Plan de Implementación

Simulación de Datos de Disrupciones en Tokamaks

Pipeline DINA + DREAM para Generación de Dataset de Entrenamiento

Duración Total: 6 Semanas (~42 días)

Proyecto: tokamak-disruption-simulator

Repositorio Independiente para Generación de Datos Sintéticos de Alta Fidelidad

📋 Resumen Ejecutivo

Objetivo Principal

Desarrollar un sistema de generación de datos sintéticos de alta fidelidad que simule disparos de tokamak (disruptivos y no-disruptivos) utilizando los códigos de física de plasmas DINA y DREAM. El dataset generado servirá como entrada para entrenar modelos de predicción de disrupciones basados en Fourier Neural Operators (FNO) en el proyecto hermano tokamak-fno.

¿Por Qué Este Proyecto?

Los datos sintéticos simples (gaussianos con ruido) producen resultados perfectos (100% accuracy) que no son útiles para validar modelos de ML. Este proyecto resuelve ese problema generando datos con física realista:

- **DINA:** Simula la evolución temporal realista de parámetros del plasma (Ip, βN, q95, ne, li) durante operación normal y aproximación a límites operacionales.
- **DREAM:** Simula la física de la disrupción propiamente dicha (Thermal Quench, Current Quench) con modelos validados científicamente.
- **Pipeline Acoplado:** Combina ambos códigos para generar disparos completos desde inicio hasta fin de la disrupción.

Métricas de Éxito del Dataset

Métrica	Target	Justificación
Disparos Totales	1,000 - 2,000	Suficiente para proof-of-concept
Balance de Clases	50% disruptivos / 50% normales	Evitar sesgo en entrenamiento
Señales por Disparo	5 (Ip, βN, q95, ne, li)	Compatibilidad con tokamak-fno
Frecuencia de Muestreo	10 kHz	Estándar en diagnósticos de tokamaks
Variabilidad de Escenarios	> 10 configuraciones	Robustez del modelo
Validación Física	Respeta límites operacionales	Datos físicamente plausibles

Entregables Finales

1. **Repositorio GitHub completo** (tokamak-disruption-simulator) con código documentado y reproducible
2. **Dataset HDF5** con 1,000+ disparos etiquetados (formato compatible con tokamak-fno)
3. **Scripts de configuración** para DINA y DREAM con escenarios parametrizables
4. **Notebooks de análisis** con visualizaciones de los datos generados
5. **Documentación técnica** explicando la física y el pipeline

🔬 Fundamentos Teóricos

¿Qué es una Disrupción?

Una disrupción es un evento catastrófico donde el plasma de un tokamak pierde su confinamiento magnético en milisegundos, liberando toda su energía térmica y magnética. Es el problema de seguridad más crítico para reactores de fusión como ITER.

Secuencia Temporal de una Disrupción

Fase	Duración	Descripción
Precursores	100 - 300 ms	Señales de advertencia: modos MHD crecientes, aproximación a límites operacionales, pérdida gradual de confinamiento
Thermal Quench (TQ)	0.1 - 2 ms	Colapso térmico ultrarrápido: Te cae de ~10 keV a ~10 eV. Pérdida de confinamiento de energía.
Current Quench (CQ)	10 - 100 ms	Decaimiento de corriente de plasma: Ip cae de 15 MA a 0. Genera campo eléctrico inductivo intenso.
Post-disrupción	Variable	Posible generación de electrones runaway. Daño potencial a componentes.

Límites Operacionales (Causas de Disrupción)

Las disrupciones ocurren cuando el plasma viola ciertos límites físicos:

- **Límite de Greenwald (densidad):** $n/nG < 1.0$ donde $nG = Ip/(\pi a^2)$. Exceder este límite causa pérdida de confinamiento.
- **Límite de Troyon (beta):** $\beta N < 3.5$. El beta normalizado mide la eficiencia del confinamiento magnético.
- **Factor de seguridad crítico:** $q95 > 2.0$. Valores menores causan inestabilidades MHD globales.
- **Desplazamiento vertical (VDE):** Pérdida de control de posición vertical del plasma.

💻 Códigos de Simulación: DINA y DREAM

DINA - Plasma Scenario Simulator

Repositorio: https://github.com/iterorganization/DINA-IMAS

Licencia: LGPL-3.0 (Open Source)

Propósito: Simular la evolución temporal de escenarios de tokamak, desde breakdown hasta fin de disparo.

Física que Resuelve DINA

1. **Ecuación de Grad-Shafranov 2D:** Equilibrio MHD de frontera libre. Calcula la forma del plasma y distribución de corriente.
2. **Ecuaciones de circuito:** Corrientes en bobinas de campo poloidal (PF coils) y estructuras pasivas.
3. **Difusión de flujo poloidal 1D:** Evolución del perfil de corriente j(r,t).
4. **Transporte de energía 1D:** Evolución de Te(r,t) y Ti(r,t) con fuentes de calentamiento y pérdidas.
5. **Transporte de densidad 1D:** Evolución de ne(r,t) con fueling y bombeo.

Outputs de DINA (Lo que usaremos)

Señal	Descripción	Unidades
Ip(t)	Corriente de plasma total	Amperes [A]
βN(t)	Beta normalizado	Adimensional
q95(t)	Factor de seguridad en ψN=0.95	Adimensional
ne(r,t)	Perfil de densidad electrónica	m^{-3}
Te(r,t)	Perfil de temperatura electrónica	eV o keV
li(t)	Inductancia interna normalizada	Adimensional
j(r,t)	Perfil de densidad de corriente	A/m^2

Limitaciones de DINA para Disrupciones

DINA NO puede simular:

- El Thermal Quench (colapso térmico en ~1 ms)
- El Current Quench (decaimiento de Ip)
- Modos MHD 3D (tearing modes, locked modes)
- Generación de electrones runaway

Por eso necesitamos DREAM para completar la simulación de disrupciones.

DREAM - Disruption Runaway Electron Analysis Model

Repositorio: https://github.com/chalmersplasmatheory/DREAM

Licencia: MIT (Open Source)

Documentación: https://ft.nephy.chalmers.se/dream

Propósito: Simular la evolución del plasma durante una disrupción, con énfasis en electrones runaway.

Física que Resuelve DREAM

1. **Difusión de corriente:** Evolución del perfil de corriente durante el Current Quench con resistividad dependiente de temperatura.
2. **Balance de energía electrónica:** Incluye calentamiento óhmico, radiación, colisiones y transporte.
3. **Evolución de especies iónicas:** Ionización, recombinación y transporte de impurezas.
4. **Ecuación cinética de electrones:** Modelo fluid-kinetic para electrones runaway (Dreicer, avalanche, hot-tail).

Capacidades Clave de DREAM

Capacidad	Descripción
Thermal Quench	Modela el colapso de temperatura mediante transporte anómalo configurable
Current Quench	Simula el decaimiento de Ip con resistividad Spitzer y efectos de pared
Runaway electrons	Modelos completos de generación: Dreicer, avalanche, hot-tail
Shattered Pellet Injection	Simula mitigación de disrupciones con inyección de pellets
Interfaz IMAS	Puede leer condiciones iniciales directamente de bases de datos IMAS
Python frontend	API Python completa para configuración y análisis

Outputs de DREAM (Lo que usaremos)

Señal	Descripción	Unidades
Ip(t)	Corriente de plasma durante CQ	Amperes [A]
Te(r,t)	Temperatura durante TQ y CQ	eV
ne(r,t)	Densidad durante disrupción	m^{-3}
E(r,t)	Campo eléctrico inductivo	V/m
I_re(t)	Corriente de runaway (opcional)	Amperes [A]

🔄 Arquitectura del Pipeline DINA → DREAM

Visión General

El pipeline combina DINA y DREAM de manera secuencial, donde DINA simula la fase pre-disrupción y DREAM toma el relevo para simular la disrupción propiamente dicha.

Flujo para Disparos NO-Disruptivos

Solo DINA: t=0 → Breakdown → Ramp-up → Flat-top → Ramp-down → t_end

Para disparos normales, DINA simula todo el escenario completo. Los parámetros se mantienen dentro de límites operacionales seguros.

Flujo para Disparos DISRUPTIVOS

DINA + DREAM: t=0 → [DINA: evolución normal] → t_trigger → [HANDOFF] → [DREAM: TQ + CQ] → t_end

El Concepto del HANDOFF

El "handoff" es el momento crítico (t_trigger) donde transferimos el estado del plasma de DINA a DREAM:

1. **DINA evoluciona el plasma** hasta que los parámetros violan un límite operacional (ej: n/nG > 0.95)
2. **Se detecta t_trigger:** El momento exacto donde las condiciones triggearían una disrupción
3. **Se extraen perfiles:** Te(r), ne(r), j(r), Ip, q(r) en t_trigger
4. **DREAM se inicializa:** Con esos perfiles como condiciones iniciales
5. **DREAM simula la disrupción:** Thermal Quench (~1 ms) + Current Quench (~50 ms)

Datos Transferidos en el Handoff

Dato	Desde DINA	Para DREAM
Te(r)	Perfil de temperatura	Condición inicial de energía
ne(r)	Perfil de densidad	Densidad de especies
j(r)	Perfil de corriente	Distribución de corriente
Ip	Corriente total	Normalización
q(r)	Factor de seguridad	Geometría magnética
Geometría	R_0, a, κ, δ	Grid radial
Composición	Zeff, impurezas	Especies iónicas

🗓️ Cronograma Detallado: 6 Semanas

SEMANA 1: Setup del Entorno e Instalación de Códigos

Objetivo: Tener DINA y DREAM instalados y funcionando con ejemplos básicos

Día 1-2: Setup del Repositorio y Entorno Base

Horas estimadas: 6-8h

Tareas:

- ☐ Crear repositorio GitHub: tokamak-disruption-simulator
- ☐ Configurar estructura inicial de directorios
- ☐ Crear entorno conda: conda create -n tokamak_sim python=3.10
- ☐ Instalar dependencias base: numpy, scipy, h5py, matplotlib, pandas
- ☐ Documentar requisitos del sistema en README.md

Día 3-4: Instalación de DREAM

Horas estimadas: 8-10h

Requisitos previos:

- CMake >= 3.12
- Compilador C++17 (gcc >= 7.0)
- GNU Scientific Library (GSL) >= 2.4
- HDF5, PETSc
- Python 3 con h5py, matplotlib, numpy, scipy

Pasos de instalación:

- ☐ Clonar repositorio: git clone https://github.com/chalmersplasmatheory/DREAM.git
- ☐ Compilar PETSc siguiendo guía de DREAM
- ☐ Compilar DREAM: mkdir build && cd build && cmake .. && make -j4
- ☐ Configurar PYTHONPATH para el frontend Python
- ☐ Ejecutar ejemplo básico de documentación
- ☐ Verificar que dreami ejecuta sin errores

Día 5-6: Instalación de DINA-IMAS

Horas estimadas: 8-10h

Pasos de instalación:

- ☐ Clonar repositorio: git clone https://github.com/iterorganization/DINA-IMAS.git
- ☐ Ejecutar script de configuración: source ci_header.sh
- ☐ Compilar DINA: make
- ☐ Configurar archivos XML de parámetros
- ☐ Probar GUI de preparación de escenarios (tools/GUI/main.py)
- ☐ Ejecutar escenario de ejemplo de la carpeta machines/

Día 7: Integración y Verificación

Horas estimadas: 4-5h

- ☐ Verificar que ambos códigos pueden importarse desde Python
- ☐ Crear script de verificación de instalación
- ☐ Documentar proceso de instalación en docs/INSTALLATION.md
- ☐ Commit: "feat: complete installation of DINA and DREAM"

⚠ **Plan B si la instalación falla:** Continuar con modelo fenomenológico de disrupción (Sección 8 de este documento)

SEMANA 2: Configuración de Escenarios DINA

Objetivo: Generar escenarios normales y pre-disruptivos con DINA

Día 8-9: Entender Configuración de DINA

Horas estimadas: 8-10h

Archivos de configuración clave:

- **codeparam_dina.xml:** Parámetros principales de simulación
- **codeparam_kmc.xml:** Configuración del solver de equilibrio
- **wfconfig.xml:** Workflow y secuencia de ejecución
- **machines/[device]/:** Configuraciones específicas por tokamak (ITER, DIII-D, etc.)

Tareas:

- ☐ Estudiar estructura de archivos XML de configuración
- ☐ Identificar parámetros modificables: Ip, ne, heating, etc.
- ☐ Ejecutar escenario ITER 15MA de referencia
- ☐ Visualizar outputs y entender formato de datos

Día 10-11: Crear Escenarios No-Disruptivos

Horas estimadas: 8-10h

Objetivo: Crear 5+ escenarios base con variaciones

- ☐ Escenario 1: ITER 15MA estándar (referencia)
- ☐ Escenario 2: ITER con Ip reducida (12 MA)
- ☐ Escenario 3: ITER con densidad alta (pero segura)
- ☐ Escenario 4: ITER con beta alto (pero seguro)
- ☐ Escenario 5: Variaciones de potencia de calentamiento
- ☐ Guardar configuraciones en configs/normal/

Día 12-13: Crear Escenarios Pre-Disruptivos

Horas estimadas: 10-12h

Objetivo: Crear escenarios que evolucionen hacia límites

- ☐ Escenario A: Rampa de densidad hacia límite de Greenwald (n/nG → 0.95)
- ☐ Escenario B: Rampa de beta hacia límite de Troyon (βN → 3.2)
- ☐ Escenario C: Reducción de q95 hacia valor crítico (q95 → 2.2)
- ☐ Escenario D: Pérdida de potencia de calentamiento
- ☐ Implementar detector de t_trigger basado en límites
- ☐ Guardar configuraciones en configs/disruptive/

Día 14: Automatización y Scripts

Horas estimadas: 5-6h

- ☐ Crear script Python para generar variaciones de escenarios
- ☐ Implementar parametrización aleatoria dentro de rangos seguros
- ☐ Crear función de extracción de señales de output DINA
- ☐ Documentar en docs/DINA_SCENARIOS.md

✓ **Checkpoint Semana 2:** Al menos 10 escenarios DINA ejecutándose correctamente (5 normales, 5 pre-disruptivos)

SEMANA 3: Configuración de DREAM y Handoff

Objetivo: Simular disrupciones con DREAM usando condiciones iniciales de DINA

Día 15-16: Entender API de DREAM

Horas estimadas: 8-10h

Componentes clave del API Python:

- **DREAMSettings:** Objeto principal de configuración
- **ds.radialgrid:** Configuración de geometría (R0, a, B0, Nr)
- **ds.eqsys.T_cold:** Temperatura electrónica
- **ds.eqsys.n_i:** Especies iónicas
- **ds.eqsys.E_field:** Campo eléctrico
- **ds.solver / ds.timestep:** Configuración numérica

Tareas:

- ☐ Estudiar documentación de DREAM (https://ft.nephy.chalmers.se/dream)
- ☐ Ejecutar ejemplos de la carpeta examples/
- ☐ Entender estructura de DREAMOutput
- ☐ Identificar parámetros clave para simular TQ y CQ

Día 17-18: Implementar Handoff DINA → DREAM

Horas estimadas: 10-12h

Implementar función perform_handoff():

- ☐ Extraer perfiles de DINA en t_trigger
- ☐ Interpolar perfiles al grid radial de DREAM
- ☐ Configurar ds.radialgrid con geometría correcta
- ☐ Configurar ds.eqsys.T_cold con perfil de Te
- ☐ Configurar ds.eqsys.n_i con especies iónicas
- ☐ Configurar ds.eqsys.E_field self-consistent
- ☐ Probar handoff con un escenario de prueba

Día 19-20: Configurar Simulación de Disrupción

Horas estimadas: 10-12h

Configurar Thermal Quench:

- ☐ Configurar transporte anómalo (D ~ 100-500 m²/s)
- ☐ Configurar duración del TQ (~1 ms)
- ☐ Incluir radiación de impurezas

Configurar Current Quench:

- ☐ Configurar condición de borde para E_field
- ☐ Configurar inverse_wall_time
- ☐ Configurar duración del CQ (~50 ms)
- ☐ Desactivar runaway electrons (simplificación)

Día 21: Validación del Pipeline Completo

Horas estimadas: 5-6h

- ☐ Ejecutar pipeline completo DINA → DREAM para 1 disparo
- ☐ Verificar continuidad de señales en t_trigger
- ☐ Visualizar Ip(t), Te(t), ne(t) del disparo completo
- ☐ Verificar física: Ip decae a 0, Te colapsa

- ☐ Documentar en docs/HANDOFF.md

✓ **Checkpoint Semana 3:** Pipeline DINA→DREAM funcionando end-to-end para al menos 1 disparo disruptivo

SEMANA 4: Generación Masiva de Dataset

Objetivo: Generar dataset completo con 1,000+ disparos

Día 22-24: Automatización de Generación

Horas estimadas: 12-15h

Implementar clase DataGenerationPipeline:

- ☐ Método generate_non_disruptive_shots(n_shots)
- ☐ Método generate_disruptive_shots(n_shots)
- ☐ Parametrización aleatoria de escenarios
- ☐ Manejo de errores y reintentos
- ☐ Logging de progreso
- ☐ Guardado incremental (cada 10 disparos)

Día 25-26: Ejecutar Generación

Horas estimadas: Variable (depende de recursos)

Plan de ejecución:

- ☐ Generar 500 disparos no-disruptivos (solo DINA)
- ☐ Generar 500 disparos disruptivos (DINA + DREAM)
- ☐ Monitorear uso de recursos (CPU, memoria, disco)
- ☐ Verificar integridad de datos generados

Día 27-28: Post-procesamiento y Formato

Horas estimadas: 8-10h

- ☐ Combinar señales DINA + DREAM para disparos disruptivos
- ☐ Resamplear a frecuencia uniforme (10 kHz)
- ☐ Normalizar señales (z-score)
- ☐ Crear ventanas temporales (últimos 100-500 ms)
- ☐ Guardar en formato HDF5 compatible con tokamak-fno
- ☐ Generar metadata del dataset

✓ **Checkpoint Semana 4:** Dataset de 1,000 disparos en formato HDF5

SEMANA 5: Validación y Análisis del Dataset

Objetivo: Verificar calidad física y estadística del dataset

Día 29-30: Validación Física

Horas estimadas: 8-10h

- ☐ Verificar que Ip siempre es positivo y continuo
- ☐ Verificar que βN < 5 (límite físico)
- ☐ Verificar que q95 > 1 cuando está definido
- ☐ Verificar decaimiento de Ip en disparos disruptivos
- ☐ Verificar colapso de Te durante Thermal Quench
- ☐ Identificar y descartar disparos anómalos

Día 31-32: Análisis Estadístico

Horas estimadas: 8-10h

- ☐ Calcular distribución de parámetros por clase
- ☐ Verificar separabilidad de clases (diferencias significativas)
- ☐ Analizar distribución de t_trigger para disruptivos
- ☐ Analizar distribución de causas de disrupción
- ☐ Crear notebook: 01_dataset_analysis.ipynb

Día 33-34: Visualizaciones

Horas estimadas: 8-10h

- ☐ Gráficas de ejemplo: disparo normal vs disruptivo
- ☐ Histogramas de parámetros por clase
- ☐ Scatter plots de límites operacionales
- ☐ Heatmaps de correlación entre señales
- ☐ Guardar figuras en results/

Día 35: Prueba de Integración con tokamak-fno

Horas estimadas: 5-6h

- ☐ Copiar dataset a repositorio tokamak-fno
- ☐ Verificar que DataLoader de tokamak-fno carga correctamente
- ☐ Entrenar modelo baseline con nuevo dataset
- ☐ Comparar métricas con dataset sintético simple

✓ **Checkpoint Semana 5:** Dataset validado y compatible con tokamak-fno

SEMANA 6: Documentación y Entrega Final

Objetivo: Repositorio completo, documentado y reproducible

Día 36-37: Documentación Técnica

Horas estimadas: 10-12h

- ☐ Actualizar README.md con descripción completa
- ☐ Crear docs/PHYSICS.md: explicación de física subyacente
- ☐ Crear docs/PIPELINE.md: descripción del pipeline
- ☐ Crear docs/API.md: documentación de funciones
- ☐ Añadir docstrings a todas las funciones públicas

Día 38-39: Code Quality

Horas estimadas: 8-10h

- ☐ Añadir type hints a todas las funciones
- ☐ Verificar estilo con flake8/black
- ☐ Crear tests básicos: test_pipeline.py
- ☐ Crear script de verificación: scripts/verify_installation.py
- ☐ Verificar reproducibilidad con seeds fijos

Día 40-41: Empaquetado y Scripts

Horas estimadas: 8-10h

- ☐ Crear requirements.txt / environment.yml
- ☐ Crear scripts/generate_dataset.py con CLI
- ☐ Crear scripts/validate_dataset.py
- ☐ Crear configs/ con ejemplos de configuración
- ☐ Añadir LICENSE (MIT)

Día 42: Verificación Final y Release

Horas estimadas: 4-5h

- ☐ Clonar repositorio en directorio limpio
- ☐ Seguir instrucciones de README desde cero
- ☐ Verificar que todo funciona
- ☐ Crear tag de versión: v1.0.0
- ☐ Publicar dataset en repositorio o enlace externo

✓ **Checkpoint FINAL:** Repositorio showcase-ready con dataset validado

🗂 Estructura Final del Repositorio

tokamak-disruption-simulator/

```
├── README.md
├── LICENSE
├── requirements.txt
├── environment.yml
├── setup.py
│
├── configs/
│   ├── dina/
│   │   ├── iter_15ma_standard.yaml
│   │   ├── iter_density_limit.yaml
│   │   └── iter_beta_limit.yaml
│   ├── dream/
│   │   ├── thermal_quench.yaml
│   │   └── current_quench.yaml
│   └── generation.yaml
│
├── src/
│   ├── __init__.py
│   ├── dina/
│   │   ├── __init__.py
│   │   ├── scenario.py          # Configuración de escenarios
│   │   ├── runner.py            # Ejecución de DINA
│   │   └── extractor.py         # Extracción de señales
│   ├── dream/
│   │   ├── __init__.py
│   │   ├── configurator.py      # Configuración de DREAM
│   │   ├── runner.py            # Ejecución de DREAM
│   │   └── extractor.py         # Extracción de outputs
│   ├── pipeline/
│   │   ├── __init__.py
│   │   ├── handoff.py           # Transferencia DINA→DREAM
│   │   ├── detector.py          # Detección de t_trigger
│   │   ├── combiner.py          # Combinar outputs
│   │   └── generator.py         # Pipeline completo
│   └── utils/
│       ├── physics.py           # Constantes y límites
│       ├── io.py                # Lectura/escritura HDF5
│       └── visualization.py     # Gráficas
│
├── scripts/
│   ├── generate_dataset.py      # CLI principal
│   ├── validate_dataset.py      # Validación
│   └── verify_installation.py   # Test de instalación
│
├── notebooks/
│   ├── 01_dataset_analysis.ipynb
│   ├── 02_physics_validation.ipynb
│   └── 03_example_shots.ipynb
│
├── docs/
│   ├── INSTALLATION.md
│   ├── PHYSICS.md
│   ├── PIPELINE.md
│   └── API.md
│
├── data/
│   ├── README.md
```

```
│   └── tokamak_disruption_dataset.h5  # Dataset final
│
├── results/
│   └── figures/
│
└── tests/
    ├── test_dina.py
    ├── test_dream.py
    └── test_pipeline.py
```

🔧 Plan B: Modelo Fenomenológico de Disrupción

Si la instalación de DREAM resulta demasiado compleja o los recursos computacionales son insuficientes, se puede implementar un modelo fenomenológico que capture la física esencial de las disrupciones sin necesidad de ejecutar DREAM.

Física del Modelo Simplificado

Thermal Quench (TQ)

El TQ se modela como un decaimiento exponencial de la temperatura:

$$Te(t) = (Te_0 - Te\_final) \times \exp(-t/\tau\_TQ) + Te\_final$$

Donde:

- **Te_0:** Temperatura inicial (~10 keV)
- **Te_final:** Temperatura post-TQ (~10 eV)
- **τ_TQ:** Tiempo característico (~0.5-2 ms)

Current Quench (CQ)

El CQ comienza después del TQ y se modela también como decaimiento exponencial:

$$Ip(t) = Ip_0 \times \exp(-(t - t\_TQ)/\tau\_CQ)$$

Donde:

- **Ip_0:** Corriente inicial (~15 MA para ITER)
- **t_TQ:** Fin del Thermal Quench
- **τ_CQ:** Tiempo característico (~20-100 ms)

Otros Parámetros Durante Disrupción

- **βN:** Escala con Te → colapsa durante TQ
- **li:** Decae más lentamente que Te
- **q95:** Pierde sentido físico durante disrupción → NaN
- **ne:** Aproximadamente constante o ligero aumento

Ventajas y Limitaciones

Ventajas	Limitaciones
No requiere instalar DREAM	Menos preciso físicamente
Ejecución muy rápida	No modela efectos cinéticos
Fácil de parametrizar	Sin campo eléctrico inductivo
Suficiente para proof-of-concept	Sin runaway electrons

Recursos y Referencias

Repositorios de Código

- **DREAM:** https://github.com/chalmersplasmatheory/DREAM
- **DINA-IMAS:** https://github.com/iterorganization/DINA-IMAS
- **tokamak-fno (proyecto hermano):** https://github.com/BecerraMiguel/tokamak-fno

Documentación

- **DREAM docs:** https://ft.nephy.chalmers.se/dream
- **ITER IMAS:** https://www.iter.org/node/20687/release-imas-infrastructure-and-physics-models-open-source

Papers Fundamentales

DREAM

- Hoppe, Embreus, Fülöp (2021) "DREAM: A fluid-kinetic framework for tokamak disruption runaway electron simulations" - Computer Physics Communications 268, 108098

Física de Disrupciones

- Hender et al. (2007) "MHD stability, operational limits and disruptions" - Nuclear Fusion 47
- de Vries et al. (2011) "Survey of disruption causes at JET" - Nuclear Fusion 51
- ITER Physics Basis (1999) - Nuclear Fusion 39

Machine Learning para Fusión

- Kates-Harbeck et al. (2019) "Predicting disruptive instabilities in controlled fusion plasmas through deep learning" - Nature 568
- Rea et al. (2019) "Disruption prediction investigations using machine learning tools on DIII-D" - Plasma Physics and Controlled Fusion

✅ Checklist Final (Día 42)

Código

- ☐ Todo el código está en GitHub
- ☐ No hay credenciales/API keys en repositorio
- ☐ requirements.txt / environment.yml actualizados
- ☐ Código sigue estilo consistente (PEP8)
- ☐ Type hints en >80% de funciones

Dataset

- ☐ Al menos 1,000 disparos generados
- ☐ Balance de clases ~50/50
- ☐ Formato HDF5 compatible con tokamak-fno
- ☐ Validación física completada
- ☐ Metadata incluida en dataset

Documentación

- ☐ README.md completo con instrucciones
- ☐ docs/ con INSTALLATION, PHYSICS, PIPELINE, API
- ☐ Notebooks de análisis funcionales
- ☐ Docstrings en funciones públicas

Reproducibilidad

- ☐ Instrucciones funcionan en máquina limpia
- ☐ Seeds fijados donde aplique
- ☐ Versiones de librerías especificadas
- ☐ Tests básicos pasan

🔬 ¡Éxito con el proyecto! 🔬

Este proyecto contribuye directamente al futuro de la energía limpia.

La fusión nuclear está a una década de viabilidad comercial.

Tu trabajo en generación de datos realistas de disrupciones puede acelerar ese timeline. 🔥