NPE-PSQ v3.0 - Guia de Início Rápido

📦 Instalação

Opção 1: Instalação Completa (Recomendada)

Bash

```bash
# 1. Clonar/copiar o projeto
cd /home/ubuntu/npe-psq-advanced

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

# 3. Instalar dependências completas
pip install numpy scipy matplotlib cvxpy osqp torch pandas pytest
```

Opção 2: Instalação Mínima (Apenas Simulação Básica)

Bash

```bash
# Instalar apenas dependências essenciais
pip install numpy scipy matplotlib
```

Nota: Para usar MPC e Neural Network, você precisará instalar `cvxpy`, `osqp` e `torch`.

🚀 Uso Rápido

1. Teste Básico do Sistema

Bash

```bash
python3.11 test_basic.py
```

Saída esperada:

Plain Text

```
======================================================================
TESTE BÁSICO DO SIMULADOR NPE-PSQ v3.0
```

```
======================================================================
[1/5]: # "Testando constantes físicas..."
  ✓ ELEMENTARY_CHARGE = 1.602e-19 C
  ✓ ELECTRON_MASS = 9.109e-31 kg
  ✓ BOLTZMANN_CONSTANT = 1.381e-23 J/K
[2/5]: # "Testando configuração do tokamak..."
  ✓ R0 = 6.20 m
  ✓ B_T = 5.30 T
  ✓ I_p = 15.00 MA
  ✓ Volume = 832.2 m³
...
✓ TODOS OS TESTES BÁSICOS PASSARAM!
```

2. Simulação Básica (Sem Controle)

Python

```python
import sys
sys.path.append('/home/ubuntu/npe-psq-advanced')

from src.tokamak_config import create_iter_like_config, ControlActuators
from src.numerical_integration import TokamakSimulator

# Configurar tokamak
config = create_iter_like_config()

# Criar simulador
simulator = TokamakSimulator(config)

# Definir aquecimento constante
actuators = ControlActuators(
    P_NBI=20.0,   # 20 MW
    P_ECRH=10.0,  # 10 MW
    P_ICRH=15.0   # 15 MW
)

# Simular 30 segundos
history = simulator.simulate(t_end=30.0, actuators=actuators)

# Ver estado final
final_state = simulator.get_final_state()
print(f"Temperatura final: {final_state.T_e:.2f} keV")
```

3. Simulação com Controle MPC (Requer cvxpy)

Python

```python
import sys
sys.path.append('/home/ubuntu/npe-psq-advanced')

from src.tokamak_config import create_iter_like_config
from src.numerical_integration import TokamakSimulator
from src.mpc_controller import MPCController, MPCConfig

# Configurar
config = create_iter_like_config()

# Configurar MPC
mpc_config = MPCConfig(
    N=15,           # Horizonte de predição
    T_e_ref=12.0,   # Setpoint de temperatura
    Z_ref=0.0       # Setpoint de posição
)

# Criar controlador
controller = MPCController(
    config.geometry,
    config.magnetic,
    mpc_config
)

# Função de controle
def mpc_control(state, t):
    return controller.compute_control(state)

# Simular com controle
simulator = TokamakSimulator(config)
history = simulator.simulate(
    t_end=20.0,
    actuators=None,
    controller=mpc_control
)
```

4. Sistema de Segurança PSQ

Python

```python
import sys
sys.path.append('/home/ubuntu/npe-psq-advanced')

from src.tokamak_config import create_iter_like_config
from src.safety_system import PlasmaStabilityQuenching
```

```python
# Configurar
config = create_iter_like_config()

# Criar sistema de segurança
psq = PlasmaStabilityQuenching()

# Verificar segurança do estado
is_safe, action_code, message = psq.check_safety(
    config.state,
    config.geometry,
    config.magnetic
)

print(f"Estado: {'SEGURO' if is_safe else 'VIOLAÇÃO'}")
print(f"Código: {action_code.name}")
print(f"Mensagem: {message}")
```

📊 Estrutura do Projeto

Plain Text

```
npe-psq-advanced/
├── src/                          # Código fonte
│   ├── constants.py              # Constantes físicas
│   ├── tokamak_config.py         # Configuração do tokamak
│   ├── plasma_dynamics.py        # Física MHD
│   ├── numerical_integration.py  # Integrador RK4
│   ├── mpc_controller.py         # Controlador MPC
│   ├── neural_controller.py      # Rede Neural NPE
│   └── safety_system.py          # Sistema de segurança PSQ
├── examples/                     # Exemplos
│   ├── basic_simulation.py       # Simulação básica
│   └── mpc_control_simulation.py # Simulação com MPC
├── tests/                        # Testes unitários
├── docs/                         # Documentação
├── README.md                     # Documentação principal
├── QUICKSTART.md                 # Este arquivo
├── requirements.txt              # Dependências
└── test_basic.py                 # Teste rápido
```

🔧 Troubleshooting

Problema: ModuleNotFoundError: No module named 'src'

Solução:

```python
import sys
sys.path.append('/home/ubuntu/npe-psq-advanced')
```

Problema: ModuleNotFoundError: No module named 'cvxpy'

Solução:

```bash
pip install cvxpy osqp
```

Problema: ModuleNotFoundError: No module named 'torch'

Solução:

```bash
pip install torch
```

📚 Próximos Passos

1. **Ler a documentação completa:** `README.md`
2. **Explorar exemplos:** Pasta `examples/`
3. **Treinar rede neural:** Ver `src/neural_controller.py`
4. **Criar seus próprios cenários:** Modificar configurações em `src/tokamak_config.py`

🆘 Suporte

Para dúvidas ou problemas:

- Consulte o `README.md`
- Veja os exemplos em `examples/`
- Contato: guilherme@gbslabs.com

Versão: 3.0.0

Última Atualização: Dezembro 2025