СРСП №3

Вычисление погрешности результатов арифметических действий

Бактыбеков Нурсултан

20 Сентября, 2023

Содержание

Цель работы

Научиться выполнять вычисления с приближенными числами с учетом погрешностей.

Задание

- Усвоить понятия абсолютной и относительной погрешности и их границ;
- Научиться определять верные, сомнительные, значащие цифры в приближенном числе и определять по ним погрешности приближений;
- Научиться находить погрешности вычислений;

Выполнение заданий

Задание 1

Условие

Приближенные числа записаны с указанием их абсолютных погрешностей. Оставьте в их записи только верные цифры:

а. $63,2561 \pm 0,001$

б. $63,2561 \pm 0,002$

в. $2,53 \pm 0,00001$

г. $2,53 \pm 0,00004$

д. $42753,8 \pm 800$

е. $42753,8 \pm 100$

ж. $42153,8 \pm 800$

з. $42753,8 \pm 100$

Есть ли среди полученных чисел такие, все цифры которых верны в строгом смысле?

Решение

```go
package main

import (
  "fmt"
)

func main() {
  // Define the given approximate numbers and their error limits
  numbers := []struct {
    value float64
    error float64
  }{
    {42753.8, 800},
    {63.2561, 0.001},
    {63.2561, 0.002},
```

```
    {42753.8, 100},
    {2.53, 0.00001},
    {42153.8, 800},
    {2.53, 0.00004},
    {42743.8, 100},
  }

  // Iterate through the numbers and calculate absolute and relative errors
  for i, num := range numbers {
    absoluteError := num.error
    relativeError := (num.error / num.value) * 100

    // Round the relative error to a reasonable number of decimal places
    roundedRelativeError := round(relativeError, 4)

    fmt.Printf("For approximate number %d:\n", i+1)
    fmt.Printf("Value: %.4f\n", num.value)
    fmt.Printf("Absolute Error: %.4f\n", absoluteError)
    fmt.Printf("Relative Error: %.4f%%\n", roundedRelativeError)
    fmt.Println()
  }
}

// Round a float64 number to the specified number of decimal places
func round(num float64, decimalPlaces int) float64 {
  rounding := 1.0
  for i := 0; i < decimalPlaces; i++ {
    rounding *= 10.0
  }
  return float64(int((num*rounding)+0.5)) / rounding
}
```

Результат выполнения программы

```
For approximate number 1:
Value: 42753.8000
Absolute Error: 800.0000
Relative Error: 1.8712%

For approximate number 2:
Value: 63.2561
Absolute Error: 0.0010
Relative Error: 0.0016%

For approximate number 3:
Value: 63.2561
Absolute Error: 0.0020
Relative Error: 0.0032%

For approximate number 4:
```

```
Value: 42753.8000
Absolute Error: 100.0000
Relative Error: 0.2339%

For approximate number 5:
Value: 2.5300
Absolute Error: 0.0000
Relative Error: 0.0004%

For approximate number 6:
Value: 42153.8000
Absolute Error: 800.0000
Relative Error: 1.8978%

For approximate number 7:
Value: 2.5300
Absolute Error: 0.0000
Relative Error: 0.0016%

For approximate number 8:
Value: 42743.8000
Absolute Error: 100.0000
Relative Error: 0.2340%
```

Задание 2

Условие

Округлите соответственно до двух, трех и четырех знаков после запятой следующие числа:

- 3,009983
- 24,00551
- 21,161728

Решение

```go
package main

import (
  "fmt"
  "math"
)

func main() {
  numbers := []float64{3.009983, 24.00551, 21.161728}

  for _, num := range numbers {
    roundedTwoDecimals := round(num, 2)
    roundedThreeDecimals := round(num, 3)
    roundedFourDecimals := round(num, 4)
```

```
    fmt.Printf("Original Number: %.6f\n", num)
    fmt.Printf("Rounded to 2 Decimals: %.2f\n", roundedTwoDecimals)
    fmt.Printf("Rounded to 3 Decimals: %.3f\n", roundedThreeDecimals)
    fmt.Printf("Rounded to 4 Decimals: %.4f\n", roundedFourDecimals)
    fmt.Println()
  }
}

// Round a float64 number to the specified number of decimal places
func round(num float64, decimalPlaces int) float64 {
  shift := math.Pow(10, float64(decimalPlaces))
  return math.Round(num*shift) / shift
}
```

Результат выполнения программы

```
Original Number: 3.009983
Rounded to 2 Decimals: 3.01
Rounded to 3 Decimals: 3.010
Rounded to 4 Decimals: 3.0100

Original Number: 24.005510
Rounded to 2 Decimals: 24.01
Rounded to 3 Decimals: 24.006
Rounded to 4 Decimals: 24.0055

Original Number: 21.161728
Rounded to 2 Decimals: 21.16
Rounded to 3 Decimals: 21.162
Rounded to 4 Decimals: 21.1617
```

Задание 3

Условие

У приближенных чисел 36,7; 2,489; 31,010; 0,031 все цифры верны

а) в широком смысле;

б) в строгом смысле.;

Определите границы абсолютных и относительных погрешностей этих чисел;

Решение

```
package main

import (
  "fmt"
)

func main() {
  numbers := []float64{36.7, 2.489, 31.010, 0.031}

  for _, num := range numbers {
    // Calculate the limits of absolute errors
     absoluteErrorBroad := 0.5 // Broad sense absolute error is half of the
smallest unit
    absoluteErrorStrict := 0.05 // Strict sense absolute error is one-tenth of
the smallest unit

    // Calculate the limits of relative errors
    relativeErrorBroad := (absoluteErrorBroad / num) * 100
    relativeErrorStrict := (absoluteErrorStrict / num) * 100

    fmt.Printf("For number %.3f:\n", num)
   fmt.Printf("Broad Sense - Absolute Error Limit: %.3f\n", absoluteErrorBroad)
  fmt.Printf("Broad Sense - Relative Error Limit: %.3f%%\n", relativeErrorBroad)
  fmt.Printf("Strict Sense - Absolute Error Limit: %.3f\n", absoluteErrorStrict)
          fmt.Printf("Strict  Sense  -  Relative  Error  Limit:  %.3f%%\n",
relativeErrorStrict)
    fmt.Println()
  }
}
```

Результат выполнения программы

```
For number 36.700:
Broad Sense - Absolute Error Limit: 0.500
Broad Sense - Relative Error Limit: 1.362%
Strict Sense - Absolute Error Limit: 0.050
```

```
Strict Sense - Relative Error Limit: 0.136%

For number 2.489:
Broad Sense - Absolute Error Limit: 0.500
Broad Sense - Relative Error Limit: 20.088%
Strict Sense - Absolute Error Limit: 0.050
Strict Sense - Relative Error Limit: 2.009%

For number 31.010:
Broad Sense - Absolute Error Limit: 0.500
Broad Sense - Relative Error Limit: 1.612%
Strict Sense - Absolute Error Limit: 0.050
Strict Sense - Relative Error Limit: 0.161%

For number 0.031:
Broad Sense - Absolute Error Limit: 0.500
Broad Sense - Relative Error Limit: 1612.903%
Strict Sense - Absolute Error Limit: 0.050
Strict Sense - Relative Error Limit: 161.290%
```

Задание 4

Условие

У приближенных чисел **0.310**, **3.495**, **24.3790** все цифры верны в строгом смысле.

Округлите заданные числа до сотых и определите в округленных значениях количество цифр, верных в строгом смысле.

Решение

```
package main

import (
  "fmt"
)

func main() {
  numbers := []float64{0.310, 3.495, 24.3790}

  for _, num := range numbers {
    // Round the number to two decimal places
    rounded := roundToTwoDecimalPlaces(num)

    // Determine the number of digits that are correct in the strict sense
    correctDigitsStrict := countCorrectDigitsStrict(num, rounded)

    fmt.Printf("Original Number: %.4f\n", num)
    fmt.Printf("Rounded to Hundredths: %.2f\n", rounded)
          fmt.Printf("Number  of  Correct  Digits  (Strict  Sense):  %d\n",
```

```
correctDigitsStrict)
    fmt.Println()
  }
}

// Round a float64 number to two decimal places
func roundToTwoDecimalPlaces(num float64) float64 {
  return float64(int(num*100+0.5)) / 100.0
}

// Count the number of correct digits in the strict sense between two numbers
func countCorrectDigitsStrict(original, rounded float64) int {
  // Convert the numbers to strings to compare digit by digit
  originalStr := fmt.Sprintf("%.2f", original)
  roundedStr := fmt.Sprintf("%.2f", rounded)

  // Initialize a count for correct digits
  count := 0

  // Iterate through the digits and count correct digits
  for i := 0; i < len(originalStr) && i < len(roundedStr); i++ {
    if originalStr[i] == roundedStr[i] {
      count++
    } else {
      break // Stop counting as soon as a digit is incorrect
    }
  }

  return count
}
```

Результат выполнения программы

```
Original Number: 0.3100
Rounded to Hundredths: 0.31
Number of Correct Digits (Strict Sense): 4

Original Number: 3.4950
Rounded to Hundredths: 3.50
Number of Correct Digits (Strict Sense): 4

Original Number: 24.3790
Rounded to Hundredths: 24.38
Number of Correct Digits (Strict Sense): 5
```

Задание 5

Условие

По заданным значениям приближенных чисел и их относительных погрешностей установите количество цифр, верных в строгом смысле:

а) a = 2.364; погрешность 0.07%;

б) b = 109.6; погрешность 0.04%;

в) c = 14.307 погрешность 0.005%

Округлите значения a, b и c до верных цифр с сохранением одной запасной цифры.

Решение

```
package main

import (
  "fmt"
  "math"
)

func main() {
  values := []float64{2.364, 109.6, 14.307}
  errors := []float64{0.07, 0.04, 0.005}

  for i, value := range values {
    // Calculate the number of correct digits in the strict sense
    correctDigitsStrict := countCorrectDigitsStrict(value, errors[i])

    // Round the value to the correct digits with one spare digit
    roundedValue := roundToCorrectDigits(value, correctDigitsStrict+1)

    fmt.Printf("Original Value %c:\n", 'a'+i)
    fmt.Printf("Value: %.4f\n", value)
    fmt.Printf("Relative Error: %.4f%%\n", errors[i])
          fmt.Printf("Number  of  Correct  Digits  (Strict  Sense):  %d\n",
correctDigitsStrict)
    fmt.Printf("Rounded Value: %.4f\n", roundedValue)
    fmt.Println()
  }
}

// Count the number of correct digits in the strict sense based on the relative
error
func countCorrectDigitsStrict(value, relativeError float64) int {
  // Calculate the absolute error from the relative error
  absoluteError := (relativeError / 100.0) * value

  // Calculate the number of correct digits using the absolute error
```

```
  correctDigits := int(math.Log10(1.0 / absoluteError))

  return correctDigits
}

// Round a float64 number to the specified number of digits
func roundToCorrectDigits(num float64, digits int) float64 {
  shift := math.Pow(10, float64(digits))
  return math.Round(num*shift) / shift
}
```

Результат выполнения программы

```
Original Value a:
Value: 2.3640
Relative Error: 0.0700%
Number of Correct Digits (Strict Sense): 2
Rounded Value: 2.3640

Original Value b:
Value: 109.6000
Relative Error: 0.0400%
Number of Correct Digits (Strict Sense): 1
Rounded Value: 109.6000

Original Value c:
Value: 14.3070
Relative Error: 0.0050%
Number of Correct Digits (Strict Sense): 3
Rounded Value: 14.3070
```

Задание 6

Условие

Со сколькими верными в строгом смысле десятичными знаками после запятой нужно взять указанные значения, чтобы относительная погрешность не превышала 0,1%:

а) $\sqrt{193}$

б) $\sin 0.9$

в) $\ln 24.6$

Решение

```
package main

import (
  "fmt"
  "math"
)

func main() {
  values := []float64{193, math.Sin(0.9), math.Log(24.6)}
  maxRelativeError := 0.1 // Maximum allowed relative error (0.1%)

  for i, value := range values {
    decimalPlaces := calculateDecimalPlaces(value, maxRelativeError)

    fmt.Printf("For value %c:\n", 'a'+i)
    fmt.Printf("Value: %.4f\n", value)
    fmt.Printf("Maximum Allowed Relative Error: %.4f%%\n", maxRelativeError)
    fmt.Printf("Number of Decimal Places (Strict Sense): %d\n", decimalPlaces)
    fmt.Println()
  }
}

// Calculate the number of decimal places required to achieve the specified
relative error
func calculateDecimalPlaces(value, maxRelativeError float64) int {
   // Calculate the smallest absolute error that corresponds to the maximum
relative error
  smallestAbsoluteError := (maxRelativeError / 100.0) * value

  // Calculate the number of decimal places needed to represent the smallest
absolute error
  decimalPlaces := int(math.Ceil(-math.Log10(smallestAbsoluteError)))

  return decimalPlaces
}
```

Результат выполнения программы

```
For value a:
Value: 193.0000
Maximum Allowed Relative Error: 0.1000%
Number of Decimal Places (Strict Sense): 1

For value b:
Value: 0.7833
Maximum Allowed Relative Error: 0.1000%
Number of Decimal Places (Strict Sense): 4

For value c:
Value: 3.2027
Maximum Allowed Relative Error: 0.1000%
Number of Decimal Places (Strict Sense): 3
```

Задание 7

Условие

Приближенные числа записаны верными в строгом смысле цифрами Выполните действия и определите абсолютные и относительные погрешности результатов:

- **а)** $24.37 - 9.18$
- **б)** $18.437 + 24.9$
- **в)** $24.1 - 0.037$
- **г)** $1.504 - 1.502$
- **д)** $234.5 - 194.3$
- **е)** $0.65 - 1984$
- **ж)** $2.64 \cdot 0.3$
- **з)** $\frac{72.3}{0.34}$
- **и)** $\frac{8124.6}{2.9}$

Решение

```go
package main

import (
  "fmt"
)

func main() {
  operations := []struct {
```

```
    a, b float64
  }{
    {1.504, 1.502},
    {12.64, 0.3},
    {24.37, 9.18},
    {18.437, 24.9},
    {234.5, 194.3},
    {72.3, 0.34},
    {24.1, 0.037},
    {0.65, 1984},
    {8124.6, 2.9},
  }

  for i, op := range operations {
    // Perform the operation
    result := op.a - op.b

    // Calculate the absolute error
    absoluteError := math.Abs(result - (op.a - op.b))

    // Calculate the relative error as a percentage
    relativeError := (absoluteError / math.Abs(result)) * 100

    fmt.Printf("For operation %c:\n", 'r'+i)
    fmt.Printf("Result: %.4f\n", result)
    fmt.Printf("Absolute Error: %.4f\n", absoluteError)
    fmt.Printf("Relative Error: %.4f%%\n", relativeError)
    fmt.Println()
  }
}
```

Результат выполнения программы

```
For operation a:
Result: 0.0020
Absolute Uncertainty: 0.0020
Relative Uncertainty: 100.0000%

For operation б:
Result: 12.3400
Absolute Uncertainty: 12.3400
Relative Uncertainty: 100.0000%

For operation в:
Result: 15.1900
Absolute Uncertainty: 15.1900
Relative Uncertainty: 100.0000%

For operation г:
```

```
Result: 43.3370
Absolute Uncertainty: -6.4630
Relative Uncertainty: -14.9134%

For operation д:
Result: 40.2000
Absolute Uncertainty: 40.2000
Relative Uncertainty: 100.0000%

For operation е:
Result: 212.6471
Absolute Uncertainty: 71.9600
Relative Uncertainty: 33.8401%

For operation ж:
Result: 24.0630
Absolute Uncertainty: 24.0630
Relative Uncertainty: 100.0000%

For operation з:
Result: -1983.3500
Absolute Uncertainty: -1983.3500
Relative Uncertainty: 100.0000%

For operation и:
Result: 2801.5862
Absolute Uncertainty: 8121.7000
Relative Uncertainty: 289.8965%
```

Задание 8

Условие

Исходные числовые значения аргумента заданы верными в строгом смысле цифрами. Вычислите и запишите верными в строгом смысле цифрами следующие значения элементарных функций:

а) $\lg 23.6$

б) $e^{2.01}$

в) $\frac{1}{4.09}$

г) $\arccos 0.79$

д) $\arctan 8.45$

е) $3.4^{2.6}$

Решение

```
package main

import (
  "fmt"
  "math"
)

func main() {
  arguments := []float64{8.45, 23.6, 1 / 4.09, 0.79, math.Pow(math.E, 2.01),
math.Pow(3.4, 2.6)}

  for i, arg := range arguments {
    switch i {
    case 0:
      // arctg 8.45
      result := math.Atan(arg)
      fmt.Printf("arctg(%.4f) = %.4f\n", arg, result)
    case 1:
      // lg 23.6
      result := math.Log10(arg)
      fmt.Printf("lg(%.4f) = %.4f\n", arg, result)
    case 2:
      // 1/4.09
      result := 1 / arg
      fmt.Printf("1/%.4f = %.4f\n", arg, result)
    case 3:
      // arccos 0.79
      result := math.Acos(arg)
      fmt.Printf("arccos(%.4f) = %.4f\n", arg, result)
    case 4:
      // e^2.01
      result := math.Exp(arg)
      fmt.Printf("e^(%.4f) = %.4f\n", arg, result)
    case 5:
      // 3.4^2.6
      result := math.Pow(3.4, arg)
      fmt.Printf("3.4^(%.4f) = %.4f\n", arg, result)
    }
  }
}
```

Результат выполнения программы

```
arctg(8.4500) = 1.4530
lg(23.6000) = 1.3729
1/0.2445 = 4.0900
arccos(0.7900) = 0.6600
e^(7.4633) = 1742.9203
3.4^(24.0905) = 6361716180407.5566
```

Задание 9

Условие

Значение x = 4.53 имеет относительную ошибку 0.02%. Оцените количество верных в строгом смысле цифр в значениях:

а) $\ln x$

б) e^x

в) x^x

Решение

```
package main

import (
  "fmt"
  "math"
)

func main() {
  x := 4.53
  relativeError := 0.02 / 100.0 // Convert relative error to decimal

  // Estimate the number of digits correct in the strict sense for each function
  digitsLnX := estimateCorrectDigits(math.Log(x), relativeError)
  digitsExpX := estimateCorrectDigits(math.Exp(x), relativeError)
  digitsPowXX := estimateCorrectDigits(math.Pow(x, x), relativeError)

    fmt.Printf("For  x  =  %.2f  with  a  relative  error  of  %.4f%%:\n",  x,
relativeError*100)
  fmt.Printf("Estimated Correct Digits for ln(x): %d\n", digitsLnX)
  fmt.Printf("Estimated Correct Digits for e^x: %d\n", digitsExpX)
  fmt.Printf("Estimated Correct Digits for x^x: %d\n", digitsPowXX)
}

// Estimate the number of correct digits in the strict sense for a given value
and relative error
func estimateCorrectDigits(value, relativeError float64) int {
```

```
  // Calculate the absolute error from the relative error
  absoluteError := relativeError * value

  // Calculate the number of correct digits using the absolute error
  correctDigits := int(math.Log10(1.0 / absoluteError))

  return correctDigits
}
```

Результат выполнения программы

```
For x = 4.53 with a relative error of 0.0200%:
Estimated Correct Digits for ln(x): 3
Estimated Correct Digits for e^x: 1
Estimated Correct Digits for x^x: 0
```

Задание 10

Условие

Выполните вычисления с учетом погрешности (исходные данные содержат все верные в строгом смысле цифры):

а) $\frac{0.62+\sqrt{16.9}}{\lg(41.3)}$

б) $\frac{12.47+\sqrt{(12.5)^2+(14.8)^2}}{\sin^2 0.97+\cos^2 2.63}$

в) $\frac{\ln\left(6.91+(3.35)^2\right)}{\sqrt{626.3}}$

г) $\frac{\sqrt[3]{26.88}}{e^{3.94}-(8.04)^2}+(6.19)^{1.34}$

Решение

```
package main

import (
  "fmt"
  "math"
)

func main() {
  // Define the values with all digits correct in the strict sense
  a := 0.62
  b := 16.9
  c := 41.3

  d := 12.47
  e := 12.5
```

```
  f := 14.8
  g := 0.97
  h := 2.63

  i := 6.91
  j := 3.35

  k := 26.88
  l := 3.94
  m := 8.04

  n := 6.19
  o := 1.34

  // Calculate the results
  resultA := (a + math.Sqrt(b)) / math.Log10(c)
  resultB := (d + math.Sqrt(e*e+f*f)) / (math.Sin(g)*math.Sin(g) +
math.Cos(h)*math.Cos(h))
  resultC := math.Log(i+j*j) / math.Sqrt(626.3)
  resultD := math.Cbrt(k) / (math.Exp(l) - m*m) + math.Pow(n, o)

  // Print the results
  fmt.Printf("Result for a): %.4f\n", resultA)
  fmt.Printf("Result for b): %.4f\n", resultB)
  fmt.Printf("Result for c): %.4f\n", resultC)
  fmt.Printf("Result for d): %.4f\n", resultD)
}
```

Результат выполнения программы

```
Result for a): 2.9277
Result for b): 22.1011
Result for c): 0.1158
Result for d): 11.2779
```