

Análise Realista do Projeto NPE-PSQ Tokamak

Autor: Guilherme Brasil de Souza

Data da Análise: 22 de dezembro de 2025

Nível de Desenvolvimento: Intermediário-Avançado (em progresso)



Avaliação Geral

Seu projeto é **sólido e ambicioso**. Você desenvolveu um simulador completo de tokamak com controle MPC, segurança e diagnósticos. Não é trivial. Porém, há espaço significativo para refinamento e profundidade. Vou ser honesto e construtivo.



Pontos Fortes

1. Arquitetura Bem Estruturada

- Uso correto de `@dataclass` para organizar dados (ConstantesFisicas, ConfiguracaoTokamak, EstadoPlasma)
- Separação clara de responsabilidades: configuração, estado, controle, segurança, diagnósticos
- Código legível com comentários em português técnico

2. Física Implementada com Rigor

- Constantes físicas corretas:** Permeabilidade magnética, carga elementar, massa do elétron, etc.
- Seção de choque de fusão D-T:** Aproximação de Bosch-Hale com três regimes ($T < 1$ keV, $1-10$ keV, > 10 keV)
- Geometria tokamak realista:** Raio maior (R_0), raio menor (a), alongamento (κ), triangularidade (δ)
- Campos magnéticos:** Toroidal (B_0) e poloidal calculados corretamente
- Limite de Greenwald:** Implementado com densidade crítica

3. Controle Avançado (MPC)

- Controlador PID multivariável com ganhos K_p , K_i , K_d

- Rastreamento de múltiplas variáveis: corrente, q95, posição vertical, beta
- Anti-windup no termo integral
- Histórico de controle para análise

4. Sistema de Segurança Robusto

- Monitoramento de 7 condições críticas (q95 baixo, beta alto, densidade alta, etc.)
- Cálculo de risco agregado (produto de fatores de risco)
- Mitigação automática: Massive Gas Injection (MGI), Killer Pulse, redução de aquecimento
- Histórico de disrupções e sucessos de mitigação

5. Diagnósticos e Visualização

- Gráficos em tempo real com matplotlib
- 3 subplots: corrente/temperatura, estabilidade (q95/ β_N), potência de fusão
- Cores diferenciadas por variável
- Linhas de limite para referência visual

Limitações e Pontos de Melhoria

1. Física Simplificada em Alguns Aspectos

a) Dinâmica do Plasma

- O modelo atual é **quasi-estático**: você atualiza o estado com base em controles, mas não resolve as equações MHD (magnetohidrodinâmica) completas
- **Realidade**: Um tokamak real tem dinâmica de plasma muito mais complexa:
 - Instabilidades MHD (tearing modes, ballooning modes)
 - Transporte de energia (difusão térmica, convecção)
 - Efeitos de rotação do plasma

Nível atual: ~6/10 em fidelidade física

O que falta: Equações diferenciais acopladas para evolução temporal do plasma

b) Modelo de Aquecimento

Python

```
# Seu código
potencia_echr: float = 20.0 # Electron Cyclotron Resonance Heating
potencia_icrh: float = 30.0 # Ion Cyclotron Resonance Heating
potencia_nbi: float = 33.0 # Neutral Beam Injection
```

- Você define potências **fixas**, mas não modela:
 - Eficiência de acoplamento (depende da densidade, temperatura, posição)
 - Perfil radial de deposição de energia
 - Interação com o plasma (absorção, reflexão)

Recomendação: Adicionar funções que calculem eficiência em função do estado do plasma

c) Transporte de Energia

- Não há modelo de difusividade térmica (χ)
- A temperatura é atualizada apenas pelo aquecimento, sem perda radiativa realista
- **Realidade:** Perdas radiativas (bremsstrahlung, radiação de linha) são críticas

2. Controle MPC Simplificado

Seu MPC é um **PID multivariável**, não um MPC verdadeiro:

- Um MPC real resolve um problema de otimização em horizonte futuro (N passos)
- Seu código usa apenas erro atual + integral + derivada

Python

```
# Seu código (PID)
forca_vertical = (
    self.Kp * erros['posicao'] +
    self.Ki * self.erro_integral['posicao'] +
    self.Kd * derivativos['posicao']
)
```

Nível: 5/10 em sofisticação de controle

Para ser MPC real: Você precisaria:

1. Modelo dinâmico linearizado do tokamak
2. Otimizador (quadratic programming) a cada passo
3. Predição de N passos futuros
4. Restrições explícitas (limites de corrente, campo, etc.)

Alternativa prática: Usar biblioteca `do-mpc` (Python) ou `YALMIP` (MATLAB)

3. Validação e Testes

- **Sem testes unitários:** Não há verificação automática de:
 - Conservação de energia
 - Estabilidade numérica
 - Casos limite (densidade zero, temperatura zero)
- **Sem comparação com dados reais:** Você não valida contra:
 - Dados de tokamaks reais (ITER, JT-60SA, EAST)
 - Publicações científicas
 - Simuladores estabelecidos (TRANSP, CORSICA)

Recomendação: Adicionar testes para casos conhecidos

4. Fidelidade Numérica

- Integração temporal: Você usa **Euler explícito** (implícito no loop)

Python

```
estado.velocidade_vertical += aceleracao * dt
estado.posicao_vertical += estado.velocidade_vertical * dt
```

- Isso é **primeira ordem**, instável para dt grande
- **Melhor:** RK4 (Runge-Kutta 4ª ordem) ou métodos implícitos para stiffness

5. Documentação Técnica

- Código bem comentado, mas **falta documentação de design**:
 - Não há explicação das aproximações físicas usadas
 - Não há referências a papers ou modelos base
 - Não há diagrama de fluxo de dados

Classificação por Nível

Aspecto	Nível	Justificativa
---------	-------	---------------

Estrutura de Código	8/10	Bem organizado, mas poderia usar type hints mais rigorosos
Física Implementada	6/10	Correta mas simplificada; falta dinâmica MHD, transporte
Controle	5/10	PID multivariável, não MPC real
Segurança	8/10	Robusto, com mitigação automática
Visualização	7/10	Gráficos úteis, mas poderia incluir mais diagnósticos
Validação	3/10	Sem testes, sem comparação com literatura
Documentação	6/10	Código comentado, mas falta design doc

Nível Geral: 6.5/10 (Intermediário-Avançado)

Próximos Passos Recomendados

Curto Prazo (1-2 semanas)

1. Adicionar testes unitários

Python

```
def test_conservacao_energia():  
    # Verificar que energia total não aumenta sem aquecimento  
def test_limite_greenwald():  
    # Verificar que densidade não ultrapassa limite
```

2. Melhorar integração temporal

- Implementar RK4 em vez de Euler
- Adicionar adaptive time-stepping

3. Validação contra caso simples

- Comparar com TRANSP para configuração ITER padrão

Médio Prazo (1 mês)

1. Implementar transporte de energia

- Modelo de difusividade térmica (χ)
- Perdas radiativas realistas

2. MPC verdadeiro

- Usar biblioteca `do-mpc`
- Incluir restrições de controle

3. Documentação técnica

- Escrever design document
- Incluir referências a papers

Longo Prazo (3+ meses)

1. Dinâmica MHD simplificada

- Modelo de instabilidades (tearing modes)
- Acoplamento com rotação

2. Validação contra dados reais

- Comparar com shots de tokamaks (EAST, JT-60SA)
- Publicar resultados



Observações Finais

Você fez um trabalho sólido. Este é um projeto que:

- Demonstra compreensão profunda de tokamaks
- Implementa física real (não é brinquedo)
- Tem arquitetura profissional

Porém, para ser "production-ready" ou publicável:

- Precisa de validação rigorosa
- Precisa de comparação com simuladores estabelecidos
- Precisa de documentação técnica formal

Minha recomendação: Este é um excelente **projeto de pesquisa em progresso**. Se você:

- Adicionar testes e validação

- Publicar em repositório com documentação
- Comparar com TRANSP ou similar

...você teria um **contributo significativo** para a comunidade de fusão nuclear.

Nível Final: Intermediário-Avançado (6.5/10)

Potencial: Alto (poderia chegar a 9/10 com refinamento)

Recomendação: Continue! É um projeto promissor.