Computational Geometry Lab:
QUADRATURE ON A TRIANGULATION

John Burkardt
Information Technology Department
Virginia Tech
http://people.sc.fsu.edu/∼jburkardt/presentations/cg_lab_triangulation_quadrature_2009_fsu.pdf

March 25, 2024

1 Introduction

In our previous discussion, we considered the problem of estimating the integral of a function $f(x, y)$ over a single triangle T, using a quadrature rule, so that

$$\int_T f(x,y)\,dx\,dy \approx \sum_{1\leq j\leq n} w_j f(x_j, y_j)$$

Now suppose that we have a region $\mathcal{R}$ for which we have a triangulation $\mathcal{T} = \{T_i : 1 \leq i \leq N\}$, with the triangles T_i having disjoint interiors and whose union is $\mathcal{R}$. Suppose that we wish to estimate the integral

$$I(\mathcal{R}, f) = \int_{\mathcal{R}} f(x,y)\,dx\,dy$$

Since $\mathcal{R}$ is identical to the extent of $\mathcal{T}$, and since $\mathcal{T}$ is the disjoint sum of the triangles T_i, an integral over $\mathcal{R}$ is the sum of the integrals over the triangles:

$$I(\mathcal{R}, f) = \int_{\mathcal{T}} f(x,y)\,dx\,dy$$

$$= \sum_{i=1}^{N} \int_{T_i} f(x,y)\,dx\,dy = \sum_{i=1}^{N} I(T_i, f)$$

and, if we now apply a quadrature rule Q to approximate the integral over each triangle, we have:

$$I(\mathcal{R}, f) = \sum_{i=1}^{N} I(T_i, f) \approx \sum_{i=1}^{N} Q(T_i, f)$$

In other words, to approximate an integral over a triangulated region, we may use a quadrature rule to approximate the integral of the function over each triangle in the triangulation and sum the result.

2 Quadrature Rules #1 through #5 for the Unit Triangle

Here are quadrature rules for the unit triangle, with the order N, precision P, weights W, and abscissas (X, Y):

Table 1: Quadrature Rules for the Unit Triangle.

N	P	W	X	Y
1	1	1.000000	0.333333	0.333333
3	2	0.333333	0.500000	0.000000
		0.333333	0.500000	0.500000
		0.333333	0.000000	0.500000
4	3	-0.562500	0.333333	0.333333
		0.520833	0.600000	0.200000
		0.520833	0.200000	0.600000
		0.520833	0.200000	0.200000
6	4	0.109951	0.816847	0.091576
		0.109951	0.091576	0.816847
		0.109951	0.091576	0.091576
		0.223381	0.108103	0.445948
		0.223381	0.445948	0.108103
		0.223381	0.445948	0.445948
7	5	0.225000	0.333333	0.333333
		0.125939	0.797427	0.101287
		0.125939	0.101287	0.797427
		0.125939	0.101287	0.101287
		0.132394	0.059716	0.470142
		0.132394	0.470142	0.059716
		0.132394	0.470142	0.470142

3 Program #1: Quadrature Over a Triangulation

Write a program which estimates the integral of a function over a triangulated region by applying a quadrature rule to each triangle in the triangulation.

Your program should:

- read the number of triangles **T_Num**;
- read the triangles;
- read the order of the quadrature rule **N**;
- read the weights and abscissas of the quadrature rule;
- apply the quadrature rule to each triangle
- print the estimated value of the integral.

Use the following simple triangulation:

```
{ { {2,0}, {2,2}, {0,2} },
  { {1,0}, {2,0}, {1,1} },
  { {0,1}, {1,1}, {0,2} } }
```

This triangulation has "hanging nodes" but that won't be a problem for our calculation.

The function $f(x, y)$ to integrate is

$$f(x, y) = \sqrt{x^2 + y^2}$$

The value of this integral is 5.35637...(Thanks, Mathematica!) Run your program with quadrature rule #3 from the table.

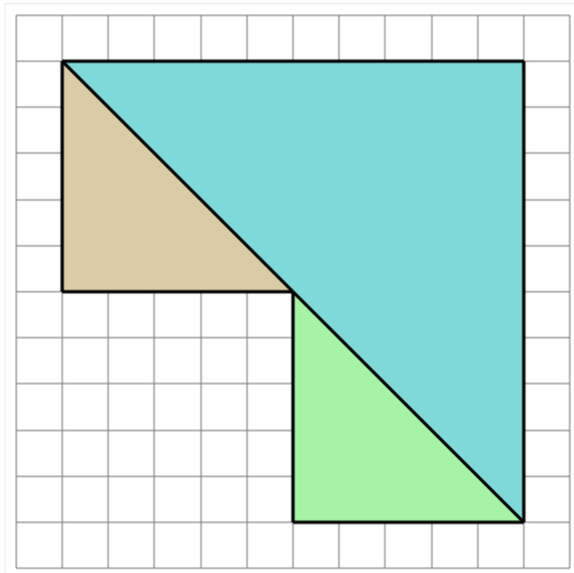

Figure 1: The triangulation to be used for the quadrature calculation.

4 Improving a Quadrature Estimate

The value returned by a quadrature rule is an estimate of an integral. Unless the integrand is a polynomial for which the rule is precise, the estimate will have a certain amount of error.

If our quadrature rule has precision p, and our integrand $f(x,y)$ is smooth enough, we would expect that the error made over triangle Δ_i is of order $C * h_i^{p+1} * \text{Area}(\Delta_i)$, where C is a bound on the integrand derivatives of order $p+1$, and h_i is the length of the longest side or "characteristic length" of Δ_i. Our total error is the sum of all these errors, so it can then be estimated by

$$|\text{Error}| \leq \sum_{i=1}^{N} C * h_i^{p+1} * \text{Area}(\Delta_i) \leq C * h_{max}^{p+1} * \text{Area}(\mathcal{T}),$$

where h_{max} is the maximum value of h_i and Area($\mathcal{T}$) is the total area of the triangulated region.

By looking at the formula for the error, it seems that one way to reduce the error for an integral over a triangulation is to keep the triangulation fixed, but to use a quadrature rule of higher precision $p2 > p$. If our integrand has bounded derivatives of order $p2+1$, then our error estimate will go down because the exponent of h_{max} has increased.

A second approach would be to refine the triangulation; that is, to reduce the value of h_{max} by replace some or all of the triangles by smaller ones. A simple procedure can be used to replace any triangle of characteristic size h by 4 triangles of characteristic size $h/2$. If we refine every triangle in this way, but use the same quadrature rule as before, then p stays the same, but h_{max} has been reduced by a factor of 2 so the new error estimate is divided by 2^p. This procedure may be beneficial if the integrand has limited differentiability, or if we simply don't have access to a quadrature rule of higher precision.

If accuracy is important, it may be be desirable to estimate the size of the error, so that corrective action can be taken, if necessary. A simple way to estimate the error is to carry out the approximation process at least twice, using for the second estimate a rule with better accuracy, either by increasing the exponent p or reducing the characteristic length h_{max}. If we have two such estimates, the difference between them suggests the amount of error in our estimate. If the estimated error seems large, we may need to reduce p or h_{max} yet again, and compare our second and third results.

5 Program #2: Repeated Quadrature Over a Fixed Triangulation

Modify your program from the previous exercise. Approximate an integral using one rule, and then estimate the error by carrying out a second approximation with a better rule and taking the difference.

Your program should:

- read the number of triangles **T_Num**;
- read the triangles;
- read the order of the quadrature rule # 1: **N1**;
- read the weights and abscissas of the quadrature rule # 1;
- compute **Q1**, the first estimate;
- read the order of the quadrature rule # 2: **N2**;
- read the weights and abscissas of the quadrature rule # 2;
- compute **Q2**, the second estimate;
- print **Q1**, **Q2**, and the error estimate | **Q1-Q2** |.

Run your program on the same problem as before, but now compare quadrature rules #1 and #2, then #2 and #3, and so on up to rules #4 and #5. You should expect to see the integral estimates improve, and converge towards the correct value.