Felipe Fill Cardoso 113.305-7

Integração numérica

O objetivo deste trabalho é demonstrar o funcionamento do algoritmo de integração numérica de Newton-Cotes aplicado a alguns exemplos apresentados em classe.

Regras de integração

O algoritmo foi implementado com três regras diferentes para o cálculo da aproximação da integral:

- Regra do Ponto Médio
- Regra do Trapézio
- Regra de Simpson

Métodos

É possível fazer a aproximação através da aplicação do algoritmo com um número predeterminado de intervalos (implementação clássica) e através do método de quadratura adaptativa.

Estrutura do projeto

O projeto é composto por três classes: `NumericalIntegrationBase`, `NewtonCotes` e `AdaptiveQuadrature`. Elas possuem uma relação de herança entre si, sendo que a primeira é a base para a segunda e ela para a terceira. É possível observar essa relação no diagrama de classes.

Descrição das classes

NumericalIntegrationBase

- `NumericalIntegrationBase(double (*function)(double), float interval_begin, float interval_end, int slicing)`: construtor da classe base, recebe a função a ser integrada, o inicio do intervalo, fim do intervalo e a quantidade de “fatias” em que o intervalo deve ser dividido.
- `virtual ~NumericalIntegrationBase()`: destrutor virtual da classe base.
- `virtual double Integrate()`: método virtual que as classes filhas devem implementar para cálculo da integral.
- `float step_size() const`: devolve o tamanho do passo (subintervalos).

NewtonCotes

- `NewtonCotes(double (*function)(double), float interval_begin, float interval_end, int slicing)`: construtor que recebe os mesmos argumentos da classe pai.
- `~NewtonCotes()`: destrutor da classe.
- `double Integrate()`: implementação do método de integração (virtual na classe pai).
- `double ErrorWithDerivative(double(*derivative)(double))`: devolve o erro com a utilização da função derivada informada.
- `double MidPointIntegration()`: calcula a integral com a regra do ponto médio.
- `double TrapezoidIntegration()`: calcula a integral com a regra do trapézio.
- `double SimpsonIntegration()`: calcula a integral com a regra de Simpson.
- `double MidPointError(double(*derivative)(double))`: calcula o erro da regra do ponto médio.
- `double TrapezoidError(double(*derivative)(double))`: calcula o erro com a regra do trapézio.
- `double SimpsonError(double(*derivative)(double))`: calcula o erro com a regra de Simpson.
- `void set_integration_type (NewtonCotesIntegrationType type)`: configura o tipo de integração (ponto médio, trapézio ou Simpson).

AdaptiveQuadrature

- `AdaptiveQuadrature(double (*function)(double), float interval_begin, float interval_end)`: construtor que recebe os mesmos argumentos da classe pai.
- `~AdaptiveQuadrature()`: destrutor da classe.
- `double Integrate()`: implementa a integração (sobreescreve o método da classe pai).
- `void set_threshold(float threshold)`: configura a tolerância do método de integração.

Resultados

Resultados apresentados após a execução do algoritmo (código de teste presente em `main.cc`):

Método “Clássico”

Exercício A:

- MidPoint: 1.71757 with error: 0.126042
- Trapezoid: 1.71971 with error: -0.252083
- Simpson: 1.71828 with error: -0.000766754

Exercício B:

- MidPoint: 0.788103 with error: -0.126042
- Trapezoid: 0.77613 with error: 0.252083
- Simpson: 0.784112 with error: 0.00230026

Exercício C:

- MidPoint: 0.747131 with error: -0.252083
- Trapezoid: 0.746211 with error: 0.504167
- Simpson: 0.746824 with error: -0.00920105

Método de Quadratura Adaptativa

Exercício A:

- MidPoint: 1.71828
- Trapezoid: 1.71828
- Simpson: 1.71828

Exercício B:

- MidPoint: 0.785398
- Trapezoid: 0.785398
- Simpson: 0.785398

Exercício C:

- MidPoint: 0.746824
- Trapezoid: 0.746824
- Simpson: 0.746824