# Relatório do Exame de CCI-22

# Projeto de Exame – Métodos de Integração Múltipla


**Aluno:**

Bruno Costa Alves Freire


**Turma:**

T 22.4


**Professor:**

Marcos Ricardo Omena de Albuquerque Máximo


Data:

03/07/2018


**Instituto Tecnológico de Aeronáutica – ITA**
**Departamento de Computação**

# 1. Introdução Teórica

A integração numérica de funções é um problema de amplo interesse para diversas aplicações. Mais notadamente, a formulação desses problemas como *problemas de valor inicial* em equações diferenciais ordinárias é talvez a sua principal utilidade. A rigor, em uma dimensão o problema da integração numérica não é tão desafiador de modo geral, sempre havendo, é claro, casos de funções mal comportadas ou de integrais impróprias.

Contudo, o problema de calcular integrais múltiplas é radicalmente mais complicado que a integração unidimensional, tanto analiticamente quanto numericamente. Do ponto de vista teórico, surge uma nova dificuldade ao lidarmos com funções de mais de uma variável, e outra dificuldade advinda do conjunto em que estamos integrando essas funções. Combinadas, essas duas dificuldades podem tornar o problema severamente não trivial.

Para atacar os possíveis problemas de integração multidimensional, existem algumas alternativas, algumas funcionando melhor para uma determinada classe de problemas do que outras, mas no geral não existe uma abordagem genérica que funcione bem para todos os tipos de problema. Neste trabalho, vamos explorar métodos baseados em integração iterada (Teorema de Fubini) e os chamados métodos de Monte Carlo, que possuem uma abordagem estocástica e estatística.

## 1.1. Métodos de Integração Iterada

Quando tratamos de integração múltipla numericamente, a primeira coisa que devemos tentar é reduzir a dimensionalidade do problema, e se possível, analiticamente. Isso é importante, pois como veremos, a quantidade de chamadas à função integrando cresce exponencialmente com a dimensão do problema. Simetrias na região de integração ou do integrando também podem ser exploradas.

Formalmente, o problema que estamos tentando atacar é o de calcular a integral

$$\int_{\Omega} f(x)\ dx$$

onde $\Omega \subset \mathbb{R}^n$ é o fecho de um conjunto aberto (i.e., de interior não vazio), e f é uma função real de *n* variáveis. Nossa abordagem será escrever a integral como

$$\int_{a_1}^{b_1} dx_1 \int_{a_2(x_1)}^{b_2(x_1)} dx_2 \ldots \int_{a_n(x_1,\ldots,x_{n-1})}^{b_n(x_1,\ldots,x_{n-1})} f(x_1, \ldots, x_n)\, dx_n$$

onde a integral na variável $x_n$ é realizada mais internamente, e a integral em $x_1$ é a mais externa. Note que nessa construção, os limites superior e inferior da integral mais externa são simplesmente números reais $a_1$ e $b_1$, ao passo que os limites de integração das integrais mais internas são funções das variáveis mais externas a elas, de modo a definir o contorno da região $\Omega$. A rigor, nem toda região $\Omega$ pode ser expressa dessa forma, e para integrais impróprias podem haver problemas de convergência para expressar a integral original numa ordem específica de integrais iteradas, mas vamos ignorar essas tecnicalidades aqui.

O procedimento para calcular a integral nesse formato é bem direto. Vamos definir funções intermediárias $I_k$, k = 1, ..., n, que realizam as integrações, da seguinte forma:

$$I_n(x_1, \dots, x_{n-1}) = \int_{a_n(x_1,\dots,x_{n-1})}^{b_n(x_1,\dots,x_{n-1})} f(x_1, \dots, x_n)\, dx_n,$$

$$I_k(x_1, \dots, x_{k-1}) = \int_{a_k(x_1,\dots,x_{k-1})}^{b_k(x_1,\dots,x_{k-1})} I_{k+1}(x_1, \dots, x_k)\, dx_k\,, \forall 1 < k < n,$$

$$I_1 = \int_{a_1}^{b_1} I_2(x_1)\, dx_1.$$

A forma de implementação desse procedimento faz uso de recursão. Chamamos uma rotina de integração unidimensional para avaliar $I_1$, a qual deve chamar essa mesma rotina várias vezes para avaliar $I_2$ ponto a ponto, que por sua vez irá chamar a mesma rotina ainda mais vezes para avaliar $I_3$, e assim por diante. É fácil ver como esse método pode explodir em número de chamadas à função *f*.

Por outro lado, enquanto na integração simples havia um *trade-off* relativamente equilibrado entre atingir uma determinada precisão e economizar nas chamadas à função integrando, aqui temos um desequilíbrio radical: para obter uma dada precisão no resultado, temos que realizar muito mais chamadas à função, numa relação que cresce exponencialmente com a dimensão.

Para visualizar o drama do custo computacional, basta considerar que na integral mais externa, digamos, $I_1$, devemos calcular $I_2$(x) uma quantidade suficiente de vezes para atingir uma dada precisão. Mas qual é o custo de uma chamada a $I_2$? Ora, é o custo de calcular uma integral *n*-1 dimensional com a precisão requerida. Se destrincharmos a pilha de execução desse procedimento até chegarmos a uma integral unidimensional, teremos uma quantidade de chamadas, digamos, inversamente proporcional à tolerância. Subindo novamente a pilha de execução, é fácil ver que o número de chamadas cresce exponencialmente com a dimensionalidade.

Há ainda o problema de como controlar a precisão. Quando realizamos uma integral unidimensional, precisamos amostrar os valores da função integrando em vários pontos, e naturalmente assumimos que essa amostragem é feita sem nenhum erro associado, ou seja, estamos calculando os valores exatos da função em cada ponto. No caso multidimensional, como temos que avaliar nas integrais mais externas funções que são, por sua vez, integrais de outras funções, a precisão que temos na amostragem dessas funções é dada pela precisão que exigimos ao efetuar aquela integração. Dessa forma, temos imprecisão no valor das funções intermediárias sendo integradas, o que irá se propagar através de todas as dimensões. Por essa razão, é recomendado utilizar uma precisão boa nas integrais mais internas, uma vez que estas possuem maior potencial de propagação de erros, e por razões de custo computacional, utilizar uma tolerância maior nas integrais mais externas.

Fica claro pelo exposto acima que a metodologia de integração múltipla via integração iterada (Fubini) sofre com algumas dificuldades, a chamada "maldição da dimensionalidade" que causa crescimento exponencial do custo computacional, a dificuldade em controlar a precisão, além de requerer muito pré-processamento para ser utilizada (precisamos escrever cada um dos limites de integração como funções).

Uma outra abordagem ainda fora do âmbito probabilístico dos métodos de Monte Carlo se baseia em encontrar fórmulas para as chamadas integrais ponderadas, que são expressões do tipo

$$\int_{\Omega} w(x)\, f(x)\ dx,$$

onde w(x) é uma função *peso*. O objetivo dessa metodologia é encontrar, para algumas funções w específicas, fórmulas para aproximar essa integral com base numa combinação linear de valores da função f, como

$$\int_{\Omega} w(x)\, f(x)\ dx \ \approx \ \sum_{i} W_i\, f(x_i).$$

Enquanto para integrais unidimensionais existem várias fórmulas dessa natureza, sob o nome de Gauss e algum coautor (Gauss-Hermite, Gauss-Chebyshev, Gauss-Legendre, Gauss-Laguerre...), para integrais múltiplas essa abordagem é mais escassa, mas alguns desenvolvimentos podem ser encontrados em [2]. Outras abordagens se baseiam na Análise Funcional, e tratam o erro da integração numérica como um funcional e buscam minimizar sua norma.

## 1.2. Métodos de Monte Carlo

A ideia básica do método de Monte Carlo reside na ideia de média de uma função numa região. Seja $f: \mathbb{R}^n \to \mathbb{R}$ e $\Omega \subseteq \mathbb{R}^n$ um conjunto mensurável, temos que a *média de* f *no conjunto* $\Omega$ é dada por

$$\overline{f} = \frac{\int_\Omega f(x)\, dx}{\int_\Omega dx},$$

onde $\int_\Omega dx = V(\Omega)$ é normalmente referido como o volume n-dimensional de $\Omega$. A ideia é que podemos *aproximar* essa *média real* por uma *média amostral*. Se amostrarmos a função *f* em *N* pontos, $x_1, \dots, x_N$, podemos dizer que

$$\overline{f} \approx \langle f \rangle \equiv \frac{1}{N}\sum_{k=1}^{N} f(x_k) \Longrightarrow \int_\Omega f(x)\, dx \approx V(\Omega) \cdot \langle f \rangle,$$

e dessa forma temos um meio de aproximar nossa integral numericamente. Mas quão boa exatamente é essa aproximação? Da estatística, temos que a média real está concentrada em torno da média amostral com uma certa folga, que é controlada pelo desvio padrão,

$$\sigma_f = \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}},$$

de modo que podemos escrever, um pouco mais precisamente,

$$\int_\Omega f(x)\, dx \approx V(\Omega) \cdot \left(\langle f \rangle \pm \sigma_f\right),$$

contudo, ainda não podemos garantir que o desvio padrão nos dará um limite seguro para o erro da integração, mas podemos toma-lo como uma estimativa para o erro cometido.

Tendo isso em mente, um possível algoritmo para calcular a integral de uma função numa dada região consiste em considerar um bloco n-dimensional retangular que contenha essa região, e amostrar a função em pontos aleatoriamente distribuídos nesse bloco. Caso o ponto amostrado esteja fora da região de integração, atribuímos o valor 0 à função. Isso pode ser formulado como a multiplicação pela *função característica* do conjunto de integração, que vale 1 dentro do conjunto e 0 fora dele. Dessa forma, podemos calcular a média do produto da função *f* pela característica de $\Omega$ e multiplica-la pelo volume do bloco retangular, que pode ser calculado diretamente.

Na escolha de um bloco que contenha o conjunto $\Omega$ é importante que este se encaixe o máximo possível de $\Omega$, de modo a deixar o mínimo de volume de folga. Isso porque a amostragem de pontos fora do conjunto faz com que a variância das amostras aumente, sem contribuir com informação relevante acerca da função. De modo geral, pontos onde a função é muito próxima de 0 contribuem pouco para sua média, e prejudicam a variância.

Pensando nisso, uma possibilidade de aperfeiçoar o método de Monte Carlo no caso em que o integrando assume valores muito pequenos na região de integração é realizar uma mudança de variáveis, de modo a tornar o integrando o mais próximo possível de uma função constante. Como consequência, amostraremos mais pontos na região em que a função assume valores maiores, dessa forma aproveitando mais os pontos amostrados. No entanto, há uma certa dificuldade de implementar corretamente essa estratégia, visto que em exemplos testados os resultados divergiam muito do esperado, portanto, não implementaremos essa funcionalidade.

A despeito da técnica de aplicar mudanças de variáveis, há várias outras estratégias desenvolvidas para resolver o principal problema do método de Monte Carlo: sua convergência lenta. A estimativa do erro cometido decresce com o inverso da raiz quadrada da quantidade de amostras, e isso pode se mostrar uma taxa consideravelmente lenta de convergência. O objetivo então é modificar a forma como os pontos são amostrados, visando a *redução da variância*.

Uma primeira ideia é tentar amostrar pontos bem espalhados homogeneamente ao longo da caixa, e em seguida ir “refinando” a distribuição destes, como se estivéssemos gradualmente preenchendo o espaço de integração. Esse tipo de distribuição é chamada de *sequência quasi-aleatória*, pois, em certo sentido, ela é contra um certo tipo de aleatoriedade.

Os tipos de sequências quasi-aleatórias mais conhecidas são as sequências de Halton, as sequências de Sobol, e as sequências conhecidas como Hipercubo Latino. De maneira resumida, as sequências de Halton usam diferentes bases de primos para formar sucessivamente refinamentos de partições do intervalo unitário em cada dimensão. As sequências de Sobol por sua vez utilizam base 2 para formar esses refinamentos, e então reordenam as coordenadas em cada dimensão. O hipercubo latino, por sua vez, consiste em amostrar pontos muito esparsamente, de modo que a cada ponto amostrado, os seguintes não podem ter em comum nenhuma coordenada (é como tentar dispor 8 torres no tabuleiro de xadrez sem que duas estejam em posição de se atacar).

O funcionamento das sequências de Sobol é extremamente sofisticado, e foge ao escopo deste trabalho discuti-lo. Para nossa aplicação, cabe apenas dizer que com essas sequências podemos obter uma redução da variância com $N$ na forma de $(\ln N)^d/N$, que tende a 0 muito mais rápido que $1/\sqrt{N}$. Além disso, o desempenho das sequências de Sobol é melhor quando a função não possui descontinuidades severas (como, por exemplo, a função característica de um conjunto). As outras duas não possuem propriedades de redução de variância tão boas.

### 1.2.1. Redução da Variância com técnicas de amostragem

Duas das mais sofisticadas técnicas de integração baseadas em Monte Carlo, os algoritmos VEGAS [4] e MISER [5], os quais não serão abordados aqui devido a sua complexidade, são baseados em métodos de redução da variância que modificam a forma como os pontos são amostrados. As três técnicas nesse sentido que serão discutidas aqui são *amostragem por importância*, *amostragem antitética* (de antítese, ou reflexão), e *amostragem estratificada*.

A amostragem por importância se baseia em amostrar os pontos na região de integração segundo uma função de distribuição de probabilidade diferente da uniforme, que busca minimizar a variância. Se escrevermos a função integrando, $f(x)$ como produto de uma outra função, $h(x)$, por uma distribuição de probabilidade em $\Omega$ (a região de integração), temos

$$\int_{\Omega} f(x)\,dx = \int_{\Omega} h(x)\cdot p(x)\,dx \approx \langle h\rangle \pm \sqrt{\frac{\langle h^2\rangle - \langle h\rangle^2}{N}},$$

desde que tenhamos

$$\begin{cases} p(x) \geq 0, & x \in \Omega \\ p(x) = 0, & x \notin \Omega \\ \int_{\Omega} p(x)\,dx = 1 \end{cases},$$

de modo que $\langle h\rangle$ é a média aritmética de *N* amostras de *h* tomadas com a distribuição de probabilidade *p*. Para escolhermos a melhor distribuição *p*, que visa minimizar a variância em *h*, vamos escrever $h = f/p$ e escolher *p* que minimize o funcional

$$S = \left\langle \left(\frac{f}{p}\right)^2 \right\rangle - \left\langle \frac{f}{p} \right\rangle^2 \approx \int \frac{f^2}{p}\,dx - \left[\int f\,dx\right]^2$$

Aplicando uma técnica de cálculo variacional, para minimizar *S* temos

$$0 = \frac{\delta S}{\delta p} = \frac{\delta}{\delta p}\left(\int \frac{f^2}{p}\,dx - \left[\int f\,dx\right]^2 + \lambda \int p\,dx\right),$$

onde $\lambda$ é um multiplicador de Lagrange. Dessa equação, temos

$$-\frac{f^2}{p^2} + \lambda = 0 \implies p = \frac{|f|}{\int_{\Omega}|f|dx},$$

onde $\lambda$ foi escolhido de modo normalizar a distribuição *p*. Note que essa distribuição significa que devemos amostrar os pontos na região $\Omega$ de acordo com o módulo de *f*, isto é, amostrar mais onde *f* é mais intensa.

Na prática, no entanto, obter um conhecimento apurado dessa distribuição ótima é um problema circular, pois é equivalente a conhecer com

precisão o valor da integral que queremos calcular. Objetivamos, então, utilizar uma distribuição *p* que torne o integrando mais simples, isto é, mais próximo de uma constante, e que seja fácil de utilizar na prática.

A segunda abordagem, de amostragem antitética, consiste em amostrar simultaneamente um ponto ***x***, e um ponto simétrico a ***x*** com relação aos limites da caixa de integração. Se **a** e **b** são os limites inferior e superior da caixa, e tomarmos **X'** = **a** + **b** – **X**, onde **X** e **X'** são variáveis aleatórias, podemos calcular o valor médio de *f* como

$$\langle f \rangle = \frac{1}{2N} \sum_{i=1}^{N} [f(x_i) + f(x_i')],$$

de modo que a variância de $\langle f \rangle$ será proporcional a $Var[\phi + \phi']$, onde $\phi = f(x_i)$ e $\phi' = f(x_i')$. Teremos então

$$Var[\phi + \phi'] = \mathbb{E}[(\phi + \phi' - \mathbb{E}[\phi + \phi'])^2] = \mathbb{E}[(\phi - \mathbb{E}[\phi] + \phi' - \mathbb{E}[\phi'])^2]$$

$$= Var[\phi] + 2Cov[\phi, \phi'] + Var[\phi']$$

Se *f* possuir alguma simetria de modo que $\phi \approx c - \phi'$, para alguma constante *c*, teremos

$$Cov[\phi, \phi'] = \mathbb{E}[\phi \cdot \phi'] - \mathbb{E}[\phi] \cdot \mathbb{E}[\phi'] = \mathbb{E}[\phi \cdot (c - \phi)] - \mathbb{E}[\phi] \cdot \mathbb{E}[c - \phi]$$

$$= c \cdot \mathbb{E}[\phi] - \mathbb{E}[\phi^2] - c \cdot \mathbb{E}[\phi] + (\mathbb{E}[\phi])^2 = -Var[\phi] \leq 0,$$

de modo que teremos a variância total reduzida, dada por $Var[\phi + \phi'] = Var[\phi] - Var[\phi']$.

A implementação dessa técnica é bem direta e requer pouca alteração do código original.

Trataremos agora da técnica de amostragem estratificada. A ideia básica é dividir o espaço em sub-regiões, e amostrar pontos nessas sub-regiões em quantidade proporcional à variância da função nessas regiões. Basicamente, escolhemos uma partição (retangular, por simplicidade) da região de integração, e aplicamos o método de Monte Carlo simples em cada sub-região isoladamente, visando atender um critério de variância em cada sub-região. Isso fará com que, naturalmente, sejam amostrados mais pontos onde a variância da função for maior. Em seguida, combinamos os resultados de cada região para produzir o resultado final.

### 1.2.2. Outras técnicas na literatura

A literatura de métodos probabilísticos para integração múltipla é muito mais rica que a de métodos baseados em integração iterada. Diversos variantes do método de Monte Carlo foram propostos e aprimorados, utilizando várias técnicas de redução de variância. Dentre elas estão a amostragem estratificada adaptável, amostragem estratificada recursiva, e amostragem por importância adaptável, todas ideias presentes na implementação dos algoritmos VEGAS [6] e MISER [7].

Outra abordagem aparentemente promissora é a chamada quadratura Bayesiana, que se baseia num *processo Gaussiano*, e trata a integração de uma função como um problema de inferência. No entanto, não abordaremos essa técnica aqui.

# 2. Implementação

A implementação de todos os métodos aqui discutidos foi feita por meio de *scripts* MATLAB, o mesmo *software* através do qual foram gerados as figuras e gráficos para análise e comparação dos algoritmos.

## 2.1. Método de Integração Iterada (baseado em Fubini)

Foi implementada no *script* `intmult.m` uma rotina que calcula uma integral multidimensional de uma função *f* de *n* variáveis, com a região de integração sendo definida por um *cell array* de funções anônimas, e um vetor de tolerâncias em cada dimensão de integração, além de uma variável indicando a dimensão *n* da integração. Os parâmetros da rotina estão documentados no próprio arquivo `intmult.m`, por meio de comentários.

Falaremos brevemente da implementação. Expressamos nossa integral como uma integral iterada, em que $I_k$ representa a k-ésima integral, da mais externa para a mais interna. A integração da função *f* é feita de modo análogo à técnica de quadratura adaptativa unidimensional. Começamos calculando a integral mais externa, $I_1$, nos extremos e no ponto médio do intervalo de integração. Esse cálculo é feito invocando a rotina `intmult` recursivamente, passando como argumentos novos *function handles*, que agora representam a função original com o primeiro argumento fixado. Após finalizados esses cálculos, invocamos uma subrotina auxiliar que realiza a bissecção do intervalo de integração, para realizar o refinamento da partição, de modo a atender um critério de tolerância.

O desempenho da implementação feita foi testado no *script* `demointmultiterada.m`, onde são feitos alguns exemplos, os quais servem para ilustrar o funcionamento da rotina de integração, bem como a forma de construir os parâmetros da função adequadamente.

## 2.2. Métodos de Monte Carlo

O método de Monte Carlo simples, com suporte para o uso de sequências quasi-aleatórias, foi implementado no *script* `intmontecarlo.m`.

Outras duas implementações realizadas foram a abordagem de amostragem antitética, e amostragem estratificada, implementadas nos *scripts* `intMCAnti.m` e `intMCEstr.m`.

Por limitações da ferramenta MATLAB, a qual não permite gerar facilmente números aleatórios segundo uma distribuição de probabilidade multivariada qualquer, não foi implementada a amostragem por importância. Contudo, caso essa dificuldade fosse removida, bastaria alterar uma única linha de código no método de Monte Carlo simples.

A implementação do Método de Monte Carlo simples foi feita com suporte a dois modos, via número de iterações, e via uma tolerância relativa a ser atingida. A escolha entre esses dois modos é feita por meio de um parâmetro de texto, e espera um argumento definindo seja a quantidade de iterações ou a tolerância. Caso não seja especificado o critério de parada ou o modo, assume uma configuração padrão. Há ainda a opção de utilizar um gerador de números quasi-aleatórios, com as opções 'halton' e 'sobol'. Por reprodutibilidade, essa mesma opção permite definir uma semente para o gerador de número (pseudo-)aleatórios comum.

O algoritmo inicializa as variáveis de retorno (média, desvio padrão e número de iterações), e calcula o volume n-dimensional da caixa. Em seguida, inicia a geração de pontos aleatórios, de acordo com o método escolhido. Em cada iteração, gera um ponto dentro da caixa, e em seguida acumula o valor da função (multiplicada pela característica do conjunto de integração) e de seu quadrado, e então incrementa o número de amostras e calcula as estimativas de média e desvio padrão. A iteração termina quando o critério de parada (máximo de iterações ou tolerância relativa) for atingido.

As variantes com amostragem antitética e amostragem estratificada seguem a mesma estrutura básica do algoritmo simples, mas com menos opções em sua interface. Na amostragem antitética, a única mudança é que na hora de amostrar um ponto, é criado também o seu par antitético, de modo que a contagem de pontos passa a ser o dobro do número de iterações.

Já na amostragem estratificada, o algoritmo recebe um vetor P com as quantidades de sub-regiões em cada dimensão. Com isso, ele calcula o número de subcaixas, e itera sobre todas elas, invocando o método normal de Monte Carlo para cada uma, exigindo uma tolerância especificada. Em seguida, acumula os resultados de todas as subcaixas para produzir o resultado final. Note que nessa implementação não há a opção de rodar um certo número de iterações (modo 'numIter'), apenas o modo 'relTol'.

## 2.3. Scripts de Análise

Foram criados dois *scripts* para experimentar as implementações, a saber, `demointmultiterada.m` e `demomontecarlo.m`. Nesses *scripts*, são criados exemplos e feitas análises de vários aspectos dos dois métodos.

# 3. Manual de Execução

Os métodos foram implementados nos *scritps* `intmult.m`, `intmontecarlo.m`, `intMCAnti.m` e `intMCEstr.m`. Além deles, foram inclusos os *scripts* auxiliares do método de integração iterada, `intQA.m` e `IntegracaoSimpson.m`. Toda a parte da análise (Resultados e Discussões) foi produzida por meio dos *scripts* principais `demointmultiterada.m` e `demomontecarlo.m`, que produzem todos os gráficos e dados de tabelas do relatório.

A execução dos *scripts* principais é bem direta, basta executá-los no MATLAB, e estes já geram todos os gráficos e exemplos. Detalhes de como utilizar cada um dos métodos implementados (como construir os argumentos a serem passados, como coletar sua saída) podem ser encontrados nos próprios *scripts* em forma de comentários. No entanto, alerta-se para o tempo de execução desses *scripts*, que são relativamente pesados.

# 4. Resultados e Discussões

## 4.1. Método de Integração Iterada

A primeira análise feita foi o cálculo de um exemplo da implementação da integração múltipla iterada. Consideramos a integral que dá a massa de um tronco de paraboloide definido pelas equações

$$x = \left(\frac{y}{2}\right)^2 + z^2; \quad 0 \leq x \leq 9,$$

com densidade dada por

$$\rho(x, y, z) = x \cdot z^4.$$

A integral da massa pode ser escrita como

$$M = \int_0^9 dx \int_{-2\sqrt{x}}^{2\sqrt{x}} dy \int_{-\sqrt{x-(y/2)^2}}^{\sqrt{x-(y/2)^2}} dz\, \rho(x, y, z) = \frac{59049\,\pi}{20}$$

A rotina `intmult` foi utilizada para calcular a integral com parâmetros de precisão $\varepsilon$ = [$10^{-3}$, $10^{-4}$, $10^{-6}$], $\varepsilon'$ = [$10^{-3}$, $10^{-3}$, $10^{-6}$] e $\varepsilon''$ = [$10^{-3}$, $10^{-3}$, $10^{-3}$]. O objetivo é então comparar como se comporta a precisão do método nos dois casos, em comparação com o valor exato da integral, obtido analiticamente. Os resultados obtidos nos três casos foram compilados na tabela 1.

Tabela 1: Erro absoluto e quantidade de pontos da integração com diferentes parâmetros de precisão.

| PRECISÃO | ERRO ABSOLUTO | QUANTIDADE DE PONTOS |
|---|---|---|
| $\boldsymbol{\varepsilon}$ | $3.25928951497 \cdot 10^{-4}$ | 1375169 |
| $\boldsymbol{\varepsilon'}$ | $2.00060032148 \cdot 10^{-3}$ | 837501 |
| $\boldsymbol{\varepsilon''}$ | $2.09110598425 \cdot 10^{-2}$ | 146105 |

Pela tabela 1 fica bem claro que, quando escolhemos valores de precisão para as integrais internas da mesma ordem de grandeza que as integrais externas, o erro cometido é subestimado, ou seja, a precisão no resultado final não é atendida. Isso ocorre porque as integrais externas não enxergam o erro cometido pelas integrais internas, logo, elas carregam esse erro sem corrigi-lo. Por essa razão, é importante que usemos mais precisão nas integrais internas, para evitar que o erro se acumule de tal forma a violar o requisito de precisão no resultado final.

Note que a quantidade de pontos utilizados na quadratura cresce conforme exigimos mais precisão. Com base nessas informações, podemos planejar um requisito de precisão de modo a garantir que este seja atendido, ou ao menos reduzir a propagação de erros. De uma integral interna para outra imediatamente mais externa, podemos estimar a propagação do erro como sendo proporcional à quantidade de pontos usados na quadratura interna. Essa quantidade de pontos vai estar associada à variação da função em determinada região. Uma possível abordagem é variar os requisitos de precisão em cada dimensão e extrapolar o resultado para encontrar o vetor $\varepsilon$ capaz de atingir uma determinada precisão final sem acumular erros das integrações internas.

A segunda análise é para ilustrar a distribuição dos pontos utilizados na quadratura adaptativa multidimensional (no caso, faremos em 2D por questões de visualização) de acordo com o comportamento da função na região de integração. Vamos considerar a integral da função

$$f(x) = \sin(10x) \cdot \cos(6y) \cdot e^{-4(x-1)^2-2(y-2)^2},$$

na região [0, 4]×[0, 4], onde ela assume um comportamento mais oscilatório numa parte da região, e mais suave em outra, conforme pode ser visto no gráfico 3D da figura 1.

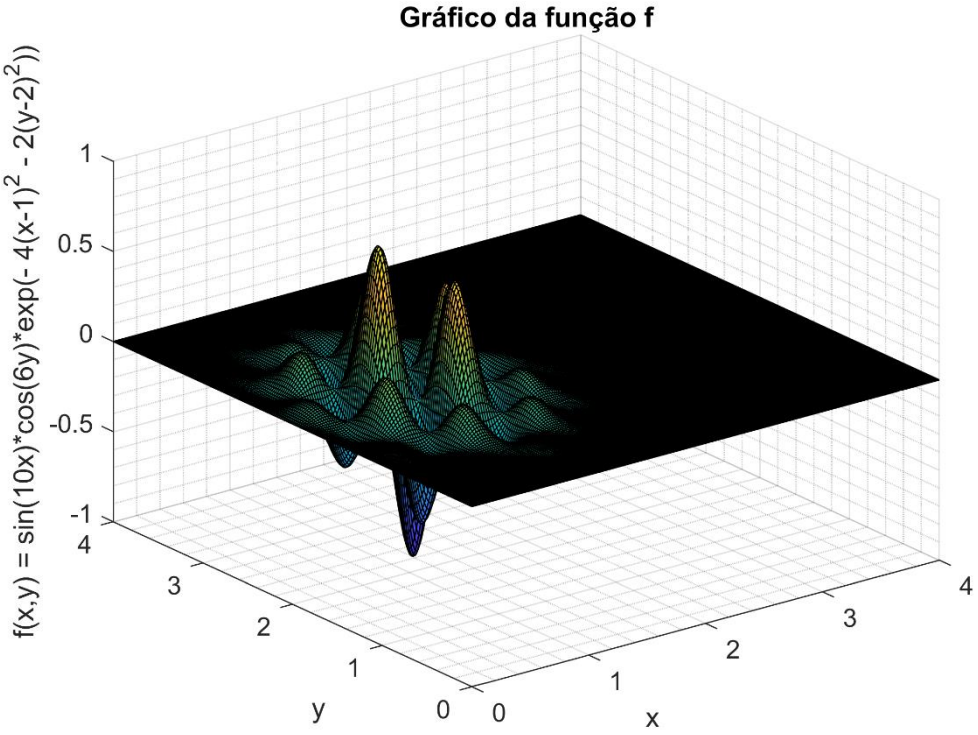


Figura 1: Gráfico da função f (x, y). Note como a oscilação se concentra na parte com x < 2.

Calculamos a integral nessa região e coletamos os pontos utilizados na quadratura adaptativa, para inferirmos sobre sua distribuição. O gráfico com a distribuição dos pontos consta na figura 2.

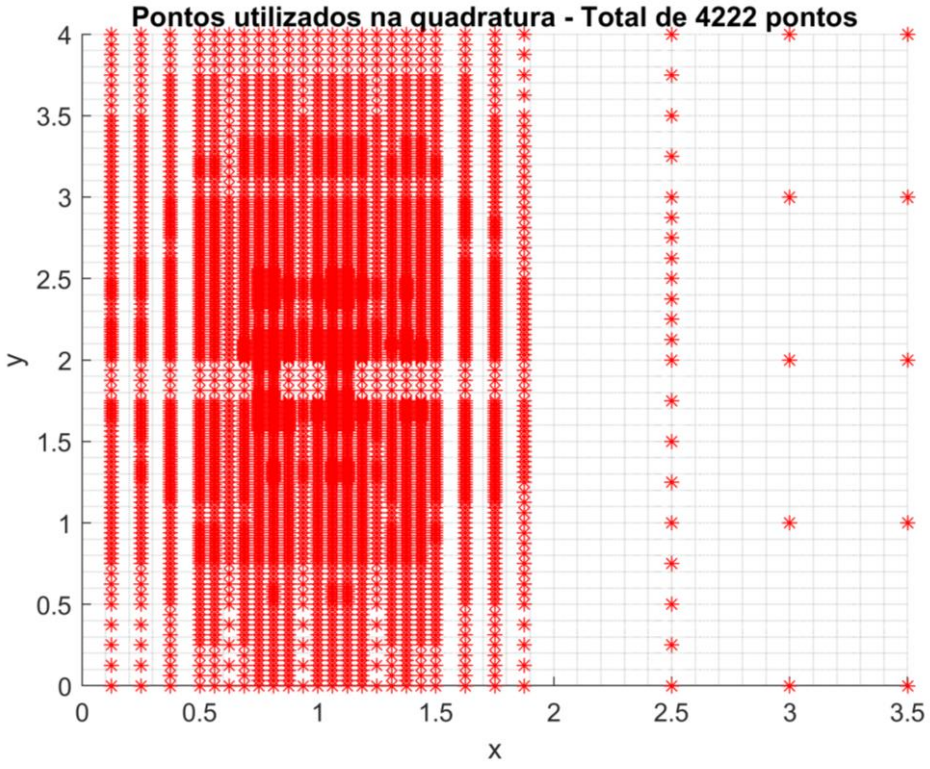


Figura 2: Mapa dos pontos utilizados na quadratura da função f (x, y).

Comparando as figuras 1 e 2, podemos notar que a quadratura da função *f* concentrou fortemente os pontos na região em que a função oscila mais rapidamente. Outro aspecto que podemos observar é que, por conta da integral ser calculada em uma dimensão de cada vez, temos que os pontos se distribuem sempre sobre retas verticais (para cada valor de x), cuja densidade varia conforme a intensidade da variação da função.

A terceira análise que podemos fazer é de grande importância para a aplicabilidade do método de integração iterada, e diz respeito ao crescimento da complexidade com a dimensão da integral. Já discutimos na Introdução que a quantidade de pontos amostrados cresce exponencialmente com a dimensão. Portanto, é de se esperar que o tempo de execução também cresça da mesma forma.

Para estudar essa relação, vamos utilizar uma função simples, num domínio um pouco menos simples. Vamos calcular o volume da bola unitária n-dimensional. A integral pode ser escrita como

$$V_n = \int_{-1}^{1} dx_1 \int_{-\sqrt{1-x_1^2}}^{\sqrt{1-x_1^2}} dx_2 \ldots \int_{-\sqrt{1-x_1^2-\cdots-x_{n-1}^2}}^{\sqrt{1-x_1^2-\cdots-x_{n-1}^2}} 1 \, dx_n,$$

onde o integrando é a função constante igual a 1. Essa integral pode ser avaliada analiticamente, e podemos mostrar que seu valor é igual

$$V_n = \frac{\pi^{n/2}}{\frac{n}{2} \cdot \Gamma\left(\frac{n}{2}\right)}.$$

Calculamos a integral para diversos valores de *n*, variando de 1 a 5, pois acima desse valor o tempo de execução se torna proibitivo. Configuramos o vetor das precisões de modo que se obtenha um resultado acurado, e para cada valor de *n* calculamos a integral 10 vezes para reduzir o ruído na medição de tempo, exceto para *n* > 3, pois para estes valores o tempo de execução já é consideravelmente longo. Em seguida, plotamos os gráficos do tempo de execução e da quantidade de pontos usados na quadratura em função do número de dimensões, nas figuras 3 e 6. Além disso, plotamos a curva do volume de em função de *n* calculada numericamente em comparação com o valor analítico (figura 4), bem como uma análise do erro absoluto cometido (figura 5).

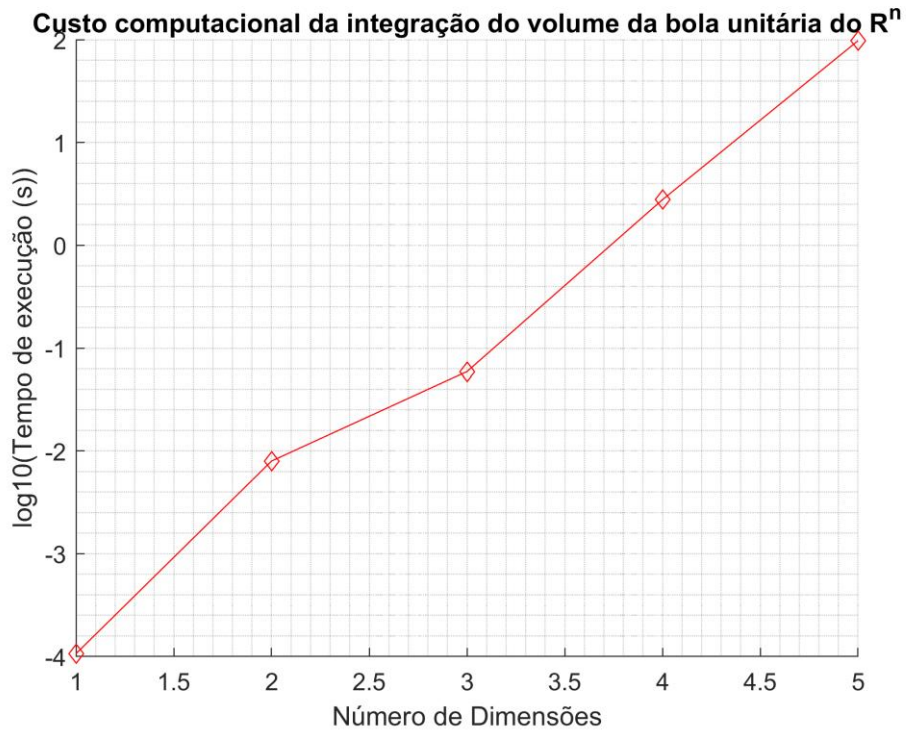


Figura 3: Gráfico dos tempos de execução da integral em função do número de dimensões. Note que a escala é logarítmica.

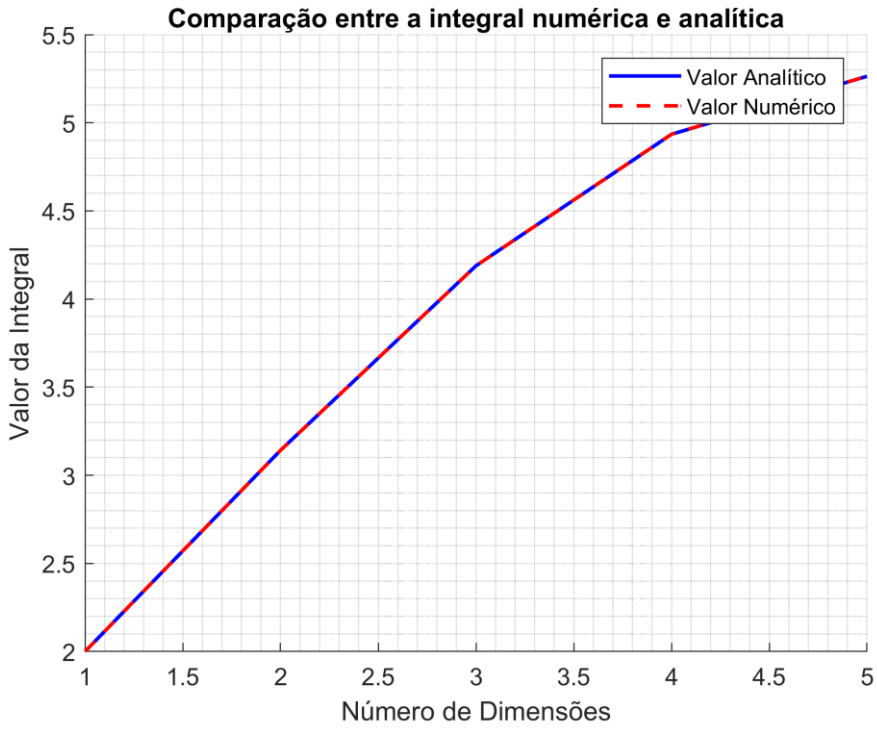


Figura 4: Gráfico do valor analítico da integral junto ao valor numérico calculado, em função da dimensão.

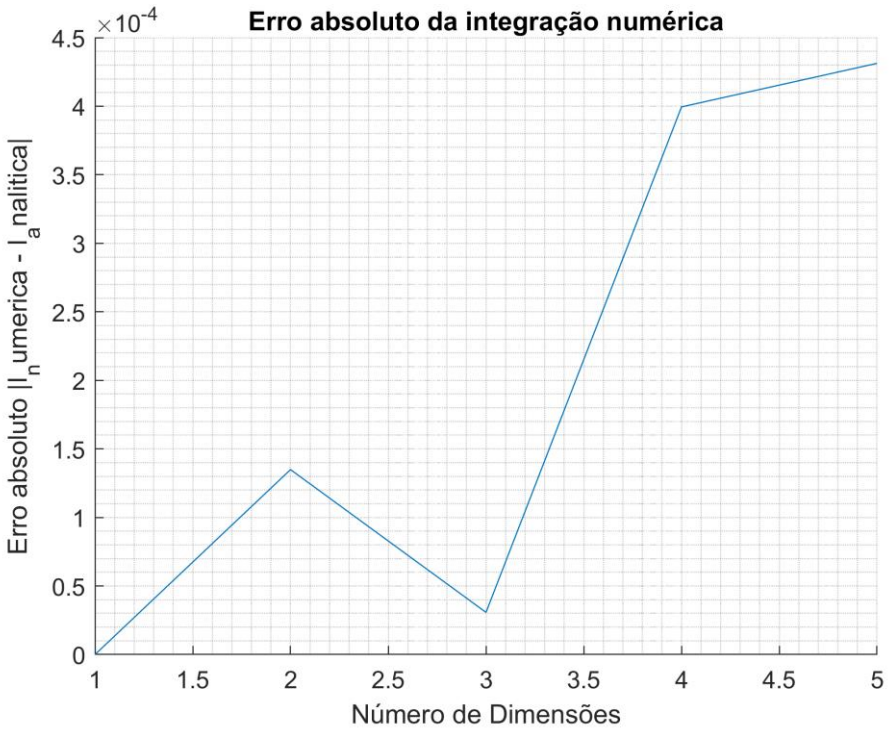


Figura 5: Gráfico do erro absoluto da integração numérica com relação ao valor analítico da integral, em função de n.

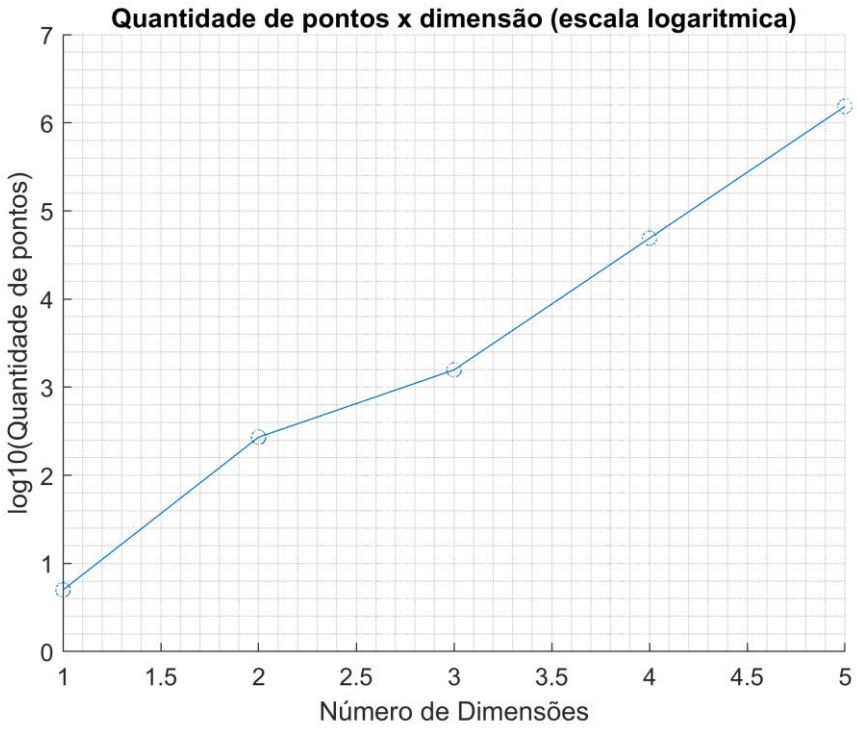


Figura 6: Gráfico da quantidade de pontos utilizados pela quadratura adaptativa em função da dimensão. Escala logarítmica.

A partir das figuras 3 e 6 podemos observar que tanto o custo computacional quanto a quantidade de pontos usados na quadratura adaptativa crescem exponencialmente com o número de dimensões, o que é evidenciado pelo aspecto das curvas nos gráficos em escala logarítmica.

Agora avaliando a precisão do método, vemos que o erro absoluto da integração se comporta de maneira irregular, mas crescente. O vetor de precisões utilizado foi de $10^{-3}$ para a integral mais externa, e uma ordem de grandeza mais baixo para cada integral mais interna. O resultado, conforme podemos ver na figura 4 é razoavelmente satisfatório, dado que as curvas produzidas analítica e numericamente se sobrepõem, e pela figura 5, vemos que o erro não ultrapassou nem metade da precisão final.

Todos os resultados da seção 4.1 foram produzidos através do *script* `demointmultiterada.m`.

## 4.2. Métodos de Monte Carlo

A primeira análise que podemos fazer nos métodos de Monte Carlo é estudar como a média e o desvio padrão de uma integração múltipla evoluem conforme realizamos mais iterações do método. Além disso, podemos comparar o desempenho do método MC simples com suas variantes quasi-aleatórias, utilizando sequências de Halton e de Sobol.

Para analisar isso, vamos considerar a função

$$f(x, y, z) = e^{-(x^2+y^2+z^2)},$$

no conjunto $[-1, 1]^3$, e calcular a integral através das três variações do método de Monte Carlo para diferentes quantidade de iterações. As quantidades de iterações tomadas serão de $10^3$ até $10^5$, pulando de 1000 em 1000. Para termos uma referência do valor da integral, vamos utilizar a função nativa do MATLAB `integral3` para calcular a integral, e usaremos esse valor como referência.

Calculamos então as integrais com as referidas quantidades de pontos, e plotamos os gráficos das figuras 7, 8 e 9. Na figura 7, temos as curvas de evolução do valor da integral calculado pelos três métodos, juntamente com o valor exato e com as margens de desvio padrão de cada método. Na figura 8, temos uma análise do erro absoluto cometido por cada método (com relação ao valor de referência), e na figura 9 temos o gráfico do desvio padrão em função do número de iterações.

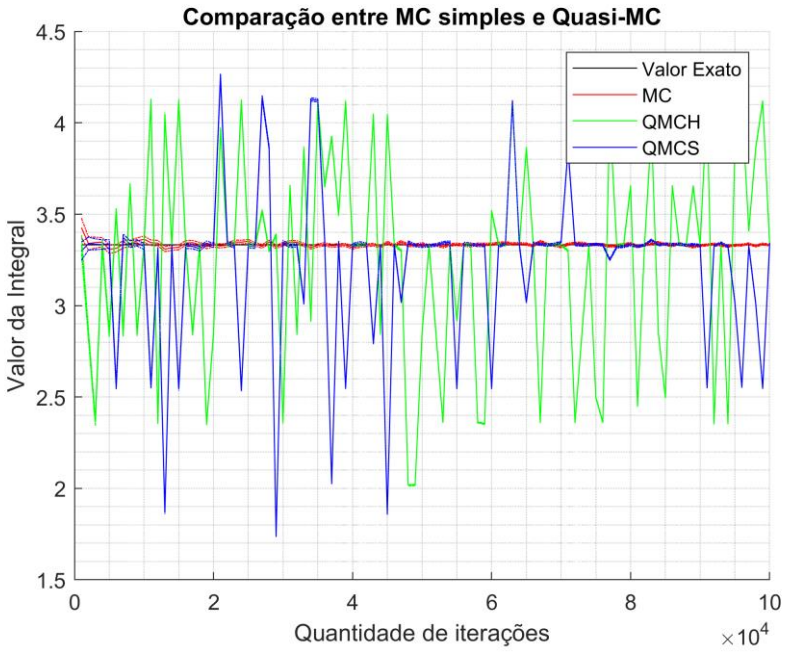


Figura 7: Comparação entre MC, QMC-Halton e QMC-Sobol na integração de uma gaussiana 3D, com diferentes quantidades de iterações.

Da figura 7 podemos observar que as variantes quasi-aleatórias foram desastrosas no cálculo da integral. Uma possível explicação para isso é que, por conta de sua distribuição ser voltada para preencher o espaço de integração de maneira mais uniforme, diferentes quantidades de pontos podem fazer com que os pontos se concentrem mais em uma região ou outra, o que pode *enviesar a média*, fazendo com que a integral calculada seja diferente do valor correto.

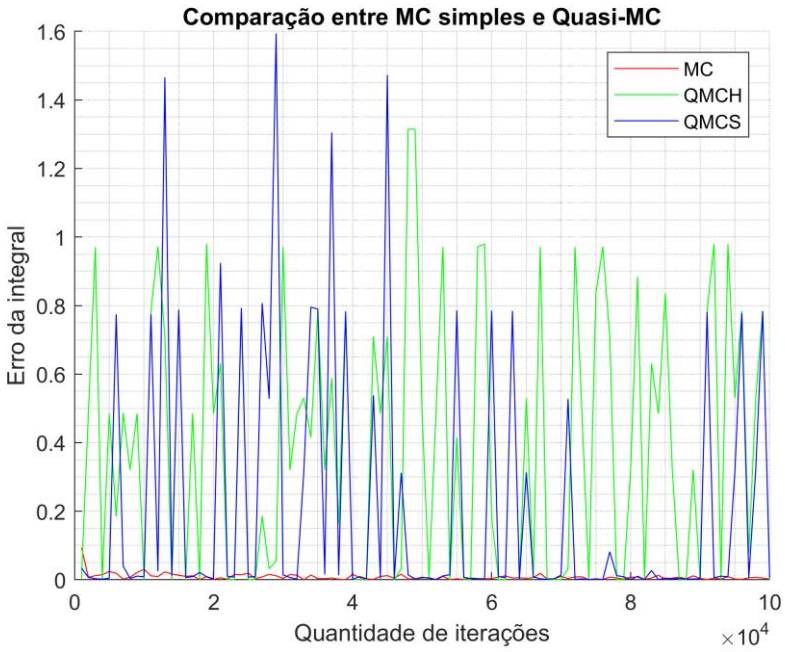


Figura 8: Erro absoluto da integração dos três métodos.

A figura 8 apenas corrobora o que já foi constatado na figura 7. Novamente, a explicação reside na distribuição desigual dos pontos quando se utiliza uma quantidade arbitrária destes, levando ao surgimento de um viés na média, que é responsável pelo erro da integral.

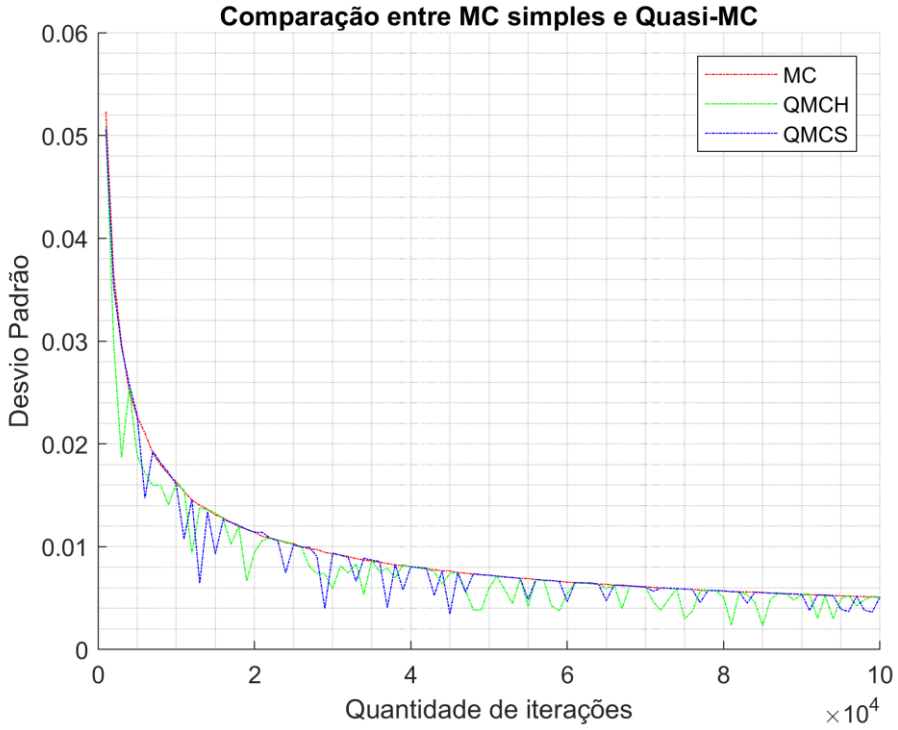


Figura 9: Evolução dos desvios padrão da integração por meio dos três métodos.

Contudo, na figura 9 podemos observar onde as sequências quasi-aleatórias se destacam. A variância (quadrado do desvio padrão) evolui com a quantidade de pontos segundo uma relação $1/\sqrt{N}$, o que pode ser observado na curva vermelha do gráfico. As curvas azul e verde, por sua vez, apesar de terem cometidos erros absolutos consideráveis na integração, apresentam variâncias quase sempre menores que a do método Monte Carlo simples. Isso mostra sua efetividade no quesito de redução da variância. No entanto, para que estas exibam um desempenho aceitável, as quantidades de pontos não podem ser escolhidas ao acaso, mas de maneira bastante seleta. Em [1], é mencionado que as sequências de Sobol preenchem o espaço de integração em épocas de refinamento a cada potência de 2 de número de pontos.

Uma segunda análise que podemos fazer é uma comparação de desempenho entre o método de Monte Carlo com amostragem comum e suas implementações com redução de variância, via amostragem antitética, e amostragem estratificada.

Para essa análise, utilizaremos a função de Rastrigin 3D, dada por

$$R(x) = 3 \cdot A - \sum_{i=1}^{3} [x_i^2 - A \cdot \cos(2\pi x_i)], \qquad A = 10,$$

na região cúbica $[-5, 5]^3$. A referida integral pode ser calculada analiticamente, e tem seu valor dado por 55000.

Vamos então calcular a integral numericamente pelos 3 métodos, configurados para atingir a tolerância relativa de $10^{-2}$. Para o método com amostragem estratificada, precisamos definir uma partição do espaço de integração. Por motivos de custo computacional, escolhemos a partição como sendo 2×2×2, obtendo assim 8 subregiões. Os dados de erro absoluto, desvio padrão, e quantidade de pontos amostrados pelos três métodos foram compilados na tabela 2.

Tabela 2: Dados de erro absoluto, desvio padrão e quantidade de pontos amostrados pelos métodos de Monte Carlo simples, com amostragem antitética e amostragem estratificada, na integração da função de Rastrigin 3D.

| MÉTODO | ERRO ABSOLUTO | DESVIO PADRÃO | NÚMERO DE ITERAÇÕES |
|---|---|---|---|
| **MC SIMPLES** | 726.4291 | 557.1234 | 1013 |
| **MC ANTITÉTICO** | 91.6020 | 549.0615 | 513 |
| **MC ESTRATIFICADO** | 48.7847 | 68.6864 | 66652 |

Podemos notar pela tabela 2 que os variantes com amostragem antitética e estratificada obtiveram grande avanço em relação ao MC simples. Não apenas o erro absoluto diminuiu bastante (note que o valor da integral é da ordem de $10^5$, logo os erros estão dentro da tolerância de $10^{-2}$ especificada), como também o desvio padrão. No caso, o MC antitético obteve um desvio padrão apenas um pouco menor, mas com metade das iterações, e com erro absoluto uma ordem de grandeza menor. Já o MC estratificado obteve um desvio padrão extremamente menor, e ainda mais acurácia, contudo, ao custo de uma quantidade de iterações muito maior.

Por fim, a última análise que podemos fazer é comparar o desempenho do Método de Monte Carlo com o método de integração iterada, no problema do cálculo do volume da bola unitária n-dimensional. O problema é o mesmo, e foram coletadas as mesmas informações, que agora constam nas figuras 10 a 13.

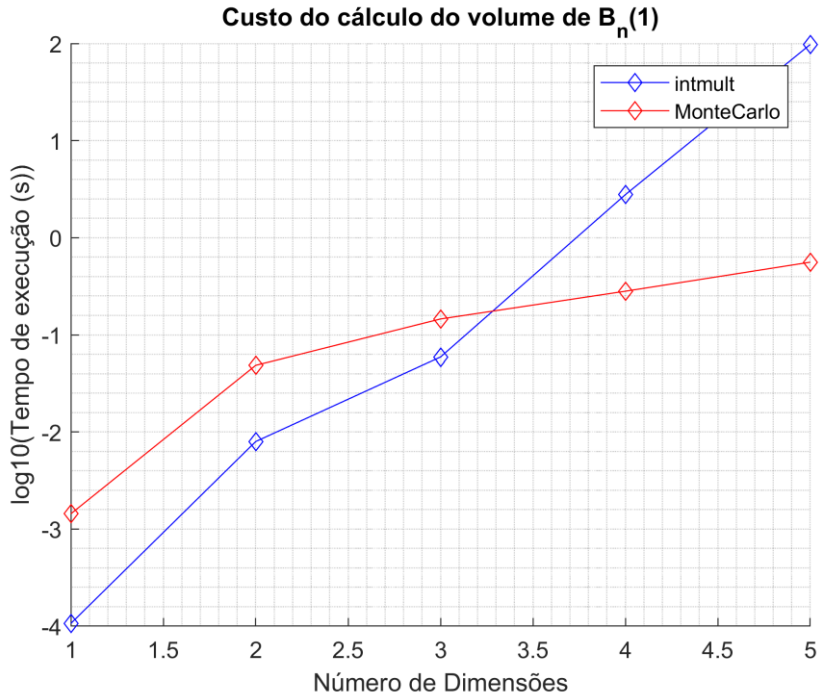


Figura 10: Tempos de execução da integração iterada vs. integração de Monte Carlo no problema do volume da bola n-dimensional.

A precisão requerida no cálculo das integrais foi de $10^{-2}$ apenas, por motivos de logística de tempo. Levando isso em conta, podemos observar que o método de Monte Carlo em geral demora mais para convergir.

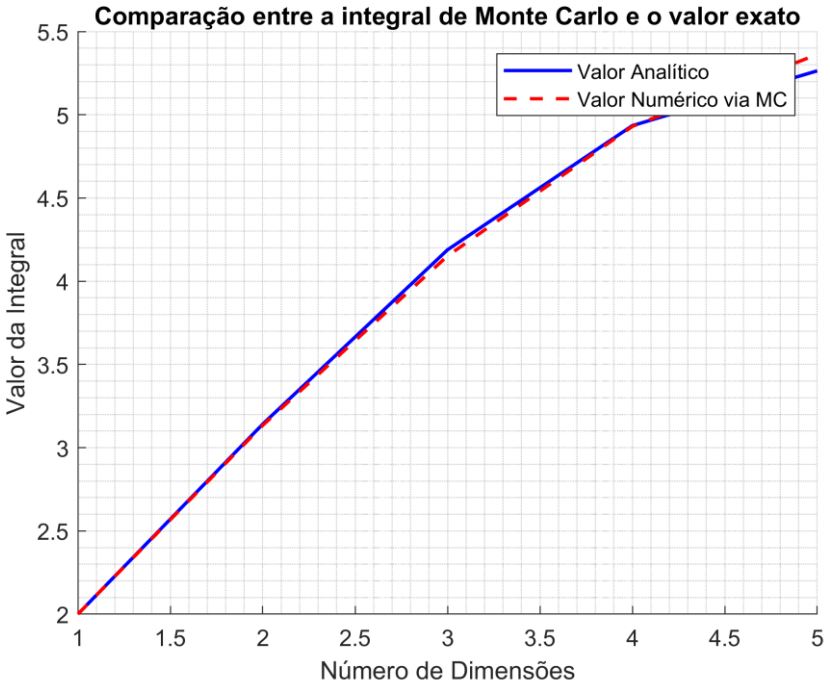


Figura 11: Comparação entre o valor analítico e o valor numérico obtido pela integração de Monte Carlo.

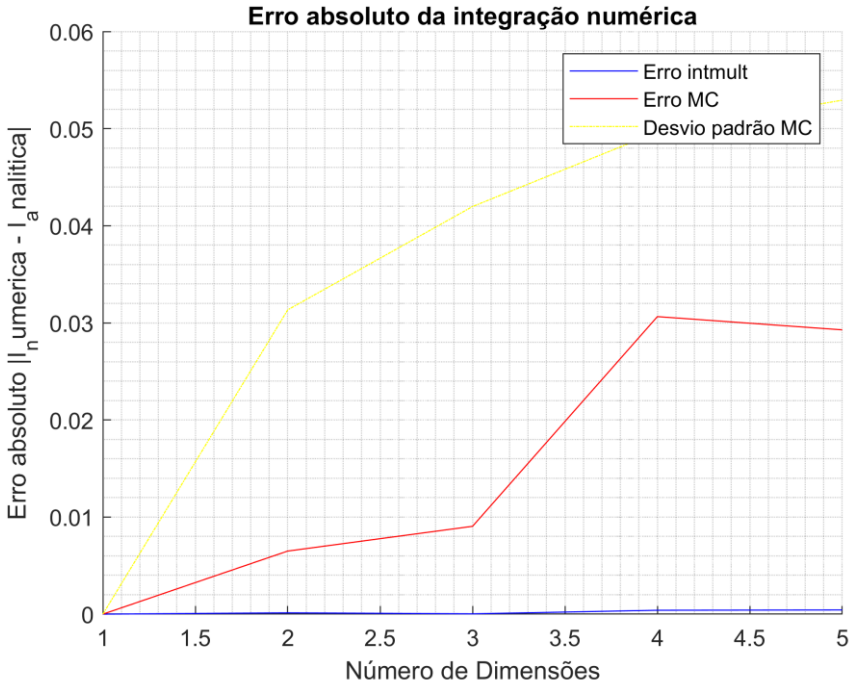


Figura 12: Erro absoluto da integração numérica via integração iterada e via Monte Carlo.

Como o requisito de precisão para o método de Monte Carlo foi menor, é de se esperar que seu erro seja muito maior, contudo, isso não reflete uma falha do algoritmo estocástico em relação ao algoritmo iterado.

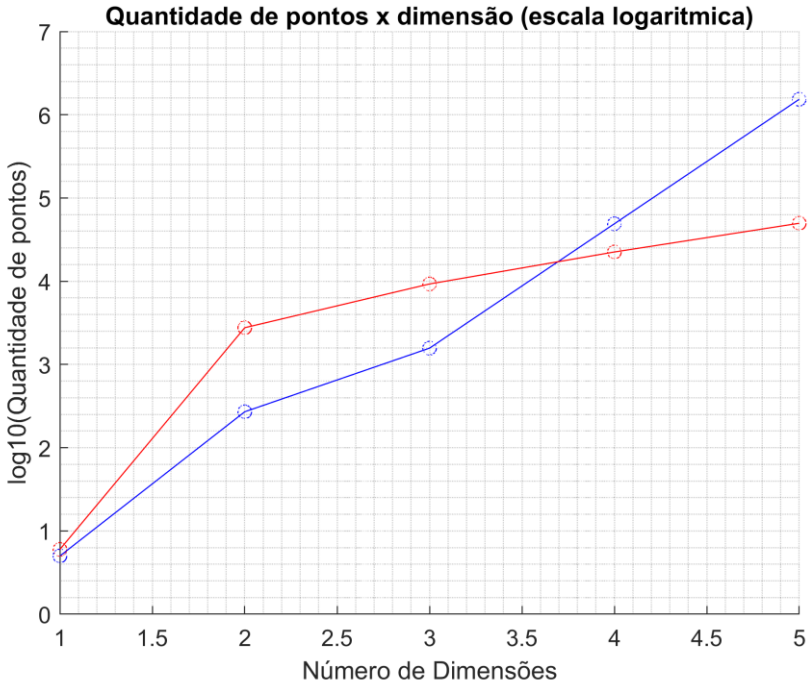


Figura 13: Quantidade de pontos usados na integração pelo método iterado e por Monte Carlo.

Podemos inferir das figuras 10 e 13 que o crescimento do custo computacional e da quantidade de pontos não é exatamente exponencial no caso do método de Monte Carlo, ao contrário do método iterado, que apresenta um comportamento claramente exponencial nesse sentido.

# 5. Conclusão

A primeira conclusão que podemos ter diante de todo o exposto, é que integração múltipla numérica é muito difícil. A respeito do método de integração iterada, vimos que controlar sua precisão é um processo bastante sutil, e que requer cuidado. Além disso, é imprescindível notar que o método escala muito mal no número de dimensões, com seu custo aumentando exponencialmente com a dimensionalidade da integral. Além disso, um resultado não muito surpreendente é a concentração dos pontos da quadratura adaptativa, em torno das regiões de mais oscilação da função.

A respeito dos métodos estocásticos, podemos concluir que são, em geral, a abordagem mais promissora do assunto. A convergência do método de Monte Carlo básico é um dos gargalos do método, o que pode ser combatido com as diversas técnicas de redução da variância conhecidas. As sequências quase-randômicas, por sua vez, são uma técnica que podem ser úteis em alguns casos, mas se não forem bem empregadas, dão resultados desastrosos. As técnicas de amostragem antitética e estratificada dão bons resultados, melhorando razoavelmente o desempenho do Monte Carlo padrão, contudo, um alerta a ser feito a respeito da amostragem estratificada é o risco de explosão do número de partições, que pode levar o custo computacional a crescer exponencialmente, embora resulte em variâncias consideravelmente menores.

Por fim, numa comparação final entre o método de integração iterada e o método de Monte Carlo, vemos que a abordagem estocástica tem suas limitações no que diz respeito à velocidade de convergência. É muito difícil obter com este método erros arbitrariamente baixos. No entanto, a escalabilidade dos métodos de Monte Carlo, com respeito à dimensionalidade da integral, é evidentemente melhor que a do método iterado.

# 6. Referências Bibliográficas

[1] William H. Press et al. *Numerical Recipes – The Art of Scientific Computing*, 3rd Edition. Cap. 4, Seção 8; Cap. 7, Seções 7-9. Cambridge University Press, 2007.

[2] HAMMER, Preston C., WYMORE, A. Wayne. *Numerical Evaluation of Multiple Integrals I*. Disponível em https://www.ams.org/journals/mcom/1957-11-058/S0025-5718-1957-0087220-6/S0025-5718-1957-0087220-6.pdf.

[3] DORNELLES, A. A. **A Simulação de Variáveis Aleatórias e os Métodos Monte Carlo e Quase-Monte Carlo na Quadratura Multidimensional**. Dissertação (Mestrado em Matemática Aplicada) – UFRGS, Caxias do Sul, 2000.

[4] Lepage, G.P. 1980, ***VEGAS: An Adaptive Multidimensional Integration Program***, Publication CLNS-80/447, Cornell University.

[5] Press, W.H., and Farrar, G.R. 1990, ***Recursive Stratified Sampling for Multidimensional Monte Carlo Integration***, Computers in Physics, vol. 4, pp. 190–195.

[6] Código Completo do VEGAS em C++:
http://numerical.recipes/webnotes/nr3web9.pdf

[7] Código Completo de MISER em C++:
http://numerical.recipes/webnotes/nr3web10.pdf