Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 5

Тема Численное интегрирование

Студент Кононенко С.С.

Группа ИУ7-43Б

Оценка (баллы) ____________

Преподаватель Градов В.М.

Москва — 2020 г.

1. Задание

Построить алгоритм и программу для вычисления двукратного интеграла при фиксированном значении параметра τ

$$\epsilon(\tau) = \frac{4}{\pi} \int_0^{\frac{\pi}{2}} d\phi \int_0^{\frac{\pi}{2}} [1 - \exp(-\tau \frac{1}{R})] \; cos\theta \; sin\theta \; d\theta \; d\varphi$$

где $\frac{1}{R} = \frac{2cos\theta}{1 - cos^2\theta sin^2\varphi}$, θ, φ - углы сферических координат.

Применить метод последовательного интегрирования. По одному направлению использовать формулу Гаусса, а по другому – формулу Симпсона.
Ввод: количество узлов сетки N, M; значение параметра τ, методы для направлений при последовательном интегрировании.
Вывод: значение интеграла при заданном параметре, график зависимости $\epsilon(\tau)$ в диапазоне $\tau = 0.05 - 10$.

2. Описание алгоритма

Имеем $\int_{-1}^{1} f(t)dt = \sum_{i=1}^{n} A_i f(t_i)$, положим $\int_{-1}^{1} t^k dt = \sum_{i=1}^{n} A_i f(t_i^k), \; k = 0, 1, 2, ..., 2n - 1$

Имеем систему:

$$\begin{cases} \sum_{i=1}^{n} A_i = 2 \\ \sum_{i=1}^{n} A_i t_i = 0 \\ \sum_{i=1}^{n} A_i t_i^2 = \frac{2}{3} \\ \dots \\ \sum_{i=1}^{n} A_i t_i^{2n-1} = 0 \end{cases}$$

Системая нелинейная, найти решение сложно. Для нахождения A_i и t_i можно воспользоваться полиномом Лежандра. Формула полинома:

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}[(x^2 - 1)^n], \; n = 0, 1, 2$$

Узлами формулы Гаусса являются нули полинома Лежандра $P_n(t)$, а A_i можно найти из вышеуказанной системы уравнений.

При вычислении интеграла на произвольном интервале $[a, b]$, для применения квадратурной формулы Гаусса необходимо выполнить преобразование переменной:

$$x = \frac{b + a}{2} + \frac{b - a}{2} t$$

В таком случае, получаем конечную формулу для произвольного интервала $[a, b]$:

$$\int_a^b f(x)dx = \frac{b - a}{2} \sum_{i=1}^{n} A_i f(x_i)$$

Так же, существует квадратнурная формула Симпсона:

$$\int_a^b f(x)dx \approx \frac{h}{3} \sum_{i=0}^{\frac{N}{2}-1} (f_{2i} + 4f_{2i+1} + f_{2i+2})$$

Однако, эти методы можно применять и для приближенной оценки двукратных (и не только) интегралов. Рассмотрим интеграл по прямоугольной области:

$$I = \int_c^d \int_a^b f(x,y)dx\ dy = \int_a^b F(x)dx, \quad \text{где } F(x) = \int_c^d f(x,y)dy$$

По каждой координате введем сетку узлов. Каждый однократный интеграл вычисляют по квадратурным формулам. Для разных направлений можно использовать квадратурные формулы разных порядков точности, в т.ч. и Гаусса.

Конечная формула:

$$I = \int\int_G f(x,y)dx\ dy = \sum_{i=1}^{n} \sum_{j=1}^{m} A_i B_{ij} f(x_i, y_j)$$

где $A_i B_{ij}$ – известные постоянные.

3. Результаты работы программы

3.1. Алгоритм вычисления n корней полинома Лежандра n-ой степени.

Все корни полинома лежат на интервале [-1, 1]. При этом стоит заметить, что интервалы [-1, 0] и [0, 1] – симметричны, так что при поиске корней достаточно рассмотреть интервал [0, 1].

Корни полинома можно вычислить итеративно по методу Ньютона

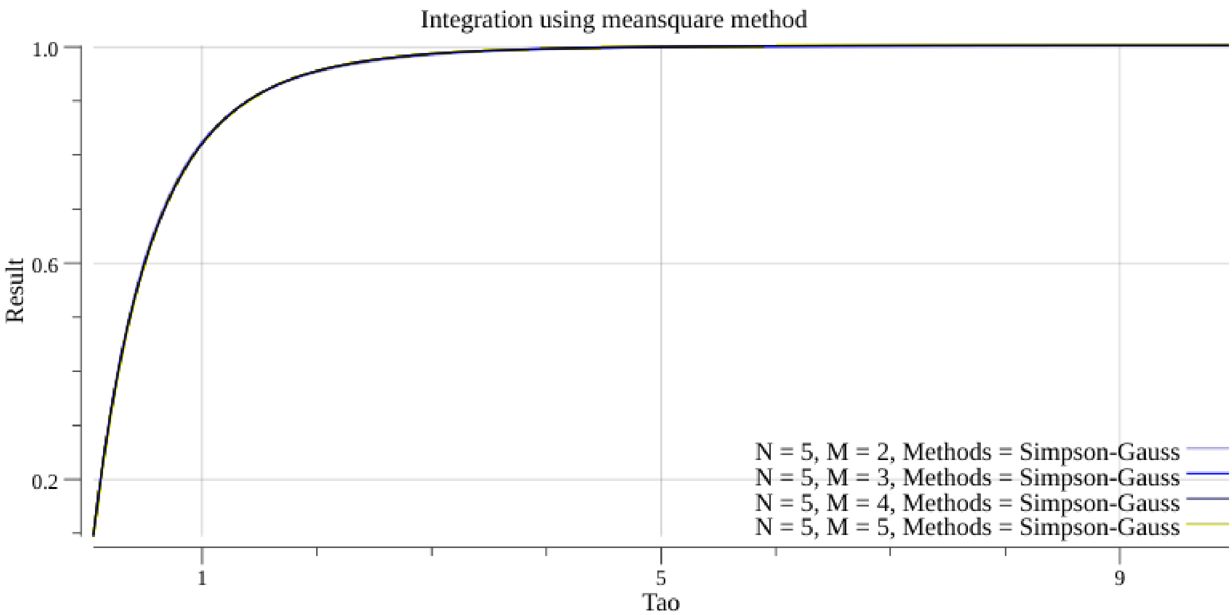
$$x_i^{(k+1)} = x_i^k - \frac{P_n(x_i)^{(k)}}{P_n'(x_i)^{(k)}}$$

причем начальное приближение для i-го корня берется по формуле:

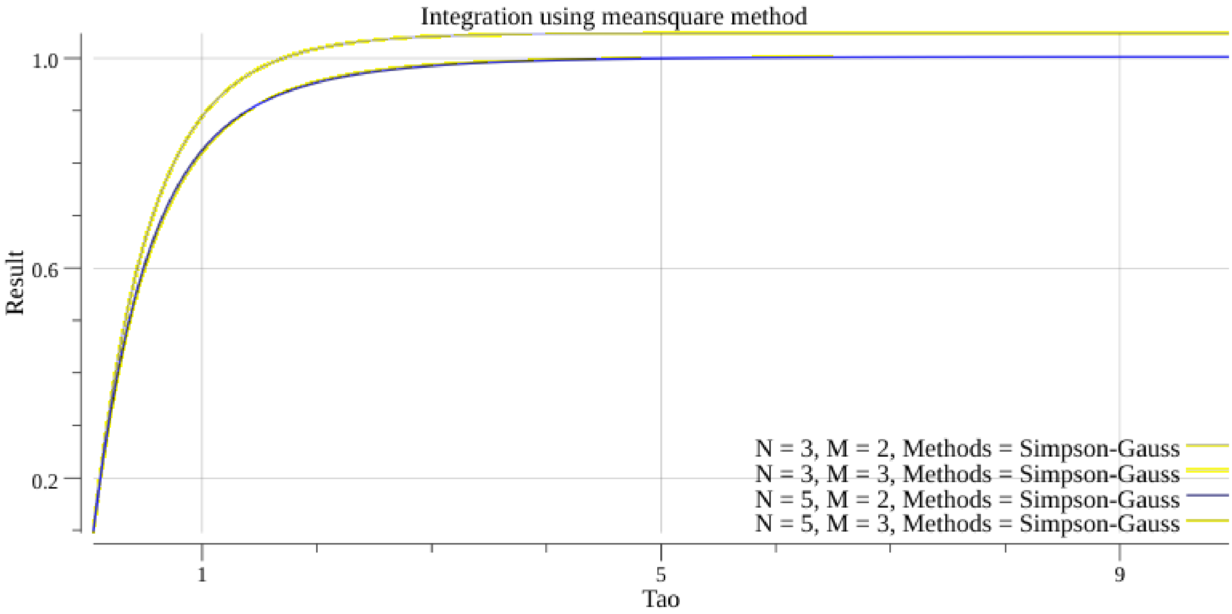
$$x_i^{(0)} = cos[\frac{\pi(4i-1)}{4n+2}]$$

3.2. Влияние количества выбираемых узлов сетки по каждому направлению на точность расчетов.

При использовании метода Симпсона в качестве «внешнего» метода интегрирования и при задании для него 5 узлов, метод Гаусса с различным количеством узлов будет давать одни и те же результаты.

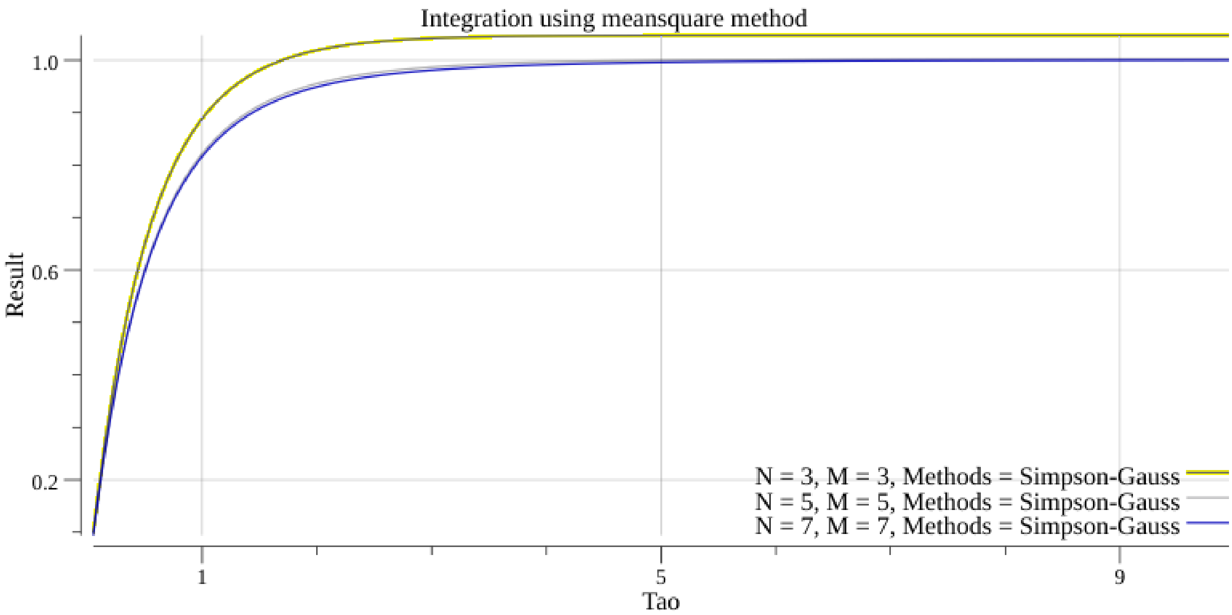


Если для метода Симпсона задать меньшее количество узлов, получится расхождение с физическим смыслом - больший вклад будет вносить метод, являющийся «внешним».



Все измерения проводились для параметра $\tau = 1$.

3.3. График зависимости $\epsilon(\tau)$.



Все измерения проводились для параметра $\tau = 1$ Как видно из графика, оптимальное значение достигнуто на сетке 5×5.

4. Ответы на вопросы для защиты лабораторной работы

1. В каких ситуациях теоретический порядок квадратурных формул численного интегрирования не достигается?

Теоретический порядок квадратурных формул численного интегрирования не достигается в ситуациях, когда подынтегральная функция не имеет соответствующих производных. Порядок точности равен номеру последней существующей производной.

2. Построить формулу Гаусса численного интегрирования при одном узле.

$$\sum_{i=1}^{n} A_i = 2 \;\; P_1(x) = x \Rightarrow x = 0$$

Полученная формула:

$$\int_a^b f(x)dx = \frac{b-a}{2} 2f(\frac{b+a}{2} + \frac{b-a}{2} \cdot 0) = (b-a)f(\frac{b+a}{2})$$

3. Построить формулу Гаусса численного интегрирования при двух узлах.

$$P_2(x) = \frac{1}{2}(3x^2 - 1) \;\Rightarrow x = \pm\frac{1}{\sqrt{3}}$$

$$\begin{cases} A_1 + A_2 = 2 \\ -\frac{1}{\sqrt{3}}A_1 + \frac{1}{\sqrt{3}}A_2 = 0 \end{cases} \Rightarrow A_2 = A_1 = 1$$

$$\int_{-1}^{1} f(f)df = f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$$

Полученная формула:

$$\int_a^b f(x)dx = \frac{b-a}{2}(f(\frac{b+a}{2} - \frac{b-a}{2} \cdot \frac{1}{\sqrt{3}}) + f(\frac{b+a}{2} + \frac{b-a}{2} \cdot \frac{1}{\sqrt{3}}))$$

4. Получить обобщенную кубатурную формулу, на основе методе трапеций, с тремя узлами на каждом направлении.

Полученная формула:

$\int_c^d \int_a^b f(x,y)dxdy = \int_a^b dx \int_c^d f(x,y)dy = \int_a^b F(x)dx = h_x(\frac{1}{2}F_0 + F_1 + \frac{1}{2}F_2) = h_x h_y(\frac{1}{2}(\frac{1}{2}f(x_0,y_0) + f(x_0,y_1) + \frac{1}{2}f(x_0,y_2)) + \frac{1}{2}f(x_1,y_0) + f(x_1,y_1) + \frac{1}{2}f(x_1,y_2) + \frac{1}{2}(\frac{1}{2}f(x_2,y_0) + f(x_2,y_1) + \frac{1}{2}f(x_2,y_2)))$

где $h_x = \frac{b-a}{2}, \;\; h_y = \frac{d-c}{2}$

5. Код программы

5.1. Основной пакет приложения.

```go
package main

import (
  "fmt"
  "math"

  "./integrate"
)

func main() {
  var (
    n, m, md1, md2, end int
    p                   float64
    f1, f2              integrate.Integrator
    sc                  []float64
  )

  sc = []float64{0.05, 0.05, 10}
  end = 0
  pl := integrate.CreatePlot("Integration using meansquare method", "Tao", "Result")
  for end != 1 {
    fmt.Printf("Enter N: ")
    fmt.Scan(&n)
    fmt.Printf("Enter M: ")
    fmt.Scan(&m)
    fmt.Printf("Enter parameter (tao): ")
    fmt.Scan(&p)
    fmt.Printf("Choose mode for outer integration (0 - Gauss, 1 - Simpson): ")
    fmt.Scan(&md1)
    if md1 == 0 {
      f1 = integrate.Gauss
    } else {
      f1 = integrate.Simpson
    }
    fmt.Printf("Choose mode for inner integration (0 - Gauss, 1 - Simpson): ")
    fmt.Scan(&md2)
    if md2 == 0 {
      f2 = integrate.Gauss
    } else {
      f2 = integrate.Simpson
    }

    lm := [][]float64{{0, math.Pi / 2}, {0, math.Pi / 2}}
    ns := []int{n, m}
    igs := []integrate.Integrator{f1, f2}
```

```
    pint := func(p float64) float64 {
      return integrate.Integrate(integrate.IntegratedFunc(p), lm, ns, igs)
    }

    fmt.Printf("Result with %.2f as a parameter is %.7f\n", p, pint(p))
    ds := integrate.GenDots(pint, sc)
    integrate.DrawPlot(pl, ds, n, m, md1, md2)
    fmt.Printf("Stop execution?: ")
    fmt.Scan(&end)
  }
  integrate.SavePlot(pl, "points.png")
}
```

5.2. Пакет реализации численного интегрирования.

5.2.1. Реализация численного интегрирования.

```
package integrate

import (
  "math"

  "gonum.org/v1/gonum/integrate/quad"
)

// Integrator type used to represent integrator function.
type Integrator func(func(float64) float64, float64, float64, int) float64

// Integrated type used to represent integrated function.
type Integrated func(float64, float64) float64

// Simpson function used to integrate with Simpson method.
func Simpson(f func(float64) float64, a, b float64, n int) float64 {
  if n < 3 || n&1 == 0 {
    panic("Error")
  }

  h := (b - a) / float64(n-1)
  x := a
  res := 0.0

  for i := 0; i < (n-1)/2; i++ {
    res += f(x) + 4*f(x+h) + f(x+2*h)
    x += 2 * h
  }

  return res * (h / 3)
}

// Gauss function used to integrate with Gauss-Legendre quadrature.
func Gauss(f func(float64) float64, a, b float64, n int) float64 {
  var (
    x, weight []float64
    l         quad.Legendre
    res       float64
  )
  x = make([]float64, n)
  weight = make([]float64, n)

  l.FixedLocations(x, weight, -1, 1)
  res = 0.0
```

```go
  for i := 0; i < n; i++ {
    res += (b - a) / 2 * weight[i] * f(pToV(x[i], a, b))
  }

  return res
}

// IntegratedFunc function used to choose inner and outer direction of integration.
func IntegratedFunc(p float64) Integrated {
  sf := func(x, y float64) float64 {
    return 2 * math.Cos(x) / (1 - math.Pow(math.Sin(x), 2)*math.Pow(math.Cos(y), 2))
  }

  f := func(x, y float64) float64 {
    return 4 / math.Pi * (1 - math.Exp(-p*sf(x, y))) * math.Cos(x) * math.Sin(x)
  }

  return f
}

// Integrate function used to split integration directions.
func Integrate(f Integrated, lm [][]float64, n []int, fn []Integrator) float64 {
  inner := func(x float64) float64 {
    return fn[1](fWrap(f, x), lm[1][0], lm[1][1], n[1])
  }
  return fn[0](inner, lm[0][0], lm[0][1], n[0])
}

func fWrap(f func(float64, float64) float64, val float64) func(float64) float64 {
  return func(val1 float64) float64 {
    return f(val, val1)
  }
}

func pToV(p, a, b float64) float64 {
  return (b+a)/2 + (b-a)*p/2
}
```

5.2.2. Отрисовка графика.

```go
package integrate

import (
  "fmt"
  "image/color"
  "math/rand"
  "os"
  "time"
```

```go
	"gonum.org/v1/plot"
	"gonum.org/v1/plot/plotter"
	"gonum.org/v1/plot/vg"
)

func genLabel(n, m, md1, md2 int) string {
	var f1s, f2s string

	if md1 == 0 {
		f1s = "Gauss"
	} else {
		f1s = "Simpson"
	}

	if md2 == 0 {
		f2s = "Gauss"
	} else {
		f2s = "Simpson"
	}

	return fmt.Sprintf("N = %v, M = %v, Methods = %v-%v", n, m, f1s, f2s)
}

func genRandomNumber(min, max int) int {
	rand.Seed(time.Now().UnixNano())
	return rand.Intn(max-min) + min
}

// GenDots used to generate dots for plot representation.
func GenDots(f func(p float64) float64, sc []float64) plotter.XYs {
	ds := plotter.XYs{}
	for i := sc[0]; i < sc[2]; i += sc[1] {
		d := plotter.XY{
			X: i,
			Y: f(i),
		}
		ds = append(ds, d)
	}

	return ds
}

// CreatePlot used to create base plot object.
func CreatePlot(title, x, y string) *plot.Plot {
	pl, err := plot.New()
	if err != nil {
		fmt.Println("Error:", err)
		os.Exit(1)
```

```
  }
  pl.Title.Text = title
  pl.X.Label.Text = x
  pl.Y.Label.Text = y
  pl.Add(plotter.NewGrid())

  return pl
}

// DrawPlot used to draw plot with given dots.
func DrawPlot(p *plot.Plot, ds plotter.XYs, n, m, md1, md2 int) {

  l, err := plotter.NewLine(ds)
  if err != nil {
    fmt.Println("Error:", err)
    os.Exit(1)
  }
  l.LineStyle.Width = vg.Points(1)
  l.LineStyle.Color = color.RGBA{B: uint8(genRandomNumber(0, 255)), A: uint8(genRandomNu

  p.Add(l)
  p.Legend.Add(genLabel(n, m, md1, md2), l)
}

// SavePlot used to save gonum plot to file.
func SavePlot(p *plot.Plot, f string) {
  if err := p.Save(8*vg.Inch, 4*vg.Inch, f); err != nil {
    panic(err)
  }
}
```