```go
package main

import (
  "fmt"
  "math"
)

var ()

func LUViaCholeskyDecomposition(c configuration) (L, U [][]float64) {
  //Inicializando as matrizes necessárias
  L = InitializeMatrixWithZeros(c.systemOrder, c.systemOrder)
  U = InitializeMatrixWithZeros(c.systemOrder, c.systemOrder)
  n := c.systemOrder
  // Decomposing a matrix into Lower Triangular
  for i := 0; i < n; i++ {
    for j := 0; j <= i; j++ {
      var sum float64
      if j == i { // summation for diagonals
        for k := 0; k < j; k++ {
          sum += math.Pow(L[j][k], 2)
        }
        L[j][j] = math.Sqrt(c.matrixA[j][j] - sum)
      } else {
        // Evaluating L(i, j) using L(j, j)
        for k := 0; k < j; k++ {
          sum += (L[i][k] * L[j][k])
        }
        L[i][j] = (c.matrixA[i][j] - sum) / L[j][j]
      }
    }
  }

  //Transpose L to find U
  for i := 0; i < n; i++ {
    for j := 0; j < n; j++ {
      U[i][j] = L[j][i]
```

```
38      }
39    }
40    return L, U
41 }
42
43 func SolutionViaCholeskyDecomposition(c configuration) (res [][]float64) {
44    L, U := LUViaCholeskyDecomposition(c)
45    Lstring := CreateMatrixString(L)
46    Ustring := CreateMatrixString(U)
47    //Escrevendo em arquivo
48    Pw(OUTPUT_FILE_PATH, "Matriz L encontrada\n")
49    Pw(OUTPUT_FILE_PATH, Lstring)
50    Pw(OUTPUT_FILE_PATH, "Matriz U encontrada\n")
51    Pw(OUTPUT_FILE_PATH, Ustring)
52    res1 := forwardSubstitution(L, c.vectorB)
53    res2 := backwardsSubstitution(U, res1)
54    res2String := CreateMatrixString(res2)
55    Pw(OUTPUT_FILE_PATH, fmt.Sprintf("Resultado final\n%s\n", res2String))
56    return res2
57 }
58
```