Electronic Letters on Science & Engineering 16(2) (2020)
Available online at dergipark.org.tr/else

Automatic Solution of Some Mathematical Problems Based on a Symbolic Computing Methodology

Muhammed MİLANİ[1*] ,Özlem ORHAN[2] , Bahar MİLANİ[3]

[1]Department of Computer Engineering, Bandirma Onyedi Eylul University, Balikesir- Turkey
Email: mmilani@bandirma.edu.tr ORCID: 0000-0003-2450-0280
[2]Department of Engineering Science, Bandirma Onyedi Eylul University, Balikesir- Turkey
Email: oorhan@bandirma.edu.tr ORCID: 0000-0003-0058-0431
[3]Department of Computer Engineering, Bandirma Onyedi Eylul University, Balikesir- Turkey
Email: bmilani@bandirma.edu.tr ORCID: 0000-0002-5295-4215

Abstract: In recent years, there has been a lot of attention to the use of computers in education, and various software has been developed for it. Meanwhile, problem-based education is considered as one of the efficient methods of teaching science, especially mathematics, because it improves in this type of long-term memory training. On the other hand, e-learning systems are expanding recently. All of which illustrates the need for a system that can produce and solve mathematical problems automatically. In this paper, the issues of Functions, Equation, Polynomial, and Derivative topics have been investigated and an Authentication Method has been developed to solve step-by-step related issues using the Abstract Syntax Tree (AST) based methodology. Using the methods presented in this article, you can create a system for displaying all or a number of problem-solving steps based on an appropriate training model.

Keywords: AST, Computer algebra system, Problem solver, Problem-based learning, Step-by-Step solving.

Sembolik Hesaplama Metodolojisine Dayalı Bazı Matematiksel Problemlerin Otomatik Çözümü

Özet: Son yıllarda, eğitim alanında bilgisayarların kullanılmasına çok önem verilmiş ve bunun için çeşitli yazılımlar geliştirilmiştir. Ayrıca, probleme dayalı eğitim şekli, fen bilimleri özellikle de matematik öğretiminin etkili yöntemlerinden biri olarak kabul edilmiştir çünkü probleme dayalı eğitim uzun süreli hafıza eğitimini geliştirmektedir. Öte yandan, e-öğrenme sistemlerinin sayısı son zamanlarda hızla artmaktadır. Bunların hepsi, matematiksel problemleri otomatik olarak üretebilen ve çözebilen bir sisteme olan ihtiyacı ortaya çıkarmaktadır. Bu makalede, en önemli matematik konularından fonksiyonlar, denklem, polinom ve türev konuları incelenmiş ve Soyut Söz dizimi Ağacı (AST) tabanlı bir yöntem kullanılarak ilgili soruları adım adım çözmek için bir Doğrulama Yöntemi geliştirilmiştir. Sadeleştirme, çarpanlara ayırma, dağıtım ve yerine koyma gibi cebirsel birçok işlem, çözüm yöntemleri içerisinde yer alır. Bu makalede sunulan yöntemler kullanılarak ve uygun bir eğitim modeli seçilerek, problem çözme adımlarının tümünü veya bir kısmını görüntüleyebilen bir sistem oluşturabilir.

Anahtar Kelimeler: AST, Adım Adım çözme, Bilgisayar cebir sistemleri, Probleme dayalı öğrenme, Problem çözme.

Reference to this paper should be made as follows:
Milani M. ,Orhan Ö. ,Milani B. ,'Automatic Solution of Some Mathematical Problems Based on a Symbolic Computing Methodology', Elec Lett Sci Eng , vol. 16(2), (2020), 130-142.

1. Giriş

Bilim ve teknoloji alanındaki yenilikler bilgi teknolojisinin gelişmesiyle birlikte tüm alanlarda etkisini göstermektedir. Bilimin ve teknolojinin gelişmesinden en çok etkilenen alanlardan biri eğitimdir. Öğretim yöntemleri üzerine yapılan araştırmalara göre, probleme dayalı öğrenmenin geleneksel öğretim yöntemlerinden çok daha etkili olduğu ve öğrencilerin problem çözme yeteneğini artırabileceği gözlemlenmiştir [1].

Öğrenciler problem çözme yeteneklerini, çeşitli ve hedeflenen sorunları çözerek geliştirebilirler. Ders kitapları problem çözme yeteneğini geliştirebilmek için uygun bir kaynak olamaz, çünkü ders kitapları sınırlıdır ve etkileşimli değildir. Ayrıca, kitaplar problemleri adım adım çözemez ve öğrencilere doğru seçenekleri sunamaz. Aslında, bu özellikleri sağlayacak çevrimiçi kaynak genellikle yoktur. Bu nedenle, eğitim için bilgi teknolojisi araçlarının kullanımı daha fazla dikkat çekmektedir. Bu arada fizik [2], matematik [3, 4] veya elektronik [5] gibi özel bilimler için problemleri oluşturup sonrasında çözen sistemler [6, 7, 8] vardır.

Matematiksel sistemlerde bilgi teknolojisinden yararlanılmasına, matematiksel ifadelerin geliştirilmesi ve değerlendirilmesine katkıda bulunulmasına yarayan sistemlere Bilgisayar Cebiri Sistemleri veya CAS denir. CAS sistemleri kullanımının artması, matematik öğretiminde bilgisayar sistemlerinin kullanımını yaygınlaştırmıştır. Matematik eğitimi için kullanılmak üzere birçok sistem geliştirilmiştir, bunlardan bazılarına ticari CAS sistemleri denir [9, 10]. Bazı sistemler de akademiktir ve bu akademik sistemler yapısal olarak tasarlanmış sistemler şeklinde adlandırılabilir [11, 12]. Genellikle, bu tür bir sistem ağaç yapısına göre tasarlanmıştır.

Son zamanlarda, matematik problemlerini üretebilen [13] ve çözebilen yöntemleri [14] geliştirebilmek için çalışmalar yapılmıştır. Bu makalede, [14]'te sunulan yönteme dayalı matematiksel problemlerden birkaçını çözmek için yeni yöntemler sunulacaktır. Bir sorunun adım adım çözülmesi, öğrencilere problem çözme eğitiminde yardımcı olur. Öğrenciler problem çözme adımlarını inceleyerek matematik problemleri çözme yöntemlerini sistematik olarak öğrenebilirler.

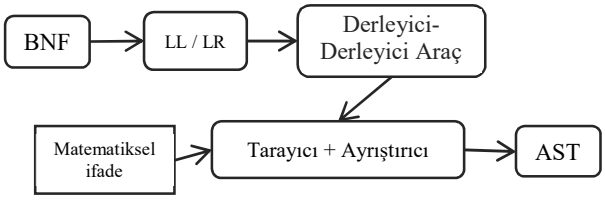
Bu makalede bazı matematiksel problemlerin çözümlerinin bulunması için kullanılan yöntem, [14] de ele alınan yönteme dayanmaktadır. Bu yöntemi uygulamak için fonksiyonlar, denklemler, polinomlar ve türev uygulamaları konuları ele alınmıştır. Çalışmada incelenecek yöntemlerden bir diğeri, bir fonksiyonun değerinin verilen değişkenler cinsinden nasıl hesaplanacağıdır. Makalede birinci ve ikinci dereceden denklemleri çözmek için farklı yöntemler sunulacaktır. Ayrıca, polinom bölmesi, en Büyük ortak bölenler (EBOB) ve En Küçük Ortak Katsayı (EKOK) bulunması ve çarpanlara ayırma gibi konular incelenecektir.

Bu çalışmada sistemi tasarlamak için java programlama dili kullanılmıştır. Çünkü bu dil işletim sistemlerinden bağımsızdır ve tüm cihazlarda ve hatta internette kullanılabilir. Matematiksel ifadeyi ayrıştırmak için JavaCC kullanılmıştır. Önerilen sistemler, kişiselleştirilmiş eğitim sisteminin bir parçası olarak kullanılabilir veya e-öğrenme sistemlerinde uygulama ve kendi kendini inceleme için bir araç olarak kullanılabilir. Sonraki bölümlerde yöntem detaylı incelenecek ve daha sonra yönteme dayalı olarak ele alınan konuların uygulama yöntemleri çalışılacaktır.

2. Yöntemin Açıklanması

Matematiksel ifadeleri adım adım çözmek için [14]'te verilen makalede gramer tabanlı bir yöntem tanımlanmıştır. Sistem, giriş verilerini çözümlemek için genişletilmiş bir dilbilgisi kullanır ve daha sonra belirli bir matematiksel ifadeyi kolayca işlemek için uygun bir model sağlayan bir ağaç veri yapısına dönüştürür. Cebirsel problemlerin sembolik çözümünü elde etmek için kullanılan sistem, büyük ifadelerin küçük parçalara ayrıldığı ve her parçanın orijinal problemin tam bir çözümünü elde etmek için ayrı ayrı ele alındığı bir bölme ve elde etme stratejisini kullanır. Problem parçalarının çözümü, bölme ve elde etme stratejisini kullanarak ara veya adım adım çözümlerin üretilmesine yardımcı olur, böylece basitleştirme ve yazdırma aşamaları yönetilebilir, örneğin ifadelerin türetilmesi için gerekli olan nesneye-yönelik programlama yapıları, sınıflar arasındaki kalıtımın desteklenmesi sonucunda bu stratejinin uygulanmasını kolaylaştırır.

Yönteme uygun olarak, matematiksel ifade Soyut Sözdizimi Ağacında (AST) Şekil 1'de gösterilen adımları izleyerek sunulmaktadır.



Şekil 1. Matematiksel dilbilgisine AST üretmek için gerekli adımlar

Daha sonra, belirli modeller kullanılarak uygun işlem, üretilen ağaç üzerinde gerçekleştirilir. Matematiksel ifadeleri yeniden görüntülemek için, AST'deki matematiksel ifadeleri insan tarafından okunabilir formata veya *LaTex* formatına dönüştürmenin bazı yolları bulunmaktadır. Makalenin amacına ulaşmak için gerekli olan bazı ifadeleri sadeleştirmek gerekir, bunun için kullanılan bazı yöntemler vardır.

3. Yöntemin Uygulanması

Bu bölümde, matematiğin aşağıda verilen önemli konuları ele alınmıştır ve ifadeler üzerinde doğru işlemleri yapabilmek için yönteme dayalı uygulamalar önerilmiştir.

3.1. Fonksiyon

Fonksiyonlar dilbilgimizde belirli bir yapıya sahip olan matematiksel ifadelerdir. Tanım kümesi X ve değer kümesi Y olan f fonksiyonu aşağıdaki şekilde gösterilir:

$$f: X \to \mathrm{Y} \qquad \text{ya da} \qquad \mathrm{X} \xrightarrow{f} \mathrm{Y}$$

X'in elemanlarına *f* 'in argümanları denir. Tanım kümesindeki her $x \in \mathrm{X}$ argümanına değer kümesinde karşılık gelen $y \in \mathrm{Y}$ elemanına, *f* altında *x*'in görüntüsü denir. *f*, *x*'in *y* ile eşleştiğini veya *y* ile *x*'in ilişkilendirildiğini ifade eder. " $f(x) = \mathrm{Y}$" eşitliğine basit form denir. Çeşitli fonksiyon formları Toplama, Çıkarma, Çarpma ve Bölme işlemleri için sırası ile $(f + g)(x) = f(x) + g(x)$, $(f - g)(x) = f(x) - g(x)$, $(f * g)(x) = f(x) * g(x)$, $(\frac{f}{g})(x) = \frac{f(x)}{g(x)}$ ki $\mathrm{g(x)} \neq 0$ şeklinde tanımlayabiliriz.

Ayrıca, iki fonksiyonun bileşkesi olan fonksiyonları elde etmek için başka bir yol daha vardır. $f(x)$ ve $g(x)$ 'in bileşkesi, $(f \circ g)(x) = f(g(x))$ olarak sembolize edilir, burada "*f* bileşke g'nin *x*'i" olarak telaffuz edilir. Kavram basittir, *g*'nin *x*'de aldığı değer *f* için bağımsız değişken olarak kullanılacaktır.

İki fonksiyon için bu beş işlem, AST'de fonksiyonların iki biçimi ve beş işlenemden biri kullanılarak oluşturulmuş bir ifade varsa kullanılabilir. Örneğin, $f(x) = \frac{(\cos(x)+x^2)}{2x}$ girdi dizesini ayrıştırma, Şekil 2'de gösterilen bir AST ağacı şeklinde yazılabilir.

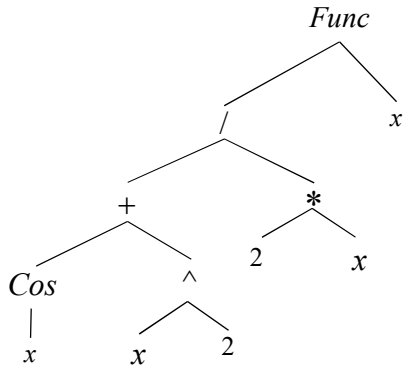
Ana dilbilgimizdeki fonksiyonları destekleyen dilbilgisi Liste 1'de gösterilmiştir.

Liste 1. Fonksiyonların değerleri için dilbilgisi

$$< start > \longrightarrow f(<expr>) = <expr> [, [f \mid g](<expr>) = (\text{“?”} \mid <expr>)]?$$

Bu dilbilgisinin örnek kullanımları aşağıdaki bölümlerde verilmiştir.

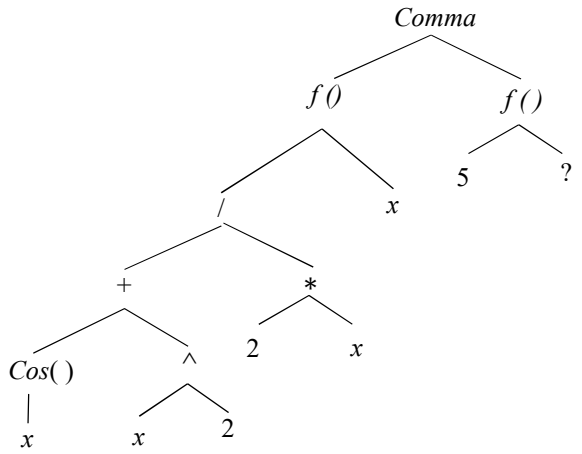
```
Func (
    Divide (
        Sum (Cos(Var()), Power(Var(),Num(2))),
        Times(Num(2), Var())
    ),
    Var()
)
```



Şekil 2. $f(x) = \frac{(\cos(x)+x^2)}{2x}$ ifadesi için Ayrıştırma ağacı ve Nesne görünümü

3.1.1. Fonksiyon değerinin hesaplanması

Liste 1'deki dilbilgisi "$f(x)$ = Y" ve "$f(x)$ = Y, $f(Num)$ =?" şeklindeki matematiksel ifadeleri kabul eder. Örneğin, "$f(x) = \frac{(\cos(x)+x^2)}{2x}, f(5)$ =?" ayrıştırıcısı, Şekil 3'te gösterilen şekilde bir AST üretir.



Şekil 3. "$f(x) = \frac{(\cos(x)+x^2)}{2x}, f(5)$ =?" ifadesi için AST ağac

Fonksiyonun bu formdaki tüm ifadeleri aynı AST'ye sahiptir. myExp nesnelerinin her düğümü, çocuklarını değerlendirerek değerini hesaplayabilecek bir *calc* () yöntemine sahiptir. Aşağıdaki modelde, iki *Func* düğümü olan bir *Comma* düğümümüz vardır. Sol *Func* nesnesini hesaplamak

için Liste 2'de gösterildiği gibi sağ *Func* nesnesinin sol değerini kullanarak fonksiyon değerini hesaplamak mümkündür.

Liste 2. Bir fonksiyon değerinin hesaplanması

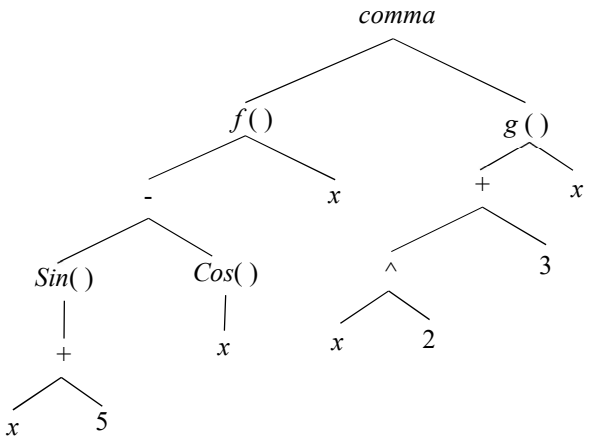
```
public class Comma extends myExp{
  ...
  public myExp calc(){
    double val = exp2.exp1.getNum();
    return exp1.calc(val);
  }
  ...
}
```

3.1.2.Fonksiyonlarda İşlemler

Önceki bölümde yapılan uygulamalara benzer şekilde, iki fonksiyon üzerinde ekstra işlemler uygulamak ve değerlendirmek mümkündür. Tablo 1'de belirtildiği gibi temel fonksiyonlar, fonksiyonsuz basit ifadelerin değerlendirilmesine benzerdir. Aşağıdaki örnek dikkate alındığında:

$$f(x) = \sin(x+5) - cos\,x,\ g(x) = x^2 + 3 \qquad (1)$$

(1) denkleminin AST ağacı Şekil 4'teki gibidir:



Şekil 4. $f(x) = \sin(x+5) - cos\,x,\ g(x) = x^2 + 3$ ifadesi için AST

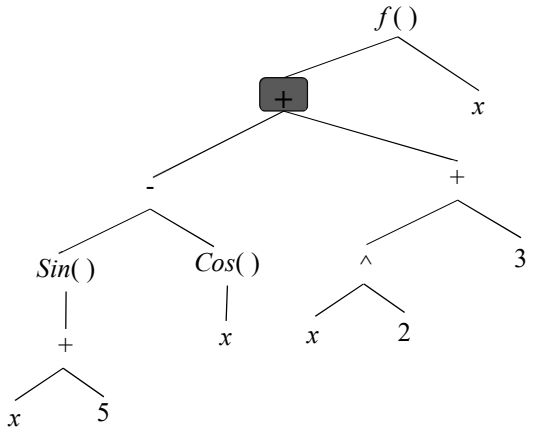
Şekil 4'teki AST'ye göre, tüm fonksiyon işlemlerini düğümlerine uygulamak mümkündür. Liste 3'te tüm işlemleri gerçekleştirmek için gerekli örnek kodlar verilmektedir.

Liste 3. Bir fonksiyonun değerinin hesaplanması

```
public class Comma extends myExp{
  …
  public myExp getAdd(){
    return new Func(new Sum(exp1, exp2), Var.X());
  }
  public myExp getMinus(){
    return new Func(new Minus(exp1, exp2), Var.X());
  }
  public myExp getMul(){
    return new Func(new Mul(exp1, exp2), Var.X());
  }
```

```
  public myExp getDiv(){
    return new Func(new Div(exp1, exp2), Var.X());
  }
  public myExp getCompose(){
    return new Func(exp1.eval(exp2), Var.X());
  }
  …
}
```

Liste 3'te yer alan yeniden düzenlenmiş *Comma* sınıfı, Şekil 4'teki iki fonksiyonun toplamı için Şekil 5'i üretir.



Şekil 5. İki fonksiyonun toplamı

Çıkarma, çarpma ve bölme işlemleri, toplama işlemi ile benzer şekilde yapılır. Bununla birlikte, iki fonksiyonun bileşkesini veren AST, ikinci fonksiyonun değer kümesinin kullanıldığı mevcut bölümde ele alınan fonksiyonun değerinin hesaplanmasına benzer şekilde bulunur.

3.2. Denklemler

Bir fonksiyon ifadesi, bir fonksiyon ve bazı "=" işaretleri ile başlar. Bununla birlikte, iki cebirsel ifade eşit ise "*x*-5=3" ifadesi gibi denklemler için AST, hesaplanabilen fonksiyonel olmayan bir denklem olarak işaretlenmelidir. "=" işaretinin sol tarafındaki ifadeye denklemin sol tarafı, sağ tarafındaki ifadeye denklemin sağ tarafı denir. Bir denklemin sonucu "3+2=5" gibi doğru bir cümle, "9-5=1" gibi bir yanlış cümle veya *x* 'in değeri veya kökünü bulmak için çözülmesi gereken "*x*+3=9" gibi açık bir cümle olabilir.
Bu çalışmada, birinci derece denklemler ve ikinci dereceden denklemler ele alınmıştır. Birinci derece denklemler ile ikinci derece denklemler arasındaki temel fark *x* değişkeninin üstel kuvvetidir. Birinci dereceden denklemlerde *x* değişkeninin üstel kuvveti 1'dir, kuadratik denklemlerde *x* değişkeninin üstel kuvveti 2 veya daha az olabilir fakat denklemde x^2 ifadesi olmalıdır.

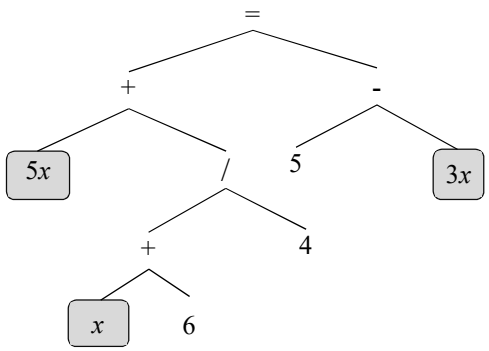
3.2.1. Birinci Dereceden Denklemler

Birinci dereceden denklemleri ayrıştırmak için aşağıda verilen Liste 4'te ek LL (1) dilbilgisi kuralları kullanılmıştır.

Liste 4. Birinci dereceden denklemler için dilbilgisi kuralları

```
<start>  ⟶ <expr> " = " <expr>
<expr>   ⟶ <term> { ("+" | "-") <term> }*
<term>   ⟶ <num>  ("x")? | "x" (<num>)?
```

Bu dilbilgisi, denklemleri "*ifade* = *ifade*" şeklinde kabul eder. Birinci dereceden denklemlerin normal formu, *a* ve *b* sabitler ve *x* değişkeninin kuvveti 1 olmak üzere "$ax + b = 0$" şeklindedir. Giriş denklemi normal forma sadeleştirilmelidir. Şekil 6'da "$5x + (x+6) / 4=5- 3x$" ifadesi için bir örnek verilmiştir. $3x$'in $3*x$'ten farklı olduğuna ve bileşke terim olarak kullanıldığına dikkat ediniz.



Şekil 6. "$5x + (x + 6) / 4=5 - 3x$" ifadesi için AST

Sadeleştirme ve birinci dereceden denklemlerin çözümü "=" tanımlanmasını gerektirir. Ayrıca *Equ* sınıfı, denklemin sol ve sağ taraflarını takip etmek için kullanılır. Şekil 6'daki AST kökü bir *Equ* nesnesidir. Liste 5, *java* programlama dilinde *Equ* sınıfı tanımını göstermektedir.

Liste 5. Java'da Equ Sınıfı Tanımı

```
public class Equ extends myExp{
  public Equ Simplify() {
    Data d1 = exp1.Simplify().eval();
    Data d2 = exp2.Simplify().eval();
    double a = d1.a() + d2.a();
    double b = d1.b() + d2.b();
    Return new Equ(Var.X(), new Num(b / a));
  }
  public String Display() {
    String str1 = exp1.Display();
    String str2 = exp2. Display();
    return str1 + "x= " + str2;
  }
}
```

Data sınıfı, "$ax + b = 0$" denkleminde, *a* ve *b* parametrelerini elde etmek için kullanılan basit bir sınıftır.

3.2.2.İkinci Dereceden Denklemler

"$ax^2 + bx + c = 0$" şeklinde ifadeler ikinci dereceden denklemler olarak tanımlanır. Burada

a, *b* ve c sabit sayılar ve $a \neq 0$. *a*, *b* ve c değerleri, birinci derece denklemdeki prosedüre benzer şekilde Data sınıfı tarafından elde edilir. Liste 6'da bu tür bir ifade için ek dilbilgisi kuralları gösterilmektedir.

Liste 6. İkinci dereceden denklemler için dilbilgisi kuralları

```
<start>   ⟶  <expr> "= "<expr>
<expr>    ⟶  <term> {("+" | "- ") <term>} *
<term>    ⟶  {<term1>|<<term2>} {"*" <term3>}*
<term>    ⟶  <term2> "*" <term2>
<term1>   ⟶  <element> ("^2")?
<term2>   ⟶  <element>
<term3>   ⟶  <num>
<element> ⟶  <num> | <"x">
```

İkinci dereceden denklemleri çözmek için çarpanlara ayırma, tam kareye tamamlama ve diskriminant gibi çeşitli yöntemler kullanılır. Gerçek veya kompleks katsayılardan oluşan ikinci dereceden bir denklemin kök adı verilen iki çözümü vardır. Bu iki çözüm, farklı veya reel sayı olabilir veya olmayabilir. Bu makalede, diskriminant yöntemi kullanılmıştır.
Diskriminant adı verilen ve **Δ** sembolü ile gösterilen ve Delta şeklinde okunan fonksiyon aşağıdaki şekilde hesaplanır:

$$\Delta = b^2 - 4ac \tag{1}$$

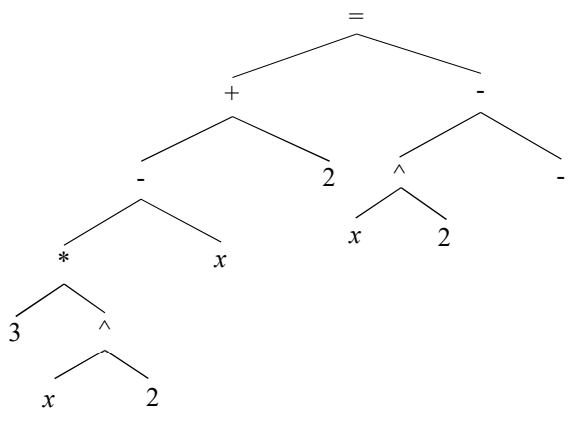
Diskriminantın değerine göre kök sayısı ve bu köklerin cinsi (reel ya da kompleks sayı) aşağıdaki şekillerde belirlenir:

$$\begin{cases} x_1, x_2 & 0 < \Delta \\ x_1 = x_2 & 0 = \Delta \\ kök\ yoktur & 0 > \Delta \end{cases} \tag{2}$$

x_1 ve x_2 kökleri aşağıdaki şekilde hesaplanır:

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}, \qquad x_2 = \frac{-b - \sqrt{\Delta}}{2a} \tag{3}$$

AST'de bir x^2 terimi olmalıdır. *a*, *b* ve *c* değerlerini hesaplayabilmek için bu terim sadeleştirilmelidir. Şekil 7'de bu kuralı örnekleyen "$3x^2$-x+2=x-12" ifadesinin AST'si görülmektedir.



Şekil 7. $3x^2$-x+2=x-12 ifadesi için AST

Dönüşümden sonra *a*, *b* ve *c* değerlerinin elde edilmesi mümkündür. Denklemin kökleri, (4). denklemde tanımlanan formüllerle hesaplanabilir.

3.3. Polinomlar

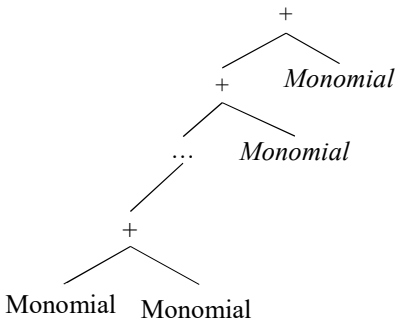
Bir polinom, değişken, belirsiz terim ve katsayılar içeren bir ifadedir. Polinom ifadesi, kuvvet operatörlerinin üsleri için toplama, çıkarma, çarpma operatörlerini ve negatif olmayan bir tamsayı kullanır.

$$\sum_{i=0}^{n} c_i x^i = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_2 x^2 + c_1 x + c_0,\ 2 \leq n \qquad (5)$$

i ve *n* negatif olmayan tamsayılar olmak üzere, c_n, …,c_0 sabit sayılardır ve *x* bir değişkendir. Bir polinom, monomialların (tek terimli sayılar) toplamıdır. "$3x^7$" gibi bir monomial bir terimli sayıların ve değişkenlerin sayısı veya ürünü olabilir. Monomialların sayı kısmına katsayı denir.

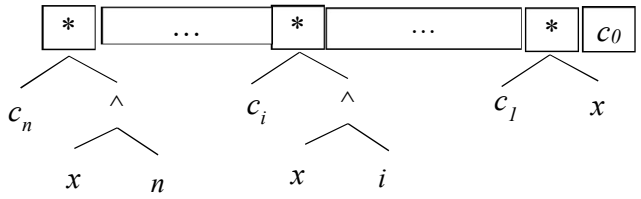
3.3.1. Polinomların Standartlaştırılması

Polinom giriş dizesi, standart bir monomialin olması gereken standart formuna dönüştürülmelidir. Ek olarak, monomialler, *x* değişkeninin veya terimin derecesinin kuvvetine göre düzenlenmelidir. Şekil 8, bir polinom ifadesi için ayrıştırıcının AST çıktısını göstermektedir. Her monomial düğüm, polinom terimleri için bir Prod nesnesinin alt ağacıdır.



Şekil 8. Bir polinom ifadesinin AST'si

Bir polinomun standartlaştırılması, her monomialin standartlaştırılması ile başlar. Bu nedenle, bir polinomun her terimi sadeleştirilmeli ve "$c_i x_i$" formuna dönüştürülmelidir; burada myExp nesnesi olarak c_i, AST'de x_i'nin ürünü olacaktır. Data sınıfı her bir monomiali sadeleştirecektir. Bir sonraki adım, her monomialin derecesine göre sıralanmasıdır. Bu adım, tüm monomialleri bir dizi listesine koyarak ve sıralayarak gerçekleştirilir. Aşağıda verilen Şekil 9'da, bu kavram gösterilmektedir.



Şekil 9. Bir polinom için monomialların dizisi

Bir dizideki bir polinomunun AST'si *Sum* sınıfından elde edilir. Şekil 8'e göre AST düğümleri *Sum* nesneleridir. Bu nedenle, [14]'te verilen *sameOpList* () yöntemi ele alınacak ve monomiallerin üstel değerini elde edebilmek için Liste 7'deki kodlar kullanılacaktır.

Liste 7. Monomiallerin üstel değerini elde etmek için kodlar

```
public static int getMonomialDegree(myExp e){
  int result=0;
  if(exp2 instanceof Power)
    result = ((Power) exp2).exp2.getNumValue();
  else if ( exp2 instanceof Var)
    result = 1;
  return result;
}
```

Şekil 9'daki diziyi sıralamak ve diziyi *Sum* düğümlerinde AST'ye dönüştürmek, polinom ifadelerini standart hale getirecektir.

3.3.2. Polinom Bölmesi

Polinom bölümü, bir polinomun aynı veya daha düşük derecedeki başka bir polinom ile bölünmesidir. Aşağıdaki gibi iki monomial olduğunu varsayalım:

$$c_i x^i \quad , \quad c_j x^j \tag{6}$$

İki monomialin bölünmesi Liste 8'de ifade edilmiştir.

Liste 8. İki monomialin oluşturulması

```
function getMonomialDivision(c_i x^i, c_j x^j)
  while(j≠0 and i≥j){
    b = c_i / c_j
    m = i − j
  }
return b x^m
```

Bu yazıda kullandığımız Öklid Bölmesinde kullanılan polinom bölmesi şunları sağlamaktadır:

$$P_1 = P_2 Q + R, \qquad \deg(P_1) \leq \deg(P_2)$$

Burada *Q*-bölüm ve R kalandır. *Q* ve *R* aynı zamanda polinom olabilirler. Yöntem, polinom argümanındaki en yüksek dereceyi dönüştürür. $P_1 \neq 0$ ve $P_2 \neq 0$ için iki standart polinom vardır, Liste 9'daki algoritma sırasıyla *Q* ve *R*'yi dönüştürecektir.

Liste 9. İki polinomun geliştirilmesi

$q_0 = 0$
$r_0 = P_1$
$i = 1$
Repeat {
$q_i = q_{i-1} + \frac{LC(r_{i-1})}{LC(p_2)} x^{\deg(r_{i-1}) - \deg(P_2)}$
$r_i = r_{i-1} - \frac{LC(r_{i-1})}{LC(p_2)} x^{\deg(r_{i-1}) - \deg(P_2)} . P_2$
} Until $\deg(r_i) < \deg(P_2)$
Return (q_i and r_i)

LC(), verilen polinomdaki en yüksek monomial katsayısını dönüştürecektir. Her r_i, monomiyallerin sıralı bir dizi listesi olduğundan, *LC* değerini bulmak kolay olacaktır. Polinom katsayılarının sabit sayılar olduğunu belirtmek gerekir, ancak bu teoremi katsayıların başka bir değişken olduğu durumda da genelleştirmek mümkündür. Tablo 2, polinom bölünmesinin bir örneğini göstermektedir.

Tablo 2. "$(x\ (\ 2x+1) + 3x^3) / (x+1)$" ifadesinin polinom bölümü

P_1	P_2	Q	R
$((x^2-1)\ (2x+3)+\ 3x^3-x)$	$(x+1)$		
$5x^3+3x^2-3x-3$	$x+1$	$5x^2$	$-2x^2-3x-3$
$-2x^2-3x-3$	$x+1$	$5x^2-2x$	$-x-3$
$-x-3$	$x+1$	$5x^2-2x-1$	-2
-2	$x+1$	$5x^2-2x-1$	-2
Sonuç:		$5x^2-2x-1$	-2

3.3.3. Polinomların En Büyük Ortak Bölenleri

İki polinomun EBOB'u (En Büyük Ortak Bölen), her iki polinomun ortak çarpanı olan en yüksek derecedeki polinomdur. Bu polinomların EBOB'unun tamsayıların EBOB'u ile aynı olduğunu gösterir. Tamsayılar için tüm EBOB kuralları polinomlar için de geçerlidir.

3.3.4. Polinomların En Küçük Ortak Katları

İki polinomun EKOK ()'u iki tamsayının EKOK'una benzer şekilde bulunur.

3.3.5. Polinomu Çarpanlara Ayırma

Sadeleştirme, EBOB ve EKOK vb. gibi diğer işlemlerde kullandığımız işlemlerden biri polinomu çarpanlara ayırmadır. Bir polinomu çarpanlarına ayırmak için çeşitli yöntemler vardır, bu yazıda En Büyük Ortak Çarpan yöntemi incelenecektir.

3.3.6. En Büyük Ortak Çarpan Yöntemi

Bu yöntem tüm monomiallerin EBOB'una dayanır. Fikir, sayıların EBOB'una benzer. En Büyük Ortak Çarpan algoritması Liste 10'da verilmiştir.

Liste 10. Polinomların GCF'si

```
M = poly.monomials;
sort(M)
gcf = M1
for i = 2 to M.length {
   gcf = GCD(gcf, Mi)
}
return product(gcf, div(poly, gcf).Simplify())
```

Liste 10'da, bir polinom ifadesinden tüm monomialleri getirdikten sonra, sıralı bir öğe listesine dönüştürülecektir. GCF (*En büyük ortak çarpan*) için başlangıç değeri listenin ilk öğesidir.

Ardından, bu değer listedeki diğer öğeleri toplamak için kullanılacaktır. Nihai değer bir GCF ürünü ve polinomun GCF üzerinden bölünmesi sonucu olacaktır. Tablo 3, bir GCF örneğidir:

Tablo 3. " $35ab^2 - 75a^2 b + 15ab$" ifadesi için GCF

Polinom	*GCF*
$35a\,b^2 - 75a^2 b + 15ab$	$35a\,b^2$
$35a\,b^2 - 75a^2 b + 15ab$	$5ab$
$35a\,b^2 - 75a^2 b + 15ab$	$5ab$
$35a\,b^2 - 75a^2 b + 15ab$	$5ab$
Dönüşüm: $5ab\,(7b - 15a + 3)$	

4. Sonuç

Bu yazıda [14]'te verilen matematiksel ifadeleri adım adım çözme yöntemi ele alınmıştır. Önerilen yöntem ile matematiksel ifadeler üzerinde özel işlemler yapılabilir ve bu ifadeler sistematik ve otomatik olarak çözülebilir. Ayrıca, önerilen yöntem, bir giriş dizesini operatörün önceliğinin uygulandığı, görünür düğümler arasındaki ilişkilerin, farklı gösterimlerde çeşitli çıktıların üretilebileceği ve değerlendirme, sadeleştirme, optimizasyon gibi operatörlerin uygulandığı yapılandırılmış bir ağaç olarak modellediği Özet Sözdizimi Ağacı (AST) tanıtılmıştır. LISP ve Maxima gibi iyi bilinen uygulamalar, ifadelerin işlenmesi için benzer bir ağaç yapısı kullanır.
Bu yazıda, bazı matematik konularının öğretiminde ve problemlerin otomatik çözülmesinde kullanılabilecek sistemler geliştirilmiştir. Bir problemin adım adım çözülmesiyle, öğrenciler amaçlanan matematik konularını kolayca öğrenebilirler.
AST değerlendirmesi oldukça basittir. Ağaç değişkenler ve sayılar içerir, bu nedenle x, a ve b gibi değişkenler kullanılarak verilen değerler için sayısal sonuçlar hesaplanabilir. Yöntemin özelliklerine göre, girdi ifadesinin tipine bağlı olarak çeşitli operatörler kullanılabilir. Bu yüzden, ifade tiplerini tespit etmek için bir sistem uygulanmıştır. Bu, önerilen sistemi mümkün olan fonksiyonlar üzerindeki operatörler gibi daha karmaşık ve güçlü operatörleri kullanması için güçlendirir.
Önerilen yöntemin gücünü ve kullanılabilirliğini göstermek için bazı uygulamalar incelenmiştir. Fonksiyonlar, birinci derece ve ikinci dereceden denklemlerin çözümleri, bölme ve çarpanlara ayırma polinomların ve ifadelerin türetilmesi üzerine çeşitli işlemler ele alınmıştır. Bu çalışma, bu yöntemin problem çözmeye dayalı eğitim sistemlerinde nasıl kullanıldığını açıklamıştır.

Kaynakça

[1] Kwan, C. Y. (2000). What is problem-based learning (PBL): It is magic, myth and mindset. Centre for development of Teaching and Learning, 3(3), 1-6.
[2] Thacker, B. A. (2003). Recent advances in classroom physics. Reports on progress in physics, 66(10), 1833.
[3] Kynigos, C., & Moustaki, F. (2014, June). Designing digital media for creative mathematical learning. In Proceedings of the 2014 conference on Interaction design and children (pp. 309-312).
[4] Jones, K., Geraniou, E., & Tiropanis, T. (2013). Patterns of collaboration: towards learning mathematics in the era of the semantic web. In Visual mathematics and cyberlearning (pp. 1-21). Springer, Dordrecht.
[5] Al Rekhawi, H. A., & Abu Naser, S. S. (2018). An Intelligent Tutoring System for Learning Android Applications Ui Development. International Journal of Engineering and Information Systems (IJEAIS), 2(1), 1-14.
[6] Escobar, I., Cebrian, B., Arribas, E., Franco, T., Suarez, C., Vidales, S., ... & Belendez, A. (2016). Learning Physics with Wolfram Alpha. In Proceedings of INTED 2016 Conference (pp. 5598-5602).

[7] Říhová, V., Jílková, E., & Wossala, J. WOLFRAM ALPHA IN MATHEMATICS AND ECONOMICS. INTERNATIONAL DAYS OF SCIENCE 2020, 156.
[8] Grasso, F., Luchetta, A., Manetti, S., Piccirilli, M. C., & Reatti, A. (2016). SapWin 4.0–a new simulation program for electrical engineering education using symbolic analysis. Computer Applications in Engineering Education, 24(1), 44-57.
[9] Char, B. W., Geddes, K. O., Gentleman, W. M., & Gonnet, G. H. (1983, March). The design of Maple: A compact, portable, and powerful computer algebra system. In European Conference on Computer Algebra (pp. 101-115). Springer, Berlin, Heidelberg.
[10] Noro, M., & Takeshima, T. (1992, August). Risa/Asir—a computer algebra system. In Papers from the international symposium on Symbolic and algebraic computation (pp. 387-396).
[11] Fioravera, M., Marchisio, M., Di Caro, L., & Rabellino, S. (2020). Learning Through a "Route Planner": Human-Computer Information Retrieval for Automatic Assessment. In Technology Supported Innovations in School Education (pp. 115-141). Springer, Cham.
[12] Havola, L. (2012). Assessment and learning styles in engineering mathematics education. Licentiate thesis. Aalto University.
[13] Xiong, Y., Ramachandran, G. K., Ganesan, R., Jajodia, S., & Subrahmanian, V. S. (2020). Generating Realistic Fake Equations in Order to Reduce Intellectual Property Theft. IEEE Transactions on Dependable and Secure Computing.
[14] Hosseinpour, S.; Alavi Milani, M.M.R.; Pehlivan, H., 2018, "A Step-by-Step Solution Methodology for Mathematical Expressions. " Symmetry , 10, 285.