Aluno: Léo Willian Kölln
Matrícula: 0513227-4

Classe MatrizOtimizada

```
public class MatrizOtimizada
{
	private double[] t, r, d, b;

	public MatrizOtimizada(double[][] matriz)
	{
		this.t = new double[matriz.length];
		this.r = new double[matriz.length];
		this.d = new double[matriz.length];
		this.b = new double[matriz.length];

		this.t[0] = 0;
		this.d[matriz.length - 1] = 0;
		for(int i = 0; i < matriz.length; i++)
		{
			this.r[i] = matriz[i][i];
			this.b[i] = matriz[i][matriz.length];
			if(i > 0)
			{
				this.t[i] = matriz[i][i - 1];
			}
			if(i < matriz.length - 1)
			{
				this.d[i] = matriz[i][i + 1];
			}
		}
	}

	private int getDesloc(int i)
	{
		return i - 1;
	}

	public double getT(int i)
	{
		return this.t[this.getDesloc(i)];
	}

	public void setT(int i, double valor)
	{
		this.t[this.getDesloc(i)] = valor;
	}

	public double getR(int i)
	{
		return this.r[this.getDesloc(i)];
	}
```

```
public void setR(int i, double valor)
{
	this.r[this.getDesloc(i)] = valor;
}

public double getD(int i)
{
	return this.d[this.getDesloc(i)];
}

public void setD(int i, double valor)
{
	this.d[this.getDesloc(i)] = valor;
}

public double getB(int i)
{
	return this.b[this.getDesloc(i)];
}

public void setB(int i, double valor)
{
	this.b[this.getDesloc(i)] = valor;
}

public int getN()
{
	return this.b.length;
}

public double getAsMatriz(int i, int j)
{
	if(i == j)
	{
		return this.getR(i);
	}

	if(i == j + 1)
	{
		return this.getT(i);
	}

	if((j == i + 1) && (j <= this.getN()))
	{
		return this.getD(i);
	}

	if(j == this.getN() + 1)
	{
		return this.getB(i);
	}

	return 0;
}

public String getStringMatriz()
```

```
	{
		String resultado = new String("");
		for(int i = 1; i <= this.getN(); i++)
		{
			resultado += "| ";
			for(int j = 1; j <= this.getN() + 1; j++)
			{
				resultado += String.valueOf(this.getAsMatriz(i, j)) + " ";
			}
			resultado += "|\n";
		}

		return resultado;
	}

	public String getStringVetores()
	{
		String resultadoT = new String("T: {");
		String resultadoR = new String("R: {");
		String resultadoD = new String("D: {");
		String resultadoB = new String("B: {");
		String resultado = new String("");
		for(int i = 1; i <= this.getN(); i++)
		{
			resultadoT += String.valueOf(this.getT(i));
			resultadoR += String.valueOf(this.getR(i));
			resultadoD += String.valueOf(this.getD(i));
			resultadoB += String.valueOf(this.getB(i));
			if(i != this.getN())
			{
				resultadoT += "; ";
				resultadoR += "; ";
				resultadoD += "; ";
				resultadoB += "; ";
			}
		}

		resultado = resultadoT +"}\n"+ resultadoR +"}\n"+ resultadoD +"}\n"+ resultadoB
+"}";
		return resultado;
	}
}
```

Classe Aplicação

```
public class Aplicacao
{
	static public void escalonamentoTridiagonal(MatrizOtimizada matriz)
	{
		for(int i = 2; i <=  matriz.getN(); i++)
		{
			double aux = matriz.getT(i) / matriz.getR(i - 1);
			matriz.setR(i, (matriz.getR(i) - aux * matriz.getD(i - 1)));
			matriz.setB(i, (matriz.getB(i) - aux * matriz.getB(i - 1)));
```

```
                matriz.setT(i, 0);
            }
        }

        static public double[] retroSubstituicao(MatrizOtimizada matriz)
        {
            double[] solucao = new double[matriz.getN()];

            solucao[solucao.length - 1] = matriz.getB(solucao.length) /
matriz.getR(solucao.length);

            for(int i = solucao.length - 1; i >= 1; i--)
            {
                solucao[i - 1] = (matriz.getB(i) - (matriz.getD(i) * solucao[i])) / matriz.getR(i);
            }

            return solucao;
        }

        static public String retornaResiduos(MatrizOtimizada matrizO, double[] solucao)
        {
            String resultado = new String();
            double eMax = 0;

            for(int i = 1; i <= matrizO.getN(); i++)
            {
                double res = 0;
                for(int j = 1; j <= matrizO.getN(); j++)
                {
                    res += matrizO.getAsMatriz(i, j) * solucao[j - 1];
                }
                res = Math.abs(matrizO.getB(i) - res);
                if(res > eMax)
                {
                    eMax = res;
                }

                resultado += "Erro X-"+ String.valueOf(i) +": "+ String.valueOf(res) +"\n";
            }
            resultado += "Erro Máximo = "+ String.valueOf(eMax);

            return resultado;
        }

        public static double[][] geraMatrizAtividade(int n)
        {
            double[][] matriz = new double[n][n + 1];

            matriz[0][0] = 1;
            matriz[0][1] = 2;
            matriz[1][0] = -2;

            for(int i = 1; i < n - 1; i++)
            {
                matriz[i][i] = 3;
                matriz[i][i + 1] = 4;
```

```
                matriz[i + 1][i] = -2;
            }

            matriz[n - 1][n - 1] = 7;
            matriz[n - 1][n - 2] = -5;

            matriz[0][n] = 1;
            for(int i = 1; i < n - 1; i++)
            {
                matriz[i][n] = -2;
            }
            matriz[n - 1][n] = 4;

            return matriz;
        }

        public static void main(String[] args)
        {
            int nAtividade = 5; // Tamanho da matriz da atividade

            MatrizOtimizada matrizO = new MatrizOtimizada(geraMatrizAtividade(nAtividade));
            MatrizOtimizada matriz = new MatrizOtimizada(geraMatrizAtividade(nAtividade));

            escalonamentoTridiagonal(matriz);
            double[] solucao = retroSubstituicao(matriz);

            System.out.println("\nMatriz Original:\n"+ matrizO.getStringMatriz() +"Em
Vetores:\n"+ matrizO.getStringVetores());
            System.out.println("\nMatriz Transformada:\n" + matriz.getStringMatriz() +"Em
Vetores:\n"+ matriz.getStringVetores());

            System.out.println("\nResultados:");
            for(int i = 0; i < matriz.getN(); i++)
            {
                System.out.println("X-"+ String.valueOf(i + 1) +": "+
String.valueOf(solucao[i]));
            }

            System.out.println("\nResíduos:\n"+ retornaResiduos(matrizO, solucao));
        }

}
```

Saída

(com n = 5)

```
Matriz Original:
| 1.0 2.0 0.0 0.0 0.0 1.0 |
| -2.0 3.0 4.0 0.0 0.0 -2.0 |
| 0.0 -2.0 3.0 4.0 0.0 -2.0 |
| 0.0 0.0 -2.0 3.0 4.0 -2.0 |
| 0.0 0.0 0.0 -5.0 7.0 4.0 |
Em Vetores:
T: {0.0; -2.0; -2.0; -2.0; -5.0}
R: {1.0; 3.0; 3.0; 3.0; 7.0}
D: {2.0; 4.0; 4.0; 4.0; 0.0}
```

```
B: {1.0; -2.0; -2.0; -2.0; 4.0}

Matriz Transformada:
| 1.0 2.0 0.0 0.0 0.0 1.0 |
| 0.0 7.0 4.0 0.0 0.0 0.0 |
| 0.0 0.0 4.142857142857142 4.0 0.0 -2.0 |
| 0.0 0.0 0.0 4.931034482758621 4.0 -2.9655172413793105 |
| 0.0 0.0 0.0 0.0 11.055944055944057 0.9930069930069929 |
Em Vetores:
T: {0.0; 0.0; 0.0; 0.0; 0.0}
R: {1.0; 7.0; 4.142857142857142; 4.931034482758621; 11.055944055944057}
D: {2.0; 4.0; 4.0; 4.0; 0.0}
B: {1.0; 0.0; -2.0; -2.9655172413793105; 0.9930069930069929}

Resultados:
X-1: 1.1922833649588869
X-2: -0.09614168247944344
X-3: 0.168247944339026
X-4: -0.6742567994939912
X-5: 0.08981657179000631

Resíduos:
Erro X-1: 0.0
Erro X-2: 0.0
Erro X-3: 0.0
Erro X-4: 0.0
Erro X-5: 0.0

Erro Máximo = 0.0

- - - - - - - - - - - -
```

(com n = 100)

```
Matriz Original:
(muita coisa para colocar aqui...)

Matriz Transformada:
(muita coisa para colocar aqui...)

Resultados:
X-1: 0.7886872490764039
X-2: 0.10565637546179808
...
(muita coisa para colocar aqui...)
...
X-99: -0.7628739515058706
X-100: 0.026518606067235412

Resíduos:
Erro X-1: 0.0
Erro X-2: 0.0
...
(muita coisa para colocar aqui...)
...
Erro X-99: 4.440892098500626E-16
Erro X-100: 8.881784197001252E-16
```

```
Erro Máximo = 8.881784197001252E-16
```