Lista de Comandos FEniCSx Identificados

🏷 Checklist Completo de Comandos FEniCSx

✅ Imports e Módulos

- [ ] `import dolfinx` - Biblioteca principal
- [ ] `from dolfinx import mesh, fem, io` - Módulos core
- [ ] `from dolfinx.fem import FunctionSpace, Function, Constant` - Elementos FEM
- [ ] `from dolfinx.fem.petsc import NonlinearProblem` - Problemas não-lineares
- [ ] `from dolfinx.nls.petsc import NewtonSolver` - Solver Newton
- [ ] `from mpi4py import MPI` - Paralelização MPI
- [ ] `from petsc4py import PETSc` - Tipos de dados PETSc
- [ ] `import ufl` - Unified Form Language
- [ ] `from ufl import grad, dot, dx, ds, inner, TestFunction` - Operadores UFL

✅ Malha e I/O

- [ ] `io.XDMFFile(comm, filename, mode)` - Leitor/escritor XDMF
- [ ] `xdmf.read_mesh(name="malha")` - Carregamento de malha
- [ ] `xdmf.read_meshtags(mesh, name="tags")` - Leitura de Physical Groups
- [ ] `xdmf.write_mesh(mesh)` - Escrita de malha
- [ ] `xdmf.write_function(function, time)` - Escrita de função
- [ ] `mesh.topology.create_connectivity(dim-1, 0)` - Conectividade de facetas
- [ ] `mesh.topology.dim` - Dimensão da malha

✅ Espaços de Função

- [ ] `fem.functionspace(mesh, ("Lagrange", 1))` - Criação de espaço FEM
- [ ] `Function(V)` - Função no espaço FEM
- [ ] `TestFunction(V)` - Função de teste
- [ ] `Constant(mesh, value)` - Constantes na formulação
- [ ] `PETSc.ScalarType(value)` - Conversão de tipos para PETSc

✅ Formulação UFL

- [ ] `dx(domain=mesh, subdomain_data=cell_tags)` - Integração por domínio
- [ ] `ds(domain=mesh, subdomain_data=facet_tags)` - Integração de superfície
- [ ] `grad(function)` - Operador gradiente
- [ ] `dot(vector1, vector2)` - Produto interno
- [ ] `inner(vector1, vector2)` - Produto interno
- [ ] `dx_tags(domain_id)` - Integração em domínio específico
- [ ] `ds_tags(boundary_id)` - Integração em contorno específico

✅ Resolução de Sistema

- [ ] `NonlinearProblem(F, u, bcs)` - Definição do problema não-linear
- [ ] `NewtonSolver(comm, problem)` - Criação do solver Newton
- [ ] `solver.convergence_criterion` - Critério de convergência
- [ ] `solver.rtol` - Tolerância relativa
- [ ] `solver.solve(u)` - Resolução do sistema
- [ ] `solver.max_it` - Máximo de iterações

✅ Comunicação MPI

- [ ] `MPI.COMM_WORLD` - Comunicador MPI mundial
- [ ] `comm.Get_rank()` - Rank do processo atual
- [ ] `comm.Get_size()` - Número total de processos

✅ Manipulação de Dados

- [ ] `function.x.array[:]` - Acesso aos dados da função
- [ ] `function.name` - Nome da função para output
- [ ] `np.unique(tags.values)` - Extração de tags únicas
- [ ] `cell_tags.values` - Valores das tags de células
- [ ] `facet_tags.values` - Valores das tags de facetas

🔧 Comandos por Seção do Código

📂 Carregamento de Malha

```
# 🔧 FEniCSx: Seção de carregamento
with io.XDMFFile(self.comm, self.xdmf_file, "r") as xdmf:
    self.mesh = xdmf.read_mesh(name="malha")
    self.mesh.topology.create_connectivity(self.mesh.topology.dim-1, 0)
    self.cell_tags = xdmf.read_meshtags(self.mesh, name="malha_cells")
    self.facet_tags = xdmf.read_meshtags(self.mesh, name="malha_facets")
```

🧮 Configuração de Espaços

```
# 🔧 FEniCSx: Definição de espaços
self.V = fem.functionspace(self.mesh, ("Lagrange", 1))
self.T = Function(self.V)
self.Tn = Function(self.V)
self.v = TestFunction(self.V)
```

📐 Formulação Variacional

```
# 🔧 FEniCSx: Formulação UFL
dt = Constant(self.mesh, PETSc.ScalarType(dt_val))
theta = Constant(self.mesh, PETSc.ScalarType(self.theta))
dx_tags = dx(domain=self.mesh, subdomain_data=self.cell_tags)
ds_tags = ds(domain=self.mesh, subdomain_data=self.facet_tags)

# Termo temporal
F += rho * cp * (self.T - self.Tn) / dt * self.v * dx_tags(domain_id)

# Termo de difusão
T_theta = theta * self.T + (1 - theta) * self.Tn
F += k * dot(grad(T_theta), grad(self.v)) * dx_tags(domain_id)

# Condição de contorno Robin
F += h * (T_boundary - T_ext) * self.v * ds_tags(boundary_tag)
```

🔍 Resolução do Sistema

```
# 🔧 FEniCSx: Resolução não-linear
problem = NonlinearProblem(F, self.T, bcs)
solver = NewtonSolver(self.comm, problem)
solver.convergence_criterion = "incremental"
solver.rtol = 1e-6
n_iterations, converged = solver.solve(self.T)
```

💾 Salvamento de Resultados

```
# 🔧 FEniCSx: Exportação para Paraview
with io.XDMFFile(self.comm, output_file, "w") as xdmf:
    self.T.name = "Temperatura"
    xdmf.write_mesh(self.mesh)
    xdmf.write_function(self.T, current_time)
```

🎯 Comandos Críticos para Descoberta Automática

🔍 Descoberta de Physical Groups

```
# 🔧 FEniCSx: Extração automática de tags
self.discovered_cell_tags = np.unique(self.cell_tags.values)
self.discovered_facet_tags = np.unique(self.facet_tags.values)
```

📊 Processamento Dinâmico

```
# 🔧 FEniCSx: Formulação adaptativa
for domain_id in self.discovered_cell_tags:
    if domain_id in self.active_layers and self.active_layers[domain_id]:
        # Processar apenas domínios ativos
        F += termo_variacional * dx_tags(domain_id)
```

🏷️ Aplicação de Condições de Contorno

```
# 🔧 FEniCSx: BC automática
for boundary_tag, bc_config in self.boundary_conditions.items():
    if bc_config['tipo'] == 'conveccao':
        F += h * (T_boundary - T_ext) * self.v * ds_tags(boundary_tag)
```

📈 Estatísticas de Uso

Categorias de Comandos

- **Malha e I/O**: 7 comandos
- **Espaços de Função**: 5 comandos
- **Formulação UFL**: 7 comandos
- **Resolução**: 6 comandos
- **Paralelização**: 3 comandos
- **Dados**: 5 comandos

Total de Comandos FEniCSx: 33 comandos únicos

🚀 Comandos Mais Críticos

⭐ Top 5 Comandos Essenciais

1. `io.XDMFFile()` - Base para I/O
2. `fem.functionspace()` - Definição de espaços
3. `dx(subdomain_data=...)` - Integração por domínio
4. `NonlinearProblem()` - Definição do problema
5. `NewtonSolver()` - Resolução do sistema

⭐ Top 5 para Descoberta Automática

1. `xdmf.read_meshtags()` - Leitura de Physical Groups
2. `np.unique(tags.values)` - Extração de tags
3. `dx_tags(domain_id)` - Integração específica
4. `ds_tags(boundary_id)` - Contorno específico
5. `mesh.topology.create_connectivity()` - Conectividade

🎓 Observações Finais

✅ Padrões FEniCSx Identificados

- **Contexto MPI**: Todos os comandos consideram paralelização
- **Tipos PETSc**: Conversões adequadas para compatibilidade
- **Formulação UFL**: Linguagem unificada para elementos finitos
- **I/O XDMF**: Formato padrão para visualização

✅ Boas Práticas Implementadas

- **Tratamento de Erros**: Try/except em operações críticas
- **Verificação de Rank**: Logs apenas no processo principal
- **Gestão de Recursos**: Context managers para arquivos
- **Validação de Dados**: Verificação de existência antes de uso

Esta lista serve como **referência completa** de todos os comandos FEniCSx utilizados no código genérico, demonstrando o uso abrangente e sofisticado da biblioteca.