# PROIECT NR.3:SERII TAYLOR

A elaborat: Dmitri COJOCARI, gr. IS11Z

A verificat: Vitalie ȚÎCĂU, lector universitar

## 1. Formularea problemei

1. De elaborat un program ce va calcula următoarea serie numerică

1 – x2/2! + x4/4! – x6/6! + … + (–1)nx2n/(2n)! –> cos(x)

## 2. Studiu preliminar

O serie Taylor este o reprezentare a unei funcții ca o sumă infinită de termeni, dată de formula termenului general.

Întreaga serie poate fi percepută ca o progresie cu rația q. Din formula termenului general, putem exprima Tn-1, iar având la dispoziție Tn-1 și Tn, putem calcula rația.

Tn = ; Tn-1 =

q =

Pentru compararea rezultatelor, vom apela la 3 metode de calcul a funcției (directă, utilizând o precizie de calcul, suma parțială). Este important să transformăm valoarea pentru unghi în grade, întrucât funcția predefinită cos ia ca parametru unghiul exprimat în radiani.

## 3. Programul

#include <iostream>

#include <cmath>

#include <math.h>

#include <iomanip>

using **namespace** std;

**double** factorial(**int** n)

{

**double** fact = 1.0;

    for (**int** i = 1; i <= n; i++)

    {

        fact = fact \* i;

    }

    return fact;

}

**int** main()

{

    cout << "Proiect nr.3" << endl;

    cout << "Tema: Serii Taylor" << endl;

    cout << "Elaborat de Cojocari Dmitri" << endl;

**double** x = 65, eps = 1e-9;

**int** n = 10;

*//cout << "Value for degree : ";*

*//cin >> x;*

    x = (x \* M\_PI) / 180;

*//cout << "Input precission : ";*

*//cin >> eps;*

*//cout << "Number of terms : ";*

*//cin >> n;*

**int** exponent, key, i;

**double** res, numarator, numitor, t;

    do

    {

        cout << "\n1. Metoda directa";

        cout << "\n2. Metoda recurenta - precizia de calcul";

        cout << "\n3. Metoda recurenta - suma partiala";

        cout << "\n0. Exit";

        cout << "\n\n\tAlege (de la 0 la 3) : ";

        cin >> key;

        switch (key)

        {

        case 1:

            res = numarator = numitor = 0.0;

            for (**int** i = 0; i < n; i++)

            {

                exponent = 2 \* i;

                numarator = pow(x, exponent);

                numitor = factorial(exponent);

                t = (numarator \* pow(-1, i)) / numitor;

                res = res + t;

                cout << "Numarator: " << numarator << " Numitor: " << numitor << " T" << i << " = " << t << " S" << i << " = " << setprecision(16) << res << endl;

            }

            cout << "Cosinus computed with Taylor series (direct) = " << res << endl;

            break;

        case 2:

**double** s, q;

            i = 0;

            s = t = 1.0;

            while (abs(t) > eps)

            {

                i++;

                q = -(x \* x) / ((2 \* i - 1) \* (2 \* i));

                t = t \* q;

                s += t;

                cout << "T" << i << " = " << t << ". S" << i << " = " << setprecision(16) << s << endl;

            }

            cout << "Cosinus computed with recurrence formula after " << i << " steps = " << setw(16) << setprecision(16) << s << endl;

            cout << "Cosinus (standart function) = " << cos(x) << endl;

            cout << "Error = " << setprecision(3) << abs(s - cos(x)) << endl;

            break;

        case 3:

            s = t = 1.0;

            for (i = 1; i <= n; i++)

            {

                q = -(x \* x) / ((2 \* i - 1) \* (2 \* i));

                t = t \* q;

                s += t;

                cout << "T" << i << " = " << t << ". S" << i << " = " << setprecision(16) << s << endl;

            }

            cout << "Cosinus computed with recurrence formula = " << setw(16) << setprecision(16) << s << endl;

            cout << "Cosinus (standart function) = " << cos(x) << endl;

            cout << "Error = " << setprecision(3) << abs(s - cos(x)) << endl;

            break;

        }

    } while (key);

    return 0;

}

## 4. Rezultatele si analiza lor

## Proiect nr.3

## Tema: Serii Taylor

## Elaborat de Cojocari Dmitri

## 1. Metoda directa

## 2. Metoda recurenta - precizia de calcul

## 3. Metoda recurenta - suma partiala

## 0. Exit

## Alege (de la 0 la 3) : 2

## T1 = -0.643504. S1 = 0.3564957007005781

## T2 = 0.06901629720280666. S2 = 0.4255119979033848

## T3 = -0.00296081893144885. S3 = 0.4225511789719359

## T4 = 6.804641827980198e-005. S4 = 0.4226192253902157

## T5 = -9.730702825550967e-007. S5 = 0.4226182523199332

## T6 = 9.487498641586485e-009. S6 = 0.4226182618074318

## T7 = -6.709061720283877e-011. S7 = 0.4226182617403412

## Cosinus computed with recurrence formula after 7 steps = 0.4226182617403412

## Cosinus (standart function) = 0.4226182617406994

## Error = 3.58e-013

## 1. Metoda directa

## 2. Metoda recurenta - precizia de calcul

## 3. Metoda recurenta - suma partiala

## 0. Exit

## Alege (de la 0 la 3) : 3

## T1 = -0.644. S1 = 0.3564957007005781

## T2 = 0.06901629720280666. S2 = 0.4255119979033848

## T3 = -0.00296081893144885. S3 = 0.4225511789719359

## T4 = 6.804641827980198e-005. S4 = 0.4226192253902157

## T5 = -9.730702825550967e-007. S5 = 0.4226182523199332

## T6 = 9.487498641586485e-009. S6 = 0.4226182618074318

## T7 = -6.709061720283877e-011. S7 = 0.4226182617403412

## T8 = 3.597758384389876e-013. S8 = 0.4226182617407009

## T9 = -1.513184959604854e-015. S9 = 0.4226182617406994

## T10 = 5.124952774426031e-018. S10 = 0.4226182617406994

## Cosinus computed with recurrence formula = 0.4226182617406994

## Cosinus (standart function) = 0.4226182617406994

## Error = 0

## 5. Concluzii

În urma realizării prezentei lucrări, am observat că putem calcula o serie Taylor în 2 moduri(direct și prin sume parțiale). În cazul metodei cu sume parțiale, am interpretat seria ca o progresie cu rația care am determinat-o în studiul preliminar. Pentru verificarea valorilor calculate prin aceste 2 metode, am calculate și valoarea funcției cosinus direct, utilizând funcția matematică corespunzătoare din limbajul C++.