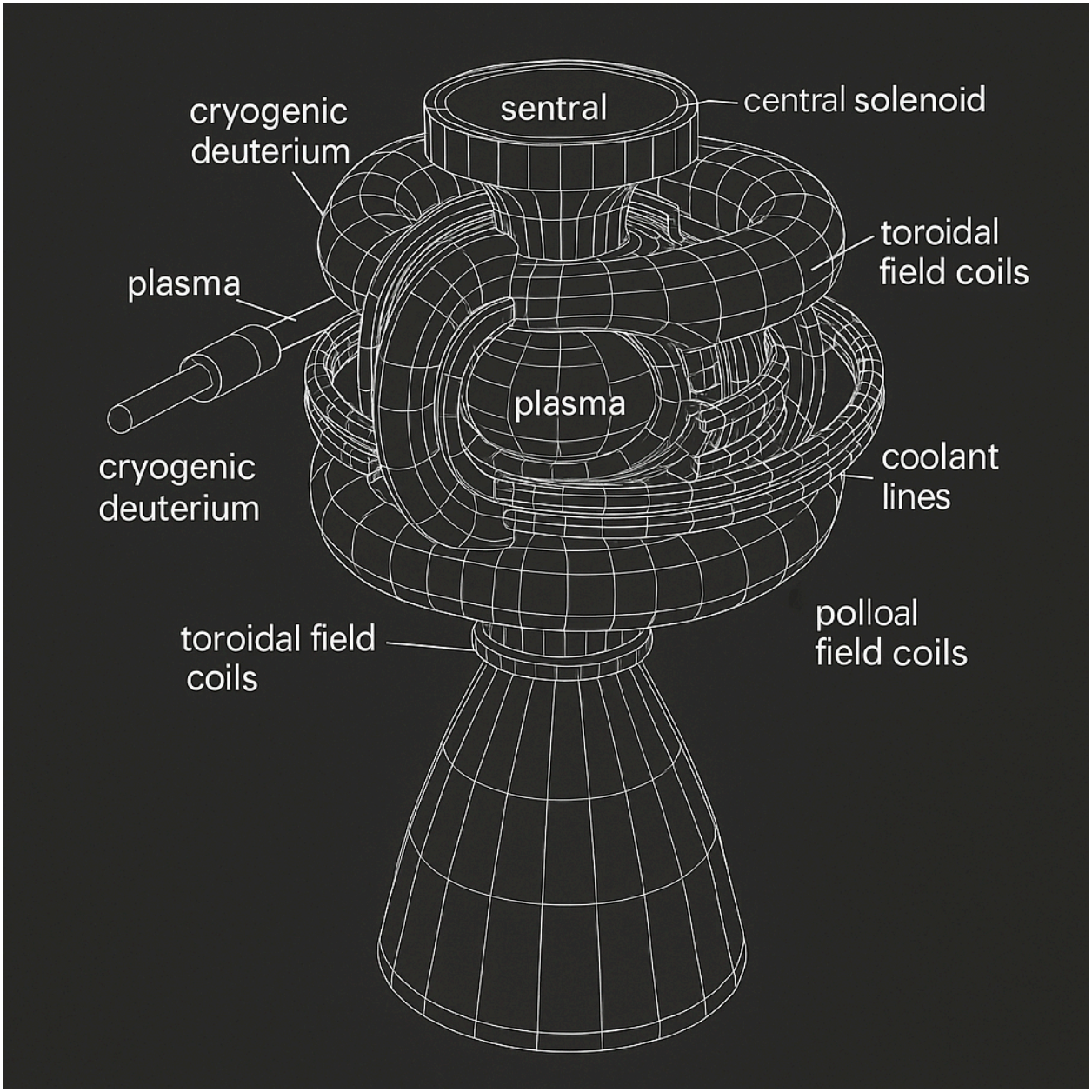
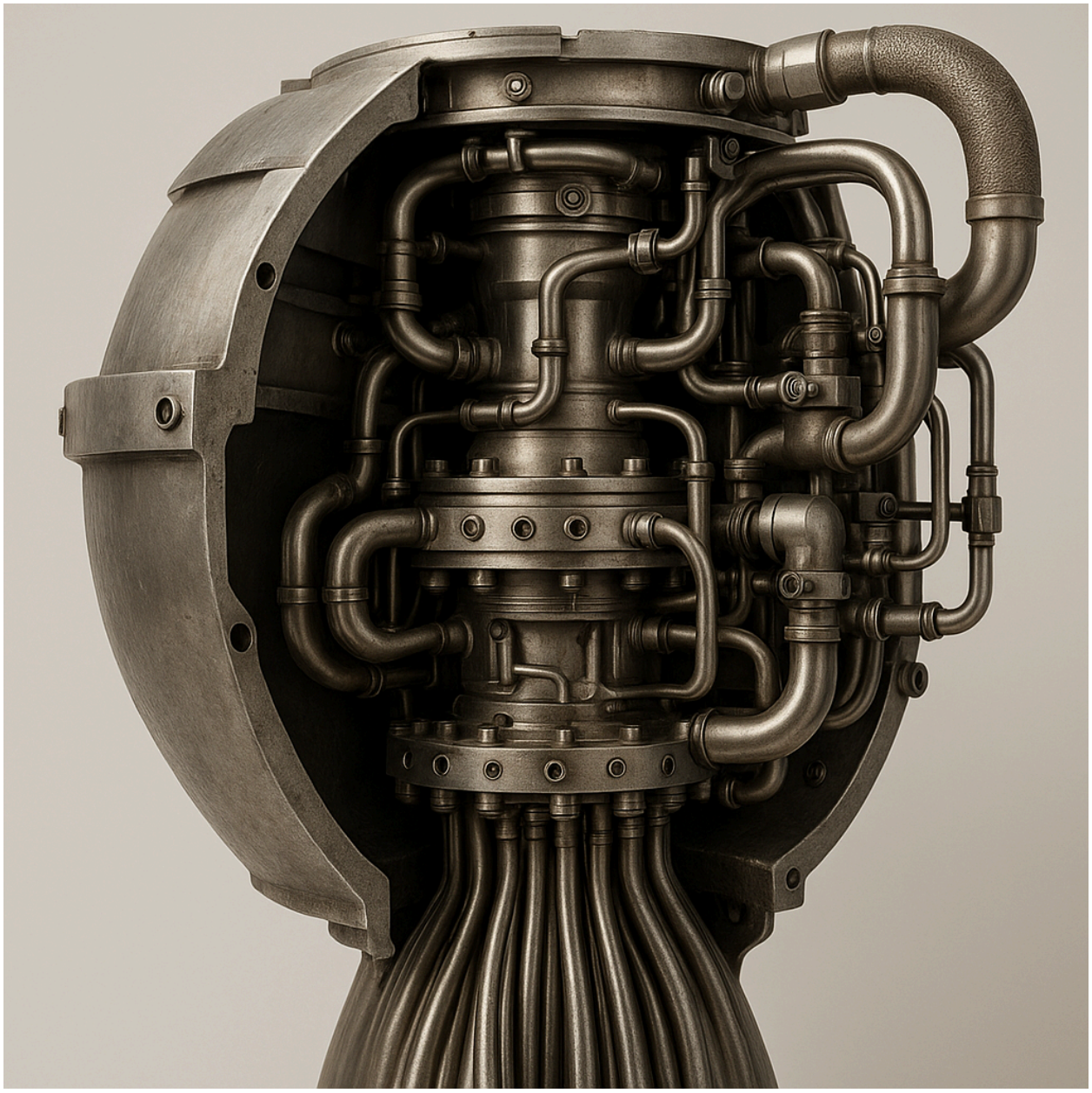
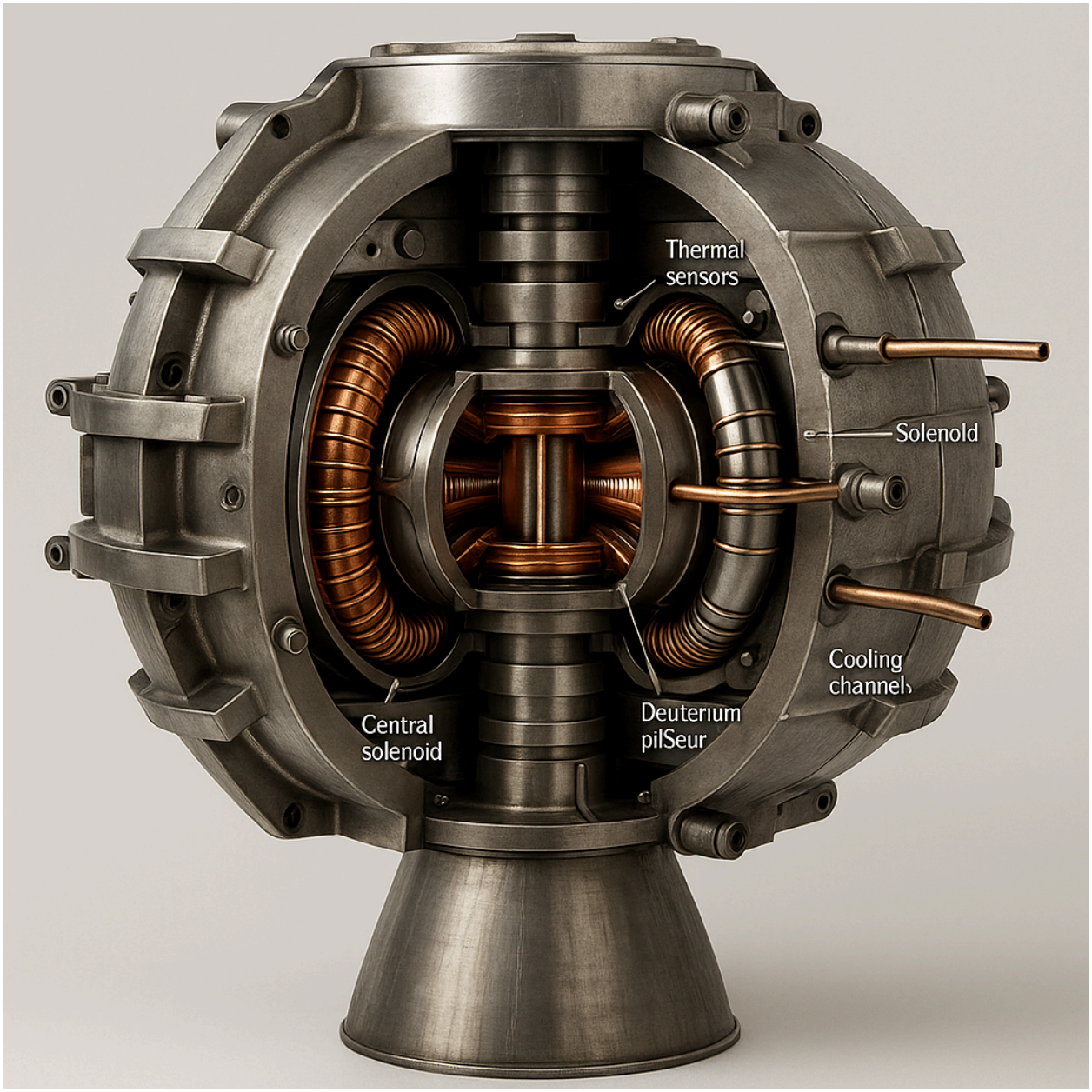
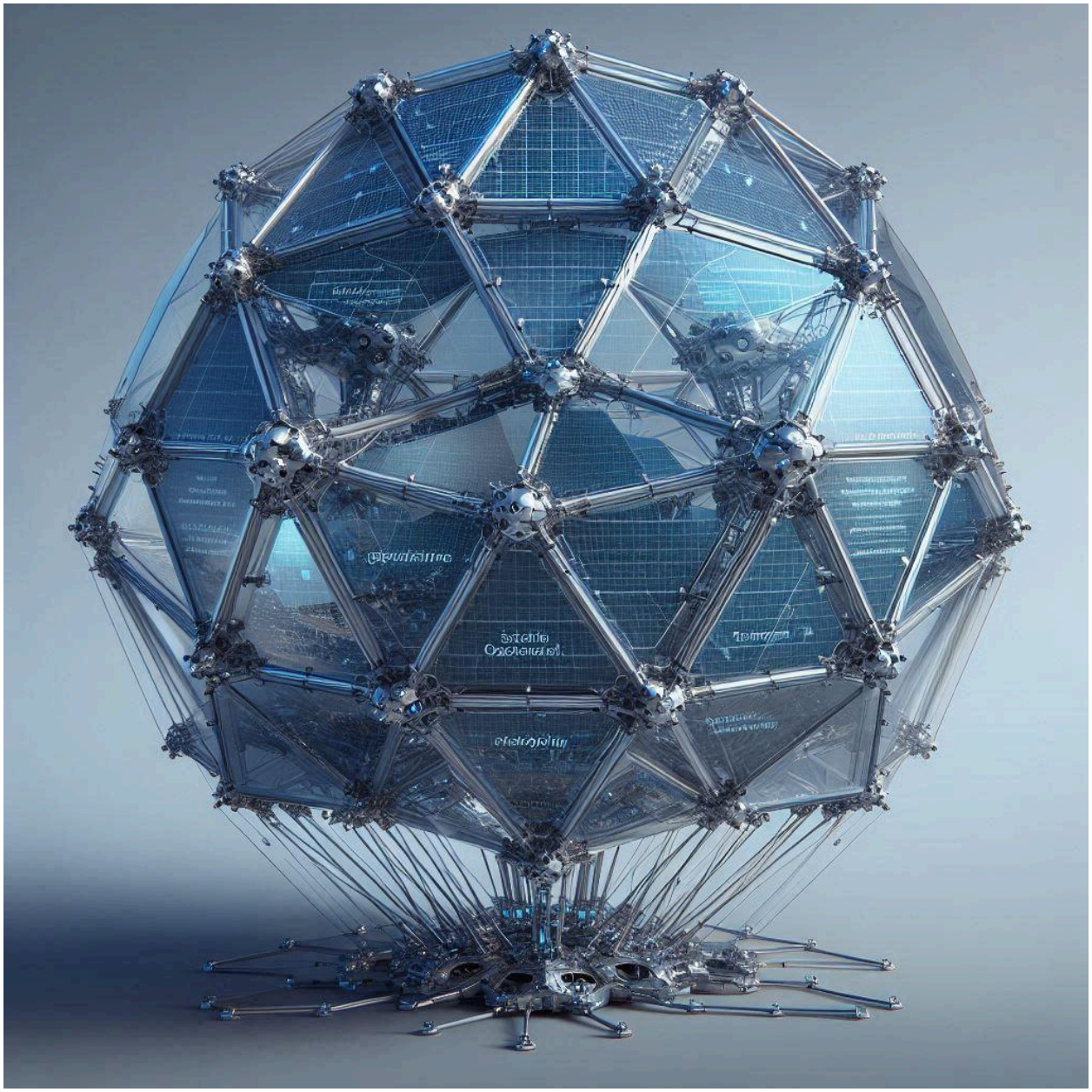
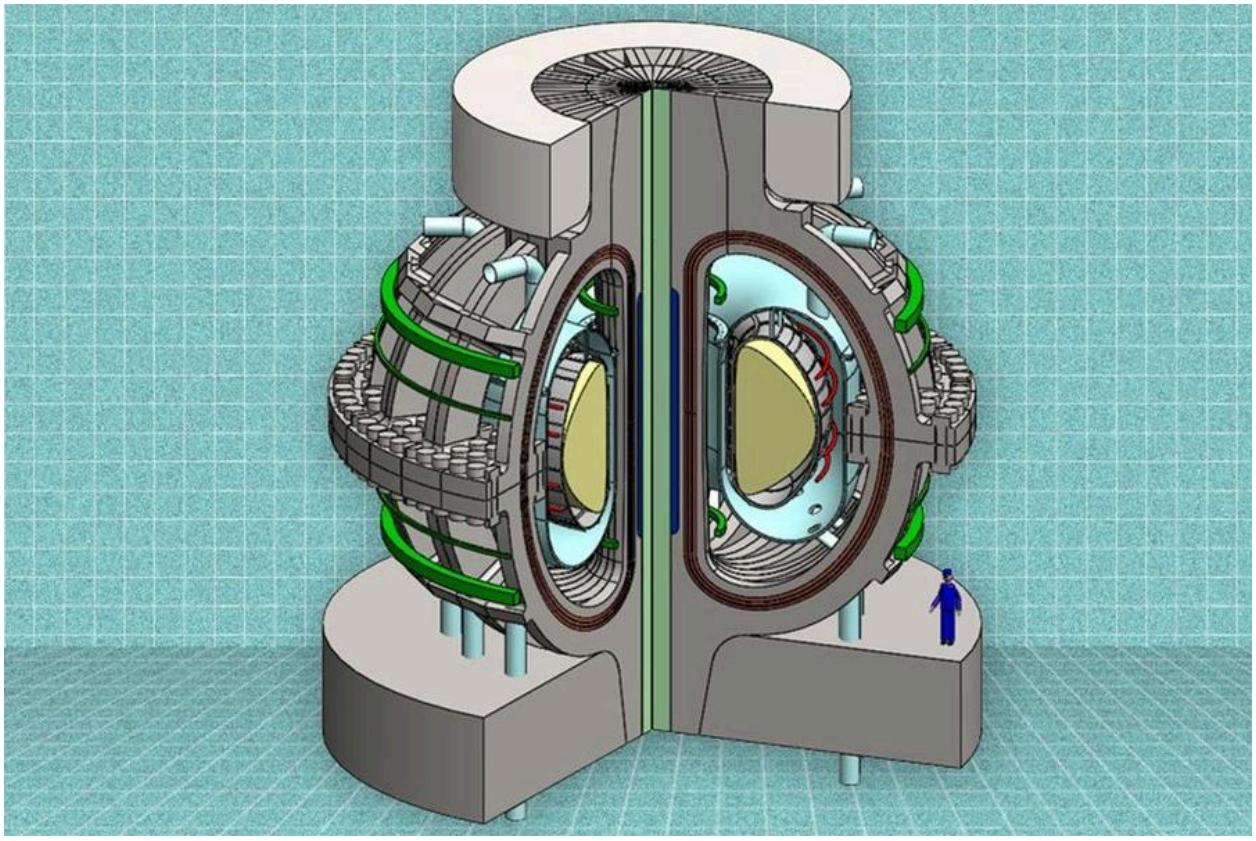
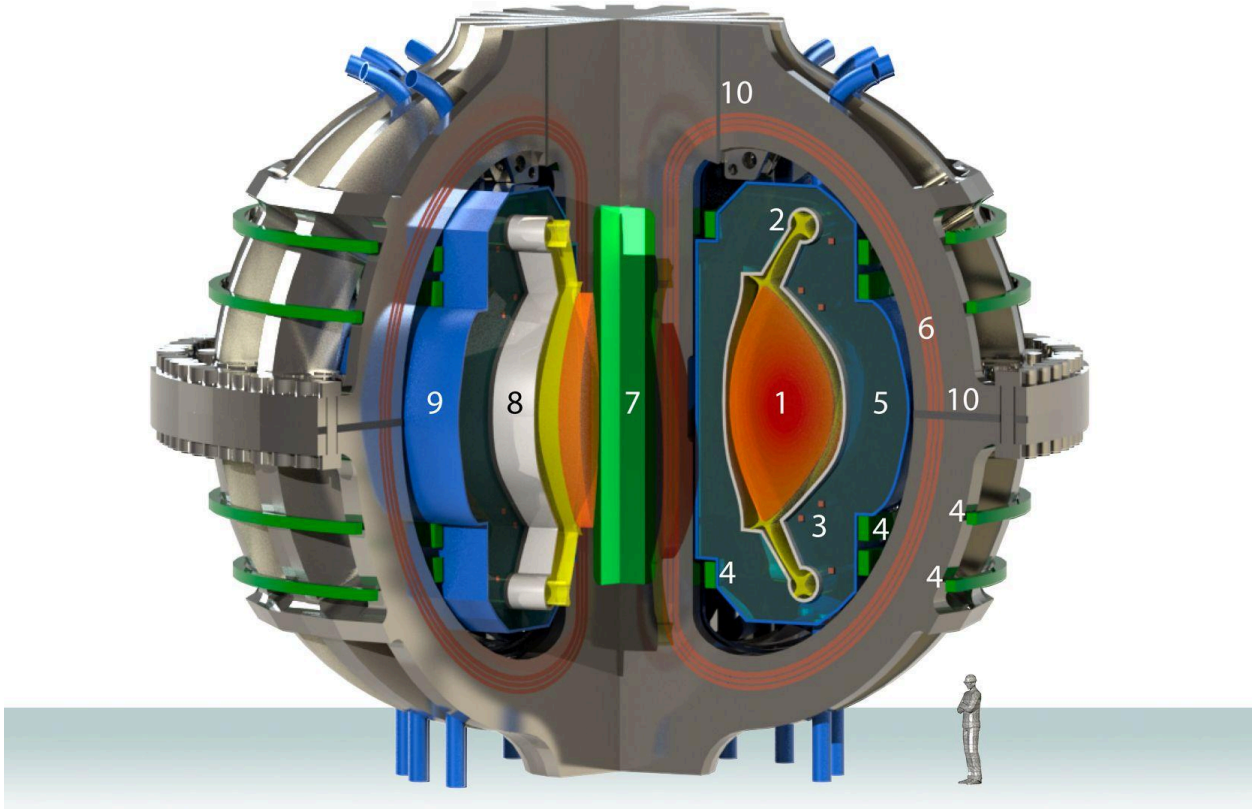

cryogenic
deuterium
sentral
central solenoid
toroidal
field coils
plasma
plasma
cryogenic
deuterium
coolant
lines
toroidal field
coils
polloal
field coils

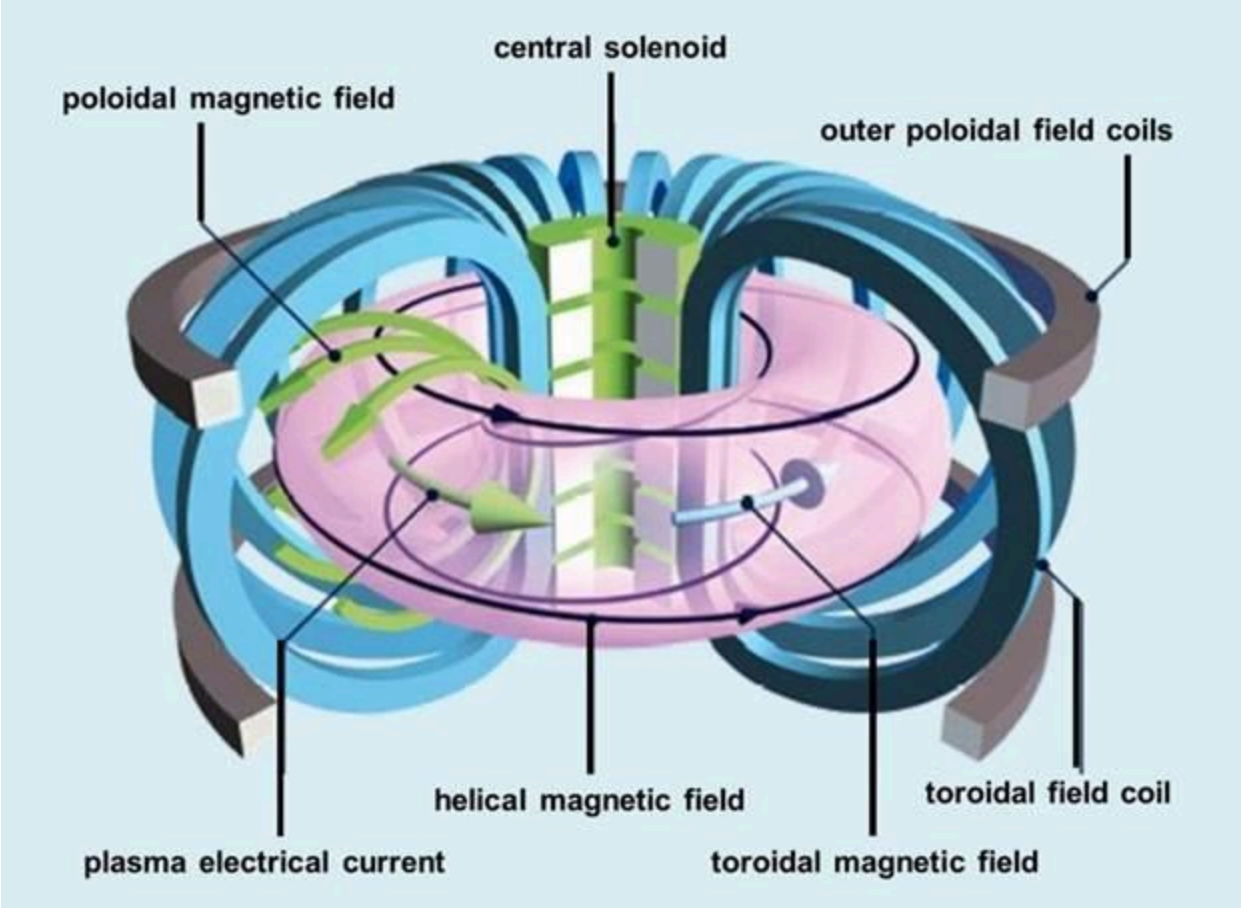


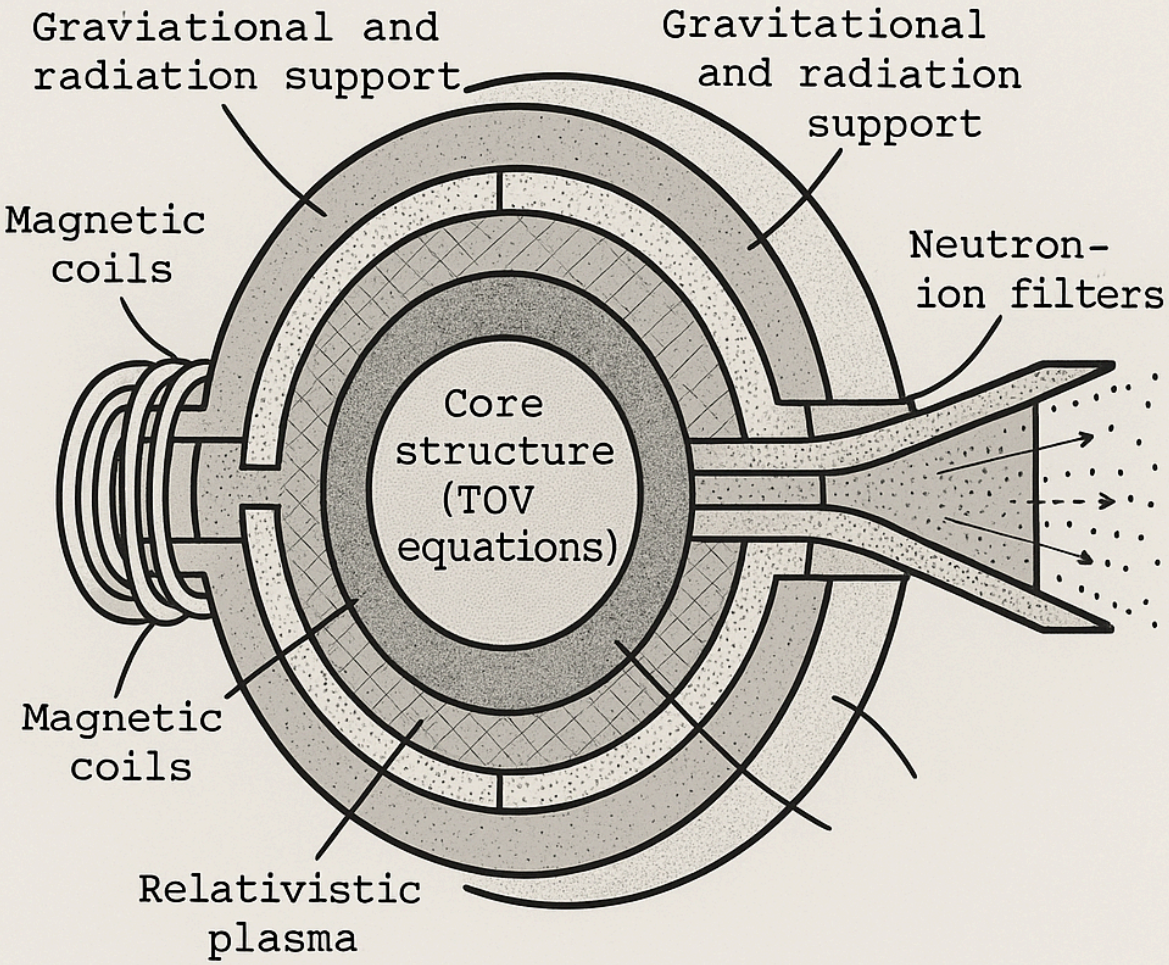

Thermal
sensors
Solenold
Cooling
channels
Central
solenoid
Deuterium
pilSeur



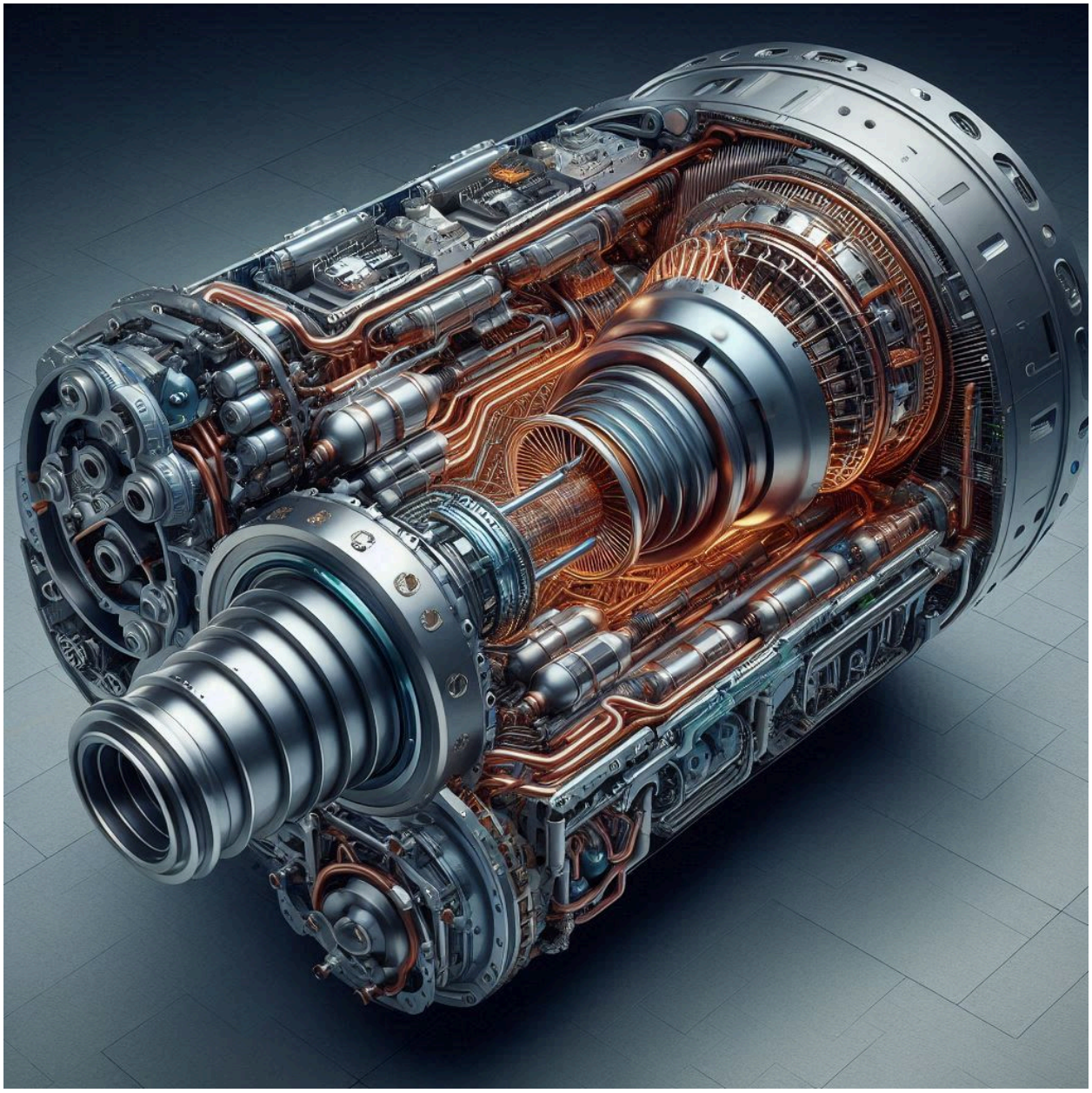


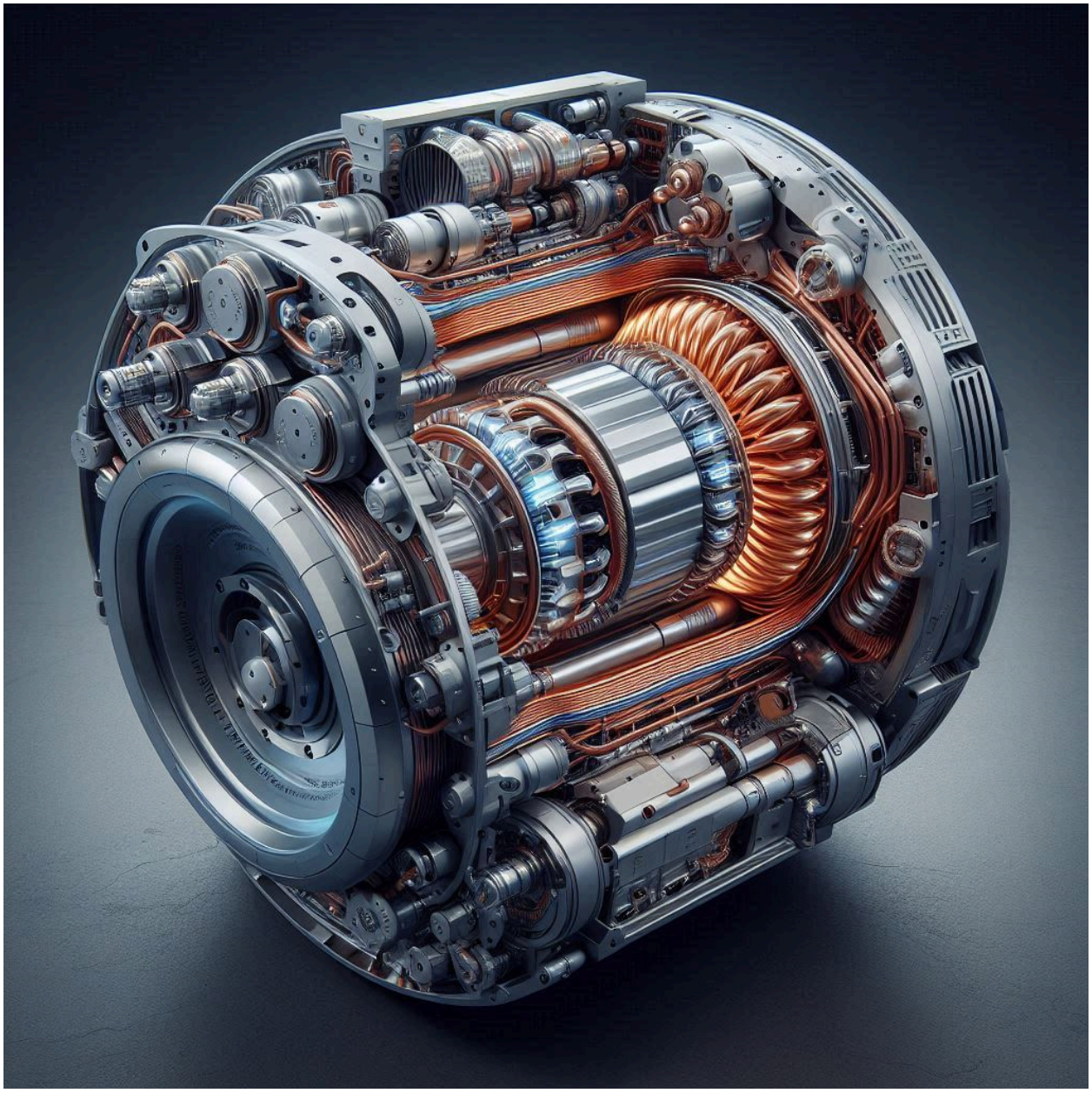

10
2
6
9
8
7
1
5
10
3
4
4
4
4

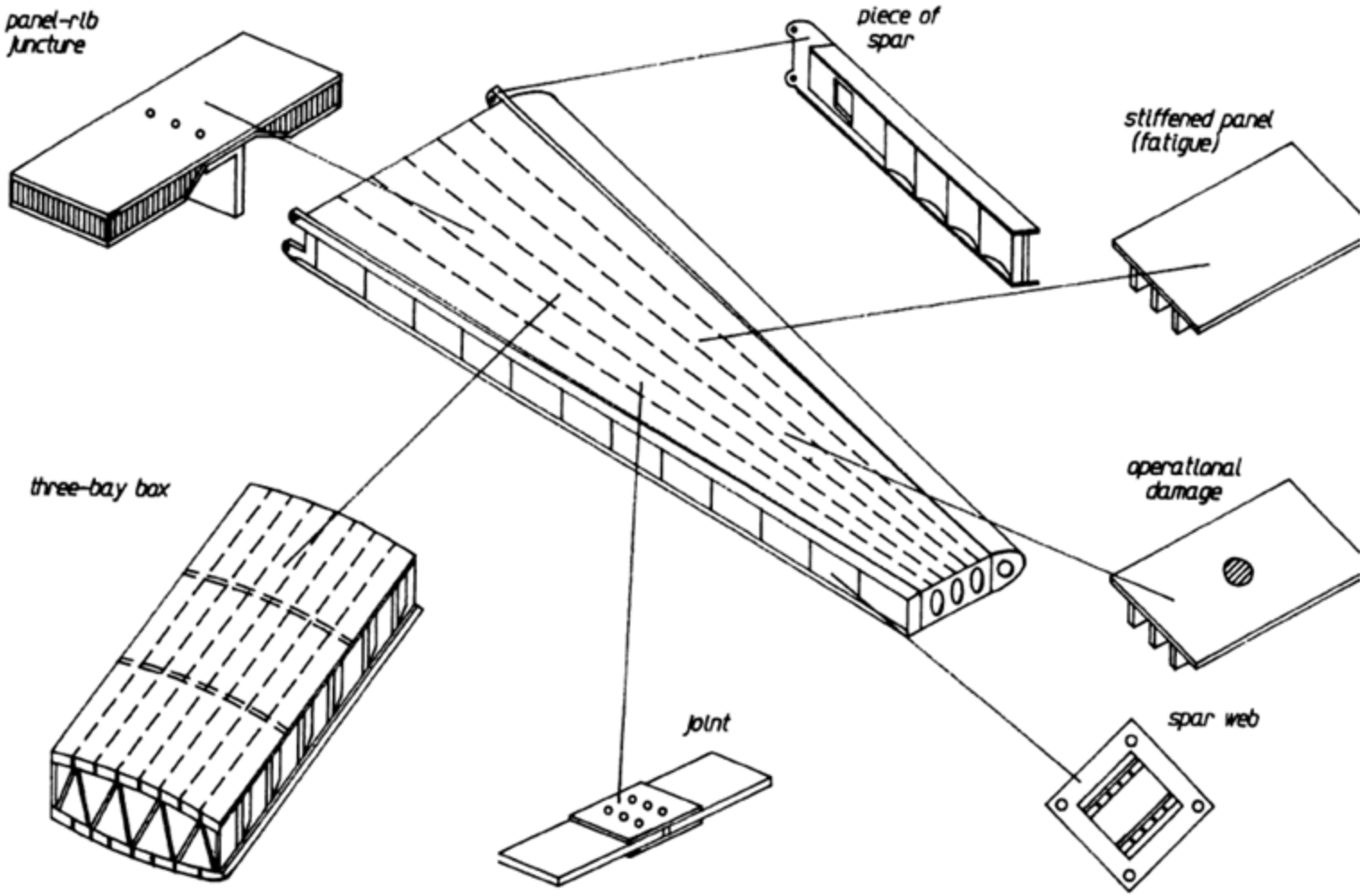

central solenoid
poloidal magnetic field
outer poloidal field coils
helical magnetic field
toroidal field coil
plasma electrical current
toroidal magnetic field

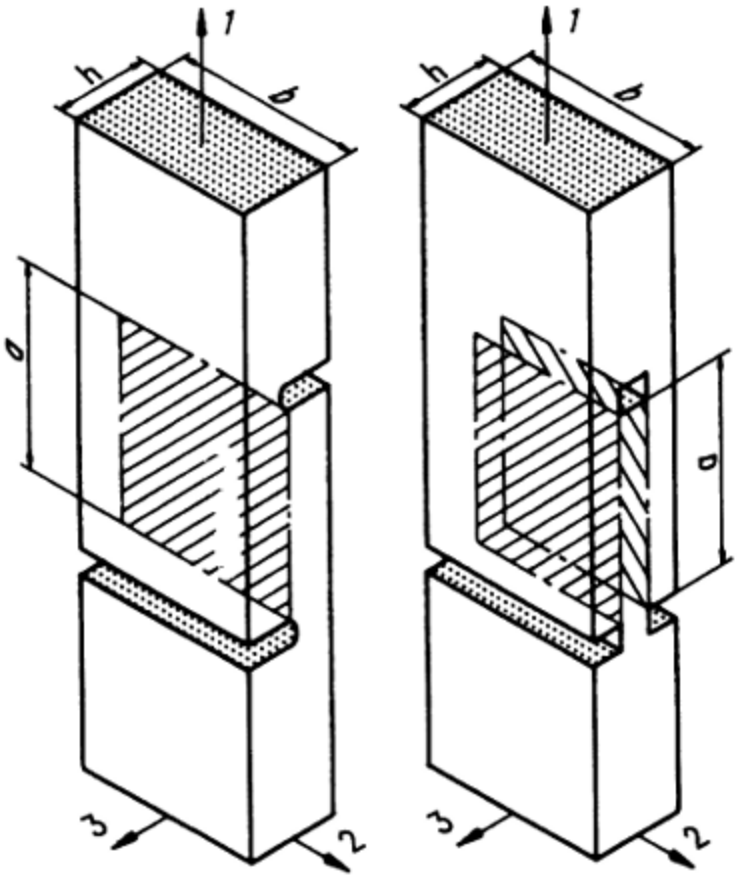


Relativistic plasma reactor with neutron-ion filtering, designed to withstand extreme gravity and radiation



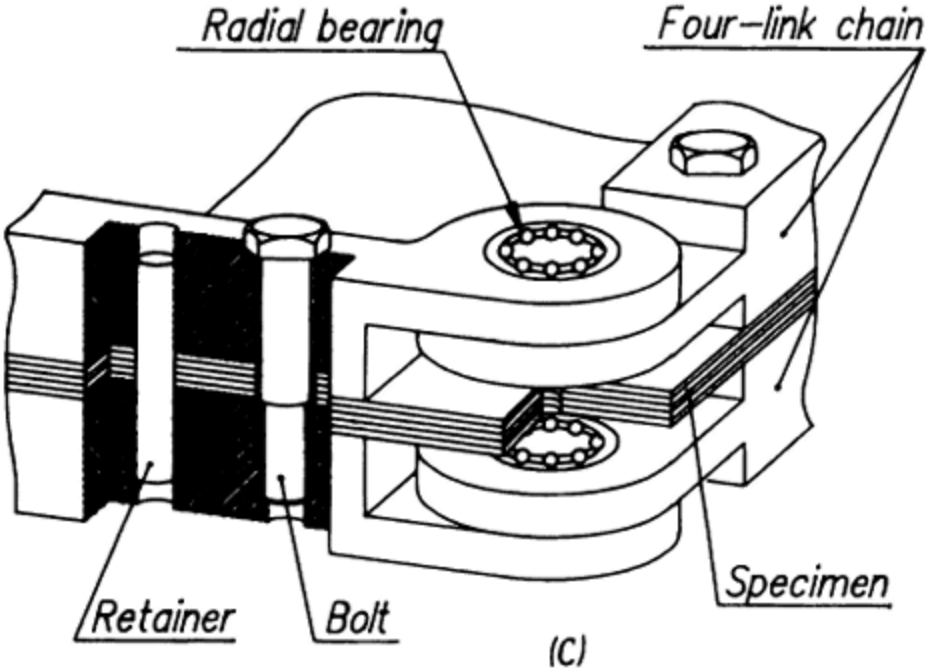



panel-rib juncture
piece of spar
stiffened panel (fatigue)
three-bay box
operational damage
joint
spar web

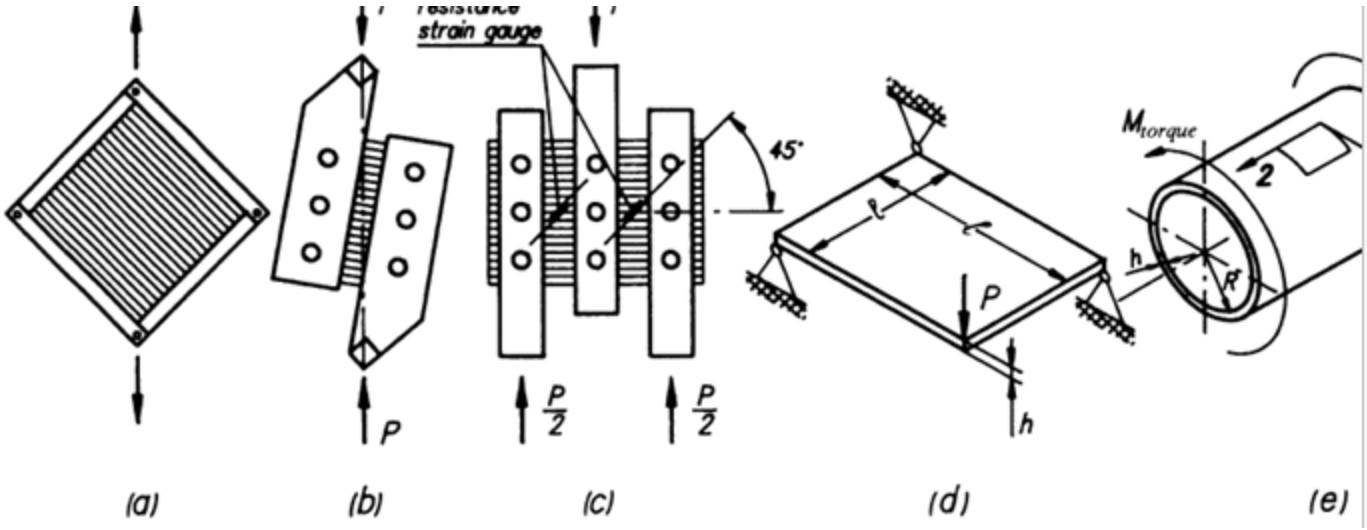


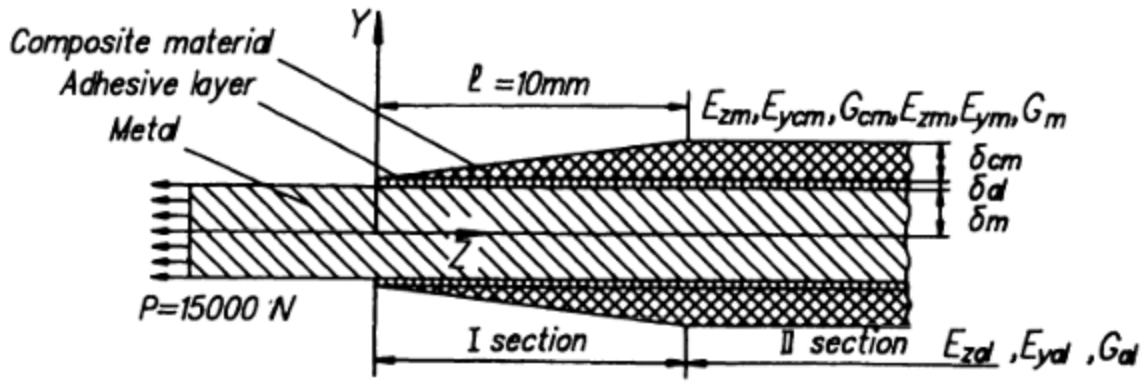
(a) (b)

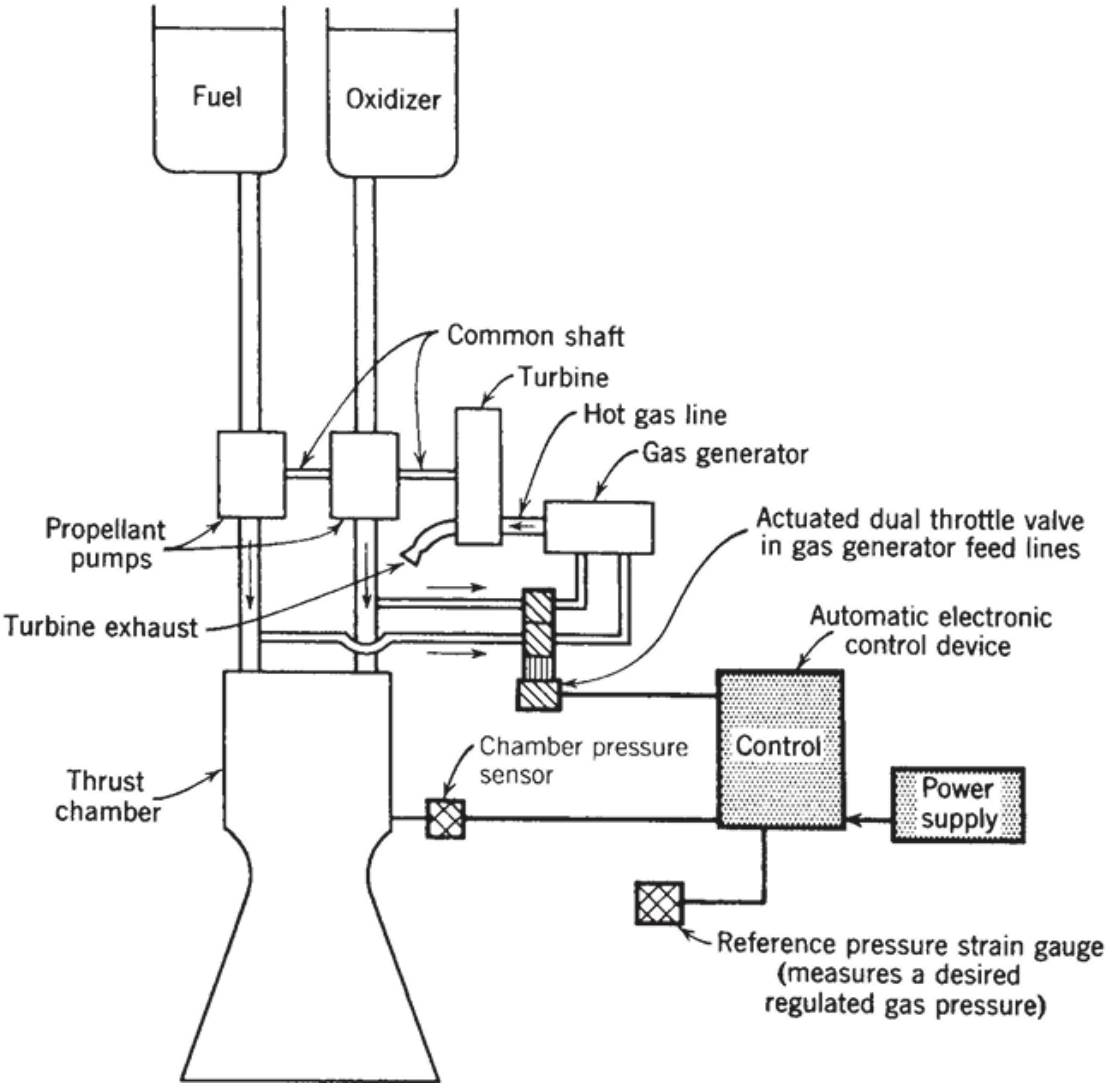
Detail I



(C)


strain gauge
P
P/2
P/2
45°
l
l
P
h
M_torque
2
h
R
(a)
(b)
(c)
(d)
(e)


Composite material
Adhesive layer
Metal
Y
Z
ℓ =10mm
$E_{zm}, E_{ycm}, G_{cm}, E_{zm}, E_{ym}, G_m$
δ_{cm}
δ_{al}
δ_m
P=15000 N
I section
II section
E_{zal} , E_{yal} , G_{al}


Fuel
Oxidizer
Common shaft
Turbine
Hot gas line
Gas generator
Propellant pumps
Actuated dual throttle valve in gas generator feed lines
Turbine exhaust
Automatic electronic control device
Control
Chamber pressure sensor
Thrust chamber
Power supply
Reference pressure strain gauge (measures a desired regulated gas pressure)

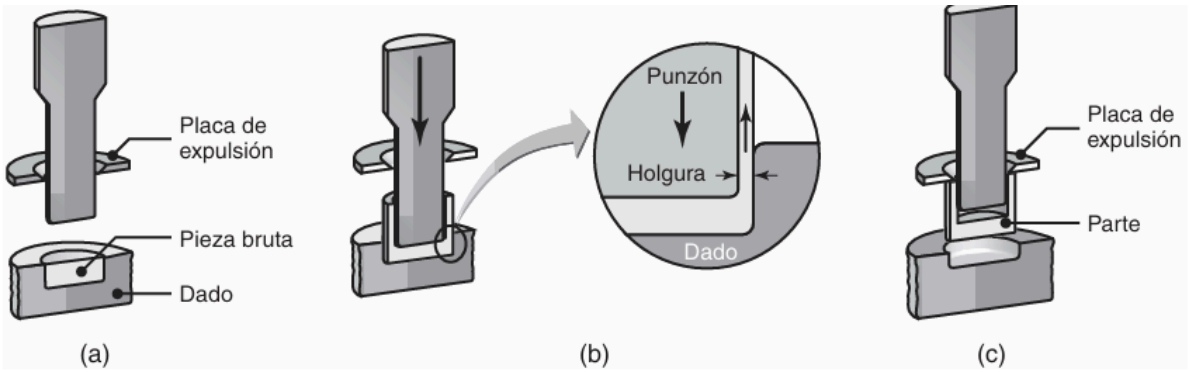
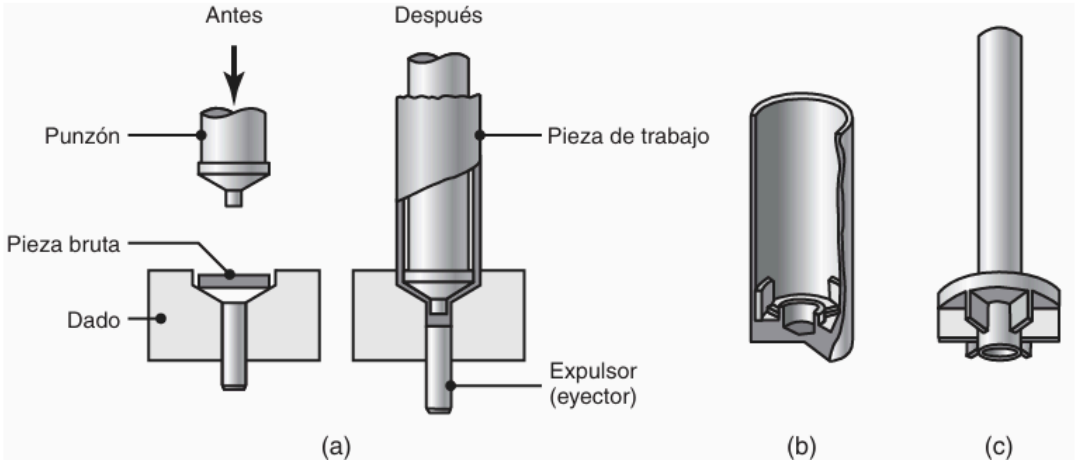
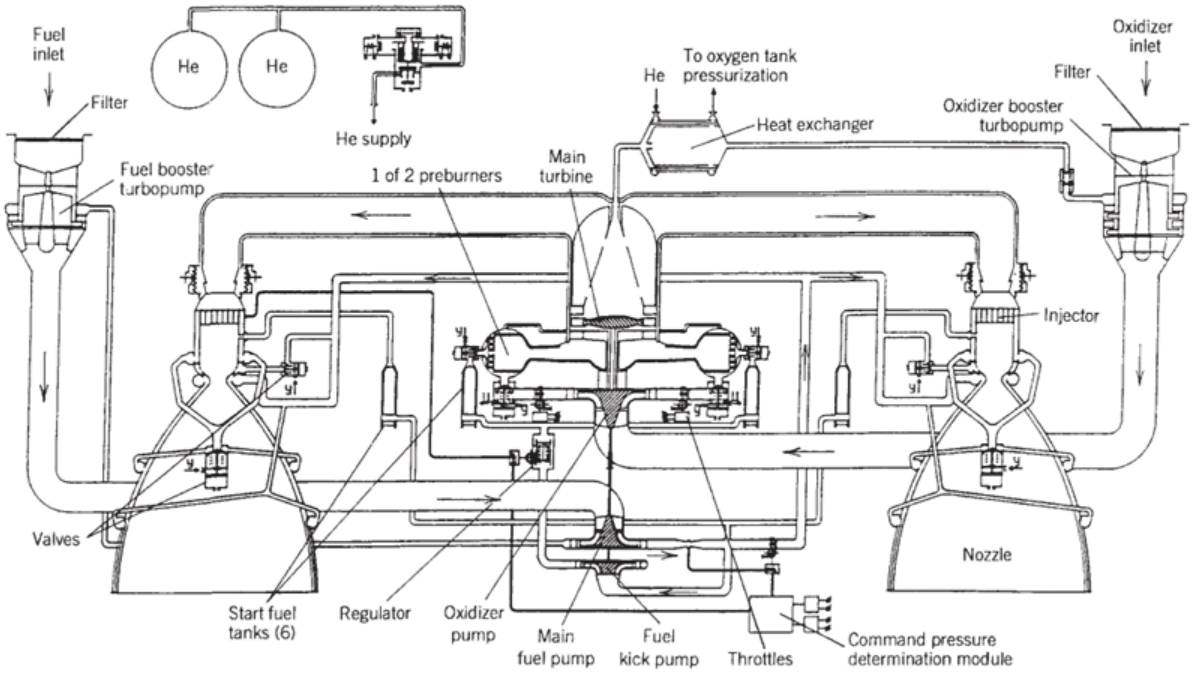
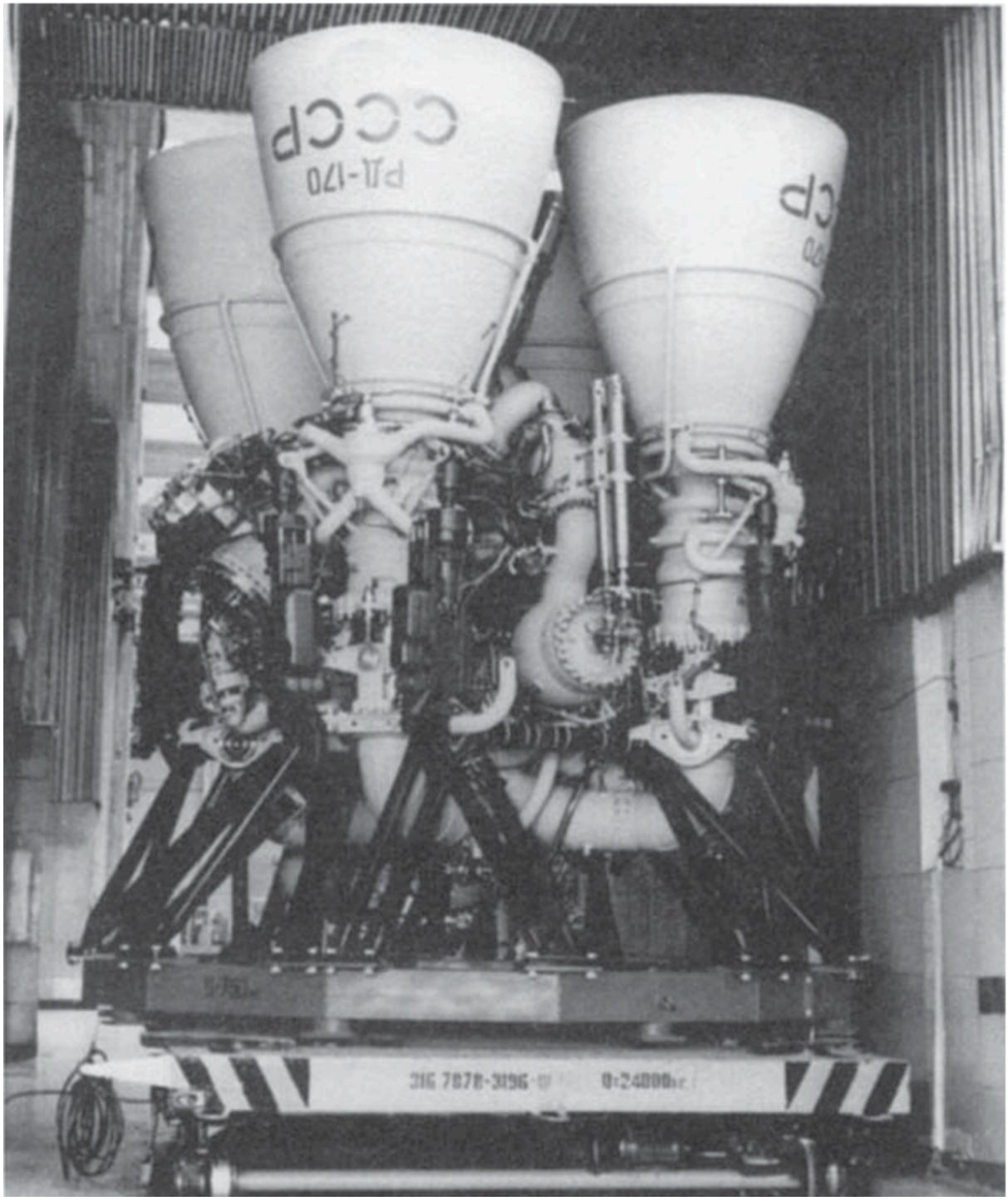
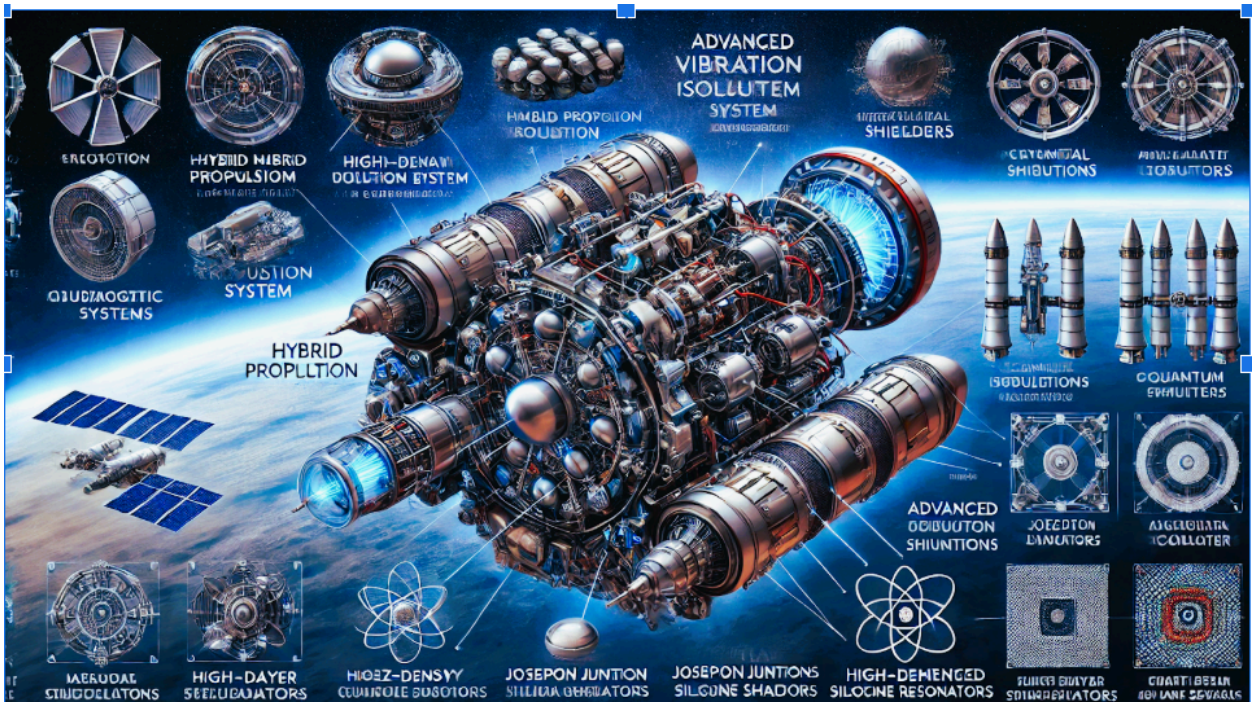


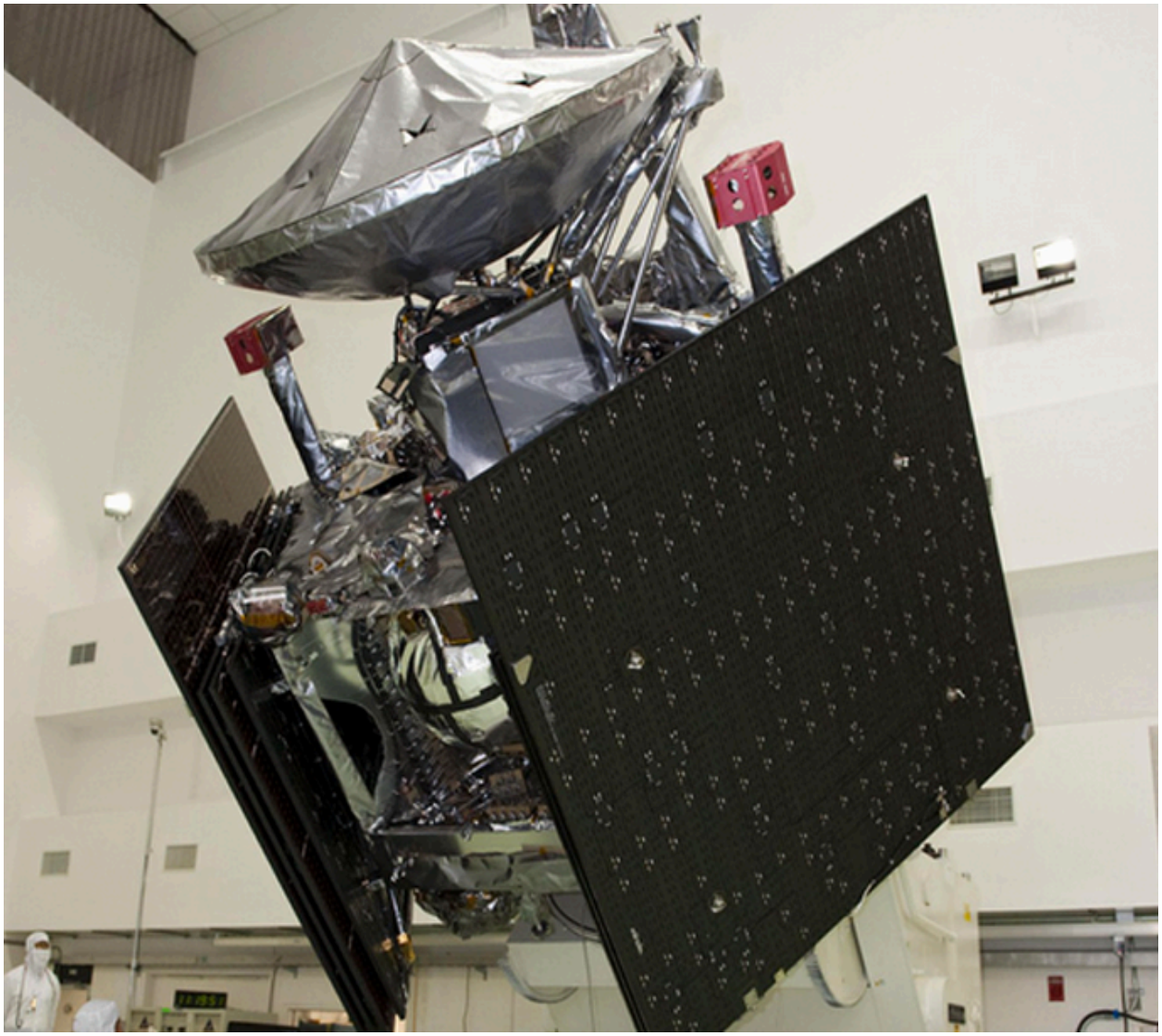
FIGURA 15.14 Esquema del proceso de extrusión por impacto. Las partes extruidas se expulsan mediante el uso de una placa de expulsión, ya que tienden a adherirse al punzón.




Fuel inlet
Filter
He
He
He supply
He
To oxygen tank pressurization
Heat exchanger
Oxidizer inlet
Filter
Oxidizer booster turbopump
Fuel booster turbopump
1 of 2 preburners
Main turbine
Injector
Valves
Nozzle
Start fuel tanks (6)
Regulator
Oxidizer pump
Main fuel pump
Fuel kick pump
Throttles
Command pressure determination module


СССР
РД-170
СССР
316 7878-3196
0-24000


ADVANCED
VIBRATION
ISOLLUTEM
SYSTEM
HYBRID
PROPLLTION
ADVANCED
SHIUNTIONS
HIGH-DENSYY
JOSEPON JUNTION
JOSEPON JUNTIONS
SILGUNE SHADORS
HIGH-DEMENGED
SILOGINE RESONATORS
HIGH-DAYER
ISOLLUTIONS
QUANTUM



CODE

```
import bpy
import math
import mathutils

# Limpia la escena
bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

# Escala base
SCALE = 1.0

# === Tobera estilo Merlin9 ===
bpy.ops.mesh.primitive_cone_add(
```

```
    radius1=0.9 * SCALE, radius2=0.25 * SCALE,
    depth=2.0 * SCALE, location=(0, 0, 0)
)
merlin_nozzle = bpy.context.object
merlin_nozzle.name = "Merlin_Nozzle"

# === Cámara de fusión Tokamak ===
bpy.ops.mesh.primitive_torus_add(
    major_radius=0.75 * SCALE,
    minor_radius=0.15 * SCALE,
    location=(0, 0, 1.5 * SCALE),
    rotation=(0, 0, 0)
)
tokamak_chamber = bpy.context.object
tokamak_chamber.name = "Tokamak_Chamber"

# === Solenoide central ===
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.08 * SCALE, depth=1.4 * SCALE,
    location=(0, 0, 1.5 * SCALE)
)
solenoid = bpy.context.object
solenoid.name = "Central_Solenoid"

# === Bobinas toroidales ===
for angle in range(0, 360, 30):
    x = 0.75 * math.cos(math.radians(angle))
    y = 0.75 * math.sin(math.radians(angle))
    bpy.ops.mesh.primitive_torus_add(
        major_radius=0.05 * SCALE, minor_radius=0.015 * SCALE,
        location=(x, y, 1.5 * SCALE),
        rotation=(math.pi/2, 0, 0)
    )

# === Conductos de refrigeración ===
for i in range(4):
    angle = math.radians(i * 90)
    x = 0.6 * math.cos(angle)
    y = 0.6 * math.sin(angle)
    bpy.ops.mesh.primitive_cylinder_add(
        radius=0.015 * SCALE, depth=0.5 * SCALE,
        location=(x, y, 1.8 * SCALE),
        rotation=(math.pi / 2, 0, angle)
    )

# === Inyector de deuterio ===
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.02 * SCALE, depth=0.4 * SCALE,
    location=(0.9 * SCALE, 0, 1.6 * SCALE),
    rotation=(0, math.pi / 2, 0)
)
```

```
# === Escudo térmico externo ===
bpy.ops.mesh.primitive_cylinder_add(
    radius=1.05 * SCALE, depth=0.4 * SCALE,
    location=(0, 0, 1.5 * SCALE)
)
shield_outer = bpy.context.object
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.95 * SCALE, depth=0.4 * SCALE,
    location=(0, 0, 1.5 * SCALE)
)
bpy.ops.object.modifier_add(type='BOOLEAN')
shield_outer.modifiers['Boolean'].object = bpy.context.object
shield_outer.modifiers['Boolean'].operation = 'DIFFERENCE'
bpy.ops.object.modifier_apply(modifier="Boolean")

# === Nodo de control ===
bpy.ops.mesh.primitive_cube_add(
    size=0.1 * SCALE,
    location=(-0.4 * SCALE, 0, 1.8 * SCALE)
)

# === Materiales básicos (opcional realismo) ===
material_steel = bpy.data.materials.new(name="Steel")
material_steel.diffuse_color = (0.3, 0.3, 0.3, 1)

material_copper = bpy.data.materials.new(name="Copper")
material_copper.diffuse_color = (0.6, 0.3, 0.1, 1)

for obj in bpy.context.scene.objects:
    if "Nozzle" in obj.name:
        obj.data.materials.append(material_steel)
    elif "Solenoid" in obj.name or "Chamber" in obj.name:
        obj.data.materials.append(material_copper)
```

```
$fn = 100;

// Tobera Merlin
difference() {
    cone(h=2.0, r1=0.25, r2=0.9);
    translate([0, 0, -0.1]) cylinder(h=2.1, r=0.2);
}

// Toroide Tokamak
translate([0, 0, 2.0])
    torus(R=0.75, r=0.15);
```

```
// Solenoide central
translate([0, 0, 2.0])
    cylinder(h=1.4, r=0.08);

// Bobinas toroidales
for (a = [0 : 30 : 330]) {
    rotate([0, 0, a])
    translate([0.75, 0, 2.0])
        rotate([90, 0, 0])
            torus(R=0.05, r=0.015);
}

// Inyector
translate([0.9, 0, 2.2]) rotate([0, 90, 0]) cylinder(h=0.4, r=0.02);

// Conductos
for (a = [0 : 90 : 270]) {
    rotate([0, 0, a])
    translate([0.6, 0, 2.3]) cylinder(h=0.5, r=0.015);
}

// Escudo térmico
difference() {
    translate([0, 0, 1.9]) cylinder(h=0.4, r=1.05);
    translate([0, 0, 1.9]) cylinder(h=0.4, r=0.95);
}

// Nodo de control
translate([-0.4, 0, 2.3]) cube([0.1, 0.1, 0.1], center=true);

module torus(R, r) {
    rotate_extrude(angle = 360)
        translate([R, 0, 0])
            circle(r);
}
```

SIM

```
# Instalaremos y crearemos una simulación básica de un modelo 3D tipo
"Tokamak-Merlin" usando Trimesh
# Este script crea las primitivas y las agrupa en una escena para
visualizar el motor híbrido.
import trimesh
import numpy as np

# Crear la tobera tipo Merlin (cono truncado)
```

```
nozzle_height = 2.0
nozzle_top_radius = 0.25
nozzle_bottom_radius = 0.9
nozzle_cone = trimesh.creation.cone(radius=nozzle_bottom_radius,
height=nozzle_height, sections=64)
nozzle_cone.apply_translation([0, 0, nozzle_height / 2])

# Cámara toroidal tipo Tokamak
tokamak_major_radius = 0.75
tokamak_minor_radius = 0.15
tokamak_torus = trimesh.creation.torus(radius=tokamak_major_radius,
tube_radius=tokamak_minor_radius)
tokamak_torus.apply_translation([0, 0, 2.0])

# Solenoide central (cilindro vertical)
solenoid = trimesh.creation.cylinder(radius=0.08, height=1.4,
sections=32)
solenoid.apply_translation([0, 0, 2.0])

# Bobinas toroidales (pequeños toros alrededor)
bobinas = []
for angle_deg in range(0, 360, 30):
    angle_rad = np.radians(angle_deg)
    x = tokamak_major_radius * np.cos(angle_rad)
    y = tokamak_major_radius * np.sin(angle_rad)
    coil = trimesh.creation.torus(radius=0.05, tube_radius=0.015)
    coil.apply_translation([x, y, 2.0])
    coil.apply_rotation(trimesh.transformations.rotation_matrix(np.pi
/ 2, [1, 0, 0]))
    bobinas.append(coil)

# Inyector de deuterio (cilindro lateral)
injector = trimesh.creation.cylinder(radius=0.02, height=0.4,
sections=32)
injector.apply_translation([0.9, 0, 2.2])
injector.apply_transform(trimesh.transformations.rotation_matrix(np.p
i / 2, [0, 1, 0]))

# Conductos de refrigeración
conductos = []
for i in range(4):
    angle_rad = np.radians(i * 90)
    x = 0.6 * np.cos(angle_rad)
    y = 0.6 * np.sin(angle_rad)
    tube = trimesh.creation.cylinder(radius=0.015, height=0.5,
sections=16)
    tube.apply_translation([x, y, 2.3])
    tube.apply_rotation(trimesh.transformations.rotation_matrix(np.pi
/ 2, [1, 0, 0]))
    conductos.append(tube)
```

```
# Crear la escena
scene = trimesh.Scene()
scene.add_geometry(nozzle_cone, node_name="Nozzle")
scene.add_geometry(tokamak_torus, node_name="Tokamak")
scene.add_geometry(solenoid, node_name="Solenoid")
scene.add_geometry(injector, node_name="Injector")
for i, coil in enumerate(bobinas):
    scene.add_geometry(coil, node_name=f"Coil_{i}")
for i, tube in enumerate(conductos):
    scene.add_geometry(tube, node_name=f"Tube_{i}")

scene.show()
```

Trimesh

```
import trimesh
import numpy as np

def create_torus(R=0.75, r=0.15, segments=64):
    theta = np.linspace(0, 2 * np.pi, segments)
    phi = np.linspace(0, 2 * np.pi, segments)
    theta, phi = np.meshgrid(theta, phi)
    x = (R + r * np.cos(theta)) * np.cos(phi)
    y = (R + r * np.cos(theta)) * np.sin(phi)
    z = r * np.sin(theta)
    vertices = np.stack([x, y, z], axis=-1).reshape(-1, 3)
    faces = []
    for i in range(segments):
        for j in range(segments):
            i0 = i * segments + j
            i1 = ((i + 1) % segments) * segments + j
            i2 = ((i + 1) % segments) * segments + (j + 1) % segments
            i3 = i * segments + (j + 1) % segments
            faces.append([i0, i1, i2])
            faces.append([i0, i2, i3])
    return trimesh.Trimesh(vertices=vertices, faces=faces)

scene = trimesh.Scene()

# Tobera Merlin
nozzle = trimesh.creation.cone(radius=0.9, height=2.0, sections=64)
nozzle.apply_translation([0, 0, 1.0])
scene.add_geometry(nozzle, node_name="Nozzle")

# Toroide Tokamak
tokamak_torus = create_torus()
tokamak_torus.apply_translation([0, 0, 3.0])
```

```
scene.add_geometry(tokamak_torus, node_name="Tokamak")

# Solenoide
solenoid = trimesh.creation.cylinder(radius=0.08, height=1.4)
solenoid.apply_translation([0, 0, 3.0])
scene.add_geometry(solenoid, node_name="Solenoid")

# Bobinas toroidales
for angle_deg in range(0, 360, 30):
    angle_rad = np.radians(angle_deg)
    x = 0.75 * np.cos(angle_rad)
    y = 0.75 * np.sin(angle_rad)
    coil = create_torus(R=0.05, r=0.015)
    coil.apply_translation([x, y, 3.0])

coil.apply_transform(trimesh.transformations.rotation_matrix(np.pi /
2, [1, 0, 0]))
    scene.add_geometry(coil, node_name=f"Coil_{angle_deg}")

# Inyector de deuterio
injector = trimesh.creation.cylinder(radius=0.02, height=0.4)
injector.apply_translation([0.9, 0, 3.3])
injector.apply_transform(trimesh.transformations.rotation_matrix(np.p
i / 2, [0, 1, 0]))
scene.add_geometry(injector, node_name="Injector")

# Conductos de refrigeración
for i in range(4):
    angle = np.radians(i * 90)
    x = 0.6 * np.cos(angle)
    y = 0.6 * np.sin(angle)
    tube = trimesh.creation.cylinder(radius=0.015, height=0.5)
    tube.apply_translation([x, y, 3.5])

tube.apply_transform(trimesh.transformations.rotation_matrix(np.pi /
2, [1, 0, 0]))
    scene.add_geometry(tube, node_name=f"Tube_{i}")

scene.show()
```

Geometry/

```
import bpy
import math
from math import pi

# Limpiar escena
```

```
bpy.ops.object.select_all(action='SELECT')
bpy.ops.object.delete(use_global=False)

# Función auxiliar: crear toroide
def add_torus(name, major, minor, location, rotation=(0, 0, 0)):
    bpy.ops.mesh.primitive_torus_add(
        major_radius=major,
        minor_radius=minor,
        location=location,
        rotation=rotation,
        abso_major_rad=1.25,
        abso_minor_rad=0.75)
    obj = bpy.context.object
    obj.name = name

# Tobera Merlin (cono invertido)
bpy.ops.mesh.primitive_cone_add(
    vertices=128,
    radius1=0.85,
    radius2=0.25,
    depth=2.0,
    location=(0, 0, 1.0))
bpy.context.object.name = "Merlin_Nozzle"

# Tokamak toroidal chamber
add_torus("Tokamak_Core", major=0.75, minor=0.15, location=(0, 0, 3.0))

# Solenoide central
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.08,
    depth=1.5,
    location=(0, 0, 3.0))
bpy.context.object.name = "Solenoid_Center"

# Bobinas toroidales (x12)
for angle in range(0, 360, 30):
    rad = math.radians(angle)
    x = 0.75 * math.cos(rad)
    y = 0.75 * math.sin(rad)
    add_torus(
        name=f"Toroidal_Coil_{angle}",
        major=0.05,
        minor=0.015,
        location=(x, y, 3.0),
        rotation=(pi / 2, 0, 0))

# Inyector de deuterio
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.02,
    depth=0.4,
    location=(0.9, 0, 3.3),
```

```
    rotation=(0, pi/2, 0))
bpy.context.object.name = "Deuterium_Injector"

# Conductos de refrigeración (4 simétricos)
for i in range(4):
    angle = math.radians(i * 90)
    x = 0.6 * math.cos(angle)
    y = 0.6 * math.sin(angle)
    bpy.ops.mesh.primitive_cylinder_add(
        radius=0.015,
        depth=0.5,
        location=(x, y, 3.5),
        rotation=(pi/2, 0, 0))
    bpy.context.object.name = f"Coolant_Tube_{i}"

# Escudo térmico exterior
bpy.ops.mesh.primitive_cylinder_add(
    radius=0.9,
    depth=0.5,
    location=(0, 0, 3.0))
outer = bpy.context.object
outer.name = "Thermal_Shield"

bpy.ops.mesh.primitive_cylinder_add(
    radius=0.75,
    depth=0.5,
    location=(0, 0, 3.0))
inner = bpy.context.object
inner.name = "Thermal_Shield_Inner"

# Realizar diferencia booleana (hueco interno)
mod_bool = outer.modifiers.new(name="cut", type='BOOLEAN')
mod_bool.operation = 'DIFFERENCE'
mod_bool.object = inner
bpy.context.view_layer.objects.active = outer
bpy.ops.object.modifier_apply(modifier="cut")
bpy.data.objects.remove(inner)

# Nodo de control electrónico
bpy.ops.mesh.primitive_cube_add(
    size=0.1,
    location=(-0.6, 0, 3.6))
bpy.context.object.name = "Control_Node"
```

1. Tobera Merlin invertida de titanio
2. Cámara toroidal Tokamak
3. Solenoide vertical
4. Bobinas toroidales rotadas
5. Inyector de deuterio
6. Conductos de refrigeración simétricos
7. Escudo térmico metálico con hueco
8. Caja de control electrónica

cámara de hidrógeno/deuterio

Almacenamiento y regulación del combustible

- El deuterio no se inyecta directamente desde el exterior. Se necesita un sistema de almacenamiento presurizado que contenga gas deuterio o pellets sólidos.
- Este sistema debe permitir una inyección controlada y precisa para mantener la densidad del plasma dentro de los márgenes operativos (~10^{19} partículas/m^3 en muchos reactores).

Precalentamiento y acondicionamiento

- En algunos diseños, el gas pasa por etapas de acondicionamiento térmico o ionización parcial antes de entrar al tokamak.
- Esto se hace para mejorar la eficiencia de la ionización dentro del toroide.

Sincronización con campos magnéticos

- El sistema de inyección (pellet o gas puffing) debe estar coordinado con el sistema magnético para evitar que el combustible desestabilice el plasma.

Elementos de diseño de la cámara de hidrógeno

los elementos que podrías modelar en OpenSCAD o Blender:

Tanque presurizado (fuel storage)

- Cilindro metálico externo (acero inoxidable o titanio) con válvulas y reguladores.
- Puede tener aislamiento térmico si almacena deuterio líquido o sólido.

Válvulas de inyección

- Electroválvulas de alta precisión montadas en línea con los inyectores.
- Están sincronizadas con el sistema de control y el estado del plasma.

Conductos de distribución

- Tubos metálicos o cerámicos que llevan el gas desde el tanque hasta el inyector.
- Pueden incluir sensores de presión, temperatura y caudal.

Inyector o lanzador de pellets

- Alternativamente, puede diseñarse un sistema de pellet injection, donde se disparan pequeños pellets sólidos de deuterio hacia el núcleo.
- Ideal para reinyectar combustible sin perturbar la estabilidad del plasma.

Inverted Titanium Merlin Nozzle (Combustion Nozzle / Exhaust Bell)

The Merlin nozzle is typically a bell-shaped expansion chamber designed to convert thermal and pressure energy from combustion into directed thrust. In this hybrid design:

- Material: Titanium alloy is used for high-temperature resistance and weight efficiency.
- Function: Instead of chemical combustion, it exhausts heated plasma confined in the tokamak, creating thrust through magnetic-expelled plasma streams.

- Design Parameters: Expansion ratio, throat diameter, and diverging cone angle must be tuned to the specific energy profile of plasma exhaust.

Toroidal Tokamak Plasma Chamber

The tokamak is a torus-shaped chamber used to confine plasma via powerful magnetic fields:

- Vacuum Vessel: The core chamber where low-pressure deuterium gas is ionized into plasma.
- Magnetic Field: Created by external toroidal and poloidal coils to confine the plasma.
- Fusion Goal: Deuterium nuclei collide and fuse under thermal pressure, releasing high-energy neutrons.

Theory: The tokamak relies on the magnetic pinch effect and toroidal confinement to contain plasma, governed by the Grad–Shafranov equation and MHD stability criteria.

Vertical Solenoid (Central Solenoid)

The central solenoid is a key component in plasma ignition and confinement:

- Role: Induces a toroidal current within the plasma, enabling magnetic confinement (acts like a transformer).
- Design: Composed of stacked superconducting coils for high magnetic field strength (~5–10 Tesla).
- Operation: Pulsed induction aligns plasma particles and initiates the fusion process.

Rotated Toroidal Field Coils

These coils generate the toroidal component of the magnetic field:

- Positioning: Equally spaced around the torus for symmetric field distribution.
- Shape: Often D-shaped or circular to follow the vessel profile.
- Material: Typically copper or niobium-tin (Nb_3Sn) for superconductivity.

They stabilize the plasma, prevent drift, and sustain long confinement times for sustained fusion.

Deuterium Injector

Deuterium is the fusion fuel used in this system:

- Injection: Injected as gas or pellets into the chamber at timed intervals.
- Ionization: Exposed to radiofrequency (RF) fields or neutral beam injectors for ionization.
- Mass Flow Control: Regulated to maintain optimal density and fuel rate.

Fusion reactions:
$D + D \rightarrow He^3 + n + 3.3\ MeV$
$D + T \rightarrow He^4 + n + 17.6\ MeV$ (if Tritium is later included)

Symmetrical Cooling Ducts (Thermal Channels)

Due to extreme heat from plasma and magnetic coils, thermal management is essential:

- Structure: Liquid-cooled copper or Inconel piping embedded around the vessel and nozzle walls.
- Coolants: Can use liquid hydrogen, helium or water depending on pressure.
- Purpose: Prevent thermal fatigue, structural degradation, and enable sustained operation.

Thermal Shield with Hollow Interior

The thermal shield protects structural elements and electronics from radiation and heat:

- Outer Shell: Made of high-thermal-capacity metals like tungsten, carbon composites or ceramic coating.
- Inner Hollow: Allows for insulation, reflective layers, and mechanical expansion.
- Design: Often uses a sandwich structure with multiple conductive/insulating layers.

Electronic Control Box

This module houses avionics and feedback systems:

- Includes: Real-time magnetic field regulators, thermal sensors, current monitors, PID controllers.
- Architecture: Often FPGA-based or embedded RTOS with radiation-hardened components.
- Interfaces: Communicates with flight control systems and external telemetry.

Supporting Documentation and Technical Resources

Fusion & Tokamak Design

- ITER Basics (https://www.iter.org/mach)
- MIT PSFC Tokamak Overview (https://www.psfc.mit.edu/)
- "Introduction to Plasma Physics and Controlled Fusion" – Francis F. Chen (PDFs available online)