Relatório de Implementação: Séries de Fourier Utilizando Python e C

Alec A. Gonçalves, Manoela V. Moura, Pablo A. Ramalho e Thomas A. S. Viatroski

1

Resumo. *Este documento apresenta a relatoria técnica focada no capítulo de implementação em linguagem C para o desafio de desenvolvimento de Séries de Fourier. A arquitetura do projeto foi estruturada para utilizar duas linguagens de programação com vocações distintas, em que o Python oferece a interface gráfica com o usuário e a linguagem C é encarregada de prover o serviço de cálculo subjacente.*

Sumário

1. Linguagem C

1.1. Justificativa

A escolha da linguagem C para atuar como o motor de cálculo baseia-se fundamentalmente em seu alto desempenho e eficiência em operações de nível mais baixo. Ela é classificada como uma linguagem compilada, o que significa que o código é convertido

por uma cadeia de ferramentas diretamente em binários de máquina antes da execução, conferindo a ela um tempo de execução significativamente menor e um melhor desempenho temporal em comparação com linguagens interpretadas [Ting 2022].

Em análises comparativas focadas na resolução de problemas matemáticos e computacionais clássicos, a linguagem C demonstra consistentemente o melhor desempenho e o menor tempo de execução absoluto quando avaliada contra linguagens de mais alto nível, como Java e Python, sendo a escolha tecnicamente mais robusta para serviços em que tempo e processamento são fatores críticos [Seabra et al. 2018].

A linguagem C se encaixa perfeitamente na proposta ao assumir exclusivamente o processamento iterativo e pesado exigido pela aproximação de sinais via Séries de Fourier. A implementação se beneficia fortemente do aproveitamento otimizado da precisão de ponto flutuante em C, o que garante a precisão matemática fundamental para o cálculo trigonométrico da somatória.

1.2. Aplicação

Como a arquitetura requer comunicação com o Python, o uso isolado de módulos C provou-se vantajoso, pois a biblioteca padrão *ctypes* do Python permite empacotar *wrapping* e invocar funções de bibliotecas dinâmicas puramente escritas em C de maneira flexível e sem a necessidade de compilar intermediários complexos [Kloss 2009].

Como exemplo direto desta aplicação, o ambiente de cálculo em C foi isolado nos arquivos *calculo.c* e *calculo.h*, englobando funções independentes como $squareWave(intN, intA, doublet)$, $sawtoothWave(intN, intA, doublet)$ e $triangleWave(intN, intA, doublet)$. Cada uma destas funções processa o instante de tempo t, a amplitude A e a precisão em número de harmônicas N, aplicando recursos nativos da biblioteca $math.h$ e constantes de alta precisão, como PI, para computar o resultado em formato double.

Uma vez compilado em formato de biblioteca compartilhada nativa como *libcalculo.so* ou *libcalculo.dll*, o Python mapeia os tipos de dados originais *ctypes.c_int* e *ctypes.c_double*, e invoca o módulo em C para varrer um vetor de tempo, alocando a linguagem C estritamente na sua vocação de processamento algorítmico.

2. Linguagem Python

2.1. Justificativa

A escolha da linguagem Python justifica-se por sua natureza de alto nível, sintaxe intuitiva e pelo gerenciamento automático de memória (coleta de lixo), características que conferem menor complexidade de desenvolvimento e alta produtividade em comparação com linguagens de médio e baixo nível [Ting 2022, Seabra et al. 2018]. Embora Python apresente tempos de execução superiores e menor eficiência em processamentos matemáticos brutos quando comparado ao C, a linguagem destaca-se excepcionalmente em tarefas que exigem alta abstração estrutural, suporte nativo a estruturas de dados complexas e rápida prototipagem [Seabra et al. 2018, Ting 2022].

No escopo da proposta da arquitetura, o Python se encaixa perfeitamente ao assumir o papel de orquestrador de alto nível, encarregando-se da interface gráfica e da

manipulação de matrizes de dados, com as bibliotecas *PyQt5* e *NumPy*), enquanto delega o processamento algorítmico pesado para a linguagem C. O uso da biblioteca *ctypes* permite ao Python carregar bibliotecas dinâmicas compiladas nativamente, como arquivos com extensão $.so$ no Linux/macOS ou $.dll$ no Windows, e invocar as funções matemáticas em C de maneira flexível.

2.2. Aplicação

Como exemplo prático de implementação do projeto, o código no arquivo interface.py invoca o motor de cálculo utilizando o comando $ctypes.CDLL(".\/libcalculo.so")$ ou $libcalculo.dll$ dependendo da plataforma. Para que a execução ocorra sem erros de *segmentation fault*, as assinaturas das funções são declaradas explicitamente antes do uso, mapeando a correta conversão de tipos de dados. Um exemplo direto é a definição $fourier_lib.squareWave.argtypes = [ctypes.c_int, ctypes.c_int, ctypes.c_double]$, que instrui o interpretador Python de que a função em C necessita de dois parâmetros do tipo inteiro e um de dupla precisão, definindo também o retorno via $restype = ctypes.c_double$.

A aplicação utiliza as bibliotecas NumPy e Matplotlib sob o backend interativo *qt5agg*. O sistema instancia um vetor temporal de alta resolução, como em $np.linspace(0, 4*np.pi, 2000)$ e itera sequencialmente sobre estes valores. Em cada etapa do laço de repetição, as variáveis em Python são empacotadas para os tipos de C correspondentes, por exemplo, $ctypes.c\_double(t)$ e passadas para as funções matemáticas da libcalculo, que retornam a amplitude $f(t)$ calculada. Esta matriz de dados resultante é utilizada pela Matplotlib para plotar gráficos estáticos das séries harmônicas com possibilidade de exportação para arquivos físicos $--savearquivo.png$ ou, alternativamente, produzir uma renderização contínua e dinâmica na tela do usuário utilizando o recurso de animação fluida da classe FuncAnimation a uma taxa de atualização de 20 milissegundos.

3. Biblioteca CTYPES

A biblioteca ctypes consiste em um pacote fundamental padrão disponibilizado de forma nativa pela linguagem Python (desde a versão 2.5), cujo principal propósito tecnológico é permitir a invocação direta de procedimentos e funções embarcadas dentro de bibliotecas de vínculo dinâmico ou compartilhado [Kloss 2009].

3.0.1. Tipagem Cruzada e Assinatura de Parâmetros

A grande barreira sistêmica superada pela biblioteca trata do gerenciamento e manipulação discrepante de memória entre as duas ferramentas adotadas, haja visto que Python emprega tipagem baseada em referências dinâmicas de abstração baseadas em objetos, enquanto a linguagem C atua em nível de endereçamento primitivo com tipagem puramente estática e pré-dimensionada [Ting 2022].

Para prevenir falhas de acesso de memória perante esse conflito arquitetural, a biblioteca impõe que o desenvolvedor realize o preenchimento das assinaturas estáticas via código para cada sub-rotina mapeada [Kloss 2009]. Isso é realizado alterando-se diretamente as propriedades expostas do próprio módulo ctypes.

3.0.2. Fluxo de Execução Transacional

Ao compreendermos o ciclo de vida transacional no instante das invocações dos cálculos em C, é possível estabelecer um fluxo de quatro estágios de operação e acoplamento gerenciado pelo componente ctypes [Kloss 2009]:

1. A linguagem orquestradora, Python, encapsula iterativamente o domínio paramétrico originário, por exemplo, pontos gerados fluidamente pela base de vetores, em tipos tipados da respectiva classe importada de dados de baixo nível como em *ctypes.c_double(t)* [Kloss 2009].
2. Mediante o espelhamento contido na camada CDLL, o próprio interpretador Python chama e empilha as instruções matemáticas presentes na biblioteca C vinculada dinamicamente, transmitindo as matrizes sem sobrecarga do interpretador abstrato [Kloss 2009].
3. A ferramenta subjacente (Linguagem C) extrai a performance integral da Unidade Lógica-Aritmética do hardware e devolve as predições iterativas como um valor escalar da matriz compilada em padrão de dupla precisão [Seabra et al. 2018, Ting 2022].
4. A estrutura interna final do ctypes intercepta no barramento local este valor referenciado de duplo ponto e cuida de restabelecer esse conteúdo numéricas dinamicamente à modelagem de ponto flutuante interna do Python, permitindo sua inclusão transparente nas listagens visuais para a apresentação na interface construída [Kloss 2009].

Nesta via unificada de operações, garante-se o sucesso do projeto arquitetural: aplica-se a melhor ferramenta para abstração amigável e desenvolvimento gerencial com Python, atrelando seus componentes numéricos com o alto ganho de escala matemática entregue pela precisão das operações iterativas de baixo nível contidas no escopo da linguagem C [Seabra et al. 2018, Ting 2022].

4. Implementação

4.1. Motor de Cálculo

O motor de cálculo das Séries de Fourier foi implementado nos arquivos *calculo.c* e *calculo.h*. Foram desenvolvidas três funções principais, cada uma responsável por calcular a aproximação de um tipo de onda específica, squareWave, sawtoothWave e triangleWave, recebendo como parâmetros o número de harmônicas N, a amplitude de pico A e o instante de tempo em segundos t.

4.1.1. Onda Quadrada (squareWave)

O algoritmo utiliza apenas as harmônicas ímpares da série. A fórmula é baseada na iteração da variável k até o limite de precisão N, sendo calculada no modelo:

$$f(t) = \frac{4A}{\pi} \sum_{k=1}^{N} \frac{\sin((2k-1) \cdot 2\pi \cdot t)}{2k-1}$$

4.1.2. Onda Dente de Serra (sawtoothWave)

Utiliza as harmônicas pares e ímpares, aplicando um fator de alternância de sinal e assegurando o decaimento linear da onda por meio de sua divisão por k. A expressão algébrica codificada é:

$$f(t) = \frac{2A}{\pi} \sum_{k=1}^{N} \frac{(-1)^{k+1}}{k} \sin(k \cdot 2\pi \cdot t)$$

4.1.3. Onda Triangular (triangleWave)

Implementa também apenas as harmônicas ímpares onde $n = 2k - 1$, com alternância de sinal a cada termo ímpar iterado e promovendo um decaimento quadrático para suavizar a onda geométrica. A função computacional equivale à fórmula:

$$f(t) = \frac{8A}{\pi^2} \sum_{k=1}^{N} \frac{(-1)^{k-1}}{(2k-1)^2} \sin((2k-1) \cdot 2\pi \cdot t)$$

Para viabilizar a comunicação interlinguagens, o código C não compila como um executável binário final padrão, mas sim como uma biblioteca dinâmica compartilhada, expondo os métodos diretamente ao código orquestrador.

4.2. Interface e Apresentação

O módulo *interface.py* atua como a camada de visualização e utilização do usuário. A integração direta com o código C ocorre através da biblioteca nativa *ctypes*. A sua principal vocação consiste em carregar bibliotecas de vínculo dinâmico baseadas em C e executar suas rotinas diretamente, contornando a necessidade de escrever adaptadores intermediários (wrappers) e mantendo a portabilidade [Kloss 2009].

Durante o carregamento da ferramenta matemática a aplicação em Python efetua a vinculação das assinaturas de rotina, convertendo explicitamente a tipagem de dados da plataforma interpretada para o padrão tipado do C. A biblioteca é inicializada por meio da chamada *ctypes.CDLL* e assina os protótipos das funções, informando que elas recebem três argumentos *c_int*, *c_int*, *c_double* e possuem retorno em *c_double*. Sem esta declaração, o interpretador Python não saberia mapear os tipos de dados nativos ao interagir com o ambiente em C, prevenindo também que o interpretador Python lance exceções ao alocar memória e formatar as chamadas iterativas de baixa latência [Kloss 2009].

Para a geração visual dos resultados, o código gera um vetor com valores de tempo no domínio de 0 a 4π utilizando a biblioteca NumPy [Kloss 2009, Ting 2022]. A aplicação varre este vetor por meio de uma estrutura de repetição e invoca a respectiva função da biblioteca em C para cada elemento do domínio.

A representação dos dados recebidos foi construída em interface com os pacotes Matplotlib e PyQt5, escolhidos por viabilizarem o desenho tanto de gráficos estáticos a serem exportados em formato PNG quanto de modelos gráficos fluidos com animações [Seabra et al. 2018, Ting 2022].

Referências

Kloss, G. K. (2009). Automatic C library wrapping – Ctypes from the trenches. *Res. Lett. Inf. Math. Sci.*, 13.

Seabra, R. D., Drummond, I. N., and Gomes, F. C. (2018). Análise comparativa de linguagens de programação a partir de problemas clássicos da computação. *Revista de Sistemas e Computação*, 8.

Ting, W. S. (2022). Referência comparativa rápida entre Python e C para sistemas com recursos limitados. EA871 – Notas de aula – Faculdade de Engenharia Elétrica e de Computação (FEEC).