CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS (CEFET-MG)
CAMPUS DIVINÓPOLIS

Detector de Pulsares Educativo v2.0

Sistema Interativo de Processamento Digital de Sinais Aplicado à Detecção de Pulsares Astronômicos

Matheus Emanuel da Silva
João Paulo Faria da Cunha

Sinais e Sistemas Lineares
Engenharia de Computação – 5º Período

Divinópolis, MG
23 de julho de 2025

Sumário

1 Sobre o Projeto **3**
1.1 O que são Pulsares? . . . 3

2 Objetivos Educacionais **4**
2.1 Conceitos de Sinais e Sistemas . . . 4
2.2 Habilidades Práticas . . . 4

3 Instalação e Configuração **5**
3.1 Pré-requisitos . . . 5
3.2 Instalação Automática . . . 5
3.2.1 Windows . . . 5
3.2.2 PowerShell . . . 5
3.2.3 Linux/Mac . . . 5
3.3 Instalação Manual . . . 6
3.4 Dependências . . . 6

4 Tutorial Completo **7**
4.1 Modos de Operação . . . 7
4.1.1 Modo Educativo . . . 7
4.1.2 Modo Aleatório . . . 7
4.2 Passo a Passo Detalhado . . . 7
4.2.1 Passo 1: Geração de Componentes . . . 7
4.2.2 Passo 2: Superposição Linear . . . 8
4.2.3 Passo 3: Adição de Ruído . . . 8
4.2.4 Passo 4: Filtragem Digital . . . 8
4.2.5 Passo 5: Análise FFT . . . 8

5 Fundamentos Teóricos **9**
5.1 Matemática dos Sinais . . . 9
5.1.1 Síntese por Superposição . . . 9
5.1.2 Amostragem e Discretização . . . 9
5.1.3 Filtragem Digital . . . 9
5.1.4 Transformada de Fourier Discreta (DFT) . . . 9
5.2 Física dos Pulsares . . . 10
5.2.1 Formação e Estrutura . . . 10
5.2.2 Mecanismo de Emissão . . . 10

6 Como Usar **11**
6.1 Interface do Usuário 11
6.2 Fluxo de Trabalho 12

7 Desenvolvimento **13**
7.1 Estrutura do Projeto 13
7.2 Arquitetura do Sistema 13

Capítulo 1

Sobre o Projeto

Este simulador educativo combina **astrofísica** e **processamento digital de sinais** para criar uma experiência de aprendizado única. Os usuários exploram desde conceitos básicos de superposição de ondas até técnicas avançadas de análise espectral, tudo no contexto da detecção de pulsares – estrelas de nêutrons que são alguns dos objetos mais extremos do universo.

1.1 O que são Pulsares?

Pulsares são estrelas de nêutrons altamente magnetizadas que rotacionam rapidamente (até 700 vezes por segundo!). Elas emitem feixes de radiação eletromagnética que, quando apontados para a Terra, são detectados como pulsos regulares – funcionando como "faróis cósmicos"com precisão comparável a relógios atômicos.

Características importantes:

- **Massa**: 1.4 massas solares comprimidas em ~20 km de diâmetro.
- **Rotação**: De milissegundos a segundos por rotação.
- **Emissão**: Radiação em múltiplas frequências simultaneamente.
- **Precisão**: Alguns são mais precisos que relógios atômicos.

Capítulo 2

Objetivos Educacionais

2.1 Conceitos de Sinais e Sistemas

- **Síntese de Sinais**: Superposição de componentes senoidais.
- **Amostragem Digital**: Teorema de Nyquist e discretização.
- **Sistemas LTI**: Linearidade e invariância temporal.
- **Filtragem Digital**: Filtros passa-baixa Butterworth.
- **Análise Espectral**: Transformada de Fourier (FFT).
- **Processamento de Ruído**: Relação sinal-ruído (SNR).

2.2 Habilidades Práticas

- Implementação de algoritmos de processamento digital.
- Análise de sinais no domínio tempo-frequência.
- Detecção de periodicidade em sinais ruidosos.
- Interpretação de espectros de potência.
- Configuração de parâmetros de filtragem.

Capítulo 3

Instalação e Configuração

3.1 Pré-requisitos

```
Python 3.8 ou superior
Git (para clonagem do reposit rio)
```

3.2 Instalação Automática

3.2.1 Windows

```
# Execute o script de instalacao
run.bat
```

3.2.2 PowerShell

```
# Execute o script PowerShell
.\\run.ps1
```

3.2.3 Linux/Mac

```
# Clone o repositorio
git clone https://github.com/Matheus-Emanue123/Trabalho-Final-SSL.git
cd Trabalho-Final-SSL

# Crie ambiente virtual
python3 -m venv venv
source venv/bin/activate

# Instale dependencias
pip install -r requirements.txt

# Execute a aplicacao
python src/main_app.py
```

3.3 Instalação Manual

1. **Clone o repositório:**

```
git clone https://github.com/Matheus-Emanue123/Trabalho-Final-SSL.git
cd Trabalho-Final-SSL

```

2. **Crie um ambiente virtual:**

```
python -m venv .venv

# Windows
.venv\Scripts\activate

# Linux/Mac
source .venv/bin/activate

```

3. **Instale as dependências:**

```
pip install -r requirements.txt

```

4. **Execute a aplicação:**

```
python src/main_app.py

```

3.4 Dependências

- **numpy**: Computação numérica e arrays.
- **matplotlib**: Visualização de gráficos e plots.
- **scipy**: Algoritmos científicos (filtros, FFT).
- **tkinter**: Interface gráfica (incluído no Python).

Capítulo 4

Tutorial Completo

4.1 Modos de Operação

4.1.1 Modo Educativo

- **Características**: Parâmetros fixos e reproduzíveis.
- **Componentes**: 3 senoides bem definidas (5, 15, 25 Hz).
- **Objetivo**: Demonstração clara dos conceitos.
- **Ideal para**: Primeira experiência e aprendizado.

4.1.2 Modo Aleatório

- **Características**: Simulação realística.
- **Variações**: Pulsar típico, sinal irregular, apenas ruído.
- **Desafio**: Nem sempre há pulsar detectável!
- **Ideal para**: Teste de conhecimento e casos reais.

4.2 Passo a Passo Detalhado

4.2.1 Passo 1: Geração de Componentes

O que acontece:

- Criação de 2-5 componentes senoidais independentes.
- Cada componente tem frequência, amplitude e fase específicas.
- Visualização individual de cada onda.

Conceitos aplicados:

$$x_i(t) = A_i \cdot \sin(2\pi f_i \cdot t + \phi_i) \tag{4.1}$$

Onde A_i é a amplitude, f_i é a frequência em Hz e ϕ_i é a fase inicial.

4.2.2 Passo 2: Superposição Linear

O que acontece:

- Soma algébrica de todas as componentes.
- Demonstração do princípio da superposição.

Conceitos aplicados:

$$x(t) = \sum_i x_i(t) = \sum_i A_i \cdot \sin(2\pi f_i \cdot t + \phi_i) \tag{4.2}$$

4.2.3 Passo 3: Adição de Ruído

O que acontece:

- Adição de ruído gaussiano branco.
- Simulação de condições reais de detecção.

Conceitos aplicados:

$$y(t) = x(t) + n(t) \tag{4.3}$$

Onde $n(t)$ é o ruído gaussiano. A Relação Sinal-Ruído (SNR) é calculada como:

$$\text{SNR} = 10 \cdot \log_{10}\left(\frac{P_{\text{sinal}}}{P_{\text{ruído}}}\right) \quad [\text{dB}] \tag{4.4}$$

4.2.4 Passo 4: Filtragem Digital

O que acontece:

- Aplicação de filtro Butterworth passa-baixa.
- Remoção de componentes de alta frequência (ruído).

Parâmetros configuráveis:

- Frequência de corte: 35 Hz (ajustável).
- Ordem: 5ª ordem.
- Tipo: Butterworth (resposta maximalmente plana).

4.2.5 Passo 5: Análise FFT

O que acontece:

- Transformada de Fourier do sinal filtrado.
- Cálculo do espectro de potência.
- Detecção automática de picos de frequência.

Conceitos aplicados: A Transformada de Fourier $X(f)$ e o Espectro de Potência $P(f)$.

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt \tag{4.5}$$

$$P(f) = |X(f)|^2 \tag{4.6}$$

O período é então calculado como $T = 1/f$.

Capítulo 5

Fundamentos Teóricos

5.1 Matemática dos Sinais

5.1.1 Síntese por Superposição

A base matemática dos pulsares reside na síntese de Fourier. Para nosso simulador, usamos uma aproximação finita:

$$x(t) = \sum_{i=1}^{N} A_i \sin(2\pi f_i t + \phi_i) \tag{5.1}$$

5.1.2 Amostragem e Discretização

O teorema de Nyquist garante que a frequência de amostragem (f_s) deve ser pelo menos o dobro da frequência máxima do sinal (f_{max}):

$$f_s \geq 2 \cdot f_{\text{max}} \tag{5.2}$$

No projeto, usamos $f_s = 1000$ Hz para um f_{max} de 50 Hz, garantindo uma margem de segurança de 10x (oversampling).

5.1.3 Filtragem Digital

O filtro Butterworth de ordem N tem uma resposta em frequência ao quadrado dada por:

$$|H(j\omega)|^2 = \frac{1}{1 + (\omega/\omega_c)^{2N}} \tag{5.3}$$

Onde ω_c é a frequência de corte angular e N é a ordem do filtro.

5.1.4 Transformada de Fourier Discreta (DFT)

A DFT é calculada eficientemente pela FFT (Fast Fourier Transform):

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi kn}{N}} \tag{5.4}$$

A complexidade computacional é da ordem de $O(N \log N)$.

5.2 Física dos Pulsares

5.2.1 Formação e Estrutura

- **Colapso gravitacional**: Estrela massiva se torna uma estrela de nêutrons.
- **Conservação do momento angular**: Resulta em rotação extremamente rápida.
- **Campo magnético**: Intenso, da ordem de 10^8 a 10^{15} Gauss.

5.2.2 Mecanismo de Emissão

- **Aceleração de partículas**: O campo magnético rotativo acelera partículas.
- **Radiação síncrotron**: Elétrons relativísticos emitem radiação.
- **Efeito farol**: O cone de emissão, alinhado com o eixo magnético, varre o espaço. Se cruzar a Terra, detectamos um pulso.

Capítulo 6

Como Usar

6.1 Interface do Usuário

Painel Lateral Contém a barra de progresso, botões de controle, seletor de modo e o banco de conhecimento.

Área Principal Exibe os gráficos em tempo real, com uma barra de ferramentas para navegação (zoom, pan, salvar).

Barra de Status Mostra a operação atual e um indicador visual de status.

A imagem 6.1 apresenta a intefeace gráfica do software em questão, ilustrando detalhadamente cada aspecto citado acima.

Figura 6.1: Interface do software exibindo os sinais obtidos após o processamento, incluindo o sinal original com ruído, o sinal filtrado e o espectro de potência.

6.2 Fluxo de Trabalho

1. **Inicialização**: Escolha o modo (educativo/aleatório) e clique em "Gerar Componentes".

2. **Execução Sequencial**: Execute os passos em ordem, observando as visualizações e lendo as explicações.

3. **Análise de Resultados**: Interprete os espectros de potência para identificar as frequências do pulsar.

4. **Experimentação**: Use "Reinicializar Sistema"para novas simulações e explore diferentes cenários.

De forma detalhada, a imagem 6.2 apresenta os gráficos gerados após executadas todas as etapas de processamento do sinais gerados pelo software.

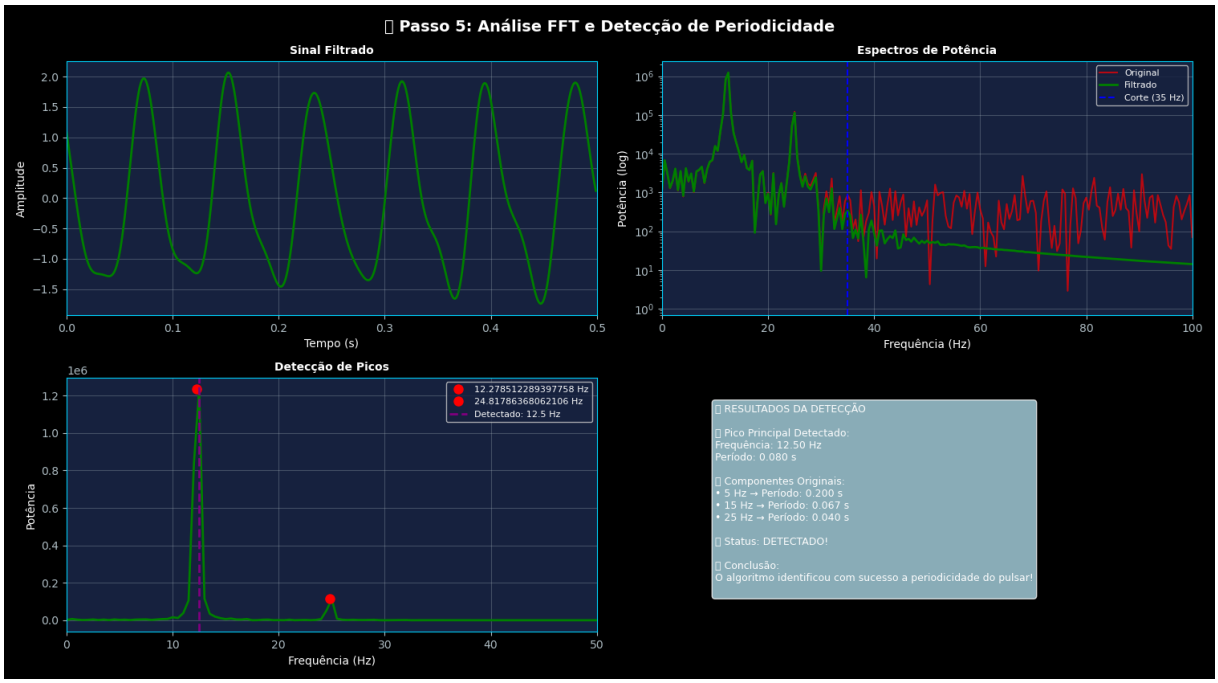

Figura 6.2: Gráficos gerados pelo software, mostrando os sinais filtrados após o processamento.

Capítulo 7

Desenvolvimento

7.1 Estrutura do Projeto

```
Trabalho-Final-SSL/
|-- src/
|   |-- main_app.py          # Interface principal e logica
|   |-- generate_signal.py   # Geracao de sinais sinteticos
|   |-- process_signal.py    # Processamento e filtragem
|   `-- __pycache__/
|-- requirements.txt         # Dependencias Python
|-- run.bat                  # Script Windows
|-- run.ps1                  # Script PowerShell
|-- README.md                # Documentacao
`-- LICENSE                  # Licenca MIT
```

7.2 Arquitetura do Sistema

Classe Principal: `PulsarDetectorApp` Responsável pela interface gráfica e coordenação geral, seguindo um padrão próximo ao MVC (Model-View-Controller).

Módulo de Geração: `generate_signal.py` Contém a lógica para criar os sinais sintéticos nos modos educativo e aleatório.

Módulo de Processamento: `process_signal.py` Implementa os algoritmos de filtragem (Butterworth), análise espectral (FFT) e detecção de picos.