Задание 1. Найти абсолютную и относительную погрешность выражения

a=4.5, b=3.75, c=2.78. Все цифры чисел верные. Полученный результат представить с максимальным количеством верных цифр.

Выпишем погрешности чисел a,b,c: , , .

Вычислим величину f и запишем результат с пятью знаками после запятой:

.

Абсолютную погрешность величины f вычислим по формуле:

=

=

=

,

,

.

Тогда . Значит, первые два разряда величины f верные и f следует записать следующим образом:

.

Вычислим относительную погрешность величины f:

%= %=10.21%.

Задание 2. Составить таблицу значений производной функции f(x) на промежутке [0,2] с шагом 0.1, вычисленной по приближенным формулам правой, левой и центральной производной.

f(x)=

;

;

.

Точность .

1) правая производная:

|  |  |
| --- | --- |
| h |  |
| 0.01 | 44.33 |
| 0.005 | 26.09 |
| 0.0025 | 13.87 |
| 0.00125 | 7.12 |
| 0.006265 | 3.61 |
| 0.0003125 | 1.81 |
| 0.00015625 | 0.91 |
| 0.000078125 | 0.46 |
| 0.0000390625 | 0.23 |
| 0.00001953125 | 0.10 |
|  | 0.05 |
|  | 0.09 |
|  | 0.12 |
|  | 0.19 |
|  | 0.19 |
|  | 1.02 |
|  | 2.10 |

При дальнейшем увеличении шага h разность между предыдущим и следующим значениями только увеличивается. Очевидно, что требуемой точности достичь не получается. Это объясняется некорректностью задачи численного дифференцирования — уменьшение шага не гарантирует повышения точности. Минимальная разница между предыдущим и последующим решением получена при .

2) левая производная

|  |  |
| --- | --- |
| h |  |
| 0.01 | 64.98 |
| 0.005 | 32.32 |
| 0.0025 | 15.19 |
| 0.00125 | 7.45 |
| 0.006265 | 3.69 |
| 0.0003125 | 1.83 |
| 0.00015625 | 0.91 |
| 0.000078125 | 0.45 |
| 0.0000390625 | 0.23 |
| 0.00001953125 | 0.13 |
|  | 0.06 |
|  | 0.07 |
|  | 0.13 |
|  | 0.18 |
|  | 0.17 |
|  | 1.01 |
|  | 2.08 |

При дальнейшем увеличении шага h разность между предыдущим и следующим значениями только увеличивается. Очевидно, что требуемой точности достичь не получается. Это объясняется некорректностью задачи численного дифференцирования — уменьшение шага не гарантирует повышения точности. Минимальная разница между предыдущим и последующим решением получена при .

3) центральная производная

|  |  |
| --- | --- |
| h |  |
| 0.01 | 10.32 |
| 0.005 | 2.62 |
| 0.0025 | 0.66 |
| 0.00125 | 0.16 |
| 0.006265 | 0.04 |
| 0.0003125 | 0.01 |
| 0.00015625 | 0.0038 |
| 0.000078125 | 0.0034 |
| 0.0000390625 | 0.001 |
| 0.00001953125 | 0.016 |
|  | 0.007 |
|  | 0.065 |
|  | 0.128 |
|  | 0.18 |
|  | 0.18 |
|  | 1.02 |
|  | 2.09 |

И в этом случае не получается достичь требуемой точности. Минимальная разность между решениями при h и h/2 достигается при h=.

Вычислим значения производной по формулам правой, левой, центральной разности при указанных значениях шага h.

Погрешность представим в таблице:

|  |  |  |
| --- | --- | --- |
| правая разность | левая разность | центральная разность |
| 0,000402 | 0,000798 | 8,17E-05 |
| 0,000759 | 0,000541 | 6,06E-05 |
| 0,000222 | 2,15E-05 | 8,15E-05 |
| 0,000539 | 0,000861 | 1,92E-05 |
| 0,000165 | 0,000635 | 0,000165 |
| 0,000566 | 0,000334 | 1,40E-05 |
| 0,000773 | 0,001427 | 5,69E-05 |
| 0,000358 | 0,000342 | 5,20E-05 |
| 0,000484 | 0,000616 | 2,36E-05 |
| 0,000661 | 0,001039 | 2,14E-05 |
| 0,001886 | 0,000286 | 0,000274 |
| 0,002058 | 0,002842 | 0,000218 |
| 0,005931 | 0,001069 | 0,000859 |
| 0,004144 | 0,008956 | 0,000814 |
| 0,00879 | 0,00043 | 0,0016 |
| 0,019015 | 0,013185 | 0,001195 |
| 0,033903 | 0,023997 | 0,001217 |
| 0,037045 | 0,019255 | 0,002385 |
| 0,027363 | 0,043737 | 0,002863 |
| 0,068636 | 0,046864 | 0,003814 |
| 0,120212 | 0,104988 | 0,002998 |

Цветом выделена наибольшая погрешность во всей таблице.

Задание 3. Протабулировать заданную функцию  на отрезке [0,2] с шагом *h=*0,1, используя интерполяционный многочлен Лагранжа 

Построить графики функций 

Оценить погрешность интерполяции по формуле



где и сравнить с фактической 

Интерполяционные многочлены определять по схеме Лагранжа и Ньютона, коэффициенты многочленов выводить с точностью до пятого знака после запятой.

f(x)=

Для n=4 полином Лагранжа имеет вид:

, где

,

,

,

,

.

=



Интерполяционный многочлен в форме Ньютона имеет вид:

, где:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| =16 | =9.7071 | =5 | =1.7071 | =0 |
| =-6.2929 | =-4.7071 | =-3.2929 | =-1.7071 |  |
| =1.5858 | =1.4142 | =1.5858 |  |  |
| = - =-0.1716 | = - =0.1716 |  |  |  |
| = - = 0.3432 |  |  |  |  |

=



Вычислим погрешность для полинома Лагранжа . Так как полиномы Лагранжа и Ньютона отличаются лишь видом записи, а их значения совпадают в каждой точке, то погрешность аппроксимации полиномом Ньютона совпадает с погрешностью аппроксимации полиномом Лагранжа.

=, , , .

Аналогично для n=8:

=

=

Оценка погрешности:

=,

=, = .

Фактическая погрешность: = , = .

Изобразим функцию f(x) и 4 полинома в одних осях:



Задание 4. Вычислить приближенно с точностью , используя квадратурные формулы трапеций и Симпсона. Начальный шаг , итерационный процесс завершить по правилу Рунге. Полученный результат уточнить по формуле Ричардсона, сравнить результат с точным значением интеграла.

f(x)=

1) Метод трапеций заключается в следующем:

Отрезок интегрирования делим на отрезки длиной h и положим приближенно: . Суммируя по всем частичным отрезкам, получим приближенное значение интеграла на отрезке . При = , при . Погрешность полученного решения будем оценивать по правилу Рунге: . Будем уменьшать шаг , пока оценка погрешности по правилу Рунге не станет меньше априорно заданной погрешности . При .

Уточним решение по Ричардсону: .

Оценка погрешности

Точное значение интеграла: ;

.

2) Вычислительная схема метода Симпсона совпадает со схемой метода трапеций, с той разницей, что значение интеграла на отрезке вычисляется по формуле: .

При значение интеграла =, а = , a = <. Таким образом, достигнута необходимая точность и вычисления можно прекратить.

.

Уточним решение по Ричардсону: = .

Оценим погрешность:

= .

Задание 5. Найти все корни уравнения ****на отрезке [a,b] точностью .

На первом этапе следует провести процесс локализации корней, то есть определить отрезки, на которых имеется только один корень. Для этого с шагом надо вычислить значения функции и определить отрезки, где имеются корни функции. Далее, уменьшая шаг продолжать процесс локализации до тех пор, пока для четырех последовательных уменьшений шага число отрезков, содержащих корни функции, не станет постоянным.

На втором этапе определить корни функции методом деления отрезка пополам.

f(x)=.

a=0, b=3.

Начиная с h=3/100, для каждого отрезка будем проверять условие f(x)\*f(x+h)<0. Если условие выполняется, то на этом отрезке расположено нечётное количество корней уравнения f(x)=0. При h=3/100 обнаружено 5 таких отрезков. На следующем шаге уменьшаем шаг h вдвое и повторяем процедуру ещё 4 раза.

При h=3/800 количество отрезков, содержащих корни, не изменилось. Будем считать, что на каждом из этих отрезков расположено по одному корню.

Итак, получено пять отрезков, содержащих по одному корню уравнения:

[, ]

[, ]

[, ]

[, ]

[, ]

Найдём каждый из пяти корней уравнения методом половинного деления.

Для этого на каждой итерации будем делить отрезок [a,b] пополам точкой с. Если f(a)f(c)<0, то на отрезке [a,c] функция меняет знак, а значит, корень уравнения принадлежит этому отрезку. Значит, следует искать корень на отрезке [a,c]. Совершим присваивание: b:=c и перейдём к следующей итерации.

Если же f(b)f(c)<0, то корень уравнения принадлежит отрезку [c,b]. Присваиваем: a:=c и переходим к следующему шагу.

Можно завершить итерационный процесс, когда |b-a|<, или когда , где – априорно заданная погрешность.

Корни уравнения:



Значения функции f в найденных точках: , , , , .

Приложение. Тексты программ (Maple 6)

Задание 1.

> f:=(a^2+b\*(cos((3\*c-b)/c))^2)/(a\*c-b^2);



частная производная f по а

> dfa:=simplify(diff(f, a));



частная производная f по b

> dfb:=diff(f,b);



>

частная производная f по с

> dfc:=diff(f,c);



> Delta\_a:=0.05; Delta\_b:=0.005; Delta\_c:=0.005;







> evalf(subs(a=4.5, b=3.75, c=2.78, dfa));evalf(subs(a=4.5, b=3.75, c=2.78, dfb));evalf(subs(a=4.5, b=3.75, c=2.78, dfc));







абсолютная погрешность f

> Delta\_f:=abs(evalf(subs(a=4.5, b=3.75, c=2.78, dfa)))\*Delta\_a+abs(evalf(subs(a=4.5, b=3.75, c=2.78, dfb)))\*Delta\_b+abs(evalf(subs(a=4.5, b=3.75, c=2.78, dfc)))\*Delta\_c;



значение функции f

> fabc:=evalf(subs(a=4.5, b=3.75, c=2.78, f));



относительная погрешность f

> delta\_f:=evalf(subs(a=4.5, b=3.75, c=2.78, Delta\_f/f))\*100;



Задание 2.

> f:=exp(x+2)\*cos(2\*x^4)/(x^2+(ln(3+x))^3);



> x0:=0; t:=0.1; h:=0.01;







xt - точки сетки

> n:=20; xt:=vector(n+1);





> for i from 1 to n+1 do

xt[i]:=x0+(i-1)\*t;

od:

> print(xt);



h=h0

массивы производных, вычисленных по первому правилу. Шаг h и h/2

> proizv0:=vector(n+1):proizv1:=vector(n+1):

> for i from 1 to n+1 do

proizv0[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i], f))/h):

od:

> h:=h/2;



> for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i], f))/h):

od:

вычисляем модуль разности значений производных, вычисленных с разными значениями h

> diff1:=vector(n+1):

> for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

максимальная разность для производных, вычисленных по первому правилу:

> max1:=max(seq(diff1[i], i=1..n+1));



Цикл, в котором делим шаг на 2 и находим производные в точках сетки

> while max1>0.0001 do

h:=h/2;

for i from 1 to n+1 do proizv0[i]:=proizv1[i]: od:

for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i], f))/h):

od:

for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

max1:=max(seq(diff1[i], i=1..n+1));

print(proizv1);

od;





 - массив производных

- шаг (поделили предыдущий шаг пополам)

- максимальная разность между предыдущим и текущим решениями

 - массив производных

Последний найденный массив производных, при самом малом шаге:

> print(proizv1);



Комментарий к этому результату: шаг h очень мал, поэтому значения производных равны нулю, что не имеет ничего общего с действительностью.

вычисляем производные по второму правилу

> h:=0.01:

> proizv0:=vector(n+1):proizv1:=vector(n+1):

> for i from 1 to n+1 do

proizv0[i]:=evalf((subs(x=xt[i], f) - subs(x=xt[i]-h, f))/h):

od:

> h:=h/2;



> for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i], f) - subs(x=xt[i]-h, f))/h):

od:

вычисляем модуль разности значений производных, вычисленных с разными значениями h

> diff1:=vector(n+1):

> for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

максимальная разность для производных, вычисленных по первому правилу:

> max1:=max(seq(diff1[i], i=1..n+1));



>

> while max1>0.0001 do

h:=h/2;

for i from 1 to n+1 do proizv0[i]:=proizv1[i]: od:

for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i], f) - subs(x=xt[i]-h, f))/h):

od:

for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

max1:=max(seq(diff1[i], i=1..n+1));

print(proizv1);

od;

вычисляем производные по третьему правилу

> h:=0.01:

> proizv0:=vector(n+1):proizv1:=vector(n+1):

> for i from 1 to n+1 do

proizv0[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i]-h, f))/2/h):

od:

> h:=h/2;



> for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i]-h, f))/2/h):

od:

вычисляем модуль разности значений производных, вычисленных с разными значениями h

> diff1:=vector(n+1):

> for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

максимальная разность для производных, вычисленных по первому правилу:

> max1:=max(seq(diff1[i], i=1..n+1));



>

> while max1>0.0001 do

h:=h/2;

for i from 1 to n+1 do proizv0[i]:=proizv1[i]: od:

for i from 1 to n+1 do

proizv1[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i]-h, f))/2/h):

od:

for i from 1 to n+1 do

diff1[i]:=abs(proizv0[i]-proizv1[i]):

od:

max1:=max(seq(diff1[i], i=1..n+1));

print(proizv1);

od;

вычисляем истинные значения производной

> diff0:=vector(n+1):

> for i from 1 to n+1 do

diff0[i]:=evalf(subs(x=xt[i], diff(f, x))):

od:

получим значения левой, правой, центральной производной при выбранном шаге h

>

> h:=.3906250000e-4:

proizv\_central:=vector(n+1):

err\_central:=vector(n+1):

max\_c:=0:

for i from 1 to n+1 do

proizv\_central[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i]-h, f))/2/h):

err\_central[i]:=abs(proizv\_central[i]-diff0[i]):

od:

max\_c:=max(seq(err\_central[k], k=1..n+1));

- максимум разности между численным значением производной и истинным (для центральной разности)

> h:=.9765625000e-5:

proizv\_right:=vector(n+1):

err\_right:=vector(n+1):

max\_r:=0:

for i from 1 to n+1 do

proizv\_right[i]:=evalf((subs(x=xt[i]+h, f) - subs(x=xt[i], f))/h):

err\_right[i]:=abs(proizv\_right[i]-diff0[i]):

od:

max\_r:=max(seq(err\_right[k], k=1..n+1));

- максимум разности между численным значением производной и истинным (для правой разности)

> proizv\_left:=vector(n+1):

err\_left:=vector(n+1):

max\_l:=0:

for i from 1 to n+1 do

proizv\_left[i]:=evalf((subs(x=xt[i], f) - subs(x=xt[i]-h, f))/h):

err\_left[i]:=abs(proizv\_left[i]-diff0[i]):

od:

max\_l:=max(seq(err\_left[k], k=1..n+1));

- максимум разности между численным значением производной и истинным (для левой разности)

>

выводим погрешность в файлы

> file\_c:= fopen("central.txt", WRITE, TEXT):

writedata(file\_c, err\_central):

fclose(file\_c):

file\_r:= fopen("right.txt", WRITE, TEXT):

writedata(file\_r, err\_right):

fclose(file\_r):

file\_l:= fopen("left.txt", WRITE, TEXT):

writedata(file\_l, err\_left):

fclose(file\_l):

Задание 3.

> f:=sin(Pi\*x/2)+4\*(x-2)^2;



табулируем функцию при h=0.1

> n:=20: h:=0.1:

> x20:=vector(n+1): f20:=vector(n+1):

> for i from 1 to n+1 do

x20[i]:=x0+(i-1)\*h:

f20[i]:=evalf(subs(x=x20[i], f)):

od:

> l:=[[x20[k], f20[k]] $k=1..n+1]:

строим полином Лагранжа при n=4

> x0:=0; xn:=2; n:=4; h:=(xn-x0)/n;









точки интерполяции

> xpoints:=vector(n+1):

> for i from 1 to n+1 do xpoints[i]:=x0+h\*(i-1): od:

считаем по схеме Лагранжа

> Lagr4:=vector(n+1):

вычисляем коэффициенты полинома Лагранжа

> for i from 1 to n+1 do

Lagr4[i]:=1:

for j from 1 to i-1 do

Lagr4[i]:=Lagr4[i]\*(x-xpoints[j])/(xpoints[i]-xpoints[j]):

od:

for j from i+1 to n+1 do

Lagr4[i]:=Lagr4[i]\*(x-xpoints[j])/(xpoints[i]-xpoints[j]):

od:

od:

вычисляем полином Лагранжа при n=4

> fLagr4:=evalf(sum(subs(x=xpoints[k], f)\*Lagr4[k], k=1..n+1));



полином Ньютона при n=4

С - матрица разделённых разностей

> C:=matrix(n+1, n+1):

> for i from 1 to n+1 do C[1,i]:=evalf(subs(x=xpoints[i], f)): od:

> for i from 2 to n+1 do

for j from 1 to n+2-i do

C[i,j]:=C[i-1, j+1]-C[i-1, j]:

od:

od:

> print(C);



> Newton4:=sum(C[k,1]/h^(k-1)/factorial(k-1)\*product((x-xpoints[m]), m=1..k-1), k=1..n+1);



>

погрешность

> wn:=product((x-xpoints[k]), k=1..n+1);



> M:=evalf(maximize(abs(diff(f,x,x,x,x,x)), x=0..2));



> evalf(maximize(abs(wn), x=0..2));



> error4 := M\*evalf(maximize(abs(wn), x=0..2))/factorial(n+1);



>

n=8

> n:=8: h:=(xn-x0)/n:

> xpoints:=vector(n+1):

> for i from 1 to n+1 do xpoints[i]:=x0+h\*(i-1): od:

считаем по схеме Лагранжа

> Lagr8:=vector(n+1):

вычисляем коэффициенты полинома Лагранжа

> for i from 1 to n+1 do

Lagr8[i]:=1:

for j from 1 to i-1 do

Lagr8[i]:=Lagr8[i]\*(x-xpoints[j])/(xpoints[i]-xpoints[j]):

od:

for j from i+1 to n+1 do

Lagr8[i]:=Lagr8[i]\*(x-xpoints[j])/(xpoints[i]-xpoints[j]):

od:

od:

> fLagr8:=evalf(sum(subs(x=xpoints[k], f)\*Lagr8[k], k=1..n+1));



полином Ньютона

> C:=matrix(n+1, n+1):

> for i from 1 to n+1 do C[1,i]:=evalf(subs(x=xpoints[i], f)): od:

> for i from 2 to n+1 do

for j from 1 to n+2-i do

C[i,j]:=C[i-1, j+1]-C[i-1, j]:

od:

od:

> Newton8:=sum(C[k,1]/h^(k-1)/factorial(k-1)\*product((x-xpoints[m]), m=1..k-1), k=1..n+1);



>

>

>

>

погрешность

> wn:=product((x-xpoints[k]), k=1..n+1);



> M:=evalf(maximize(abs(diff(f,x,x,x,x,x,x,x,x,x)), x=0..2));

> error8 := M\*evalf(maximize(abs(wn), x=0..2))/factorial(n+1);





фактическая погрешность

> error4exact:=evalf(maximize(abs(f-fLagr4), x=0..2)); error8exact:=evalf(maximize(abs(f-fLagr8), x=0..2));





рисуем

>

> p1:=plot(l, x=x0..xn, color=green, legend="f(x)"):

> p2:=plot(fLagr4, x=x0..xn, color=blue, legend="Lagrange polynomial, n=4"):

> p3:=plot(fLagr8, x=x0..xn, color=red, legend="Lagrange polynomial, n=8"):

> p4:=plot(Newton4, x=x0..xn, color=magenta, legend="Newton polynomial, n=4"):

> p5:=plot(Newton8, x=x0..xn, color=magenta, legend="Newton polynomial, n=8"):

> with(plots):

Warning, the name changecoords has been redefined

> display([p1, p2, p3, p4, p5]);

Задание 4.

> f:=x\*sin(Pi\*x^2/2)+2\*(x-2)^2;



> h:=0.1;



> a:=0; b:=2; n:=trunc((b-a)/h);







метод трапеций

> tr0:=0:

заполняем сетку

> xpoints:=vector(n+1):

> for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

> for i from 1 to n do

tr0:=tr0+evalf((subs(x=xpoints[i], f)+subs(x=xpoints[i+1] ,f))/2)\*h:

od:

> tr0;



>

> h:=h/2; n:=trunc((b-a)/h);





> xpoints:=vector(n+1):for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

> tr:=0:

> for i from 1 to n do

tr:=tr+evalf((subs(x=xpoints[i], f)+subs(x=xpoints[i+1] ,f))/2)\*h:

od:

> tr;



> while abs(tr-tr0)/3>0.0001 do

tr0:=tr:

tr:=0:

h:=h/2; n:=trunc((b-a)/h):

xpoints:=vector(n+1):for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

for i from 1 to n do

tr:=tr+evalf((subs(x=xpoints[i], f)+subs(x=xpoints[i+1] ,f))/2)\*h:

od:

od:

точное значение интеграла

> Ih:=evalf(int(f, x=a..b));



погрешность

> abs(Ih-tr);



уточняем по Ричардсону

> Itr:=1/3\*(4\*tr-tr0);



> abs(Itr-Ih);



формула Симпсона

> h:=0.1;n:=trunc((b-a)/h);





> IS0:=0:

> xpoints:=vector(n+1): for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

> for i from 1 to n do

IS0:=IS0+evalf(h/6\*(subs(x=xpoints[i], f) + 4\*subs(x=(xpoints[i]+xpoints[i+1])/2, f) + subs(x=xpoints[i+1], f)));

od:

> IS0;



> h:=h/2; n:=trunc((b-a)/h);





> IS:=0:

> xpoints:=vector(n+1): for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

> for i from 1 to n do

IS:=IS+evalf(h/6\*(subs(x=xpoints[i], f) + 4\*subs(x=(xpoints[i]+xpoints[i+1])/2, f) + subs(x=xpoints[i+1], f)));

od:

> IS;



> while abs(IS-IS0)/15>0.0001 do

IS0:=IS:

IS:=0:

h:=h/2; n:=trunc((b-a)/h):

xpoints:=vector(n+1):for i from 1 to n+1 do xpoints[i]:=a+(i-1)\*h: od:

for i from 1 to n do

IS:=IS+evalf(h/6\*(subs(x=xpoints[i], f) + 4\*subs(x=(xpoints[i]+xpoints[i+1])/2, f) + subs(x=xpoints[i+1], f)));

od:

od:

> n; IS0; IS; 1/15\*(IS-IS0);









уточнение по Ричардсону

> IS\_h:=1/15\*(16\*IS-IS0);



погрешности

> abs(IS-Ih);abs(IS\_h-Ih);





Задание 5.

> f:=exp(x-1)\*cos(2\*x^2)-2;



> a:=0; b:=3;





> h:=(b-a)/100;



Определим число корней

> num:=0:

> n:=trunc((b-a)/h);



> for i from 1 to n do

if evalf(subs(x=a+h\*(i-1), f)\*subs(x=a+h\*i, f))<0 then num:=num+1: end if:

od:

> num;



> h:=h/2; n:=trunc((b-a)/h);





> num:=0:

for i from 1 to n do

if evalf(subs(x=a+h\*(i-1), f)\*subs(x=a+h\*i, f))<0 then num:=num+1: end if:

od:

num;



> h:=h/2; n:=trunc((b-a)/h);





> num:=0:

for i from 1 to n do

if evalf(subs(x=a+h\*(i-1), f)\*subs(x=a+h\*i, f))<0 then num:=num+1: end if:

od:

num;



> h:=h/2; n:=trunc((b-a)/h);





> num:=0:

for i from 1 to n do

if evalf(subs(x=a+h\*(i-1), f)\*subs(x=a+h\*i, f))<0 then num:=num+1: end if:

od:

# число корней:

num;



z - матрица, содержащая в первом столбце левые границы отрезков, содержащих корень, а во втором столбце - правые

> z:=matrix(num, 2):

> k:=1:

for i from 1 to n do

if evalf(subs(x=a+h\*(i-1), f)\*subs(x=a+h\*i, f))<0 then

z[k, 1]:=evalf(a+h\*(i-1)):

z[k,2]:=evalf(a+h\*i):

k:=k+1:

end if:

od:

отрезки, содержащие корень

> print(z);



корни

> roots\_h:=vector(num):

> for k from 1 to num do

a:=z[k,1]: b:=z[k,2]:

while evalf(abs(subs(x=(a+b)/2, f)))>10^(-7) do

c:=(a+b)/2;

if evalf(subs(x=a, f) \* subs(x=c,f))<0 then b:=c;

else a:=c;

end if;

od;

roots\_h[k]:=(a+b)/2:

od:

> print(roots\_h);



Вычислим значение функции в вычисленных точках

> for i from 1 to num do

print(evalf(subs(x=roots\_h[i],f)));

od;









