UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
DISCIPLINA: Cálculo Numérico para Computação - INE5409
PROFESSOR: Júlio Felipe Szeremeta

TRABALHO 03: INTGRAÇÃO NUMÉRICA UTILIZANDO GAUSS LEGENDRE

Glaucia De Pádua
Lucas Pereira

Florianópolis
Junho de 2011

SUMÁRIO

1 INTEGRAL IMPRÓPRIA POR GAUSS LEGENDRE

O objetivo é resolver uma integral indo de zero até infinito pelo método de Gauss Legendre. Essa é inclusive umas das vantagens do método de Gauss para resolução de integrais comparado com os métodos de Newton, já que nesses últimos não conseguimos resolver integrais impróprias, enquanto que com Gauss conseguimos. A equação a ser resolvida é a seguinte:

$$\int_{0}^{\infty} \frac{dx}{\sqrt{x}(1+x)}$$

Para solver está integral realizamos duas chamadas ao método **GaussLegendreComposto**. A primeira chamada calcula a integral de 0 até ***"a"*** e a segunda chamada calcula a integral de **"a"** até infinito. No caso de se calcular uma integral de ***"a"*** até infinito vimos que com Gauss Legendre podemos fazê-la de 0 até 1/***"a"***. Tento isso em mente percebemos que a melhor escolha para ***"a"*** é o valor 1, já que com isso teremos duas integrais iguais (indo de 0 até 1). Assim, precisamos apenas chamar o método **GaussLegendreComposto** uma vez e multiplicar o resultado por dois.

É importante notar que poderíamos resolver está integral utilizando um valor de ***"a"*** diferente de 1, porém neste caso precisaríamos chamar o método **GaussLegendreComposto** duas vezes, já que teríamos duas integrais diferentes, onde uma vai de 0 até ***"a"*** e a outra vai de ***"a"*** até infinito.

```
Procedimento Principal
    Execute GaussLegendreComposto(0, 1, 10^-8 | I)
    integral = 2*I
    Escreva "Integral: ", integral
Fim Procedimento Principal

Procedimento GaussLegendreComposto
    Ler a, b, e
    Execute GaussLegendre(a, b | I)
    integral1 = I
    erro = 10*e
    k = 2
    Enquanto erro > e faça
        passo = (b-a)/k
        c = a
        integral2 = 0
```

```
        Repita i = 1 até k
          d = c + passo
          Execute GaussLegendre(c, d | I)
          integral2 = integral2 + I
          c = d
        Fim Repita
        erro=|integral1-integral2|/max{|integral1|,|integral2|}
        k=k+k
        integral1 = integral2
    Fim Enquanto
    Retorne integral1
Fim Procedimento GaussLegendreComposto

Procedimento GaussLegendre
    Ler a, b
    m = 4
    Ler ((c_im, t_im), i = 1, m)
    aux1 = (a-b)/2
    aux2 = (a+b)/2
    soma = 0
    Repita i = 1 até m
        xx = aux1*t_im+aux2
        Execute F(xx | fx)
        soma = soma + a_im*fx
    Fim Repita
    integral = aux1*soma
    Retornar integral
Fim Procedimento GaussLegendre

Procedimento F
    Ler x
    fx = 1/(√x(1+x))
    Retornar fx
Fim Procedimento
```

Para resolver o problema descrito anteriormente utilizamos ***m*** igual a 80, porém não conseguimos chegar na precisão 10^{-8}. A maior precisão que conseguimos chegar foi 10^{-7}. Nossa primeira tentativa para resolver esta integral foi utilizando um ***m*** igual a 4 e conseguimos chegar no máximo na precisão 10^{-5}. Somente após isso é que decidimos tentar resolvê-la com um ***m*** maior para tentar chegar ao resultado com uma melhor precisão.

Assim, utilizando ***m*** igual a 80 conseguimos encontrar a solução da integral com precisão 10^{-7} após 30 iterações no método **GaussLegendreComposto**, onde o ***k*** chegou a 1073741824. O resultado encontrado foi: **3,141592**.

O pseudo-códgio acima foi implementado na linguagem Java e seu código encontra-se anexo a este trabalho.

2 INTEGRAL DUPLA POR GAUSS LEGENDRE

Nesta parte o objetivo é solver uma integral dupla por Gauss Legendre em uma precisão desejada. A integral a ser resolvida se encontra abaixo:

$$\int_1^2 \int_3^5 e^{\frac{y}{x}} \, dydx$$

Para resolver esta integral utilizamos um procedimento **GaussLengendreDupla** e um **GaussLegendreCompostoDupla** que encontra o resultado na precisão desejada. Fazendo os testes conseguimos encontrar o resultado da integral com precisão 10^{-13}. Acima disso ainda é possível porém demoraria muito para obter o resultado. O pseudo-códgio dos procedimentos para solver a integral dupla se encontra abaixo:

```
Procedimento Principal
    Execute GaussLegendreCompostoDupla(1, 2, 3, 5, 10^-13 | I)
    integral = I
    Escreva "Integral: ", integral
Fim Procedimento Principal

Procedimento GaussLegendreCompostoDupla
    Ler a, b, c, d e
    Execute GaussLegendreDupla(a, b, c, d | I)
    integral1 = I
    erro = 10*e
    k = 2
    Enquanto erro > e faça
        integral2 = 0
        passo1 = (b-a)/k
        passo2 = (d-c)/k
        f = a
        Repita i = 1 até k
          g = f + passo1
          q = c
          Repita j = 1 até k
               r = q + passo2
               Execute GaussLegendreDupla(f, g, q, r | I)
               integral2 = integral2 + I
               q = r
          Fim Repita
          f = g
        Fim Repita
        erro=|integral1-integral2|/max{|integral1|,|integral2|}
        k=k+k
```

```
        integral1 = integral2
    Fim Enquanto
    Retorne integral1
Fim Procedimento GaussLegendreCompostoDupla

Procedimento GaussLegendreDupla
    Ler a, b, c, d
    m = 4
    Ler ((s_im, t_im), i = 1, m)
    aux1 = (a-b)/2
    aux2 = (a+b)/2
    auy1 = (d-c)/2
    auy2 = (d+c)/2
    soma1 = 0
    Repita i = 1 até m
        yy = auy1*t_im+auy2
        soma2 = 0
        Repita j = 1 até m
          xx = aux1*t_jm+aux2
          Execute F(xx, yy | fxy)
          soma2 = soma2 + a_jm*fxy
        Fim Repita
        soma1 = soma1 + soma2*a_im
    Fim Repita
    integral = aux1*auy1*soma1
    Retornar integral
Fim Procedimento GaussLegendreDupla

Procedimento F
    Ler x, y
    fxy = e^(y/x)
    Retornar fxy
Fim Procedimento
```

Ao tentar solver esta integral dupla na precisão 10^{-13} utilizamos 10 iterações do **GaussLegendreCompostoDupla** onde ***k*** chegou a 1024 e o valor encontrado foi: **41,8480151304563**. O pseudo-códgio foi implementado na linguagem Java utilizando um ***m*** igual a 4 e encontra-se anexo a este trabalho.

ANEXO A – GAUSS LEGENDRE COMPOSTO

```
public class GaussLegendreComposto {
	private static final int ZERO = 0;
	private static final int DOIS = 2;
	private static final int DEZ = 10;

	public static double calcular(double a, double b, double e) {
		double integral1 = GaussLegendre.calcular(a, b);
		double k = DOIS;
		double erro = DEZ*e;
		int iterações = ZERO;
		while (erro > e) {
			double integral2 = ZERO;
			double passo = (b-a)/k;
			double c = a;
			for (int cont = ZERO; cont < k; cont++) {
				double d = c+passo;
				integral2 = integral2 + GaussLegendre.calcular(c,
d);
				c = d;
			}
			erro = Math.abs(integral1-
integral2)/Math.max(Math.abs(integral1), Math.abs(integral2));
			k = k+k;

	System.out.println(k+"\t"+integral1+"\t"+integral2+"\t"+erro);
			integral1 = integral2;
			iterações++;
		}
		System.out.println("Iterações: "+iterações);
		System.out.println("K: "+k/2);

		return integral1;
	}
}
```

ANEXO B – GAUSS LEGENDRE

```
public class GaussLegendre {
	private static final int ZERO = 0;
	private static final double MEIO = 0.5;
	private static final int UM = 1;
	private static final double[] am = {0.34785484, 0.65214516,
0.65214516, 0.34785484};
	private static final double[] tm = {-0.86113631, -0.33998104,
0.33998104, 0.86113631};

	public static double calcular(double a, double b) {
		int m = am.length;
		double aux1 = (b-a)*MEIO;
		double aux2 = (b+a)*MEIO;
		double soma = ZERO;
		for (int cont = ZERO; cont < m; cont++) {
			double xx = aux1*tm[cont]+aux2;
			soma = soma + am[cont]*f(xx);
		}

		return aux1*soma;
	}

	private static double f(double x) {
		return UM/(Math.pow(x, MEIO)*(UM+x));
	}
}
```

ANEXO C – GAUSS LEGENDRE COMPOSTO DUPLA

```
public class GaussLegendreCompostoDupla {
      private static final int ZERO = 0;
      private static final int DOIS = 2;
      private static final int DEZ = 10;

      public static double calcular(double a, double b, double c, double d,
double e) {
            double integral1 = GaussLegendreDupla.calcular(a, b, c, d);
            double k = DOIS;
            double erro = DEZ*e;
            int iterações = ZERO;
            while (erro > e) {
                  double integral2 = ZERO;
                  double passo1 = (b-a)/k;
                  double passo2 = (d-c)/k;
                  double f = a;
                  for (int contA = ZERO; contA < k; contA++) {
                        double g = f+passo1;
                        double h = c;
                        for (int contB = ZERO; contB < k; contB++) {
                              double i = h+passo2;
                              integral2 = integral2 +
GaussLegendreDupla.calcular(f, g, h, i);
                              h = i;
                        }
                        f = g;
                  }
                  erro = Math.abs(integral1-
integral2)/Math.max(Math.abs(integral1), Math.abs(integral2));
                  k = k+k;
                  integral1 = integral2;
                  iterações++;
            }
            System.out.println("Iterações: "+iterações);
            System.out.println("K:"+k/2);

            return integral1;
      }
}
```

ANEXO D – GAUSS LEGENDRE DUPLA

```java
public class GaussLegendreDupla {
      private static final int ZERO = 0;
      private static final int DOIS = 2;
      private static final double[] am = {0.34785484, 0.65214516,
0.65214516, 0.34785484};
      private static final double[] tm = {-0.86113631, -0.33998104,
0.33998104, 0.86113631};

      public static double calcular(double a, double b, double c, double d)
{
            int m = am.length;
            double aux1 = (b-a)/DOIS;
            double aux2 = (b+a)/DOIS;
            double auy1 = (d-c)/DOIS;
            double auy2 = (d+c)/DOIS;
            double integralX = 0;
            for (int contY = ZERO; contY < m; contY++) {
                  double integralY = 0;
                  double yy = auy1*tm[contY]+auy2;
                  for (int contX = ZERO; contX < m; contX++) {
                        double xx = aux1*tm[contX]+aux2;
                        integralY += am[contX]*f(xx, yy);
                  }
                  integralX += integralY*am[contY];
            }

            return aux1*auy1*integralX;
      }

      public static double f(double x, double y) {
            return Math.pow(Math.E, y/x);
      }
}
```