Pİ SAYISININ MONTE-CARLO METHODU VE GREGORY/LEIBNIZ FORMÜLÜYLE HESAPLANMASI

Ahmet Turan Gültekin[1]

Prof. Dr. Musa Hakan Asyalı[2]

ÖZET: Bu çalışmada, pi *(TT)* sayısını Monte-Carlo metoduyla ve Gregory-Leibniz formülüyle hesaplama yolları araştırılmış ve karşılaştırılmıştır. Monte-Carlo metodu karesel bir alan içinde kalacak şekilde tekdüze dağılımdan üretilen gelişigüzel noktalardan dairesel bir bölgeye düşen noktaları saymaya dayanır. Gregory-Leibniz formülü ise arctanjant fonksiyonun Taylor serisi açılımını yaparak ***n*** sayısını bulur. Bu iki metodu da C programlama dilinde gerçekledik ve sonuçlarını hassasiyet ve hesaplama hızı açısından karşılaştırdık. Elde ettiğimiz sonuçlara göre, her ne kadar Monte-Carlo metodu anlaması/anlatması daha kolay bir metot ise de, Gregory-Leibniz formülü, ***7t*** sayısını hesaplamak için daha hassas ve hızlı sonuçlar üretir. ***Anahtar Kelimeler:*** Pi sayısı, Monte-Carlo metodu, Gregory-Leibniz formülü, Taylor serisi.

ABSTRACT: In this study, we discuss and compare two methods of calculating the number pi *(Tİ)*. The Monte-Carlo method is based on generating random numbers from uniform density within a square region and counting the number of points that fall inside a circular area. The second method known Gregory-Leibniz Formula is based on the Taylor expansion of arctangent functions. We implemented both methods in C programming language and compared them in terms of accuracy and computational complexity. Our results indicate that although Monte-Carlo method is of high instructional value, the Gregory-Leibniz method offers a more accurate and fast method to calculate the number **;r**. ***Keywords:*** Estimating pi, Monte-Carlo method, Gregory-Leibniz formula, Taylor series.

[1] Bilgisayar Mühendisliği Bölümü, Yaşar Üniversitesi, ahmetturan@gultekin.info

[2] Bilgisayar Mühendisliği Bölümü, Yaşar Üniversitesi musa.asyali@yasar.edu.tr

GİRİŞ

Pi **O**) sayısını okul yıllarından anımsamayan yok gibidir. Bir dairenin çevresinin çapına olan oranı bize ***n*** sayısını verir. Pi'nin yüzyıllar boyunca gizemli bir sayı olduğunu düşünenler, ***n*** hayranları, ona kutsallık yakıştıranlar bile olmuştu. Pi kutsal bir sayı değilse de, üzerine yüzlerce sayfalık kitaplar yazılan oldukça ilginç bir sayıdır.

"Bu ***7t*** sayısı öyle garip özelliklere sahiptir ki, hiç ummadığınız bir anda mantar gibi karşınızda bitiverir… Örneğin, daire 360 derecedir ve bu gerçek garip bir biçimde ***n*** sayısıyla bağlantılıdır. Şimdi ;r'nin 360. basamağına bir göz atalım. Basamakları saymaya en baştaki 3 sayısından başlarsak, ***n*** sayısının 359, 360 ve 361. basamakları sırasıyla 3, 6 ve 0'dır, yani 360." [1]

M.Ö. 2000 yılı civarında ;r'nin tüm çemberler için sabit bir değer olduğunun farkına varıldı. Böylece ***n*** sayısının serüveni başlamış oldu. Pi sayısının esrarının anlaşılması ile ilgili çalışmaları incelediğimizde karşımıza çıkan ilk bulgu, Mısırlıların ve Babillilerin kullandıkları ***n*** değerinin, bugün bilinen sayısal değerine yakın olduğudur. [1]

Mısırlılardan kalan ve 1858'de Nil kıyısında Thebes'de yıkılmış bir binada bulunan Rhind Papirüsü' ne göre; bir kenarı 8 birim olan bir karenin alanının, çapı 9 birim olan dairenin alanına

eşit olduğunu kabul eden Mısırlılar ***n*** = 4 - = 3.16049bağıntısını buldular. Babilller ise daire

içine düzgün bir altıgen çizerek, bu altıgenin çevresinin kendini çevreleyen çemberin çevresine

oranını hesaplayarak ***n*** = 3- = 3.125 bağıntısını buldular.

M.Ö 287-212 yılları arasında yaşayan Arkhimedes ise bir daireyi, içinden ve dışından n kenarlı düzgün çokgenlerle sınırlandırdı. Elde ettiği şekilde, içerideki çokgenin çevresi de, biri yukarıdan diğeri de aşağıdan, dairenin çevresine yaklaşıyordu. Arkhimedes düzgün altıgen ile başlayıp kenar sayılarını 2' ye katlayarak, 96 kenarlı bir çokgene ulaştı ve pi için şu bağıntıyı elde etti:

$$3\frac{10}{71} < \boldsymbol{n} < 3\frac{1}{7}$$

Bu yöntem sonraki 1800 yıl içinde ;r'yi hesaplamak için temel alındı ve François Viète belki de bu formülü kullanan son kişi olarak ***n*** için ilk sonsuz açılımı verdi:

$\pi = V2\ .\ \ 2 + 2V2\ .\ \ 2 + 2V2\ .\ \ 2 + 2V2 + 2V2\ .$

Bu bağıntıyla karşı karşıya kalınca korkmamak elde değil! Ama bir kere Viète bu tarz bağıntıların önünü açmış oldu ve daha sonra ilk adımı aşağıdaki bağıntıyla atan John Wallis oldu:

$^{\pi}$ 2.2.4.4.6.6.8.8.10.10...

1.3.3.5.5.7.7.9.9.11...

Wallis' in ardından ise Brouncker şu ifadeyi ortaya çıkardı:

4 _____ 1 9

2 - --------

5

2 + -----

2 + ...

2

Rakamların birbirleriyle olan ilişkilerinin sergilediği garip düzenlilik ve süreklilik, insanoğluna π 'nin büyülü olduğunu düşündürüyor. Ama gerçekte bunlar, diferansiyel ve entegral teorileri kullanılarak kolaylıkla elde edilebilecek formüller. Bu teorilerin ortaya çıkması hem daha çok basamak hesaplanmasına hem de π 'nin özellikleri hakkında daha çok bilgi edinmemize olanak verdi [1].

1671' de James Gregory' in kullandığı yöntem, gelecekte π hesabının seyrini değiştirecek kadar güçlüydü. Gregory Taylor serileri açılımını kullanarak

π 11 1

— = 1-- + -------+ ...

4 3 5 7

ifadesini buldu. Seri açılım kullanılarak yapılan bu hesaplama o kadar yayıldı ki; Newton ve Leibniz gibi ünlü matematikçiler bile π 'nin basamaklarını hesaplamakla uğraştılar. Hatta 1706'da John Machin, benzer bir yöntemle pi sayısını 100. basamağına kadar hesapladı [1].

Pi sayısı için yapılan diğer yaklaştırmalara [2] bakacak olursak: •

M.Ö. 1100'de Çinliler π = 3 değerini kullandılar;

377

= 38′30″ =

142

ang Fau π =

kullandılar:

- 2. yy.da Claudius Ptolemy π = 3 o 8′30″ =------ = 3.1466... değerini kullandı;

120

- 3. yy.da Wang Fau π = ------ = 3.155... , Chung Hing ise,π = >10 = 3.166... değerlerini

• 263'de Liu Hui π 3.14 50 62,832 değerini kullandı; = 3.1416

• 530'da Aryabhata π 20,000 değerini kullandı;

• 1220'de Leonardo de Pisa (Fibonacci) π= 3.141818... değerini buldu;

• 1573'de Valentinus Otho π 355 3.1415929 değerini buldu;

• 1736'da Leonhard Euler 1 + . 1 1 2 1 2 + 1 3 2 + 3 7t bağıntısını buldu.

Tablo 1'de π sayısının hesaplanmasındaki hassasiyetin (doğru basamak sayısının) tarihsel gelişiminin bir özeti sunulmuştur [3].

Tablo 1: Pi'nin Nümerik Tarihi

Kişi	**Zaman**	**Doğru Basamak Sayısı**
Archimedes	M.Ö. 240	3
Ptolemy	150	3
Liu Hui	263	5
Tsu Ch'ung Chi	480?	7
Al-Kashi	1429	14
Romanus	1593	15
Van Ceulen	1615	35
Sharp	1699	71
Machin	1706	100
Strassnitzky ve Dase	1844	200
Rutherford	1853	440
Shanks	1874	527
Reitwiesner ve ark. (ENIAC)	1949	2,037
Genuys	1958	10,000
Shanks ve Wrench	1961	100,265
Guilloud ve Bouyer	1973	1.001.250
Miyoshi ve Kanada	1981	2.000.036
Kanada, Yoshino ve Tamura	1982	16.777.206
Gosper	1985	17.526.200
Bailey Jan.	1986	29.360.111
Kanada ve Tamura	Eylül 1986	33.554.414
Kanada ve Tamura	Ekim 1986	67.108.839
Kanada ve ark.	Ocak 1987	134.217.700
Kanada ve Tamura	Ocak 1988	201.326.551
Chudnovskys	Mayıs 1989	480.000.000
Kanada ve Tamura	Temmuz 1989	536.870.898
Kanada ve Tamura	Kasım 1989	1.073.741.799
Chudnovskys	Ağustos. 1991	2.260.000.000
Chudnovskys	Mayıs 1994	4.044.000.000
Kanada ve Takahashi	Ekim 1995	6.442.450.938
Kanada ve Takahashi	Temmuz 1997	51.539.600.000
Kanada ve Takahashi	Eylül 1999	206.158.430.000
Kanada, Ushiro ve Kuroda	Aralık 2002	1.241.100.000.000

Pİ SAYISININ HESAPLANMASI İÇİN İKİ METOT

Monte-Carlo Metodu

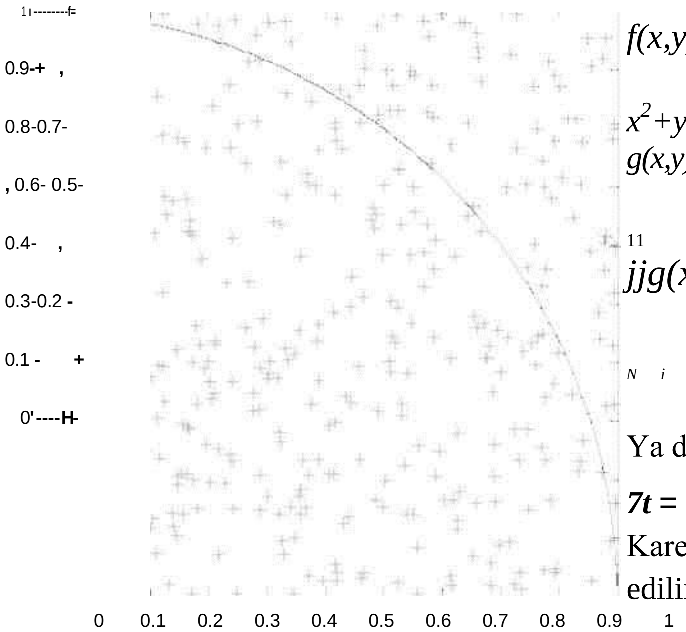
Monte-Carlo metodunu genel olarak istatistiksel simülâsyonların yapılması için rasgele sayılardan faydalanılan bir metot olarak tanımlayabiliriz. Monte-Carlo Nicholas Constantine Metropolis tarafından bulundu ve Stanislaw Ulam tarafından yaygınlaştırıldı [4,5]. Monte-Carlo metodunun en yaygın uygulaması aşağıda formülü verilen integralinin hesaplanmasıdır.

$$W = \int_0^1\int_0^1 \ldots \int_0^1 f(u_1,u_2,\ldots,u_n)\,du_1 du_2 \ldots du_n \qquad \int_{(0,1)^n} f(\mathbf{u})\,d\mathbf{u}$$

Burada *f* çok değişkenli bir fonksiyonu, *u* 'lar ise değişkenleri temsil eder. Genelde böyle karmaşık integrallerin analitik metotlarla bulunması, eğer imkânsız değilse, çok zordur.

Monte-Carlo yaklaşımında böyle integralleri hesaplamak için *u* değişkenlerinden gelişigüzel *binlerce* kombinasyon üretilir. Burada *binlerce* ifadesi istenilen hassasiyet seviyesi ve kullanılan bilgi-işlem kapasitesine göre değişiklik gösterebilecek büyük rakamlara karşılık gelebilir. Bu yaklaşım istatistik dışında çeşitli bilim dallarında da kullanılmaktadır.

Monte-Carlo integralini karesel bir alan içinde kalacak şekilde tekdüze (uniform) dağılımdan üretilen gelişigüzel noktalardan dairesel bir bölgeye düşen noktaları sayma işlemine adapte edersek, aşağıdaki bağıntı elde edilir (Şekil 1).



$f(x,y) = 1$

$$g(x,y) = \begin{cases} & x^2+y^2>1 \\ + & x^2+y^2<1, \end{cases}$$

$$\int_0^1\int_0^1 g(x,y)f(x,y)\,dxdy$$

$N \quad i$ [5].

Ya da daha açık bir ifadeyle,

π = Çemberin içindeki nokta sayısı 4 Karenin içindeki nokta sayısı bağıntısı elde edilir.

Şekil 1: Pi sayısının Monte-Carlo metoduyla bulunması.

Gregory-Leibniz Formülü

James Gregory ve Gottfried Leibniz birbirlerinden bağımsız olarak arctanjant fonksiyonun Taylor serisi açılımını kullanarak**;r** ’yi hesaplamak amacıyla aşağıdaki bağıntıyı buldular [6]:

f(x) = f(a) + f(a)(x-a) + 2! (*x - a*) + 3! + ... = *n* =0*Şn* ! (*x-a*)

" (-1)*n* 1 1 1 1 ***n***
2^*n* =-------+----- +... =—
n=0 •^" + 1 1 3 5 7 4

SONUÇLAR

Her iki metodu da C programlama dilinde gerçekledik ve Tablo 2 ve 3’deki sonuçları elde ettik. (Metotları gerçekleyen kaynak kodları makelenin sonunda **Ekler** bölümünde verilmiştir.)

Tablolardaki ***n*** Monte-Carlo metodu için oluşturulacak rasgele sayıların sayısını, Gregory-Leibniz formülü için ise terim sayısını ifade etmektedir. Tablolardaki ***4nS*** ifadesi ise ***n*** değerlerine göre oluşan ***n*** sayısını temsil etmektedir. Burada parantez içindeki değerler ***n*** ’nin yanlış basamakları olup dışındakiler ise doğru basamaklarıdır. Bu iki metodun ;r’yi ne kadar zamanda hesapladığını incelemek/karşılaştırmak için tabloların son sütünlarında zaman (saniye olarak) rapor edilmiştir. İki metodun da C dilinde yazılan kodları aynı platformda/bilgisayarda çalıştırılmıştır. Microsoft XP Media Center Edition işletim sistemli bu bilgisayarın donanım özellikleri şöyledir: AMD Turion 64 Çift çekirdekli 1.81 GHz mikroişlemci ve 1 GB RAM (random erişimli bellek).

Tablo 2: Monte-Carlo Metodu Sonuçlar ı

n	**4*nS***	**Zaman (sn)**
50	3.(15789473684211000)	0.005
100	3.14(285714285714000)	0.010
500	3.(39037433155080000)	0.178
1000	3.1(3620689655172400)	0.689
1.000.000	3.141(80000000000000)	1
1.000.000.000	3.1415(3717599999990)	182

Tablo 3: Gregory-Leibniz Formülü Sonuçları

n	**4*nS***	**Zaman (sn)**
50	3.1(611986129870506)	0.001
100	3.1(315929035585537)	0.002
500	3.14(35886595869501)	0.078
1000	3.14(05926538397940)	0.156
1.000.000	3.14159(16535897743)	0.42
1.000.000.000	3.14159265(25880504)	60

Sonuç olarak, bu iki metodu da karşılaştırdığımız zaman Monte-Carlo metodunun görsellik ve π'nin basamaklarının nasıl hesaplandığını öğrenme açısından daha avantajlı olduğunu görüyoruz. Gregory-Leibniz formülü ise π'nin basamaklarının daha hassas/doğru ve hızlı hesaplanmasında kullanılmasının daha verimli olduğunu söyleyebiliriz.

Referanslar

[1] *Gizemli Bir Sayı P*i, Sayfa 64–68, Bilim Teknik Dergisi Mart 96,Sayı 340.

[2] http://www.mathsci.appstate.edu/~sjg/class/3010/final/andy.pdf

[3] http://crd.lbl.gov/~dhbailey/dhbpapers/dhb-kanada.pdf

[4] http://orion.math.iastate.edu/msm/ReeceBMSMSS06.pdf

[5] http://www.phy.hr/~laci/para/mc/mc.html#Integration

[6] http://en.wikipedia.org/wiki/Leibniz_formula_for_pi

Ekler (C Kodları)

- **Monte-Carlo Metodu'nun C programlama dilinde gerçeklenmesi**

```
#include        <stdio.h>
#include       <stdlib.h>
#define seed 391891246

int main( )

{

double iteration=0.0,count=0.0;
double x,y,z,pi;

printf("Enter the number of iterations used to estimate pi: ");
```

```
scanf("%lf", &iteration);

srand (seed);

for (double i=0; i < iteration ; i++) {

    x = (double)rand( ) / RAND_MAX;
    y = (double)rand( ) / RAND_MAX;
    z = (x*x) + (y*y);

    if (z<=1) count++;

    }

    pi=(double)count/iteration*4;

    printf("# of trials= %g , estimate of pi is %20.20f \n",iteration,pi);
    system("pause");
    return 0;

    }
```

- **Gregory/Leibniz formülünün C programlama dilinde gerçeklenmesi**

```
#include  <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()

{

double pi=0.0,num=4.0,denom=1.0;
double loop,accuracy;

printf("Set Accuracy : ");
scanf("%lf",&accuracy);

printf("term\t\t\t pi\t\t\ttime\n");

for (loop=1;loop<=accuracy;loop++){

    if (loop / 2 !=floor(loop / 2))
    pi+=(num/denom);

    else pi-=(num/denom);

        denom+=2.0;

        }

        printf("%g\t\t %20.20f \n",loop-1,pi);
        system("PAUSE");
        return 0;
        }
```