БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа № 1
«Интерполяция алгебраическими многочленами»

Выполнил
Студент 11 группы
Сергиенко Лев Эдуардович

Минск, 2024

Способ выбора узлов

1. Равноотстоящие узлы

$$x_i = a + i\frac{b-a}{n}, i = \overline{0, n}$$

2. Чебышевские узлы

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{\pi(2i+1)}{2n+2}, i = \overline{0, n}$$

Представление, использованное при построении интерполяционных многочленов

- Для построения интерполяционного многочлена Ньютона используется формула:

$$P(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \ldots + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1})$$

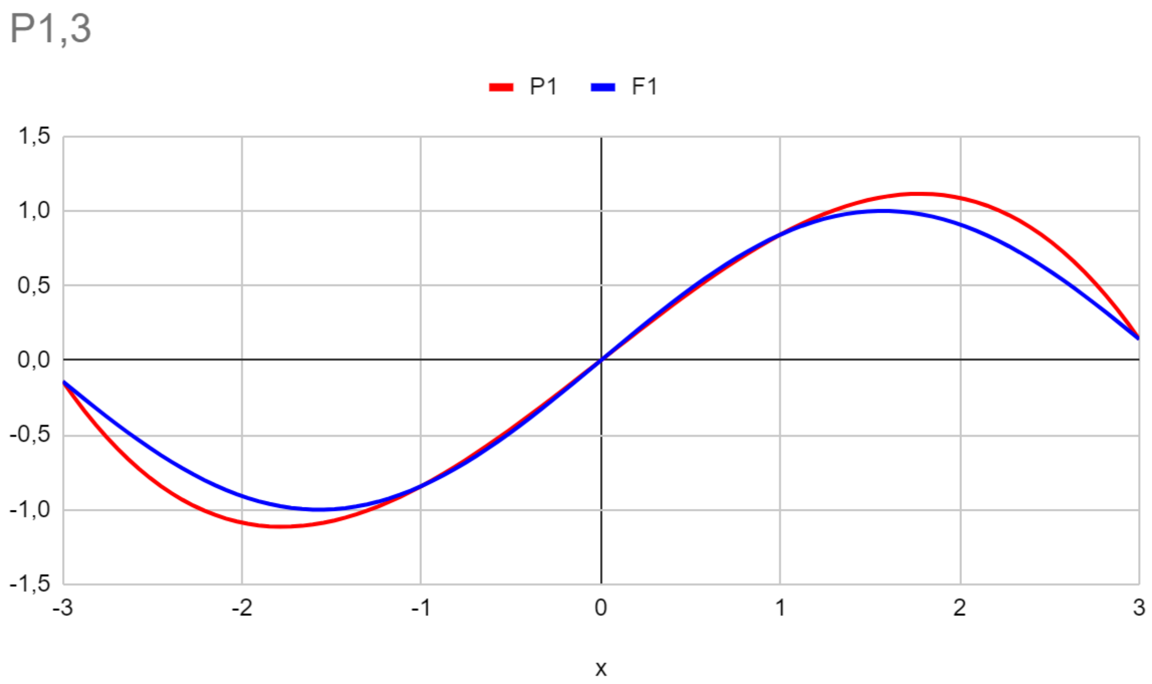
, где f - разделенные разности

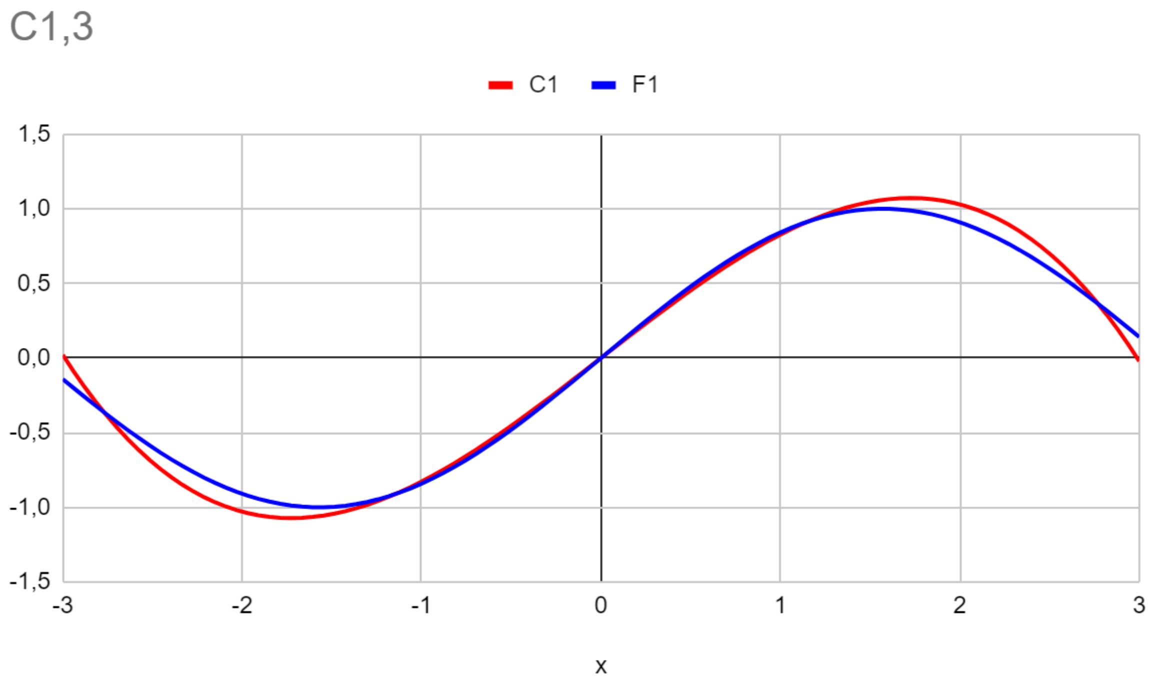
- Разделенные разности удобно вычислять путем построения треугольной таблицы следующего вида:

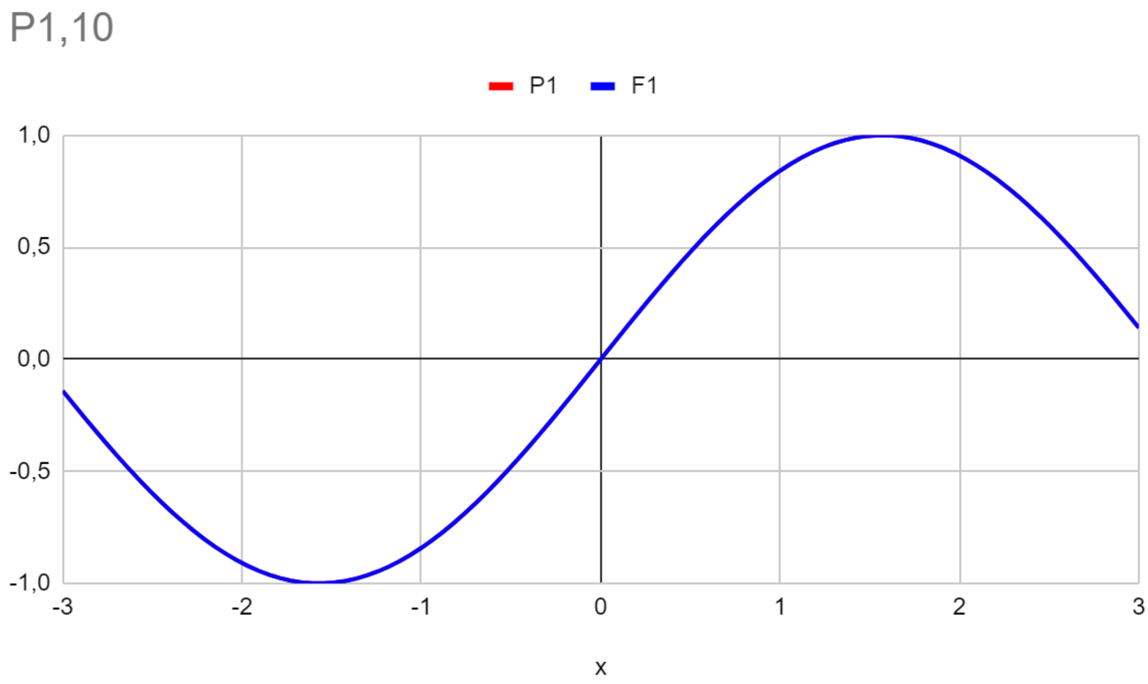
$$\begin{array}{lllllll}
x_0 & f[x_0] & & & & & \\
 & & f[x_0, x_1] & & & & \\
x_1 & f[x_1] & & f[x_0, x_1, x_2] & & & \\
 & & f[x_1, x_2] & & \ldots & & \\
x_2 & f[x_2] & & f[x_1, x_2, x_3] & \ldots & f[x_0, \ldots, x_{n-1}] & \\
 & \cdot & f[x_2, x_3] & \cdot & \ldots & & f[x_0, \ldots, x_n] \\
 & \cdot & \cdot & \cdot & \ldots & f[x_1, \ldots, x_n] & \\
 & \cdot & \cdot & \cdot & \ldots & & \\
x_{n-1} & f[x_{n-1}] & \cdot & f[x_{n-2}, x_{n-1}, x_n] & & & \\
 & & f[x_{n-1}, x_n] & & & & \\
x_n & f[x_n] & & & & &
\end{array}$$

Однако нет необходимости хранить все значения. Разделенные разности для интерполяционного многочлена Ньютона можно посчитать, используя лишь массив размерности n+1.

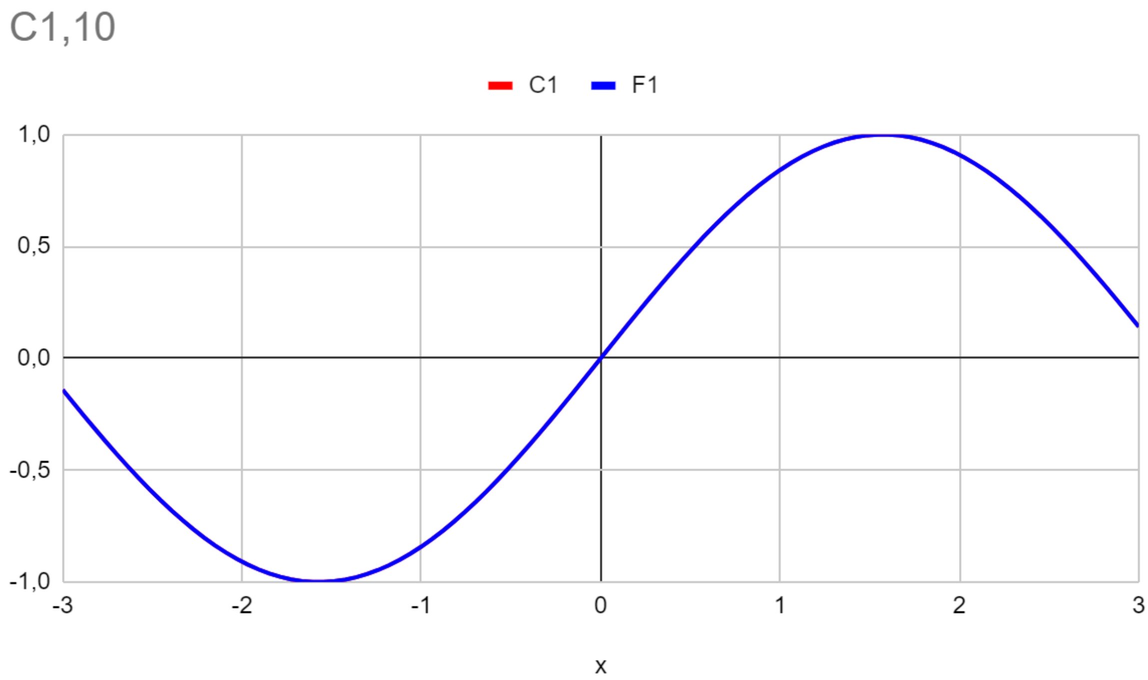
Три пары графиков: график P1,n и график C1,n для n = 3, 10, 30





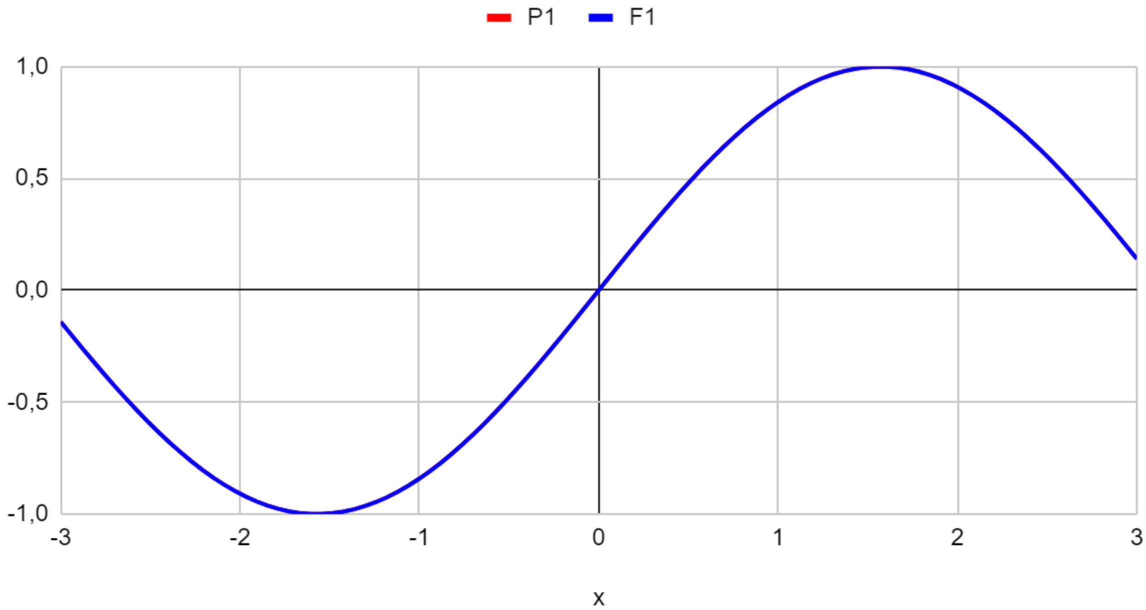


P1,10
P1
F1
1,0
0,5
0,0
-0,5
-1,0
-3
-2
-1
0
1
2
3
x



C1,10
C1
F1
1,0
0,5
0,0
-0,5
-1,0
-3
-2
-1
0
1
2
3
x

P1,30



C1,30

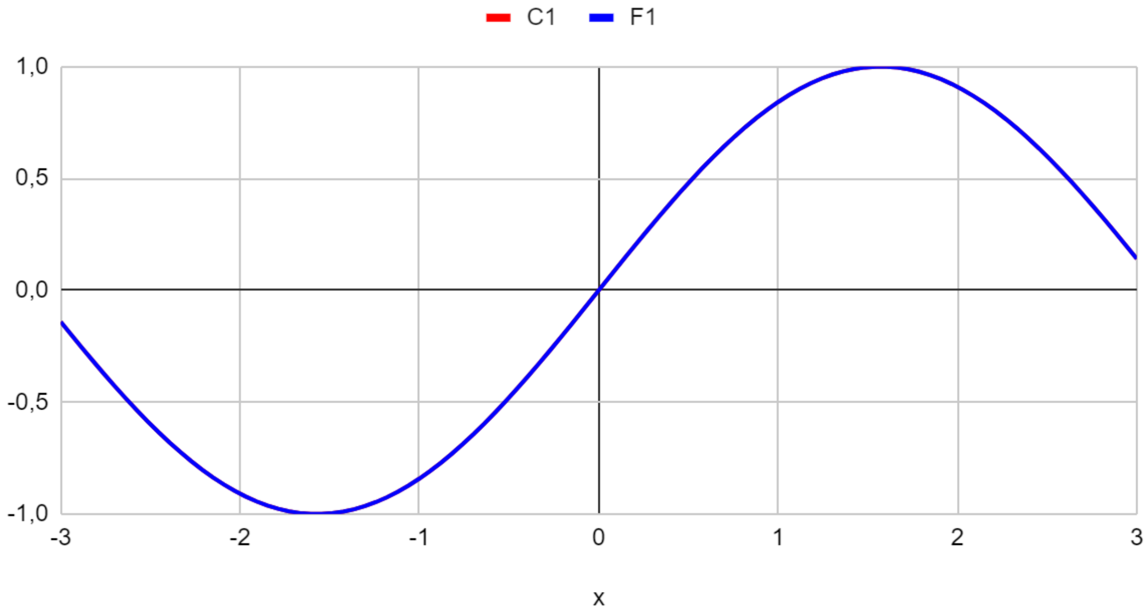
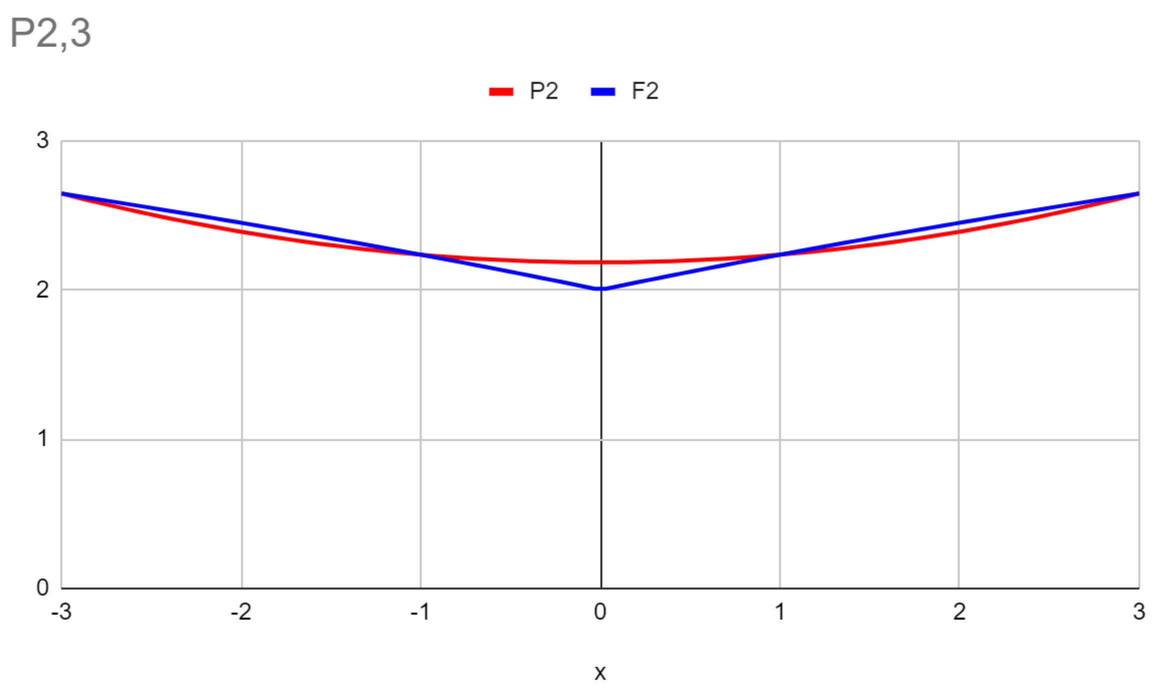


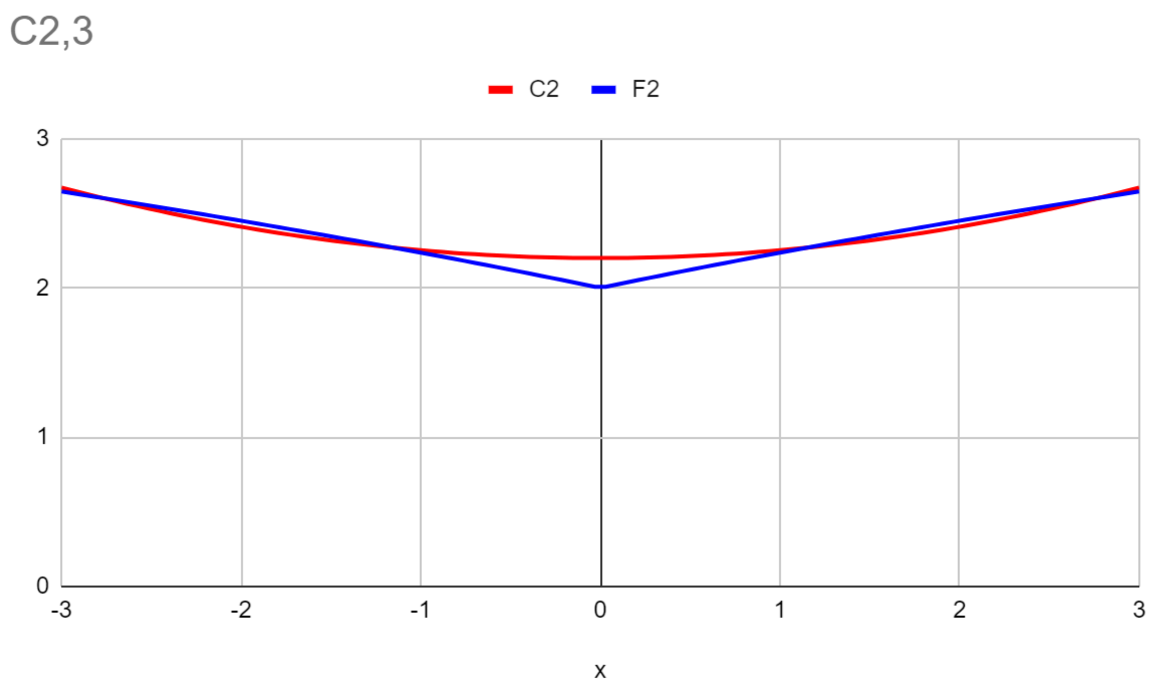
Таблица погрешностей интерполяции для f1

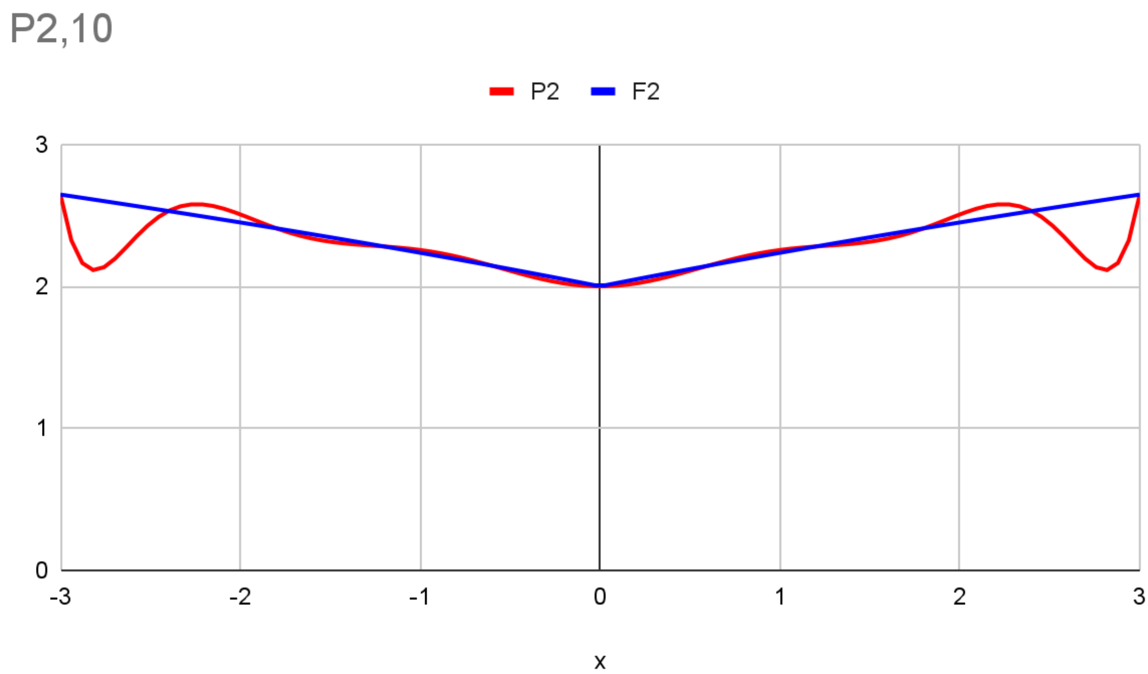
n	$\max\limits_{i=0,100}\left\|P_{1,n}\left(\bar{x}_i\right)-f_1\left(\bar{x}_i\right)\right\|$	$\max\limits_{i=0,100}\left\|C_{1,n}\left(\bar{x}_i\right)-f_1\left(\bar{x}_i\right)\right\|$
3	0.2105284764	0.1622549658
4	0.1497679232	0.0943220846
5	0.0199019363	0.0098586685
6	0.0140370852	0.0053914996
7	0.0011424276	0.0003305093
8	0.0008116830	0.0001755981
9	0.0000442162	0.0000070707
10	0.0000315878	0.0000036711
11	0.0000012084	0.0000001052
12	0.0000008772	0.0000000540
13	0.0000000259	0.0000000012
14	0.0000000186	0.0000000006
15	0.0000000004	0.0000000000
16	0.0000000003	0.0000000000
17	0.0000000000	0.0000000000
18	0.0000000000	0.0000000000
19	0.0000000000	0.0000000000
20	0.0000000000	0.0000000000
21	0.0000000000	0.0000000000
22	0.0000000000	0.0000000000
23	0.0000000000	0.0000000000
24	0.0000000000	0.0000000000
25	0.0000000000	0.0000000000
26	0.0000000000	0.0000000000

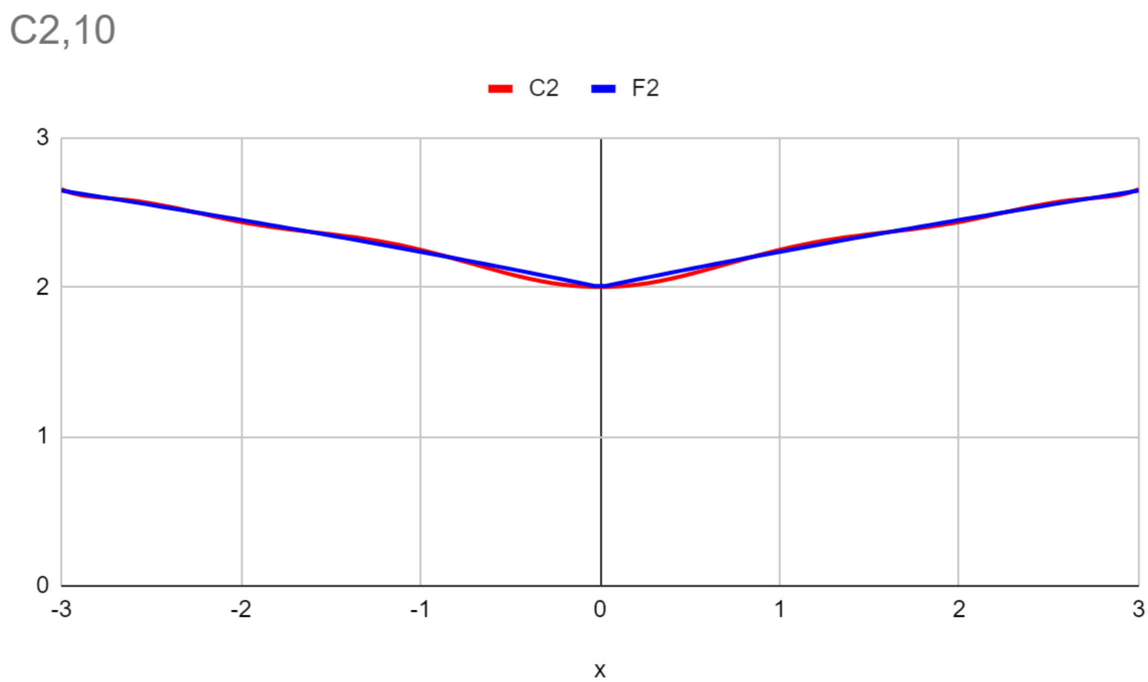
27	0.0000000000	0.0000000000
28	0.0000000000	0.0000000000
29	0.0000000000	0.0000000000
30	0.0000000000	0.0000000000

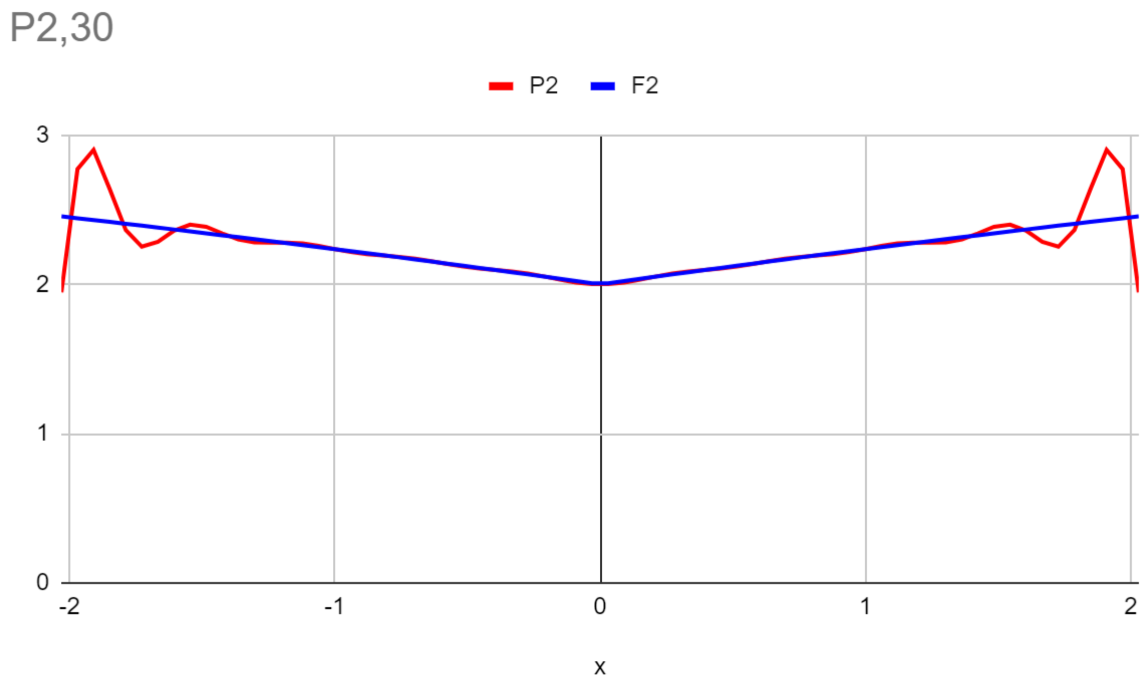
Три пары графиков: график P2,n и график C2,n для n = 3, 10, 30






P2,10
P2
F2
3
2
1
0
-3
-2
-1
0
1
2
3
x


C2,10
C2
F2
3
2
1
0
-3
-2
-1
0
1
2
3
x


P2,30
P2
F2
3
2
1
0
-2
-1
0
1
2
x

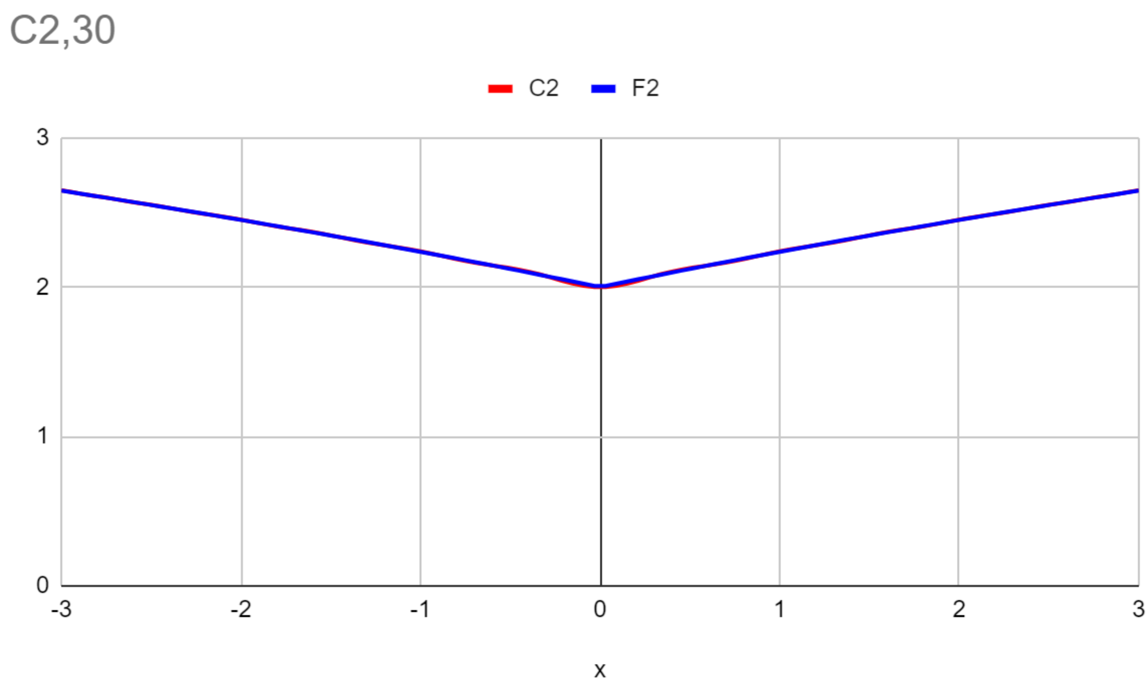

C2,30
C2
F2
3
2
1
0
-3
-2
-1
0
1
2
3
x

Таблица погрешностей интерполяции для f2

n	$\max\limits_{i=0,100}\left\|P_{2,n}\left(\overline{x}_i\right)-f_2\left(\overline{x}_i\right)\right\|$	$\max\limits_{i=0,100}\left\|C_{2,n}\left(\overline{x}_i\right)-f_2\left(\overline{x}_i\right)\right\|$
3	0.1773431223	0.1923882919
4	0.1092645377	0.0919078224
5	0.0975686822	0.1212070582
6	0.1356222356	0.0646748439
7	0.0657009195	0.0878694694
8	0.2361291755	0.0499459691
9	0.0486722162	0.0684000468
10	0.4967403070	0.0408782321
11	0.0807590254	0.0556029498
12	1.1624938940	0.0344364239
13	0.1658212101	0.0465416390
14	3.0419798313	0.0298028293
15	0.3764096666	0.0397870442
16	8.1308040011	0.0263353633
17	0.8868899597	0.0345582999
18	22.5742723974	0.0235411984
19	2.2854533283	0.0303919778
20	69.6506552836	0.0208642223
21	6.3828692760	0.0269953972
22	217.6058150628	0.0193732994
23	18.2081668631	0.0241745045
24	686.6594565122	0.0179270864
25	52.8463071105	0.0217955038
26	2184.2138583529	0.0165213728

27	155.5779281032	0.0197631255
28	6993.1878484552	0.0151584231
29	463.4839295871	0.0180076768
30	22509.1433197204	0.0138403406

Выводы о сходимости интерполяционного процесса по равноотстоящим и чебышевским узлам

Из полученных данных можно сделать следующие выводы относительно сходимости интерполяционного процесса метода Ньютона по равноотстоящим и чебышёвским узлам для функций f1 и f2:

Для функции f1:

Равноотстоящие узлы:

- С увеличением степени интерполяционного многочлена n погрешность уменьшается.
- Погрешность сходится быстро, и уже при n=10 она становится достаточно малой.
- Для n>17 погрешность практически стремится к нулю.

Чебышёвские узлы:

- Погрешность с чебышёвскими узлами сходится ещё быстрее и остается на более низком уровне.
- Уже при n=6 погрешность значительно меньше, чем у равноотстоящих узлов, и продолжает уменьшаться с увеличением n.
- Стремится к нулю с увеличением n.

Для функции f2:

Равноотстоящие узлы:

- С увеличением степени интерполяционного многочлена n погрешность растет.
- Сходимость к нулю отсутствует, и погрешность становится очень высокой даже при небольших значениях n.

Чебышёвские узлы:

- Погрешность также растет с увеличением n, но находится на более низком уровне по сравнению с равноотстоящими узлами.
- Сходимость к нулю отсутствует, но погрешность все равно остается на более низком уровне.

Выводы:

- Для обеих функций использование чебышёвских узлов приводит к более быстрой и стабильной сходимости интерполяционного процесса, чем равноотстоящие узлы.
- При использовании равноотстоящих узлов, сходимость может быть заметно медленнее, и в случае функции f2 даже отсутствовать.
- Использование чебышёвских узлов предпочтительно для достижения более точных результатов интерполяции.

Листинг программы с комментариями

Структура программы:

```
├── internal/
│   ├── interPoly/
│   │   ├── builder.go
│   │   └── interPoly.go
│   └── node/
│       ├── builder.go
│       └── node.go
└── main.go
```

builder.go

```
package interPoly

import (
  "mv2.1/internal/node"
)

// NewtonPolyBuilder строит полином Ньютона на основе заданных узлов интерполяции
// nodes - узлы для интерполяции
// Возвращает полином Ньютона, представленный в виде структуры NewtonPoly
func NewtonPolyBuilder(nodes []node.Node) NewtonPoly {
  // Инициализация полинома Ньютона
  poly := NewtonPoly{
    separatedDif: make([]float64, len(nodes)),
    nodes:        make([]node.Node, len(nodes)),
  }
  // Копирование узлов
  copy(poly.nodes, nodes)

  // Инициализация первых разделенных разностей значениями Y из узлов
  for i := 0; i < len(poly.separatedDif); i++ {
    poly.separatedDif[i] = poly.nodes[i].Y
  }

  // Вычисление разделенных разностей
```

```
  for j := 1; j < len(poly.separatedDif); j++ {
    for i := len(poly.separatedDif) - 1; i >= j; i-- {
      poly.separatedDif[i] = (poly.separatedDif[i] - poly.separatedDif[i-1]) /
(poly.nodes[i].X - poly.nodes[i-j].X)
    }
  }

  // Возвращение построенного полинома Ньютона
  return poly
}
```

interPoly.go

```
package interPoly

import (
  "fmt"
  "mv2.1/internal/node"
)

// NewtonPoly представляет интерполяционный полином Ньютона.
type NewtonPoly struct {
  nodes       []node.Node // Узлы интерполяции
  separatedDif []float64   // Разделенные разности
}

// Solve вычисляет значение интерполяционного полинома Ньютона в точке x.
func (n NewtonPoly) Solve(x float64) float64 {
  result := n.separatedDif[0]
  temp := 1.0

  // Вычисляем значение интерполяционного полинома в точке x
  for i := 1; i < len(n.nodes); i++ {
    temp *= x - n.nodes[i-1].X
    result += n.separatedDif[i] * temp
  }

  return result
}

// String возвращает строковое представление интерполяционного полинома
Ньютона.
func (n NewtonPoly) String() string {
  polyString := fmt.Sprintf("%.3f", n.separatedDif[0])
```

```
  // Формируем строку полинома с использованием разделенных разностей
и узлов интерполяции
  for i := 1; i < len(n.separatedDif); i++ {
    polyString += " + "

    polyString += fmt.Sprintf("%.3f", n.separatedDif[i])

    for j := 0; j < i; j++ {
      polyString += fmt.Sprintf(" * (x - %.3f)", n.nodes[j].X)
    }
  }

  return polyString
}
```

builder.go

```
package node

import (
  "math"
)

// BuildEquidistantNodes строит равноотстоящие узлы
// f - интерполируемая функция, a и b - границы интервала, n - степнь
многочлена
// Возвращает массив узлов Node, представляющих точки интерполяции
func BuildEquidistantNodes(f func(float64) float64, a float64, b float64, n int)
[]Node {
  // Создаем слайс для хранения узлов
  nodes := make([]Node, n+1)
  // Вычисляем шаг между узлами
  h := (b - a) / float64(n)

  // Заполняем массив узлами, где X - равномерно распределенные точки, Y -
значение функции в этих точках
  for i := 0; i <= n; i++ {
    x := a + float64(i)*h
    nodes[i] = Node{X: x, Y: f(x)}
  }

  // Возвращаем массив узлов для использования при интерполяции
  return nodes
}

// BuildChebyshevNodes строит чебышѐвские узлы
// Возвращает массив узлов Node, представляющих точки интерполяции
```

```go
func BuildChebyshevNodes(f func(float64) float64, a float64, b float64, n int)
[]Node {
  // Создаем слайс для хранения узлов
  nodes := make([]Node, n+1)

  // Заполняем массив узлами, где X - точки, определенные методом
Чебышева
  for i := 0; i <= n; i++ {
    x := (a+b)/2 + (b-a)/2*math.Cos(math.Pi*(2*float64(i)+1)/(2*float64(n)+2))
    nodes[i] = Node{X: x, Y: f(x)}
  }

  return nodes
}
```

node.go

```go
package node

// Node представляет узел для интерполяции в двумерном пространстве.
type Node struct {
  X float64 // Координата X узла в пространстве для интерполяции.
  Y float64 // Координата Y узла в пространстве для интерполяции.
}
```

main.go

```go
package main

import (
  "fmt"
  "math"
  "mv2.1/internal/interPoly"
  "mv2.1/internal/node"
  "os"
)

const (
  a       = -3.0
  b       = 3.0
  nPoints = 100
)

var (
  f1 = func(x float64) float64 {
    return math.Sin(x)
  }

  f2 = func(x float64) float64 {
```

```
    return math.Sqrt(4 + math.Abs(x))
  }
)

func main() {
  var n int

  fmt.Println("Input n: ")
  _, _ = fmt.Scan(&n)

  var (
    // P1 Построение полинома Ньютона для f1 с равноотстоящими узлами
    P1 = interPoly.NewtonPolyBuilder(node.BuildEquidistantNodes(f1, a, b, n))
    // C1 Построение полинома Ньютона для f1 с Чебышевскими узлами
    C1 = interPoly.NewtonPolyBuilder(node.BuildChebyshevNodes(f1, a, b, n))
    // P2 Построение полинома Ньютона для f1 с равноотстоящими узлами
    P2 = interPoly.NewtonPolyBuilder(node.BuildEquidistantNodes(f2, a, b, n))
    // C2 Построение полинома Ньютона для f1 с Чебышевскими узлами
    C2 = interPoly.NewtonPolyBuilder(node.BuildChebyshevNodes(f2, a, b, n))
  )

  // Вывод полиномов Ньютона в консоль
  fmt.Println("P1: ", P1)
  fmt.Println("C1: ", C1)
  fmt.Println()
  fmt.Println("P2: ", P2)
  fmt.Println("C2: ", C2)

  // Сохранение результатов в файлы и вычисление погрешности
интерполяции
  saveToFile(fmt.Sprintf("%s_%d_%d.txt", "P1", n, nPoints), P1, f1)
  saveToFile(fmt.Sprintf("%s_%d_%d.txt", "C1", n, nPoints), C1, f1)
  saveToFile(fmt.Sprintf("%s_%d_%d.txt", "P2", n, nPoints), P2, f2)
  saveToFile(fmt.Sprintf("%s_%d_%d.txt", "C2", n, nPoints), C2, f2)
}

// saveToFile сохраняет результаты интерполяции в файл и вычисляет
погрешность
func saveToFile(filename string, poly interPoly.NewtonPoly, f func(float64) float64)
{
  file, err := os.Create(filename)
  if err != nil {
    fmt.Println("Error creating file:", err)
    return
  }
```

```go
    defer file.Close()

    // Вычисление шага для точек интерполяции
    step := (b - a) / float64(nPoints-1)
    interErr := 0.0

    // Запись точек интерполяции и реальных значений функции в файл
    for i := 0; i < nPoints; i++ {
        x := a + float64(i)*step
        y := poly.Solve(x)
        yReal := f(x)

        // Обновление максимальной погрешности
        interErr = math.Max(interErr, math.Abs(yReal-y))

        //Запись в файл в формате "x y интерполяция y реальное значение"
        _, err := file.WriteString(fmt.Sprintf("%.10f %.10f %.10f\n",x, y, yReal))
        if err != nil {
            fmt.Println("Error writing to file:", err)
            return
        }
    }

    //Запись погрешности интерполяции в файл
    _, err = file.WriteString(fmt.Sprintf("%s %.10f", "", interErr))
    if err != nil {
        fmt.Println("Error writing to file:", err)
        return
    }
}
```