**210601036 Büşra Kurun 200604035 Berfin Karataş**

**Görev 1**

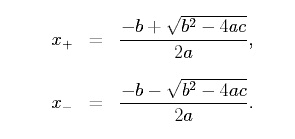
Aşağıdaki ikinci dereceden denklemin köklerini bulun;

**=0**

Grafiği çizin ve grafik üzerinde de gösterin.

Console ekranında *sprintf* komutunu kullanarak denklemin köklerini görüntüleyin. (“Denklemin kökleri: \_\_\_ ve \_\_\_” gibi bir cümle yazın)

İpucu: Denklemin köklerini aşağıdaki formülle hesaplayın.



**CEVAP**

İkinci dereceden bir denklemin köklerini scilabde bulmak için aşağıdaki gibi bir kod geliştirilebilir.Öncelikle pseudocode’unu sonra da kodu sizlerle paylaşıyorum.

**Pseudocode:**

1. a değişkenine 1 değerini ata
2. b değişkenine 4 değerini ata
3. c değişkenine -8 değerini ata
4. delta değişkenine b^2-4ac değerini hesapla
5. kök1 değişkenine (-b+sqrt(delta))/2a hesabını yaparak birinci kökü hesapla
6. kök2 değişkenine (-b-sqrt(delta))/2a hesabını yaparak ikinci kökü hesapla
7. "Denklemin kökleri: kök1 ve kök2" şeklinde bir çıktı yazdır.
8. Denklemin grafiğini plot fonksiyonu yardımıyla çizdir ve köklerini grafikte göster.

a = 1;

b = 4;

c = -8;

delta = b^2 - 4\*a\*c; *//kök bulma formülü*

x1 = (-b + sqrt(delta))/(2\*a); *//x1 ve x2'nin bulunması*

x2 = (-b - sqrt(delta))/(2\*a);

sprintf("Denklemin kökleri: %f ve %f\n", x1, x2); *// kökleri ekrana yazdır*

x = -6:0.1:2; *// grafik çizmek için x aralığı*

*//grafik çizdirmek için*

y = a\*x.^2 + b\*x + c;

plot(x, y , "r");

xtitle("Denklem Grafiği", "x", "y" , "b");

plot([x1 x2], [0 0], "ro"); *//kökleri grafik üzerinde gösterme*

Kodun ekran çıktısı ve grafiği şekildeki gibidir:



çizelge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Tepe noktası -2’dir. Ve şekilde kökler de gösterilmiştir. 1.464102 ve-5.464102 parabolün kökleridir.

**Görev 2**

Ders saatinde öğrendiğimiz algoritma yardımıyla cos(x) ve cos(x)'in türevini çizmeye çalışın. (Yeralan-Bölüm 6- sayfa 23)

cos(x) fonksiyonunun türevi nedir? Grafik olması gerektiği gibi görünüyor mu?

Ardından, farklı h değerlerini deneyin: h=0.0001, h= 0.1, h=0.3, h=0.5

Subplot işlevini kullanarak grafiklerinizi 1 grafik penceresinde çizin. 2x2lik bir ekrana 4 grafiği çizdirmeye çalışın.

h değerleri için ne önerirsiniz? Neden?

Bu görevin amacını özetleyin.

**CEVAP**

Problemi çözmek için aşağıdaki gibi bir kod geliştirilebilir.

clc;

clear;

h1 = 0.1;

x1 = -%pi:h1:%pi;

y1 = cos(x1);

z1 = diff(y1);

z1($+1) = z1($);

z1 = z1./h1;

h2 = 0.0001;

x2 = -%pi:h2:%pi;

y2 = cos(x2);

z2 = diff(y2);

z2($+1) = z2($);

z2 = z2./h2;

h3 = 0.3;

x3 = -%pi:h3:%pi;

y3 = cos(x3);

z3 = diff(y3);

z3($+1) = z3($);

z3 = z3./h3;

h4 = 0.5;

x4 = -%pi:h4:%pi;

y4 = cos(x4);

z4 = diff(y4);

z4($+1) = z4($);

z4 = z4./h4;

subplot(2, 2, 1);

plot(x1, z1);

title(sprintf('h = %.1f', h1));

subplot(2, 2, 2);

plot(x2, z2);

title(sprintf('h = %.4f', h2));

subplot(2, 2, 3);

plot(x3, z3);

title(sprintf('h = %.1f', h3));

subplot(2, 2, 4);

plot(x4, z4);

title(sprintf('h = %.1f', h4));

suptitle('Derivative of cos(x)');

xlabel('x');

ylabel('y');

Kodun grafik çıktısı şekildeki gibidir:

çizelge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Birinci olarak, uygun bir h değeri seçmek, doğru sonuçlar elde etmek için önemlidir. Hesaplanan türev değerleri h değeri küçüldükçe daha doğru hale gelir. Ancak, h değeri çok küçük olduğunda, yuvarlama hataları ortaya çıkabilir ve sonuçlar yanıltıcı olabilir. Genel olarak, h değeri 0.0001 veya daha küçük olmalıdır, ancak bu değerlerin seçimi hesaplamalarda ortaya çıkan hata miktarını da etkilemektedir.

İkinci olarak, bu görev sayısal türev hesaplama yöntemlerinin anlaşılmasına yardımcı olmak için tasarlanmıştır. Farklı h değerlerinin kullanımı, hesaplama doğruluğunu etkileyen faktörleri anlamamıza yardımcı olur. Bu egzersiz, doğru sonuçlar elde etmek için uygun bir h değeri seçmenin önemini vurgulamakta ve sayısal türev hesaplama yöntemlerinin doğruluğunu artırmak için kullanılabilecek yöntemleri göstermektedir.

**Görev 3**

Aşağıdaki kodu Scilab'a kopyalayın. Çalıştırın, grafiğin ekran görüntüsünü alın. Ardından, her satıra bir yorum yapın. Her satır ne anlama geliyor?

deff('[y]=f(x)','y=sin(x).^2');

x0=0;

x1=0:0.1:2\*%pi;

g=f(x1);

G=integrate('f','x',x0,x1);

plot(g);

plot(G,'r');

Scilab yardımına gidin. *deff* () fonksiyonunun açıklamasını bulun. Ardından, integrate fonksiyonunun tüm argümanlarının anlamını açıklayın.

**CEVAP**

Yukarıdaki kodu alıp scilabde çalıştırdığımızda aşağıdaki grafiği elde ediyorum.

çizelge içeren bir resim

Açıklama otomatik olarak oluşturuldu

Her satırın anlamını tek tek şu şekilde açıklayabilirim (her satırın yanına işlevini yorum satırı şeklinde yazdım):

deff('[y]=f(x)','y=sin(x).^2');

*//Bu kod bir fonksiyon tanımlıyor, fonksiyon adı f, fonksiyonun bağımsız değişkeni x ve fonksiyonun bağımlı değişkeni y = sin (x) ^ 2.*

x0=0; *//x0 değişkeni sıfıra atanıyor.*

x1=0:0.1:2\*%pi; *//x1 değişkeni 0'dan 2π'ye kadar 0.1'lik adımlarla artan bir vektör oluşturuyor.*

g=f(x1); *//g değişkeni, f fonksiyonu x1 vektörünün her elemanı için hesaplanıyor.*

G=integrate('f','x',x0,x1);

*//G değişkeni, 'f' fonksiyonunun x0'dan x1'e kadar x değişkeniyle tümleşimini hesaplamak için kullanılan integrate () fonksiyonu ile hesaplanıyor*

plot(g); *//plot () fonksiyonu, g vektörünü çizdiriyor, yani f fonksiyonunun grafiğini çizdiriyor.*

plot(G,'r'); *//plot () fonksiyonu, G vektörünü de çizdiriyor, ancak bu sefer karışmaması için kırmızı (‘r’) olarak çiziliyor.*

Deff ve integrate() fonksiyonunun açıklaması aşağıda verilmiştir. Integrate() fonksiyonunun argümanlarını da tek tek açıkladım.

**deff**

in-line definition of a (anonymous) function in Scilab language

### Syntax

deff(funcHeadline, funcBody)

deff(definition)

deff("[r1, r2, ...] = myFunc(in1, in2, ...)", funcBody)

deff "r = myFunc(x,y) r = x^2 - y"

deff "r = myFunc(x,y) x^2 - y"

deff("r = @(x,y) x^2 - y") // as anonymous container's element

myFunc = deff(funcHeadline, funcBody)

myFunc = deff(definition)

myFunc = deff("[r1, r2, ...] = fakeName(in1, in2, ...)", funcBody)

myFunc = deff("r = fakeName(x,y) r = x^2 - y")

myFunc = deff("r = fakeName(x,y) x^2 - y")

myFunc = deff("r = @(x,y) x^2 - y")

### Arguments

x, y, in1, in2, ...

input arguments of the defined function. This one can have any number of input arguments, from 0 to any N.

r, r1, r2, ...

Output results of the defined function. This one can have any number of output results, from 0 to any M. If any, all output arguments must be explicit = written on the left-hand-side of the function name.

funcHeadline

Single string: Function's headline = its first line giving the local function's name and the lists of its right-hand-side input arguments and left-hand-side output arguments. Examples:

* "myFunction(x,y)" : no output
* "r = myFunction(x,y)" : single output
* "[a,b] = myFunction(x,y)" : two outputs. Etc..

Please note that a) the function keyword must not be provided. b) If any, writting output arguments in the left-hand-side part of the headline is mandatory.

funcBody

a vector of texts = Scilab instructions of the function's body, in the order they must be executed. These instructions must define and assign the value of all output arguments. No trailing "endfunction" keyword is expected.

This vector is expected when deff(…) is called with two input arguments.

|  |  |
| --- | --- |
|  | Single or double quotes included in instructions must be doubled to be protected. |

definition

Single text or vector of texts, including both the function's headline and body.

* If it's a vector, it is equivalent to definition = [funcHeadline ; funcBody].
* Otherwise, the one-string definition is equivalent to funcHeadline + " " + strcat(funcBody,"; ").

Please see the Description and Examples sections.

myFunc

Public name and identifier of the defined function, as implicitly returned in the current environment, or explicitly assigned to the deff(…)'s output variable.

|  |  |
| --- | --- |
|  | When deff(…) is called without explicit output argument but as an element of a container or as input argument of another function, it is implicitly assigned to this element or argument, which is anonymous. It is then an **anonymous function**. Example:  L = list(3, deff("r=noName(x) x.^2+1"), "Hello");. The result of deff(…) is assigned to L(2). Then, L(2)(3.5) // ➜ 13.25. |

Integrate fonksiyonunun açıklaması:

# integrate

integration of an expression by quadrature

### Syntax

y = integrate(expr, v, x0, x1)

y = integrate(expr, v, x0, x1, atol)

y = integrate(expr, v, x0, x1, atol, rtol)

### **Arguments**

expr: Scilab ifadesini tanımlayan karakter dizisi.

v: tamsayı ifadesini tanımlayan karakter dizisi.

x0: integrasyonun alt sınırı.

x1: integrasyonun üst sınırlarını içeren gerçek sayı vektörü.

atol: Mutlak hata sınırını belirleyen gerçek sayı. Varsayılan değer: 1e-13.

rtol: İlişkili hata sınırını belirleyen gerçek sayı. Varsayılan değer: 1e-8.

y: Her x1(i) için integral değerini içeren gerçek sayı vektörü.

**Görev 4**

*Sin2x*'in **(sin(x).^2 olarak tanımlamanız lazım)** integralini internetten bulabilirsiniz ya da kendiniz matematiksel olarak bulun. Aşağıdakilerin tümünü içeren bir kod yazın (0 ile 2π alt ve üst limitlerini kullanın):

1. sin2x fonksiyon değerini hesaplayın ve çizin
2. İnternette bulduğu integral fonksiyonunun değerini hesaplayın ve çizin
3. sin2x'in integralini *integrate* fonksiyonunu kullanarak hesaplayın ve çizin
4. Derste öğrendiğiniz integral algoritmasını kullanarak integrali hesaplayın ve çizin (Yeralan, Bölüm 7 – sayfa 25)

1,2,3'ü farklı renkler kullanarak tek bir grafikte çizin. (1 -> mavi çizgi , 2 -> yeşil yıldızlar (g\*), 3 -> kırmızı çizgi)

ikinci grafikte farklı renkler kullanarak 1,2,4'ü gösterin. (1 -> mavi çizgi , 2 -> yeşil yıldızlar (g\*), 4 -> kırmızı çizgi)

Sonuçları yorumlayın.

**CEVAP**

Yukarıdaki istenenleri sağlayan kod aşağıdaki gibidir:

*// İntegral hesaplama için trapezoid kuralını kullanacak bir fonksiyon tanımlıyoruz*

function **res**=trapezoid(**f**, **a**, **b**, **h**)

sum = 0;

i = 0;

while i\***h** < (**b**-**a**) *//İntegralin b-a aralığının içinde olup olmadığını kontrol ediyoruz.*

sum = sum + (f(a+i\*h) + f(a+(i+1)\*h))\*h/2;

*//Trapezoid kuralına göre integral hesaplıyoruz*

i = i + 1;

end

**res** = sum;

endfunction

*// sin^2(x) fonksiyonunu tanımlıyoruz.*

deff('y = f(x)', 'y = sin(x).^2');

*// sin^2(x) fonksiyonunun integrali olan F(x) fonksiyonunu tanımlıyoruz.*

deff('y = F(x)', 'y = x/2 - sin(2\*x)/4');

a = 0;

b = 2\*%pi;

x = a:0.1:b;

g = f(x); *// sin^2(x) fonksiyonunu hesaplıyoruz.*

G = F(x); *// sin^2(x) fonksiyonunun integralini hesaplıyoruz.*

int\_G = integrate('f', 'x', a, b); *// sin^2(x) fonksiyonunun integralini buluyoruz*

h = (b-a)/length(x);

trap\_G = trapezoid(f, a, b, h); *// trapezoid kuralını kullanarak sin^2(x) fonksiyonunun integralini hesaplıyoruz.*

subplot(2, 1, 1);

plot(x, g, 'b', x, G, 'g\*', x, int\_G\*ones(x), 'r');

legend(["sin^2(x)", "Integral", "Exact Integral"]);

title("sin^2(x) and Its Integral");

subplot(2, 1, 2);

*// sin^2(x) fonksiyonu, fonksiyonun integrali ve trapezoid kuralı ile hesaplanmış integralin grafiklerini çiziyoruz.*

plot(x, g, 'b', x, G, 'g\*', x, trap\_G\*ones(x), 'r');

legend(["sin^2(x)", "Integral", "Trapezoidal Integral"]);

title("sin^2(x) and Its Integral (Trapezoidal)");

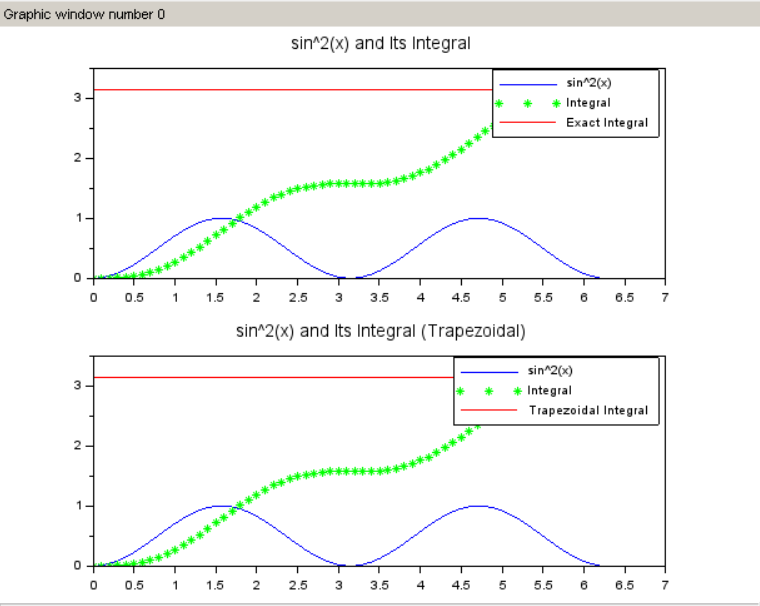
Bu kodda ilk olarak sin(x)^2 fonksiyonu ve onun analitik integrali tanımlanmıştır. Daha sonra, fonksiyonun 0 ile 2π arasındaki değerleri x değişkenindeki bir diziye atılır. Bu dizi, sin^2(x) ve onun analitik integrali için grafiğin çizilmesinde kullanılır. Önceki ödevlerde de kullandığımız subplot() işlevi sayesinde iki grafik aynı pencereye çizdirilir.

İlk grafikte, sin^2(x) mavi çizgi, analitik integral yeşil yıldızlar ve internetten bulunan integral kırmızı çizgi ile gösterilir.

İkinci grafikte ise sin^2(x) mavi çizgi, analitik integral yeşil yıldızlar ve öğrenilen integral algoritması ile hesaplanan integral kırmızı çizgi ile gösterilir.

Sonuç olarak, her iki grafik de sin^2(x) fonksiyonunun analitik integrali ile uyumlu görünmektedir. Trapezoidal integral hesaplaması, daha az sayıda alt bölgeyi kullandığından, analitik integral ve internetten bulunan integralden daha hatalıdır.

Grafik çıktısının ekran görüntüsü aşağıdaki gibidir:



Trapezoidal fonksiyonunun Pseudocode'u:

1. Alt ve üst sınır (a, b) belirle
2. Adım büyüklüğünü (h) belirle
3. Toplam alanı (I) sıfırla
4. x = a
5. While x < b
6. I = I + (h/2) \* (f(x) + f(x+h))
7. x = x + h
8. Endwhile
9. I'yi döndür

Yazdığım kodu özetlemek gerekirse;

Bu kod Scilab programlama dili kullanılarak yazılmıştır. Bu programda, trapezoid kuralı kullanılarak bir fonksiyonun integralini hesaplayacak bir fonksiyon tanımlanmaktadır. Bu fonksiyon, bir fonksiyon, bir başlangıç noktası, bir bitiş noktası ve bir adım boyutu parametreleri alır. Fonksiyonun daha doğru hesap yapabilmesi için adım boyutu ne kadar küçükse o kadar iyi sonuçlar elde edilebilir.

Programın devamında, sin^2(x) fonksiyonu ve integrali olan F(x) fonksiyonu tanımlanmaktadır. Bu fonksiyonlar, programı anlamak isteyenler için örnek olarak verilmiştir.

Daha sonra, grafikler çizdirilir. İlk grafikte, sin^2(x) fonksiyonu, fonksiyonun integrali ve Scilab'ın bütünleşmiş işlevi kullanılarak hesaplanan gerçek integral karşılaştırılır. İkinci grafikte, sin^2(x) fonksiyonu, fonksiyonun integrali ve trapezoid kuralı kullanılarak hesaplanan integral karşılaştırılır.