ve $\frac{F(x)\pm F(-x)}{2}$ ise 2m simetri sayısına sahiptir.

Eğer F serisi, bir FKS ise bu durumda

(g) $\int F$ `te hipergeometrik tiptir [8].

İspat: F FKS`nin katsayıları için TE (2.25) nin geçerli olduğunu kabul edelim

(a) ve (b) yer değiştirme (shift) işlemleridir. xF serisi, b_k 0 a_{k-1} katsayılarına sahiptir ve bu nedenle TE $b_{k+m} = a_{(k-1)+m} = R(k-1) = R(k-1)b_k$ sabittir. R, k değişkeni için rasyonel olduğundan R(k-1) de rasyoneldir ve xF serisi hipergeometrik tiptir.

(b) Benzer şekilde $b_{k+m} = R(k+1)b_k$ şeklindedir.

(c) F(Ax), $b_k=A^k a_k$ katsayılarına sahiptir. Bu nedenle TE

$$b_{k+m} = A^{k+m} a_{k+m} = R(k)A^m b_k \text{ dir.}$$

(d) $F(x^n)$ in b_k katsayıları, b_{nk} 0 a_k şeklindedir ve dolayısıyla

$$b_{nk+nm} = b_{n(k+m)} = a_{k+m} = R(k)a_k = R(k)b_{nk} \text{ dir}$$

$b_{k+nm} = R(k/n)b_k$ olduğundan. $F(x^n)$ hipergeometrik tiptir.

(e) $G=\frac{F(x)+F(-x)}{2}$. F serisinin çift kısmını temsil etmektedir. Bundan dolayı tek indisli katsayıları sıfırdır; b_k çift indisli katsayıları F serisinin katsayılarına eşittir, yani $b_k = a_k$ dır. Bundan dolayı bir k çift indeksi için

$b_{k+2m} = a_{k+2m} = R(k+m)a_{k+m} = R(k+m)\,R(k)a_k = R(k+m)\,R(k)b_k$ dir.

Bu formül tek katsayılar için de sağlanır ve bu nedenle G serisi, 2m simetri sayılı hipergeometrik tiptedir. FLS $\frac{F(x)-F(-x)}{2}$, F serisinin tek kısmını temsil ettiğinden benzer bir yaklaşım bu durumda da uygulanabilir.

(f) (2.26) deki tanımıyla F' serisi, $b_k=(k+1)a_{k+1}$ katsayılarına sahiptir ve bundan dolayı hipergeometrik tip TE

$b_{k+m} = (k+1+m)a_{k+1+m} = (k+1+m)R(k+1)a_{k+1} = \frac{k+1+m}{k+1}R(k+1)b_k$ dir

(g) Burada $b_k=\frac{a_{k-1}}{k}$ ve dolayısıyla $b_{k+m} = \frac{k}{k+m}R(k-1)b_k$ şeklindedir.

Teorem 2.2.4. Hipergeometrik tipte olan her FLS basit bir DD`i (diferensiyel denklem) sağlar [8].

İspat. F serisi

$$F(x) = \sum_{k=k_0}^{\infty} a_k x^k \qquad (a_{k_0} \neq 0)$$

şeklinde verilmiş olsun. θ diferensiyel operatör olmak üzere

$$\theta F(x) = \sum_{k=k_0}^{\infty} k a_k x^k$$

olduğundan bütün $j \in IN$`ler için tümevarım yoluyla

$$\theta^j F(x) = \sum_{k=k_0}^{\infty} k^j a_k x^k$$

bulunur. P herhangi bir polinom olmak üzere yukarıdaki eşitliği

$$P(\theta)F(x) = \sum_{k=k_0}^{\infty} P(k)a_k x^k \qquad (2.27)$$

şeklinde yazabiliriz.

Şimdi, TE (2.25) den, P ve Q bir polinom ise, $k \in IN$ için

$$a_{k+m} = R(k)a_k = \frac{P(k)}{Q(k)}a_k$$

olur. Bunu da

$$Q(k)a_{k+m} = P(k)a_k \qquad (2.28)$$

şeklinde yazılabilir ve genelliği yitirmeksizin P ve Q polinomlarının

$Q(k_0 -1) = Q(k_0 - 2) = \ldots = Q(k_0 - m) = 0$ şeklinde seçilmiş olduğunu kabul edelim (P ve Q` nun her ikisini de $(k - k_0 + j)$ $(j = 1, \ldots, m)$ çarpanlarıyla çarpılarak ulaşılır). Böylece

$Q(\theta-m)F(x) = \sum_{k=k_0}^{\infty} Q(k-m)a_k x^k$ (2.27) dan, Q bir polinom olduğu için

$= \sum_{k=k_0}^{\infty} Q(k-m)a_k x^k$ $Q(k_0 -1) = Q(k_0 - 2) = \ldots = Q(k_0 - m) = 0$

$= \sum_{k=k_0}^{\infty} Q(k)a_{k-m} x^{k+m}$ bir indeks değiştirmesi yoluyla

$= x^m \sum_{k=k_0}^{\infty} P(k)a_k x^k$ (2.28) kullanılarak

$= x^m P(\theta)F(x)$ tekrar (2.27) kullanılarak elde edilir.

Bu, F için (2.23) teki şekilde bir DD`yi ifade etmektedir. $m = 1$ ve $k_0 = 0$ için (2.22) ye bulunur.

$x_0 = 0$ noktasında x değişkenine göre f fonksiyonunun bir Laurent serileri genişleme katsayısını veren bir algoritma sunulacak ve MAPLE dilinde simulasyonu yapılacaktır.

3. KUVVET SERİLERİNİN HESAPLANMASI

Bu bölümde algoritmanın ana hatları verilecektir.

3.1. Algoritma

Algoritma 3.1

(1) *Rasyonel fonksiyonlar* (§3.3`e bakınız)

Eğer f , x`te rasyonel ise, §3.3 *rasyonel algoritmasını* kullanın.

(2) *Basit bir DD bulun* (Ayrıntılar için § 3.4`e bakınız)

(a) DD`nin aranan mertebesi için maksimum bir $N_{maks} \in IN$ sayısı belirleyin; uygun bir değer $N_{maks} = 4$`tür.

(b) N= 1`için uygulayın.

(c) $f^{(N)}$ i hesaplayın: $f^{(N)}$ türevi rasyonel ise §3.3 rasyonel algoritmasını uygulayın ve integrali alın

(d) ya da p_k (k = 1, ... N) nin x değişkeninde polinom olmak üzere

$$\sum_{k=0}^{N} p_k f^{(k)} = 0 \qquad (3.1)$$

f fonksiyonu için N. mertebeden bir basit DD bulun.

(e) Eğer (c) başarılı sonuç vermezse N `yi bir arttırın ve $N = N_{maks}$ oluncaya kadar (c) ye geri dönün

(3) *Uygun TE`yi bulun* (§ 3.5`e bakınız)

İkinci adımda bir basit DD bulduysanız bu DD`yi $a_n$ katsayıları için bir TE`ye aktarın. O zaman TE, r_k`nın n`de rasyonel fonksiyonlar ve $M \in IN$ için

$$a_{n+1} = \sum_{k=0}^{M} r_k a_{n-k} \qquad (3.2)$$

özel tipi olur. Bu da.

$$x^{(j)} f^{(k)} \rightarrow (n + 1 - j)_k \cdot a_{n+k-j} \qquad (3.3)$$

yer değiştirmesiyle yapılır.

(4) *TE`nin tipi* (§3.6`ye bakınız)

Aşağıdaki listeye göre TE`nin tipini belirleyin.

(a) Eğer TE (3.2) de eşitliğin sağ tarafı, $r_j a_{n-j}$ `nin rakamlarından yalnız birinin toplam şeklindeki bir ifadesini içeriyorsa, bu durumda f fonksiyonu hipergeometrik tiptir; m=j+1 simetri sayısına sahiptir. Kuvvet serisinin katsayıları (2.20) hipergeometrik katsayı formülü ve bazı başlangıç koşullarıyla bulunabilir.

(b) Eğer DD,

$$\sum_{k=0}^{Q} c_k f^{(k)} = 0 \,.\, (c_k \in C \; (k = 0, \ldots, Q)$$

sabit katsayılarına sahip ise f fonksiyonu, exp-like tiptir. Bu durumda $b_n=n!.a_n$ eşitliği bizi DD ile aynı sabit katsayılara sahip

$$\sum_{k=0}^{Q} c_k b_{n+k} = 0 \qquad (3.4)$$

TE`sine götürür ve ilk Q başlangıç katsayılarını kullanarak bulunabilir.

(c) Eğer TE yukarıdaki tiplerin hiç biri değilse bunu başka bilinen TE çözücüleri ile çözmeye çalışılır. (Örneğin, Mathematica programında bulunan DiscreteMath paketindeki "Rsolve" komutu gibi)

3.2. Uygulamalar

Bu alt bölümde algoritmanın kullanımına yönelik bazı örnekler verilecektir.

Örnek 3.2.1. $f(x) = e^x \sin x$ olsun, $f'(x)= e^x(\sin x+\cos x)$ ve $f''(x) = 2e^x \cos x$ olur. Algoritmanın ilk adımı uygulanmaz. N01 için ikinci adımda

$A_0=-(1+\cot x)$ ifadesi bulunur ve bu ifade x'te rasyonel değildir. N02 için

$$f''+A_1f'+A_0f = 2e^x\cos x+A_1e^x(\sin x+\cos x)+A_0e^x\sin x$$

ifadesini elde ederiz. Buradaki terimler

$$2e^x\cos x,\ A_1e^x\sin x,\ A_1e^x\cos x,\ A_0e^x\sin x$$

olup fonksiyonel bağımsız iki terim vardır ve bunlar $e^x\cos x$ ve $e^x\sin x$ dır. Bu ifadelerin katsayı değerleri sıfıra eşitlenirse.

$$2+A_1=0\ ,\qquad A_1+A_0=0$$

yazılır ve $A_1=-2, A_0=2$ çözümleri bizi f fonksiyonu için

$$f''-2f'+2f=0 \tag{3.5}$$

DD'ine karşılık getirir. Bu DD sabit katsayılara sahiptir, bu nedenle f fonksiyonu, exp-like tiptedir. $b_n=n!a_n$ için

$$b_{n+2}-2b_{n+1}+2b_n=0$$

TE'sine sahip oluruz. Başlangıç koşulları

$$b_0=a_0=f(0)=0 \quad \text{ve} \quad b_1=a_1=\frac{f'(0)}{1!}=1$$

dir. TE için sabit katsayılarla $b_n=\lambda^n$ değişikliği bir çözüm verir. λ için olası değerler $\lambda^{n-2}-2\lambda^{n-1}+2\lambda^n=0$ eşitliğinin ya da

$$\lambda^2-2\lambda+2=0$$

karakteristik denkleminin çözümüdür yani $\lambda_{1,2}=1\pm i$ dir. Bu değerler için $A.B\in C$ sabitleriyle $b_n=A\lambda_1^n+B\lambda_2^n$ genel çözümleri bulunur. O zaman başlangıç koşulları yerine konursa $0=b_0=A+B$, $1=b_1=A(1+i)+B(1-i)$ ve $A=\frac{1}{2i}$, $B=-\frac{1}{2i}$ bulunur. Sonuç olarak

$$a_n=\frac{b_n}{n!}=\frac{1}{n!}\frac{1(1+i)^n-(1-i)^n}{2i}=\frac{1}{n!}\mathrm{Im}(1+i)^n=\frac{1}{n!}\mathrm{Im}\left(\sqrt{2}e^{i\frac{\pi}{4}}\right)^n=\frac{1}{n!}2^{\frac{n}{2}}\sin\frac{n\pi}{4}$$

ve

$$F(x)=\sum_{n=0}^{\infty}\frac{1}{2!}2^{\frac{n}{2}}\sin\frac{n\pi}{4}x^n$$

serisi bulunur.

Örnek 3.2.2. $f(x)=\arcsin x$ olsun. Kolayca $f'(x)=\frac{1}{\sqrt{1-x^2}}$ ve $f''(x)=\frac{x}{\sqrt{1-x^2}^3}$ yazılır. Algoritmanın birinci adımı atlanır ve ikinci adımında, N=1 için:

$$\frac{1}{\sqrt{1-x^2}}+A_0\arcsin x=0\Rightarrow A_0=-\frac{1}{\sqrt{1-x^2}\arcsin x}$$

bulunur. A_0, rasyonel olmadığı için N yi bir arttırırız. N=2 için:

$$\frac{x}{(\sqrt{1-x^2})^3}+A_1\frac{1}{\sqrt{1-x^2}}+A_0\arcsin x=0,$$

$$x+A_1(1-x^2)+A_0(\sqrt{1-x^2})^3\arcsin x=0$$

elde edilir.

$(\sqrt{1-x^2})^3\arcsin x$ ifadesi, x ve $(1-x^2)$ ile fonksiyonel bağımsız olduğundan

$A_0(\sqrt{1-x^2})^3\arcsin x=0$ ve $x+A_1(1-x^2)=0$

yazılır ve

$A_0=0$ ve $A_1=-\frac{x}{1-x^2}$ bulunur. Bu sonuçlar yerlerine yazılırsa

$$(1-x^2)f'' - xf' = 0 \qquad (3.6)$$

DD`si bulunur. (3.3) kuralı uygulanırsa

$$(n+1)_2a_{n+2}\text{-}(n+1\text{-}2)_2a_n\text{-}(n+1\text{-}1)_1a_{n+1\text{-}1}$$

$$(n+1)(n+2)a_{n+2}\text{-}(n\text{-}1)na_n\text{-}na_n=0$$

$$(n+2)(n+1)a_{n+2}\text{-}n^2a_n=0$$

TE`si elde edilir. Dönüşüm, yalnızca iki tane toplama elemanı içerdiğinden, f hipergeometrik tiptedir.

Bundan başka, f fonksiyonunun m simetri sayısının 2`ye eşit olduğu hemen görülür. f(0) = 0 olması bize f fonksiyonunun tek olduğunu ifade eder. Bu nedenle

$f(x) = xh(x^2)$ olan FKS

$$H(x)=\sum_{k=0}^{\infty} c_k x^k$$

ile çalışmak uygun olacaktır ve böylelikle $c_k = a_{2k+1}$ dir. Bu durumda

$n \rightarrow 2k + 1$ yerine koyma işlemi, c_k için

$$c_{k+1} = \frac{\left(k+\frac{1}{2}\right)^2}{\left(k+\frac{3}{2}\right)(k+1)} c_k$$

için TE`sine götürür. Başlangıç koşulu $c_0=a=f'(0)=1$ dir. Böylece (2.20) yoluyla

$$ck = \frac{\left(\frac{1}{2}\right)_k \cdot \left(\frac{1}{2}\right)_k}{\left(\frac{3}{2}\right)_k k!}$$

$$= \frac{\left(\frac{1}{2}\frac{3}{2}\cdots\frac{(2k-1)}{2}\right)\left(\frac{1}{2}\frac{3}{2}\cdots\frac{(2k-1)}{2}\right)}{\left(\frac{3}{2}\frac{5}{2}\cdots\frac{(2k+1)}{2}\right)k!}$$

$$= \frac{\frac{1}{2^k}(1.3\ldots(2k-1)).\frac{1}{2^k}.(1.3\ldots(2k-1))}{\frac{1}{2^k}(3.5\ldots((2k+1)).k!}$$

$$= \frac{1}{2^k}.\frac{(2.4\ldots2k)(2.4\ldots2k)(1.3\ldots(2k-1))(1.3\ldots(2k-1))}{(2.4\ldots2k)(2.4\ldots2k)(3.5\ldots(2k+1)).k!}$$

$$= \frac{1}{2^k}\frac{((2k)!)^2}{(2k+1)!.k!.2^k(1.2\ldots k)}$$

$$= \frac{(2k)!}{(2k+1)4^k(k!)^2}$$

bulunur ve

$$F(x)=\sum_{k=0}^{\infty}\frac{(2k)!}{(2k+1)4^k(k!)^2}x^{2k+1}$$

olur.

Örnek 3.2.3. $f(x)=\operatorname{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}$ hata fonksiyonu olsun. Burada $f'(x)=\frac{2}{\sqrt{\pi}}e^{-x^2}$ ve $f''(x)=-\frac{4}{\sqrt{\pi}}xe^{-x^2}$ dir. Algoritma

$$f''+2xf'=0$$

DD ve dolayısıyla

$$(n+2)(n+1)a_{n-2}+2na_n=0$$

TE`i verir. Bu. hipergeometrik tiptir ve Örnek 3.2.2`deki yöntem, f fonksiyonunun tek olduğunu gösterir. Benzer işlemlerle H serisinin

$$c_{k+1}=-\frac{k+\frac{1}{2}}{\left(k+\frac{3}{2}\right)(k+1)}c_k$$

katsayılarını ve

$$c_0=a_1=f'(0)=\frac{2}{\sqrt{\pi}}$$

başlangıç koşulları ile elde edlir. öyle ki

$$c_k=\frac{2}{\sqrt{\pi}}\frac{(-1)^k}{(2k+1)k!}$$

ve

$$F(x)=\sum_{k=0}^{\infty}\frac{2}{\sqrt{\pi}}\frac{(-1)^k}{(2k+1)k!}x^{2k+1}$$

olur.

Örnek 3.2.4. f(x)=arctanx olsun. Burada $f'(x)=\frac{1}{1+x^2}$. rasyoneldir. Bu nedenle rasyonel algoritması uygulanır (§3.3).

$$f'(x)=\frac{1}{1+x^2}=\frac{1}{2}\left(\frac{1}{1+ix}+\frac{1}{1-ix}\right)$$

$$f'(x)=\frac{1}{2i}\left(\frac{1}{\frac{1}{i}+x}+\frac{1}{\frac{1}{i}-x}\right)$$

$$f'(x)=\frac{1}{2i}\left(\frac{1}{x-\left(-\frac{1}{i}\right)}-\frac{1}{x-\frac{1}{i}}\right)$$

kompleks PFD'sini (partial fraction decomposition) bulunur. $F'(x)=\sum_{n=0}^{\infty} b_n x^n$ ifadesindeki b_n katsayıları bölüm 3.3 deki algoritmanın 2. adımındaki eşitlikler ve binom serilerini kullanılarak.

$$b_n=\frac{1}{2i}\left(\frac{(-1)^1.1}{\left(-\frac{1}{i}\right)^{n+1}}-\frac{(-1)^1.1}{\left(\frac{1}{i}\right)^{n+1}}\right)$$

$$b_n=\frac{1}{2i}\left(-(-i)^{n+1}+(i)^{n+1}\right)$$

$$b_n=\frac{1}{2i}\left(-(-i)^n(-i)+(i)^{n+1}\right)$$

$$b_n=\frac{1}{2}(i^n+(-i)^n)=\frac{i^n}{2}(1+(-1)^n)$$

bulunur.

$$b_{2k-1}=\frac{i^{2k+1}}{2}(1+(-1)^{2k+1})=0$$

hesaplamasıyla F' serisinin çift olduğu ortaya çıkar.

$$b_{2k}=\frac{i^{2k}}{2}(1+(-1)^{2k})=(-1)^k$$

ve dolayısıyla $F'(x)=\sum_{k=0}^{\infty}(-1)^k x^{2k}$ olur. Bu serinin integrali alınırsa ise

$$F(x)=\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}x^{2k+1}$$

elde edilir.

Belirtelim ki. Arctan fonksiyonu da Örnek 3.2.2' dekine benzer hipergeometrik prosedürle işleme tabi tutulabilir.

Örnek 3.2.5. Şimdi de bir rasyonel örneğe bakalım. $f(x)=\frac{1}{x^2+2x+2}$ olsun.

Rasyonel algoritmayı uygulanırsa

$$f(x)=\frac{1}{x^2+2x+2}=\frac{i}{2}\left(\frac{1}{x+1-i}+\frac{1}{x+1+i}\right)$$
$$=\frac{i}{2}.\frac{1}{1-i}.\frac{1}{1+\frac{x}{1-i}}+\frac{i}{2}.\frac{1}{1+i}.\frac{1}{1+\frac{x}{1+i}}$$

kompleks PFD bulunur. FKS'nin katsayıları

$$a_n=-\frac{i}{2}\left(\frac{-1}{1-i}\right)^{n+1}+\frac{i}{2}\left(\frac{-1}{1+i}\right)^{n+1}$$
$$=\frac{(-1)^n i}{2}\left(\left(\frac{e^{i\frac{\pi}{4}}}{\sqrt{2}}\right)^{n+1}-\left(\frac{e^{-i\frac{\pi}{4}}}{\sqrt{2}}\right)^{n+1}\right)=\frac{(-1)^{n+1}}{\sqrt{2}^{n+1}}\sin\left((n+1)\frac{\pi}{4}\right)$$

olup

$$F(x)=\sum_{k=0}^{\infty}\frac{(-1)^{k+1}}{\sqrt{2}^{k+1}}\sin\left((k+1)\frac{\pi}{4}\right)x^k$$

şeklindedir.

Örnek 3.2.6. $f(x)=e^x$ olsun. Türevi $f'(x)=e^x$ dir. $A_0=-1$ çıkar ki bu x te rasyoneldir. Dolayısıyla DD yi $f'-f=0$ olarak bulunur.

$b_{n+1}-b_n=0$

$\lambda-1=0$ ise $\lambda=1$ dır.

$b^n=\lambda^n$ ise $a_n=\frac{b_n}{n!}=\frac{1}{n!}$

ve böylece

$$F(x)=\sum_{n=0}^{\infty}\frac{1}{n!}x^{n}$$

elde edilir.

3.3. Rasyonel Algoritma

Bu alt bölümde rasyonel algoritmanın detayları verilecektir

Algoritma 3.3.1.

1. Paydada bir rasyonel çarpanlara ayırma olduğunda f fonksiyonunun en azından algoritmik olarak yapılabilen bir

$$f(x)=\sum_{k=1}^{n}\left(\frac{c_{k1}}{x-z_k}+\frac{c_{k2}}{(x-z_k)^2}+\cdots+\frac{c_{kp_k}}{(x-z_k)^{p_k}}\right)$$

kompleksini hesaplayın.

2. Bu prosedürle f fonksiyonu, binom serileri tarafından genişletilebilen

$$\frac{c_{kj}}{(x-z_k)^j}=\frac{(-1)^j c_{kj}}{z_k^j\left(1-\frac{x}{z_k}\right)^j}$$

ifadelerinin toplamıdır ve

$$(\alpha_n)_{kj}=\frac{(-1)^j c_{kj}}{z_k^{j+n}}\binom{j+n-1}{n}$$

katsayılarına sahiptir. Rasyonel fonksiyonların, r_k (k=0.....M) sabit katsayıları ile (3.2) tipindeki bir TE'ye denk gelir.

3.4. Basit Diferensiyel Denklemler

Bu alt bölümde bir fonksiyonun diferensiyel denkleminin nasıl hesaplanacağı konusundan bahsedilecektir.

Bir f fonksiyonu verilmiş olsun ve (2.23) şeklinde bir DD, yani bir basit DD arayalım. Eğer bir basit DD mevcut ise, f fonksiyonu için TE si farklı olan birinciden farklı ikinci bir basit DD vardır (§3.5`e bakınız). Örneğin $f(x)=e^x$ üssel fonksiyonu için $a_{n+1}=\frac{1}{n+1}a_n$ TE`sine uygun

$$f' = f \tag{3.7}$$

basit DD`i bulunur. Fakat aynı zamanda f fonksiyonu, $f''=f'$ DD`sinin de bir çözümüdür (seri katsayıları için aynı TE`ye götürdüğünden aslında farklı değildir). Kombinasyon yoluyla, hipergeometrik tip olmayan

$$(n+2)(n+1)a_{n+2} = -(n+1) + 2a_n$$

TE ve (3.7) den farklı

$$f'' + f' = 2f \tag{3.8}$$

diferensiyel denklemi elde edlir. Burada eğer mevcut ise bir basit DD`yi bulan bir algoritmayı aşağıda verilmiştir. Bulunan DD nin mertebesi, f fonksiyonunun çözümü olduğu DD`ler içinde minimumdur.

Algoritma 3.4.1

(a) N=1 için f fonksiyonun birinci mertebeden basit DD`si olup olmadığını araştırın. Bunun için f fonksiyorunun türevini alın ve A_0 için

$$f'(x)+A_0f(x)=0$$

lineer eşitliğini çözün: yani $A_0=-\frac{f'(x)}{f(x)}$ denklemini çözün. A_0, x`te rasyonel ise, A_0 ın paydasıyla DD yi çarpın.

(b) Aranan DD için N sınırını bir arttırın.

$$f^{(N)}(x)+A_{N-1}f^{(N-1)}(x)+...+A_0f(x) \qquad (3.9)$$

ifadesini ($A_0, A_1, \ldots A_{N-1}$) nin değerlerine göre yazın. Terimlerin $A_0, A_1, \ldots A_{N-1}$ sayılarını sabit olarak alan fonksiyonel bağımsız N adet terimi içerip içermediğini test edin. Eğer bir çözüm bulunuyorsa; fonksiyonel bağımsız terimlere göre ayrım yapın ve katsayılarını sıfıra eşitleyerek bir lineer denklem sistemi oluşturun. Bu sistemi $A_0, A_1, \ldots A_{N-1}$ sayıları için çözün. Bunlar x`te rasyonel fonksiyonlardır ve tek bir çözüm vardır.

$A_0, A_1, \ldots A_{N-1}$ in ortak paydalarıyla bir çarpımdan sonra aranılan DD'yi bulun.

(c) Eğer (b) başarılı olmazsa N`yi bir artırarak (b) adımını tekrarlayın.

İspat. f fonksiyonu için bir

$$f^{(N)} + A_{N-1} f^{(N-1)} + \ldots A_1 f' + A_0 f = 0$$

DD`sini ararken algoritmanın N adımında bulunuyorsak ya (3.9) un fonksiyonel bağımsız terimlerinin sayısı N e eşittir ve fonksiyonel bağımsız terimlerin katsayılarını sıfıra eşitleyerek bulduğumuz lineer eşitlikler sistemi bir tek $A_0, A_1, \ldots A_{N-1}$ çözümüne sahiptir ya da, (3.9) un fonksiyonel bağımsız terimlerinin sayısı N `den daha büyüktür ve bir çözüm yoktur.

f fonksiyonu verilmiş olsun. N=1 ile başlanır. Bu durumda DD (3.9) fonksiyonel bağımsız en az bir tane terim içerir, yani rasyonel fonksiyonlar bu prosedürün dışında bırakılır. Eğer fonksiyonel bağımsız terimlerin sayısı 1`e eşitse (ve bu nedenle f', f nin bir rasyonel katsayısı ise), bir tek $A_0=-f'/f$ çözümü elde edilir. Eğer fonksiyonel bağımsız terimlerin sayısı 1`den farklıysa, fonksiyonel bağımsız terimlerin sayısı en az 2'dir ve N=2 ile başlamak gerekir. Bu durumda, fonksiyonel bağımsız terimlerin sayısı ikiden

az ya da ikiye eşit olsun. Öyleyse bir (A_0,A_1) çözüm vektörü bulunabilir. Bu çözümün tek olduğunu gösterelim. F fonksiyonu için

$$f''+A_1f'+A_0f=0 \quad \text{ve} \quad f''+a_1f'+a_0f=0$$

şeklinde iki DD olsun. Bu halde,

$$(A_1-a_1)f'+(A_0-a_0)f=0 \tag{3.10}$$

farkları, f fonksiyonu için geçerli bir DD olur. (3.10) basittir ve 1. mertebedendir. Fakat yukarıdaki adımdan, f fonksiyonu için 1. mertebeden basit bir DD olmadığını biliyoruz. Bundan dolayı (3.10) nun bütün katsayılarının sıfıra eşit olması yani $a_k=A_k$ (k=1,2) olması gerekir ve dolayısıyla çözüm tektir. Diğer durumda, fonksiyonel bağımsız terimlerin sayısı N'den büyüktür ve N=3 ile başlamak gerekir. Genel N için ispat, tümevarım kullanılarak yapılabilir.

Hipergeometrik tipte f fonksiyonları için, algoritmanın hipergeometrik tipte bir TE'ye uygun DD bulması kesin değildir. Örneğin $f(x)=e^x\sin x$ fonksiyonu için algoritma, $f''-2f'+2f=0$ DD'sini bulur. Bu DD uygun TE'ye transfer edirlise (§3.5'ya bakınız) hipergeometrik tipte olmayan

$$(n+2)(n+1)a_{n+2} = 2(n+1)a_{n+1} - 2a_n$$

TE'si elde edilir. Bununla birlikte f fonksiyonu, hipergeometrik tipte

$$(n+4)(n+3)(n+2)(n+1)a_{n+4} = -4a_n$$

TE'sine uygun $f^{(iv)}+4f=0$ dördüncü mertebeden DD'nin çözümüdür. Fakat bu, sık görülmeyen bir durumdur. Sadece birkaç durum dışında hipergeometrik tipte FLS, verilmiş algoritmalar yoluyla tanınır.

3.5. Tekrarlama Eşitliğini elde etme

Bir basit DD`den TE bulmanın yöntemi §3.1`te verilmişti. Bu bölümde bu ifadeler ispatlanacaktır.

$$F(x)= \sum_{n=k_0}^{\infty} a_n x^n$$

gösterimiyle bir F FLS si.

$$\sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} x^j F^{(k)}(x) = 0 \qquad (3.11)$$

DD`sine uysun. Tümevarım yoluyla

$$F^{(k)}(x)= \sum_{n=k_0}^{\infty} (n+1-k)_k a_n x^{n-k} \qquad (3.12)$$

yazılır. (3.12) (3.11) de yerine konursa.

$$\begin{aligned}
0 &= \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} x^j \sum_{n=k_0}^{\infty} (n+1-k)_k a_n x^{n-k} \\
&= \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} \sum_{n=k_0}^{\infty} (n+1-k)_k a_n x^{n-k+j} \\
&= \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} \sum_{n=k_0-k+j}^{\infty} (n+1-j)_k a_{n+k-j} x^n \\
&= \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} \left(\sum_{n=k_0-k+j}^{k_0+Q-1} (n+1-j)_k a_{n+k-j} + \sum_{n=k_0+Q}^{\infty} (n+1-j)_k a_{n+k-j} \right) x^n \\
&= \sum_{n=k_0+Q}^{\infty} \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} (n+1-j)_k a_{n+k-j} x^n + \sum_{j=0}^{Q}\sum_{k=0}^{Q} c_{jk} \sum_{n=k_0-k+j}^{k_0+Q-1} (n+1-j)_k a_{n+k-j} x^n
\end{aligned}$$

elde edilir. Eşitlenen katsayılar, $n \geq k_0+Q$ için (3.3)`e götürürler, fakat a_n ($n<k_0+Q$) başlangıç katsayılarının sonlu sayılarının ayrıca incelenmesi gereklidir.

$(n+1-j)_k$ her sabit j ve k için n`de bir polinom olduğundan, $r_k$ (k=0.....M) nın n`de rasyonel fonksiyonlar ve $M \in IN$ olduğu zaman hesaplanmış TE (3.2) nin özel tipinde olması gözleme kalmıştır.

3.6. Tekrarlama Eşitliğinin çözümü

Bu alt kesimde exp-like tip olma durumu incelenecektir. (3.4) den çıkan $b_n=n!a_n$ sayıları için,

$$\sum_{k=0}^{Q} c_k F^{(k)}(x) = 0 \quad c_k \ (k=0,....,Q) \tag{3.13}$$

DD`si kullanılarak exp-like tip olma durumunu incelenecektir. (3.12) (3.13)`e uygulanırsa

$$0 = \sum_{k=0}^{Q} c_k \sum_{n=k_0}^{\infty} (n+1)_k \, a_{n+k} x^n = \sum_{n=k_0}^{\infty} \sum_{k=0}^{Q} c_k (n+1)_k \frac{b_{n+k}}{(n+k)!} x^n = \sum_{n=k_0}^{\infty} \frac{1}{n!} \sum_{k=0}^{Q} c_k b_{n+k} x^n$$

elde edilir. Katsayılar eşitlenirse (3.4) elde edilir.

Şimdi de bu tip TE`yi çözmek için cebirsel tertibi verelim.

Algoritma 3.6.1.

(1) $b_n=\lambda^n$ düzenlemesi TE (3.4)`e yerleştirilirse

$$\sum_{k=0}^{Q} c_k \lambda^k = 0$$

karakteristik denklemi oluşur. Bu eşitliği λ için çözün ve α_k (k=1.....R) değerlerine sahip R farklı çözümü λ_k(k=1.....R)`yi sırasıyla elde edin.

(2) Genel çözüm, Q sayıdaki A_{kj} (k=1.....R (j=1.....α_k)) katsayıları ile

$$b_n = \sum_{k=1}^{R} \sum_{j=1}^{\alpha_k} A_{kj} n^{j-1} \lambda_k^n \tag{3.14}$$

şeklindedir.

(3) Q nun başlangıç değerlerini (3.14)'ye yerleştirince Q tane bilinmeyen A_{kj} ($k=1.....\alpha_k$) için Q adet denklemden oluşan bir lineer sistem oluşur. Bu lineer sistem regülerdir. Bu sistemi b_n için bir gösterim elde etmek amacıyla çözün .

İspat. λ_k ($k=1.....Q$) çözümlerinin $\alpha_k=1$ çokluklarına sahip olması ve dolayısıyla çiftlerin farklı olması durumunda çözüm örneğin (Walter (1985)) te bulunabilir. Genel durum benzer şekilde tümevarım ile ispat edilebilir.
Bir exp-like f fonksiyonu için Algoritma 3.4.1, sabit katsayılara sahip olmayan bir DD bulamayabilir. Bu nedenle exp-like tipi tanımakta başarısız olabilir. Bulunan DD'nin f fonksiyonu için de geçerli olan sabit katsayılı DD'den daha düşük bir mertebeye sahip olduğu açıktır. Örneğin, $f(x)=(1+x)e^x$ için Algoritma 3.4.1 değişken katsayılı ve birinci mertebe basit $(1+x)f'=(2+x)f$ DD'sini bulur. Diğer taraftan, f fonksiyonu, ikinci mertebeden sabit katsayılı $f''-2f'+f=0$ DD'sinin çözümü olduğundan exp-like tiptir.

3.7. Genelleştirme

Bu alt kesimde, hipergeometrik tipte fonksiyonlar sınıfını genişletmek için farklı bir tanım verilecektir. Ayrıca bu yaklaşımca kapsanan fonksiyonlar sınıfı örneklerle sergilenecektir.

Tanım 3.7.1. Eğer §3.4 prosedürü, bir hipergeometrik tipte TE'ye (2.25) çevrilen, §3.5'da tanımlanan uygulaması tarafından bir basit DD'ye götürüyorsa, bu f fonksiyonuna hipergeometrik tip denir [8].

Örnek 3.7.2. $f(x)=\ln x$ olsun. Bu durumda algoritma, DD'yi $f'+xf''=0$ olarak bulur. Bu DD, sıfır rasyonel fonksiyonu ile hipergeometrik tipte $a_n=0$ ($n\in IN$ için) TE'nin içine aktarılır. Bu f' nün hipergeometrik tipte (yani, negatif üslü bir tek terimli) olması anlamına gelir.

Benzer şekilde, $f(x)=\int \ln x dx=-x+x\ln x$ için, $f-xf'+x^2f''=0$ DD'sini ve f'' nin negatif üslü bir tek terimliye eşit $a_n=0$ ($n\geq 2$ için) TE'sini elde ederiz.

Bu nedenle yukarıdaki genişletilmiş tanımımız, FLS'nin integrallerini kapsar. Bir FLS'nin integrali

$$\int F(x)dx = \sum_{\substack{k=k_0 \\ k\neq -1}}^{\infty} \frac{a_{k-1}}{k} x^k + a_{-1}\ln x = \sum_{\substack{k=k_0+1 \\ k\neq 0}}^{\infty} \frac{a_{k-1}}{k} x^k + a_{-1}\ln x$$

şeklindedir.

Örnek 3.7.3. $f(x)=e^{\sqrt{x}}$ olsun. Bu f fonksiyonu için DD $-f+2f'+4xf''=0$ ve

$$2(1+2n)(1+n)a_{n+1}=a_n \qquad (3.15)$$

TE'si elde edilir. Diğer yandan,

$$f(x)=e^{\sqrt{x}}=\sum_{k=0}^{\infty}\frac{1}{k!}x^{\frac{k}{2}}=\sum_{n=0}^{\infty}\frac{1}{(2n)!}x^n \quad \text{öyleki } n=0,\frac{1}{2},1,\frac{3}{2},\ldots$$

dır. Gerçekten $a_n=\frac{1}{(2n)!}$ için TE (3.15) geçerlidir.

Benzer şekilde, $f(x)=\frac{1}{1-\sqrt[3]{x}}$ fonksiyonu için $f(x)=g(x^{1/3})$ ve g, $g(x)=\frac{1}{1-x}$ hipergeometrik fonksiyonu olsun. Bu f fonksiyonu için

$$2f+(-2+38x)f'+(-18x+45x^2)f''+(-9x^2+9x^3)f'''=0$$

DD'si ve $a_{n+1}=a_n$ TE'si elde edilir.

Bu nedenle, yeni tanım, f fonksiyonu hipergeometrik tipte bir F FLS sine denk ise $f(x^{1/n})(n\in IN)$ şeklindeki fonksiyonları da kapsamına alır.

Örnek 3.7.4. $f(x)=e^{1/x}$ olsun. $f+x^2f'=0$ DD'sini ve uygun TE olarak $a_{n+1}=-na_n$ eşitliği elde edilir. Burada TE, $a_1=0$ ve dolayısıyla bütün $k\in IN$ için $a_k=0$ bilgisini içerir. Bu, yalnızca pozitif olmayan indislerin gerçekleşmesini ve

dolayısıyla $g(x) = f(1/x)$ fonksiyonunun bir FKS olmasını garantiler. Gerçekten de üstel serilerden.

$$f(x) = \sum_{k=0}^{\infty} \frac{1}{k!} x^{-k}$$

olduğunu görülür.

Şimdi bu üç durumun yeni yaklaşımımızın kapsamında olduğunu göstereceğiz.

Teorem 3.7.5. f fonksiyonu, (2.6) gösterimiyle bir hipergeometrik tipte bir FLS `ye karşılık gelsin. Bu durumda

(a) $\int f$. (b) $f(x^{1/n})(n \in IN)$. (c) $f\left(\frac{1}{x}\right)$

hipergeometrik tipte fonksiyonlardır [8].

İspat. F `nin katsayıları için (2.25) şeklinde bir TE olsun.

(a) $\int f$ fonksiyonu, türevi gibi bir basit DD`nin çözümüdür. (3.4) den çıkan $\int f$ `nin b_n katsayıları için(logaritmik terimi ihmal ederek), Lemma 2.2.3 deki durum elde edilir.

(b) F (2.24) şeklinde hipergeometrik tipte bir FLS ve $g(x) = f(x^{1/n})$ olsun. Bu durumda G serisi

$$G(x) = \sum_{k=k_0}^{\infty} a_k x^{k/n} = \sum_{\substack{k=(k_0+j)\cdot n \\ j\in IN_0}} b_k x^k \qquad (3.16)$$

şeklinde gösterilir. Eğer F serisinin simetri sayısı m ise , P ve Q polinomları ile n.m simetri sayılı bir

$$Q(k)a_{k-nm} = P(k)a_k$$

gösterimi vardır. Bu polinomlar ve θ yerine $\theta_n = nx\frac{d}{dx}$ kullanıldığında Teorem 2.2.4`den, G serisi

$$Q(\theta_n - nm)g = x^m P(\theta_n)g$$

Şeklinde bir basit DD`nin çözümüdür. Bu nedenle G serisi verilen yaklaşım tarafından kapsanır. G formal serisinin b_k $\left(k = \frac{k_0 + j}{n}\right)$($j \in IN_0$) katsayıları için (3.16) kullanılarak $b_k = a_{nk}$ eşitliği bulunur,

$$b_{k+m} = a_{nk+m} = R(nk)a_{nk} = R(nk)b_k$$

ve g fonksiyonu hipergeometrik tiptedir.

(c) (a)`da olduğu gibi, f fonksiyonu için bir basit DD olduğundan dolayı, $g(x) = f\left(\frac{1}{x}\right)$ için bir basit DD vardır. TE (2.25)`de k, - k - m ile değiştirilirse

$$a_{-k-m} = \frac{1}{R(-k-m)}a_{-k}$$

eşitliğini elde edilir. G nin b_k katsayıları için $b_k = a_{-k}$ olur. Buradan

$$b_{k+m} = a_{-k-m} = \frac{1}{R(-k-m)}a_{-k} = \frac{1}{R(-k-m)}b_k$$

yazılır ve g fonksiyonu hipergeometrik tiptir.

Lemmanın kuralları hipergeometrik tip değişmeden tekrarlamalı olarak uygulanabilir.

3.8. Algoritmanın bir sadeleştirici olarak kullanımı

Algoritma bir sadeleştirici olarak kullanılabilir. Bulunan FKS sonlu bir toplam ise bir polinoma sahip olunur. Örnek olarak f(x)=cos(4arccos x) fonksiyonunu ele alalım. Bu fonksiyon bir Chebyshev polinomudur. Bununla birlikte,

algoritmada f`fonksiyonunun bir polinom olduğunu anlamanın cebirsel bir yolu yoktur. dolayısıyla rasyonel algoritma uygulanamaz. Algoritmanın bir uygulaması

> with(share):with(FPS):

See ?share and ?share.contents for information about the share library

[]

Share Library: FPS

Author: Gruntz. Dominik.

Description: FPS function attempts to find a formal power seriesexpansion for a function in terms of a formula for the coefficients

["Abramowitz". "ChebyshevT". "ChebyshevU". "ComplexApart"."Fibonacci". "FindDE". "FormalPowerSeries". "GegenbauerC"."HermiteH". "JacobiP". "LaguerreL". "Legendre_P". "Legendre_Q". "Limit". "PS". "PSInt". "Pochhammer". "RationalAlgorithm". "RecursionSimplify". "SimpleDE". "SimpleRE". "UpdateCoeff". "constantRE". "de2re". "hypergeomRE". "hypergeomRsolve". "printDE". "simp". "simpl"."simplify/Pochhammer". "simplify/factorial"]

> FPS(cos(4*arccos(x)).x):

$$8\,x^4 - 8\,x^2 + 1$$

sonucunu verir. Böylece algoritmamız. f fonksiyonu. 2 simetri sayılı hipergeometrik tip olduğundan f`nin kuvvet serilesinin katsayıları için bir kapalı şekil bulabilir ve f `nin bir polinom olup olmadığına karar verebilir.

Aşağıdakiler algoritma tarafından yapılan sadeleştirme örnekleri listesidir.

FPS(Sin(x)^2+Cos(x)^2. x) → 1

FPS(Cos(ArcSin(x))-Sqrt(1-x^2). x) → 0

FPS(Exp(ArcSinh(x))-Sqrt(1+x^2), x) → x

FPS(ArcSin(x)+I ArcSinh(I x), x, 0) → 0

FPS(ArcTan((x+y)/(1-x y))-ArcTan(x), x] → ArcTan(y)

FPS(ArcSin(x)+ArcCos(x), x) → Pi/2

FPS(Exp(x+y)/Exp(x), x) → Exp(y)

FPS(Exp(y/x))^(x), x) → Exp(y)

FPS(Sin(3*ArcCos(x))/Sqrt(1-x^2), x) → -1 + 4*x^2

FPS(Beta(x, 1/2, 2)*Sqrt(x), x, 0) → 2*x - (2*x^2)/3

4. SONUÇLAR

Bu bölümde algoritmanın bazı sonuçları verilecektir. Bunun için

$$\left(\frac{\arcsin x}{x}\right)^2 = \sum_{k=0}^{\infty} \frac{4^k (k!)^2}{(1+k)(1+2k)!} x^{2k} \tag{3.17}$$

$$e^{\arcsin hx} = x - \sum_{k=0}^{\infty} \frac{(2k)!}{(-4)^k (2k-1)(k!)^2} x^{2k} \tag{3.18}$$

$$e^{\arcsin x} = \sum_{k=0}^{\infty} \frac{4^k \prod_{j=1}^{k} \left(\frac{1}{4}(j-1)^2\right)}{(2k)!} x^{2k} + \sum_{k=0}^{\infty} \frac{4^k \prod_{j=1}^{k} \left(\frac{1}{2} - j + j^2\right)}{(2k+1)!} x^{2k+1} \tag{3.19}$$

$$\sqrt{\frac{1-\sqrt{1-x}}{x}} = \sum_{k=0}^{\infty} \frac{(4k)!}{16^k \sqrt{2}(2k)!(1+2k)!} x^k \tag{3.20}$$

formüllerini göz önüne alalım. (3.20). $\sqrt{\frac{1-\sqrt{1-x}}{x}}$ fonksiyonunun tek kuvvetleri için hesaplanabilir.

Bununla birlikte, bütün bu örnekler bilinmektedirler. Örnek (3.17), [3] ve [5] de bulunabilir ve Clausen formülü denen özel durumda eğer parametreler ${}_3F_2\left(\begin{matrix} 2a & 2b & a+b \\ a+b+1/2 & 2a+2b \end{matrix}\right) = \left({}_2F_1\left(\begin{matrix} a & b \\ a+b+1/2 \end{matrix}\middle| x\right)\right)^2$ ise bir ${}_2F_1$ hipergeometrik fonksiyonunun serisi bir ${}_3F_2$ hipergeometrik fonksiyonudur. Problemimizde a=b=1/2 dir ve Clausen'in formülü

$$\left(\frac{\arcsin x}{x}\right)^2 = {}_3F_2\left(\begin{matrix} 1 & 1 & 1 \\ 3/2 & 2 \end{matrix}\middle| x^2\right) = \left({}_2F_1\left(\begin{matrix} 1/2 & 1/2 \\ 3/2 \end{matrix}\middle| x^2\right)\right)^2$$

eşitliğini gösterir. İkinci örnek (3.18), ters hiperbolik sinüs fonksiyonu için

$$e^{\operatorname{arcsinh} x} = x + \sqrt{1 + x^2} \tag{3.21}$$

gösteriminden kolayca çıkar. Bu, kuvvet serilerine açıkça sahip olan bir binom ifadesi gösterimidir [5]. Diğer taraftan, algoritma bu gösterimi (3.21) formülünün açık bir bilgisi olarak bulmuştur ki, bu olay §3.8`de daha ayrıntılı incelenmiştir.

Üçüncü ifade (3.19), [5] de bulunabilir. Bununla birlikte, [3] da bu fonksiyonun kuvvet serileri katsayıları için (bu yazarlar bir genel formül veremediklerinden) $e^{\arcsin x}$ in ilk beş katsayısı verilmiştir. Her iki formül, (3.18) ve (3.19), orijinin bir komşuluğunda geçerli olan

$$\left(x + \sqrt{1 + x^2}\right)^a = \sum_{k=0}^{\infty} \frac{2^k \left(\frac{a}{2} - \frac{k}{2} + 1\right)_k}{(1 + k/a)k!} x^k$$

özdeşliğinin a = 1 ve a = i (x→ ix) özel durumlarıdır [5], ve bu formül uygulamamız tarafından bulunur.

Formül (3.20), orijinin bir komşuluğunda geçerli olan

$$\sqrt{\frac{1 - \sqrt{1 - x}}{x}} = \frac{\sqrt{1 + \sqrt{x}} - \sqrt{1 - \sqrt{x}}}{\sqrt{2x}}$$

özdeşliğince bulunabilir.

Uygulamamızın diğer başarılı örnekleri,

$$e^x - 2e^{-\frac{x}{2}} \cos\left(\frac{\sqrt{3}x}{2} + \frac{\pi}{3}\right) = \sum_{k=0}^{\infty} \frac{3}{(1 + 3k)!} x^{1+3k}$$

ve

$$\frac{1}{2} \ln\left(\frac{1 + x}{1 - x}\right) - \arctan x = \sum_{k=0}^{\infty} \frac{2}{3 + 4k} x^{3+4k}$$

dır [9].

KAYNAKLAR

1. Buchberger, B., 1982, **Computer Algebra: Symbolic and Algebraic Computation**, Springer-Verlag Wien,

2. Davenport, J.H., Siret, Y., Tournier, E., 1993, **Computer Algebra Systems and Algorithms for Algebraic Computation,** Acedemic Press, London-San Diago

3. Gradshteyn, I.S.,Ryzhik, I.M., 1980, **Table of integrals, series, and products-corrected and enlarged version**, New York-London:Acedemic Press

4. Hacısalihoğlu, H.H., 1996, Balcı, M., **Temel ve Genel Matematik**, Ankara

5. Hansen, E.R., 1975, **A Table of Series And Products**, Englewood Cliffs,NJ:Prentice Hall

6. Harper, D., 1991, **A Guide to Computer Algebra Systems**, Jhon Wiley Chichester

7. Henrici, P., 1974, Applied and computational complex analysis I., **Pure and Applied Mathematics,** New York-London

8. Koepf, W., 1992, Power Series in Computer Algebra, **Journal of Symbolic Computation**, Vol 13, 581-603

9. Marsden, J.E., 1974, **Elemantary Classical Analysis**, University of California, Berkeley, San Francisco

10. Rivlin, T.J., 1990, **Chebyshev Polinomials: From Approximation and Number**, Wiley, New York

11. Ruiz, J.M., 1993, **The basic theory of power series**, Vieweg Wiesbaden

12. Saff, E.B., Snider, A.D., 1993, **Foundamentals of Complex Analysis for Mathematics**, Science, and Engineering, New Jersey: Prentice Hall

13. Smith, K.T., 1987, **Power series from a computational point of view**, Springer-Verlag, New York

14. Tournier, E., 1989, **Computer Algebra and Differential Equations**, Academic Press, London

15. Winkler, F., 1996, **Polynomial Algorithms in Computer Algebra**, Springer Wien, New York

EKLER

EK1: ALGORİTMANIN MAPLE KODU

```
#
## <SHAREFILE=analysis/FPS/FPS.mpl >
## <DESCRIBE>
## <NOSHIFT>
##  (update)
## The FormalPowerSeries (FPS) function tries to find a formal
## power series expansion for a function in terms of a formula
## for the coefficients.  For example
##
## > FormalPowerSeries(exp(x)*sin(x),x);
##
##                  infinity
##                  -----    k 1/2            k
##                   \     (2 )   sin(1/4 k Pi) x
##                    )    ------------------------
##                   /              k!
##                  -----
##                  k = 0
##
## > FPS(exp(x^2)*(1-erf(x)),x=infinity,left);
##
##                  infinity
##                  -----          (- k)   (- k)    (2 k + 1)
##                  \     (2 k)! 4     (-1)     (1/x)
##                   )    ------------------------------------
##                  /                 k!
##                  -----
##                  k = 0
##                  ------------------------------------------------
##                                    1/2
##                                   Pi
## AUTHOR: Dominik Gruntz, gruntz@inf.ethz.ch
## </DESCRIBE>
## <UPDATE=R4update >


# FormalPowerSeries 1.3.1995
##############################
alias( PS = `FPS/FPS` ):
alias( ComplexApart = `FPS/ComplexApart` ):
alias( RationalAlgorithm = `FPS/RationalAlgorithm` ):
alias( constantRE = `FPS/constantRE` ):
alias( hypergeomRE = `FPS/hypergeomRE` ):
alias( UpdateCoeff = `FPS/UpdateCoeff` ):
alias( PSInt = `FPS/Int` ):
alias( de2re = `FPS/de2re` ):
alias( hypergeomRsolve = `FPS/hypergeomRsolve` ):
alias( printDE = `FPS printDE` ):
alias( DIRECTION = `FPS DIRECTION` ):
alias( simpl = `FPS simpl` ):
alias( FindDE = `FPS FindDE` ):
alias( Limit = `FPS Limit` ):
alias( Recursion =  `FPS Recursion` ):
```

```
alias( RecursionSimplify = `FPS RecursionSimplify` ):

FormalPowerSeries := proc(f, x) local i, z, S, dir, argseq; global infolevel:
option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  if not assigned(infolevel['FormalPowerSeries']) then infolevel['FormalPowerSeries'] := 1 fi:
  argseq := NULL:

  for i from 3 to nargs do
    if member(args[i], {left,right,real,complex}) then dir := args[i]
    else argseq := argseq,args[i]
    fi;
  od:

  if type(x, equation) then z := lhs(x);
    if  rhs(x)= infinity then
      if dir = right or dir = real then
        ERROR(`inconsistent direction with infinities`) fi;
      userinfo(2, 'FormalPowerSeries', `=> f := `, subs(z=1/z, f) );
      if dir = left or not assigned(dir) then
        S := traperror(subs(z= 1/z, PS(subs(z= 1/z, f), z, right, argseq)));
      else
        S := traperror(subs(z= 1/z, PS(subs(z= 1/z, f), z, dir, argseq)));
      fi
    elif rhs(x)=-infinity then
      if dir = left or dir = real then
        ERROR(`inconsistent direction with infinities`) fi;
      userinfo(2, 'FormalPowerSeries', `=> f := `, subs(z=-1/z, f) );
      if dir = right or not assigned(dir) then
        S := traperror(subs(z=-1/z, PS(subs(z=-1/z, f), z, right, argseq)));
      else
        S := traperror(subs(z=-1/z, PS(subs(z=-1/z, f), z, dir, argseq)));
      fi
    else
      userinfo(2, 'FormalPowerSeries', `=> f := `, subs(z = z+rhs(x), f) );
      if assigned(dir) then
        S := traperror(subs(z=z-rhs(x), PS(subs(z = z+rhs(x), f), z, dir, argseq)))
      else
        S := traperror(subs(z=z-rhs(x), PS(subs(z = z+rhs(x), f), z, argseq)))
      fi
    fi
  else
    if assigned(dir) then
      S := traperror(PS(f, x, dir, argseq))
    else
      S := traperror(PS(f, x, argseq))
    fi
  fi:
  if S=lasterror then 'procname(args)'
  else S
  fi;

end: # FormalPowerSeries
FPS := FormalPowerSeries:
```

```
SimpleDE := proc(f, x)
  local Nmax, A, F, G, DF, degreeOfDE, deq, i, j, result;
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  for i from 3 to nargs do
    if type(args[i], integer) or args[i]=infinity then Nmax := args[i] fi;
    if type(args[i], name) then F := args[i] fi;
  od;
  if not assigned(Nmax) then Nmax := 5 fi;

  DF[0] := simpl(f);
  for degreeOfDE while degreeOfDE <= Nmax do
    DF[degreeOfDE] := simpl(diff(DF[degreeOfDE-1], x));
    A := 'A';
    deq := FindDE(DF, x, degreeOfDE, A, G);
    if deq=FAIL then next fi;

    for j from 0 to degreeOfDE-1 do
      if not assigned(A[j]) then A[j] := 0 fi
    od;

    deq := collect(deq, {seq(G(i), i=0..degreeOfDE)}, recursive, factor);
    RETURN(eval(subs(G(0)=F(x), subs(['xx'=x,'FF'=F], G = (n -> diff(FF(xx),xx$n))), deq=0)))
  od;
  ERROR(`no simple DE found`)
end;

SimpleRE := proc(f,x)
  local Nmax, A, F, DF, degreeOfDE, deq, i, j, a, k, req;
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  for i from 3 to nargs do
    if type(args[i], integer) or args[i]=infinity then Nmax := args[i] fi;
    if type(args[i], name) then
      if not assigned(a) then a :=args[i]
      elif not assigned(k) then k := args[i]
      fi
    fi
  od;
  if not assigned(Nmax) then Nmax := 5 fi;

  DF[0] := simpl(f);
  for degreeOfDE while degreeOfDE <= Nmax do
    DF[degreeOfDE] := simpl(diff(DF[degreeOfDE-1], x));
    A := 'A';
    deq := FindDE(DF, x, degreeOfDE, A, F);
    if deq=FAIL then next fi;

    for j from 0 to degreeOfDE-1 do
      if not assigned(A[j]) then A[j] := 0 fi
    od;

    req := de2re(deq, F, x, a, k);
    RETURN(collect(req, indets(req, 'a'(anything)), recursive, factor)=0)
  od;
```

```
  ERROR(`no simple DE found`)
end:

# Private Part
###############
# The whole algorithm assumes, that intermediate powerseries are always represented
# in the form
#   Sum(ak*x^f(k), k=0..infinity)
# i.e. the indeterminante x is present in a term of type `^`
# and the range always goes from 0 to infinity.
#

PS := proc(f, x)
  local A, degreeOfDE, DF, F, j, n, s, ind, deq, eq, req, ak, ak1, ak2, Nmax, M, eqq, eqr,
      R, i, m, leadcoeff, a, k, m0, S, result, RAresult, parameters, r, direction, mode,
      sol, sol1, sol2, deg, deg0, DEfound, reqnew, indnew;
  global DIRECTION;
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  for i from 3 to nargs do
    if  member(args[i], {'right','left','real','complex'}) then direction := args[i]
    elif member(args[i], {'explike','rational','hypergeometric'}) then mode := args[i]
    elif type(args[i], integer) or args[i]=infinity then Nmax := args[i]
    fi
  od;

  # default values:
  if not assigned(direction) then direction := complex fi;
  if not assigned(Nmax) then Nmax := 5 fi;

  DIRECTION := direction;
  _EnvNestLevel := 0;
  DEfound := 0; if mode=rational then DEfound:=1 fi;

  DF[0] := simpl(f);
  if type(DF[0], polynom(anything,x)) then
    RETURN(DF[0])
  fi;

  for degreeOfDE while degreeOfDE <= Nmax do

    DF[degreeOfDE] := simpl(diff(DF[degreeOfDE-1], x));

    if mode <> rational then
      # Search for a simple DE
      userinfo(3, 'FormalPowerSeries', `looking for DE of degree`, degreeOfDE);
      A := 'A';
      deq := FindDE(DF, x, degreeOfDE, A, F);
      if deq <> FAIL then DEfound := DEfound+1;
        req := de2re(deq, F, x, a, k);
        ind := map(x -> op(1,x), indets(req, a(anything)));
        M := max(op(ind));
        parameters := indets(req, name) intersect {seq(A[j], j=0..degreeOfDE-1)};
        if parameters <> {} then # try to set free parameters
          if nops(ind) > 2 and not mode='explike' then
```

```
# 1. try to get hypergeometric case (nops(ind)=2)
for i in sort(convert(ind minus {M}, list), (a,b) -> evalb(a-b>0)) do
  # this special sorting will make the symmetry number minimal!
  map( (k, req, a) -> coeff(req, a(k)), ind minus {M,i}, req, a):
  s := solve(%, parameters):
  if s <> NULL then
    if not has(s,x) then
      userinfo(3, 'FormalPowerSeries', `hypergeometric type found`, s):
      assign(s):
      req := expand(normal(req)):
      ind := map(x -> op(1,x), indets(req, a(anything))):
      ASSERT(nops(ind)<=2, `assertion 1`):
      break
    else
      reqnew := de2re(subs(s, deq), F, x, a, k):
      indnew := map(x -> op(1,x), indets(reqnew, a(anything))):
      if nops(indnew) <= 2 then
        userinfo(3, 'FormalPowerSeries', `hypergeometric type found`, s):
        assign(s):
        req := reqnew:
        ind := indnew:
        break
      fi
    fi
  fi:
od:
fi:
if nops(ind) > 2 and has(deq, x) and not mode='hypergeometric' then
  # 2. try to generate an 'explike' DE
  eq := {seq(coeff(deq, F(j)), j=0..degreeOfDE-1)}:
  eq := map((t,p) -> normal(t/p), eq, coeff(deq, F(degreeOfDE))):
  eq := select((t,p,x) -> has(t,x) or has(t,p), eq, parameters, x):
  if parameters minus eq = {} then # all parameters must be constant.
    eq := eq minus parameters:
    eqq := map((t,x) -> quo(numer(t),denom(t),x), eq, x):
    eqr := map((t,x) -> rem(numer(t),denom(t),x), eq, x):
    eq := map(coeffs, eqq, x) minus map(coeff, eqq, x, 0)
        union map(coeffs, eqr, x):
    sol := solve(eq, parameters):
    if sol <> NULL then assign(sol) fi:
  elif not has(map((t,p,x)->has(t,x) and not has(t,p),eq,parameters,x),true) then
    eq := {seq(op(j,eq)=C(j), j=1..nops(eq))}:
    sol1 := solve(%, parameters):
    if sol1 <> NULL then
      # lprint(`sol1 <> NULL`):
      map(lhs, eval(subs(sol1, eq))): # these must all be constant:
      map(normal, %): map(diff, %, x):
      sol2 := solve(%, {seq(C(j), j=0..nops(eq))}):
      if sol2 <> NULL and not has(sol2,x) then
        # lprint(`sol2 <> NULL`):
        sol2 := map(t -> if op(1,t)=op(2,t) then op(1,t)=0 else t fi, sol2):
        sol := eval(subs(sol2, sol1)):
        # sol := map(t -> if op(1,t)=op(2,t) then op(1,t)=0 else t fi, sol):
        assign(sol):
      fi:
```

```
      fi;
    fi;
    deq := expand(numer(eval(deq))):
    req := de2re(deq, F, x, a, k):
  fi;
  # set values to 0
  for j from 0 to degreeOfDE-1 do
    if not assigned(A[j]) then A[j] := 0 fi
  od:
  deq := eval(deq):
  req := numer(eval(req)):
  ind := map(x -> op(1,x), indets(req, a(anything))):
  M := max(op(ind)):
fi; # parameters <> {}

req := collect(req, map((t,a)->a(t), ind, a));
leadcoeff := numer(coeff(req, a(M))); n := nops(ind);
req := factor(readlib('isolate')(req, a(M)));

userinfo(3, 'FormalPowerSeries', `DE of degree`, degreeOfDE, `found.`);
userinfo(3, 'FormalPowerSeries', `DE = `, print(printDE(deq,F)));
userinfo(3, 'FormalPowerSeries', `RE = `, print(req));

if n=1 and not mode='explike' then # funktionen mit endlicher darstellung
  userinfo(3, 'FormalPowerSeries', `ps with finite number of non-zero coefficients`);
  # a(k+1) = R
  R := subs(k=k+(k+1-M), rhs(req))/a(k);
  leadcoeff := subs(k=k+(k+1-M), leadcoeff);
  # solve RE a(k+1) = R(k) given DF, R, leadcoeff
  ASSERT(R=0, `R <> 0`);
  result := traperror(constantRE(1, leadcoeff, DF, k, x));
  if result <> lasterror then RETURN(result) fi;
elif n=2 and not mode='explike' then # RE is of hypergeometric type
  m := abs(op(1,ind)-op(2,ind));
  R := subs(k=k+(k+m-M), rhs(req))/a(k);
  leadcoeff := subs(k=k+(k+m-M), leadcoeff);
  # solve hypergeometric RE a(k+m) = R(k) * a(k) given DF, m, R, leadcoeff
  result := traperror(hypergeomRE(m, R, leadcoeff, DF, k, x));
  if result <> lasterror then RETURN(result) fi;
#  else
#    req := {req} union {seq(a(j) = limit(DF[j], x=0, DIRECTION), j=0..degreeOfDE-1)};
#    ak := rsolve(req, a);
#    if op(0,ak)=rsolve then result := `RE could not be solved`
#    else RETURN(Sum(ak*x^k, k=0..infinity))
#  fi;
#  # result := `RE could not be solved`
fi;

deq := normal(deq/coeff(deq, F(degreeOfDE)));

if not has(deq, x) and not mode='hypergeometric' then # DE has constant coefficients

  req := a(k+degreeOfDE) = sum(A['j']*a(k+'j'), 'j'=0..degreeOfDE-1);
  userinfo(3, 'FormalPowerSeries', `DE has constant coefficients`);
```

```
      S := 0:
      for i from 0 while not has(req, a(k+i)) do
        S := S + Limit(DF[i], x=0, DIRECTION)*x^i
      od: m0 := i:
      for i from m0 to degreeOfDE-1 do
        req := req, a(i) = Limit(DF[i], x=0, DIRECTION):
      od:
      r := rsolve({req}, a):
      if type(r,'set') then r := eval(subs(r, op(1,{req}))) fi:
      ak1 := r/k!: ak2 := expand(evalc(ak1)):
      if length(ak1) < length(ak2) then ak := ak1 else ak := ak2 fi:
      if ak <> 0 then S := S+Sum(ak*x^k, k=m0..infinity) fi:
      RETURN(S)
    fi:
  fi: # deq <> fail
fi: # mode <> rational

# either no differential equation found, or the
# Hypergeometric algorithm failed to compute the FormalPowerSeries.
# Try the rational algorithm in the latter case. if possible
if (DEfound>0 or degreeOfDE=Nmax)
and not (mode='explike' or mode='hypergeometric') then

  if DEfound=1 then deg0 := 0 else deg0 := degreeOfDE fi:
  for deg from deg0 to degreeOfDE do

    if type(DF[deg], ratpoly(anything,x))
    and not (mode='explike' or mode='hypergeometric') then

      userinfo(3, 'FormalPowerSeries', deg, `. derivative is rational`);

      RAresult := traperror(RationalAlgorithm(DF[deg], x)):
      if RAresult <> lasterror then
        for j to deg do
          RAresult := PSInt(RAresult,x);
          eval(subs(Sum = (
            (ak,eq) -> if not has(op(2,eq),RootOf) then subs(op(1,eq)=0, ak)
                  else sum(ak,eq)
                  fi), RAresult));
          s := Limit(DF[deg-j]-%, x=0, DIRECTION):
          if has(s, infinity) then ERROR(`should not happen`) fi:
          if s=undefined then ERROR(`no initialisation found`) fi:
          RAresult := s + RAresult
        od:
        RETURN(RAresult)
      fi:
    fi:
  od:
  DEfound := DEfound-1:
fi:
if result = `no initialisation found`
or result = `infinite loop detected`
or result = `essential singularity`
or result = `not a power series`
then break fi:
```

```
  od:

  if not assigned(result) then
    if assigned(req) then
      userinfo(1, 'FormalPowerSeries', `ERROR: RE could not be solved`);
      ERROR(`RE could not be solved`);
    elif deq = FAIL then
      userinfo(1, 'FormalPowerSeries', `ERROR: no DE found up to order`,Nmax);
      userinfo(1, 'FormalPowerSeries', `==> increase order (last argument)`);
      ERROR(`no DE found up to order`,Nmax,`increase order`)
    else
      userinfo(1, 'FormalPowerSeries', `ERROR: no derivative is rational up to order`,Nmax);
      ERROR(`Ratalgo did not succeed`);
    fi;
  else
    userinfo(1, 'FormalPowerSeries', `ERROR: `, result);
    ERROR(result)
  fi;

end: # FormalPowerSeries


FindDE := proc(DF, x, degreeOfDE, A::name, F::name)
  local eq, j, J, deq, i, ind, list1, list2, s, terms;

  eq := expand(DF[degreeOfDE] + sum(A['j']*DF['j'], 'j'=0..degreeOfDE-1));
  eq := RecursionSimplify(eq):
  eq := expand(numer(eq));

  if type(eq,`+`) then list1 := convert(eq, list) else list1 := [eq] fi;
  terms := {};
  while list1 <> {} do list2 := {}; j := list1[3]; J := j;
    for i from 2 to nops(list1) do
      if type(normal(j/list1[i]), ratpoly(anything, x)) then J := J + list1[i]
      else list2 := list2 union {list1[i]} fi:
    od; terms := terms union {J};
    list1 := list2;
  od;

# c := indets(eq, function);      # coeffs must be rationally indipendant
# s := solve({coeffs(eq,c)}, ind): #-> risch convert to exp etc.
  ind := {seq(A[j], j=0..degreeOfDE-1)};
  s := traperror(solve(terms, ind));
  if s = NULL or s=lasterror then RETURN(FAIL) fi;

  assign(s);
  for j from 0 to degreeOfDE-1 do
    if not type(A[j], ratpoly(anything, x)) then
      A[j] := normal(A[j]); A[j] := simplify(convert(A[j],exp))
    fi
  od;

  deq := F(degreeOfDE) - sum(A['j']*F('j'), 'j'=0..degreeOfDE-1);
  deq := primpart(deq, {seq(F(j), j=0..degreeOfDE)} ):  #deq := numer(normal(deq));
end:
```

```
ComplexApart := proc(e,x) local f, s, location, pfd, n, d;
                global sharename, libname:

  f := normal(e):
  n := numer(f): d := denom(f): n:= n/lcoeff(d,x):
  if _EnvExplicit = false then
    s := []
  else
    userinfo(2, 'FormalPowerSeries', `ComplexApart: calling solve`):
    s := [solve(d,x)]
  fi:
  if s <> [] and not has(s,RootOf) then
    convert(map((z,x) -> x-z, s, x),`*`):
    convert(n/%, parfrac, x, true)
  else
    if not assigned(sharename) then
      readlib('share'):
      location := `share/find`():
      if location <> FAIL then
        sharename := location: libname := libname,sharename
      else
        userinfo(2, 'FormalPowerSeries', `ComplexApart: sharelibrary not found!`):
      fi
    fi:
    if assigned(sharename) then
      readshare(fparfrac, calculus):
      pfd := traperror(fparfrac(f,x)):
      if pfd = lasterror then
        ERROR(`fullparfrac expects rational function in x`) fi:
      pfd
    else
      ERROR(`share library not found`)
    fi:
  fi:
end: # ComplexApart

RationalAlgorithm := proc(f,x)
  local PFD, S, ak, bk, t, d, c, j, k, xk, s, p:
  # first compute the complex partial fraction decomposition (PFD) of f
  # and store the terms in the list PFD.

  userinfo(2, 'FormalPowerSeries', `RationalAlgorithm called, f := `, f):

  PFD := ComplexApart(f,x):

  if type(PFD,`+`) then PFD := [op(PFD)] else PFD := [PFD] fi:
  S := 0: ak := 0:

  for t in PFD do
    if not has(t,x) then S := S+t
    elif type(t, polynom(anything,x)) then S := S+t
    elif has(t, Sum) then # fullparfrac-term
      if type(t,`*`) then s := select(has, t, Sum): c := t/s else s := t: c := 1 fi:
```

```
      t := op(1,s):
      d := 1 select(has, p*t, x): c := c*t*d:
      if type(d,`^`) then j := op(2,d): d := op(1,d) else j := 1 fi:
      # Sum(c*d^j, alpha=RootOf()) where d=x-alpha
      ASSERT(d=x-'alpha'):
      ak := ak + Sum(c*(-1)^j*'alpha'^(j-k)*(j-k-1)!/k!/(j-1)!, op(2,s)):
    else
      # t = c*(x-xk)^j
      # d is the term (x-xk)^j, and c is the constant.
      # the variable p only assures that p*t is a product.
      d := 1 select(has, p*t, x): c := t*d:
      if type(d,`^`) then j := op(2,d): d := op(1,d) else j := 1 fi:
      if d=x then S := S+c/x^j; next fi:
      ASSERT(type(d,`+`), `RationalAlgorithm: d should be +`):
      # x may still by scaled by a factor s
      s := select(has, d, x)/x: d := d/s: c := c/s^j:
      xk := x-d: c := c*(-1)^j/xk^j: ak := ak+c*(j+k-1)!/k!/(j-1)!/xk^k
    fi
  od:
  ak := normal(ak):
  if ak = 0 then RETURN(S) fi:
  if expand(subs(k=2*k, ak))=0 then
    bk := simplify(subs(k=2*k+1, ak), power, 'symbolic'):
    RETURN(S + Sum(bk*x^(2*k+1), k=0..infinity))
  fi:
  if expand(subs(k=2*k+1, ak))=0 then
    bk := simplify(subs(k=2*k, ak), power, 'symbolic'):
    RETURN(S + Sum(bk*x^(2*k), k=0..infinity))
  fi:
  RETURN(S+Sum(ak*x^k, k=0..infinity))
end: # RationalAlgorithm

constantRE := proc(m, leadcoeff, DF, k, x)
# solve constant RE
#
#    leadcoeff * a(k+1) = 0
#
# where leadcoeff was the leading coeffitient of the RE
# and DF is a table where DF[i] = diff(f, x$i)
#

  local m0, m1, fract, DF2, S, i, c0, q, a, A, m2, S0, g:
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`:

  _EnvNestLevel := _EnvNestLevel-1:
  if _EnvNestLevel>20 then ERROR(`infinite loop detected`) fi:

  a := factor(leadcoeff):

  userinfo(3, 'FormalPowerSeries', `RE is constant`):
  # userinfo(3, 'FormalPowerSeries', `leadcoeff := `, a):
  userinfo(3, 'FormalPowerSeries', `RE: `, print(a*'a'(k+m)=0) ):

  A := select(t -> not has(t, 1), [solve(a, k)]) :
```

```
fract := convert(map(t -> denom(t), A), set) minus {1};
if fract <> {} then q := lcm(op(fract));
  g := subs(x=x^q, DF[0]);
  indets(g, (identical(x)^anything)^anything);
  map((t,x) -> t=x^(op(2,op(1,t))*op(2,t)), %, x);
  DF2[0] := eval(subs(%, g));
  # DF2[0] := simplify(subs(x=x^q, DF[0]), power, 'symbolic');
  userinfo(3, 'FormalPowerSeries', `RE modified to `, k = k/q);
  userinfo(2, 'FormalPowerSeries', `=> f := `, DF2[0] );
  S := constantRE(q*m, subs(k=k/q, leadcoeff), DF2, k, x);
  S := eval(subs(x=x^(1/q), S));
  S := simplify(S, power);
  RETURN(S)
fi;
# {solutions in A are integer}

if A <> [] then
  m1 := min(op(A));
  if type(m1, function) and op(0,m1)='min' then
    m2 := eval(select(type, m1, rational));
    if m2 = infinity then m2 := -m fi;
    userinfo(1, 'FormalPowerSeries', `provided that `, m2, ` <= `, m1);
    m1 := m2;
  fi;
  m1 := m1+m;
  if m1 <> 0 then
    DF2[0] := simplify(x^(-m1)*DF[0], power);
    userinfo(3, 'FormalPowerSeries', `working with x^`,eval(-m1),`*f `);
    userinfo(2, 'FormalPowerSeries', `=> f := `, DF2[0]);
    S := constantRE(m, subs(k=k+m1, leadcoeff), DF2, k, x);
    S := UpdateCoeff(S, x, m1);
    RETURN(S)
  fi
fi; # { min(op(A)) = -1}

A := select(type, A, integer);

if A = [] then m0 := 0
else m0 := max(op(A))+1;
fi;
userinfo(3, 'FormalPowerSeries', `RE valid for all k >=`, m0);

c0 := Limit(DF[0], x=0, DIRECTION);
if type(c0,infinity) then
  # now we have either a essential or a logarithmic singularity.
  c0 := limit(x*DF[0], x=0, DIRECTION);
  if type(c0,infinity) then
    ERROR(`essential singularity`) fi;
fi;

S := 0;
for i from 0 to m0+m-1 do
  if not assigned(DF[i]) then DF[i] := simpl(diff(DF[i-1],x)) fi;

  c0 := Limit(DF[i], x=0, DIRECTION) i!;
```

```
    userinfo(3, 'FormalPowerSeries', cat(`a(`,i,`) = `),c0);

    if has(c0, infinity) then
      # SEEMS TO BE A LOGARITHMIC SINGULARITY since the poles have
      # already been removed!!! log(x)*x^i
      DF2[0] := simpl(diff(x^(-i)*DF[0], x));
      userinfo(3, 'FormalPowerSeries', `logarithmic singularity, working with (f*`,
                                  x^(-i),`)'`);
      userinfo(2, 'FormalPowerSeries', `=> f := `,DF2[0]);

      if  type(DF2[0], polynom(anything,x)) then
        S := DF2[0]
      else
        S := constantRE(m, subs(k=k+i+1, leadcoeff)*(k+1), DF2, k, x);
      fi;
      S := PSInt(S,x);
      S0 := Limit(x^(-i)*DF[0]-S, x=0, DIRECTION);
      S := UpdateCoeff(S0+S, x, i);
      RETURN(S);
    fi;

    S := S + c0*x^i
  od;
  RETURN(S)
end: # constantRE

hypergeomRE := proc(m, R, leadcoeff, DF, k, x)
# solve hypergeometric RE
#
#     a(k+m) = R(k) * a(k)
#
# where leadcoeff was the leading coeffitient of the RE
# and DF is a table where DF[i] = diff(f, x$i)
#
  local fract, i, m0, m1, m2, c0, ck, S, C, DF2, q, R2, a, b, A, B,
      S0, RSOLVE, R1, lc1, lc2, cond, j, Const, g;
  global DIRECTION;
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  _EnvNestLevel := _EnvNestLevel+1:
  if _EnvNestLevel>20 then ERROR(`infinite loop detected`) fi:

  a := factor(leadcoeff):
  b := factor(leadcoeff*R):

  userinfo(3, 'FormalPowerSeries', `RE is of hypergeometric type.`):
  userinfo(3, 'FormalPowerSeries', `Symmetry number m := `, m):
  # userinfo(3, 'FormalPowerSeries', `leadcoeff := `, a):
  userinfo(3, 'FormalPowerSeries', `RE: `, print(a*'a'(k+m)=b*'a'(k)) ):

  A := select(t -> not has(t, I), {solve(a, k)}):

  fract := map(t -> denom(t), A) minus {1}:
  if fract <> {} then q := lcm(op(fract)):
    g := subs(x=x^q, DF[0]):
```

```
    indets(g, (identical(x)`anything)`anything):
    map((t,x) -> t=x^(op(2,op(1,t))*op(2,t)), %, x):
    DF2[0] := eval(subs(%, g)):
    # DF2[0] := simplify(subs(x=x^q, DF[0]), power, 'symbolic'):
    userinfo(3, 'FormalPowerSeries', `RE modified to `, k = k/q):
    userinfo(2, 'FormalPowerSeries', `=> f := `, DF2[0]):
    S := hypergeomRE(q*m, subs(k=k/q, R), subs(k=k/q, leadcoeff), DF2, k, x):
    S := eval(subs(x=x^(1/q), S)):
    S := simplify(S, power):
    RETURN(S)
  fi:
  # {solutions in denR are integer}

  # Problem: Symbole in denR? extract these elements and generate
  # a condition list.
  if A <> {} then
    m1 := min(op(A));
    if type(m1, function) and op(0,m1)='min' then
      m2 := eval(select(type, m1, rational));
      if m2 = infinity then m2 := -m fi:
      userinfo(1, 'FormalPowerSeries', `provided that `, m2, ` <= `, m1);
      m1 := m2;
    fi;
    m1 := m1+m;
    if m1 <> 0 then
      DF2[0] := simplify(x^(-m1)*DF[0], power);
      userinfo(3, 'FormalPowerSeries', `working with x^`,eval(-m1),`*f`);
      userinfo(2, 'FormalPowerSeries', `=> f := `, DF2[0]):
      S := hypergeomRE(m, subs(k=k+m1, R), subs(k=k+m1, leadcoeff), DF2, k, x);
      S := UpdateCoeff(S, x, m1):
      RETURN(S)
    fi
  fi: # { min(op(A)) = -1}

  A := select(type, A, integer):

  if A = {} then m0 := 0
  else m0 := max(op(A))+1;
  fi;
  userinfo(3, 'FormalPowerSeries', `RE valid for all k >= `, m0):

  B := select(t -> not has(t, I), {solve(b, k)}):
  B := select(type, B, integer):

#
# test for essential singularity:

  c0 := Limit(DF[0], x=0, DIRECTION):
  if type(c0,infinity) then
    # now we have either a essential or a logarithmic singularity.
    c0 := limit(x*DF[0], x=0, DIRECTION):
    if type(c0,infinity) then
      ERROR(`essential singularity`) fi:
  fi:
```

```
S := 0: C := {}:
for i from 0 to m0+m-1 do
  if not assigned(DF[i]) then DF[i] := simpl(diff(DF[i-1],x)) fi;

  if not member(i, C) then

    c0 := Limit(DF[i], x=0, DIRECTION)/i!:
    userinfo(3, 'FormalPowerSeries', cat(`a(`,i,`) =`),c0):

    if has(c0, infinity) then
      # SEEMS TO BE A LOGARITHMIC SINGULARITY since the poles have
      # already been removed!!! log(x)*x^i
      DF2[0] := simpl(diff(x^(-i)*DF[0], x)):
      userinfo(3, 'FormalPowerSeries',
        `logarithmic singularity, working with (f*`,x^(-i),`')`):
      userinfo(2, 'FormalPowerSeries',
        `=> f := `,DF2[0]):

      if  type(DF2[0], polynom(anything,x)) then
        S := DF2[0]
      else
        R1  := subs(k=k+i, R);
        lc1 := subs(k=k+i, leadcoeff);
        R2  := normal( subs(k=k+1, R1)*(k+m+1) / (k+1) );
        lc2 := subs(k=k+1, lc1)*(k+1);
        S := hypergeomRE(m, R2, lc2, DF2, k, x);
      fi;
      S := PSInt(S,x);
      eval(subs(Sum=0,S));
      S0 := Limit(x^(-i)*DF[0]-%, x=0, DIRECTION):
      S := UpdateCoeff(S0+S, x, i);
      RETURN(S);
    fi;
    # { c0 is finite }

    cond := A={} or (A intersect {seq(i+j*m, j=0..iquo(max(op(A))-i,m) )}={});
    if  (member(i, B) or (c0 = 0)) and cond then
      userinfo(3, 'FormalPowerSeries', 'a'(i+'j'*m)=0, `for all j>0.`);
      C := C union {seq(i+j*m, j=1..iquo(m0,m)+1)};
      S := S + c0*x^i;
    elif cond then
      C := C union {seq(i+j*m, j=1..iquo(m0,m)+1)};
      ck := traperror(hypergeomRsolve(subs(k=m*k+i, R), k, c0, 'RSOLVE'));
      if ck=lasterror then ERROR(`not a power series`) fi;
      if type(ck, `*`) then Const := select( (t,k)->not has(t,k), ck, k) else Const := 1 fi:
      ck := ck/Const;
      if ck = 0 then S := S + c0*x^i;
      elif RSOLVE = 'finite' then S := S+Const*expand(sum(ck*x^(m*k+i), k=0..infinity))
      else          S := S + Const*Sum(ck*x^(m*k+i), k=0..infinity)
      fi:
    elif c0 <> 0 then # single terms
      S := S + c0*x^i;
    fi:
  fi
od:
```

```
  RETURN(S)
end: # hypergeomRE

UpdateCoeff := proc(S, x, m) local Sx, SSum, C;
# C * Sum(ck*x^k) => C * Sum(ck*x^(k+m))
  if not has(S,x) then
    S*x^m
  elif type(S, `+`) then
    map(procname, S, x, m)
  elif type(S, `*`) then
    Sx := select(has, S, x); C := S/Sx;
    if C <> 1 then
      C*procname(Sx,x,m)
    else # all terms have x
      SSum := select(has, S, Sum); C := S/SSum;
      if type(SSum, `*`) then ERROR(`UpdateCoeff: products of Sums not allowed`)
      elif SSum=1 then simplify(S*x^m, power)
      else C*procname(SSum,x,m)
      fi;
    fi;
  elif type(S, Sum(anything, `=`) ) then
    Sum( simplify(op(1,S)*x^m, power), op(2,S) )
  else
    simplify(S*x^m, power)
  fi
end: # UpdateCoeff

PSInt := proc(S, x) local SSum, kk, Sx, C, S1, T, n, k, xn;
# C * Sum(ck*x^k) => C * Sum(ck*x^(k+1)/(k+1))
  if not has(S,x) then S*x
  elif type(S, `+`) then
    map('procname', S, x)
  elif type(S, `*`) then
    Sx := select(has, S, x); C := S/Sx;
    if C <> 1 then
      C*procname(Sx,x)
    else # all terms have x
      SSum := select(has, S, Sum); C := S/SSum;
      if type(SSum, `*`) then ERROR(`PSInt: products of Sums not allowed`)
      elif SSum=1 then int(S,x)
      else C*procname(SSum,x)
      fi
    fi;
  elif type(S, Sum(anything, `=`) ) then
    S1 := op(1,S); xn := op(select(has, indets(S1, `^`), x)); n := op(2,xn);
    k := op(indets(n,name)); T := 0;
    for kk from 0 while subs(k=kk,n)<0 do
      T := T + simplify(subs(k=kk, S1))
    od;
    n  := subs(k=k+kk, n);
    S1 := subs(k=k+kk, op(1,S));
    int(T,x) + Sum( S1 * n*x^(n-1)/(n-1), op(2,S) )
  else int(S,x)
  fi
```

```
end: # PSInt

unprotect(Pochhammer):
Pochhammer := proc(n,k) local i:
  if n=1/2 then (2*k)!/4^k/k!
  elif type(n,integer) then
    if type(n,posint) then (n+k-1)!/(n-1)!
    elif type(n,negint) then (-1)^k*(-n)! / (-n-k)!
    else 'procname(args)'
    fi
  elif type(n,rational) and denom(n)=2 then
    if  n>0 then Pochhammer(2*n-1,2*k)/(4^k* Pochhammer((2*n-1)/2,k))
    elif n<0 then (-1)^(-n+1/2)*Pochhammer(1/2,-n+1/2) * Pochhammer(1/2,k+n-1/2)
    fi
  elif has(n, I) then 'procname(args)'
  elif type(k, integer) then product(n+i, i=0..k-1)
  else 'procname(args)'
  fi
end:

`simplify/Pochhammer` := proc(e)
  local P, R, k, p, pk, pn, q, qk, qn, r, j;
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;

  if  type(e,`+`) then map(`simplify/Pochhammer`, e)
  elif type(e,`*`) then
    if denom(e) <> 1 then
      `simplify/Pochhammer`(numer(e))/`simplify/Pochhammer`(denom(e))
    else
      P := select(t -> type(t, 'Pochhammer'(anything, anything)), convert(e, set));
      R := e/convert(P,`*`);
      while P <> {} do p := op(1,P); P := P minus {p}; pn := op(1,p); pk := op(2,p);
        for q in P do qn := op(1,q); qk := op(2,q);
          if pk=qk then k := pk;
            if has(pn, I) or has(qn, I) then
              if pn = evalc(conjugate(qn)) then
                P := P minus {q}; p := 1;
                r := Product((evalc('Re'(pn))+'j')^2+evalc('Im'(pn))^2, 'j'=0..k-1);
                R := R*r; break
              fi
            elif pn-qn=1/2 then
              P := P minus {q}; r := Pochhammer(2*qn, 2*k)/4^k; p := 1;
              if type(r, 'Pochhammer'(anything, anything)) then
                P := P union r else R := R*r fi;
              break
            elif qn-pn=1/2 then r := Pochhammer(2*pn, 2*k)/4^k;
              P := P minus {q}; p := 1;
              if type(r, 'Pochhammer'(anything, anything)) then
                P := P union r else R := R*r fi;
              break
            elif (pn=1/3 and qn=2/3) or (pn=2/3 and qn=1/3) then
              r := (3*k)! (k!*27^k);
              P := P minus {q}; p := 1; R := R*r; break
            elif (pn=2/3 and qn=4/3) or (pn=4/3 and qn=2/3) then
              r := (1+3*k)! (27^k*k!);
```

```
            P := P minus {q}; p := 1; R := R*r; break
          elif qn-pn=1/3 and type(pn-1/3, integer) and pn-1/3 <> 0 then
            r := Pochhammer(3*pn,3*k)/(27^k*Pochhammer(pn+2/3,k));
            P := P minus {q}; p := 1; R := R*r; break
          elif pn-qn=1/3 and type(qn-1/3, integer) and qn-1/3 <> 0 then
            r := Pochhammer(3*qn,3*k)/(27^k*Pochhammer(qn+2/3,k));
            P := P minus {q}; p := 1; R := R*r; break
          fi;
        fi;
      od;
      R := R*p
    od; R
  fi
 else e
 fi
end: # `simplify/Pochhammer`

# de2re : converts a simple DE (i.e. a homogeneous linear DE with polynomial coefficients)
# to a recurrence equation:  f(n) = (D@@n)(f)
de2re := proc(de, f, x, a, n) local j, k, X, F, C;
 if type(de, `+`) then map(de2re, de, f, x, a, n)
 elif type(de, f(integer)) then k := op(1,de);
   Pochhammer(n+1, k) * a(n+k)
 elif type(de, name^integer) and op(1,de) = x then
   j := op(2,de); a(n-j)
 elif type(de, `*`) then
   X := select(has, de, x);
   F := select(has, de, f);
   C := de/X/F;
   if X=1 then j := 0
   elif X=x then j := 1
   elif type(X, `^`) then j := op(2,X)
   else ERROR(`assumption 1 violated`) fi;
   if type(F, f(integer)) then k := op(1,F)
   else ERROR(`assumption 2 violated`) fi;
   C * Pochhammer(n+1-j,k) * a(n+k-j)
 else ERROR(`assumption 0 violated`, de)
 fi
end:

hypergeomRsolve := proc(R, k, a0, RSOLVE::name)
# solves the recurrence equation
#
#    a(k+1) = R(k) * a(k),  a(0) = a0
#
 local Re, N, D, C, P, Q, d, n, e, ak, sols, II;
 options `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`;
 Re := normal(R*(k+1)); C := 1; P := NULL; Q := NULL;
 N := numer(Re); if type(N, `*`) then N := convert(N, list) else N := [N] fi;
 D := denom(Re); if type(D, `*`) then D := convert(D, list) else D := [D] fi;
 for n in N do
   if not has(n, k) then C := C*n
   elif type(n, polynom(anything, k)) then
     if type(n, polynom(anything, k)`posint) then e := op(2,n); n := op(1,n) else e := 1 fi;
     if degree(n,k) = 1 then
```

```
        C := C*coeff(n,k,1)^e: P := P, coeff(n,k,0) coeff(n,k,1) $ e
      elif degree(n,k) = 2 then
        C := C*coeff(n,k,2)^e:
        sols := [solve(n,k)]: sols := map(t->-t, sols): P := P, sols[3] $ e, sols[3] $ e
      else ERROR(degree, degree(n,k), `this case not yet implemented`)
      fi
    else ERROR(`don't know what to do with`, n):
    fi:
  od:
  for d in D do
    if not has(d, k) then C := C d
    elif type(d, polynom(anything, k)) then
      if type(d, polynom(anything, k)^posint) then e := op(2,d): d := op(1,d)
      else e := 1
      fi:
      if degree(d,k) = 1 then
        C := C/coeff(d,k,1)^e: Q := Q, coeff(d,k,0)/coeff(d,k,1) $ e:
      elif degree(d,k) = 2 then
        C := C/coeff(d,k,2)^e:
        sols := [solve(d,k)]: sols := map(t->-t, sols):
        Q := Q, sols[3] $ e, sols[3] $ e
      else ERROR(degree, degree(d,k), `this case not yet implemented`)
      fi
    else ERROR(`don't know what to do with`, d):
    fi:
  od:
  P := map(simplify, [P]): Q := map(simplify, [Q]):

  # Falls im Zaehler negative Zahl, dann endliche Reihe !!!
  if select(t -> type(t,integer) and t<=0, P) <> []
  then RSOLVE := 'finite' else RSOLVE := 'infinite' fi:

  P := convert(map((t,k) -> Pochhammer(t,k), P, k), `*`):
  P := simplify(P, 'Pochhammer'):
  Q := convert(map((t,k) -> Pochhammer(t,k), Q, k), `*`):
  Q := simplify(Q, 'Pochhammer'):
  if Q=0 then ERROR(`0 in denominator detected`) fi:
  ak := a0*C^k*P/Q/k!:
  ak := simplify(expand(subs(I=II, ak)), power):

  # simplifications of the result:
  # 1) combine powers like (-1)^k*4^k to (-4)^k
  if type(ak,`*`) then convert(ak,list) else [ak] fi:
  C  := convert(select((t,k) -> type(t, `^`) and has(op(2,t),k), %, k), `*`):
  ak := ak, C * combine(simplify(C)):
  ak := subs(II=I, ak):

  # 2) combine factors like k! * (k+1) to (k+1)!
  simplify(ak, factorial):
end: # hypergeomRsolve

`simplify/factorial` := proc(e)
  local num, den, simp:
  option `Copyright 1993 Dominik Gruntz, Wissenschaftliches Rechnen, ETH Zurich`:
```

```
  if  type(e,`+`) then map(procname, e)
  elif type(e,`*`) then
    num:= numer(e); if type(num,`*`) then num:= convert(num,set) else num:= {num} fi;
    den:= denom(e); if type(den,`*`) then den:= convert(den,set) else den:= {den} fi;
    simp := proc(Nin, Din, Nout, Dout) local s, f,n,d; n := Nin; d := Din;
      s := select(t -> type(t,`!`), n);
      while s <> {} do f := op(1,s); s := s minus {f}; f := op(1,f); # f!
        if member(f-1, n) then
          n := n minus {f!, f-1} union {(f-1)!}; s := s union {(f-1)!};
        elif member(f, d) then
          n := n minus {f!} union {(f-1)!}; d := d minus {f}; s := s union {(f-1)!};
        fi
      od;
      Nout := n; Dout := d;
    end:
    simp(num,den,'num','den');
    simp(den,num,'den','num');
    convert(num, `*`)/convert(den, `*`)
  else e
  fi
end: # `simplify/factorial`

printDE := proc(de, F) local t, n, i, w; global x;
  n := max(op(map(op, indets(de, F(integer)))));
  t := NULL;
  for i from 0 to n do t := t, coeff(de, F(i) ) od;
  w := 0; for i from n by -1 to 0 do w := w + t[i+1]*cat(F,`$i)(x) od;
  w=0
end:

simpl := proc(e) local subspat, f;
  subspat := indets(e, 'trig') union indets(e, 'arctrig');
  subspat := map(f -> f=normal(expand(f)), %);
  subspat := subspat minus select(evalb, %);
  f := subs(subspat, e);
  f := simplify(f, 'arctrig');
  f := simplify(f, 'trig');
end:

Limit := proc() local c0;
  c0 := limit(args);
  if type(c0, range) then
    userinfo(3, 'FormalPowerSeries', `the following limit is a range: `,
      'limit'(args)=c0);
    ERROR(`no initialisation found`)
  fi;
  if has(c0, limit) then
    userinfo(3, 'FormalPowerSeries', `Maple could not do the following limit: `, c0);
    ERROR(`no initialisation found`)
  fi;
  if has(c0, 'undefined') then
    userinfo(3, 'FormalPowerSeries', `the following limit is undefined: `,
     'limit'(args)=c0);
    ERROR(`no initialisation found`)
  fi;
```

```
  c0
end:

# Recursions of the form F(n, X) = A*F(n-1, X)+B*F(n-2, X)
# AS (9.1.27)
Recursion[Hankel1,2] := (n,x) ->
    - Hankel1(n-2,x) + (2*(n-1)/x)*Hankel1(n-1,x):
# AS (9.1.27)
Recursion[Hankel2,2] := (n,x) ->
    - Hankel2(n-2,x) + (2*(n-1)/x)*Hankel2(n-1,x):
# AS (13.4.1)
Recursion[KummerM,3] := (a,b,x) -> 1/(a-1)*
    ((b-a+1)*KummerM(a-2,b,x) + (2*a-2-b+x)*KummerM(a-1,b,x)):
# AS (13.4.15)
Recursion[KummerU,3] := (a,b,x) -> -1/((a-1)*(a-b))*
    (KummerU(a-2,b,x) + (b-2*a+2-x)*KummerU(a-1,b,x)):
# AS (13.4.29)
Recursion[WhittakerM,3] := (n,m,x) -> 1/(2*m+2*n-1)*
    ((3+2*m-2*n)*WhittakerM(n-2,m,x) + (4*n-4-2*x)*WhittakerM(n-1,m,x)):
# AS (13.4.31)
Recursion[WhittakerW,3] := (n,m,x) -> 1/4*
((-9+4*m^2+12*n-4*n^2)*WhittakerW(n-2,m,x) - (8*n-8-4*x)*WhittakerW(n-1,m,x)):
# AS (8.5.3)
Recursion[Legendre_P,3] := (a,b,x) -> 1/(a-b)*
    (-(a-1+b)*Legendre_P(a-2,b,x) + (2*a-1)*x*Legendre_P(a-1,b,x)):
Recursion[Legendre_Q,3] := (a,b,x) -> 1/(a-b)*
    (-(a-1+b)*Legendre_Q(a-2,b,x) + (2*a-1)*x*Legendre_Q(a-1,b,x)):
# AS (22.7)
Recursion[JacobiP,4] := (n,a,b,x) -> 1/(2*n*(a + b + n)*(-2 + a + b + 2*n))*
((2*(1-a-n)*(-1+b+n)*(a+b+2*n)*JacobiP(n-2,a,b,x)) +
((a^2-b^2)*(-1+a+b+2*n)+(-2-a+b+2*n)*(-1+a+b+2*n)*(a+b+2*n)*x)*
JacobiP(n-1,a,b,x)):
Recursion[GegenbauerC,3] := (n,a,x) -> 1/n*(
    -(n+2*a-2)*GegenbauerC(n-2,a,x) + 2*(n-1+a)*x*GegenbauerC(n-1,a,x)):
Recursion[ChebyshevT,2] := (n,x) -> - ChebyshevT(n-2,x) +2*x*ChebyshevT(n-1,x):
Recursion[ChebyshevU,2] := (n,x) -> - ChebyshevU(n-2,x) +2*x*ChebyshevU(n-1,x):
Recursion[Legendre_P,2] := (n,x) -> 1/n*(-(n-1)*Legendre_P(n-2,x) +
(2*n-1)*x*Legendre_P(n-1,x)):
Recursion[LaguerreL,2] := (n,x) -> 1/n*(-(n-1)*LaguerreL(n-2,x) +
(2*n-1-x)*LaguerreL(n-1,x)):
Recursion[LaguerreL,3] := (n,a,x) -> 1/n*
    (-(n-1+a)*LaguerreL(n-2,a,x) + (2*n+a-1-x)*LaguerreL(n-1,a,x)):
Recursion[HermiteH,2] := (n,x) -> -2*(n-1)*HermiteH(n-2,x)+2*x*HermiteH(n-1,x):
Recursion[Bateman, 2] := (n,x) ->
  (Bateman(n-2,x) *(2 - n) + ( 2*(-1 + n) - 2*x)* Bateman(n-1,x))/n:
Recursion[Fibonacci, 2] := (n,x) ->
  x*Fibonacci(n-1,x) + Fibonacci(n-2,x):
Recursion[AiryAi,2] := (n,x) -> x*AiryAi(n-2,x) + (n-2)*AiryAi(n-3,x):
Recursion[ExpIntegralE,2] := (n,x) -> -1/(n-1)*
  (-x*ExpIntegralE(n-2,x)+(2-n+x)*ExpIntegralE(n-1,x)):
Recursion[Abramowitz,2] := (n,x) -> (n-1)*2*Abramowitz(n-2,x)+
  x^2*Abramowitz(n-3,x):

#   old definitions
```

```
RecursionSimplify := proc(e) local eq, L1, L2, L3, L4, L5, F, Nargs, n, a, X;
  eq := e;
  L1 := indets(eq,function);
  L2 := map(t -> [op(0,t), nops(t)], L1); # function name and nargs
  for a in L2 do
    if assigned(Recursion[op(a)]) then
      F := op(1,a); Nargs := op(2,a);

      # select those functions.
      L3 := select((t,F,n) -> op(0,t)=F and nops(t)=n, L1, F, Nargs);
      # separaete in different Arguments, e.g. L(n,X) and L(n,Y)
      L4 := map((t,N)-> [op(2..N,t)], L3, op(2,a));
      for X in L4 do
        L5 := map(t->op(1,t), select((t,X,n)->[op(2..n,t)]=X,L3,X,Nargs));
        while nops(L5) > 2 do
          n := max(op(L5)); if type(n,function) then n := op(1,n) fi;
          if member(n-1,L5) and member(n-2,L5) then
            eq := subs(F(n,op(X))=Recursion[F,Nargs](n,op(X)), eq);
          fi;
          L5 := L5 minus {n}
        od
      od
    fi
  od;
  eq
end:

`diff/Hankel1`:= proc(n,f,x);
  if nargs <> 3 or has(n,x) then
    'diff'(Hankel1(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) * (Hankel1(n-1,f) - (n/f) * Hankel1(n,f))
  fi
end:

`diff/Hankel2`:= proc(n,f,x);
  if nargs <> 3 or has(n,x) then
    'diff'(Hankel2(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) * (Hankel2(n-1,f) - (n/f) * Hankel2(n,f))
  fi
end:

`diff/Legendre_P`:= proc(a,b,f,x) local i;
  for i from 1 to nargs-2 do
    if has(args[i], args[nargs]) then 'diff'(Legendre_P(args[1..nargs-1]),args[nargs]) fi;
  od;
  if nargs = 4 then
    diff(f,x) * 1 /(1-f^2)*((a-b)*Legendre_P(a-1,b,f) - a*f*Legendre_P(a,b,f))
  elif nargs = 3 then
    diff(b,f)*(a*(1-b^2)*Legendre_P(a-1,b)-a*b*(1-b^2)*Legendre_P(a,b))
  else
    'diff'(Legendre_P(args[1..nargs-1]),args[nargs])
  fi
end:
```

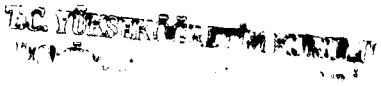
```
`diff/Legendre_Q`:= proc(a,b,f,x):
  if nargs <> 4 or has([a,b],x) then
    'diff'(Legendre_Q(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) * 1/(1-f^2)*((a+b)*Legendre_Q(a-1,b,f) - a*f*Legendre_Q(a,b,f))
  fi
end:

`diff/JacobiP`:= proc(n,a,b,f,x) local i;
  for i from 1 to nargs-2 do
    if has(args[i], args[nargs]) then 'diff'(JacobiP(args[1..nargs-1]),args[nargs]) fi
  od;
  if nargs = 5 then
    diff(f,x) * ((2*(a + n)*(b + n)*JacobiP(n-1, a, b, f))/
    ((a + b + 2*n)*(1 - f^2)) + (n*(a - b - (a + b + 2*n)*f)*
    JacobiP(n, a, b, f))/((a + b + 2*n)*(1-f^2)))
  elif nargs = 3 then
    diff(a,b)*(n/(1-a^2)*JacobiP(n-1,a)-n*a/(1-a^2)*JacobiP(n,a))
  else
    'diff'(JacobiP(args[1..nargs-1]),args[nargs])
  fi
end:

`diff/ChebyshevT`:= proc(n,f,x);
  if nargs <> 3 or has(n, x) then
    'diff'(ChebyshevT(args[1..nargs-1]),args[nargs])
  else diff(f,x) * 1/(1-f^2)*(n*ChebyshevT(n-1,f)-n*f*ChebyshevT(n,f))
  fi
end:

# laguerreL
`diff/LaguerreL`:= proc(n,a,f,x) local i;
  for i from 1 to nargs-2 do
    if has(args[i],args[nargs]) then 'diff'(LaguerreL(args[1..nargs-1]),args[nargs]) fi;
  od;
  if nargs = 4 then
    diff(f,x) * 1/f*(-(n+a)*LaguerreL(n-1,a,f)+n*LaguerreL(n,a,f))
  elif nargs = 3 then
    diff(a,f) *((-n/a)*LaguerreL(n-1,a) + (n,a)*LaguerreL(n,a) )
  else ERROR(`incorrect number of arguments`)
  fi
end:

`diff/HermiteH`:= proc(n,f,x):
  if nargs <> 3 or has(args[3], args[5]) then
    'diff'(HermiteH(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) * (2 * n * HermiteH(n-1, f))
  fi
end:

`diff/ChebyshevU`:= proc(n,f,x)
  if nargs <> 3 or has(args[3], args[5]) then
    'diff'(ChebyshevU(args[1..nargs-1]),args[nargs])
```

```
  else
    diff(f,x) * 1 (1-f^2)*((n-1)*ChebyshevU(n-1,f)-n*f*ChebyshevU(n,f))
  fi
end:

`diff/Bateman`:= proc(n,f,x)
  if nargs <> 3 or has(args[3], args[5]) then
    'diff'(Bateman(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) * ((1-n)*Bateman(n-1,f) + (n-f)*Bateman(n,f))/f
  fi
end:

`diff/GegenbauerC`:= proc(n,a,f,x);
  if nargs <> 4 or has([n,a], x) then
    'diff'(GegenbauerC(args[1..nargs-1]),args[nargs])
  else
    diff(f,x)*1/(1-f^2)*((n+2*a-1)*GegenbauerC(n-1,a,f)-n*f*GegenbauerC(n,a,f))
  fi
end:

`diff/Fibonacci` := proc(k,f,x):
  if nargs <> 3 or has(k, x) then
    'diff'(Fibonacci(args[1..nargs-1]),args[nargs])
  else
    diff(f,x) *
      ((k-1)*f*Fibonacci(k,f)+2*k*Fibonacci(k-1,f))/(f^2+4)
  fi
end:

`diff/ExpIntegralE`:= proc(n,f,x):
  if nargs <> 3 or has(n,x) then
    'diff'(ExpIntegralE(args[1..nargs-1]),args[nargs])
      else
       - diff(f,x) * ExpIntegralE(n-1,f)
    fi
end:

`diff/Abramowitz`:= proc(n,f,x):
  if nargs <> 3 or has(n,x) then
    'diff'(Abramowitz(args[1..nargs-1]),args[nargs])
      else
      -diff(f,x) * Abramowitz(n-1,x)
  fi
end:

# Initial values of special functions
# AS (8.6.1)
Legendre_P := proc(n,m,x)
  if nargs = 3 and x=0 then
    if m=0 then
      cos(n*Pi 2)*n! (2^n*((n 2)!)^2)
    else
      2^m sqrt(Pi)*cos((n-m)*Pi 2)*GAMMA((n-m-1) 2) GAMMA((n-m-2) 2)
    fi
```

```
  elif nargs = 2 and m=0 then
    cos(n*Pi 2)*n! (2 n*((n 2)!) 2)
  else 'procname(args)'
  fi
end:

# AS (8.6.2)
Legendre_Q := proc(n,m,x)
  if x=0 then
   -2 (m-1)/sqrt(Pi)*sin((n+m)*Pi/2)*GAMMA((n+m+1)/2)/GAMMA((n-m+2)/2)
  else 'procname(args)'
  fi
end:

# AS (22.4)
GegenbauerC := proc(n,a,x)
  if x=0 then
   if a=0 then 2*cos(n*Pi/2)/n
   else cos(n*Pi/2)*GAMMA(a+n/2)/(GAMMA(a)*(n/2)!) fi
  else 'procname(args)'
  fi
end:

ChebyshevT := proc(n,x)
  if x=0 then cos(n*Pi/2)
  else 'procname(args)'
  fi
end:

ChebyshevU := proc(n,x)
  if x=0 then cos(n*Pi/2)
  else 'procname(args)'
  fi
end:

LaguerreL := proc(n,a,x)
  if nargs = 3 and x = 0 then binomial(n+a,n)
  elif nargs = 3 and is(a<0) and is(a,integer) then
    # Szegoe, Orthogonal Polynomials, (5.2.1)
    (-1)^(-a)*x^(-a)*(n+a)! n!*LaguerreL(n+a,-a,x)
  elif nargs = 2 and a = 0 then 1
  else 'procname(args)'
  fi
end:

HermiteH := proc(n,x)
  if x=0 then cos(n*Pi/2)*n! (n/2)!
  else 'procname(args)'
  fi
end:

JacobiP := proc(n,a,b,x)
  if nargs = 4 and is(b 0) then
    (-1) n*JacobiP(n,b,a,-x) # Szegoe, Orthogonal Polynomials, (4.2.13)
  elif nargs = 4 and is(a<0) and is(a,integer) then
```

```
                    # Szegoe, Orthogonal Polynomials, (4.22.2)
    binomial(n+b,-a)*((binomial(n,-a))*(1-2*x-1^2)^(-a))*JacobiP(n+a,-a,b,x)
  elif x=1 then binomial(n+a,n)
  elif x=-1 then (-1)^n*binomial(n+b,n)
  else
      'procname(args)'
    fi
end:

Fibonacci := proc(n,x)
  if  n=0 then 0
  elif n=1 then 1
  elif x=0 then sin(n*Pi/2)^2
#   elif x=0 and is(n, even) then 0
#   elif x=0 and is(n, odd ) then 1
  else 'procname(args)'
  fi
end:

Abramowitz := proc(n,x)
  if x=0 then GAMMA((n+1)/2)/2
  else 'procname(args)'
  fi
end:


# save FormalPowerSeries, FPS, SimpleDE, SimpleRE,
#     RationalAlgorithm, ComplexApart,
#     constantRE, hypergeomRE,
#     PS, FindDE, de2re, hypergeomRsolve, UpdateCoeff, PSInt,
#     printDE, simpl, Limit,
#     Pochhammer, `simplify/Pochhammer`, `simplify/factorial`,
#     Recursion, RecursionSimplify,
#     `diff/Hankel1`, `diff/Hankel2`,
#     `diff/KummerM`, `diff/KummerU`,
#     `diff/WhittakerM`, `diff/WhittakerW`,
#     `diff/Legendre_P`, `diff/Legendre_Q`,
#     `diff/JacobiP`, `diff/ChebyshevT`,
#     `diff/LaguerreL`, `diff/HermiteH`,
#     `diff/ChebyshevU`, `diff/Bateman`, `diff/GegenbauerC`,
#     `diff/AiryAi`, `diff/ExpIntegralE`, `diff/Abramowitz`,
#     `diff/Fibonacci`,
#     Legendre_P, Legendre_Q, GegenbauerC, ChebyshevT, ChebyshevU, LaguerreL,
#     HermiteH, JacobiP, Fibonacci, Abramowitz,
#     `FPS.m`:
#
# done:

read "FPS.msl":
#savelib( '_Share',
    'FormalPowerSeries', 'FPS', 'SimpleDE', 'SimpleRE',
     'RationalAlgorithm', 'ComplexApart',
     'constantRE', 'hypergeomRE',
     'PS', 'FindDE', 'de2re', 'hypergeomRsolve', 'UpdateCoeff', 'PSInt',
     'printDE', 'simpl', 'Limit',
```

```
    'Pochhammer', `simplify/Pochhammer`, `simplify/factorial`,
    'Recursion', 'RecursionSimplify',
    `diff/Hankel1`, `diff/Hankel2`,
    `diff/Legendre_P`, `diff/Legendre_Q`,
    `diff/JacobiP`, `diff/ChebyshevT`,
    `diff/LaguerreL`, `diff/HermiteH`,
    `diff/ChebyshevU`, `diff/Bateman`, `diff/GegenbauerC`,
    `diff/ExpIntegralE`, `diff/Abramowitz`,
    `diff/Fibonacci`,
    'Legendre_P', 'Legendre_Q', 'GegenbauerC', 'ChebyshevT', 'ChebyshevU',
    'LaguerreL', 'HermiteH', 'JacobiP', 'Fibonacci', 'Abramowitz',
   `FPS.m`);
#quit
alias( PS = PS ):
alias( ComplexApart = ComplexApart ):
alias( RationalAlgorithm = RationalAlgorithm ):
alias( constantRE = constantRE ):
alias( hypergeomRE = hypergeomRE ):
alias( UpdateCoeff = UpdateCoeff ):
alias( PSInt = PSInt ):
alias( de2re = de2re ):
alias( hypergeomRsolve = hypergeomRsolve ):
alias( printDE = printDE ):
alias( DIRECTION = DIRECTION ):
alias( simpl = simpl ):
alias( FindDE = FindDE ):
alias( Limit = Limit ):
alias( Recursion =  Recursion ):
alias( RecursionSimplify = RecursionSimplify ):
```



ÖZGEÇMİŞ

1975 yılında Kayseri ili Pınarbaşı ilçesinde doğdu. İlk, orta ve lise öğrenimini Ankara'da tamamladı. Gazi Üniversitesi, Eğitim Fakültesi, Matematik Eğitimi Bölümünden 1996 yılında mezun oldu. Aynı yıl Gazi Üniversitesi, Fen Bilimleri Enstitüsünde yüksek lisans öğrenimine başladı. 1998 yılında Çorum ili, Sungurlu İlçesine öğretmen olarak atandı. Yine 1998 yılında Gazi Üniversitesi, Kastamonu Eğitim Fakültesi, Fen Bilimleri Eğitimi Bölümüne Bilgisayar Uzmanı olarak atandı. Halen Gazi Üniversitesi, Kastamonu Eğitim Fakültesi, İlköğretim Bölümü, Fen Bilgisi Öğretmenliği Programında Bilgisayar Uzmanı olarak görev yapmaktadır.