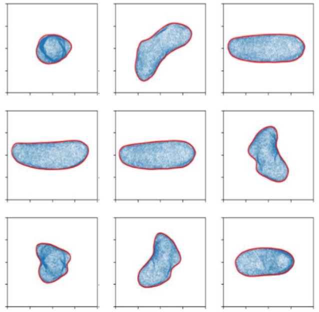
Automatic Differentiation Beyond High-Energy Physics

Differentiable Analysis Blueprint

Giacomo Acciarini
European Space Agency's Advanced Concepts Team
6 March 2026

Differentiable Intelligence for Space Flight Dynamcs

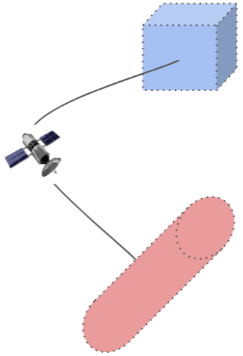
Thermospheric density modeling



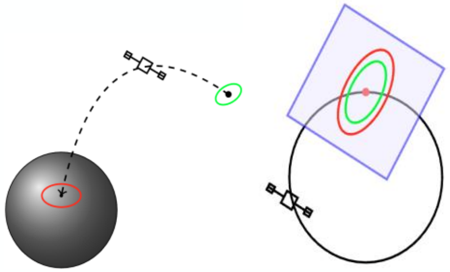
Irregular Small Body Silhouettes

Differentiable Orbit Propagation

Gradient based



Moment Generating Functions



Event Transition Tensors

Higher order

Differentiable Programming

Methods leveraging the end-to-end differentiability of computer programs

Enabled by:

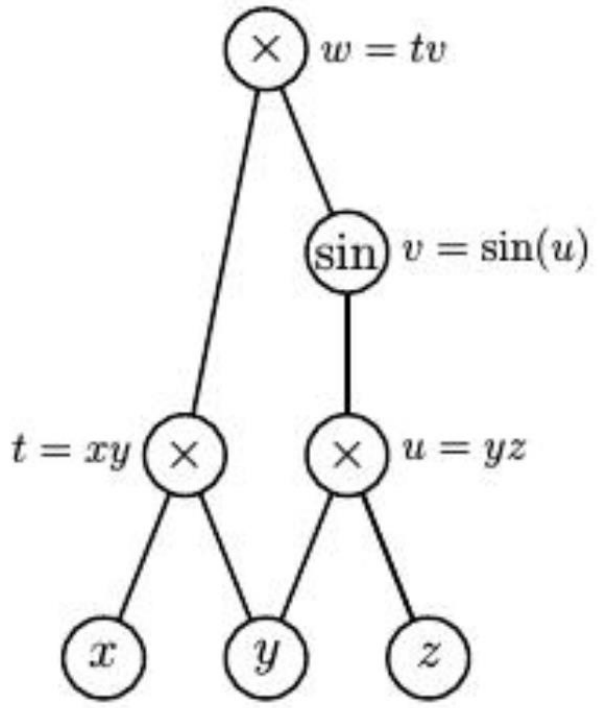
- Advances in automated differentiation (forward, backward, Taylor mode).

Built upon familiar techniques:

- Taylor methods.
- Feed forward neural networks.
- Variational & adjoint equations.
- Algebra of truncated Taylor polynomials.
- Optimization (e.g. stochastic gradient descent, etc..).
- Uncertainty propagation.

And leveraging less familiar ones:

- Efficient and seamless computer implementations.
- Periodic activation functions (SIREN networks).
- Implicit neural representations (NERFs, PINNs).
- MGFs (moment Generating Functions) for non gaussian uncertainties.
- Polynomial map inversion techniques.



Differentiable Programming Tools

PyTorch

TensorFlow

pyaudi

torchquad

Heyoka

∂SGP4

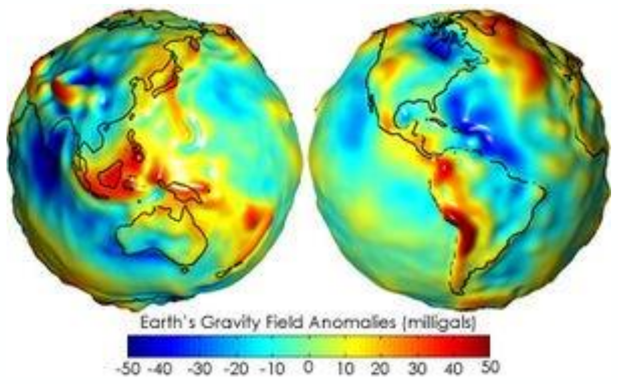
Gradient Based

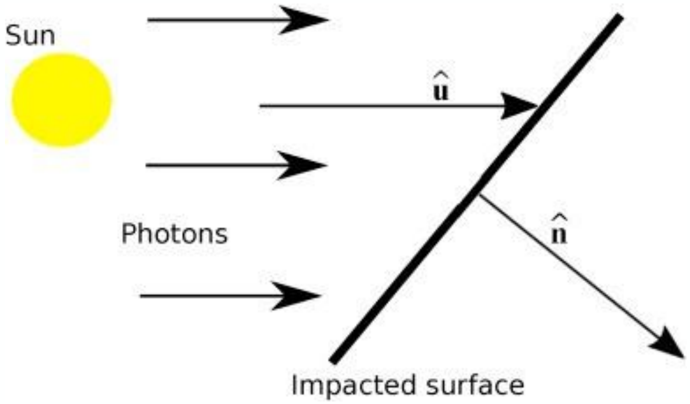
Acciarini, G., Baydin, A. G., Izzo, D.: **Closing the Gap Between SGP4 and High-Precision Propagation via Differentiable Programming**. To appear in: *Acta Astronautica,* 2025. https://doi.org/10.1016/j.actaastro.2024.10.063

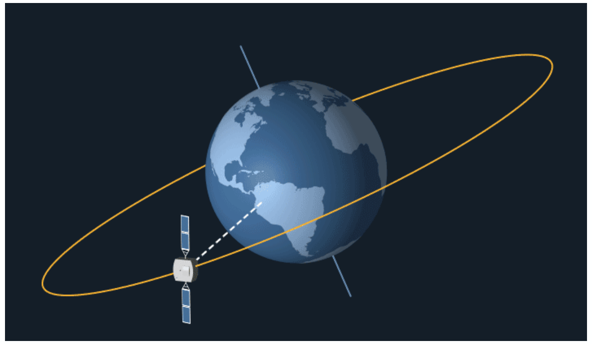
Acciarini, G., Izzo, D., and Biscani, F.: **EclipseNETs: Learning irregular small celestial body silhouettes**. *Acta Astronautica,* 2025. https://doi.org/10.1016/j.actaastro.2025.06.002

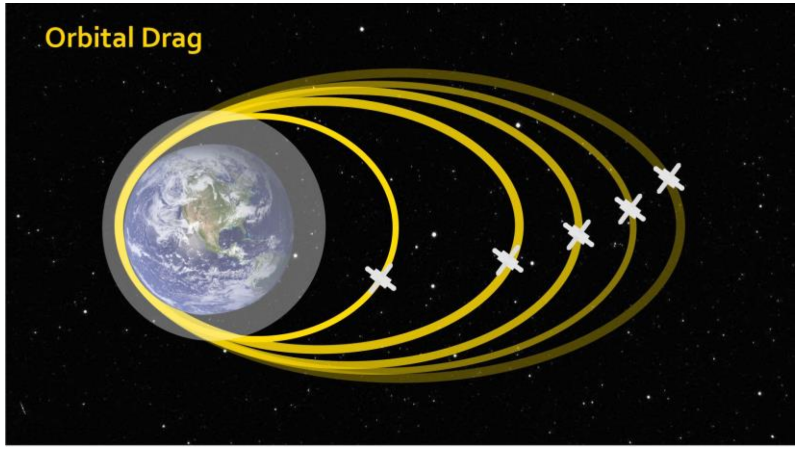
Izzo, D., Acciarini, G. and Biscani, F.: **NeuralODEs for VLEO simulations: Introducing thermoNETs for Thermosphere Modeling**. Submitted to the 9th International Symposium on Space Flight Dynamics (ISSFD), April 2024

Propagating Satellite Orbits







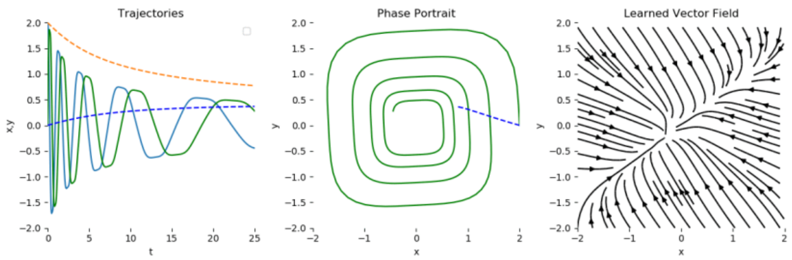


Dynamical Systems and Differentiable Programming

- The **2018 NeurIPS paper** introduced the concept of **NeuralODEs** → track gradients through **ODE+NN** on the right hand-side

- These are ubiquitous in space:
  - **Unmodeled** forces (e.g. thermospheric drag)
  - **Events** (e.g. eclipses)
  - **Controllers** (guidance and control networks)

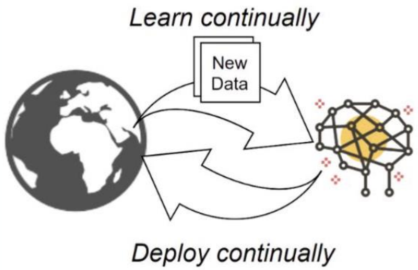
- The overall idea is that after having done expensive training on ground, NeuralODEs can be used to perform onboard corrections
  - **Continuous learning**
  - **Onboard** refinements
  - **Robustness**



Chen, R. T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems, 31.*

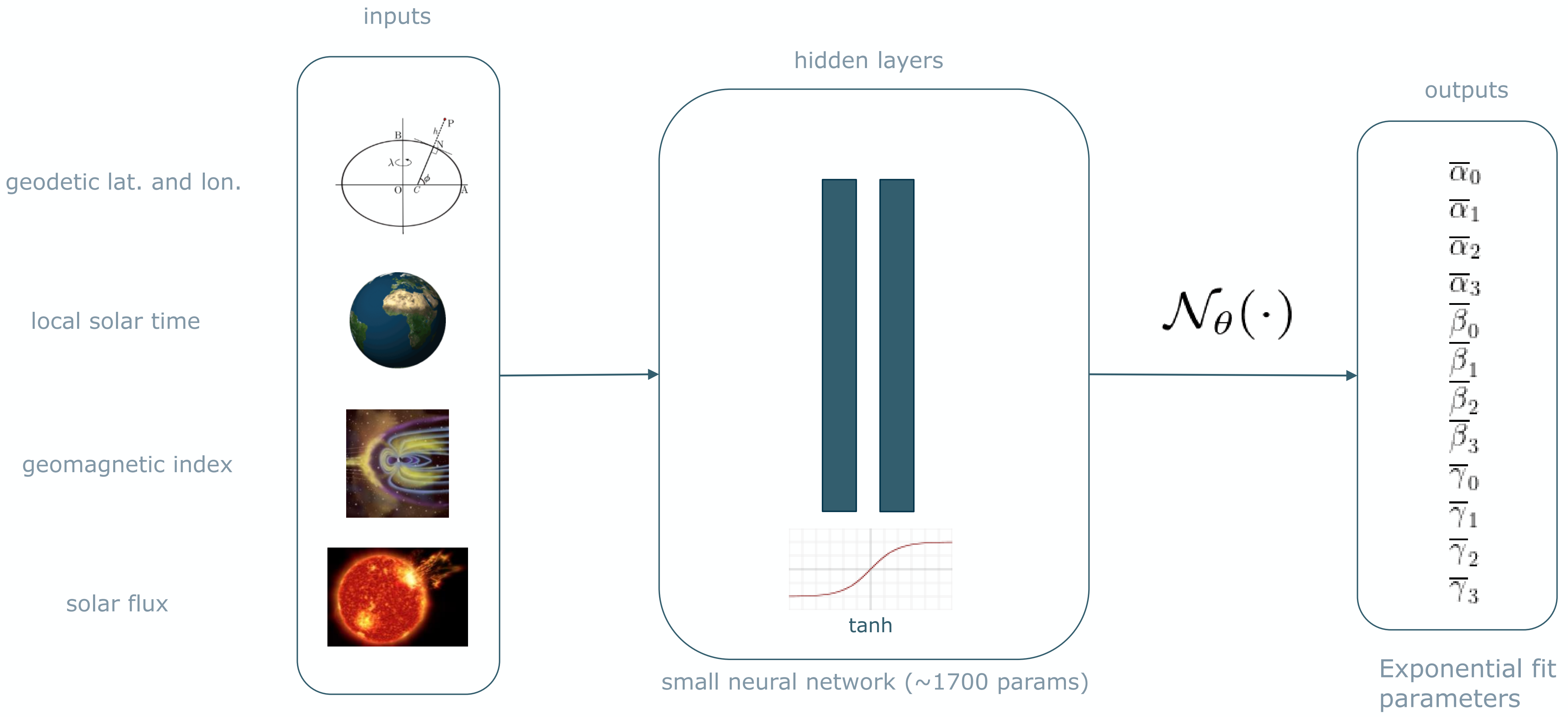
$$H = H_{CR3BP} + \mathcal{N}_{\boldsymbol{\theta}}(x, y, z)$$

$$\frac{d\boldsymbol{\varphi}}{dt} = \nabla_{\mathbf{x}} \mathcal{N}_{\theta}(\mathbf{x}) \cdot \boldsymbol{\varphi} + \frac{\partial \mathcal{N}_{\theta}(\mathbf{x})}{\partial \boldsymbol{\theta}}$$

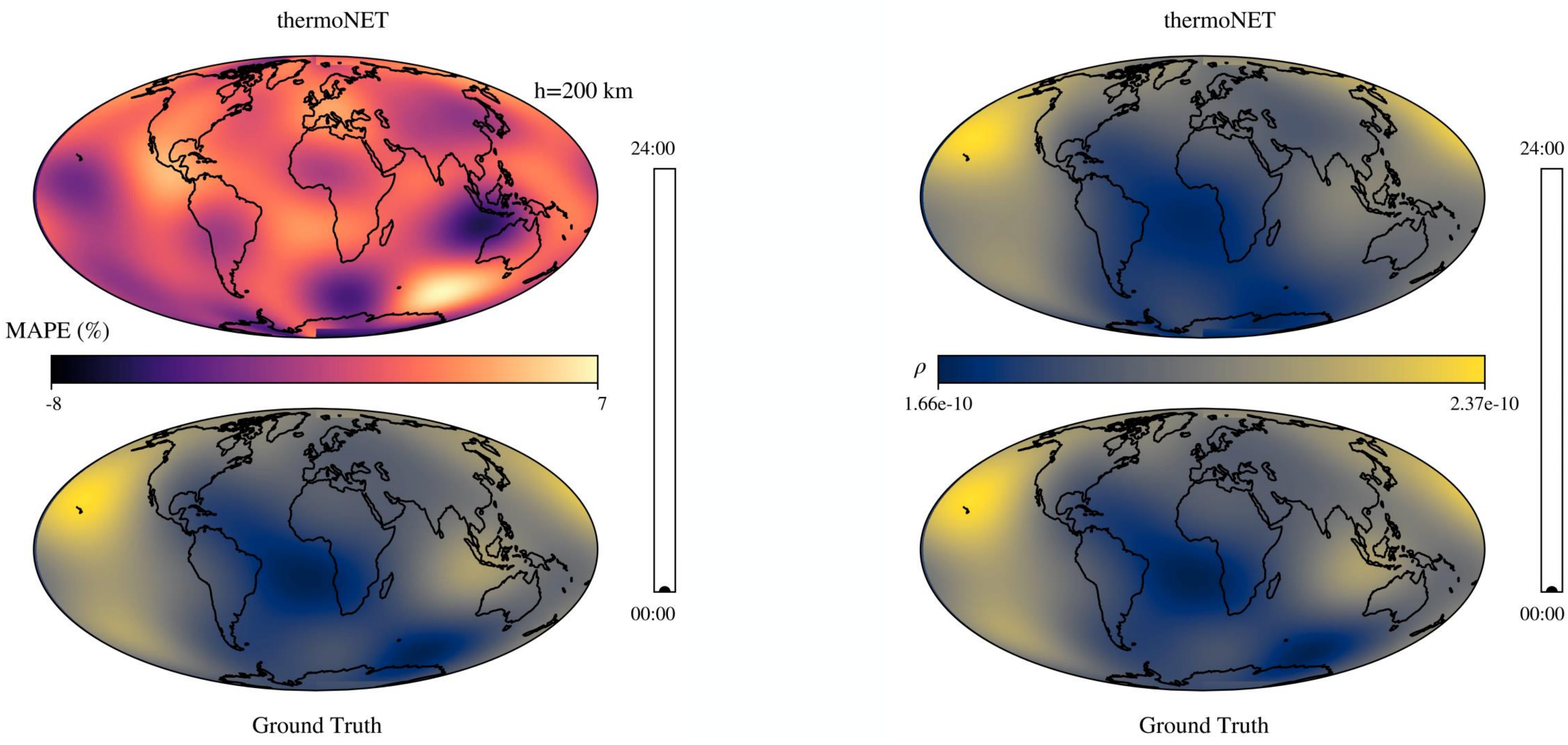


Atmospheric drag

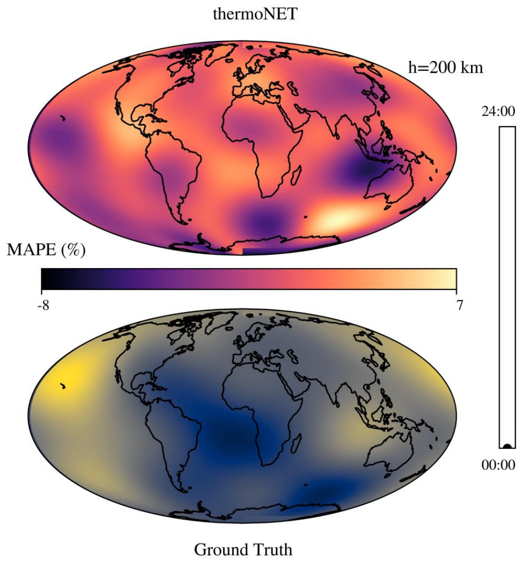
A Differentiable Atmosphere

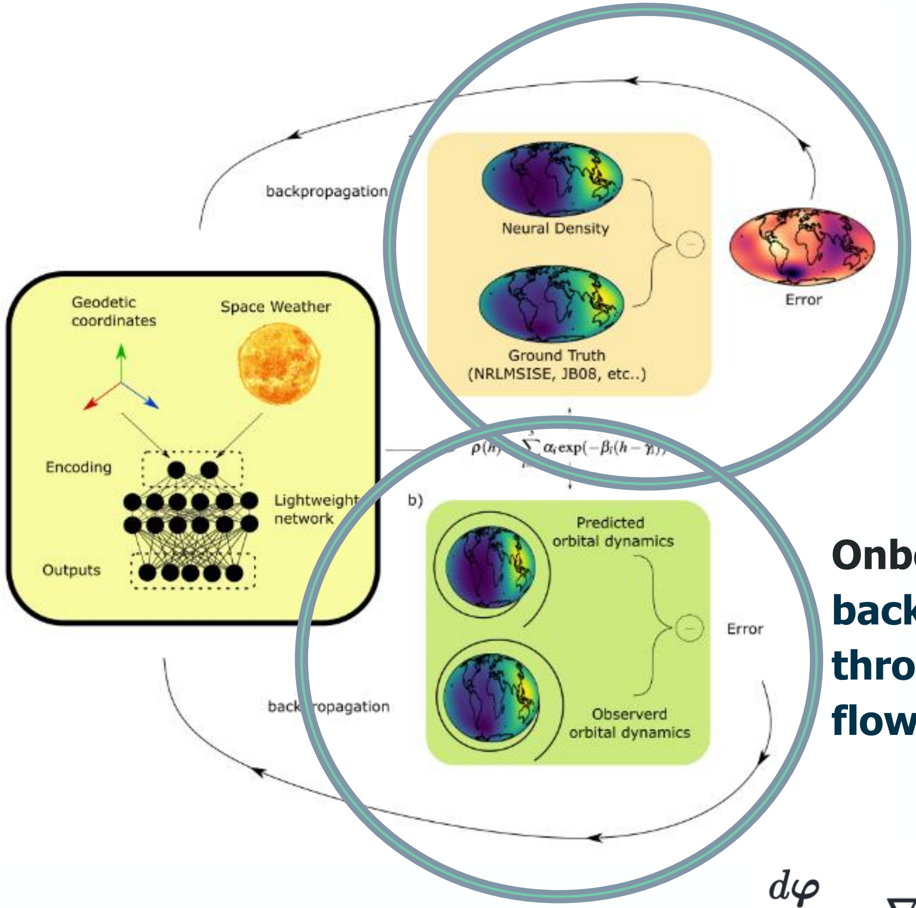


Trained Models: Density Predictions



Continuous Learning via NeuralODE



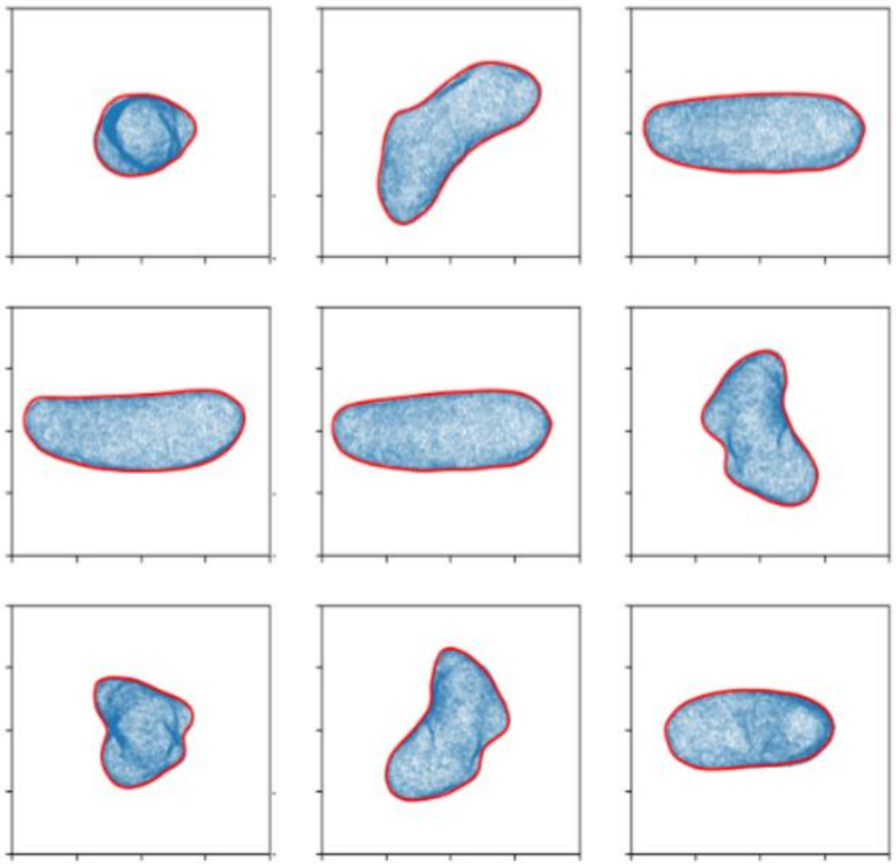


Onboard: backpropagating through the ODE flow

$$\frac{d\boldsymbol{\varphi}}{dt} = \nabla_{\mathbf{x}}\mathcal{N}_\theta(\mathbf{x}) \cdot \boldsymbol{\varphi} + \frac{\partial \mathcal{N}_\theta(\mathbf{x})}{\partial \boldsymbol{\theta}}$$

Solar radiation forces: eclipse modelling

Modelling Small Body Irregular Silhouettes

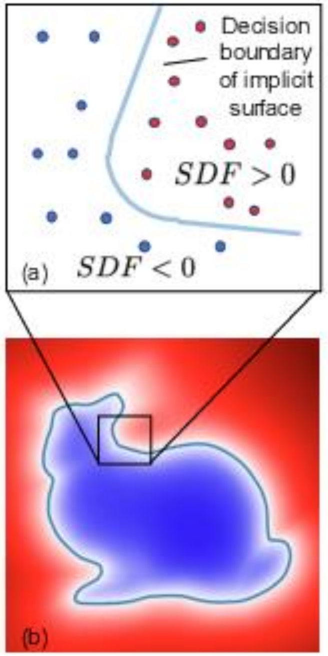


Acciarini, G., Izzo, D., and Biscani, F.: **EclipseNETs: Learning irregular small celestial body silhouettes**. *Acta Astronautica,* 2025. https://doi.org/10.1016/j.actaastro.2025.06.002

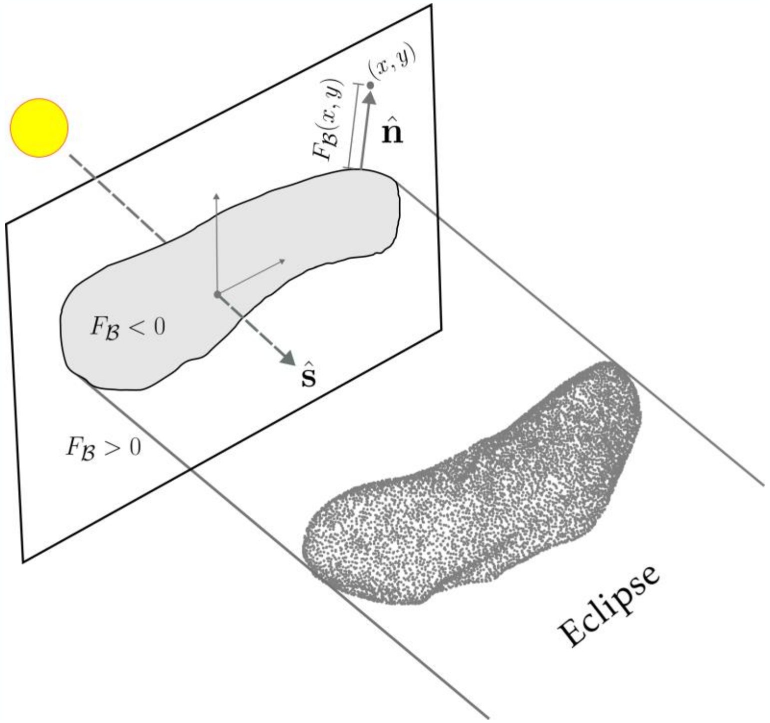
Biscani, F. and Izzo, D., 2022. **Reliable event detection for Taylor methods in astrodynamics**. *Monthly Notices of the Royal Astronomical Society*, *513*(4), pp.4833-4844.

eclipseNETs

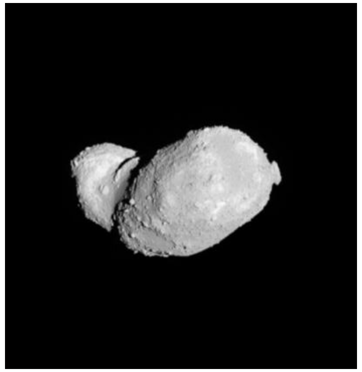
Learning the distance from the **shadow cone** of an irregular body --> **signed distance function (SDF)** can be used:

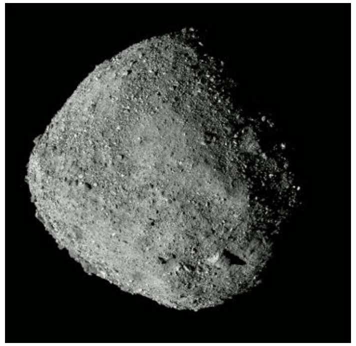


$$F_{\mathcal{B}}(x, y, \hat{\mathbf{s}}) = \begin{cases} d(x, y, \partial\Omega_{\hat{\mathbf{s}}}) \\ -d(x, y, \partial\Omega_{\hat{\mathbf{s}}}) \end{cases}$$

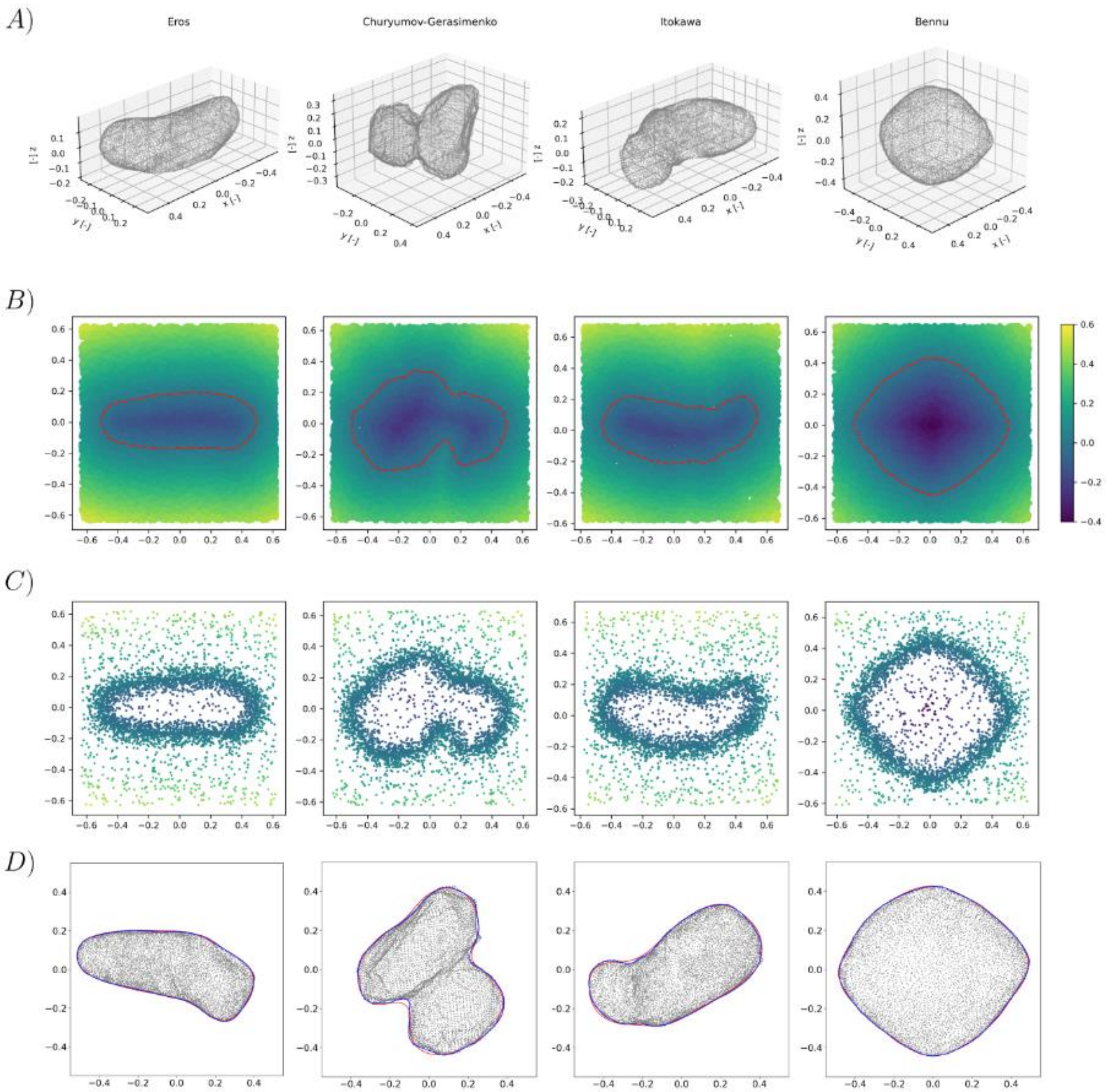


Training Neural Models for Irregular Bodies





	$\mathcal{L}_{MSE}$ ReLU	$\mathcal{L}_{MSE}$ Siren	Dataset
$\mathcal{N}_{Bennu}$	3.5295×10^{-5}	1.99059×10^{-5}	$\mathcal{D}_{train,Bennu}$
$\mathcal{N}_{Itokawa}$	8.0184×10^{-5}	4.7082×10^{-5}	$\mathcal{D}_{train,Itokawa}$
$\mathcal{N}_{67/P}$	1.49711×10^{-4}	7.3122×10^{-5}	$\mathcal{D}_{train,67/P}$
$\mathcal{N}_{Eros}$	9.3738×10^{-5}	4.2137×10^{-5}	$\mathcal{D}_{train,Eros}$
$\mathcal{N}_{Bennu}$	3.6508×10^{-5}	2.1773×10^{-5}	$\mathcal{D}_{valid,Bennu}$
$\mathcal{N}_{Itokawa}$	8.4136×10^{-5}	4.9863×10^{-5}	$\mathcal{D}_{valid,Itokawa}$
$\mathcal{N}_{67/P}$	1.59457×10^{-4}	7.75078×10^{-5}	$\mathcal{D}_{valid,67/P}$
$\mathcal{N}_{Eros}$	9.8826×10^{-5}	4.5850×10^{-5}	$\mathcal{D}_{valid,Eros}$



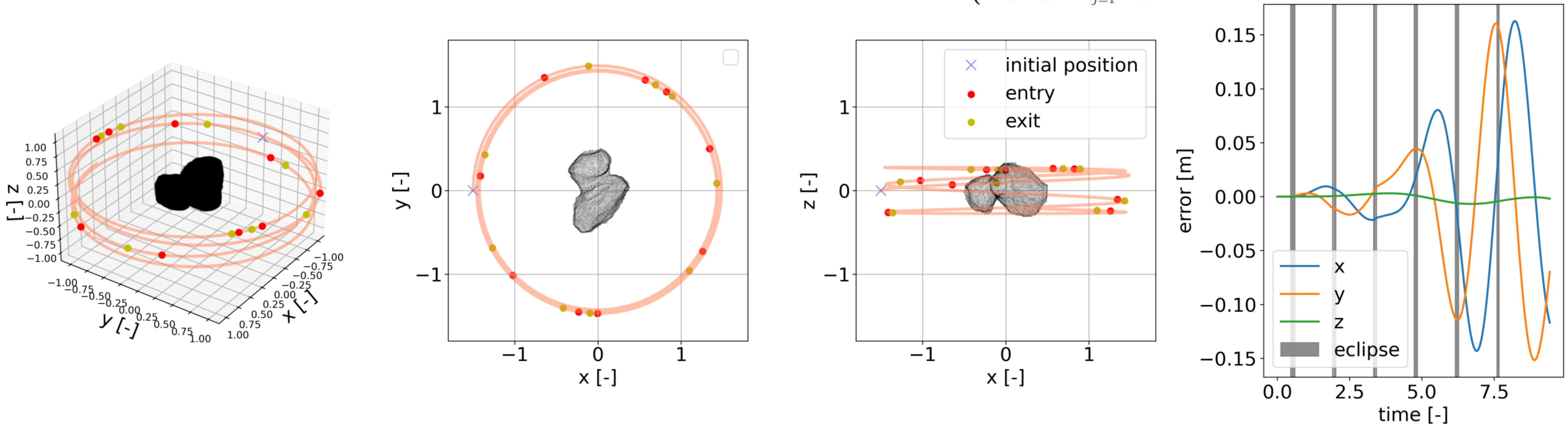
Differentiable Event Detection

- Replacing complex **ray tracing** algorithms with **neural networks**
- This produces a **differentiable** representation of **eclipses**
- Allowing for their **reliable** detection in a Taylor propagator
- Online refinements are then possible via **Neural Event ODEs**

$$\ddot{\mathbf{r}} = -G\sum_{j=0}^{N}\frac{m_j}{|\mathbf{r}-\mathbf{r}_j|^3}(\mathbf{r}-\mathbf{r}_j) - 2\boldsymbol{\omega}\times\mathbf{v} - \boldsymbol{\omega}\times\boldsymbol{\omega}\times\mathbf{r} - \eta\hat{\mathbf{s}}(t)$$

$$\begin{cases} \dot{\boldsymbol{x}} & = \boldsymbol{f}_{\boldsymbol{\theta}}(t,\boldsymbol{x}) \\ \dot{e}_{\boldsymbol{\theta}} & = g_{\boldsymbol{\theta}}(t,\boldsymbol{x}) \\ \frac{d}{dt}\left(\frac{\partial x_i}{\partial \theta_k}\right) & = \sum_{j=1}^{n}\frac{\partial f_{i,\boldsymbol{\theta}}}{\partial x_j}\frac{\partial x_j}{\partial \theta_k} + \frac{\partial f_{i,\boldsymbol{\theta}}}{\partial \theta_k} \\ \frac{d}{dt}\left(\frac{\partial e_{\boldsymbol{\theta}}}{\partial \theta_k}\right) & = \sum_{j=1}^{n}\frac{\partial g_{\boldsymbol{\theta}}}{\partial x_j}\frac{\partial x_j}{\partial \theta_k} + \frac{\partial g_{\boldsymbol{\theta}}}{\partial \theta_k}, \end{cases}$$

$$g_{\boldsymbol{\theta}} = \nabla_{\hat{\mathbf{s}}} e_{\boldsymbol{\theta}} \cdot \dot{\hat{\mathbf{s}}}(t) + \nabla_{\boldsymbol{x}} e_{\boldsymbol{\theta}} \cdot \boldsymbol{f}_{\boldsymbol{\theta}}$$

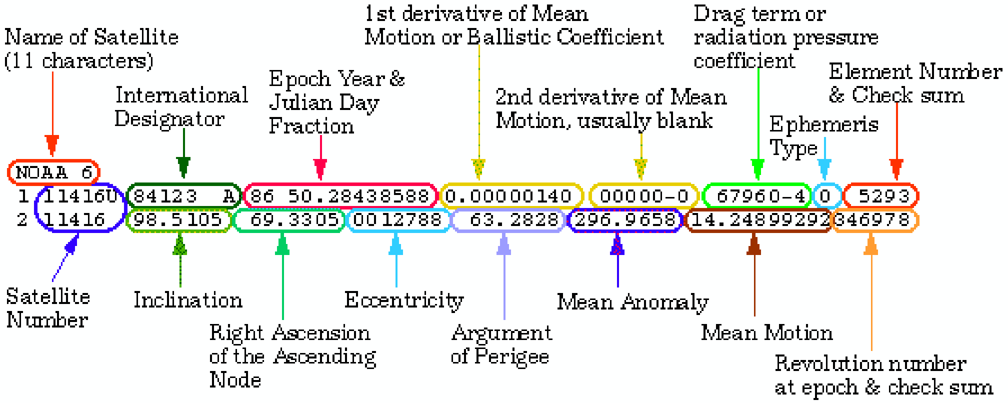


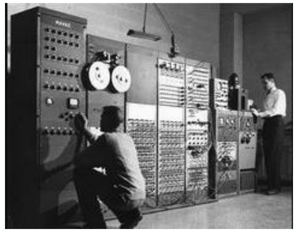
Differentiable propagation end-to-end

dSGP4

Acciarini, G., Baydin, A. G., Izzo, D.: **Closing the Gap Between SGP4 and High-Precision Propagation via Differentiable Programming**. To appear in: *Acta Astronautica,* 2025. https://doi.org/10.1016/j.actaastro.2024.10.063

dSGP4





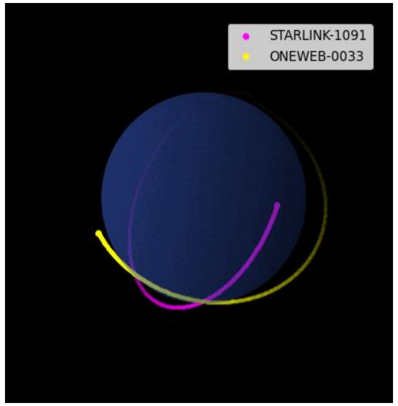
Originally designed to fit 80 character punch cards (1960s)

- So far we discussed force models to be used in **numerical integrators**..

  However, simplified integrators (**semi-analytical**) also exist

- **SGP4** is a popular simplified propagator that is used with **TLE** data (very commonly used)

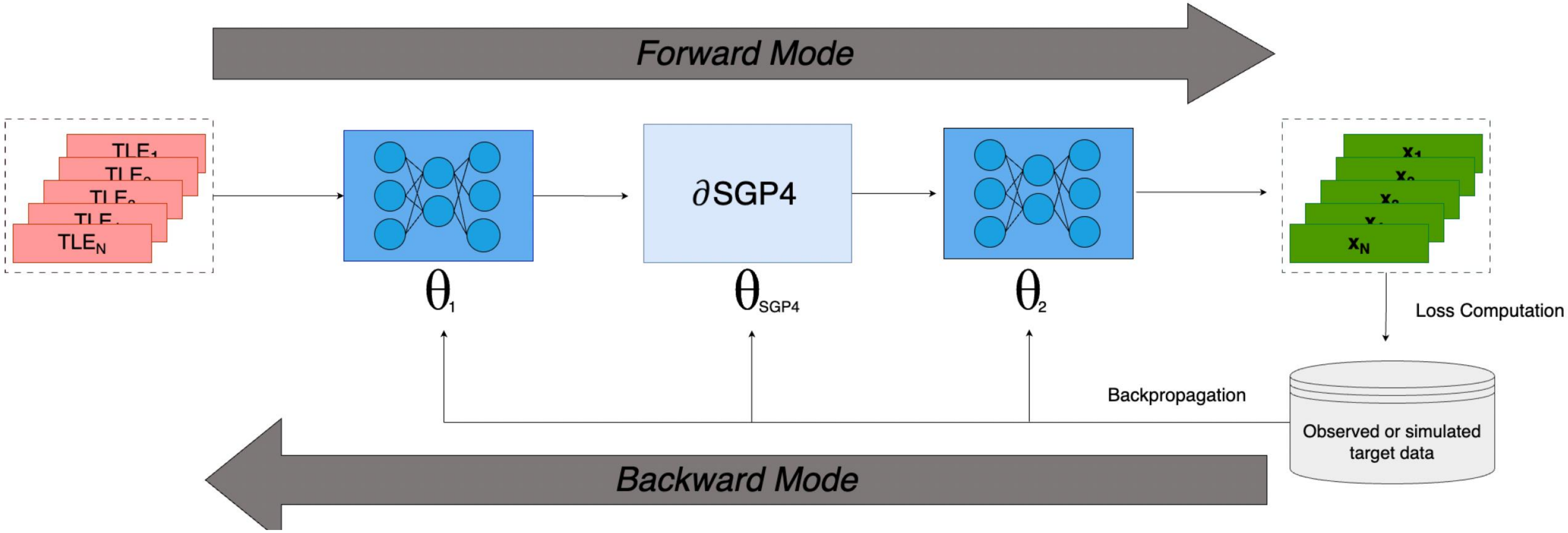
- We rewrote SGP4:
  - **Differentiable** SGP4 (via torch autodiff engine)
  - Compatible with modern **ML** software & hardware (TPUs, GPUs, CPUs..)
  - **Open source**



https://github.com/esa/dSGP4

ML & dSGP4

- **Stochastic gradient descent** can be applied to learn the NN & dSGP4 parameters to bridge the gap between **SGP4** and **observed/simulated** data



$$\mathcal{P}:\begin{cases}\text{given:} & \mathbf{x}_0, t\\ \text{find:} & \boldsymbol{\theta}_1, \boldsymbol{\theta}_{\text{SGP4}}, \boldsymbol{\theta}_2\\ \text{s.t. min:} & J(\mathbf{x}(\mathbf{x}_0, t, \boldsymbol{\theta}_1, \boldsymbol{\theta}_{\text{SGP4}}, \boldsymbol{\theta}_2)) = ||\mathbf{x} - \mathbf{x}_{\text{HPOP}}||^2\end{cases}$$

$$\boldsymbol{\theta}_{1,i+1} = \boldsymbol{\theta}_{1,i} - \eta_1 \nabla_{\boldsymbol{\theta}_1} J$$

$$\boldsymbol{\theta}_{\text{SGP4},i+1} = \boldsymbol{\theta}_{\text{SGP4},i} - \eta_{\text{SGP4}} \nabla_{\boldsymbol{\theta}_{\text{SGP4}}} J$$

$$\boldsymbol{\theta}_{2,i+1} = \boldsymbol{\theta}_{2,i} - \eta_2 \nabla_{\boldsymbol{\theta}_2} J$$

Torch autodiff

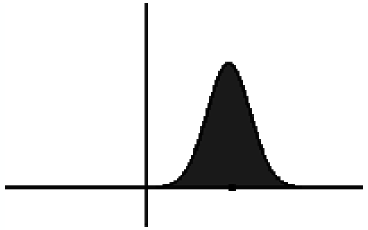
High Order

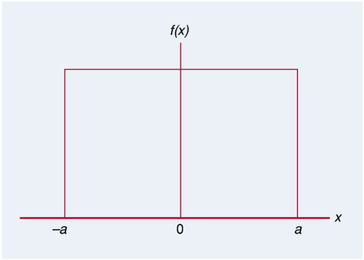
Acciarini, G., Baresi, N., Lloyd, D., and Izzo, D.: **Nonlinear Propoagation of Non-Gaussian Uncertainties.** Journal of Guidance, Control and Dynamics, 2025. https://arc.aiaa.org/doi/10.2514/1.G008717

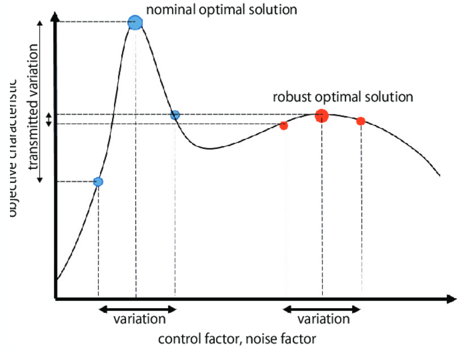
Izzo, D., Origer, S., Acciarini, G., Biscani, F.: **High-Order Expansion of Neural Ordinary Differential Equation flows**. Science Advances, 2025. https://www.science.org/doi/10.1126/sciadv.ady1348

Uncertainty Propagation

$$\frac{\partial}{\partial t}P(x,t) = -\frac{\partial}{\partial x}\left[A(x,t)\,P(x,t)\right]$$







- Objective is to propagate **statistical moments** of a distribution
- We leverage **high-order Taylor expansion** of the flow to do this
- The resulting technique can quickly and accurately **map uncertainties** at future times or events
- The **state** and **parameters** of the **dynamical system** are considered **as uncertain** and described by **any** pdf (uniform, Gaussian, etc.)

Acciarini, G., Baresi, N., Lloyd, D., Izzo, D., 2024. **Nonlinear Propagation of Non-Gaussian Uncertainties.** In: Journal of Guidance, Control and Dynamics

Mathematical Framework

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{q}) \longrightarrow \delta x_f^i = \mathcal{P}_i^k(\delta \boldsymbol{x}_0, \delta \boldsymbol{q}) + \mathcal{O}(k) \longrightarrow \delta x_f^i \approx \mathcal{P}_i^k(\delta \boldsymbol{z}) = \sum_{|\alpha|=1}^{k} \frac{1}{\alpha!} (\partial^\alpha x_f^i) \Big|_{(\overline{\boldsymbol{x}}_0, \overline{\boldsymbol{q}})} \delta \boldsymbol{z}^\alpha$$

$$\mathbb{E}[\delta x_f^i] \approx \mathbb{E}[\mathcal{P}_i^k(\delta \boldsymbol{z})] = \sum_{|\alpha|=1}^{k} \frac{1}{\alpha!} (\partial^\alpha x_f^i) \Big|_{(\overline{\boldsymbol{x}}_0, \overline{\boldsymbol{q}})} \mathbb{E}[\delta \boldsymbol{z}^\alpha]$$

Similarly, higher moments can also be derived

State Transition Tensors (STT): partial of the final state components w.r.t. the initial state. If there is an event → map inversion[1], substitution and the polynomial of the state at the event can be found → **Event Transition Tensors (ETT)**

$$e(\boldsymbol{x}_f(t^*; \boldsymbol{x}_0, \boldsymbol{q}), \boldsymbol{q}) = 0$$

High order moments of the initial distributions

[1]Berz, Martin. Modern map methods in particle beam physics. Academic Press, 1999.

Computing STT or ETT

$$\mathbb{E}[\delta x_f^i] \approx \mathbb{E}[\mathcal{P}_i^k(\delta z)] = \sum_{|\alpha|=1}^{k} \frac{1}{\alpha!}(\partial^\alpha x_f^i)\Big|_{(\bar{x}_0,\bar{q})} \mathbb{E}[\delta z^\alpha]$$

- Multiple techniques can be used for this (**mathematically equivalent**):

ML differentiable programs (leveraging forward/backward autodiff & Taylor-mode autodiff)

PyTorch

TensorFlow

Algebraic techniques (COSY, DACE, pyaudi)

dacelib / **dace**

darioizzo / **audi**

COSY INFINITY Version 9

Kyoko Makino, Martin Berz

Jet Transport

Integrating the **variational or adjoint equations** (heyoka)

Heyoka

- We use ***heyoka*** → leveraging recent advancements in it to automatically & efficiently write the VE

Moment Maps via MGF Approach

$$\mathbb{E}[\delta x_f^i] \approx \mathbb{E}[\mathcal{P}_i^k(\delta \boldsymbol{z})] = \sum_{|\alpha|=1}^{k} \frac{1}{\alpha!}(\partial^\alpha x_f^i)\Big|_{(\overline{\boldsymbol{x}}_0,\overline{\boldsymbol{q}})} \quad \boxed{\mathbb{E}[\delta \boldsymbol{z}^\alpha]}$$

- Via **moment generating functions (MGF)**, for any pdf that admits an MGF we can write:

$$\mathbb{E}[X_1^{k_1} \dots X_n^{k_n}] = \frac{\partial^k}{\partial t_1^{k_1} \dots \partial t_n^{k_n}} M_{\mathbf{X}}(\mathbf{t})\Big|_{\mathbf{t}=0}$$

Heyoka

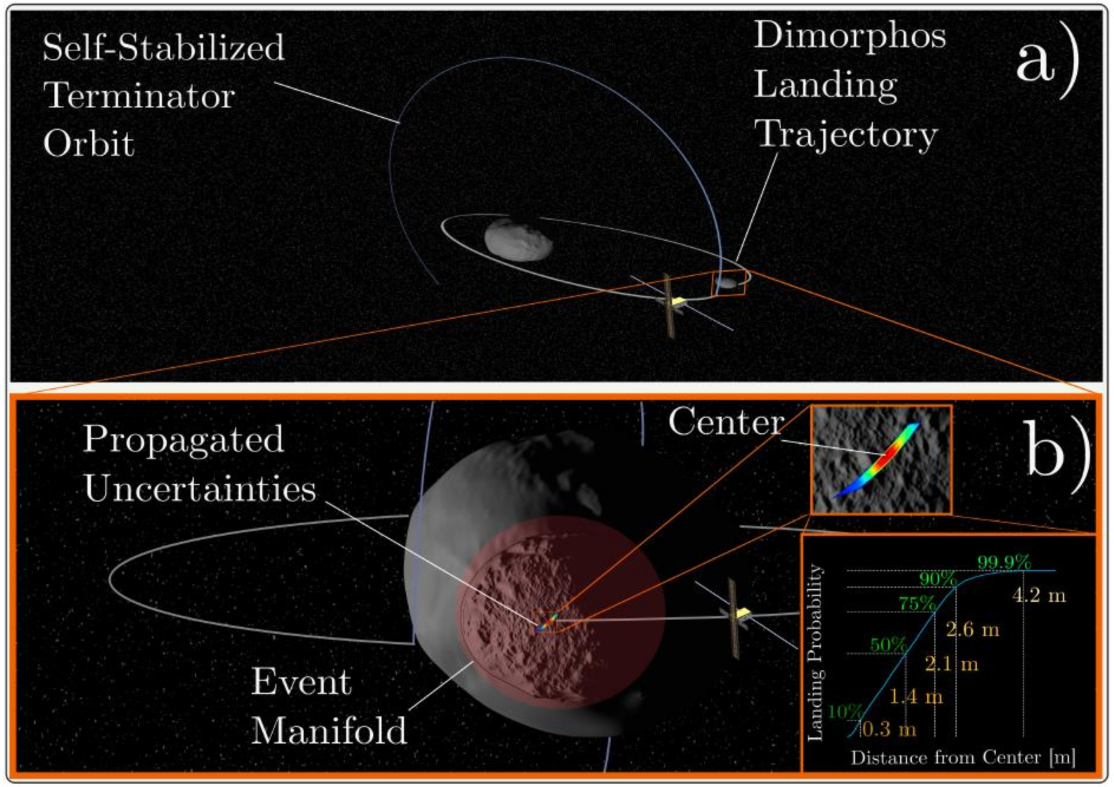
- These can be computed **offline**, only once

- Moreover, they can be computed **symbolically** w.r.t. pdf parameters, which means they can be re-used for different parameters of the pdf

Distribution	Moment-generating function $M_X(t)$
Degenerate δ_a	e^{ta}
Bernoulli $P(X=1)=p$	$1-p+pe^t$
Geometric $(1-p)^{k-1}p$	$\frac{pe^t}{1-(1-p)e^t},\ t<-\ln(1-p)$
Binomial $B(n,p)$	$(1-p+pe^t)^n$
Negative binomial $\mathrm{NB}(r,p)$	$\left(\frac{p}{1-e^t+pe^t}\right)^r,\ t<-\ln(1-p)$
Poisson $\mathrm{Pois}(\lambda)$	$e^{\lambda(e^t-1)}$
Uniform (continuous) $\mathrm{U}(a,b)$	$\frac{e^{tb}-e^{ta}}{t(b-a)}$
Uniform (discrete) $\mathrm{DU}(a,b)$	$\frac{e^{at}-e^{(b+1)t}}{(b-a+1)(1-e^t)}$
Laplace $L(\mu,b)$	$\frac{e^{t\mu}}{1-b^2t^2},\ \|t\|<1/b$
Normal $N(\mu,\sigma^2)$	$e^{t\mu+\frac{1}{2}\sigma^2t^2}$
Chi-squared χ_k^2	$(1-2t)^{-\frac{k}{2}},\ t<1/2$
Noncentral chi-squared $\chi_k^2(\lambda)$	$e^{\lambda t/(1-2t)}(1-2t)^{-\frac{k}{2}}$
Gamma $\Gamma(k,\theta)$	$(1-t\theta)^{-k},\ t<\frac{1}{\theta}$
Exponential $\mathrm{Exp}(\lambda)$	$(1-t\lambda^{-1})^{-1},\ t<\lambda$
Beta	$1+\sum_{k=1}^{\infty}\left(\prod_{r=0}^{k-1}\frac{\alpha+r}{\alpha+\beta+r}\right)\frac{t^k}{k!}$
Multivariate normal $N(\mu,\Sigma)$	$e^{\mathbf{t}^{\mathrm{T}}\left(\mu+\frac{1}{2}\Sigma\mathbf{t}\right)}$

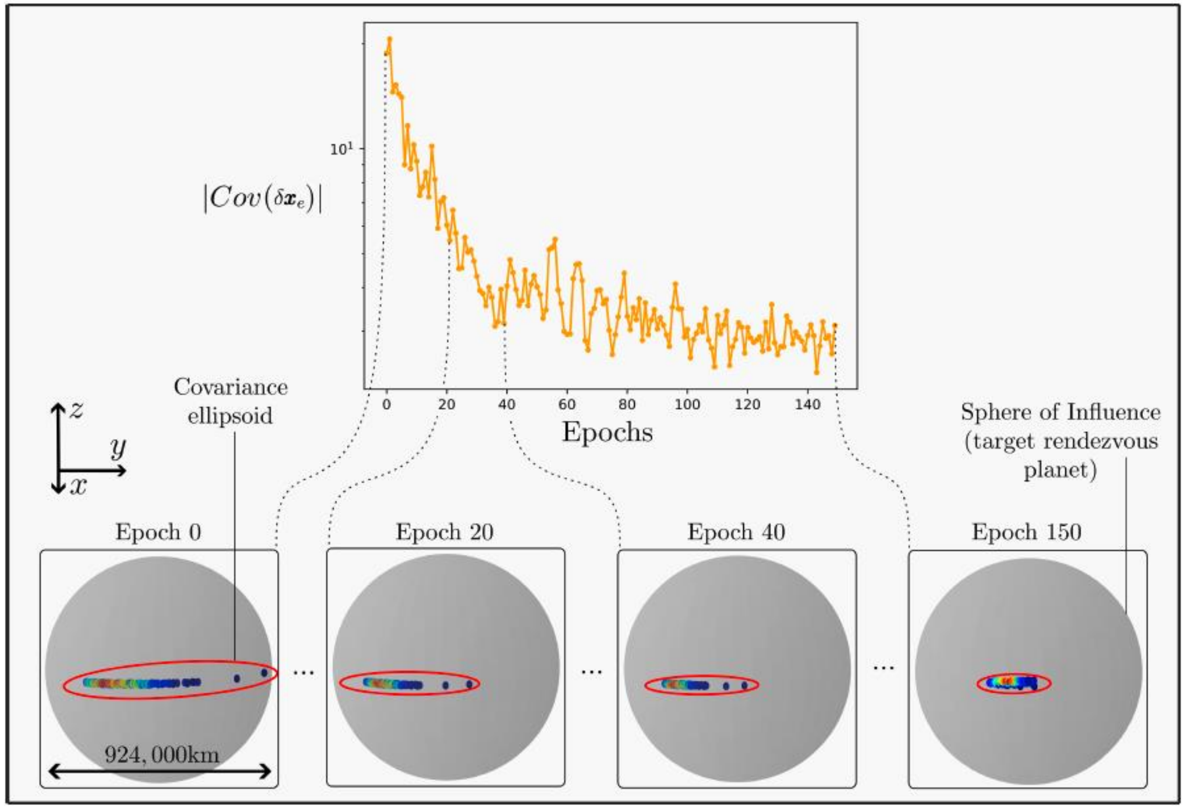
Example Application

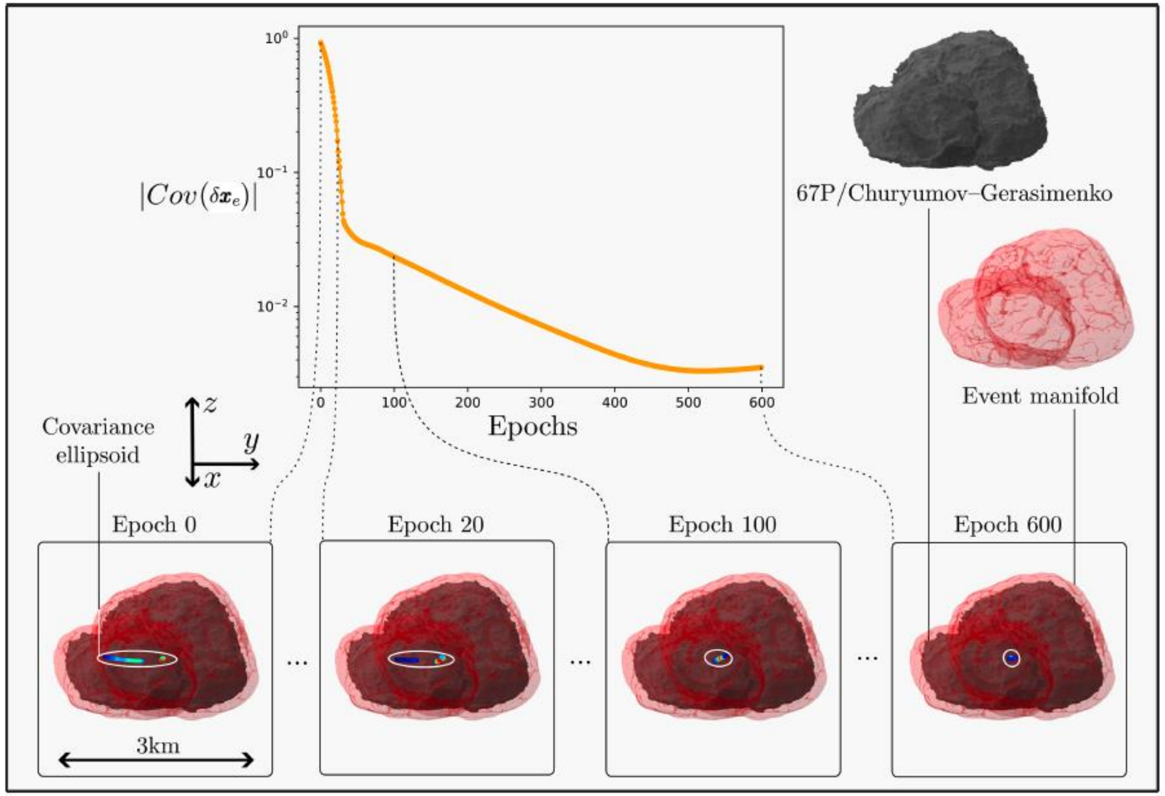
Landing on a small body in a binary asteroid environment (e.g. the one of the recent ESA's Hera mission)



Note that the discussed techniques are also applicable for cases in which there is a neural network in the right-hand side of the ODEs (since they only require differentiability)

Coming soon…. robust control





Thanks for listening!