UNIVERSIDADE FEDERAL DO CEARÁ - UFC
CIÊNCIA DA COMPUTAÇÃO
PROJETO DE MÉTODOS NUMÉRICOS I
PARTE II

Tema (2): Encontrar os raios dos círculos para o cálculo das áreas

Equipe

* Caio Viktor
* Cristiano Melo
* Lucas Falcão
* Geraldo Braz
* Matheus Mayron

Ferramentas de desenvolvimento

C++

Qt

Enunciado

São passados como parâmetro um número n (tamanho da Matriz [C] – quantidade de raios a ser calculado), a matriz [C], que armazenará os coeficientes do sistema a ser resolvido e um vetor b. Com isto, resolveremos um sistema do tipo

$$[C]\{r\} = \{b\}$$

onde o vetor r nos retornará o valor dos raios que desejamos obter.

Objetivo

* Encontrar e analisar aproximações de raios de circunferência resolvendo os sistemas lineares através dos seguintes métodos exatos:
  - Gauss;
  - Gauss-Jordan.

UML


TemplateGauss
- coefficientMatrix : Matrix*
- independentTermsMatrix : Matrix*
- unknownsMatrix : Matrix*
- results : ListResults*
- solvable : bool
- executionTime : long double
resetList() : void
saveOnList(desc : string) : void
retroSubstitutions() : void
beforeSolve() : void
afterSolve() : void
pivoting(A : Matrix*, b : Matrix*, numberOfLines, : int , k : int) : void
switchRows(m : Matrix*, line_i : int, line_j : int) : void
setSolvable(s : bool) : void
+ GaussTemplate(independentTermsMatrix : Matrix*, coefficientMatrix : Matrix*)
+ setCoefficienMatrix(matrix : Matrix*) : void
+ getCoefficienMatrix() : Matrix*
+ setIndependentTerms(Matrix* matrix) : void
+ getIndependentTerms() : Matrix*
+ getUnknownsMatrix() : Matrix*
+ setExecutionTime(executionTime : long double) : Matrix*
+ getExecutionTime() : long double
+ getResults() : ListResults*
+ isSolvable() :bool
Gauss
+ resolveSytem(usePivot : bool) : void
GaussJordan
+ resolveSytem(usePivot : bool) : void

Operações comuns

```
void GaussTemplate::beforeSolve(){
    Matrix* copy1 = getIndependentTerms()->getCopy();
    Matrix* copy2 = getCoefficienMatrix()->getCopy();
    this->independentTermsMatrixTemp = getIndependentTerms();
    this->coefficientMatrixTemp = getCoefficienMatrix();
    setIndependentTerms(copy1);
    setCoefficienMatrix(copy2);

    setSolvable(true);
    setExecutionTime(0);
    resetList();

}


void GaussTemplate::afterSolve(){
    delete getIndependentTerms();
    delete getCoefficienMatrix();
    setIndependentTerms(this->independentTermsMatrixTemp);
    setCoefficienMatrix(this->coefficientMatrixTemp);
}
```

Operações comuns

```
void GaussTemplate::retroSubstitutions(){
    Matrix *coefficientsMatrix = getCoefficienMatrix();
    Matrix *independentTermsMatrix = getIndependentTerms();

    int numberOfLines = independentTermsMatrix->getHeight();

    Matrix *unknowns = new Matrix(numberOfLines,1);
    double unknown_k;
    double sum;

    unknown_k = independentTermsMatrix->getValue(numberOfLines-1,0)/coefficientsMatrix->getValue(numberOfLines-1,numberOfLines-1);
    unknowns->setValue(numberOfLines-1,0,unknown_k);
    for(int k=numberOfLines-2; k>=0; k--){
        sum = 0;
        for (int j=(k + 1); j <= numberOfLines-1; j++){
            sum = sum + coefficientsMatrix->getValue(k,j) * unknowns->getValue(j,0);
        }

        unknown_k = (independentTermsMatrix->getValue(k,0) - sum)/coefficientsMatrix->getValue(k,k);
        unknowns->setValue(k,0,unknown_k);
    }

    if (this->unknownsMatrix!=NULL){
        delete this->unknownsMatrix;
    }
    this->unknownsMatrix = unknowns;

}
```

Operações comuns

```
void GaussTemplate::pivoting(Matrix* A, Matrix* b, int numberOfLines, int k){
    double max = A->getValue(k,k);
    int index = k;

    for( int i = k+1; i < numberOfLines; i++){
        if ( fabs( A->getValue( i, k ) ) > fabs( max ) ){
            max = A->getValue( i, k );
            index = i;
        }

    }
    if(index!=k){
        std::ostringstream description;
        description<<"Operação realizada: L"<< k <<" <-> L"<< index <<"\n";
        saveOnList(description.str());
        switchRows(A,k,index);
        switchRows(b,k,index);
    }

}
```

Operações comuns

```
void GaussTemplate::multiplyRowByScalar(Matrix* A, Matrix* b, int line_i, double scalar){
    int numberOfColumns = A->getWidth();
    double newValue;

    for(int j = line_i; j <= numberOfColumns; j++ ){
        newValue = A->getValue( line_i, j ) * scalar;
        A->setValue( line_i, j, newValue );
    }

    b->setValue( line_i, 0, b->getValue(line_i,0) * scalar);
}
```

```
void GaussTemplate::addRowByOtherRowMultipliedByScalar(Matrix *A, Matrix *b, int line_i, int line_j, double scalar){
    int numberOfColumns = A->getWidth();
    double newValue;

    for(int k = line_j; k <= numberOfColumns; k++ ){
        newValue = A->getValue(line_i, k ) - scalar * A->getValue( line_j, k );
        A->setValue( line_i, k, newValue );
    }

    newValue = b->getValue( line_i, 0 ) - scalar * b->getValue( line_j, 0 );
    b->setValue( line_i, 0, newValue );
}
```

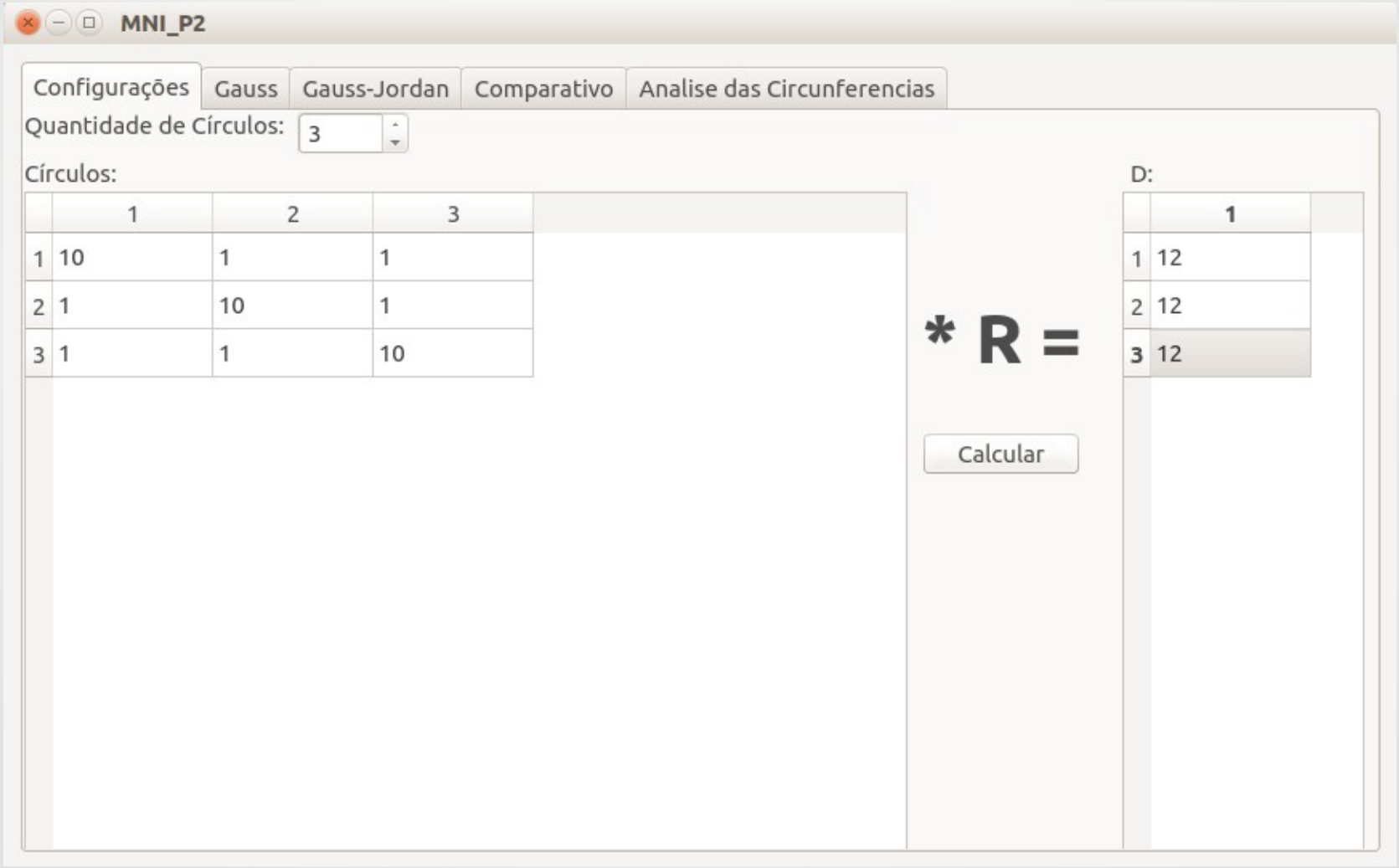
Gauss

```
beforeSolve();
try{
    for(int k = 0; k<=numberOfLines-2; k++){
        for(int i = k +1; i<=numberOfLines-1;i++){

            if(usePivot == true){ pivoting( coefficientsMatrix, independentTermsMatrix, numberOfLines, k );}

            pivo = coefficientsMatrix->getValue(k,k);

            if(pivo == 0){ throw 0; }

            multiplier = coefficientsMatrix->getValue(i,k)/pivo;

            addRowByOtherRowMultipliedByScalar(coefficientsMatrix,independentTermsMatrix,i,k,multiplier);

        }

        if(coefficientsMatrix->getValue(numberOfLines-1,numberOfLines-1) == 0){ throw 1; }

    }

}

catch(int e){
    setSolvable(false);
}
afterSolve();
```

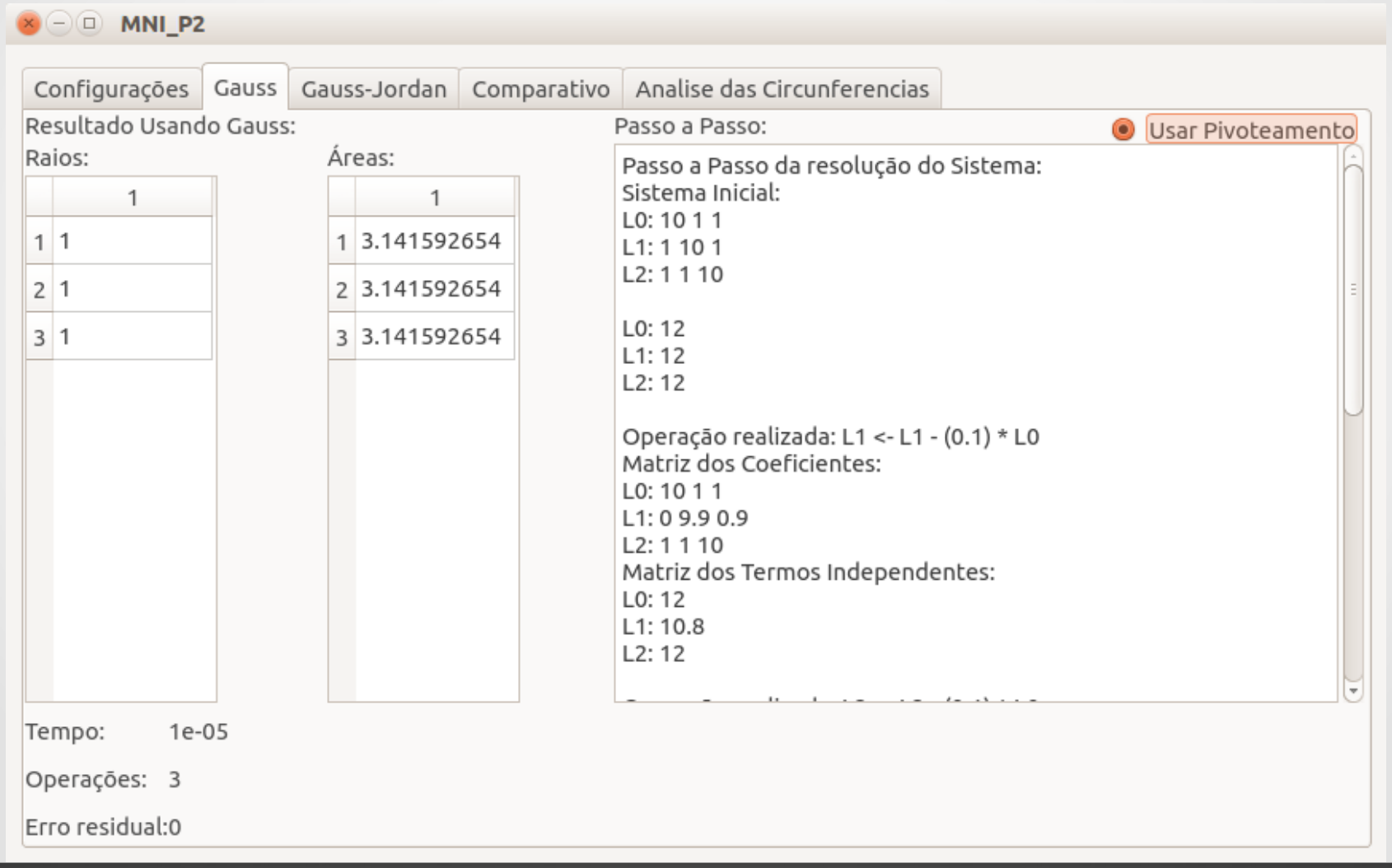
Gauss-Jordan

```
beforeSolve();
try{
    for(int k = 0; k < numberOfLines; k++ ){

        if(usePivot == true){ pivoting( coefficientsMatrix, independentTermsMatrix, numberOfLines, k ); }

        pivo = coefficientsMatrix->getValue(k,k);

        if(pivo == 0){ throw 0; }

        multiplyRowByScalar(coefficientsMatrix, independentTermsMatrix, k, 1/pivo);

        for(int i = 0; i < numberOfLines; i++ ){
            if( i != k ){
                multiplier = coefficientsMatrix->getValue(i,k);
                addRowByOtherRowMultipliedByScalar(coefficientsMatrix, independentTermsMatrix, i, k, multiplier);
            }
        }
    }

}
catch(int e){
    setSolvable(false);
}
afterSolve();
```

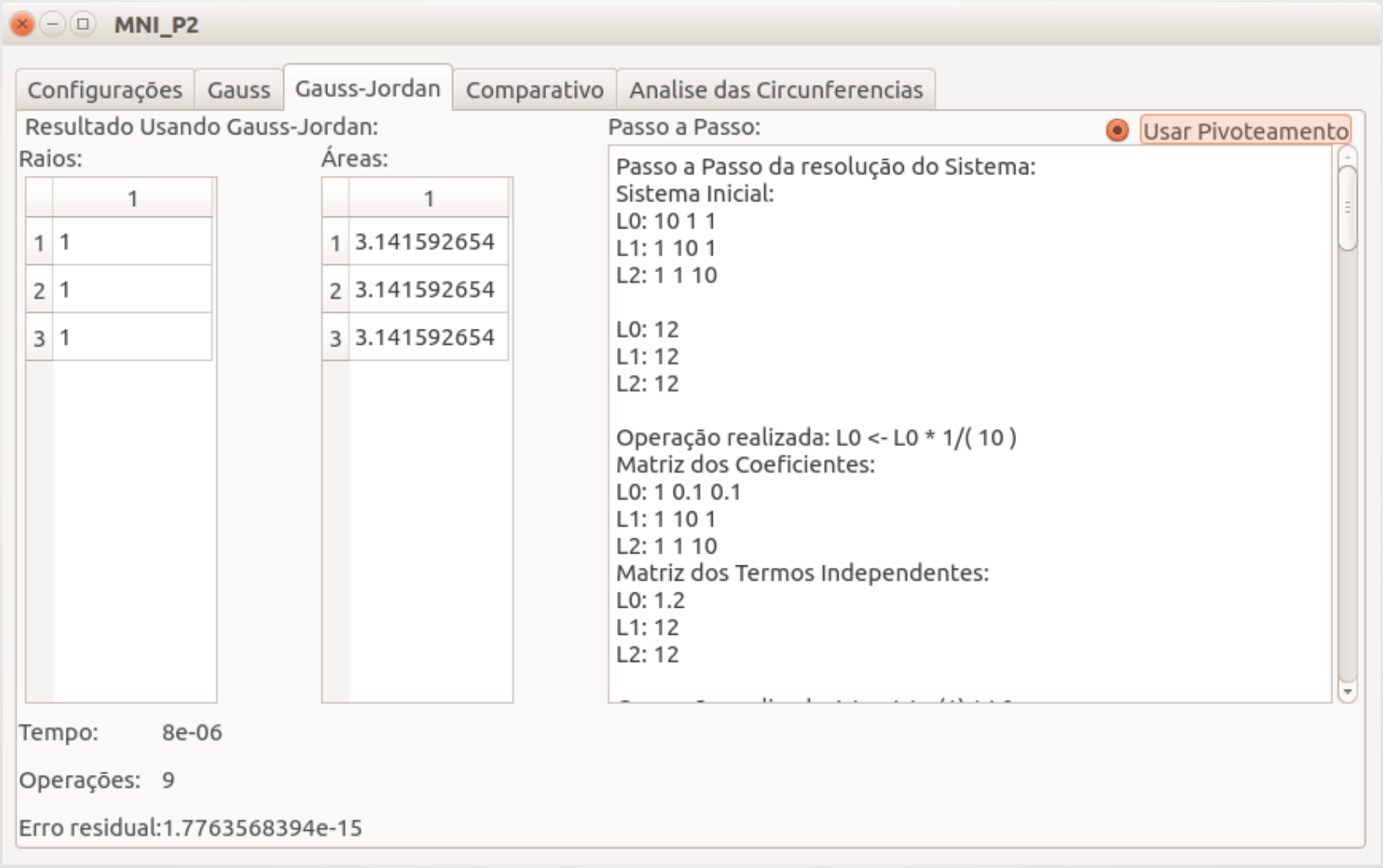
Interface


MNI_P2
Configurações
Gauss
Gauss-Jordan
Comparativo
Analise das Circunferencias
Quantidade de Círculos:
3
Círculos:
1
2
3
1 10 1 1
2 1 10 1
3 1 1 10
* R =
Calcular
D:
1
1 12
2 12
3 12

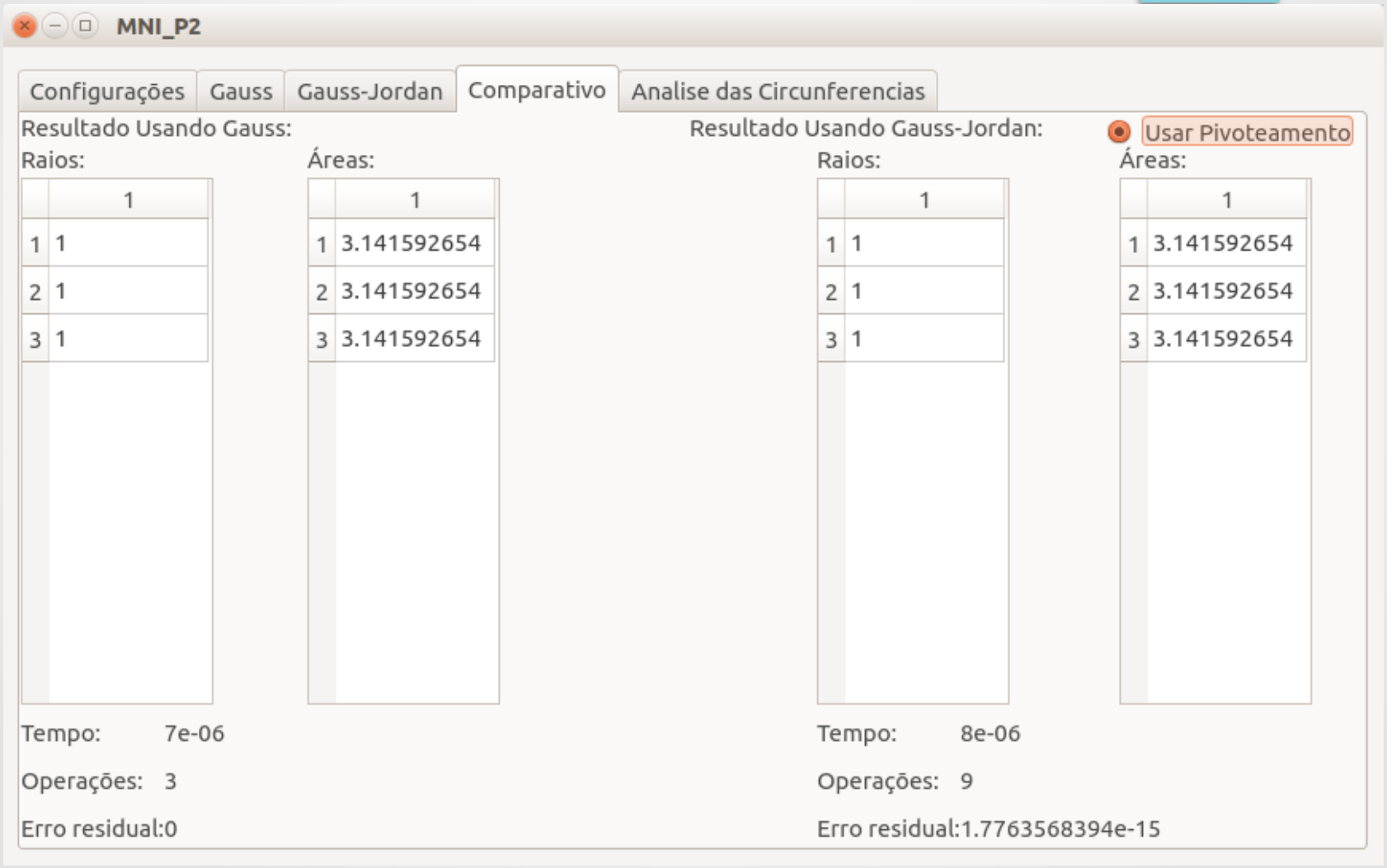
Interface


MNI_P2
Configurações
Gauss
Gauss-Jordan
Comparativo
Analise das Circunferencias
Resultado Usando Gauss:
Raios:
1
1 1
2 1
3 1
Áreas:
1
1 3.141592654
2 3.141592654
3 3.141592654
Passo a Passo:
Usar Pivoteamento
Passo a Passo da resolução do Sistema:
Sistema Inicial:
L0: 10 1 1
L1: 1 10 1
L2: 1 1 10
L0: 12
L1: 12
L2: 12
Operação realizada: L1 <- L1 - (0.1) * L0
Matriz dos Coeficientes:
L0: 10 1 1
L1: 0 9.9 0.9
L2: 1 1 10
Matriz dos Termos Independentes:
L0: 12
L1: 10.8
L2: 12
Tempo: 1e-05
Operações: 3
Erro residual:0

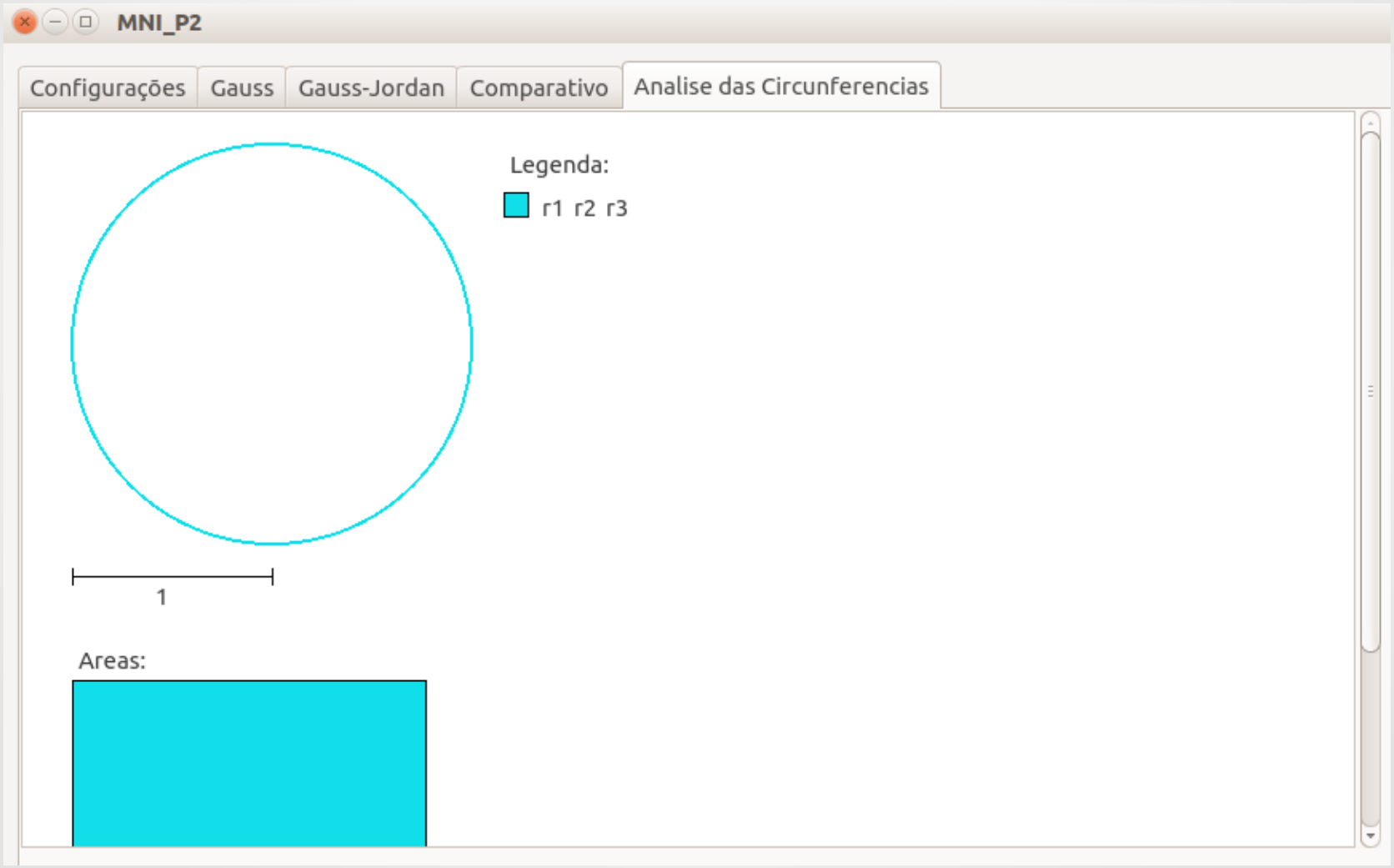
Interface


MNI_P2
Configurações
Gauss
Gauss-Jordan
Comparativo
Analise das Circunferencias
Resultado Usando Gauss-Jordan:
Raios:
1
1 1
2 1
3 1
Áreas:
1
1 3.141592654
2 3.141592654
3 3.141592654
Passo a Passo:
Usar Pivoteamento
Passo a Passo da resolução do Sistema:
Sistema Inicial:
L0: 10 1 1
L1: 1 10 1
L2: 1 1 10
L0: 12
L1: 12
L2: 12
Operação realizada: L0 <- L0 * 1/(10)
Matriz dos Coeficientes:
L0: 1 0.1 0.1
L1: 1 10 1
L2: 1 1 10
Matriz dos Termos Independentes:
L0: 1.2
L1: 12
L2: 12
Tempo: 8e-06
Operações: 9
Erro residual:1.7763568394e-15

Interface


MNI_P2
Configurações
Gauss
Gauss-Jordan
Comparativo
Analise das Circunferencias
Resultado Usando Gauss:
Raios:
1
1 1
2 1
3 1
Áreas:
1
1 3.141592654
2 3.141592654
3 3.141592654
Tempo: 7e-06
Operações: 3
Erro residual:0
Resultado Usando Gauss-Jordan:
Usar Pivoteamento
Raios:
1
1 1
2 1
3 1
Áreas:
1
1 3.141592654
2 3.141592654
3 3.141592654
Tempo: 8e-06
Operações: 9
Erro residual:1.7763568394e-15

Interface


MNI_P2
Configurações
Gauss
Gauss-Jordan
Comparativo
Analise das Circunferencias
Legenda:
r1 r2 r3
1
Areas:

Conclusão

- Dificuldades encontradas:
  - Fazer o slide;
  - Convencer alguns membros a não recriar o código;
    (It was not effective)
- Superações:
  - Ter convencido tais membros a não recriar o código.

Obrigado!