```go
package main

import (
  "fmt"
  "math"
)

func MultiplyMatrixByFloat64(matrix [][]float64, num float64) (matrixRes
[][]float64) {
  matrixNumOfLines := len(matrix)
  matrixNumOfColumns := len(matrix[0])
  for i := 0; i < matrixNumOfLines; i++ {
    for j := 0; j < matrixNumOfColumns; j++ {
      matrix[i][j] = matrix[i][j] * num
    }
  }
  return matrix
}

func SolucaoViaMetodoDaPotencia(c configuration) (autovalor float64, autovetor
[][]float64) {
  Xvelho := inicializarVetorSolucaoZero(c)
  Pw(OUTPUT_FILE_PATH, "Iteração 0")
  Pw(OUTPUT_FILE_PATH, fmt.Sprintf("X inicial %s\n",
CreateMatrixString(Xvelho)))
  AX, canMultiply := MultiplyMatrices(c.matrixA, Xvelho)

  Pw(OUTPUT_FILE_PATH, fmt.Sprintf("X+1:\n%s\n", CreateMatrixString(AX)))

  if !canMultiply {
    panic("Matrix multiplication not allowed.")
  }
  var lambdaVelho float64 = 1
  lambdaNovo := AX[0][0]

  Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Lambda inicial: %v\n", lambdaVelho))
  Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Lambda+1: %v\n", lambdaNovo))

  Xnovo := MultiplyMatrixByFloat64(AX, 1/lambdaNovo)

  residue := math.Sqrt(math.Pow(lambdaNovo-lambdaVelho, 2)) / math.Sqrt(math.Pow(lambdaNovo, 2))

  Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Residuo inicial: %v\n", residue))
  Pw(OUTPUT_FILE_PATH, SEPARADOR)

  iteration := 0
  for residue > c.TOLm {
    iteration++
    Xvelho = Xnovo
    lambdaVelho = lambdaNovo
    AX, canMultiply = MultiplyMatrices(c.matrixA, Xvelho)
    if !canMultiply {
      panic("Matrix multiplication not allowed.")
    }
    lambdaNovo = AX[0][0]
    Xnovo = MultiplyMatrixByFloat64(AX, 1/lambdaNovo)
    residue = math.Sqrt(math.Pow(lambdaNovo-lambdaVelho, 2)) / math.Sqrt(math.Pow(lambdaNovo, 2))

    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Iteração %v", iteration))
    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("X:\n%s\n", CreateMatrixString(Xnovo)))
    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Lambda: %v\n", lambdaNovo))
    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Residuo: %v\n", residue))
    Pw(OUTPUT_FILE_PATH, SEPARADOR)
  }

  autovalor = lambdaNovo
  autovetor = Xnovo
  return autovalor, autovetor
}
```