ASTRODYN-CORE

Our Orbital Propagation Framework

Jose Manuel

February 2026

What is ASTRODYN-CORE?

- **Builder-first** astrodynamics tooling
- Keeps **Orekit APIs first-class** while adding:
    - Typed configuration
    - State-file workflows
    - Mission-profile helpers
- Unified client APIs for propagation, state, mission, uncertainty, TLE, ephemeris
- **Extensible registry** for custom propagators

Design Principle

Orekit-native semantics stay visible: providers return real Orekit builders/propagators.

Architecture Overview

Two API Tiers:

1. **Stable facade tier** (recommended)
   - Start with `AstrodynClient`
   - Use domain facades
2. **Advanced low-level tier**
   - `PropagatorFactory`
   - `ProviderRegistry`
   - Fine-grained Orekit control

Implemented Features

- Numerical, Keplerian, DSST, TLE propagators
- Custom propagator registry
  - GEqOE J2 Taylor-series propagator
- Force model, attitude, spacecraft assembly
- State I/O: YAML/JSON/HDF5
- STM-based covariance propagation
- Scenario maneuver tooling

Getting Started in 10 Lines

```python
from astrodyn_core import (
    AstrodynClient, BuildContext,
    IntegratorSpec, PropagatorKind, PropagatorSpec,
)

app = AstrodynClient()

spec = PropagatorSpec(
    kind=PropagatorKind.NUMERICAL,
    mass_kg=1200.0,
    integrator=IntegratorSpec(
        kind="dp853",
        position_tolerance=10.0,
    ),
)

ctx = BuildContext(initial_orbit=initial_orbit)
builder = app.propagation.build_builder(spec, ctx)
propagator = builder.buildPropagator(builder.getSelectedNormalizedParameters())
```

Propagator Kinds Available

Built-in Providers:

- `KEPLERIAN` - Analytical Keplerian
- `NUMERICAL` - Full numerical integration
- `DSST` - Semi-analytical propagator
- `TLE` - Two-line element
- `geqoe` - Custom GEqOE J2 Taylor-series

Extensible:

- Any string `kind` for custom propagators
- Registry-based plugin system

Example: Switching Propagators

```python
# Just change the kind!
spec_kepler = PropagatorSpec(
    kind=PropagatorKind.KEPLERIAN,
    mass_kg=450.0
)

spec_dsst = PropagatorSpec(
    kind=PropagatorKind.DSST,
    mass_kg=550.0,
    integrator=IntegratorSpec(kind="dp853"),
    force_spec=GravitySpec(degree=8, order=8),
)

spec_geqoe = PropagatorSpec(
    kind="geqoe",
    mass_kg=450.0,
)
```

Numerical Propagation - With Forces

```python
from astrodyn_core import (
    AstrodynClient, BuildContext, SpacecraftSpec,
    get_propagation_model, load_dynamics_config,
)

app = AstrodynClient()
spec = load_dynamics_config(get_propagation_model("medium_fidelity"))
spec = spec.with_spacecraft(
    SpacecraftSpec(mass=600.0, drag_area=6.0, srp_area=6.0)
)

builder = app.propagation.build_builder(spec, BuildContext(initial_orbit=orbit))
propagator = builder.buildPropagator(builder.getSelectedNormalizedParameters())

# Propagate 90 minutes
state = propagator.propagate(epoch.shiftedBy(5400.0))
pos = state.getPVCoordinates(frame).getPosition()
forces = [f.getClass().getSimpleName() for f in propagator.getAllForceModels()]
```

Numerical Propagation - Results

Active Force Models

`[NewtonianAttraction, HolmesFeatherstoneAttractionModel, DragForce, SolarRadiationPressure]`

State after 90 min

```
Position (m):   [1 234 567.1,  -6 543 210.4,  892 341.7]
Velocity (m/s):   [7 234.5,  1 123.4,  -2 345.6]
```

Key Point

Load a named dynamics preset (`medium_fidelity`, `high_fidelity`, ...) and override just the spacecraft properties — no manual force model wiring needed.

TLE Propagation

```python
from astrodyn_core import TLESpec, PropagatorKind, PropagatorSpec

# Use TLE data (e.g., ISS)
tle = TLESpec(
    line1="1 25544U 98067A   24001.50000000  .00016717  00000-0  10270-3 0  9002",
    line2="2 25544  51.6400  10.0000 0006000  50.0000 310.0000 15.49000000000000",
)

app = AstrodynClient()
propagator = app.propagation.build_propagator(
    PropagatorSpec(kind=PropagatorKind.TLE, tle=tle),
    BuildContext(),
)

state = propagator.propagate(tle_epoch.shiftedBy(45.0 * 60.0))
pos = state.getPVCoordinates(FramesFactory.getGCRF()).getPosition()
```

TLE Resolution via SpaceTrack

Live TLE download:

- Resolve any NORAD ID + epoch
- Local disk cache (no re-downloads)
- Closest TLE to target epoch

Credentials: stored in `secrets.ini`

```python
from spacetrack import SpaceTrackClient
from datetime import datetime, timezone

st_client = SpaceTrackClient(
    identity=identity,
    password=password,
)
app = AstrodynClient(
    tle_base_dir=tle_cache_dir,
    tle_allow_download=True,
    space_track_client=st_client,
)
query = app.tle.build_query(
    norad_id=25544,  # ISS
    target_epoch=datetime.now(timezone.utc),
)
tle_spec = app.tle.resolve_tle_spec(query)
propagator = app.propagation.build_propagator(
    PropagatorSpec(kind=PropagatorKind.TLE,
                   tle=tle_spec),
    BuildContext()
```

GEqOE: J2 Taylor-Series Propagator

Highlights:

- Fast analytical propagator
- Includes J2 perturbation
- Taylor series expansion
- Configurable order (1-4)
- Built-in STM computation

Three Usage Levels:

1. Provider pipeline (AstrodynClient)
2. Direct Orekit adapter
3. Pure NumPy engine

Via AstrodynClient

```
spec = PropagatorSpec(
    kind="geqoe",
    mass_kg=450.0,
    orekit_options={"taylor_order": 4},
)
propagator =
↪   app.propagation.build_propagator(spec, ctx)
```

GEqOE Direct Usage

```python
from astrodyn_core.propagation.providers.geqoe import GEqOEPropagator

# Create with initial orbit
prop = GEqOEPropagator(
    initial_orbit=orbit,
    order=4,
    mass_kg=450.0,
)

# Single epoch propagation
state = prop.propagate(epoch.shiftedBy(600.0))

# Get native state (Cartesian + STM as numpy)
y, stm = prop.get_native_state(target)
print(f"State: {y}")
print(f"STM diagonal: {np.diag(stm)}")

# Batch propagation (no Orekit overhead)
dt_grid = np.linspace(0, 3600, 13)
y_out, stm_out = prop.propagate_array(dt_grid)
```

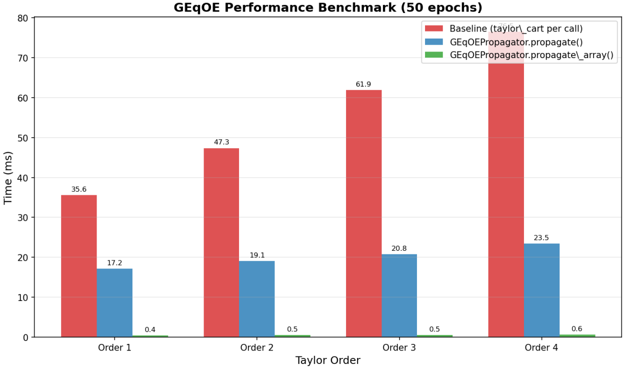
GEqOE Pure NumPy Engine

```
from astrodyn_core.propagation.geqoe.core import taylor_cart_propagator
from astrodyn_core.propagation.geqoe.conversion import BodyConstants
import numpy as np

# No Orekit needed!
y0 = np.array([6_878_137.0, 0.0, 0.0, 0.0, 7200.0, 2400.0])
body = BodyConstants(mu=3.986004418e14, j2=1.08262668e-3, re=6_378_137.0)

tspan = np.arange(0, 3601, 60)
y_out, stm = taylor_cart_propagator(tspan=tspan, y0=y0, p=body, order=4)

print(f"Trajectory shape: {y_out.shape}")  # (61, 6)
print(f"STM shape: {stm.shape}")            # (6, 6, 61)
```

Speed!

GEqOEPropagator is 10–100x faster than numerical integration for short arcs!

GEqOE Performance Benchmark



Key Results

10-50x speedup vs baseline loop — Batch propagation is fastest for multiple epochs

State Transition Matrix (STM) Propagation

Why STM?

- Fundamental for covariance propagation
- Enables sensitivity analysis

Features:

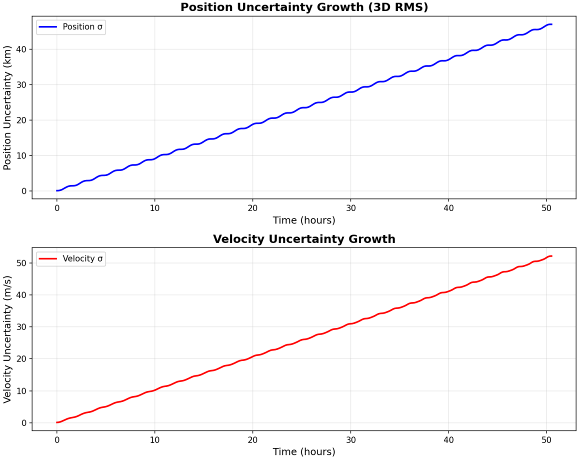
- Cartesian and Keplerian representations (using Orekit transformations)
- Automatic STM computation

```python
from astrodyn_core.uncertainty import (
    setup_stm_propagator,
    propagate_with_stm
)

stm_prop = setup_stm_propagator(propagator)

# Get state + STM at any epoch
state, phi =
↪   stm_prop.propagate_with_stm(target_epoch)

# Verify symplecticity
det_phi = np.linalg.det(phi)
print(f"|det(Phi) - 1| = {abs(det_phi -
↪   1):.3e}")
```

Covariance Propagation Example

```python
import numpy as np
from astrodyn_core import AstrodynClient, UncertaintySpec

app = AstrodynClient()

# Initial 1-sigma: 100 m position, 0.1 m/s velocity
P0 = np.diag([1e4, 1e4, 1e4, 1e-2, 1e-2, 1e-2])

# Propagate with STM
spec = UncertaintySpec(method="stm", orbit_type="CARTESIAN")
state_series, cov_series = app.uncertainty.propagate_with_covariance(
    propagator, P0, epoch_spec, spec=spec
)

# Examine growth
for rec in cov_series.records[::24]:  # every 12 hours
    cov = rec.to_numpy()
    sig_pos = np.sqrt(np.trace(cov[:3,:3]) / 3.0)
    print(f"sig_pos = {sig_pos:.1f} m")
```

Covariance Propagation Results



State File Workflows

Supported Formats:

- YAML (human-readable)
- JSON (programmatic)
- HDF5 (large datasets)
- OEM/OCM/SP3/CPF

State Types:

- Single state (epoch + orbit)
- State series (trajectory)
- Mission timeline (maneuvers)

Example: Save & Convert

```
app.state.export_trajectory_from_propagator(
    propagator,
    epoch_spec,
    "trajectory.yaml",
    series_name="leo-orbit",
    representation="keplerian",
    frame="GCRF",
)

# Load back
series = app.state.load_state_series("trajectory
↪   .yaml")

# Orekit bounded propagator
ephemeris =
↪   app.state.state_series_to_ephemeris(series)
```

Roadmap: Derivatives wrt Parameters

Satellite Parameters

- Mass
- Cross-sectional area (drag/SRP)
- Reflectivity coefficient
- Maneuver impulses

Station Parameters

- Range bias
- Range rate bias
- Tropospheric delay
- Ionospheric delay

Goal: $\frac{\partial r}{\partial p}$ for any parameter vector p

Roadmap: Field-Based Propagators

Automatic Higher-Order Derivatives:

- Use Orekit's `Field` propagators
- Get order-n derivatives with respect to *any* parameter
- Same API, just specify order

Use Cases:

- Taylor series of the trajectory
- Uncertainty propagation (higher order)
- Sensitivity analysis
- Optimization gradients

```
# Example (future API)
from astrodyn_core import
↪  FieldPropagatorSpec

spec = FieldPropagatorSpec(
    order=3,  # 3rd order
    base=PropagatorSpec(kind=.
    ↪  ..)
)

field_propagator =
↪  app.propagation.build_fiel
↪  d_propagator(spec)
# Returns FieldSpacecraftState
# with d^n r / dp^n for any p
```

Other Planned Features

Propagation

- Unscented Transform covariance
- Multi-arc propagation

Infrastructure

- More cookbook examples
- Documentation website

Summary

ASTRODYN-CORE provides:

- Unified, typed API for orbital propagation
- Multiple propagator types (Keplerian, Numerical, DSST, TLE, GEqOE)
- Easy extensibility for custom propagators
- STM/covariance propagation for uncertainty
- Clean state file workflows

Key Benefits

- **Productive:** Quick start in 10 lines
- **Flexible:** Swap propagators by changing kind
- **Configurable:** Use files to configure forces, maneuvers, spacecraft, etc.
- **Extensible:** Registry for custom providers
- **Proven:** Based on Orekit

Getting Started

Run the examples!

```
python examples/quickstart.py --mode all
python examples/geqoe_propagator.py --mode all
python examples/uncertainty.py
python examples/cookbook/multi_fidelity_comparison.py
```

Questions? **Contributions welcome!**