Manual do Usuário - NPE-PSQ Advanced Tokamak Simulator v3.0

1. Introdução

O **NPE-PSQ Advanced Tokamak Simulator** é uma ferramenta de simulação de alta fidelidade para o estudo e controle de plasmas de fusão em tokamaks. Ele integra modelos físicos avançados (MHD, transporte, fusão D-T) com sistemas de controle de ponta (MPC, Redes Neurais) e um sistema de segurança determinístico (PSQ).

2. Instalação e Configuração

2.1. Requisitos

- Python 3.11+
- Dependências listadas em `requirements.txt`

2.2. Instalação

Bash

```
# 1. Clonar o repositório
git clone https://github.com/Akirabrs/NPE-PSQ-Tokamak-Control.git
cd NPE-PSQ-Tokamak-Control

# 2. Criar e ativar ambiente virtual
python3.11 -m venv venv
source venv/bin/activate

# 3. Instalar dependências
pip install -r requirements.txt
```

3. Estrutura do Projeto

O projeto segue uma arquitetura modular para facilitar a manutenção e o desenvolvimento:

Plain Text

```
NPE-PSQ/
├── src/                          ← Códigos-fonte principais
│   ├── npe_psq_tokamak.py        ← Ponto de entrada principal
```

```
│   └── modules/                ← Módulos de funcionalidade
│       ├── fisica.py           ← Constantes, Geometria, Dinâmica de Plasma
│       ├── integracao.py       ← Integrador RK4 Adaptativo
│       ├── controladores.py    ← MPC e Redes Neurais (NPE )
│       ├── seguranca.py        ← Sistema de Segurança (PSQ)
│       └── diagnosticos.py     ← Cálculo de Diagnósticos e Plotagem
├── assets/                     ← Imagens e mídias
├── docs/                       ← Documentação técnica
├── examples/                   ← Scripts de demonstração
├── requirements.txt            ← Dependências
├── LICENSE                     ← Licença MIT
└── README.md                   ← Página principal
```

4. Modos de Simulação

O simulador pode ser executado em três modos principais, controlados pelo parâmetro `control_mode` na função `run_simulation`:

Modo	Descrição	Requisitos
fixed	Simulação sem controle ativo. A potência de aquecimento é constante. Ideal para estudos de estabilidade intrínseca.	Mínimos
mpc	Simulação com o **Model Predictive Control**. O MPC calcula a sequência ótima de atuadores para atingir um setpoint.	`cvxpy`, `osqp`
neural	Simulação com o **Neural Predictive Engine**. O NPE (Rede Neural) atua como controlador de ultra-baixa latência, protegido pelo sistema PSQ.	`torch`

5. Configuração do Tokamak

A configuração do tokamak é definida em `src/modules/fisica.py` através das classes `TokamakGeometry`, `MagneticConfiguration` e `PlasmaState`.

Exemplo de Configuração (ITER-like)

Python

```
from src.npe_psq_tokamak import create_iter_like_config

config = create_iter_like_config()
# Modificar um parâmetro
config.magnetic.I_p = 18.0 # Aumenta a corrente de plasma para 18 MA
```

6. Uso do Simulador

O ponto de entrada principal é a classe `NPEPSQSimulator` em `src/npe_psq_tokamak.py`.

6.1. Simulação Básica (Modo Fixo)

Python

```
from src.npe_psq_tokamak import NPEPSQSimulator, create_iter_like_config

config = create_iter_like_config()
simulator = NPEPSQSimulator(config)

# Executa 30 segundos de simulação com aquecimento fixo
history = simulator.run_simulation(t_end=30.0, control_mode="fixed")

# Imprime o sumário de diagnóstico do estado final
simulator.print_summary(history)
```

6.2. Simulação com MPC

Python

```
from src.npe_psq_tokamak import NPEPSQSimulator, create_iter_like_config,
MPCConfig

config = create_iter_like_config()
simulator = NPEPSQSimulator(config)

# Configura o MPC para um setpoint de 12 keV
mpc_conf = MPCConfig(
    T_e_ref=12.0,
    Z_ref=0.0,
    N=20 # Horizonte de predição
```

```
)

# Executa 20 segundos de simulação com controle MPC
history = simulator.run_simulation(t_end=20.0, control_mode="mpc",
mpc_config=mpc_conf)
```

6.3. Simulação com NPE (Neural)

Para usar o modo `neural`, o modelo deve ser treinado primeiro (clonagem comportamental do MPC) e os pesos salvos.

Python

```
from src.npe_psq_tokamak import NPEPSQSimulator, create_iter_like_config

config = create_iter_like_config()
simulator = NPEPSQSimulator(config)

# 1. Carregar o modelo neural (assume que 'npe_weights.pth' existe)
simulator.load_neural_controller("caminho/para/npe_weights.pth")

# 2. Executar simulação com controle neural e segurança PSQ
history = simulator.run_simulation(t_end=10.0, control_mode="neural")
```

7. Sistema de Segurança PSQ

O sistema PSQ (Plasma Stability Quenching) é ativado automaticamente nos modos `mpc` e `neural`. Ele monitora o estado do plasma em tempo real e, em caso de violação de limites críticos (q95, β_N, densidade, VDE), **substitui imediatamente** o controlador principal por uma ação de mitigação de emergência.

Limites Monitorados

- **q95:** Mínimo de 2.0
- **β_N:** Máximo de 3.5
- **Fração de Greenwald:** Máximo de 95%
- **Z (Posição Vertical):** Máximo de 0.1m

8. Diagnósticos e Visualização

Após a simulação, o histórico de dados pode ser processado para gerar diagnósticos e gráficos.

Python

```
from src.npe_psq_tokamak import calculate_diagnostics, plot_history

# 1. Calcular diagnósticos do estado final
final_state = simulator.get_final_state()
diagnostics = calculate_diagnostics(final_state, config.geometry,
config.magnetic)
print_summary(diagnostics)

# 2. Plotar o histórico da simulação
plot_history(history, config, filename="minha_simulacao.png")
# O gráfico será salvo em assets/minha_simulacao.png
```

9. Treinamento da Rede Neural (NPE)

O treinamento é um processo de **Clonagem Comportamental** onde a Rede Neural (NPE) aprende a imitar as ações do Controlador MPC.

1. **Gerar Dados:** Execute o simulador no modo `mpc` por um longo período, salvando o estado e as ações de controle em um arquivo CSV.
2. **Treinar:** Use as classes `NeuralPredictiveEngine` e `NPETrainer` em `src/modules/controladores.py` para carregar os dados e treinar o modelo.

Python

```
# Exemplo de Treinamento (ver src/modules/controladores.py para detalhes)
trainer = NPETrainer(model)
# ... carregar dados ...
trainer.train(X_train, Y_train, epochs=100)
torch.save(model.state_dict(), "npe_weights.pth")
```

---

10. Contato e Suporte

Para dúvidas, sugestões ou colaboração:

- **Autor:** Guilherme Brasil de Souza
- **Email:** guilherme@gbslabs.com
- **Repositório:** https://github.com/Akirabrs/NPE-PSQ-Tokamak-Control.git