YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SPLİNE FONKSİYONLARI YARDIMIYLA DİFERANSİYEL DENKLEMLERİN YAKLAŞIK ÇÖZÜMÜ

Mat. Müh. Murat YILDIZ

F.B.E. Matematik Mühendisliği Anabilim Dalında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Dr. Abdullah YILDIZ

of. Dr. Mehmet Bayramoğlu

Prof. Dr. A. Yıldız

Prof. Akın Taşdizen

İSTANBUL,1998

İÇİNDEKİLER

SİMGE LİSTESİ

$B_i = B_{i,k,t}$: t düğüm noktaları üzerindeki i. k mertebeli B-spline.

$C^n[a,b]$: {f: [a,b] -> R ; f , n kez sürekli türevi olan fonksiyondur}.

$D^j f$: f fonksiyonunun j. türevi.

$f(a^+)$: $\lim_{h \to 0+} f(a+h)$, sağdan limit.

$jump_a f$: $f(a^+)-f(a^-)$, f ' nin a daki sıçraması.

P_k : k mertebeli polinomlar uzayı.

$P_{k,\xi}$: ξ kırılma noktalarındaki k mertebeli parçalı polinomlar uzayı.

$P_{k,\xi,\nu}$: Süreklilik koşulları ν ile belli olan elemanların oluşturduğu $P_{k,\xi}$ 'deki alt uzay.

R : Reel sayılar kümesi.

[a,b] : $\{x \in R : a \leq x \leq b\}$, kapalı aralık.

(a,b) : $\{x \in R : a < x < b\}$, açık aralık.

$(x)_+$: max {x,0}, uyarlı kuvvet fonksiyonu.

τ , (τ_i) , $(\tau_i)_1^n$, $(\tau_i)_{i=1}^n$ ve $(\tau_1, \ldots, \tau_n)$: n boyutlu bir vektörün çeşitli şekilde gösterimleri.

$\$_{k,t}$: t düğüm noktaları üzerindeki k mertebeli spline'ların lineer uzayı.

$[\tau_i, \ldots, \tau_j]f$: f ' nin $\tau_i, \ldots, \tau_j$ noktalarındaki j-i mertebeli bölünmüş farkı.

$\xi = (\xi_i)_{i=1}^n$: Kırılma noktaları dizisi.

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

Sayfa

ÖZET

Spline fonksiyonları , diferansiyel denklemlerin sayısal çözümlerini elde etmede elverişli fonksiyonlardır. Bu çalışmada , adi diferansiyel denklemlerin yaklaşık-sayısal çözümlerini , spline fonksiyonları kullanarak elde etmek amaçlanmıştır.

Bu amaca yönelik olarak , spline fonksiyonlarının tanımı ve etkin hesaplanması , parçalı fonksiyonların spline'lar ile temsili ve türevlerinin hesaplanması gibi çözüm için gerekli konular açıklanmıştır. Çözümün nasıl yapıldığı teorisiyle birlikte verilmiştir.

Son kısım uygulamalara ayrılmış ve spline yaklaşımı çeşitli denklemlerde uygulanmıştır. Gerçek ve yaklaşık çözümler arasındaki farklar tespit edilerek , sonuçlar yorumlanmıştır.

ABSTRACT

The spline functions are convenient functions to obtain numerical solution of ordinary differential equations. Our goal in this study is to ontain numerical solution of ordinary differential equations by using spline functions approximation.

Because of this , definition of spline function and their efficient evaluation , representation of piecewise polinomials and calculation of their derivatives which are needed for solution are explained. Solution methods are also given together with its theory.

Last section is devoted to applications. Spline approximation is used on several ordinary differential equation. The differences between analytical solutions and numerical solutions are determined and the results are evaluated.

ÖNSÖZ

Diferansiyel denklemlerin sayısal olarak yaklaşık çözümü için son yıllarda bir çok metot geliştirilmiştir. Özellikle elektronik hesaplayıcıların gelişmesi ve hızlarının artması bu alanda daha çok çalışmanın yapılması sonucunu doğurmuştur. **Spline metodu** da bu çalışmaların ürünüdür ve sayısal çözümlemelerde avantajlı bir kullanıma sahiptir.

Çalışmamızın birinci bölümünde tanımlar için gerekli olan bölünmüş farkların özellikleri , noktaların tekrarlanması durumunda ortaya çıkan osculatory interpolasyon , çalışacağımız fonksiyonun yapısını tanıtan parçalı polinomlar konusu özetlenmiştir.

İkinci bölümde konunun temelini oluşturan B-spline bazı tanıtılarak , özellikleri sıralanmıştır. Daha sonra sayısal çözümde kullanılacak olan, özellikle de programlama algoritmasına çok iyi uyum sağlayan ardışık tekrar bağıntıları sunulmuştur. B-spline'ların ve türevlerinin hesaplanması amacıyla hazırlanmış programlar ve bunların ekran görüntüleri verilmiştir.

Üçüncü bölümde Kollakasyon metodunu kullanarak adi diferansiyel denklemlerin B-spline'lar ile çözümü anlatılmıştır.

Son olarak da tüm bu bilgiler ışığında örnek bir diferansiyel denklemi çözen bir program sunulmuş ve denklemin gerçek çözümü ile yaklaşık çözümü karşılaştırılmıştır.

Çalışma içerisinde kaynak kodları verilen programlar **C** dilinde hazırlanmıştır.

Tez çalışmam süresince , bana her konuda yardımcı olan değerli hocam Prof. Dr. Abdullah YILDIZ' a teşekkürü bir borç bilirim.

BÖLÜM I. GİRİŞ

Mühendislikte genel olarak birbirine paralel ve birbiriyle yakın ilişkileri bulunan iki çözüm yolu vardır. Bunlardan birincisi deneysel diğeri ise teorik çalışmalardır. Deney sonuçları çoğu kez tablolar halinde verilen kesikli sayısal değerlerdir. Teorik çalışma sonuçları sürekli değerler olduğu halde çoğu kez deney sonuçlarıyla karşılaştırılması amacıyla kesikli değerler halinde nümerik olarak ifade edilirler. Bazen verilen bir problemin teorik yolla kapalı bir çözümü ya yetersiz kalmakta ya da buna hiç imkan olmamaktadır. O zaman problemi deneylerle çözmek ve kesikli değerler elde etmek yararlı olur. Deneyler ile bulunan sonuçları değerlendirmek veya böyle bir problemi hiç deney gereği olmadan kesikli değerler kullanarak nümerik yolla çözmek ve sonucu yine kesikli değerler halinde vermek mümkündür. Böyle bir yola , **nümerik metot** veya **nümerik analiz yolu** diyoruz. Örneğin ,

$$\frac{dy}{dx} + 2xy = 1$$

diferansiyel denkleminin kapalı bir çözümü

$$y = e^{-x^2}\left(\int_{x_0}^{x} e^{-t^2}\,dt\right)$$

olarak verilebilir. Böyle bir çözüm , sürekli olduğu halde çok az değerlidir. Sonucun kullanılabilmesi için e^{-x^2} ve $\int e^{-x^2}dx$ değerlerini veren özel tabloların kullanılması gerekir. Bazı problemlerde , iyi veya kötü kapalı bir çözüm elde etme yeteneği olmaz. Bugün artık bu tür problemlerin çözümü nümerik metotlarla elde edilmektedir.

Nümerik metotların en büyük sakıncası , elle yapıldığında çok fazla sayıda hatta bazen imkansız ölçüde işlem ve zaman gerektirmesidir. Fakat bugün için elektronik hesaplayıcılar işlem sayısını ve toplam süre problemini büyük ölçüde azalttığı için nümerik metotların gelişmesi ve buna bağlı olarak nümerik hesaplamalarda kesme ve yuvarlatmadan doğan hataların azaltılması ile en iyi yaklaşımla problemlerimizin çözümlenebilmesi için yeni yeni metotlar geliştirilmekte ve çeşitli durumlarda neticelerin karşılaştırılması mümkün kılınmaktadır. İşte gün geçtikçe türlü problemlerde daha çok kullanılan **Spline metodu ,** bu tür metotlardan biridir. Spline fonksiyonları , parça parça (piecewise) veya kesikli polinom tipli fonksiyonların bir sınıfıdır. Polinomlardan biraz daha fazla süreklilik şartını sağlarlar. Bu nedenle polinomların genel bir hali olarak

tanımlanabilirler. Spline fonksiyonunun adı , mühendislikte kullanılan ve spline adı verilen bir mekanik aletten gelmektedir. Bu alet muntazam bir eğri çizebilmek için teknik ressamlar tarafından kulanılmaktadır. Aletin esası , çizilecek eğrinin grafikte işaretlenmiş belirli noktalardan geçmesini sağlamak için kullanılan çelik veya plastik bir şerittir.

1.1 Spline İnterpolasyonu

Bir Spline fonksiyonu kesin süreklilik şartlarında birleştirilmiş polinom parçalarından oluşur.

$t_0 , t_1 , ... , t_n$ (n+1) nokta olmak üzere bir fonksiyon tablo formunda verilsin ve $t_0 < t_1 < ..... < t_n$ koşulu sağlansın. Bu noktalar düğüm noktaları / knots olarak isimlendirilir.

k, bir tam sayı ($k \geq 0$) olmak üzere; $t_0, t_1, ... , t_n$ noktalarına göre k. dereceden bir Spline fonksiyonu öyle bir S fonksiyonudur ki;

(i) S, $[t_{i-1} , t_i]$ yarı açık aralığında derecesi k'dan küçük veya eşit ($\leq k$) bir polinomdur.

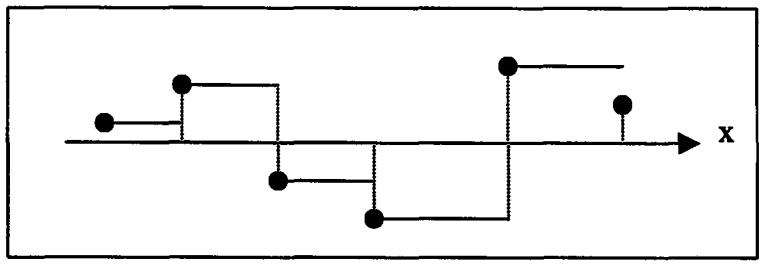
(ii) S'in $[t_0 , t_n]$ aralığında (k – 1). mertebe türevleri süreklidir.

O halde SPLİNE FONKSİYONU parçalı fonksiyonlardır ki; verilen aralıkta (k-1). mertebe türevleri süreklidir.

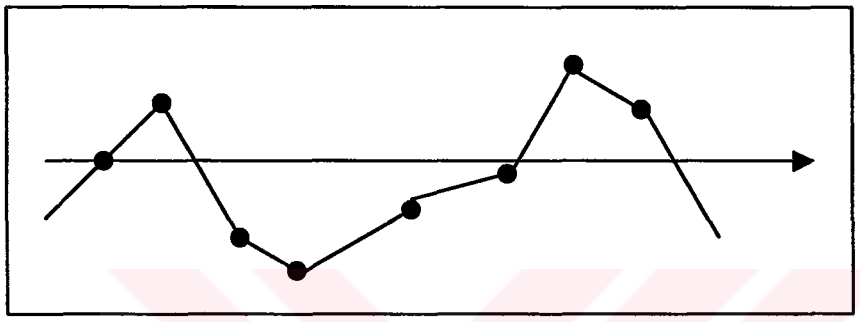
0. dereceden Spline fonksiyonu , parça parça (piecewise) sabit fonksiyonlardır Bir "0. dereceden Spline fonksiyonu" nun açık formu aşağıdaki gibidir:

$$S(x) = \begin{cases} S_0(x) = c_0 & x \in [t_0, t_1) \\ S_1(x) = c_1 & x \in [t_1, t_2) \\ \vdots & \vdots \\ \vdots & \vdots \\ S_{n-1}(x) = c_{n-1} & x \in [t_{n-1}, t_n) \end{cases}$$

Bu aralıklar $[t_{i-1} , t_i]$ birbirini bölmezler. Altı noktadan oluşan tipik bir Spline fonksiyonu Şekil 1.1 de gösterilmiştir.



Şekil 1.1 0. dereceden bir spline



Şekil 1.2 1. Dereceden bir spline

Şekil 1.2 de dokuz noktadan (node) oluşan tipik bir 1. dereceden Spline fonksiyonu görülmektedir. Açık formu ise aşağıdaki gibidir:

$$S(x)=\begin{Bmatrix} S_0(x)=a_0x+b_0 & x\in[t_0,t_1) \\ S_1(x)=a_1x+b_1 & x\in[t_1,t_2) \\ \vdots & \vdots \\ \vdots & \vdots \\ S_{n-1}(x)=a_{n-1}x+b_{n-1} & x\in[t_{n-1},t_n) \end{Bmatrix}$$

Kübik Spline'lar

Aşağıdaki tablo değerlerinin verildiğini varsayalım :

x	t_0	t_1	...	t_n
y	y_0	y_1	...	y_n

Kübik Spline (S) , tabloyu interpole etmek için kurulmuştur. Her aralıkta $[t_0 , t_1], [t_1 , t_2] , \ldots\ldots , [t_{n-1} , t_n]$ S farklı polinomlarda verilmiştir. $S_i , [t_i , t_{i-1}]$ aralığında bir kübik spline olsun. Böylece

$$S(x) = \begin{Bmatrix} S_0(x) & x \in [t_0, t_1) \\ S_1(x) & x \in [t_1, t_2) \\ \vdots & \vdots \\ \vdots & \vdots \\ S_{n-1}(x) & x \in [t_{n-1}, t_n) \end{Bmatrix}$$

S_{i-1} ve S_i Polinomları t_i noktasında aynı değeri interpole etmiştir. Yani

$$S_{i-1}(t) = y_i = S_i(t_i) \qquad (1 \le i \le n-1)$$

Böylece S otomatik olarak sürekli olur. S′ ve S″ nün sürekli olduğunu varsayalım. Bu şartlar kübik spline fonksiyonunun türevinde kullanılacaktır.

S , S′ ve S″ nün sürekliliği acaba kübik spline ' ın açklanmasında yeteri kadar şartları sağlıyor mu? Kübik polinomda 4n tane bilinmeyen var (n tana polinom , her polinomda 4 bilinmeyen). Her $[t_i, t_{i+1}]$ alt aralığında iki interpolasyon denklemi vardır. $S(t_i) = y_i$ ve $S(t_{i+1}) = y_{i+1}$ olmak üzere bize 2n adet denklem verir. S′ nün sürekliliği her bir nod'da bir denklem verir. $S'_{i-1}(t_i) = S'_i(t_i)$, (n-1) ilave denklem veriyor. Aynı şekilde S″ nün sürekliliği de (n-1) denklem verir. Böylece (4n-2) denklem (4n) bilinmeyen için belirlenir. Geriye kalan iki şart da keyfi olarak seçilir.

1.2. Bölünmüş Farklar ve Özellikleri

Tanım 1.1 ; $t_i,\ldots,t_{i+k}$ noktalarında, bir g fonksiyonunun k. bölünmüş farkı, bu noktalarda g fonksiyonuna uyan k+1 mertebeli polinomun baş katsayısıdır. Bu katsayı notasyon olarak

$$[t_i,\ldots,t_{i+k}]$$

şeklinde gösterilir.

Özellikleri:

(i) i=k ve k+1 için $p_i \in P_i$ polinomları tl,...,ti noktalarında g' ye uygun ise,

$$P_{k+1}(x) = P_k(x) + (x-t_1) \ldots\ldots (x-t_k)[t_1,\ldots\ldots,t_{k+1}]g$$

olmaktadır. Buna göre n. mertebeden bir polinom

$P_n(x) = [t_1]g + (x- t_1)[t_1, t_2]g + (x- t_1)(x- t_2)[t_1, t_2, t_3]g + \ldots\ldots+(x- t_1)\ldots\ldots(x- t_{n-1})[t_1,\ldots\ldots, t_n]g$

şeklinde ifade edilebilir. Bu yazılışa Newton Formu denmektedir.

(ii) $[t_i,\ldots\ldots,t_{i+k}]g$ argümanlarına göre simetrik bir fonksiyondur.

(iii) k. bölünmüs fark operatörü lineerdir.

(iV) (Leibniz Formülü) Eğer her x için, f(x)=g(x).h(x) ise

$$[t_i,\ldots\ldots,t_{i+k}]f = \sum_{r=i}^{i+k} ([t_i,\ldots\ldots,t_r]g)([t_r,\ldots\ldots,t_{i+k}]h)$$

dir.

(V) g , derecesi ≤ k olan bir polinom ise $[t_i,\ldots\ldots,t_{i+k}]g$ sabit bir fonksiyondur. $g \in P_k$ ise, $[t_i,\ldots\ldots,t_{i+k}]g = 0$ olur.

(Vi) $g \in C^{(k)}$ durumunda $[t_i,\ldots\ldots,t_{i+k}]g$, (t_i,t_{i+k}) aralığında k+1 argumanının sürekli bir fonksiyonudur.

(Vii) $g \in C^{(k)}$ ise, $[t_i,\ldots\ldots,t_{i+k}]g = g^{(k)}(\xi) / k!$ olacak biçimde bir ξ vardir. Bu ξ , $\tau_i,\ldots\ldots \tau_{i+k}$ yi içeren en dar aralıktadır.

(Viii) Bölünmüş farklar aşağıdaki biçimde hesaplanmaktadır.

$$[t_i,\ldots,t_{i+k}] = \begin{cases} g^{(k)}(\tau_i)/k!, & t_i = t_{i+1} = \ldots = t_{i+k} \text{ ve } g \in C^{(k)} \\ \dfrac{[t_i,..,t_{r-2},t_{r-1},..t_{i+k}]g - [t_i,..,t_{s-1},t_{s+1},..t_{i+k}]g}{t_s - t_r}, & t_r \text{ ve } t_s, t_i..,t_{i+k} \text{ dizisinde farklı iki nokta olmak koşulu ile} \end{cases}$$

I.3. Tekrarlı(Osculatory) İnterpolasyon

Tanım 1.1'e göre 1. ve 2. derece bölünmüs farklar hesaplanacak olursa,

$$[t_1]g = g(t_1)$$

$$[t_1,t_2] = \frac{g(t_1)-g(t_2)}{t_1-t_2}$$

elde edilir. Eger g , birinci türevi sürekli bir fonksiyon ise bu durumda, $t_2 > t_1$ için

$$\lim_{t_2 \to t_1} \frac{g(t_1)-g(t_2)}{t_1-t_2} = g'(t_1)$$

olduğu da bilinmektedir. Bu da

$$p(x) = g(t_1) + (x-t_1)\, g'(t_1)$$

anlamındadır. Yani $t_1 = t_2$ durumunda bölünmüş fark,

$$[t_1 , t_2]g = g'(t_1)$$

olarak ifade edilmektedir.

Tanım 1.2; Bu şekilde noktaların tekrarı halinde elde edilen interpolasyona tekrarlı("osculatory") interpolasyon denir.

1.4. Parçalı Polinomlar

Tanım 1.3; $\xi = (\xi_i)_1^{\iota+1}$ kesin artan bir noktalar dizisi olsun. k pozitif bir tamsayı ve $p_1, \ldots\ldots, p_\iota$ ι tane k mertebeli polinom dizisi ise k. mertebeden f parçalı polinomu

$$pp(x) = p_i(x) , x \in [\xi_i, \xi_{i+1}] , i = 1,2, \ldots\ldots, \iota$$

şeklindedir. Fonksiyonun R üzerinde tanımlı olması için ilk ve son polinom parçaları

$$pp(x) = f(x) = \begin{cases} P_1(x), & x < \xi_1 için \\ P_\ell(x), & \xi_{\ell+1} < x \end{cases}$$

şeklinde düşünülebilir. $\xi_2 , \ldots\ldots , \xi_\iota$ iç noktalarında fonksiyonun tanımlı olabilmesi için

$$f(\xi_i^-) = P_{i-1}(\xi_i) \text{ ve } f(\xi_i^+) = P_i(\xi_i)$$

değerlerinden biri seçilir. Burada sağdan süreklilik seçilmistir. Yani,

$$f(\xi_i) = f(\xi_i^+) \quad , \; i = 2,3, \ldots\ldots , \iota \text{ için}$$

olarak alınacaktır.

k. mertebeden, $\xi = (\xi_i)_1^{\iota+1}$ kırılma noktalarına sahip tüm parçalı polinomların oluşturduğu küme, $P_{k,\xi}$ ile ifade edilecektir. $P_{k,\xi}$, k.ι boyutlu bir lineer uzaydır. Her bir eleman ι polinomdan oluşmaktadır ve herbir polinom parçası k katsayıya sahiptir. Böylece $P_{k,\xi}$, uzayının ι kez tekrarının bir bileşimidir.

Parçalı polinomun j. türevi Djf , f ' yi oluşturan polinom parçalarının j. türevleri ile oluşan, f ile aynı kırılma noktalarına sahip (k-j). mertebeden bir parçalı polinomdur. Koşul olarak;

(i) Polinom parçalarının sayısı ve mertebe için ι ve k tam sayıları,

(ii) Kırılma noktaları olarak, artan değerlere sahip $\xi_1, \xi_1, \ldots\ldots , \xi_{\iota+1}$ dizisi

(iii) Kırılma noktarındaki sağ taraf türev değerlerini veren bir $c = (c_{ji})$, $j=1..k$; $i=1..\iota$ matrisi,

$$c_{ji} = D^{j-1}f(\xi_i^+) \quad j = 1, \ldots\ldots , k \; ; \; i = 1, \ldots\ldots , \iota$$

değerlerinin bilinmesi durumunda $f \in P_{k,\xi}$ için parçalı polinom ve türevlerinin tanımı

$$D^j f(x) = \sum_{m=j}^{k-1} C_{m+1,j}(x-\xi_i)^{m-j} / (m-j)!, \quad j = 0,1,\ldots.,k$$

şeklindedir. Burada i değeri ,

$$i = 1 \quad , x < \xi_2 \text{ için ,}$$

$$1 < i < \iota \quad , \xi_i \le x \le \xi_{i+1} \text{ için ,}$$

$$i = \iota \quad , \xi_\iota \le x \text{ için}$$

olarak seçilecektir.

Tanım 1.4; $P_{k,\xi,\nu}$ uzayı :

$f \in P_{k,\xi}$ fonksiyonu için ξ_i iç düğüm noktalarında sağ ve sol limitlerin farkı olarak sıçrama miktarı("jump") tanımlayalım ve bunu

$$jump_{\xi i} f = f(\xi_i^+) - f(\xi_i^-)$$

şeklinde yazalım. Eğer

$$jump_{\xi i} D^{j-1} f = 0 \quad (j=1,2,\ldots..,\nu_i \text{ ve } i=2,3,\ldots..,\iota)$$

ise, f ' nin kendisinin ve türevlerinin sürekliliğini ifade etmiş oluruz.

$\nu = (\nu_i)_2^\iota$ negatif olmayan tam sayıların oluşturduğu bir vektör olsun. ν_i değerleri, ξ_i noktalarındaki süreklilik koşulunun mertebesini belirten bir sayıdır. Özel olarak $\nu_i = 0$ ifadesi, ξ_i noktasında herhangi bir süreklilikten söz edilmemesi anlamını taşımaktadır. Böylece $P_{k,\xi}$ uzayının bir alt kümesi olarak $P_{k,\xi,\nu}$ tanımlanmaktadır. Bu uzay; uyarlı kuvvet fonksiyonlarını da kullanarak

$$f(x) = \sum_{j<k} f^{(j)}(\xi_1)(x-\xi_1)^j / j! + \sum_{i=2}^{1} \sum_{j<k} (jump_{\xi_i} D^j f)(x-\xi_i)_+^j / j!$$

şeklinde ifade edilmektedir.

Bu tanıma göre yapılan bazı hesaplamalar için problem kötü-koşullu olmaktadır. Katsayılardaki küçük bir değişim kendisinden sonra gelen terimleri olumsuz etkilemektedir. Ayrıca ξ dizisinin düzgün dağılmaması durumunda, baz fonksiyonları

sanki lineer bağlıymış gibi olabilmektedir. Sözü edilen sakıncaları ortadan kaldırmak, ancak B-spline temsilleri ile mümkün olmaktadır.

BÖLÜM II. B-SPLİNE'LAR İÇİN ETKİN HESAPLAMA YÖNTEMİ

2.1. B-Spline Baz Fonksiyonlarının tanımı

Reel sayı doğrusu üzerinde

$$\prod = \{t_i\}_{i=-\infty}^{\infty}, \quad t_i \le t_{i+1} \qquad tüm \quad i'ler \quad için$$

bölüntüsünü ele alalım. B-spline'ların yerel olma özellikleri nedeni ile çalışma sonlu bir aralığın sonlu bir bölüntüsü üzerinde de yapılabilir. k pozitif tam sayı olmak üzere,

$$(x-t)_+^{k-1} = \begin{cases} (x-t)^{k-1} & ,x \ge t \\ 0 & ,x < t \end{cases}$$

şeklindeki uyarlı kuvvet fonksiyonu göz önüne alınsın. Bu fonksiyon vasıtası ile aşağıdaki önemli tanımı verelim.

Tanım 2.1; $t = (t_j)_i^{i+k}$ azalmayan bir dizi olmak üzere, t düğüm noktalarındaki k. mertebeden (normalize edilmiş) i. B-spline fonksiyonu $B_{i,k,t}$ şeklinde yazılır ve

$$B_{i,k,t}(x) = (t_{i+k} - t_i)[t_i, \ldots\ldots, t_{i+k}](t-x)_+^{k-1} \;, \forall x \in R$$

olarak tanımlanır.

Tanımda x ve t gibi iki değişkene sahip olan $(t-x)_+^{k-1}$ fonksiyonunun k. bölünmüş farkı, t değişkenine göre alınır. Sonuç x değerine bağlı bir fonksiyondur. Eğer k>1 ve $\prod$ bölüntüsünde en çok k-1 ardışık t_i tekrarlanıyorsa $B_{i,k,t}$ fonksiyonu iyi-tanımlı sürekli fonksiyondur. Aksi halde katlı noktada

$$\frac{d^{k-1}}{dx^{k-1}}(x-t)_+^{k-1}$$

türevinde süreksizlik görülür. Bu durum ortaya çıktığında , tanım $x = t_i$ için süreksiz, $x \neq t_i$ için anlamlı olacak biçimde düşünülür. Örneğin k=1 durumunda

$$B_{i,1}(x) = \begin{cases} 1 & ,t_i \le x < t_{i+1} \quad için \\ 0 & ,diğer \quad durumlarda \end{cases}$$

dir.

Tanım 2.2; $\$_{k,t}$ spline uzayı:

t düğüm noktaları için k. mertebeden bir spline fonksiyonu, aynı düğümlere sahip k. mertebeden B-spline'ların lineer bir kombinasyonudur. Bu yapıdaki tüm fonksiyonların kümesi $\$_{k,t}$

$$\$_{k,t} = \left\{ \sum_i \alpha_i B_{i,k,t} : \quad \alpha_i \quad reel, \quad tüm \quad i'ler \quad için \right\}$$

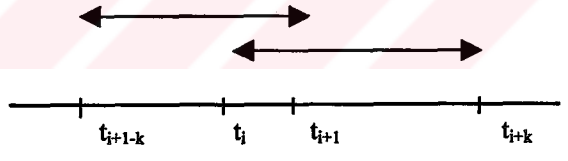
dir.

$$F(x) = \sum_i \alpha_i B_{i,k}(x), \quad x \in [t_j, t_{j+1}]$$

gibi bir spline fonksiyonunu hesaplamak için k adet $B_{i,k}(x)$, (i=j-k+1,....,j) hesaplanması gerekmektedir.

2.2 B-Spline Baz Fonksiyonlarının Özellikleri

(i) $B_i = B_{i,k,t}$ fonksiyonu $x \in [t_i, t_{i+k}]$ için sıfırdır. Yani sonlu aralıklı bir desteğe sahiptir. Bunun sonucunda $[t_j, t_{j+1}]$ aralığında yer alabilecek sıfırdan farklı B-spline sayısı k tanedir.



Şekil 2.1

Şekil 2.1, (t_i, t_{i+1}) aralığında, sıfırdan farklı olan en sağdaki ve en soldaki B-spline fonksiyonları için destek bölgesini göstermektedir.

(ii) Bir destek bölgesindeki k adet spline toplamı

$$\sum_i B_i(x) = \sum_{i=j+1-k}^{j} B_i(x)$$

$$= \sum_{i=j+1-k}^{j} [t_{i+1}, \ldots, t_{i+k}](.-x)_+^{k-1} - \sum_{i=j+1-k}^{j} [t_i, \ldots, t_{i+k-1}](.-x)_+^{k-1}$$

$$= [t_{j+1}, \ldots, t_{j+k}](.-x)_+^{k-1} - [t_{j+1-k}, \ldots, t_i](.-x)_+^{k-1}$$

$= 1+0$

şeklinde hesaplanabilir. Buna göre

$$\sum_i B_i(x) = \sum_{i=r+1-k}^{s-1} B_i(x) = 1, \quad tüm \quad t_r < x < t_s \quad için$$

eşitliği bulunmaktadır.

(iii) $t_i < x < t_{i+k}$ için $B_i(x) > 0$ 'dir. Bu özellik sonraki bölümde ispatlanacaktır.

2.3. Ardışık Tekrar Bağıntısı

B-spline'ların, tanımdan yararlanılarak yapılan hesaplamalar sırasında, bölünmüş farkların birbirinden çıkartılması, bazı anlamlı basamakların kaybolmasına neden olmaktadır. Üstelik katlı noktalar için özel hesaplamalar gerekmektedir. Bütün bu dezavantajları ortadan kaldırmak için ardışık tekrar bağıntısı kullanmak uygun olmaktadır.

$$(t-x)_+^{k-1} = (t-x)(t-x)_+^{k-2} \qquad (2.1)$$

çarpımının k. bölünmüş farkına, Leibniz Formülü [Bölüm 1.2'deki özellik (iv)] uygulanarak ardışık tekrar bağıntısı elde edilmektedir. (2.1) deki çarpıma Leibniz Formülü uygulanacak olursa,

$$\begin{aligned}[t_i,\ldots\ldots, t_{i+k}](.-x)_+^{k-1} &= [t_i](.-x)[t_i,\ldots\ldots, t_{i+k}](.-x)_+^{k-2} \\ &+[t_i, t_{i+1}](.-x)[t_{i+1}, \ldots\ldots, t_{i+k}](.-x)_+^{k-2} \\ &+[t_i, t_{i+1}, t_{i+2}](.-x)[t_{i+2}, \ldots\ldots, t_{i+k}](.-x)_+^{k-2} \\ &+ \ldots\ldots\end{aligned}$$

elde edilir. (t-x) birinci dereceden bir polinom olduğu için birinci bölünmüş farkı sabit, daha sonraki farkları ise sıfır olacaktir. Buna göre yukarıdaki eşitlik;

$$[t_i,\ldots\ldots, t_{i+k}](.-x)_+^{k-1} = [t_i - x)[t_i,\ldots\ldots, t_{i+k}](.-x)_+^{k-2} + 1.[t_{i+1}, \ldots\ldots, t_{i+k}](.-x)_+^{k-2}$$

olarak düzenlenir. Diğer taraftan,

$$[t_i,\ldots,t_{i+k}]g = \frac{[t_{i+1},\ldots,t_{i+k}]g - [t_i,\ldots,t_{i+k-1}]g}{t_{i+k} - t_i}$$

olarak hesaplanmaktadır. Buna göre,

$$[t_i,.....,t_{i+k}](.-x)_+^{k-1} = \frac{(t_i - x)}{t_{i+k} - t_i}[t_{i+1},....,t_{i+k}](.-x)_+^{k-2} - \frac{(t_i - x)}{t_{i+k} - t_i}[t_i,....,t_{i+k-1}](.-x)_+^{k-2}$$

$$+[t_{i+1},.....,t_{i+k-1}](.-x)_+^{k-2}$$

$$=[\frac{t_i - x}{t_{i+k} - t_i} + 1][t_{i+1},.....,t_{i+k}](.-x)_+^{k-2} - [\frac{t_i - x}{t_{i+k} - t_i} + 1][t_i,.....,t_{i+k-1}](.-x)_+^{k-2}$$

$$[t_i,.....,t_{i+k}](.-x)_+^{k-1} = \frac{(x - t_i)}{t_{i+k} - t_i}[t_i,....,t_{i+k-1}](.-x)_+^{k-2} + \frac{t_{i+k} - x}{t_{i+k} - t_i}[t_{i+1},.....,t_{i+k}](.-x)_+^{k-2}$$

bulunur.

En son yazılan eşitlik B-spline tanımına göre yeniden düzenlenirse,

$$\frac{B_{i,k}}{t_{i+k} - t_i} = \frac{(x - t_i)}{t_{i+k} - t_i}\frac{B_{i,k-1}}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - x)}{t_{i+k} - t_i}\frac{B_{i+1,k-1}}{t_{i+k} - t_{i+1}} \qquad (2.2)$$

şeklindeki, ardışık tekrar bağıntısı elde edilmiş olur. (2.2) bağıntısındaki katsayı fonksiyonlarının toplamı her zaman 1 dir.

$$\frac{x - t_i}{t_{i+k} - t_i} + \frac{t_{i+k} - x}{t_{i+k} - t_i} = 1$$

Ayrıca $t_i < x < t_{i+k}$ durumunda tüm katsayı fonksiyonları pozitiftir. Bu durumda eğer, $B_{j,k-1}(x) > 0$, $t_i < x < t_{j+k-1}$, (tüm j 'ler için) olduğu biliniyorsa,

$$B_{i,k}(x) > 0, \quad t_i < x < t_{i+k}$$

sağlanmış olur. Bunun için

$$B_{j,1}(x) = \begin{cases} 1 & ,t_j \le x < t_{j+1} \quad için \\ 0 & ,diğer \quad durumlarda \end{cases} \qquad (2.3)$$

tanımı ile başlayıp, tüme varım ile ispat kolayca gösterilebileceği gibi, B-spline'lar kendi destek bölgeleri üzerinde pozitiftirler.

(2.2) deki ardışık tekrar bağıntısı

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i}B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}}B_{i+1,k-1}(x) \qquad (2.4)$$

şeklinde yeniden yazılabilir ve (2.3) ile başlamak üzere her bir $B_{i,k}(x)$, pozitif niceliklerin pozitif olan lineer kombinasyonları biçiminde ardışık olarak hesaplanabilir.

$B_{i,k}$ tanımı, kırılma noktaları t dizisi içinde olan k mertebeli bir parçalı polinom olarak bilinmektedir. Ancak (2.4)'deki tanımda düzgünlük özellikleri kolayca görülmemektedir. Gerçekte ise, ardışık tekrar bağıntısındaki sonuç $B_{i,k-1}$ ve $B_{i+1,k-1}$, ' nin

özel bir kombinasyonudur. Öyle ki, elde edilen fonksiyon, bileşenlerinden bir fazla sürekli türeve daha sahip olmaktadır.

2.4. B-Spline Fonksiyonlarının Hesaplanması ve BSPLVB.C Programı

$t_i < t_{i+1}$ ve $x \in [t_i, t_{i+1}]$ olsun. x noktasında, sıfır olmayan B-spline'lar (2.4) bağıntısına göre aşağıdaki biçimde bir üçgen dizi oluşturmaktadır.

$$
\begin{array}{ccccccccc}
 & & & & 0 & & & & \\
 & & & 0 & & & & & \\
 & & \cdot & & & B_{i-k+1,k} & & & \\
 & 0 & \cdot & B_{i-k+2,k-1} & & & & & \\
0 & & \cdot & & B_{i-k+2,k} & & & & \\
 & B_{i-2,3} & \cdot & B_{i-k+3,k-1} & & & & & \\
B_{i-1,2} & & \cdot & & B_{i-k+3,k} & & & & \\
 & B_{i-1,3} & \cdot & & & & & & \\
B_{i,1} & & \cdot & & & & & & \\
 & B_{i,2} & \cdot & & & & & & \\
0 & & \cdot & B_{i-1,k-1} & & & & & \\
 & B_{i,3} & \cdot & & B_{i-1,k} & & & & \\
0 & & \cdot & & & & & & \\
 & 0 & \cdot & B_{i,k-1} & & & & & \\
 & & \cdot & & B_{i,k} & & & & \\
 & & & 0 & & & & & \\
 & & & & 0 & & & &
\end{array}
$$

Açıkça görülüyor ki, bu değerler sütunlar halinde kolayca üretilir. $B_{i,j}$ ve $B_{i-j+1,j}$ hesaplanırken, soldaki komşu değerlerden birinin sıfır olacağına dikkat edilmelidir. j sayıdaki $B_{i-j+1,j}(x)$,....., $B_{i,j}(x)$ değeri hesaplandıktan sonra bir $b = (b_r)_1^j$ vektöründe saklanır. $B_{i-j,j+1}(x)$,....., $B_{i,j+1}(x)$ değerlerinin saklandığı bir sonraki $b^1 = (b_r^1)_1^{j+1}$ vektörü bu kez j+1 elemanlıdır ve

$$b_r^1 = (x - t_{i+r-1-j})\frac{b_{r-1}}{t_{i+r-1} - t_{i+r-1-j}} + (t_{i+r} - x)\frac{b_r}{t_{i+r} - t_{i+r-j}} \qquad (2.5)$$

r=1,...j+1 ve $b_0 = b_{j+1} = 0$ koşulları ile hesaplanır. Daha etkin bir hesaplama için

$$\delta_r^{SOL} = x - t_{i+1-r}, \quad \delta_r^{SAĞ} = t_{i+1} - x \quad ; \quad r = 1,.....,k-1$$

değerleri önce hesaplanır. Buna göre (2.5) bağıntısı

$$b_r^1 = \delta_{j+2-r}^{SOL} \frac{b_{r-1}}{\delta_{r-1}^{SAĞ} + \delta_{j+2-r}^{SOL}} + \delta_r^{SAĞ} \frac{b_r}{\delta_r^{SAĞ} + \delta_{j+1-r}^{SOL}}, \quad r = 1,\ldots..,j+1 \quad (2.6)$$

şeklinde düzenlenir. Ancak unutulmaması gereken şey $t < t_{i+1}$ olacağıdır, böylece payda sıfır olmaktan korunur.

Bu algoritmayı kullanarak C dilinde yazılmış bir program ve ekran görüntüleri aşağıda verilmiştir.

```
/* BSPLVB.C */

# include "stdio.h"
# include "conio.h"
# include "math.h"

 int *bsplvb(float t2[] , int jhigh , int index , float x , int left);
 int interv(int LXT,float X,float XT[]);
 float biatx[100];

main() {

 int lxt,left,index,jhigh,i,j,k,p,index = 1;
 float deltal[100],deltar[100],saved,term,bas,top=0.0,x;
 float t[] = {-1.0,0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0,12.0};

 clrscr();
 printf("\nMertebeyi girin = ");scanf("%d",&jhigh);
 printf("\nHesaplanacak noktayı girin = ");scanf("%f",&x);
 left = interv( 13 , x , t );
 p = bsplvb( t , jhigh , index , x , left );

 /* Sonuçlar */

 clrscr();
```

```
 printf("B-Spline değerleri :\n");
 top = 0.0;
 for (i=1 ; i <= jhigh ; i++)
 {
    top += biatx[i];
    printf(" %3.3f ",biatx[i]);
  }
 printf("   %3.3f",top);
 getch();
}

 int *bsplvb(float t2[] , int jhigh , int index , float x , int left)
 {

 int lxt,i,j = 1,jp1,k;
 float deltal[100],deltar[100],saved,term,bas,top=0.0;

  switch ( index ) {
   case 1 : goto index1;
   case 2 : { j = jhigh - 1; goto index2; }
  }

  index1 :

   j = 1;
   biatx[1] = 1;
   if ( j >= jhigh ) return;

  index2 :
  do
  {
   jp1 = j + 1;
   deltar[j] = t2[left-1+j] - x;
```

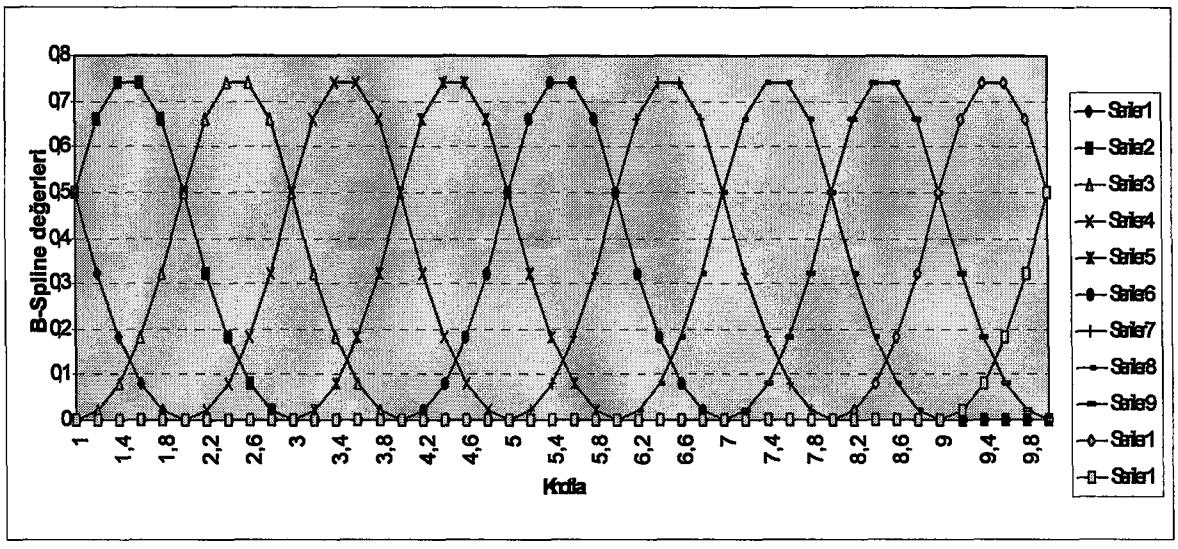
```
    deltal[j] = x - t2[left-j];
    saved = 0;
    for (i=1 ; i <= j ; i++)
    {
     term = biatx[i] / (deltar[i] + deltal[jp1-i]);
     biatx[i] = saved + deltar[i] * term;
     saved = deltal[jp1-i] * term;
    }
    biatx[jp1] = saved;
    j = jp1;
   }
  while ( j < jhigh );
  return;
}

int LXT,ILO,IHI,ISTEP;
float X,XT[100],MIDDLE;

int interv(int LXT,float X,float XT[]) {

  ILO = 1;
  IHI = ILO +1;

  if ( IHI < LXT )
  {
    if ( X >= XT[IHI-1] ) goto SUB40;
   }
 else
 {
    if ( X >= XT[LXT-1] )  return LXT;
     else if ( LXT <= 1 )  return 1;
   }
```

```
ILO = LXT -1;
IHI = ILO;

if ( X >= XT[IHI-1] ) goto SUB40;
if ( X >= XT[ILO-1] ) return ILO;
ISTEP = 1;
do {
   IHI = ILO;
   ILO = IHI - ISTEP;
   if ( ILO <= 1 )
    {  ILO = 1;
      if ( X < XT[0] ) return 1;
      goto SUB50;
    }
   }
  while (1);

SUB40 :

 ISTEP = 1;
 do {
    ILO = IHI;
    IHI = ILO + ISTEP;
    if ( IHI >= LXT )
     {
      if ( X >= XT[LXT-1] ) return LXT;
      IHI = LXT;
      goto SUB50;
     }

    if ( X < XT[IHI-1] ) goto SUB50;
```

```
      ISTEP = ISTEP * 2;
     }
  while (1);

SUB50 :

  do {
      MIDDLE = (ILO + IHI) / 2;
      if ( MIDDLE == ILO ) return ILO;

      if ( X < XT[MIDDLE-1] )
         IHI = MIDDLE;
      else
         ILO = MIDDLE;
     }
  while (1);
}
```

Çizelge 2.1 BSPLVB.C programının ekran çıktısı

X	B-1	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1	0,5	0,5	0	0	0	0	0	0	0	0	0
1,2	0,32	0,66	0,02	0	0	0	0	0	0	0	0
1,4	0,18	0,74	0,08	0	0	0	0	0	0	0	0
1,6	0,08	0,74	0,18	0	0	0	0	0	0	0	0
1,8	0,02	0,66	0,32	0	0	0	0	0	0	0	0
2	0	0,5	0,5	0	0	0	0	0	0	0	0
2,2	0	0,32	0,66	0,02	0	0	0	0	0	0	0
2,4	0	0,18	0,74	0,08	0	0	0	0	0	0	0
2,6	0	0,08	0,74	0,18	0	0	0	0	0	0	0
2,8	0	0,02	0,66	0,32	0	0	0	0	0	0	0
3	0	0	0,5	0,5	0	0	0	0	0	0	0
3,2	0	0	0,32	0,66	0,02	0	0	0	0	0	0
3,4	0	0	0,18	0,74	0,08	0	0	0	0	0	0
3,6	0	0	0,08	0,74	0,18	0	0	0	0	0	0
3,8	0	0	0,02	0,66	0,32	0	0	0	0	0	0
4	0	0	0	0,5	0,5	0	0	0	0	0	0
4,2	0	0	0	0,32	0,66	0,02	0	0	0	0	0
4,4	0	0	0	0,18	0,74	0,08	0	0	0	0	0
4,6	0	0	0	0,08	0,74	0,18	0	0	0	0	0
4,8	0	0	0	0,02	0,66	0,32	0	0	0	0	0
5	0	0	0	0	0,5	0,5	0	0	0	0	0
5,2	0	0	0	0	0,32	0,66	0,02	0	0	0	0
5,4	0	0	0	0	0,18	0,74	0,08	0	0	0	0
5,6	0	0	0	0	0,08	0,74	0,18	0	0	0	0
5,8	0	0	0	0	0,02	0,66	0,32	0	0	0	0
6	0	0	0	0	0	0,5	0,5	0	0	0	0
6,2	0	0	0	0	0	0,32	0,66	0,02	0	0	0
6,4	0	0	0	0	0	0,18	0,74	0,08	0	0	0
6,6	0	0	0	0	0	0,08	0,74	0,18	0	0	0
6,8	0	0	0	0	0	0,02	0,66	0,32	0	0	0
7	0	0	0	0	0	0	0,5	0,5	0	0	0
7,2	0	0	0	0	0	0	0,32	0,66	0,02	0	0
7,4	0	0	0	0	0	0	0,18	0,74	0,08	0	0
7,6	0	0	0	0	0	0	0,08	0,74	0,18	0	0
7,8	0	0	0	0	0	0	0,02	0,66	0,32	0	0
8	0	0	0	0	0	0	0	0,5	0,5	0	0
8,2	0	0	0	0	0	0	0	0,32	0,66	0,02	0
8,4	0	0	0	0	0	0	0	0,18	0,74	0,08	0
8,6	0	0	0	0	0	0	0	0,08	0,74	0,18	0
8,8	0	0	0	0	0	0	0	0,02	0,66	0,32	0
9	0	0	0	0	0	0	0	0	0,5	0,5	0
9,2	0	0	0	0	0	0	0	0	0,32	0,66	0,02
9,4	0	0	0	0	0	0	0	0	0,18	0,74	0,08
9,6	0	0	0	0	0	0	0	0	0,08	0,74	0,18
9,8	0	0	0	0	0	0	0	0	0,02	0,66	0,32
10	0	0	0	0	0	0	0	0	0	0,5	0,5



Şekil 2.2 BSPLVB.C programının verdiği sonuçlara göre [1,10] aralığında B-Spline fonksiyonlarının grafiği.

2.5. B-Spline FonksiyonlarınınTürevinin Hesaplanması ve BSPLVD.C Programı

$g(x) = (t-x)_+^{k-1}$ polinomunun x'e göre türevi

$$D_x(t-x)_+^{k-1} = -(k-1)(t-x)_+^{k-2}$$

olarak hesaplanmaktadır. Diğer taraftan

$$B_{i,k}(x) = [t_{i+1},.....,t_{i+k}](.-x)_+^{k-1} - [t_i,.....,t_{i+k-1}](.-x)_+^{k-1}$$
$$DB_{i,k}(x) = ([t_{i+1},.....,t_{i+k}] - [t_i,.....,t_{i+k-1}])D_x(.-x)_+^{k-1}$$

tanımından türev alınacak olursa,

$$DB_{i,k}(x) = -(k-1)[t_{i+1},.....,t_{i+k}](.-x)_+^{k-2} + (k-1)[t_i,.....,t_{i+k-1}](.-x)_+^{k-2}$$

ifadesi elde edilmektedir. Bu eşitlik yeniden yazılırsa,

$$\frac{dB_{i,k}(x)}{dx} = -(k-1)\frac{B_{i+1,k-1}}{t_{i+k} - t_{i+1}} + (k-1)\frac{B_{i,k-1}}{t_{i+k-1} - t_i} \qquad (2.7)$$

bağlantısına ulaşılır. (i-1) için bir bağıntı daha yazalım.

$$\frac{dB_{i-1,k}(x)}{dx} = -(k-1)\frac{B_{i,k-1}}{t_{i+k-1} - t_i} + (k-1)\frac{B_{i-1,k-1}}{t_{i+k-2} - t_{i-1}} \qquad (2.8)$$

(2.7) deki ikinci terim ile (2.8) deki ilk terimin birbirinin tekrarı olduğuna dikkat edilirse

$$D(\sum_i \alpha_i B_{i,k}) = \sum_i (k-1)\frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1} \qquad (2.9)$$

eşitliği kolayca görülür. Demek ki, $\sum_i \alpha_i B_{i,k}$ spline fonksiyonunun birinci türevi, B-spline katsayılarının farkları ile elde edilmektedir. (2.9) daki eşitlik sonsuz toplam ifade etmektedir. Sıfır terimler atılırsa, $\alpha_{r-1} = \alpha_{s+1} = 0$ olmak koşulu ile

$$D[\sum_{i=r}^{s} \alpha_i B_{i,k}] = \sum_{i=r}^{s+1} (k-1)\frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1}$$

yazılır. Eğer $[t_r, t_s]$ aralığında çalışılıyorsa spline fonksiyonu

$$\sum_i \alpha_i B_{i,k} = \sum_{i=r-k+1}^{s-1} \alpha_i B_{i,k}, \quad ([t_r, t_s] \quad \textit{üzerinde})$$

şeklindedir ve birinci türev

$$D[\sum_i \alpha_i B_{i,k}] = \sum_{i=r-k+2}^{s-1} (k-1)\frac{\alpha_i - \alpha_{i-1}}{t_{i+k-1} - t_i} B_{i,k-1}, \quad ([t_r, t_s] \quad \textit{üzerinde})$$

olarak elde edilir. $B_{i,k-1}$ değerleri $[t_r, t_s]$ aralığında, $i \notin [r-k+2, s-1]$ için sıfıra özdestir.

(2.9)'deki uygulama j kez tekrar edilirse,

$$D^j[\sum_i \alpha_i B_{i,k}] = \sum_i \alpha_i^{(j+1)} B_{i,k-j}, \qquad (2.10)$$

olur. Burada α_i^{j+1}

$$\alpha_r^{(i+1)} = \begin{cases} \alpha_r & , \ j = 0 \quad için \\ \dfrac{\alpha_r^{(j)} - \alpha_{r-1}^{(j)}}{(t_{r+k-j} - t_r)/(k-j)} & , \ j > 0 \quad için \end{cases} \qquad (2.11)$$

dir. Ancak $t_{r+k-j} - t_r = 0$ durumunda $B_{r,k-j} = 0$ olacağından, bu α_r^{j+1} değerini hesaplamaktan kaçınılmalıdır. Sonuç olarak $f = \Sigma \alpha_j B_j$ spline fonksiyonunun m. türevi

$$f^{(m)}(x) = \sum_{r=1}^{k-m} \alpha_{j-k+m+r}^{(m+1)} B_{i-(k-m)+r,k-m}(x) \qquad (2.12)$$

şeklinde bir toplam ile ortaya çıkmaktadır.

Bu yöntemi kullanarak C dilinde yazılmış bir program ve programın ekran görüntüsü aşağıda verilmiştir.

```
/* BSPLVD.C */

# include "stdio.h"
# include "conio.h"
# include "math.h"

 int *bsplvb(float t2[] , int jhigh , int index , float x , int left);
 int interv(int LXT,float X,float XT[]);
 float biatx[20];

main()
{
 int k,left,nderiv,ideriv,il,jlow,jp1mid,kp1,kp1mm,ldummy,m,mhigh,gec;
 float a[10][10],dbiatx[10][10],x,faktor,sum;
 float t[] = {-1.0,0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0,11.0};
 int *p,s,i,j;

 clrscr();
 printf("B-Spline mertebesini girin : %d",k);
 printf("Hesaplanacak noktayı girin : %f",x);
 printf("Hesaplanacak türev mertebesini girin : %d",nderiv);

 for ( i = 0 ; i <= k ; i++)
   for ( j = 0 ; j <= nderiv ; j++) dbiatx[i][j] = 0.0;
 gec = ( nderiv <= k ) ? nderiv : k;
 mhigh = ( gec >= 1 ) ? gec : 1;
 kp1 = k + 1;
 left = interv( 13 , x , t );
 p = bsplvb( t , kp1-mhigh , 1 , x , left );
 for ( s = 1 ; s <= kp1-mhigh ; s++ ) dbiatx[s][1] = biatx[s];
```

```
if ( mhigh == 1 ) goto sonuc;
ideriv = mhigh;
for ( m = 2 ; m <= mhigh ; m++ )
{
 jp1mid = 1;
 for ( j = ideriv ; j <= k ; j++ )
 {
  dbiatx[j][ideriv] = dbiatx[jp1mid][1];
  jp1mid += 1;
 }
 ideriv -= 1;
 p = bsplvb( t , kp1-ideriv , 2 , x , left);
 for ( s = 1 ; s <= kp1-ideriv ; s++ ) dbiatx[s][1] = biatx[s];
}

jlow = 1;
for ( i = 1 ; i <= k ; i++ )
{
 for ( j = jlow ; j <= k ; j++ ) a[j][i] = 0;
 jlow = i;
 a[i][i] = 1;
}

for ( m = 2 ; m <= mhigh ; m++ )
{
 kp1mm = kp1 - m;
 il = left;
 i = k;
 for ( ldummy = 1 ; ldummy <= kp1mm ; ldummy++ )
 {
```

```
    faktor = kp1mm / ( t[il+kp1mm] - t[il] );
    for (j = 1 ; j <= i ; j++ ) a[i][j] = ( a[i][j] - a[i-1][j] ) * faktor;
    il -= 1;
   }

  for ( i = 1 ; i <= k ; i++ )
  {
   sum = 0;
   jlow = ( i >= m ) ? i : m;
   for ( j = jlow ; j <= k ; j++ ) sum = a[j][i] * dbiatx[j][m] + sum;
   dbiatx[i][m] = sum;
  }
 }

 sonuc :

  clrscr();
  printf("dbiatx = \n");
  for ( i = 1 ; i <= k ; i++)
  {
   printf("\n");
   for ( j = 1 ; j <= nderiv ; j++) printf("%3.3f ",dbiatx[i][j]);
  }
  getch();
}

int *bsplvb(float t2[] , int jhigh , int index , float x , int left)
{

 int lxt,i,j = 1,jp1,k;
```

```
float deltal[100],deltar[100],saved,term,bas,top=0.0;

 switch ( index ) {
  case 1 : goto index1;
  case 2 : { j = jhigh - 1; goto index2; }
 }

 index1 :

  j = 1;
  biatx[1] = 1;
  if ( j >= jhigh ) return;

 index2 :
 do
 {
  jp1 = j + 1;
  deltar[j] = t2[left-1+j] - x;
  deltal[j] = x - t2[left-j];
  saved = 0;
  for (i=1 ; i <= j ; i++)
  {
    term = biatx[i] / (deltar[i] + deltal[jp1-i]);
    biatx[i] = saved + deltar[i] * term;
    saved = deltal[jp1-i] * term;
   }
  biatx[jp1] = saved;
  j = jp1;
 }
 while ( j < jhigh );
```

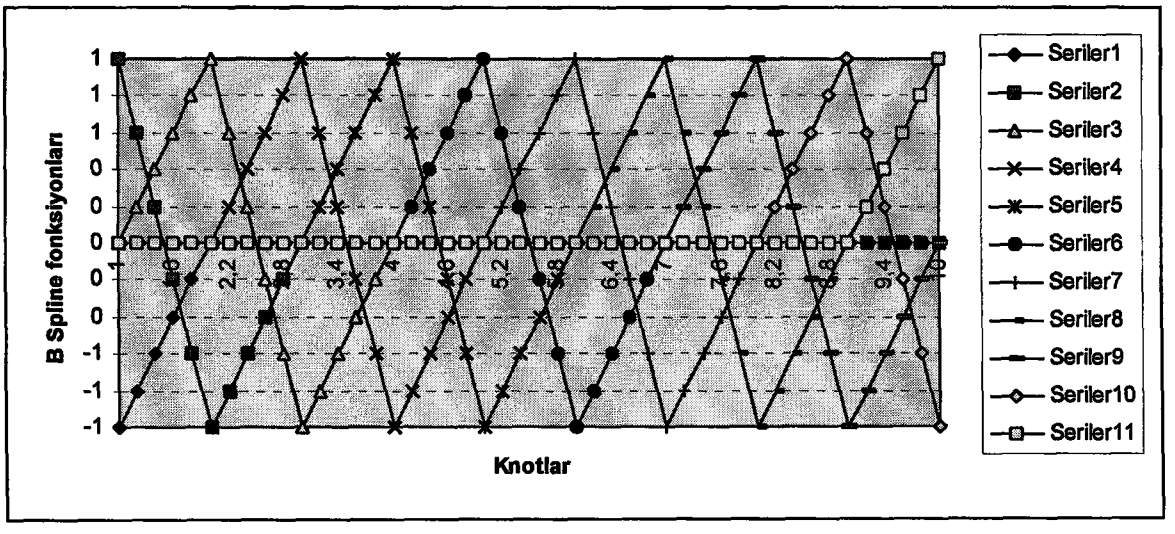
```
   return;
}

int LXT,ILO,IHI,ISTEP;
float X,XT[100],MIDDLE;

int interv(int LXT,float X,float XT[]) {

  ILO = 1;
  IHI = ILO +1;

  if ( IHI < LXT )
  {
    if ( X >= XT[IHI-1] ) goto SUB40;
   }
  else
  {
    if ( X >= XT[LXT-1] )  return LXT;
     else if ( LXT <= 1 )  return 1;
    }

  ILO = LXT -1;
  IHI = ILO;

  if ( X >= XT[IHI-1] ) goto SUB40;
  if ( X >= XT[ILO-1] ) return ILO;
  ISTEP = 1;
  do {
     IHI = ILO;
     ILO = IHI - ISTEP;
```

```
    if ( ILO <= 1 )
     {  ILO = 1;
       if ( X < XT[0] ) return 1;
       goto SUB50;
     }
    }
  while (1);

SUB40 :
 ISTEP = 1;
 do {
    ILO = IHI;
    IHI = ILO + ISTEP;
    if ( IHI >= LXT )
     {
      if ( X >= XT[LXT-1] ) return LXT;
      IHI = LXT;
      goto SUB50;
     }
    if ( X < XT[IHI-1] ) goto SUB50;
    ISTEP = ISTEP * 2;
    }
  while (1);

SUB50 :
 do {
    MIDDLE = (ILO + IHI) / 2;
    if ( MIDDLE == ILO ) return ILO;

    if ( X < XT[MIDDLE-1] )
```

```
            IHI = MIDDLE;
        else
            ILO = MIDDLE;
      }
    while (1);
}
```

Çizelge 2.2 BSPLVD.C Programının ekran çıktısı (1.türev)

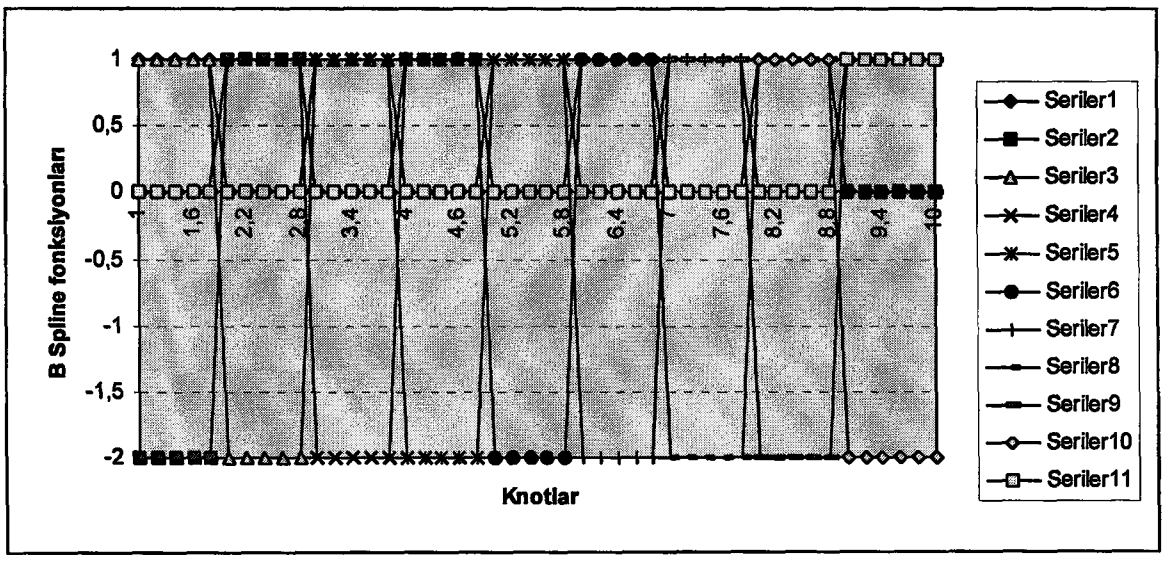
X	B-1	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1	-1	1	0	0	0	0	0	0	0	0	0
1,2	-0,8	0,6	0,2	0	0	0	0	0	0	0	0
1,4	-0,6	0,2	0,4	0	0	0	0	0	0	0	0
1,6	-0,4	-0,2	0,6	0	0	0	0	0	0	0	0
1,8	-0,2	-0,6	0,8	0	0	0	0	0	0	0	0
2	0	-1	1	0	0	0	0	0	0	0	0
2,2	0	-0,8	0,6	0,2	0	0	0	0	0	0	0
2,4	0	-0,6	0,2	0,4	0	0	0	0	0	0	0
2,6	0	-0,4	-0,2	0,6	0	0	0	0	0	0	0
2,8	0	-0,2	-0,6	0,8	0	0	0	0	0	0	0
3	0	0	-1	1	0	0	0	0	0	0	0
3,2	0	0	-0,8	0,6	0,2	0	0	0	0	0	0
3,4	0	0	-0,6	0,2	0,4	0	0	0	0	0	0
3,6	0	0	-0,4	-0,2	0,6	0	0	0	0	0	0
3,8	0	0	-0,2	-0,6	0,8	0	0	0	0	0	0
4	0	0	0	-1	1	0	0	0	0	0	0
4,2	0	0	0	-0,8	0,6	0,2	0	0	0	0	0
4,4	0	0	0	-0,6	0,2	0,4	0	0	0	0	0
4,6	0	0	0	-0,4	-0,2	0,6	0	0	0	0	0
4,8	0	0	0	-0,2	-0,6	0,8	0	0	0	0	0
5	0	0	0	0	-1	1	0	0	0	0	0
5,2	0	0	0	0	-0,8	0,6	0,2	0	0	0	0
5,4	0	0	0	0	-0,6	0,2	0,4	0	0	0	0
5,6	0	0	0	0	-0,4	-0,2	0,6	0	0	0	0
5,8	0	0	0	0	-0,2	-0,6	0,8	0	0	0	0
6	0	0	0	0	0	-1	1	0	0	0	0
6,2	0	0	0	0	0	-0,8	0,6	0,2	0	0	0
6,4	0	0	0	0	0	-0,6	0,2	0,4	0	0	0
6,6	0	0	0	0	0	-0,4	-0,2	0,6	0	0	0
6,8	0	0	0	0	0	-0,2	-0,6	0,8	0	0	0
7	0	0	0	0	0	0	-1	1	0	0	0
7,2	0	0	0	0	0	0	-0,8	0,6	0,2	0	0
7,4	0	0	0	0	0	0	-0,6	0,2	0,4	0	0
7,6	0	0	0	0	0	0	-0,4	-0,2	0,6	0	0
7,8	0	0	0	0	0	0	-0,2	-0,6	0,8	0	0
8	0	0	0	0	0	0	0	-1	1	0	0
8,2	0	0	0	0	0	0	0	-0,8	0,6	0,2	0
8,4	0	0	0	0	0	0	0	-0,6	0,2	0,4	0
8,6	0	0	0	0	0	0	0	-0,4	-0,2	0,6	0
8,8	0	0	0	0	0	0	0	-0,2	-0,6	0,8	0
9	0	0	0	0	0	0	0	0	-1	1	0
9,2	0	0	0	0	0	0	0	0	-0,8	0,84	-0,04
9,4	0	0	0	0	0	0	0	0	-0,6	0,68	-0,08
9,6	0	0	0	0	0	0	0	0	-0,4	0,52	-0,12
9,8	0	0	0	0	0	0	0	0	-0,2	0,36	-0,16
10	0	0	0	0	0	0	0	0	0,2	-0,2	0



Şekil 2.3 BSPLVD.C programının verdiği sonuçlara göre [1,10] aralığında B-Spline fonksiyonlarının 1. mertebeden türevlerinin grafiği.

Çizelge 2.3 BSPLVD.C Programının ekran çıktısı (2.türev)

X	B-1	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1	1	-2	1	0	0	0	0	0	0	0	0
1,2	1	-2	1	0	0	0	0	0	0	0	0
1,4	1	-2	1	0	0	0	0	0	0	0	0
1,6	1	-2	1	0	0	0	0	0	0	0	0
1,8	1	-2	1	0	0	0	0	0	0	0	0
2	0	1	-2	1	0	0	0	0	0	0	0
2,2	0	1	-2	1	0	0	0	0	0	0	0
2,4	0	1	-2	1	0	0	0	0	0	0	0
2,6	0	1	-2	1	0	0	0	0	0	0	0
2,8	0	1	-2	1	0	0	0	0	0	0	0
3	0	0	1	-2	1	0	0	0	0	0	0
3,2	0	0	1	-2	1	0	0	0	0	0	0
3,4	0	0	1	-2	1	0	0	0	0	0	0
3,6	0	0	1	-2	1	0	0	0	0	0	0
3,8	0	0	1	-2	1	0	0	0	0	0	0
4	0	0	0	1	-2	1	0	0	0	0	0
4,2	0	0	0	1	-2	1	0	0	0	0	0
4,4	0	0	0	1	-2	1	0	0	0	0	0
4,6	0	0	0	1	-2	1	0	0	0	0	0
4,8	0	0	0	1	-2	1	0	0	0	0	0
5	0	0	0	0	1	-2	1	0	0	0	0
5,2	0	0	0	0	1	-2	1	0	0	0	0
5,4	0	0	0	0	1	-2	1	0	0	0	0
5,6	0	0	0	0	1	-2	1	0	0	0	0
5,8	0	0	0	0	1	-2	1	0	0	0	0
6	0	0	0	0	0	1	-2	1	0	0	0
6,2	0	0	0	0	0	1	-2	1	0	0	0
6,4	0	0	0	0	0	1	-2	1	0	0	0
6,6	0	0	0	0	0	1	-2	1	0	0	0
6,8	0	0	0	0	0	1	-2	1	0	0	0
7	0	0	0	0	0	0	1	-2	1	0	0
7,2	0	0	0	0	0	0	1	-2	1	0	0
7,4	0	0	0	0	0	0	1	-2	1	0	0
7,6	0	0	0	0	0	0	1	-2	1	0	0
7,8	0	0	0	0	0	0	1	-2	1	0	0
8	0	0	0	0	0	0	0	1	-2	1	0
8,2	0	0	0	0	0	0	0	1	-2	1	0
8,4	0	0	0	0	0	0	0	1	-2	1	0
8,6	0	0	0	0	0	0	0	1	-2	1	0
8,8	0	0	0	0	0	0	0	1	-2	1	0
9	0	0	0	0	0	0	0	0	1	-2	1
9,2	0	0	0	0	0	0	0	0	1	-2	1
9,4	0	0	0	0	0	0	0	0	1	-2	1
9,6	0	0	0	0	0	0	0	0	1	-2	1
9,8	0	0	0	0	0	0	0	0	1	-2	1
10	0	0	0	0	0	0	0	0	1	-2	1



Şekil 2.4 BSPLVD.C programının verdiği sonuçlara göre [1,10] aralığında B-Spline fonksiyonlarının 2. mertebeden türevlerinin grafiği.

BÖLÜM III. ADİ DİFERANSİYEL DENKLEMLERİN KOLLAKASYON YÖNTEMİ İLE ÇÖZÜMÜ

3.1. Kollakasyon Metodu

Bu bölümde kollakasyon yöntemi ile adi diferansiyel denklemlerin sayısal çözümlerinde, B-spline'ların nasıl kullanıldığı anlatılmıstır. Diferansiyel denklem ve sınır koşulları

$$(D^m g)(x) = F(x; g(x),.....,(D^{m-1}g)(x)), \quad x \in [a,b] \quad için,$$
$$\beta_i g = c_i, \quad i = 1,2,.....,m \qquad (3.1)$$

formunda olacaktır.

Burada $F=F(x;z_o,...,z_{m-1})$ fonksiyonu, $\Re^{m+1}$ üzerinde reel değerli sürekli bir fonksiyondur. Bu fonksiyonun yeterince düzgün olduğu varsayılmaktadır.

Sınır koşulları ise

$$\sum_{j=1}^{m} w_{ij}(D^{j-1}g)(x_i) = \beta_i g = c_i, \quad i = 1,....,m \quad (3.1.1)$$

şeklindedir. Burada da w_{ij} değerleri sabitler, x_i noktaları ise $a \le x_1 \le ..... \le x_m \le b$ olarak belirlidir. $\beta_1,.....,\beta_m$ ise $C^{(m-1)}$ [a,b] 'de sürekli lineer fonksiyonellerdir ve son olarak $c_1,c_2,...,c_m$ belli sabitlerdir.

Verilenlere göre sınır koşulları lineer, diferansiyel denklem ise lineer değildir. Bu durumda (3.1) probleminin birden fazla çözümü olabilir. 0 halde arzu edilen, g'nin herhangi bir komşuluğunda g'den baska bir çözümün bulunmamasıdır. Yani g ayrık (isole) çözüm olmalıdır. (3.1) denklemi lineerleştirilerek çözüleceğinden, lineer olmayan bu denklemin çözümü ardışık yaklaşımlarla elde edilecektir. Bundan sonra yapılacak iş, g'ye yakınsayan bir çözüm dizisi elde etmektir. g'nin komşuluğu içersinde bir çözüm baslangıç olarak ele alınır ve ardışık işlemler ile aranan çözüme bir yakınsak dizi ile yaklaşılır.

g'ye yaklaşım, $P_{k+m,\xi} \cap C^{(m-1)}$ parçalı polinomlar sınıfında aranacaktır.Yani bir $f \in P_{k+m,\xi} \cap C^{(m-1)}$ çözümü $\xi = (\xi_i)_1^{\ell+1}$ dizisi ile verilmiş olan kırılma noktalarına göre parçalı polinom olarak belirlenecektir.

Kollakasyon yöntemi ile yukarıdaki diferansiyel denklemin çözümü ise; kollakasyon noktaları denilen özel noktalarda diferansiyel denklemi ve verilen koşulları tam olarak sağlayacak bir $f \in P_{k+m,\xi} \cap C^{(m-1)}$ fonksiyonu oluşturmaktır. Buna göre

$$(D^m f)(\tau_i) = F(\tau_i; f(\tau_i), \ldots\ldots, (D^{m-1} f)(\tau_i)), \quad i = 1, \ldots\ldots, k\ell$$
$$\beta_i f = c_i, \quad i = 1,2, \ldots\ldots, m \qquad (3.2)$$

olmalıdır. $(\tau_i)_1^{k\ell}$ kollakasyon noktalarını, kırılma noktalarının oluşturduğu her alt aralıkta aynı dağılıma sahip olarak ele alınsın. Örneğin

$$-1 \le \rho_1 < \rho_2 < \ldots\ldots < \rho_k \le 1 \qquad (3.2.a)$$

olmak üzere, bu noktalar

$$\tau_{(i-1)k+\nu} = [\xi_{i+1} + \xi_i + \rho_\nu(\xi_{i+1} - \xi_i)]/2 \qquad \nu = 1, \ldots\ldots, k; \quad i = 1, \ldots\ldots, \ell \qquad (3.2.b)$$

bağıntısı ile hesaplanabilir. Bu çalışmada $\rho = (\rho)_i^k$ seçimi k. mertebeden Legendre polinomunun kökleri olarak belirlenmiştir.

$$\begin{aligned} &\beta_i y = c_i \quad , \quad i = 1, \ldots\ldots, m \\ &(D^m y)(\tau_i) + \sum_{l<m} v_j(\tau_i)(D^j y)(\tau_i) = h(\tau_i), \quad i = 1, \ldots\ldots, k\ell \\ &v_j(x) = -(\frac{\partial F}{\partial z_j})(x; f_r(x), \ldots, (D^{m-1} f_r)(x)), \quad j = 0, \ldots\ldots, m-1 \\ &h(x) = F(x; f_r(x), \ldots\ldots, (D^{m-1} f_r)(x)) + \sum_{j<m} v_j(x)(D^j f_r)(x) \end{aligned} \qquad (3.3)$$

problemini gözönüne alalım. (3.3) problemindeki y fonksiyonu, B-spline'ların uygun lineer kombinasyonlarıdır. $t = (t)_i^{n+k+m}$ azalmayan noktalar dizisi, iç noktalarda (k) kez, ξ_1 ve $\xi_{\iota+1}$ sınır noktalarında ise (k+m) kez tekrarlanarak oluşturulmaktadır. Bu durumda, boyutu n=kı+m olan spline uzayı,

$\$_{k+m,t} = P_{k+m,\xi} \cap C^{(m-1)}$

şeklinde parçalı polinomlar uzayını temsil etmektedir. O halde , (3.3) probleminin aranan çözümü

$$y = \sum_{j=1}^{n} \alpha_j B_{j,k+m,t}$$

biçiminde bir spline'dir. B_j değerleri önceden bilindiği için B-katsayılarını ifade eden α vektörünün hesaplanması ile amacımıza ulaşacağız. α 'lar ise

$$\begin{aligned} &\sum_{j=1}^{n}(LB_j)(\tau_i)\alpha_j = h(\tau_i) \quad , \quad i = 1,.....,k \\ &\sum_{j=1}^{n}(\beta_i B_j)\alpha_j = c_i \quad , \quad i = 1,.....,m \end{aligned} \tag{3.4}$$

lineer denklem sisteminin çözümü olarak belirlenecektir. Yukarıda kullanılmış olan L lineer diferansiyel operatörü

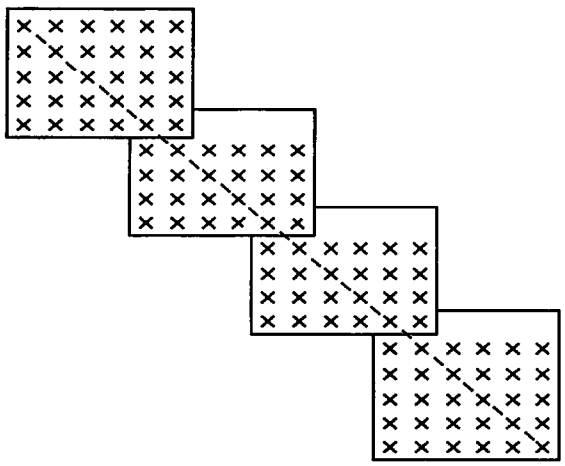
$$Ly = D^m y + \sum_{j<m} V_j(.)D^j y$$

olarak tanımlıdır, β_i 'ler ise (3.1.1) ile tanımlı olan operatörlerdir.

3.2. Denklem Sisteminin Oluşturulması

(3.4) sisteminin yapısını biraz daha yakından inceleyelim. Eğer sınır koşulları (3.2) kollakasyon denklemleri arasında uygun dağılmış iseler, elimizdeki sistem blok kösegenli bir yapıya sahiptir.

Örnek olarak ikinci mertebeden, iki nokta sınır değer problemini ele alalım. Dört aralık üzerinde tanımlı bir parçalı polinom, 6. mertebeden spline'lar ile oluşturulan bir yaklaşım fonksiyonu olarak seçildiğinde (3.4)'deki sistemin sol tarafı, sıfırdan farklı elemanları x ile belirtilecek olursa,



formunda bir yapı göstermektedir.

Yukarıda i. blok, (ξ_i, ξ_{i+1}) aralığındaki k kollakasyon noktalarına ait denklemler ile bu aralık içerisinde tanımlı olan yan şart denklemlerinden kaynaklanmaktadır ($i = 1,..., \iota$).

Biz bu lineer denklem sisteminin çözümü için, blok özelliğini göz önüne almadan, skale edilmiş satır pivotlama (scaled row pivoting) yöntemi ile klasik Gauss Eliminasyonu kullanacağız.

α katsayıları elde edildikten sonra, y spline yaklaşım fonksiyonu bulunur.

BÖLÜM IV. UYGULAMA

$$\left\{\begin{matrix} y'' + y = 2e^{t} \\ y(0) = 2 \quad , \quad y(1) = e + Cos1 \end{matrix}\right\}$$

denklem sisteminin çözümünü B-Spline yaklaşımı ile bulan program ve ekran görüntüleri :

```
/*  MAIN.C  */

# include "stdio.h"
# include "conio.h"
# include "math.h"

  int *bsplvd(float t[] , int k , float x , int left , int nderiv);
  int *bsplvb(float t2[] , int jhigh , int index , float x , int left);
  int *kokbul(float km[][] , float sag[]);
  int interv(int LXT,float X,float XT[]);
  int *colpnt(int k);
  int *knots(float breakp[] , int l , int kpm);
  int n;
  float t[100];
  float dbiatx[10][10];
  float biatx[10];
  float xk[100];
  float rho[10];

  main()
  {

//   float breakp[] = {0.0 ,0.0 ,0.1,0.2 ,0.3 ,0.4,0.5 ,0.6 ,0.7 ,0.8,0.9 ,1.0};
    float breakp[] = {0.0 ,0.0 ,0.2 ,0.4 ,0.6 ,0.8 ,1.0};
```

```
int l = 5 , k = 3 , m = 2 , *p ,*p2 ,*p3 ,*p4 , i , j ,left,s,kpm;
float colp[100] , w[100][100] , b[100] , coz[100] , gcoz[100];
float hata[100];

rho[0] = 0.0;
colp[0] = 0.0;
xk[0] = 0.0;
coz[0] = 0.0; coz[1] = 0.0;
gcoz[0] = 0.0;
hata[0] = 0.0;
for (i=0 ; i<=99 ; i++)
  for (j=0 ; j<=99 ; j++) w[i][j] = 0.0;
for (i=0 ; i<=99 ; i++)  b[i] = 0.0;

clrscr();
kpm = k + m;
p = knots(breakp , l , k+m);
p2 = colpnt( k );
/*  Kollokasyon noktalarını hesapla  */

for ( i = 1 ; i <= l ; i++ )
  for ( j = 1 ; j <= k ; j++ )
  {
    s = ( i - 1 ) * k + j;
    colp[s] = (breakp[i+1]+breakp[i]+rho[j]*(breakp[i+1]-breakp[i]))/2;
  }

/*  Katsayılar matrisini bloklar halinde oluştur  */

  left = interv( n+k+m+1 , breakp[1] , t )-1;
```

```
    p3 = bsplvd(t , kpm , breakp[1] , left , m+1);
    for ( j = 1 ; j <= kpm ; j++ )
      w[1][j] = dbiatx[j][1];
    for ( i = 1 ; i <= k*1 ; i++ )
    {
      left = interv( n+k+m+1 , colp[i] , t )-1;
      p3 = bsplvd(t , kpm , colp[i] , left , m+1);
      for ( j = 1 ; j <= kpm ; j++ )
        w[i+1][left-kpm+j] = dbiatx[j][1] + dbiatx[j][3];
}
     left = interv( n+k+m+1 , breakp[l+1] , t )-1;
     p3 = bsplvd(t , kpm , breakp[l+1] , left , m+1);
     for ( j = 1 ; j <= kpm ; j++ )
       w[n][left-2*kpm+j] = dbiatx[j][1];

 /*  Denklemin sağ tarafını oluştur  */

    b[1] = 2;
    for ( i = 1 ; i <= k*1 ; i++ ) b[i+1] = 2 * exp(colp[i]);
    b[n] = exp(1) + cos(1);

 /*  Denklemi çöz  */

     p4 = kokbul( w , b );

 /*  Kollokasyon metoduyla yaklasik cozum  */

     left = interv( n+k+m+1 , breakp[1] , t )-1;
     p3 = bsplvd(t , kpm , breakp[1] , left , m+1);
     for ( j = 1 ; j <= kpm ; j++ )
```

```
coz[1] += xk[j] * dbiatx[j][1];
   for ( i = 1 ; i <= k*l ; i++ )
   {
    left = interv( n+k+m+1 , colp[i] , t )-1;
    p3 = bsplvd(t , kpm , colp[i] , left , m+1);
    coz[i+1] = 0.0;
    for ( j = 1 ; j <= kpm ; j++ )
      coz[i+1] += xk[left-kpm+j] * dbiatx[j][1];
   }
   left = interv( n+k+m+1 , breakp[l+1] , t )-1;
   p3 = bsplvd(t , kpm , breakp[l+1] , left , m+1);
   coz[n] = 0.0;
   for ( j = 1 ; j <= 5 ; j++ )
    coz[n] += xk[left-2*kpm+j] * dbiatx[j][1];

/*  Gercek cozumu hesapla  */

   gcoz[1] = 2;
   for ( i = 1 ; i <= k*l ; i++ ) gcoz[i+1] = exp(colp[i]) + cos(colp[i]);
   gcoz[n] = exp(1) + cos(1);
/*  Hatayı hesapla  */

   for ( i = 1 ; i <= n ; i++ )  hata[i] = fabs( gcoz[i]-coz[i] );

/*  Hesaplanan cozumu , gercek cozumu ve hatayı ekrana yazdir  */

   clrscr();
   for ( i = 1 ; i <= n ; i++ )
   {
    printf("%f   %f   %f\n",coz[i],gcoz[i],hata[i]);
```

```
 if ( i == 20 ) {getch();clrscr();}
 }
 getch();

}

int *knots(float breakp[] , int l , int kpm)
{

 int iside,j,jj,jjj,k,ll,m = 2;
 float xside;

 k = kpm - m;
 n = l * k + m;
 jj = n + kpm;
 jjj = l + 1;
 for ( ll = 1 ; ll <= kpm ; ll++ )
 {
  t[jj] = breakp[jjj];
  jj -= 1;
 }
 for ( j = 1 ; j <= l ; j++ )
 {
  jjj -= 1;
  for ( ll = 1 ; ll <= k ; ll++ )
  {
   t[jj] = breakp[jjj];
   jj -= 1;
  }
 }
```

```
    for ( ll = 1 ; ll <= kpm ; ll++ ) t[ll] = breakp[1];
    return;
}

 int *colpnt(int k)
 {
    int j;
    float fkm;

    switch ( k ){
       case  1 : rho[1] = 0.0;
                 return;
       case  2 : rho[2] = 0.57735027;
                 rho[1] = -rho[2];
                 return;
       case  3 : rho[3] = 0.77459667;
                 rho[1] = -rho[3];
                 rho[2] = 0.0;
                 return;
       case  4 : rho[3] = 0.33998104;
                 rho[2] = -rho[3];
                 rho[4] = 0.86113631;
                 rho[1] = -rho[4];
return;
       case  5 : rho[4] = 0.53846931;
                 rho[2] = -rho[4];
                 rho[5] = 0.90617985;
                 rho[1] = -rho[5];
                 rho[3] = 0.0;
                 return;
```

```
      case  6 : rho[4] = 0.23861919;
                rho[3] = -rho[4];
                rho[5] = 0.66120939;
                rho[2] = -rho[5];
                rho[6] = 0.93246951;
                rho[1] = -rho[6];
                return;
      case  7 : rho[5] = 0.40584515;
                rho[3] = -rho[5];
                rho[6] = 0.74153119;
                rho[2] = -rho[6];
                rho[7] = 0.94910791;
                rho[1] = -rho[7];
                rho[4] = 0.0;
return;
      case  8 : rho[5] = 0.18343464;
                rho[4] = -rho[5];
                rho[6] = 0.52553241;
                rho[3] = -rho[6];
                rho[7] = 0.79666648;
                rho[2] = -rho[7];
                rho[8] = 0.96028986;
                rho[1] = -rho[8];
                return;
    }

   fkm = ( k - 1) / 2;
   for ( j = 1 ; j <= k ; j++ ) rho[j] = ( j - 1 ) / fkm - 1;
   return;
  }
```

```
int *bsplvb(float t2[] , int jhigh , int index , float x , int left)
{

int lxt,i,j = 1,jp1,k;
float deltal[100],deltar[100],saved,term,bas,top=0.0;

  switch ( index ) {
   case 1 : goto index1;
   case 2 : { j = jhigh - 1; goto index2; }
  }

  index1 :

   j = 1;
   biatx[1] = 1;
   if ( j >= jhigh ) return;

  index2 :
  do
  {
   jp1 = j + 1;
   deltar[j] = t2[left+j] - x;
   deltal[j] = x - t2[left+1-j];
   saved = 0;
   for (i=1 ; i <= j ; i++)
{
     term = biatx[i] / (deltar[i] + deltal[jp1-i]);
     biatx[i] = saved + deltar[i] * term;
     saved = deltal[jp1-i] * term;
```

```
    }
    biatx[jp1] = saved;
    j = jp1;
   }

   while ( j < jhigh );

   return;

  }

 int LXT,ILO,IHI,ISTEP;
 float X,XT[20],MIDDLE;

int interv(int LXT,float X,float XT[]) {

 ILO = 1;
IHI = ILO +1;

 if ( IHI < LXT )
   {
    if ( X >= XT[IHI-1] ) goto SUB40;
   }
 else
   {
    if ( X >= XT[LXT-1] )  return LXT;
     else if ( LXT <= 1 )  return 1;
   }

 ILO = LXT -1;
 IHI = ILO;
```

```
 if ( X >= XT[IHI-1] ) goto SUB40;

 if ( X >= XT[ILO-1] ) return ILO;

 ISTEP = 1;

do {
     IHI = ILO;
     ILO = IHI - ISTEP;
     if ( ILO <= 1 )
      {  ILO = 1;
        if ( X < XT[0] ) return 1;
        goto SUB50;
       }
     }
   while (1);

SUB40 :

 ISTEP = 1;
 do {
     ILO = IHI;
     IHI = ILO + ISTEP;
     if ( IHI >= LXT )
      {
       if ( X >= XT[LXT-1] ) return LXT;
       IHI = LXT;
goto SUB50;
       }
```

```
      if ( X < XT[IHI-1] ) goto SUB50;
      ISTEP = ISTEP * 2;
     }
  while (1);

SUB50 :

  do {
      MIDDLE = (ILO + IHI) / 2;
      if ( MIDDLE == ILO ) return ILO;

      if ( X < XT[MIDDLE-1] )
         IHI = MIDDLE;
      else
         ILO = MIDDLE;
     }
   while (1);
}

  float E(float km[][] , int i);
  int p[100];

  int *kokbul(float km[100][100] , float sag[])
  {

    int i , j = 0 , c , k2;
    float piv,z,s[100],max;

    clrscr();
    s[0] = 0.0;
```

```
   p[0] = 0.0;

/*  Maximumlari bul  */

   for(i=1;i<=n;i++)
   {
    max=km[i][1];
    p[i]=i;
for(j=1;j<=n;++j)
      if(fabs(km[i][j])>max) max=km[i][j];
    s[i]=max;
   }

/*  Satir sutun islemleri  */

   for(k2=1;k2<=n-1;k2++)
   {
    piv=0;
    for(i=k2;i<=n;i++)
    {
     if((fabs(km[p[i]][k2])/s[p[i]])>piv)
      {
       piv=fabs(km[p[i]][k2])/s[p[i]];
       j=i;
      }
     }
    c=p[k2],p[k2]=p[j],p[j]=c;
    for(i=k2+1;i<=n;++i)
    {
z=km[p[i]][k2]/km[p[k2]][k2];
```

```
      km[p[i]][k2]=z;
      for(j=k2+1;j<=n;++j)
      {
       km[p[i]][j]=km[p[i]][j]-z*km[p[k2]][j];
      }
     }
    }

/*  Denklem sistemini ekrana yaz  */

    for(i=1;i<=n;i++)
    {
     for(j=1;j<=n;j++)  printf("%+.4f ",km[i][j]);
     printf("  p(%d)=%d\n",i,p[i]);
    }

/*  Cozumu bul ve kokleri ekrana yaz  */

    for(k2=1;k2<=n-1;++k2)
    {
 for(i=k2+1;i<=n;i++)
       sag[p[i]]=sag[p[i]]-km[p[i]][k2]*sag[p[k2]];
     }
    for(i=n;i>=1;i--)
     xk[i]=(sag[p[i]]-E(km,i))/km[p[i]][i];
    for(i=1;i<=n;i++)
     printf("x(%d)=%f\n",i,xk[i]);
    return;

   }
```

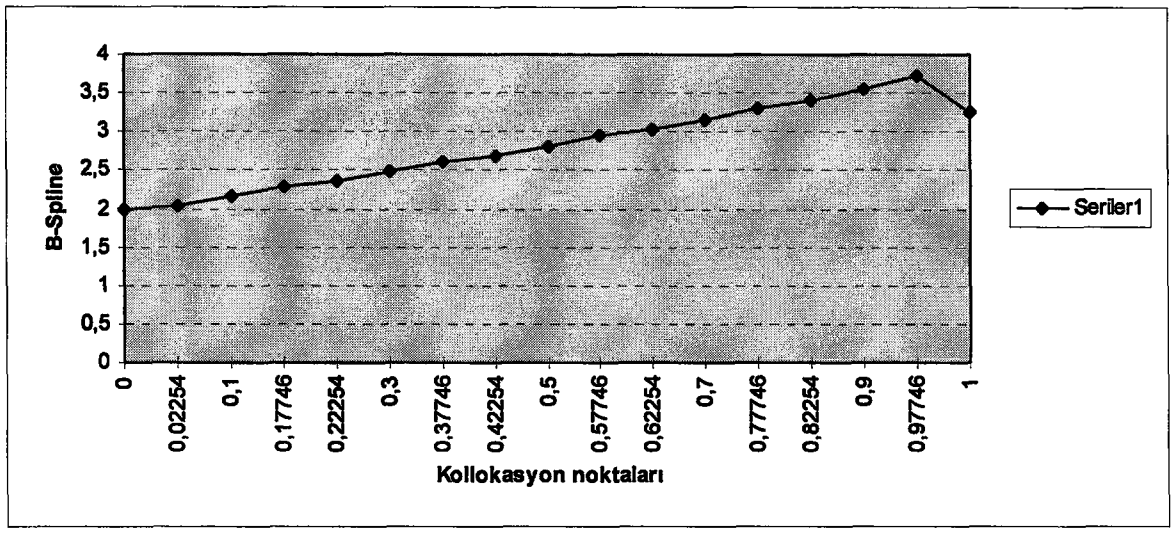
```
float E(float km[100][100] , int i)
{
 int j;
 float t=0.0;

 for(j=i+1;j<=n;j++)  t = t + km[p[i]][j] * xk[j];
 return t;
}

int *bsplvd(float t[] , int k , float x , int left , int nderiv)
{
 int ideriv,il,jlow,jp1mid,kp1,kp1mm,ldummy,m,mhigh,gec;
 float a[10][10],w[10][50],y[50],faktor,sum;
 int *p,s,i,j,z,*p2;

 for ( i = 0 ; i <= k ; i++)
  for ( j = 0 ; j <= 11 ; j++) w[i][j] = 0.0;
 for ( j = 0 ; j <= 11 ; j++) y[j] = 0.0;
 for ( i = 0 ; i <= k ; i++)
  for ( j = 0 ; j <= nderiv ; j++) dbiatx[i][j] = 0.0;

 gec = ( nderiv <= k ) ? nderiv : k;
 mhigh = ( gec >= 1 ) ? gec : 1;
 kp1 = k + 1;
 p = bsplvb( t , kp1-mhigh , 1 , x , left );
 for ( s = 1 ; s <= kp1-mhigh ; s++ ) dbiatx[s][1] = biatx[s];
 if ( mhigh == 1 ) return;
 ideriv = mhigh;
```

```
   for ( m = 2 ; m <= mhigh ; m++ )
   {
jp1mid = 1;
     for ( j = ideriv ; j <= k ; j++ )
     {
      dbiatx[j][ideriv] = dbiatx[jp1mid][1];
      jp1mid += 1;
     }
     ideriv -= 1;
     p = bsplvb( t , kp1-ideriv , 2 , x , left);
     for ( s = 1 ; s <= kp1-ideriv ; s++ ) dbiatx[s][1] = biatx[s];
    }

   jlow = 1;
   for ( i = 1 ; i <= k ; i++ )
   {
     for ( j = jlow ; j <= k ; j++ ) a[j][i] = 0;
     jlow = i;
     a[i][i] = 1;
    }

   for ( m = 2 ; m <= mhigh ; m++ )
   {
kp1mm = kp1 - m;
     il = left;
     i = k;
     for ( ldummy = 1 ; ldummy <= kp1mm ; ldummy++ )
     {
       faktor = kp1mm / ( t[il+kp1mm] - t[il] );
       for (j = 1 ; j <= i ; j++ ) a[i][j] = ( a[i][j] - a[i-1][j] ) * faktor;
```

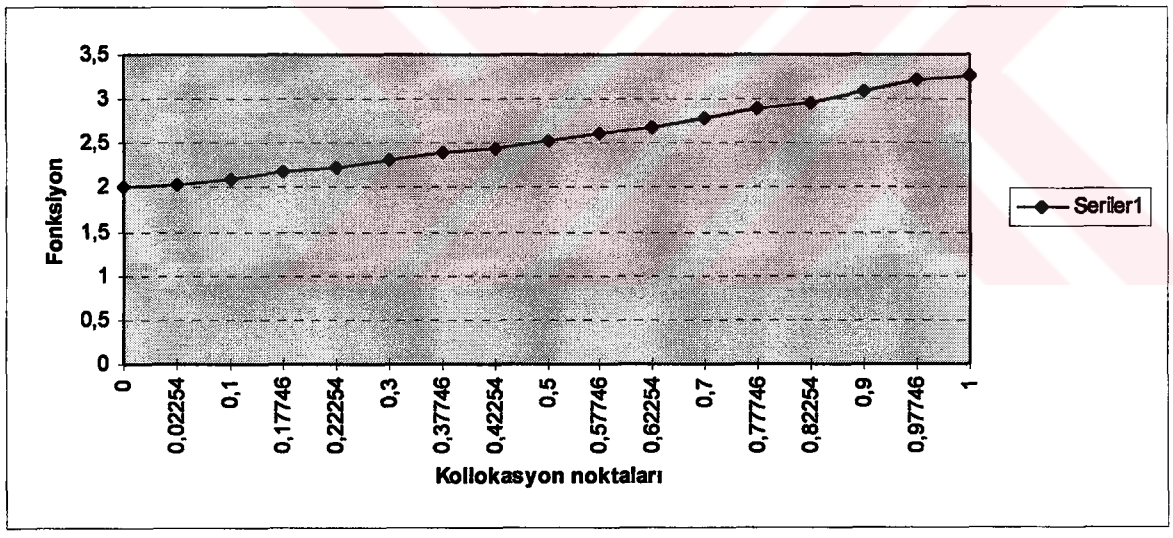
```
    il -= 1;
    i -= 1;
   }

  for ( i = 1 ; i <= k ; i++ )
  {
   sum = 0;
   jlow = ( i >= m ) ? i : m;
   for ( j = jlow ; j <= k ; j++ ) sum = a[j][i] * dbiatx[j][m] + sum;
   dbiatx[i][m] = sum;
   }
 }
} // end of bsplvd()
```

Çizelge 4.1 MAIN.C programının ekran çıktısı (L=5 İçin)

X	YAKLAŞIK	GERÇEK	HATA
0,000000	2,000000	2,000000	0,000000
0,022540	2,036139	2,022542	0,013597
0,100000	2,160402	2,100175	0,060227
0,177460	2,284971	2,178475	0,106496
0,222540	2,357733	2,224586	0,133147
0,300000	2,483474	2,305195	0,178279
0,377460	2,610520	2,388179	0,222341
0,422540	2,685271	2,437883	0,247388
0,500000	2,815526	2,526304	0,289222
0,577460	2,948683	2,619359	0,329324
0,622540	3,027824	2,676056	0,351768
0,700000	3,167233	2,778595	0,388638
0,777460	3,311813	2,888635	0,423178
0,822540	3,398761	2,956637	0,442124
0,900000	3,553771	3,081213	0,472558
0,977460	3,716985	3,216827	0,500158
1,000000	3,258584	3,258584	0,000000



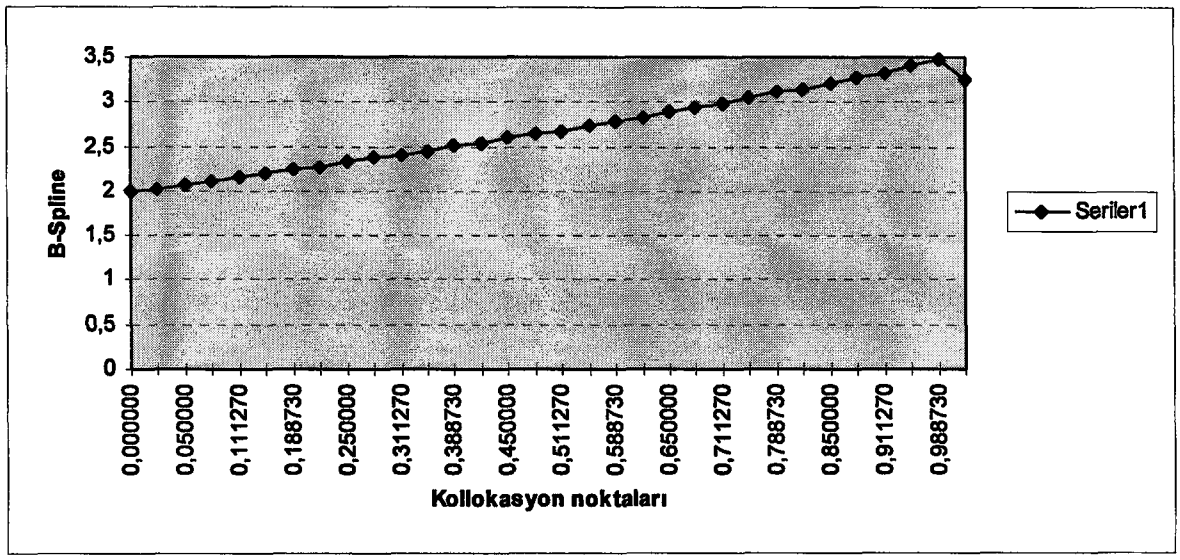
Şekil 4.1 MAIN.C programının verdiği sonuçlara göre B-Spline fonksiyonlarını kullanarak bulunan yaklaşık çözümün grafiği (L = 5 için).



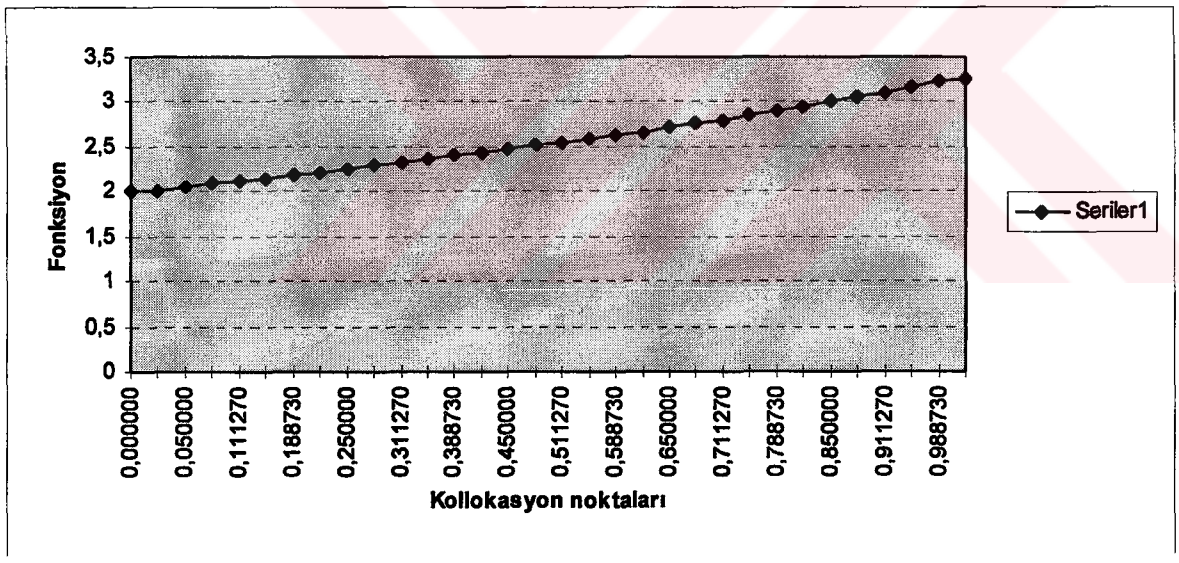
Şekil 4.2 MAIN.C programının verdiği sonuçlara göre B-Spline fonksiyonlarını kullanarak bulunan gerçek çözümün grafiği (L = 5 için).

Çizelge 4.2 MAIN.C programının ekran çıktısı (L=10 için)

X	YAKLAŞIK	GERÇEK	HATA
0,000000	2,000000	2,000000	0,000000
0,011270	2,014489	2,011271	0,003218
0,050000	2,064296	2,050021	0,014275
0,088730	2,114161	2,088851	0,025310
0,111270	2,143228	2,111513	0,031716
0,150000	2,193288	2,150605	0,042682
0,188730	2,243543	2,189958	0,053585
0,211270	2,272905	2,213011	0,059894
0,250000	2,323600	2,252938	0,070662
0,288730	2,374662	2,293337	0,081325
0,311270	2,404578	2,317103	0,087475
0,350000	2,456378	2,358440	0,097938
0,388730	2,508749	2,400497	0,108252
0,411270	2,539529	2,425347	0,114182
0,450000	2,592993	2,468759	0,124233
0,488730	2,647272	2,513174	0,134098
0,511270	2,679280	2,539531	0,139748
0,550000	2,735064	2,585778	0,149287
0,588730	2,791947	2,633345	0,158602
0,611270	2,825605	2,661690	0,163915
0,650000	2,884472	2,711625	0,172848
0,688730	2,944759	2,763239	0,181521
0,711270	2,980553	2,794110	0,186444
0,750000	3,043368	2,848689	0,194679
0,788730	3,107970	2,905347	0,202623
0,811270	3,146451	2,939343	0,207108
0,850000	3,214195	2,999630	0,214565
0,888730	3,284137	3,062437	0,221700
0,911270	3,325923	3,100222	0,225701
0,950000	3,399698	3,167393	0,232305
0,988730	3,476131	3,237570	0,238562
1,000000	3,258584	3,258584	0,000000



Şekil 4.3 MAIN.C programının verdiği sonuçlara göre B-Spline fonksiyonlarını kullanarak bulunan yaklaşık çözümün grafiği (L = 10 için).



Şekil 4.4 MAIN.C programının verdiği sonuçlara göre B-Spline fonksiyonlarını kullanarak bulunan gerçek çözümün grafiği (L = 10 için).

KAYNAKLAR

1- Boor, Carl de, 1974. Good Approximation by Splines With Variable Knots. II in "Numerical Solutions of Differential Equations", Lecture Notes in Math. No.363, Springer Verlag, 12-20.

2- Boor, Carl de, 1978. A Practical Guide to Splines, Springer Verlag, New York.

ÖZGEÇMİŞ

Doğum tarihi	04.09.1975	
Doğum yeri	Hayrabolu, Tekirdağ	
Lise	1989-1992	Hayrabolu Lisesi
Lisans	1992-1996	Yıldız Teknik Üniversitesi Kimya-Metalurji Fak. Matematik Mühendisliği Bölümü
Çalıştığı kurum	1997-Devam ediyor	İstanbul Büyükşehir Belediyesi Bilgi İşlem Merkezi