Applied Mathematics 2020, 10(1): 1-6
DOI: 10.5923/j.am.20201001.01

A Technique of Discretizing Continuous Data for Programming Adaptive Deterministic Cubature Methods in Moderate Dimensions

Dinh Van Tiep[*], **Tran Thi Hue**

Faculty of International Training, Thai Nguyen University of Technology, TNU, Thai Nguyen, Vietnam

Abstract Cubature methods has been used and well developed to approximate integrals in high dimension for a long time. However, the number of functions evaluations increasing enormously large make a weak point for such methods. In that situation, adaptive cubature is often preferred choice because of a high efficiency and a low cost of calculation it brings back for the approximation problem. However, the data of the integrated regions and of values of the integrand must be continuous due to the theory of integration. It is infeasible to store in computer memory. To deal with this, the discretization of data for both of the region and the function values are used by constructing the net of the potential mesh points. This technique is acceptable since the result we want to extract is only an approximation within a requisite error. The paper aims to present that technique and some remarkable results.

Keywords Adaptive Cubature Program, Approximation Techniques, Discretization, Continuous Data

1. Introduction

1.1. Background and Problem Statement

The algorithm of the Adaptive cubature as well as other methods of numerical integration in high dimension are developed and have been used for a very long time. It definitely has the advantage of accuracy, but also has the drawback of the cost for computation such as the consummation of time and memory since the increasing complexity in calculation. With the help of computers, the implementations of such algorithm in high dimension need to be adjusted. Moreover, in the general case of high dimension, a program for the algorithm have not been provided yet. A primary obstacle of programming the algorithm is the fact that the integrated domain in high dimension with the smooth boundary are described by hypersurfaces which are produced by continuous data of points. Unfortunately, it is impossible to find enough memorized space of hardware to store the data. This makes the algorithm somewhat theoretical and impractical with a tremendous mass of computation. The algorithm itself amends this by a prescribed error, called the tolerance, which requires the accepted estimate error of the approximation must not exceed this tolerance. This reduces quite a lot the number of calculation, but this itself is not enough to make the algorithm feasible to implement because we still need a strategy to store the information we need in the computation process about the values of the integrand on the boundary of the integrated domain.

1.2. Background and Problem Statement

The cubature is the term introduced by Krommer and Ueberhuber [1,2] to indicate the numerical computation of multiple integral. It includes many techniques such as the Monte Carlo and Quasi-Monte Carlo cubature, Bayesian cubature, adaptive cubature. To adaptive cubature, in 2003, Genz and Cools published an algorithm of adaptive cubature for simplices [3] as well as CUBPACK in FORTRAN90. We knew, in high dimensions, the Monte Carlo cubature is the most preferable choice for a numerical integration because of the advantage in dialing with the curse of dimensionality. But this method only yields, in general, a rate of convergence $O(1/\sqrt{n})$ which is quite slow for the number of n sample points. Another shortcoming of this method, especially in the case of no permission for using the probability error, is that the order of convergence is only represented in the randomized terms. That is, the estimate error produced by the Monte Carlo method is not deterministic [4] and it is unsuitable if the cubature problem needs a guaranteed error. That is an indirect reason why we still need to invoke the deterministic cubature, especially in a moderate number of dimensions (say, less than 7). The authors Genz and Malik, Berntsen and Espelid and Genz, Dooren and Ridder presented in

* Corresponding author:
tiepdinhvan@tnut.edu.vn (Dinh Van Tiep)
Published online at http://journal.sapub.org/am

[5,6,7] their works on the adaptive cubature, however, these were developed on a hyper-cube, which is the simplest region in high dimensions. A natural development for an adaptive cubature on more general region in high dimensions is discussed in this paper.

1.3. Contribution

In this paper, we derive an algorithm for the numerical integration over more general regions which are enclosed by measurable surfaces. In some sense, the storage of data describing the region is another challenge. We cannot store all data of this boundary because it composes continuous surfaces. With the need of a strategy to store the data required for calculation in each iterated step, we establish a net of the stored data with the mesh points become denser on the region over which the estimate error is not less than the corresponding tolerance distributed over this region. The technique is the way of discretizing the continuous data of the integrated region. The extracted phase to get the data is organized in an adaptive way in which only the values needed for calculation in the next iteration are invoked. The introduction of this technique for the general domain of dimension n is the main contribution of the paper. In addition, the creation of data points and well organize them in the form of a net so that we can easily access the data also reveals another challenge to program the general algorithm. In the paper, we propose a solution to this problem by a simple approach using the bisection in not too high dimensions. This approach in such dimensions is useful to simplify the structure of the program and this definitely contributes to speed up the computation process.

1.4. Organization of the Paper

The paper is divided into 4 section. The second section is intended to briefly describe the basis of the adaptive cubature for approximating the multiple integrals over the quite general region, so called the non-rectangular hyper-boxes. The idea is developed from two specified methods, the Simpson's rule and the Composite Simpson's rule with 4 subintervals developed for the multiple integrals. The challenges are not only from the curse of dimensionality, but also from the complicated of the integrated region. For this point, as the aforementioned discussion, the complication in depicting the general region can be overcome in a somewhat temporarily acceptable approach in which we store discretely a set of data points that establish the skeleton of the region. We are not going to use these points all at once, but with the accommodation to fit the requirement each checking time. This does not makes the memory of computer working without overloading. Another benefit of this approach is the same as that of a usual adaptive cubature with the flexibility in subdividing the integrated region by adjusting the size of the sub-regions for the next iteration incorporating with the estimate of the error obtained from the use of the two methods applied at this step.

The third section shows off the algorithm of the method which is organized and presented generally in the style of the Matlab language. An illustration for the proper operation of the algorithm is also presented in the case of the dimension 2, the approximation of double integrals. In the last section, we briefly conclude the feature of the algorithm discussed including its advantages and shortcoming.

2. Adaptive Cubature

2.1. Approximation of the Multiple Integrals over Non-rectangular Hyper-boxes

Consider the problem of approximating multiple integral I of a function f which is continuously differentiable up to order 4 on a region $B \subset \mathbb{R}^{n+1}$, a non-rectangular hyper-box

$$B = \{(x_1, x_2, \dots, x_{n+1}) = (\mathbf{x}, x_{n+1}) \in \mathbb{R}^n \times \mathbb{R} \mid F(\mathbf{x}) \le x_{n+1} \le G(\mathbf{x}), \mathbf{x} \in \mathrm{D} \subset \mathbb{R}^n\},$$

F and G are continuous functions on the rectangular hyper-box $D \subset \mathbb{R}^n, D = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_n, b_n]$. Our aim now is to formulate an approximation for

$$I = \int_B f(\mathbf{x}, x_{n+1})\, d\mathbf{y} = \int\limits_D d\mathbf{x} \int\limits_{F(\mathbf{x})}^{G(\mathbf{x})} f(\mathbf{x}, x_{n+1}) dx_{n+1}\,, \quad (1)$$

with $\mathbf{y} = (\mathbf{x}, x_{n+1})$, by using the Simpson's rule and the Composite Simpson's rule with $n = 4$.

Simpson's rule. Firstly, the inner integral of (1) is that of one variable, and treated as

$$\int_{F(\mathbf{x})}^{G(\mathbf{x})} f(\mathbf{x}, x_{n+1}) dx_{n+1}$$

$$= \frac{k(\mathbf{x})}{3}\left[n_1 f\left(\mathbf{x}, x_{n+1}^{\mathbf{x},1}\right) + n_2 f\left(\mathbf{x}, x_{n+1}^{\mathbf{x},2}\right) + n_3 f\left(\mathbf{x}, x_{n+1}^{\mathbf{x},3}\right)\right] - E_0, \quad (2)$$

where $E_0 = \frac{k^5(\hat{\mathbf{x}})}{90} \frac{\partial^4 f}{\partial x_{n+1}^4}\left(\hat{\mathbf{x}}, x_{n+1}^{\hat{\mathbf{x}}}\right), n_1 = n_3 = 1, n_2 = 4,$

$k(\mathbf{x}) = \frac{G(\mathbf{x}) - F(\mathbf{x})}{2}, F(\mathbf{x}) = x_{n+1}^{\mathbf{x},1} < x_{n+1}^{\mathbf{x},2} = x_{n+1}^{\mathbf{x},1} + k(\mathbf{x}) < x_{n+1}^{\mathbf{x},3} = G(\mathbf{x}), \hat{\mathbf{x}} \in D,$ the notation $x_{n+1}^{\mathbf{x},}$ indicates that $x_{n+1}^{\mathbf{x},.}$ depends on the fixed point $\mathbf{x} \in D$.

Theorem 1 (Simpson's rule)

$$I = \frac{h_1 h_2 \dots h_n}{3^{n+1}} \sum_{\substack{1 \le i_1, \dots, i_{n+1} \le 3 \\ 1 \le j_1, \dots, j_{n+1} \le 3}} \left(\prod_{q=1}^{n+1} n_{j_q}\right) \times$$

$$\times\, k\left(x_1^{i_1}, x_2^{i_2}, \dots, x_n^{i_n}\right) f\left(x_1^{i_1}, x_2^{i_2}, \dots, x_{n+1}^{\mathbf{x}, i_{n+1}}\right) + E_1, \quad (3)$$

where $h_i = \frac{b_i - a_i}{2}, x_i^{i_q} = a_i + \left(i_q - 1\right) h_i \ (\forall i, q = 1, \dots, n),$ and $\mathbf{x} = \left(x_1^{i_1}, \dots, x_n^{i_n}\right), Vol(D) = (b_1 - a_1) \dots (b_n - a_n),$

$$E_1 = \frac{Vol(D)}{90}\left[k^5(\bar{\mathbf{x}}) \frac{\partial^4 f}{\partial x_{n+1}^4}(\bar{\mathbf{x}}, x_{n+1}^{\bar{\mathbf{x}}}) + \sum_{i=1}^{n} h_i^4 \frac{\partial^4}{\partial x_i^4}\left(k(\hat{\mathbf{x}}^{(i)}) f\left(\hat{\mathbf{x}}^{(i)}, x_{n+1}^{\hat{\mathbf{x}}^{(i)}}\right)\right)\right],$$

for some $\bar{\mathbf{x}}$ and $\hat{\mathbf{x}}^{(i)} \in D$, and some $x_{n+1}^{\hat{\mathbf{x}}^{(i)}}$ in the interval

$\left(F\left(\hat{\mathbf{x}}^{(i)}\right), G\left(\hat{\mathbf{x}}^{(i)}\right)\right)$.

PROOF: Integrate both sides of (2) over D with respect to $\mathbf{x}$ with the use of Mean Value Theorem for the multiple integral of the term E_0 over D to get

$$I = \frac{1}{3}\int_D \sum_{j=1}^{3} n_j k(\mathbf{x}) f\left(\mathbf{x}, x_{n+1}^{\mathbf{x},j}\right) d\mathbf{x} + \frac{Vol(D)}{90} k^5(\bar{\mathbf{x}}) \frac{\partial^4 f}{\partial x_{n+1}^4}\left(\bar{\mathbf{x}}, x_{n+1}^{\bar{\mathbf{x}}}\right). \quad (4)$$

Reapply Simpson's rule for one variable on $[a_n, b_n]$ for terms in the sum of the first term of (4), denoting that $D = D_{n-1} \times [a_n, b_n]$, to get the first term of (4) to be

$$\frac{h_n}{3^2} \int_{D_{n-1}} \sum_{1 \le j_n, j_{n+1}, i_n \le 3} n_{j_n} n_{j_{n+1}} k\left(\mathbf{x}', x_n^{i_n}\right) f\left(\mathbf{x}', x_n^{i_n}\right) d\mathbf{x}' + \int_{D_{n-1}} Er_n(\mathbf{x}') d\mathbf{x}', \quad (5)$$

where $\mathbf{x}' = \left(x_{i_1}, \dots, x_{i_{n-1}}\right)$, and for some $\tilde{x}_n \in (a_n, b_n)$,

$$Er_n(\mathbf{x}') = \frac{1}{3}\frac{h_n^5}{90} \sum_{j_{n+1}=1}^{3} n_{j_{n+1}} \frac{\partial^4}{\partial x_n^4}\left(k(\mathbf{x}', \tilde{x}_n) f\left(\mathbf{x}', \tilde{x}_n, x_{n+1}^{(\mathbf{x}', \tilde{x}_n), j_{n+1}}\right)\right). \quad (6)$$

Applying Intermediate Value Theorem for (5), and then Mean Value Theorem for the second term of (4), we obtain

$$\int_{D_{n-1}} Er_n(\mathbf{x}') d\mathbf{x}' = \frac{Vol(D) h_n^4}{90} \frac{\partial^4}{\partial x_n^4}\left(k\left(\hat{\mathbf{x}}^{(n)}\right) f\left(\hat{\mathbf{x}}^{(n)}, x_{n+1}^{\hat{\mathbf{x}}^{(n)}}\right)\right),$$

for some $\hat{\mathbf{x}}^{(n)} \in D$. Similarly, reapplying the Simpson's rule consecutively to other $(n-1)$ dimensions of D, we complete the proof.

Composite Simpson's rule with n=4.

Set $q(\mathrm{x}) = \frac{k(\mathrm{x})}{2}, \forall \mathrm{x} \in D, p_i = \frac{h_i}{2}, \forall i = 1,2,\dots,n$. Similar to the above derivation of Simpson's rule, the following result is obtained for the Composite Simpson's rule.

Theorem 2. (Composite Simpson's rule) Let $m_1 = m_5 = 1, m_2 = m_4 = 4, m_3 = 2$ be the coefficients of the Composite Simpson's rule with $n = 4$. We have,

$$I = \frac{p_1 p_2 \cdots p_n}{3^{n+1}} \sum_{\substack{1 \le i_1, i_2, \dots, i_{n+1} \le 5 \\ 1 \le j_1, j_2, \dots, j_{n+1} \le 5}} \left(\prod_{s=1}^{n+1} m_{j_s}\right) \times q\left(z_1^{i_1}, z_2^{i_2}, \dots, z_n^{i_n}\right) f\left(z_1^{i_1}, z_2^{i_2}, \dots, z_{n+1}^{\mathbf{z}, i_{n+1}}\right) + E_2, \quad (7)$$

$z_i^{i_s} = a_i + (i_s - 1)p_i \ (\forall i, s = 1, \dots, n), \mathbf{z} = \left(z_1^{i_1}, \dots, z_n^{i_n}\right)$,

$$E_2 = \frac{1}{16}\frac{Vol(D)}{90}\left[k^5(\bar{\mathbf{z}}) \frac{\partial^4 f}{\partial x_{n+1}^4}\left(\bar{\mathbf{z}}, z_{n+1}^{\bar{\mathbf{z}}}\right) + \sum_{i=1}^{n} h_i^4 \frac{\partial^4}{\partial x_i^4}\left(k(\hat{\mathbf{z}}^{(i)}) f\left(\hat{\mathbf{z}}^{(i)}, z_{n+1}^{\hat{\mathbf{z}}^{(i)}}\right)\right)\right],$$

for some $\bar{\mathbf{z}}, \hat{\mathbf{z}}^{(i)} \in D$, and some $z_{n+1}^{\hat{\mathbf{z}}^{(i)}} \in \left(F\left(\hat{\mathbf{z}}^{(i)}\right), G\left(\hat{\mathbf{z}}^{(i)}\right)\right)$.

2.2. Adaptive Cubature in High Dimensions

Let denote S_1, S_2 the first term in the right-hand side of (3), (7), respectively. Assume that $\bar{\mathbf{x}} \approx \bar{\mathbf{z}}, \hat{\mathbf{x}}^{(i)} \approx \hat{\mathbf{z}}^{(i)}, \forall i = 1,2,\dots,n$. So, $E_1 \approx 16E_2$. Since $I = S_1 + E_1 = S_2 + E_2$, we have $E_2 \approx (S_2 - S_1)/15$. Hence,

$$|S_2 - I| \approx \frac{1}{15}|S_2 - S_1|. \quad (8)$$

This means that we can use the difference between two estimates S_1 and S_2 of I to approximate the error E_2 in the approximation revealed by (7). Therefore, we can design the size of the error to be less than a given tolerance $\varepsilon > 0$. Concretely, if $|S_2 - S_1| < 15\varepsilon$, then we could believe that S_2 approximates I to within ε. Otherwise, if $|S_2 - S_1| \ge 15\varepsilon$, we mostly get wrong when using S_2 to approximate I with the error less than ε. In the latter case, we may get a reasonable approximation by reapplying the above-mentioned procedure on smaller regions (each of such a sub-region has only a size of nearly $1/2^{n+1}$ of that of the original region B, and the expected tolerance for approximation of the integral over that sub-region is only $\varepsilon/2^{n+1}$). Now in such smaller region, we search for $\bar{\mathbf{x}} \approx \bar{\mathbf{z}}, \hat{\mathbf{x}}^{(i)} \approx \hat{\mathbf{z}}^{(i)}, \forall i = 1,2,\dots,n$. Since the size of such sub-regions becomes smaller and smaller, $|\bar{\mathbf{x}} - \bar{\mathbf{z}}| \to 0, \left|\hat{\mathbf{x}}^{(i)} - \hat{\mathbf{z}}^{(i)}\right| \to 0, \forall i = 1,2,\dots,n$, we can eventually reach the target if continuing the procedure. Theoretically, the procedure always succeeds in finding an approximation of I lying to within the given tolerance. However, a computer program cannot execute the procedure in the infinite number of times. Therefore, we set up a limitation for the search by requiring that the level of subdivision (or the number of times in which the procedure is repeated) does not exceed a prior number N. So, the program will reveal the status of failure if N is exceeded. Otherwise, we obtain a desired approximation.

3. Algorithm for the Adaptive Cubature

3.1. Pseudo Matlab Code for the Implementation

We describe the algorithm for the aforementioned procedure in the form of a pseudo-code with the use of Matlab functions. However, the notations and the structure of the repeat loops or the if-condition, the assignment operator are not the same as those in Matlab. They are changed in order to make the program familiar with the mathematic notations, so it may be simpler to analyse.

INPUT region B (including D, and functions $F(\mathbf{x}), G(\mathbf{x})$), function f, tolerance ε, limited level N.

OUTPUT approximation AP of I, or a message announces that the level N is exceeded (that is, the procedure fails!).

```
**Step 1** (% *Initiate the procedure.*)
    $AP := 0;$
    $i := 0;$
    $L_i := 1;$ (% $L_i$ *indicates the current level of subdivision.*)
    $\varepsilon_i := 15\varepsilon;$
    $n_1 := 1;$
    $n_2 := 4;$
    $n_3 := 1;$ (% $n_i's$ *are coefficients of Simpson's rule.*)
    $m_1 := 1;$
    $m_2 := 4;$
    $m_3 := 2;$
    $m_4 := 4;$
    $m_5 := 1;$
    (% $m_i's$ *are coefficients of Composite Simpson's rule.*)
    For $l = 1$ to $n$ do:
        $r_l^i := 1;$
        $s_l^i := 2^N + 1;$
    End do; (%*These are the starting and ending indices of mesh points on each interval* $[a_l, b_l]$.)
    $R_i := 0;$ (%*Initial region is numbered by* 0.)
    For $l = 1$ to $n$ do:
    (%*Set up the mesh points on each dimension* $[a_l, b_l]$.)
            For $j = 1$ to $2^N + 1$ do:
                $X_l^j := a_l + (j-1)(b_l - a_l)/2^N;$
        End do;
    End do;
    For $j_q = 1$ to $2^N + 1, \forall q = 1,2,\dots,n,$ do:
    (%See NOTE below.)
        $F_i(j_1, j_2, \dots, j_n) := F(X_1^{j_1}, X_2^{j_2}, \dots, X_n^{j_n});$
        $G_i(j_1, j_2, \dots, j_n) := G(X_1^{j_1}, X_2^{j_2}, \dots, X_n^{j_n});$
    End do;
**Step 2** While $i > 0$ do steps 3-5.
**Step 3** For $l = 1$ to $n$ do:
            $h_l^i := 0.5\left(X_l^{s_l^i} - X_l^{r_l^i}\right);$
            $t_l^i := 0.5(s_l^i - r_l^i);$
        End do;
    **IF** $(L_i \geq N)$ ***or*** $(t_l^i\ is\ odd)$ **THEN**
    **OUTPUT (**"Level exceeded!"**);**
    **STOP.**
    **ELSE** For $l = 1$ to $n$ do:
            For $j = 1$ to 3 do:
                $x_l^j := X_l^{r_l^i} + (j-1)h_l^i;$
            End do;
        End do;
    $S_1 := 0;$
    $S_2 := 0$ ; (%*Initiate the values for Simpson's and Composite Simpson's rule.*)
    $h := 1;$ (%*Initiate a value for the product of* $h_i$'s.)
    For $l = 1$ to $n$ do:
    $h := h\, h_l;$
    End do;
    For $j_q = 1$ to $3, \forall q = 1,\dots,n,$ do:
    (%*Set up data of mesh points for Simpson's rule.*)
    $k_i(j_1, j_2, \dots, j_n) := 0.5[G_i(r_1^i + (j_1-1)t_1^i, r_2^i + (j_2-1)t_2^i, \dots, r_n^i + (j_n-1)t_n^i) - F_i(r_1^i + (j_1-1)t_1^i, r_2^i + (j_2-1)t_2^i, \dots, r_n^i + (j_n-1)t_n^i)];$

    For $w := 1$ to 3 do:
    $y(j_1, j_2, \dots, j_n, w) := F_i(r_1^i + (j_1-1)t_1^i, r_2^i + (j_2-1)t_2^i, \dots, r_n^i + (j_n-1)t_n^i) + (w-1)k_i(j_1, j_2, \dots, j_n);$

    $S_1 := S_1 + 3^{-n-1}h\, n_{j_1} n_{j_2} \dots n_{j_n} n_w k(j_1, j_2, \dots, j_n) \times$
    $\times f\left(x_1^{j_1}, x_2^{j_2}, \dots, x_n^{j_n}, y(j_1, j_2, \dots, j_n, w)\right);$
    End do; (%*End For-loop for* $w$. *See the convention in NOTE below.*)
        …
    End do; (%*End For-loop for* $j_1$.)
    For $l = 1$ to $5$ do:
            For $j = 1$ to 5 do:
            $z_l^j := X_l^{r_l^i} + 0.5(j-1)h_l^i;$
            End do;
    End do;
    For $j_i = 1$ to $5, \forall i = 1,2,\dots,n,$ do:
    (%*Set up data of mesh points for Composite Simpson's rule. Reference to the convention in NOTE below.*)
    $p_i(j_1, j_2, \dots, j_n) := 0.25[G_i(r_1^i + 0.5(j_1-1)t_1^i, \dots, r_n^i + 0.5(j_n-1)t_n^i) - F_i(r_1^i + 0.5(j_1-1)t_1^i, \dots, r_n^i + 0.5(j_n-1)t_n^i)];$
    For $w := 1$ to $5$ do:
    $q(j_1, j_2, \dots, j_n, w) := F_i(r_1^i + 0.5(j_1-1)t_1^i, \dots, r_n^i + 0.5(j_n-1)t_n^i) + (w-1)p_i(j_1, j_2, \dots, j_n);$

    $S_2 := S_2 + 2^{-n}3^{-n-1}h\, m_{j_1} \dots m_{j_n} m_w k(j_1, j_2, \dots, j_n) \times$
    $\times f\left(z_1^{j_1}, z_2^{j_2}, \dots, z_n^{j_n}, q(j_1, j_2, \dots, j_n, w)\right);$
    End do; (%*End For-loop for* $w$.)
        …
    End do; (%*End For-loop for* $j_1$.)
    For $l = 1$ to $n$ do: (%*Save data before deleting level.*)
        $u_l^1 := r_l^i;$
        $u_l^2 := s_l^i;$
        $u_l^3 := t_l^i;$
        $u^4 := F_i;$
        $u^5 := G_i;$
        $u^6 := \varepsilon_i;$
        $u^7 := L_i;$
    End do;
    $leve\ l_i := u^7;$ (%*Variable level is used to store the sequence of operating level.*)
```

```
If $size(level) > 1$ then
        $count := 0$;
        $j := 1$;
While $j \le size(level) - 1$ do:
        If $level_j == level(end)$ then
        $count := count + 1$;
        End if;
        $j := j + 1$;
End do; (%End while-loop.)
$R_i := 2^{n+1} - \mathbf{mod}(count, 2^{n+1})$; (%To point out which sub-region to be operated in current level.)
End if;
Step 4 $i := i - 1$; (%Delete the current level.)
Step 5 If $|S_2 - S_1| < u^6$ then
        $AP := AP + S_2$;
        $status_{i+1} :=$ "PASS"; (%Variable $status_i$ announces the results obtained when performing the procedure on the current sub-region.)
        Else (%Add one level.)
        $status_{i+1} := "FAIL"$;
        For $K = 1$ to $2^{n+1}$ do:
        $j := \mathbf{de2bi}(K - 1, 'left - msb')$;
        $i := i + 1$; (%Set up data for the sub-region
            $K$-th.)
                For $v = 1$ to $n$ do:
                If $j_v == 0$ then
                $r_v^i := u_v^1$;
                $s_v^i := u_v^1 + u_v^3$;
                Else
                $r_v^i := u_v^1 + u_v^3$;
                $s_v^i := u_v^2$;
                End if;
                End do;
        If $j_{n+1} = 0$ then
        $F_i := u^4$;
        $G_i := 0.5(u^4 + u^5)$;
        Else
        $F_i := G_{i-1}$;
        $G_i := u^5$;
        End if;
        End do; (%To end For-loop for $K$.)
    End If. (%To end Step 5.)
END IF. (%To end IF-condition in early of Step 3.)
Step 6 OUTPUT $AP, R, level, status$;
    STOP.
```

NOTE: In the pseudo-code, we use the convention for *For-loop* of the type "For $l_j = 1$ to … ,$\forall j = 1,2,\dots,n$" to mean that there are n nested *For-loops*:

```
For $l_1 = 1$ to $2^N + 1$ do:
        For $l_2 = 1$ to $2^N + 1$ do:
            …
            For $l_n = 1$ to $2^N + 1$ do:
                Statements relate to $(l_1, l_2, \dots, l_n)$;
            End do; (%End For-loop for $l_n$.)
            …
End do; (%End For-loop for $l_1$.)
```

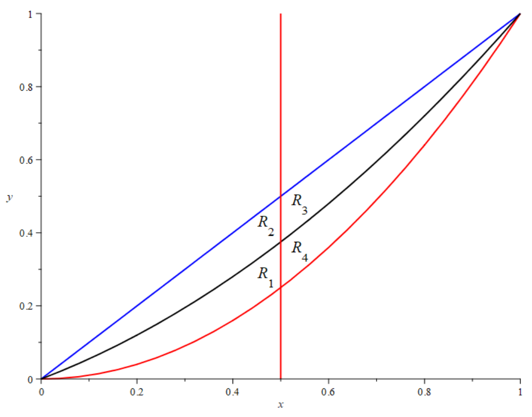

Figure 1. The region B and its 4 sub-regions in the first iteration step

3.2. Numerical Example

Consider the double integral ($\boldsymbol{n = 1}$)

$$\boldsymbol{I = \iint_B (x^2 + xy)dA,}$$

where the non-rectangular region $\boldsymbol{B = \{(x,y) | 0 \le x \le 1, x^2 \le y \le x\}}$. The exact result is $\boldsymbol{I = \frac{11}{120} = 0.0919(6)}$. An implementation use the above algorithm yields an approximation $\boldsymbol{AP = 0.0916621}$ of $\boldsymbol{I}$ to within the given tolerance $\boldsymbol{\varepsilon = 10^{-5}}$. The limit level is $\boldsymbol{N = 4}$. The whole procedure is described in the Table 1.

Table 1. The result produced by the implementation

Level$_i$	*Sub-region R_i*	*Status$_i$*	*Level$_i$*	*Sub-region R_i*	*Status$_i$*
1	0	Fail	2	2	Fail
2	4	Fail	3	4	Pass
3	4	Pass	3	3	Pass
3	3	Pass	3	2	Pass
3	2	Pass	3	1	Pass
3	1	Pass	2	1	Fail
2	3	Fail	3	4	Pass
3	4	Pass	3	3	Pass
3	3	Pass	3	2	Pass
3	2	Pass	3	1	Pass
3	1	Pass	**Procedure is Successful**		

4. Conclusions

The algorithm discussed in the paper dials with a basis problem of numerical analysis with a technique which enables to optimize the implementation of the science computers even when the setting of the multiple integration is quite general, a general iterated regions with a continuous multivariable function refer to as the integrand.

There are the repeat loops presented in the implementation which are nested loops. To realize each such loop in the program we need to makes the verification with respect to the dimension of the multiple integral. That means, each dimension has a particular program to the corresponding algorithm. Therefore, it is preferable to use the implementation for not too high dimension, say less than 7. For these sizes of dimension, the algorithm take much more advantages of a high speed of convergence comparing to the Monte Carlo and Quasi-Monte Carlo cubature which are preferable choices in very high dimensions.

ACKNOWLEGEMENTS

The authors are working in Thai Nguyen University of Technology. This work are supported by the university. We are very grateful for this help and other assistance in reaching the references we need to complete this work.

REFERENCES

[1] Krommer, A. R. and Ueberhuber, C. W. (1998). “Construction of Cubature Formulas”, Computational Integration. Philadelphia, PA: SIAM, pp. 155-165.

[2] Ueberhuber, C. W. (1997). Numerical Computation 2: Methods, Software, and Analysis. Berlin: Springer-Verlag.

[3] Genz, A. C., Cools, R. (2003), “An Adaptive Cubature Algorithm for Simplices”, ACM Trans. Math. Soft., Vol. 26, No. 3, pp. 297-308.

[4] Atanassov, E., Dimov, I. T. (2008). “What Monte Carlo models can do and cannot do efficiently?”, Applied Mathematical Modelling, Vol. 32, pp. 1477-1500.

[5] Genz, A. C. and Malik, A. A. (1980). “An adaptive for numeric integration over an N-dimensional rectangular region,” J. Comput. Appl. Math., Vol. 6, No. 4, pp. 295-302.

[6] Berntsen, J., Espelid, T. O., Genz, A. (1991). “An adaptive algorithm for the approximate calculation of multiple integrals,” ACM Trans. Math. Soft., Vol. 17, No. 4, pp. 437-451.

[7] Dooren, P., Ridder, L. (1976). “An adaptive algorithm for numerical integration over an n-dimensional cube”, J. Comput. Appl. Math., Vol. 2, No. 3, pp. 207-217.

[8] Burden, R. L., Faires, J. D. (2000). Numerical Analysis. (9th Ed.). Brook/Cole, Cengage Learning, Boston (2000).

[9] Chapra, S. C. (2012). Applied Numerical Methods with MATLAB. (3rd Ed.), McGraw-Hill, New York.

[10] Radon, J. (1948). “Zur mechanische Kubatur”. Monatsh. Math. Vol. 42, pp. 286-300.

[11] Caflisch, R. E. (1998). Monte Carlo and quasi-Monte Carlo methods. Acta Numerica, Vol. 7, pp. 1-49.

[12] Sobol, I. M. (1990). “Quasi-Monte Carlo methods”. Progress in Nuclear Energy. Vol. 24, Iss.: 1-3, pp. 55-61.

[13] Pierre, l’E., Randomized Quasi-Monte Carlo: An Introduction for Practitioners. 12th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing (MCQMC 2016), Stanford, U.S. hal-01561550.

[14] Cools, R. (2003). “An Encyclopaedia of Cubature Formulas”. J. Complexity, Vol. 19, pp. 445-453.