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 № 4

Тема Наилучшее среднеквадратичное приближение

Студент Кононенко С.С.

Группа ИУ7-43Б

Оценка (баллы) ________________________________

Преподаватель Градов В.М.

Москва — 2020 г.

1. Задание

Задана таблица значений функции вида $x_i, f(x_i), \rho_i$, где x_i - значение аргумента, $f(x_i)$ - значение функции, ρ_i - вес точки. Построить алгоритм реализации наилучшего среднеквадратичного приближения.
Ввод: степень аппроксимирующего полинома n.
Вывод: график, на котором изображен аппроксимирующий полином, и точки из исходной таблицы значений.

2. Описание алгоритма

Под близостью в среднем исходной и аппроксимирующей функций будем понимать результат оценки суммы:

$$I = \sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 \qquad (1)$$

$y(x)$ - исходная функция;
$\varphi(x)$ - множество функций, принадлежащих линейному пространству функций;
ρ_i - вес точки.

Нужно найти наилучшее приближение, то есть:

$$\sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 = min \qquad (2)$$

Разложим функцию $\varphi(x)$ по системе линейно-независимых функций $\varphi_k(x)$:

$$\varphi(x) = \sum_{k=0}^{N} a_k \varphi_k(x) \qquad (3)$$

Подставляя (3) в условие (2) получим:

$$((y - \varphi), (y - \varphi)) = (y, y) - 2\sum_{k=0}^{n} a_k (y, \varphi_k) + \sum_{k=0}^{n} \sum_{m=0}^{n} a_k a_m (\varphi_k, \varphi_m) = min \qquad (4)$$

Дифференцируя по a_k получаем:

$$\sum_{i=0}^{n} (x^k, x^m) a_m = (y, x^k), 0 \leq k \leq n \qquad (5)$$

где

$$(x^k, x^m) = \sum_{i=1}^{N} \rho_i x_i^{k+m}, (y, x^k) = \sum_{i=1}^{N} \rho_i y_i x_i^k$$

Итоговый алгоритм:
1. Выбирается степень полинома $n << N$.
2. Составляется система линейных алгебраических уравнений типа (5).
3. В результате решения СЛАУ находятся коэффицинты полинома a_k.

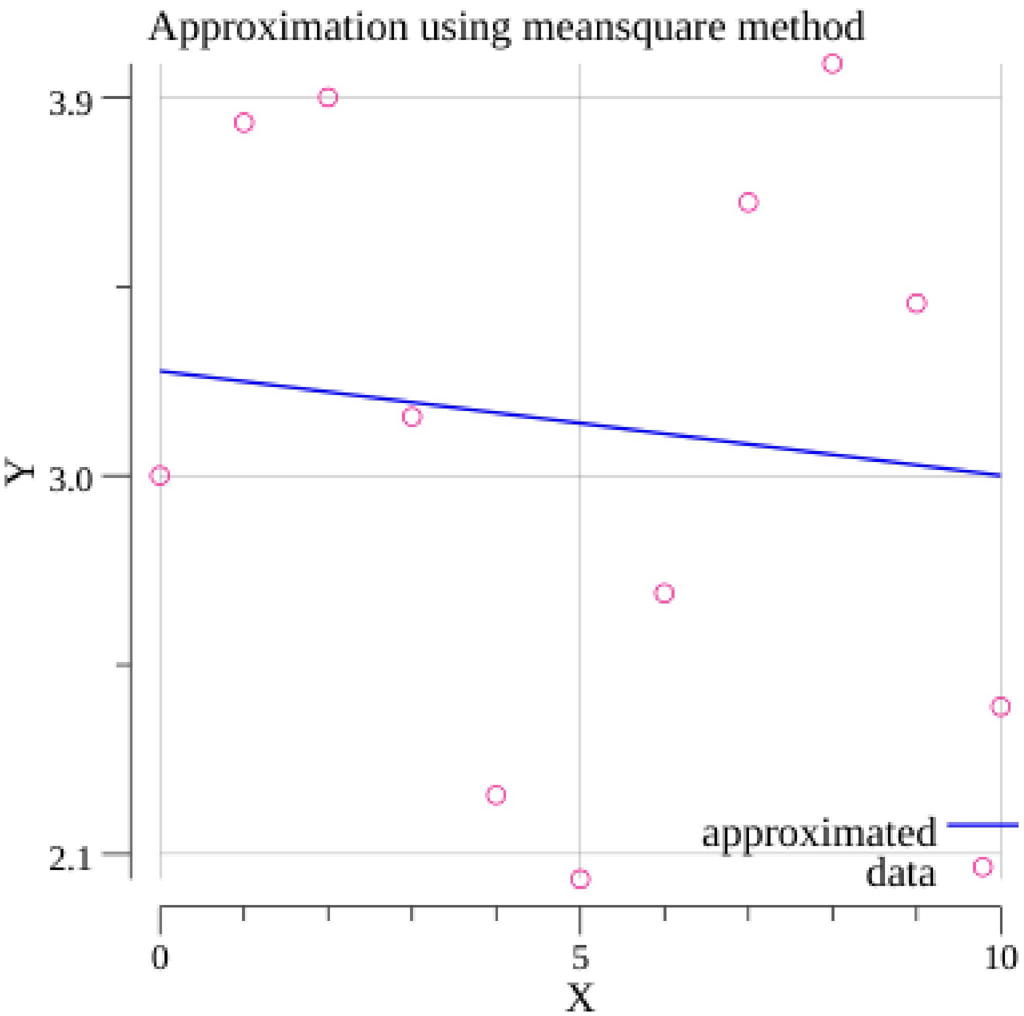
3. Результаты работы программы

1. Случай, когда веса точек равны между собой (для данного примера $\rho_i = 1$)

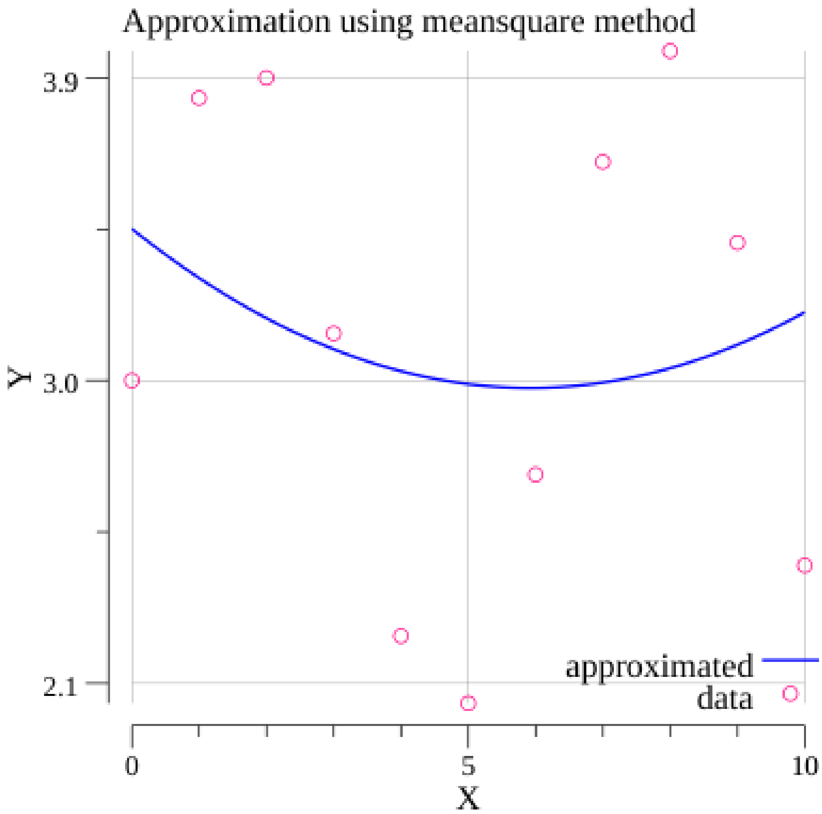
Исходная таблица:

x_i	$f(x_i)$	ρ_i
0	3.00	1
1	3.84	1
2	3.90	1
3	3.14	1
4	2.24	1
5	2.04	1
6	2.72	1
7	3.65	1
8	3.98	1
9	3.41	1
10	2.45	1

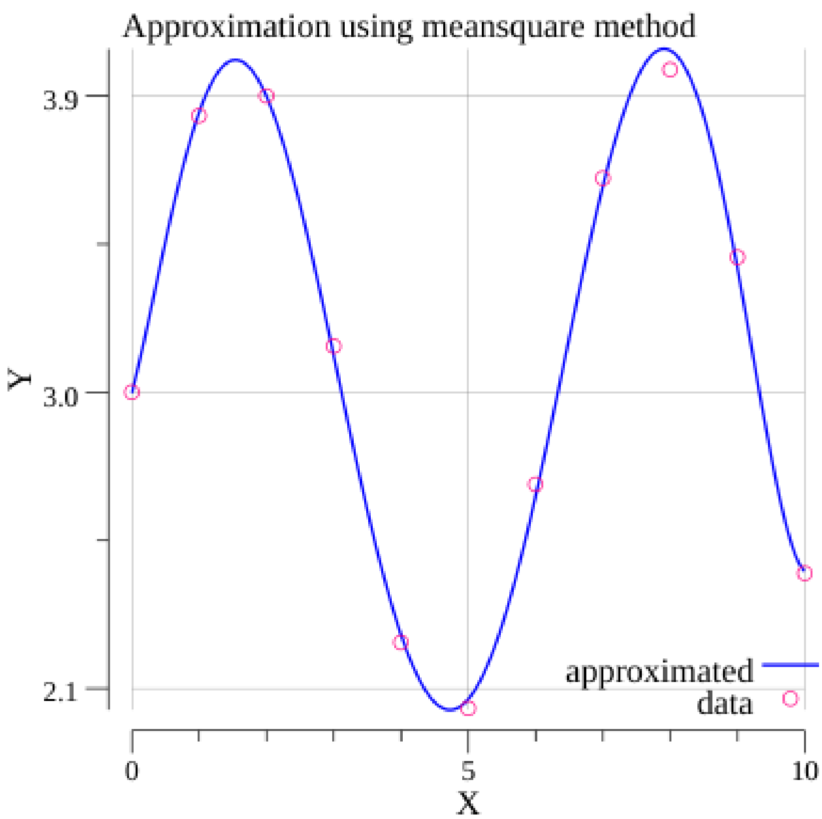
Построенный график (при $n = 1$):



Построенный график (при $n = 2$):



Построенный график (при $n = 6$):

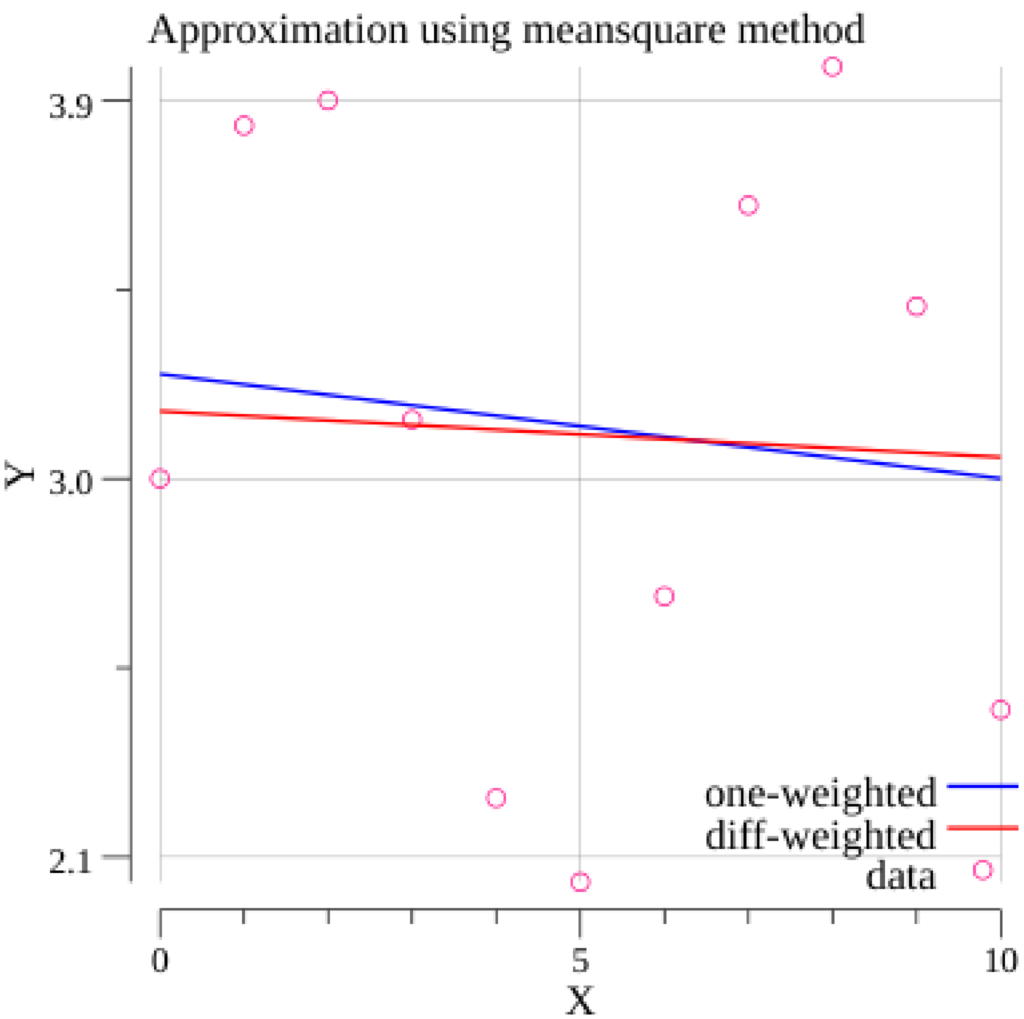


2. Случай, когда веса точек различны (для данного примера $\rho_{1_i} = 1, \rho_{2_i} \neq \rho_{1_i}$)

Исходная таблица:

x_i	$f(x_i)$	ρ_{1_i}	ρ_{2_i}
0	3.00	1	0.6
1	3.84	1	0.7
2	3.90	1	0.8
3	3.14	1	1.6
4	2.24	1	1.2
5	2.04	1	1.1
6	2.72	1	0.3
7	3.65	1	0.4
8	3.98	1	1.7
9	3.41	1	1.3
10	2.45	1	1.5

Построенный график (синяя прямая - одинаковые веса, красная прямая - различные веса):



4. Ответы на вопросы для защиты лабораторной работы

1. Что произойдет при задании степени полинома $n = N - 1$?

Для однозначного определения полинома $N - 1$ степени достаточно N точек, что означает, что полином будет построен таким образом, что его график будет проходить через все табличные точки. При такой конфигурации в выражении $\sum_{i=1}^{N} \rho_i[y(x_i) - \varphi(x_i)]^2 = min$ часть выражения, находящаяся в скобках, обратится в 0, что означает, что нет зависимости от весов (при любых заданных весах, значение полинома будет минимальным в случае прохода через табличные точки).

2. Будет ли работать Ваша программа при $n \geq N$? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?

Программа работать будет, но её работа будет некорректна, так как начиная со случая $n = N$ определитель СЛАУ, которую необходимо решить, будет тождественно равен 0 (уравнения СЛАУ не будут линейно-независимыми). Аварийная остановка программы может произойти при приведении диагональной матрицы к единичной (СЛАУ решалась методом Гаусса-Жордана, поэтому предполагается приведение диагональной матрицы к единичной), где может произойти деление на ноль. Анализ можно проводить при решении СЛАУ или же на начальном этапе (ввод степени полинома).

3. Получить формулу для коэффициента a_0 при степени полинома $n = 0$. Какой смысл имеет величина, которую представляет данный коэффициент?

Полученная формула:

$$a_0 = \frac{\sum_{i=1}^{N} \rho_i y_i}{\sum_{i=1}^{N} \rho_i}$$

где ρ_i - вес i точки. Данную формулу можно преобразовать делением числителя и знаменателя на сумму весов, из чего получится *математическое ожидание*:

$$M[X] = \sum_{i=1}^{\infty} x_i \rho_i$$

4. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда $n = N = 2$. Принять все $\rho_i = 1$.

Зададим таблицу точек:

x_i	y_i	ρ_i
x_0	y_0	1
x_1	y_1	1

Тогда имеем СЛАУ вида:

$$\begin{cases} a_0 + (x_0 + x_1)a_1 + (x_0^2 + x_1^2)a_2 = y_0 + y_1 \\ (x_0 + x_1)a_0 + (x_0^2 + x_1^2)a_1 + (x_0^3 + x_1^3)a_2 = y_0x_0 + y_1x_1 \\ (x_0^2 + x_1^2)a_0 + (x_0^3 + x_1^3)a_1 + (x_0^4 + x_1^4)a_2 = y_0x_0^2 + y_1x_0^2 \end{cases}$$

$$\Delta = (x_0^2 + x_1^2)(x_0^4 + x_1^4) + (x_0 + x_1)(x_0^3 + x_1^3)(x_0^2 + x_1^2) +$$

$$(x_0^2 + x_1^2)(x_0 + x_1)(x_0^3 + x_1^3) - (x_0^2 + x_1^2)(x_0^2 + x_1^2)(x_0^2 + x_1^2) -$$

$$(x_0^3 + x_1^3)(x_0^3 + x_1^3) - (x_0 + x_1)(x_0 + x_1)(x_0^4 + x_1^4) = 0$$

Так как $\Delta = 0$, система решений не имеет, что было упомянуто в ответе на вопрос **2**.

5. Код программы

5.1. Основной пакет приложения.

```go
package main

import (
  "fmt"
  "os"

  "./meansquare"
)

func main() {
  if len(os.Args) <= 1 {
    fmt.Printf("USAGE: lab_04 <datafile>\n")
    os.Exit(1)
  }

  f, err := os.Open(os.Args[1])
  if err != nil {
    fmt.Println("Error:", err)
    os.Exit(1)
  }

  ds := meansquare.ReadDots(f)
  fmt.Printf("Table loaded from file:\n\n")
  ds.PrintDots()

  fmt.Printf("\nEnter polynom degree: ")
  n := meansquare.ReadPolynomDegree()

  slae := meansquare.MakeSLAE(ds, n)
  fmt.Printf("\nSLAE to solve:\n\n")
  slae.PrintMatrix()

  sol := meansquare.SolveSLAE(ds, n)
  fmt.Printf("\nSolved SLAE:\n\n")
  sol.PrintMatrix()

  coeffs := meansquare.GetCoeffs(sol)
  dots := meansquare.GetApprox(ds, coeffs)

  meansquare.DrawPlot(ds, dots)
}
```

5.2. Пакет реализации алгоритма наилучшего среднеквадратичного приближения.

5.2.1. Обработка точек на плоскости.

```
package meansquare

import (
  "fmt"
  "io"
)

// Dot type used to represent plane dots with dot weight.
type Dot struct {
  X      float64
  Y      float64
  weight float64
}

// DotSet type used to represent amount of plane dots.
type DotSet []Dot

// ReadPolynomDegree used to read polynom degree.
func ReadPolynomDegree() int {
  var n int

  fmt.Scan(&n)

  return n
}

// ReadDots used to read Dot objects to DotSet object from file.
func ReadDots(f io.Reader) DotSet {
  var (
    ds     DotSet
    curDot Dot
  )

  for {
    _, err := fmt.Fscanln(f, &curDot.X, &curDot.Y, &curDot.weight)
    if err == io.EOF {
      break
    }
    ds = append(ds, curDot)
  }

  return ds
}
```

```go
// PrintDots used to print dots in table form to standart output.
func (ds DotSet) PrintDots() {
  fmt.Printf("%8s %8s %8s\n", "X", "Y", "Weight")
  for i := range ds {
    fmt.Printf("%8.2f %8.2f %8.2f\n", ds[i].X, ds[i].Y, ds[i].weight)
  }
}
```

5.2.2. Реализация среднеквадратичного приближения.

```go
package meansquare

import (
  "fmt"
  "math"
)

// F64Matrix type used to represent float64 matrix.
type F64Matrix [][]float64

// GetApprox used to find approximation dots.
func GetApprox(ds DotSet, coeffs []float64) DotSet {
  var dots DotSet

  for i := ds[0].X; i < ds[len(ds)-1].X; i += 0.01 {
    d := Dot{
      X: i,
    }
    for j := 0; j < len(coeffs); j++ {
      d.Y += math.Pow(d.X, float64(j)) * coeffs[j]
    }
    dots = append(dots, d)
  }

  return dots
}

// GetCoeffs used to get found SLAE solution.
func GetCoeffs(mat F64Matrix) []float64 {
  var coeffs []float64

  matlen := len(mat)

  for i := 0; i < matlen; i++ {
    coeffs = append(coeffs, mat[i][matlen])
  }

  return coeffs
}
```

```go
// SolveSLAE used to solve SLAE.
func SolveSLAE(ds DotSet, n int) F64Matrix {
  slae := MakeSLAE(ds, n)

  for i := 0; i < n+1; i++ {
    for j := 0; j < n+1; j++ {
      if i == j {
        continue
      }
      subCoeff := slae[j][i] / slae[i][i]
      for k := 0; k < n+2; k++ {
        slae[j][k] -= subCoeff * slae[i][k]
      }
    }
  }

  for i := 0; i < n+1; i++ {
    divider := slae[i][i]
    for j := 0; j < n+2; j++ {
      slae[i][j] /= divider
    }
  }

  return slae
}

// MakeSLAE used to make SLAE.
func MakeSLAE(ds DotSet, n int) F64Matrix {
  mat := make(F64Matrix, n+1)
  for i := range mat {
    mat[i] = make([]float64, n+2)
  }

  for i := 0; i < n+1; i++ {
    for j := 0; j < n+1; j++ {
      slaeCoeffs := 0.0
      expandedCoeff := 0.0
      for k := 0; k < len(ds); k++ {
        slaeCoeffs += ds[k].weight * math.Pow(ds[k].X, float64(i)) *
          math.Pow(ds[k].X, float64(j))
        expandedCoeff += ds[k].weight * ds[k].Y *
          math.Pow(ds[k].X, float64(i))
      }
      mat[i][j] = slaeCoeffs
      mat[i][n+1] = expandedCoeff
    }
  }
```

```
  return mat
}

// PrintMatrix used to print matrix in matrix form to standart output.
func (mat F64Matrix) PrintMatrix() {
  for i := 0; i < len(mat); i++ {
    for j := 0; j < len(mat)+1; j++ {
      fmt.Printf("%15.1f ", mat[i][j])
    }
    fmt.Printf("\n")
  }
}
```

5.2.3. Отрисовка графика.

```
package meansquare

import (
  "fmt"
  "image/color"
  "os"

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

func convertDots(ds DotSet) plotter.XYs {
  var conv plotter.XYs

  for _, d := range ds {
    cd := plotter.XY{
      X: d.X,
      Y: d.Y,
    }
    conv = append(conv, cd)
  }

  return conv
}

// DrawPlot used to draw plot by approximated dots.
func DrawPlot(ds, approx DotSet) {
  p, err := plot.New()
  if err != nil {
    fmt.Println("Error:", err)
    os.Exit(1)
  }
```

```go
  p.Title.Text = "Approximation using meansquare method"
  p.X.Label.Text = "X"
  p.Y.Label.Text = "Y"
  p.Add(plotter.NewGrid())

  conv := convertDots(approx)
  def := convertDots(ds)

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

  s, err := plotter.NewScatter(def)
  if err != nil {
    fmt.Println("Error:", err)
    os.Exit(1)
  }
  s.GlyphStyle.Color = color.RGBA{R: 255, B: 128, A: 255}

  p.Add(l, s)
  p.Legend.Add("approximated", l)
  p.Legend.Add("data", s)

  if err := p.Save(4*vg.Inch, 4*vg.Inch, "points.png"); err != nil {
    panic(err)
  }
}
```