kifmd

Kilis 7 Aralık Üniversitesi Fen ve Mühendislik Dergisi
2018 Cilt 2 Sayı 1

Mathematica® ile Sayısal Çözümleme Programları

Halil MUTUK
Ondokuz Mayıs Üniversitesi, Fen-Edebiyat Fakültesi, Fizik Bölümü, 55139, Samsun, Türkiye
İletişim yazarı: Halil Mutuk (halil.mutuk@omu.edu.tr)

Özet

Bu çalışmada, sayısal çözümleme derslerinde yer alan lineer ve lineer olmayan denklem sistemler ile diferansiyel denklemlerin çözümünde kullanılan bazı yaklaşık yöntemlerin Mathematica® yazılımı ile programları verilmiştir. Bu yöntemler lineer denklem sistemleri için Gauss yok etme, matris çarpımıyla Gauss yok etme, Aitken, lineer olmayan denklem sistemleri için basit iterasyon, Newton-Raphson, Sekant ve Regula-Falsi, diferansiyel denklemler için Euler ve Runge-Kutta yöntemleri kapsamaktadır.

Anahtar Kelimeler: Mathematica®, Sayısal Çözümleme, Yaklaşık Yöntemler, Bilgisayar Programlama

Abstract

In this work, we give some codes of approximate methods for linear, non-linear and differential equations written in Mathematica® software for numerical analysis courses. This methods consist of Gauss elimination, matrix product Gauss elimination and Aitken for linear equations system, iteration, Newton-Raphson, Secant and Regula-Falsi for non-linear equations and finally Euler and runge-Kutta methods for differential equations.

Keywords: Mathematica®, Numerical Analysis, Approximation Methods, Computer Programming

GİRİŞ

Sayısal çözümleme, diferansiyel denklemler, denklem çözümleri, doğrusal denklem takımlarının çözümü, türev, integral, eğri uydurma vb. tipteki analitik olarak çözülemeyen problemlerin sayısal olarak çözümleme işidir.

Fen ve mühendislik alanındaki güncel çalışma konularını kapsayan problemlerin birçoğu diferansiyel denklemleri içerir ve onlarla modellenir. Serbest düşen bir cismin hareketi, bir borudan akan sıvının hareketi ve elektromanyetik dalganın yayılımı gibi fiziksel olgular diferansiyel denklemleri içerir. Bir diferansiyel denklem bağımsız değişkenleri, bu değişkenlerin fonksiyonlarını ve bu fonksiyonların türevini içeren bir bağıntıdır [1]. Eğer bu denklem tek bir değişkene bağlı fonksiyon içeriyorsa *adi diferansiyel denklem,* birden fazla bağımsız değişkenin fonksiyonlarını içeriyorsa *kısmi diferansiyel denklem denir.* Genellikle diferansiyel denklem dendiğinde adi diferansiyel denklem kastedilir.

Diferansiyel denklemi çözmek kadar onu oluşturmak da önemlidir. Örneğin fiziksel olguların modellenmesinde bazı bir takım kabuller ve sınırlamalar getirebilir. Bu sınırlamalar başlangıç ve sınır değer problemlerinde görülebilir. Fiziksel modellemede ülke literatüründeki kitap sayısı çok azdır. Pala (2006) [2], bu modelleme açısından başvurulabilecek önemli bir kaynaktır. Sayısal çözümleme hem bilgisayar bilimlerini hem de matematik gibi pozitif bilimleri de içerdiğinden her iki taraftan da beslenir. Herhangi bir olguyu örneğin radyoaktif bozunma kanunu ele alalım. Bu olayın modellenmesi fizik kanunları altında matematiksel ifadelerle gerçekleşir. Sayısal çözümleme bu modeli uygun programlama dillerini kullanarak bilgisayar yardımıyla sayısal olarak çözme işidir. Daha genel bir tanımla sayısal çözümleme, matematiksel ifadelerle modellenen problemlerin çözümünde beli sayıda ve sırası belirlenmiş işlemleri bilgisayarlarla yaparak kesin veya belli bir hassasiyete sahip sonuçlar elde etmek için kullanılabilecek yaklaşık yöntemlerin bulunması, geliştirilmesi ve var olanlarından etkin olanının/olanlarının bulunması olarak tanımlanabilir [3].

Temel bilimler ve mühendislik dallarında karşılaşılan problemlerin çözümü iki ana gruba ayrılabilir: *analitik çözümler* ve *sayısal çözümler*. *Analitik çözümler,* ilgili problemlerden parametrelere ve değişkenlere bağlı bir ifade veya ifadeler grubu verdiğinden arzu edilen bir çözümdür. Bu parametreler ve değişkenler üzerinde istenilen değişiklikler yapılarak problem incelenebilir. *Sayısal çözümler,* analitik çözümlemenin yapılamadığı durumlarda ortaya çıkar [4]. Bu tip çözümler parametreler ve değişkenlerin belli kabulü altında çözümler oluşturulmaya çalışılır.

Temel bilimler ve mühendislik dalları içerisinde bazı problemlerin analitik olarak çözümü çok karmaşık olabilir. Hatta bazı durumlarda çözüm analitik olarak bulunamayabilir. Bugün halen bilinen tam bir analitik ve sayısal çözüme sahip olmayan diferansiyel denklemler vardır: matematikteki Korteweg- de Vries (KdV) lineer olmayan kısmi diferansiyel denklemi, Navier-Stokes denklemi gibi. Schrödinger denkleminin hidrojen atomu dışında kalan çok elektronlu atomlar için tam bir çözümü yoktur. Sayısal çözümleme bu tip problemlere yaklaşık çözümler bulmak için kullanılır.

Sayısal çözümleme ile her türlü problemin üstesinden gelinebileceği gibi bir algıya sahip olunabilir. Bilgisayarların teknik kapasitesine de bağlı olan bu algı aslında daha temel kuramsal bir olguyla da tezat düşer: *hesaplanabilirlik kuramı*. Bu kuram ilk olarak matematikte yer almış olsa da bilgisayar bilimlerinde de karşılığı vardır. Bu karşılık kendini algoritmalarda bulur. Matematikte, sayısal çözümlemede ve bilgisayar bilimlerinde herhangi bir algoritma ile çözülemeyecek problemler vardır. Hesaplanabilirlik ile ilgili Brookshear ve Brylow'un (2016)kitabına başvurulabilir.

Sayısal çözümleme bazen doğru sonucu verirken bazen de yaklaşık sonucu verir. Fen ve mühendislik ile ilgili bir problemin algoritmasında yer alan girdilerin bir kısmını deneysel veriler oluşturabilir. Bu deney verileri muhakkak bir hata payı içerir. Bunun neticesinde algoritmanın vereceği sonuç bir hata payı içerisinde doğru olacaktır. Bazen bu hatalar *kesme hatası* olarak bilinen algoritma kaynaklı olabilir. Bir diğeri ise bilgisayardaki aritmetik işlemlerden doğan *yuvarlama hatası*dır.

Bilgisayar bilimlerinin ve teknolojisinin ilerlemişiyle bilgisayarlar modern bilimde amaçları farklı olsa da çokça kullanılmaktadır. Bilgisayar bilimleri, bilgisayar tasarımı, programlama ve algoritmaların geliştirilmesi ve uygulanmasını içeren bir disiplindir [5]. Algoritmalar belki de bu disiplinin en önemli bileşenidir.

Algoritmalar bilgisayar bilimlerinin olduğu kadar, sayısal çözümlemenin de temelidir. Algoritma, bir problemin çözümü için gerekli olan sonlu sayıdaki işlemleri adım adım belirli bir sıra ile tanımlayan düzendir [3]. Programlama ile algoritma birbirlerine yakın kavramlar olsa da farklı tanımlamalara sahiptirler. Bir probleme dair çözümün oluşturulup bilgisayar ortamına aktarılmasına programlama, bu çözümün oluşturulmasına da algoritma oluşturma denir. Bir bilgisayar programı bilgisayara verilen görevleri yerine getirmesi için ne yapması gerektiğini içeren komutlar bütünüdür. Programlamanın kendine has bir metodolojisi vardır. Bunlar *yöntemsel programlama, fonksiyonel programlama, nesneye dayalı programlama tekrarlı programlama, kural temelli programlama* ve *mantıksal programlamadır*. Bu makalenin amacı her ne kadar bu alt dalların üzerine kapsayıcı bir şekilde bilgi vermek olmasa da kısaca değinmek yararlı olacaktır. *Yöntemsel programlama,* belirli bir algoritmayı yerine getiren bir dizi açıklamayı kapsayan programlamadır. Bu programlama türünde programı yazan insan bilgisayara ne yapacağını ve nasıl yapacağını söyleyen komutlar yazar. *Fonksiyonel programlama,* programın veya yazılımın fonksiyonlardan oluştuğu temeline dayanır. Birçok durumda bu fonksiyonlar iç içe geçmiştir ve karşılaşabilecek yeni fonksiyonlar bilinen diğer fonksiyonlar yardımıyla oluşturabilir. *Nesneye dayalı programlama,* fonksiyonel programlamaya benzerdir fakat burada fonksiyonların yerini nesneler almıştır. Programda ya da yazılımda çeşitli nesneler tanımlanır ve bu nesnelere gerektiğinde farklı işlemler yaptırmak için yöntem belirlenir. *Tekrarlı programlamada,* ilgili problemi çözmek için yazılan program isteğe göre parçalara ayrılır. Bu parçalara göre yazılan komutlar bir sonraki duruma göre birleştirilebilir veya bir önceki komuta başvurarak yeni veriler/girdiler oluşturabilir. *Kural temelli programlama,* belli dönüşümleri kurallara bağlayarak formüle eder. Bu şekilde genellikle sayısal hesaplamalar yapılır ve probleme odaklı olduğundan kapsamları dar olabilir. *Mantıksal programlama,* program diline bağlı kalmaksızın yapmak istenilen şeyi mantıksal tanımlarla kurulan algoritmalarla yapmak ister. Bu tip programlamada sonuca ulaşmanın belirli bir yolu yoktur. Sistem bütün yollar arasından istenen yolu seçer. Algoritmalar ve programlama için [6-7] kaynaklarına bakılabilir.

Bilgisayar programlamada amaç algoritmayı bilgisayara *tercüme ederek* aktarmaktır. Buradaki tercüme etmekten, istenilen algoritmanın, kodun veya programın makine diline çevrilmesidir. Bu tercüme etme işlemi iki şekilde olur: derleyici (compiler) ve çözümleyici/yorumlayıcı (interpreter). Yani bilgisayarın anlayabileceği iki dil vardır denebilir. Bu sayede programdaki/yazılımdaki kodlar makine diline derlenir ve/veya çözümlenir.

Mathematica fizik alanında doktora derecesine sahip olan Stephen Wolfram [14] tarafından ilk sürümü 1991 yılında yayımlanmış olan bir paket programdır. Mathematica matematiksel, cebirsel ve sembolik hesaplamalar yapan genel bir sistemdir. Üstün bir hesaplama ortamı sağlayan Mathematica aynı zamanda da bir programlama

dilidir. "İç verimlilik oranı temelinde en iyi stratejinin belirlenmesi" veya "yıllık dönem temelinde taksit tutarını hesaplayan" programları da Mathematica aracılığıyla yazabilirsiniz (Çınar ve Çalışkan, 1995). Mathematica, bu çalışmada bahsedilen programlama metodolojilerinden yöntemsel ve mantıksal olanları ile pek uyumlu değildir. Bilgisayar dili olarak çözümleyici/yorumlayıcı bir dildir. Derleyici diller, çözümleyici dillerden farklı olarak makine diline daha yakın olmaları ve bilgisayar donanımından aracı olmaksızın yararlanabildikleri için etkindirler.

Mathematica iki ana bölümden oluşur: matematiksel hesapları yapan *kernel* bölümü ve kullanıcı ile bilgisayarın iletişimini sağlayan *front end* bölümü [9]. Mathematica sahip olduğu kütüphane bakımından çok üstündür. Birçok matematiksel fonksiyon kütüphanesinde kullanıcılar için hazır olarak yer almaktadır. Mathematica sembolik bir yazılım olduğundan veri girişi çok kolaydır. Dolayısıyla yoğun hesaplamalar gerektiren işlemler için zaman kaybını ortadan kaldırır.

Dilimizde Mathematica için yazılmış olan az sayıda kaynak mevcuttur. Bunlardan birkaçı kaynaklarda görülebilir [8,10,11]. Diferansiyel denklemler ve sınır değer problemlerinin incelendiği ve çeviri editörünün Akın (2008) [12] olduğu kitap Maple, Matlab ve Mathematica örneklerini içeren zengin bir kaynaktır. Özellikle mühendislik fakültelerindeki sayısal çözümleme dersleri MatLab programlama diliyle yapılmaktadır. Bu programlama dilinde ise dilimizde yeterli sayıda kitap vardır.

Bu çalışmanın ikinci bölümünde lineer denklem sistemlerinin çözümüne genel bir giriş verilmiş ve bu sistemlerin çözümü için Gauss yok etme, matris çarpımlarıyla Gauss yok etme ve Aitken yöntemleri ile ilgili programlar verilmiştir. Lineer olmayan denklemlerin çözümü için, basit iterasyon, Newton-Raphson, sekant ve Regula-Falsi yöntemleri ile ilgili programlar verilmiştir. 3. Bölümde diferansiyel denklemler için Euler ve 4. Mertebeden Runge-Kutta yöntemlerini içeren programlar verilmiştir. Tartışma ve sonuç bölümünde Mathematica ile verilen progamların sayısal çözümleme derslerinde örnek olarak kullanılabileceğini ve Mathematica'nın programlamada daha yaygın bir şekilde kullanılması tartışılacaktır.

1. LİNEER DENKLEM SİSTEMLERİNİN ÇÖZÜMÜ

Lineer denklem sistemleri fen ve mühendislik problemlerinde sıkça karşılaşılan bir sistemdir. Örneğin fizikte elektrik devre analizinde kullanılan Kirchoof kuralları neticesinde bir denklem sistemi elde edilir.

Genel olarak lineer denklem sistemi

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n} &= c_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n} &= c_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn} &= c_m \end{aligned} \tag{1}$$

şeklinde gösterilir. Bu denklem sistemi matris gösteriminde

$$AX = C \tag{2}$$

daha sade bir biçimde yazılabilir. Bu denklem sisteminde C vektörünün $C = 0$ olması halinde sisteme homojen denklem takımı denir ve $X = 0$ gibi bir çözüme sahiptir. Bu çözüme aşikar çözüm denir. Bundan başka bir çözüm varsa bu çözüme aşikar olmayan çözüm denir. $C \neq 0$ olması halinde sisteme homojen olmayan denklem takımı denir.

1.1.Gauss Yok Etme Yöntemi

Gauss yok etme yöntemi anlaşılması ve uygulanması kolay bir yöntem olduğundan hem elle yapılabilir hem de bilgisayar ortamında rahatlıkla programlanabilir. Fakat büyük boyutlu matris denklemlerinde işlem sayısı arttığı için

buna bağlı olarak hata da artabilmektedir. (1) denkleminde A matrisinden katsayılar matrisi, C matrisinden de sonuç matrisi şu şekilde oluşturulabilir:

$$[A|C] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \text{L} & a_{1n}\ c_1 \\ a_{21} & a_{22} & a_{23} & \text{L} & a_{2n}\ c_2 \\ a_{31} & a_{32} & a_{33} & \text{L} & a_{3n}\ c_3 \\ \text{M} & \text{M} & \text{M} & & \text{M}\ \text{M} \\ a_{n1} & a_{n2} & a_{n3} & \text{L} & a_{nn}\ c_n \end{bmatrix} \quad 3)$$

Açık hali yazılırsa

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots\ldots + a_{1n}x_n - \ \text{c}_1 \ = \ 0 \\ &a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots\ldots + a_{2n}x_n - c_2 = 0 \\ &a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots + a_{3n}x_n - \ \text{c}_3 = \ 0 \\ &\text{M} \\ &a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots + a_{nn}x_n - \ \text{c}_n = \ 0 \end{aligned} \quad 4)$$

elde edilir ve aşağıdaki işlemler bu ifade üzerinden daha net anlaşılabilir. Bu denklemin matris elamanlarına satır ve sütun işlemleri uygulanabilir ve şu sırada takip edilebilir [4]:

1. a_{11} pivot eleman olarak seçilir. Birinci satır a_{21}/a_{11} sayısı ile çarpılıp ikinci satırdan çıkarılır. Elde edilen elemanlar ikinci satır yerine yazılır. Elde edilen bu yeni satırda $a_{21} = 0$ olacaktır.
2. Birinci satır a_{31}/a_{11} ile çarpılıp üçüncü satırdan çıkarılacak ve elde edilen yeni elemanlar üçüncü satır yerine yazılacaktır. Elde edilen bu yeni satırda $a_{31} = 0$ olacaktır.
3. Aynı işlemler sırayla n'yinci satıra kadar yapılır ve işlemler sonunda $a_{i1} = 0$ olacaktır.
4. a_{22} pivot eleman olarak seçilir ve a_{i2} $(i = 3,4,\cdots n)$ elemanları sıfırlanır. Bu sıfırlama (5) nolu denklem vasıtasıyla olacaktır.
5. Köşegenin alt tarafındaki bütün elemanlar sıfır olana kadar yukarıdaki işlemler devam ettirilir.
6. Alt üçgen matris elemanları sıfırlandıktan sonra $x_n = c_n/a_{nn}$ formunda bir ifade bulunacaktır.
7. Bunun peşine geriye doğru işlem yaparak sırayla x_{n-1}, x_{n-2}, ⋯, x_1 elemanları bulunur. x_i değerleri için (6) nolu denklem kullanılır. Sonuç olarak bütün bilinmeyenler bulunur.

$$\begin{aligned} &k = 1,2,\cdots,n-1 \ için \\ &x_{ij} \rightarrow x_{ij} - \frac{x_{ik}}{x_{kk}} x_{kj} \end{aligned} \quad 5)$$

$$\begin{aligned} x_n &= \frac{1}{c_n} a_{nn} \\ x_i &= \frac{1}{a_{ii}} \left(c_i - \sum_{k=i+1}^{n} a_{ik} x_k \right) \end{aligned} \quad 6)$$

Bu adımlardan sonra ilgili Mathematica programı Şekil 1'de verilmiştir. Programın başındaki $Clear$ komutu program her çalıştırıldığında daha önceki atanan değerleri sıfırlar. n matris mertebesi olup istenildiği kadar büyütülebilir.

```
Clear[n, a, c, x];
n = 3;
a = Array[0 &, {n, n}];
c = Array[0 &, {n, 1}];
x = Array[0 &, {n, 1}];
a = {{2, 0.3, -0.7}, {0.8, 2, 5}, {0.6, -0.2, 9}};
c = {{4}, {-2}, {3.6}};
Print["Verilen denklem takımı çarpan matrisi: A=", a // MatrixForm];
Print["Verilen denklem takımı katsayı matrisi: C=", c // MatrixForm];
For[k = 1, k ≤ n - 1, k++,
  For[i = k + 1, i ≤ n, i++,
   p = a[[i, k]] / a[[k, k]];
   c[[i]] = c[[i]] - p * c[[k]];
   For[j = k + 1, j ≤ n, j++,
    a[[i, j]] = a[[i, j]] - p * a[[k, j]]]
  ]
 ];
x[[n]] = c[[n]] / a[[n, n]];
For[k = 2, k ≤ n, k++,
  i = n - k + 1;
  topj = 0;
  For[j = i + 1, j ≤ n, j++,
   topj = topj + a[[i, j]] * x[[j]]];
  x[[i]] = (c[[i]] - topj) / a[[i, i]]
 ];
Print["Verilen denklem takımı çözüm sütunu: X=", x // MatrixForm]
```

Verilen denklem takımı çarpan matrisi: A=
$$\begin{pmatrix} 2 & 0.3 & -0.7 \\ 0.8 & 2 & 5 \\ 0.6 & -0.2 & 9 \end{pmatrix}$$

Verilen denklem takımı katsayı matrisi: C=
$$\begin{pmatrix} 4 \\ -2 \\ 3.6 \end{pmatrix}$$

Verilen denklem takımı çözüm sütunu: X=
$$\begin{pmatrix} 2.42916 \\ -2.43171 \\ 0.184018 \end{pmatrix}$$

Şekil 1. Gauss yok etme yöntemi için Mathematica dilindeki program

1.2.Matris Çarpımıyla Gauss Yok Etme

Bir önceki kısımda verilen matris satır işlemlerini sırayla takip etmek kimi zaman karışıklığa sebep olabilir. Benzer bir yok etme işlemi matris çarpımlarıyla da yapılabilir (Aktaş vd, 1991). $AX = C$ matris elemanları şu şekilde yazılabilir:

$$\begin{bmatrix} a_{11} & a_{12} & \mathrm{L} & a_{1n} \\ a_{21} & a_{22} & \mathrm{L} & a_{2n} \\ \mathrm{M} & \mathrm{M} & \mathrm{L} & \mathrm{M} \\ a_{n1} & a_{n2} & \mathrm{L} & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \mathrm{M} \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \mathrm{M} \\ c_n \end{bmatrix} \qquad 7)$$

Gauss yok etme yönteminde elde edilen ilk indirgenmiş denklem takımı matris çarpımı olarak şu şekilde elde edilebilir:

$$L_1 AX = L_1 C. \tag{8}$$

Burada

$$L_1 = \begin{bmatrix} 1 & & & \\ p_{21} & 1 & & \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \\ p_{n1} & p_{n2} & \mathrm{L} & 1 \end{bmatrix} \tag{9}$$

ve

$$p_{i1} = -\frac{a_{i1}}{a_{11}} \tag{10}$$

olarak tanımlıdır. İkinci indirgeme

$$L_2 L_1 AX = L_2 L_1 C \tag{11}$$

işlemi ile yapılır. Burada

$$L_2 = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & p_{32} & 1 & & \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{O} & \\ 0 & p_{n2} & \mathrm{L} & \mathrm{L} & 1 \end{bmatrix} \tag{12}$$

olarak verilir. Bu indirgeme işlemlerine

$$\begin{aligned} L_3 L_2 L_1 AX &= L_3 L_2 L_1 C \\ L_4 L_3 L_2 L_1 AX &= L_4 L_3 L_2 L_1 C \\ &\vdots \\ L_{n-1} \cdots L_3 L_2 L_1 AX &= L_{n-1} \cdots L_3 L_2 L_1 C \end{aligned} \tag{13}$$

şeklinde devam edilebilir. Alt üçgen matrislerin çarpımının da alt üçgen matris olacağı düşünülerek yukarıdaki son denklem

$$LAX = LC \tag{14}$$

formunda yazılabilir. $LA = U$ üst üçgen matris olmak üzere (14) nolu denklem şu şekilde yazılabilir:

$$UX = LC. \tag{15}$$

Buradan hareketle

$$X = U^{-1}LC \tag{16}$$

çözümü bulunabilir. Şekil 2'de bu yöntemin Mathematica programı verilmiştir.

```
Clear[nx, ax, cx, x, aters];
nx = 3;
ax = Array[0 &, {nx, nx}];
cx = Array[0 &, {nx, 1}];
x = Array[0 &, {nx, 1}];
ax = {{2, 0.3, -0.7}, {0.8, 2, 5}, {0.6, -0.2, 9}};
cx = {{4}, {-2}, {3.6}};
Print["Verilen denklem takımı çarpan matrisi: A=", ax // MatrixForm];
Print["Verilen denklem takımı katsayı matrisi: C=", cx // MatrixForm];
gaussjordan[n_, c_] := Module[{gj},
   b = IdentityMatrix[n];
   a = Array[0 &, {n, n}];
   a = c;
   For[i = 1, i ≤ n, i++,
        p = a[[i, i]];
        For[j = 1, j ≤ n, j++,
             a[[i, j]] = a[[i, j]] / p;
             b[[i, j]] = b[[i, j]] / p];
            For[k = 1, k ≤ n, k++,
                 If[k =!= i,
                  p = a[[k, i]];
               For[j = 1, j ≤ n, j++,
                    a[[k, j]] = a[[k, j]] - p * a[[i, j]];
                    b[[k, j]] = b[[k, j]] - p * b[[i, j]]]]]];
   gj = b];
matriscarp[v_, y_, o_, p_] := Block[{tot},
   cm = Array[0 &, {v, y}];
   ust = Dimensions[o][[2]];
   For[im = 1, im ≤ v, im++,
    For[jm = 1, jm ≤ y, jm++,
     cm[[im, jm]] = 0;
     For[km = 1, km ≤ ust, km++,
      cm[[im, jm]] = cm[[im, jm]] + o[[im, km]] * p[[km, jm]]]]];
   tot = cm];
aters = gaussjordan[nx, ax];
x = matcarp[3, 1, aters, cx];
Print["X çösüm matrisi=", x // MatrixForm]
```

Verilen denklem takımı çarpan matrisi: $A=\begin{pmatrix} 2 & 0.3 & -0.7 \\ 0.8 & 2 & 5 \\ 0.6 & -0.2 & 9 \end{pmatrix}$

Verilen denklem takımı katsayı matrisi: $C=\begin{pmatrix} 4 \\ -2 \\ 3.6 \end{pmatrix}$

X çösüm matrisi=$\begin{pmatrix} 2.42916 \\ -2.43171 \\ 0.184018 \end{pmatrix}$

Şekil 2. Matris çarpımıyla Gauss yok etkme yöntemi için Mathematica dilindeki program

1.2. Aitken Yöntemi

Bu kısımdaki inceleme Aktaş vd. (1991) kaynağından yararlanılarak yapılmıştır. Bu yöntem, $AX = C$ denkleminden hareketle A, B ve C matrislerinin çarpmaya uygun olduğu kabulünde

$$Y = BA^{-1}C \tag{17}$$

bağıntısını kullanarak çözümü bulur. Bu bağıntı $AX = C$ ve $BX = Y$ denklemlerinden hareketle bulunmuştur. A matrisinin boyutu $(m \times m)$, B'nin boyutu $(p \times m)$ ve C'nin boyutu da $(m \times n)$'dir. $AX = C$ denklem takımı bir alt üçgen matrisle seri çarpıma tutulursa $UX = F$ denklem takımı elde edilir. U, B, F ve 0 matrislerinden oluşan

$$\begin{bmatrix} U & \text{M} & F \\ \text{K} & \text{M} & \text{K} \\ B & \text{M} & 0 \end{bmatrix} \tag{18}$$

matrisi ele alalım. 0 matrisi boyutu $(p \times n)$olan bir sıfır matristir. U matrisinin ilk satırını uygun katsayılarla çarparak B matrisinin satırlarına her satırın ilk terimi sıfır olacak şekilde ilave edilir ve bu işlem diğer satırlara da uygulanır. Bu işlem sonucunda B matrisinin bütün elemanları sıfırlanacaktır. Aynı işlemi F ve 0 matrisleri arasında yapılırsa 0 matrisinin elemanları aynı zamanda $-BU^{-1}F$ matrisinin de elemanları olur. Bu sonuç matris gösteriminde kolaylıkla görülebilir:

$$\begin{bmatrix} I & \text{M} & 0 \\ \text{K} & \text{M} & \text{K} \\ K & \text{M} & I \end{bmatrix} \begin{bmatrix} U & \text{M} & F \\ \text{K} & \text{M} & \text{K} \\ B & \text{M} & 0 \end{bmatrix} = \begin{bmatrix} U & \text{M} & F \\ \text{K} & \text{M} & \text{K} \\ P & \text{M} & Q \end{bmatrix}. \tag{19}$$

Bu matris çarpımından

$$P = 0 = KU + B \tag{20}$$

$$Q = KF \tag{21}$$

denklemleri yazılabilir. $UX = F$ ve $AX = C$ denklemlerinden hareketle

$$\text{Q} = -\text{BU}^{-1}C \tag{22}$$

bulunur. Yukarıdaki işleme benzer şekilde

$$\begin{bmatrix} I & \text{M} & 0 \\ \text{K} & \text{M} & \text{K} \\ K & \text{M} & I \end{bmatrix} \begin{bmatrix} A & \text{M} & C \\ \text{K} & \text{M} & \text{K} \\ B & \text{M} & 0 \end{bmatrix} = \begin{bmatrix} A & \text{M} & C \\ \text{K} & \text{M} & \text{K} \\ KA + B & \text{M} & KC \end{bmatrix} \tag{23}$$

matris çarpımını yapar ve $KA + B = 0$ olacak şekilde bir K matrisi seçilirse $KC = -BA^{-1}C$ yazılabilir. A matrisinin U matrisine, B matrisinin de 0 matrisine dönüşümü aynı anda pivot elemanlar A matrisinden seçilerek gerçekleştirilebilir. Bu durumda

$$\begin{bmatrix} L & \text{M} & 0 \\ \text{K} & \text{M} & \text{K} \\ K & \text{M} & I \end{bmatrix} \begin{bmatrix} A & \text{M} & C \\ \text{K} & \text{M} & \text{K} \\ B & \text{M} & 0 \end{bmatrix} = \begin{bmatrix} LA & \text{M} & LC \\ \text{K} & \text{M} & \text{K} \\ KA+B & \text{M} & KC \end{bmatrix} \qquad 24)$$

ifadesi yazılabilir. Burada L birim alt üçgen matrisidir ve K matrisi $KA + B = 0$ eşitliğini sağlamalıdır (Aktaş vd, 1991). Şekil 3'te Aitken programı Mathematica'da verilmiştir.

```
Clear[nx, ax, cx, x, aters, aterss];
nx = 3;
ax = Array[0 &, {nx, nx}];
cx = Array[0 &, {nx, 1}];
x = Array[0 &, {nx, 1}];
ax = {{2, 0.3, -0.7}, {0.8, 2, 5}, {0.6, -0.2, 9}};
cx = {{4}, {-2}, {3.6}};
Print["Verilen denklem takımı çarpan matrisi: A=", ax // MatrixForm]
Print["Verilen denklem takımı katsayı matrisi: C=", cx // MatrixForm
matcarp[v_, y_, o_, p_] := Block[{tot},
   cm = Array[0 &, {v, y}];
   ust = Dimensions[o][[2]];
   For[im = 1, im ≤ v, im++,
    For[jm = 1, jm ≤ y, jm++,
     cm[[im, jm]] = 0;
     For[km = 1, km ≤ ust, km++,
      cm[[im, jm]] = cm[[im, jm]] + o[[im, km]] * p[[km, jm]]]]];
   tot = cm];
chlters[n_, a_] := Module[{ch},
   l = Array[0 &, {n, n}];
   u = Array[0 &, {n, n}];
   t = Array[0 &, {n, n}];
   s = Array[0 &, {n, n}];
   For[i = 1, i ≤ n, i++,
    l[[i, i]] = 1;
    For[j = 1, j ≤ n, j++,
     ustj = j - 1;
     usti = i - 1;
     If[usti == 0, u[[i, j]] = a[[i, j]];
      If[ustj =!= 0, l[[j, i]] = a[[j, i]] / u[[i, i]]],
      If[And[ustj =!= 0, i > j],
       topl = 0; For[k = 1, k ≤ ustj, k++,
        topl = topl + l[[i, k]] * u[[k, j]]];
       l[[i, j]] = (a[[i, j]] - topl) / u[[j, j]]];
      If[And[usti =!= 0, i ≤ j],
       topu = 0;
       For[k = 1, k ≤ usti, k++,
        topu = topu + l[[i, k]] * u[[k, j]]];
       u[[i, j]] = a[[i, j]] - topu]]]];
```

```
    For[i = 1, i ≤ n, i++,
     s[[i, i]] = 1; t[[i, i]] = 1 / u[[i, i]];
     For[j = 1, j ≤ n, j++,
      If[i =!= j,
       topt = 0; tops = 0;
       For[k = i, k ≤ j - 1, k++,
        topt = topt + (t[[i, k]] * u[[k, j]]) / u[[j, j]];
        t[[i, j]] = -topt;
        tops = tops + l[[j, k]] * s[[k, i]];
        s[[j, i]] = -tops]]]];
    aters = matcarp[n, n, t, s];
    ch = aters];
x = matcarp[3, 1, chlters[nx, ax], cx];
aterss = chlters[nx, ax];


Print["X katsayılar matrisi=", x // MatrixForm]
```

Verilen denklem takımı çarpan matrisi: $A=\begin{pmatrix} 2 & 0.3 & -0.7 \\ 0.8 & 2 & 5 \\ 0.6 & -0.2 & 9 \end{pmatrix}$

Verilen denklem takımı katsayı matrisi: $C=\begin{pmatrix} 4 \\ -2 \\ 3.6 \end{pmatrix}$

X katsayılar matrisi= $\begin{pmatrix} 2.42916 \\ -2.43171 \\ 0.184018 \end{pmatrix}$

Şekil 3. Aitken yok etme yöntemi için Mathematica dilindeki program

2. LİNEER OLMAYAN DENKLEM SİSTEMLERİNİN ÇÖZÜMÜ

Lineer olmayan denklemlerin çözümü matematikte eskiden beri çok ilgi çeken ve güncelliğini hiç kaybetmeyen bir uğraştır. Bu çözüm aslında bir denklemi sağlayan kök veya kökleri bulmaya eşdeğerdir. $[a, b]$ aralığında tanımlı bir $f(x)$ fonksiyonu için $f(x) = 0$ denklemini sağlayan x değerlerine bu fonksiyonun kökü denir.

Bütün kök bulma yöntemlerinde iterasyon tekniği kullanılır. Önce denklemi sağlayan tahmini bir x_1 noktasından başlayarak köke yaklaştıran bir kriter kullanılır ve genellikle önceden belirlenmiş bir hata payı ile kök bulunur [13]. Doğada gözlemlenen bir çok olgu genellikle doğrusal olmayan denklemlerle modellenir. Bunların analitik bir çözümü olmadığı için belli yaklaşımlarla çözülebilir. Bu bölümde bu tip bir kök bulma için basit iterasyon, Newton-Raphson, Sekant ve Regula-Falsi yöntemleri ile ilgili programlar verilecektir.

2.1. Basit İterasyon Yöntemi

Sabit nokta iterasyonu olarak da bilinen bu yöntemde lineer olmayan $f(x) = 0$ denklemi

$$x = g(x) \tag{25}$$

formuna dönüştürülür ve

$$x_{k+1} = g(x_k) \tag{26}$$

şeklinde iterasyon yapılarak kök bulmaya çalışılır. Elde edilen çözümün yakınsaklığı için

$$|g'(x_k)| < 1 \tag{27}$$

şartı sağlanmalıdır. Bir örnekle detaylandıralım [3].

$$f(x) = x^3 - x - 1 = 0 \tag{28}$$

denkleminin $x_0 = 1.3$ civarında bir kökü olduğu bilinmektedir. Basit iterasyonla çözüm için denklemi (25) nolu denklem formuna dönüştürmek gerekir:

$$x = x^3 - 1 \rightarrow g(x) = x^3 - 1 \rightarrow g'(x) = 3x^2 \rightarrow |g'(x_0)| > 1 \tag{29a}$$

$$x = \frac{1}{x^2 - 1} \rightarrow g(x) = \frac{1}{(x^2 - 1)} \rightarrow g'(x) = -\frac{2x}{(x^2 - 1)^2} \rightarrow |g'(x_0)| > 1 \tag{29b}$$

$$x = (x+1)^{1/3} \rightarrow g(x) = (x+1)^{1/3} \rightarrow g'(x) = \frac{1}{3}(x+1)^{-2/3} \rightarrow |g'(x_0)| < 1 \tag{29c}$$

Bu ifadelerden sadece (29c) nolu olanı basit iterasyonla çözülebilir. Çünkü ancak bu şekilde yakınsama sağlanır. Şekil 4'te $[a, b] = [0,1]$ aralığında tanımlı $f(x) = 0.7 - x + 0.3 \sin x$ denkleminin $x_0 = 0.5$ başlangıç değerinde $g(x) = 0.7 + 0.3 \sin x$ için iterasyon sayısını gösteren ve yaklaşık kökünü bulan Mathematica programı verilmiştir.

```
Clear[g, a, b, m0]
g[x_] = 0.7 + 0.3 * Sin[x];
m0 = 0.5;
a = 0;
b = 1;
Clear[m, k]
m[0] = N[m0];
Do[m[k] = N[g[m[k - 1]]], {k, 1, 100}]
TableForm[Table[{k, m[k]}, {k, 0, 11}]]
```

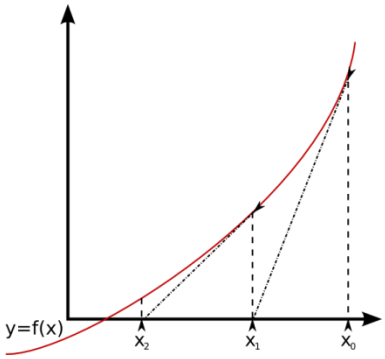
```
0     0.5
1     0.843828
2     0.924158
3     0.939434
4     0.942167
5     0.94265
6     0.942736
7     0.942751
8     0.942753
9     0.942754
10    0.942754
11    0.942754
```

Şekil 4. Basit iterasyon için Mathematica'da bir program

2.2. Newton-Raphson Yöntemi

Bu yöntem bir denklemin köklerini bulmada kullanılan en etkili yöntemlerden biridir. Doğrusal olmayan denklemlere ve fonksiyonlara uygulanabileceği gibi, doğrusal denklem ve fonksiyonlara da uygulanabilir. Sadece

bir başlangıç değeri ile başlanması ve fonksiyonun türevini de içermesi en önemli özelliğidir. Diğer bir deyişle fonksiyonun türevinin tanımlı olması bu yöntem için gerekli şarttır.



Şekil 5. Newton-Raphson yöntemi grafiği

Bu yöntem $f(x)$ eğrisine teğet olan doğrunun eksenine ulaştığı noktayla devam edilerek denklemi sağlayan köke ulaşmayı amaçlar. Şekil 5'teki x_0 noktasında $f(x)$ eğrisine teğet olan doğrunun denklemi

$$y - f(x_0) = f'(x_0)(x - x_1) \tag{30}$$

olarak yazılır. Bu teğetin x eksenini kestiği x_1 noktasını bulmak için $y = 0$ alınırsa

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{31}$$

bulunan x_1 noktası köke daha yakındır. x_1 noktasındaki teğetten faydalanarak x_2 noktası bulunur. Bu şekilde devam ettirilerek Newton-Raphson yönteminin genel formülü elde edilir:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{32}$$

Tekrarlama işlemi belli bir duyarlılığa ulaşıncaya ve kökü buluncaya kadar devam edecektir. Bu duyarlılık verilen bir ϵ değeri için $|x_{n+1} - x_n| < \epsilon$ şartı ile sağlanır. Şekil 6'da Newton-Raphson yönteminin Mathematica dilindeki programı verimiştir.

```
f[x_] = 0.7 - x + 0.3 Sin[x];
p0 = 0.5;
a = 0;
b = 1;
g[x_] = x - f[x]/f'[x];
Clear[p]
p[0] = p0;
Do[p[n] = N[g[p[n - 1]], 50], {n, 1, 100}]
TableForm[Table[{n, p[n]}, {n, 0, 10}]]

0     0.5
1     0.966697
2     0.942839
3     0.942754
4     0.942754
5     0.942754
6     0.942754
7     0.942754
8     0.942754
9     0.942754
10    0.942754
```

Şekil 6. Newton-Raphson yönteminin Mathematica'daki bir programı

2.3. Sekant Yöntemi

Newton-Rapshon yönteminde fonksiyonun türevini de bilinmesini gerektirdiğinden bazı problemlerde sorun oluşturabilir. Sekant yöntemi (32) nolu ifadeyi bu türevden arındırdığı için Newton-Rapshon yöntemine göre avantaj sağlar. Bir $f(x_n)$ fonksiyonun türevi

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{33}$$

ile bulunur. Bu ifade, (32) nolu denklemde yerine yazılırsa

$$x_{n+1} = \frac{x_{n-1}\, f(x_n) - x_n\, f(x_{n-1})}{f(x_n) - f(x_{n-1})} \tag{34}$$

bağıntısı elde edilir ki bu sekant yönteminin bağıntısıdır. Bu bağıntı ile $f(x) = 0$ denkleminin yaklaşık kökünü bulma yöntemi de olduğundan kesen yöntemi de denir. Sekant (kesen) yönteminde $f'(x)$ türevinin analitik olarak hesaplanmasına gerek yoktur. Bu yöntemle ilgili bir diğer önemli husus, başlangıç değerleri olarak iki değer kullanılsa da kökün bu aralıkta olmasına gerek yoktur. Şekil 7'de ilgili program verilmiştir.

```
f[x_] = 0.7 - x + 0.3 * Sin[x]
p[0] = 0.5;
p[1] = 1;
Do[p[n] = N[p[n - 1] - f[p[n - 1]] * (p[n - 1] - p[n - 2]) / (f[p[n - 1]] - f[p[n - 2]]), 50], {n, 2, 100}]
TableForm[Table[{n, p[n]}, {n, 0, 10}]]
```

```
0.7 - x + 0.3 Sin[x]
```

```
Power::infy : Infinite expression 1/0. encountered. >>
Infinity::indet : Indeterminate expression 0. ComplexInfinity encountered. >>
```

0	0.5
1	1
2	0.939243
3	0.942724
4	0.942754
5	0.942754
6	0.942754
7	0.942754
8	Indeterminate
9	Indeterminate
10	Indeterminate

Şekil 7. Sekant yönteminin Mathematica'daki bir programı

Şekil 7'den de görüleceği üzere Mathematica iterasyondaki bir adımda $\frac{1}{0}$ ile karşılaştığı için tanımsız uyarısı vermiştir. Başka denklemlerde ilgili başlangıç değerleri için bu uyarı olmayabilir.

2.4. Regula-Falsi Yöntemi

Değişken kesen ya da ters konumdaki noktalar yöntemi olarak da bilinen bu yöntemde sekant (kesen) yöntemindeki formül kullanılır ve iki tane başlangıç değeri gereklidir. İterasyon için kullanılacak iki nokta kökün birer tarafından alınır. Diğer bir deyişle $f(x_{n-1})$ ve $f(x_n)$ değerleri ters işaretli olup $f(x_{n-1})f(x_n) < 0$ şartı sağlanır.

Yöntemin ilerleyişi şu şekildedir: (34) nolu denklemden hareket edilerek $f(x_0)f(x_1) < 0$ şartını sağlayan x_0 ve x_1 noktaları seçilir. Bu değerler (34) nolu bağıntıda yerine yazılırsa x_2 bulunur:

$$x_2 = \frac{x_0\, f(x_1) - x_1\, f(x_0)}{f(x_1) - f(x_0)}. \qquad 35)$$

Bu x_2 değeri bulunduktan sonra $f(x_1)f(x_2) < 0$ şartını sağlandıktan sonra x_3 değeri bulunur ve işlemler devam ettirilir. x_n ve x_{n+1} noktalarından x_{n+2}'yi elde etmek için (Aktaş vd, 1991)

a) eğer $f(x_{n+1})f(x_{n-1}) < 0$ ise x_{n+1} ve x_{n-1} kullanılır.
b) eğer $f(x_{n+1})f(x_{n-1}) > 0$ ise x_{n+1} ve x_n kullanılır.

Şekil 8'de yöntemin Mathematica'daki programı verilmiştir. İterasyon sayısı artırıldığından programın çalışma süresi uzamakla beraber daha doğru sonuç elde edilir.

```
n = 100;
x = Array[0, n];
x[[0]] = -2.;
x[[1]] = 0.;
a = 1.;
f[x_] := a^x + x^3 - x;
eps = 0.00001
iter = 1;
k = 1;
While[And[k == 1, iter ≤ n],
  y = f[x[[iter]]];
  x[[iter + 1]] = x[[iter]] - (x[[iter]] - x[[iter - 1]])/(y - f[x[[iter - 1]]]) * y;
  If[Abs[x[[iter + 1]] - x[[iter]]] <= eps,
   k = 2;
   Print["x= ", N[x[[iter + 1]]]]; Print["iter= ", iter],
   iter = iter + 1;
   Print[iter];
  ];
 ];
```

```
0.00001

2

3

4

5

x= 1.125

iter= 5
```

Şekil 8. *Regula-Falsi yönteminin Mathematica'daki örnek bir programı*

3. DİFERANSİYEL DENKLEMLERİN YAKLAŞIK ÇÖZÜMLERİ

Diferansiyel denklemler fizik, mühendislik ve uygulamalı matematik alanlarında yer bulur. Özellikle fiziksel olgular çoğunlukla diferansiyel denklemlerle modellenir ve diferansiyel denklemin çözümleriyle açıklanır. Analitik çözümlere sahip olmayan diferansiyel denklemlerin çözümünde sayısal yöntemler devreye girer. Bununla birlikte analitik olarak çözülebilen denklemlere de sayısal yöntemler uygulanabilir.

Çözüm tekniği bakımından diferansiyel denklemler üçe ayrılır [13]:

1. Başlangıç Değer Problemleri: Bu tip problemlerde bir noktadan hareketle çözüm araştırılır. Örneğin zamana bağlı fiziksel problemlerde $t = 0$ başlangıç anında sistemin durumu belirlenir ve bu durumdan hareketle daha sonraki t değerleri için çözüm araştırılır. Bu tip problemlerde sistemin dinamiği üzerinde başka bir sınırlama yoktur.

Sistemin evrimi bağımsız değişkenin bir değeri için belirlenir. $\frac{dy}{dt} = f(y, y', t)$ diferansiyel denkleminde bir başlangıç şartına ihtiyaç varken, $\frac{d^2y}{dt^2} = f(y, y', t)$ denkleminin iki başlangıç şartına ihtiyaç vardır. Genel olarak $n.$ dereceden diferansiyel denklem için n tane başlangıç koşulunun verilmesi gerekir.

2. Sınır Değer Problemleri: Bu tip problemlerde belli bir bölge sınırında koşullar verilir ve çözüm o bölge dahilinde aranır. Bu bölgenin sınırları bağımsız değişkenin iki farklı değeri ile belirlenir. Örneğin $\frac{d^2y}{dx^2} = f(y, y', x)$ denkleminin $x = [0, L]$ aralığında çözümü araştırılacaksa $y(0)$ ve $y(L)$ değerlerinin verilmesi gerekir.

3. Özdeğer Problemleri: Bu tip problemlerde, diferansiyel denklemin sınır koşulları verilmiş olsa bile problemdeki bir parametrenin ancak belli değerleri için çözüm vardır. Fizikteki Schrödinger ve Klein-Gordon dalga denklemleri özdeğer problemidir ve sadece belli enerji özdeğerlerinde çözülebilir.

Bu bölümde diferansiyel denklemlerin sayısal çözümleri için Euler ve Runge-Kutta yöntemleri verilecektir. Bununla birlikte literatürde diferansiyel denklemlerin sayısal çözümleri için çok fazla yöntem vardır: Adomnian Ayrıştırma Yöntemi, Diferansiyel Dönüşüm Yöntemi, Pertürbatif Yöntem gibi. Bu da diferansiyel denklemlerde sayısal çözümlemenin çok canlı olduğunun kanıtıdır.

3.1. Euler Yöntemi

Herhangi bir fonksiyonun $x = x_0$ noktasındaki Taylor açılımı

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \frac{y'''(x_0)}{3!}(x - x_0)^3 + \cdots \quad 36)$$

ile verilir. Bu seride birinci türevden sonraki terimlerin ihmal edilmesiyle

$$y(x) \cong y(x_0) + y'(x_0)(x - x_0) + hata \quad 37)$$

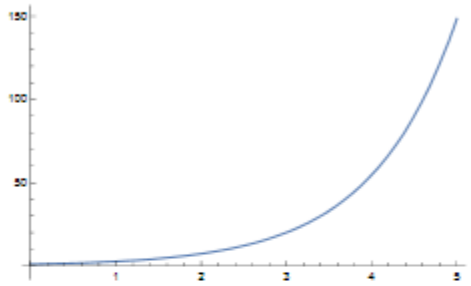
olarak elde edilen denklemin iteratif şekilde çözümü Euler yöntemi olarak bilinir. Yüksek mertebede türev içermediğinden kolay bir yöntemdir fakat hassas sonuç elde etmek için adım aralığı küçük seçilmelidir.

$y'(x) = f(y, x)$, $y(x = 0) = y_0$ başlangıç koşullu diferansiyel denklemini ele alalım. Bu diferansiyel denklemi Euler yöntemiyle çözmek demek eşit aralıklı (h) sıralanmış $x_1, x_2, x_3, \cdots, x_n$ değerlerine karşılık fonksiyonun bu noktalarda aldığı $y_1, y_2, y_3, \cdots, y_n$ değerlerini bulmak demektir. (37) nolu denklemi bu şekilde iteratif olarak genelleştirilirse

$$y_{n+1} = y_n + hf(x_n, y_n) + \square(h^2) \quad 38)$$

elde edilir. İkinci dereceden türevler ihmal edildiği için adım aralığı çok geniş seçilirse gerçek çözüme yakın sonuçlar vermez. Şekil 9'da Euler yöntemi için bir program örneği verilmiştir. İlk önce Mathematica'da diferansiyel denklemin analitik çözümü yapılmış ve daha sonra Euler yöntemi ile uygulanmıştır.

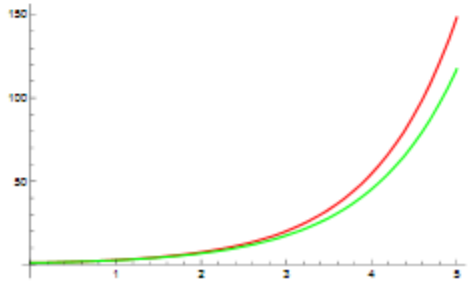
```
Clear[f, t, y, α, a, b]
f[t_, y_] = y;
α = 1;
a = 0;
b = 5;
cozum = DSolve[{y'[t] == f[t, y[t]], y[a] == α}, y[t], t];
y[t_] = y[t] /. cozum[1]
Plot[y[t], {t, a, b}]
e^t
```




```
Clear[m, i, h, t, grafm, n]
h = 0.1;
t[i_] = a + i h;
n = Ceiling[(b - a)/h];
m[0] = α;
Do[m[i + 1] = m[i] + h f[t[i], m[i]], {i, 0, n - 1}]

llist = Table[{t[i], w[i]}, {i, 0, n}];

grafm[t_] = Interpolation[llist][t];
Plot[{y[t], grafm[t]}, {t, a, b}, PlotStyle → {Red, Green}]
```



Şekil 9. Euler yöntemi için Mathematica'daki örnek bir program

3.2. Runge-Kutta Yöntemi

Euler yönteminde x_n noktasından x_{n+1} noktasına (38) nolu denklemi uygulayarak geçilebilir. Bu iki noktanın arasında bir nokta alınarak tekrar x_{n+1} noktası elde edilebilir. Bunun için k_1 ve k_2 gibi

$$k_1 = hf(x_n, y_n) \tag{39a}$$

$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \tag{39b}$$

iki bağıntı tanımlanırsa (38) nolu denklem şu hale gelir:

$$y_{n+1} = y_n + k_2 + \square(h^3). \tag{40}$$

Bu iki adımlı Runge-Kutta formülüdür. Euler yöntemine göre daha iyi ve hassas sonuç vermekle birlikte her adımda $f(x,y)$ fonksiyonu iki kere hesaplanır [13]. Bu çalışmada 4 adımlı Runge-Kutta programı verilecektir:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ k_3 &= hf\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\ k_4 &= hf(x_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \square(h^5). \end{aligned} \qquad 41)$$

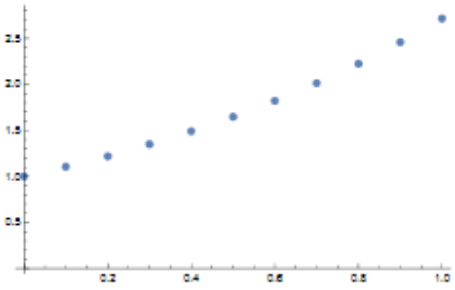
4 adımlı Runge-Kutta yönteminde hata payı $\square(h^5)$ olduğundan çok hassas sonuç verir. Fakat yine her adımda fonksiyonu 4 kez hesaplar. Bu da bazı problemlerde işlem süresini uzatabilir. Şekil 10'da Runge-Kutta yöntemi için bir program verilmiştir.

```
Clear[x, y, f, h, adim]
f[x_, y_] := y;
x[0] = 0;
y[0] = 1;
x[n] = 1;
h = 0.1;
adim = N[(x[n] - x[0]) / h];
x[n_] := x[0] + n * h;
y[n_] := Module[{k1, k2, k3, k4},
  k1 = h (f[x[n - 1], y[n - 1]]);
  k2 = h (f[x[n - 1] + h / 2, y[n - 1] + k1 / 2]);
  k3 = h (f[x[n - 1] + h / 2, y[n - 1] + k2 / 2]);
  k4 = h (f[x[n - 1] + h, y[n - 1] + k3]);
  y[n] = y[n - 1] + 1 / 6 (k1 + 2 k2 + 2 k3 + k4)]
runkut = Table[{x[i], y[i]}, {i, 0, adim}]
TableForm[runkut]
```

```
{{0, 1}, {0.1, 1.10517}, {0.2, 1.2214},
 {0.3, 1.34986}, {0.4, 1.49182}, {0.5, 1.64872}, {0.6, 1.82212},
 {0.7, 2.01375}, {0.8, 2.22554}, {0.9, 2.4596}, {1., 2.71828}}
```

```
0      1
0.1    1.10517
0.2    1.2214
0.3    1.34986
0.4    1.49182
0.5    1.64872
0.6    1.82212
0.7    2.01375
0.8    2.22554
0.9    2.4596
1.     2.71828
```

```
ListPlot[runkut]
```



Şekil 10. Runge-Kutta yöntemi için Mathematica'da örnek bir program

4. TARTIŞMA VE SONUÇ

Bu çalışmada lineer ve lineer olmayan denklem sistemleri ile birlikte diferansiyel denklemlerin sayısal çözümleme programları Mathematica dilinde verilmiştir. Şüphesiz Mathematica sahip olduğu bazı komutlarla (DSolve, NDSolve) diferansiyel denklemleri çözebilmektedir. Buradaki amacımız Mathematica'nın programlamaya uygun bir dil olduğunu göstermekti.

Çalışmada elde edilen sonuçlar Matlab, Python, Fortran gibi başka dillerde de elde edilebilir. Mathematica'nın bu dillere göre avantajı sahip olduğu fonksiyon kütüphanesi ve üstün grafik özellikleridir. Sayısal çözümleme derslerinde Mathematica'nın kullanımı yararlı olacaktır.

REFERANSLAR

[1] Tuncer, T. (1996) *Diferansiyel Denklemler,* Alfa Basım Yayım Dağıtım, İstanbul

[2] Pala, Y. (2006) *Modern Uygulamalı Diferansiyel Denklemler,* Nobel Yayın Dağıtım, Ankara

[3] Aktaş, Z., Öncül, H., Ural, S. (1991) *Sayısal Çözümleme,* ODTÜ Yayınları, Ankara

[4] Tapramaz, R. (2002) *Sayısal Çözümleme,* Literatür Yayınları, İstanbul

[5] Brookshear, J. G. ve Brylow, D. Çev. Edt: Demirci, B.B. (2016) *Bilgisayar Bilimlerine Giriş,* Nobel Akademik Yayıncılık, Ankara

[6] Vatansever, F. (2015), *Algoritma Geliştirme ve Programlamaya Giriş,* Seçkin Yayıncılık, Ankara

[7] Vatansever, F. (2006), *İleri Programlama Uygulamaları,* Seçkin Yayıncılık, Ankara

[8] Çınar, M. (2000) *Mathematica 3.0 ve 4.0 Sürümü*, Seçkin Yayıncılık, Ankara

[9] Çınar, M. ve Çalışkan, F. (1995) *Mathematica ile Programlama,* Beta Basım Yayım Dağıtım, İstanbul

[10] Sınıksıran, E. ve Aktütün, A. (2009) *Matematik ve İstatistik Uygulamalarıyla Mathematica,* Türkmen Kitapevi, İstanbul

[11] Cesur, Y. (2015) *Diferansiyel Denklemler ve Mathematica,* Kişisel Yayın

[12] Akın, Ö. (Çeviri Editörü) (2008), *Diferansiyel Denklemler ve Sınır Değer Problemleri,* Palme Yayıncılık, Ankara

[13] Karaoğlu, B. (2013) *Fortran ve Python ile Sayısal Fizik,* Seçkin Yayınları, Ankara

[14] Wolfram, S. (1991) *Mathematica: A System for Doing Mathematics by Computer,* Addison Wesley Publishing Company