SU(3) holonomy/structure metrics for CICYs and Toric Varieties

FABIAN RUEHLE

string_data 2021

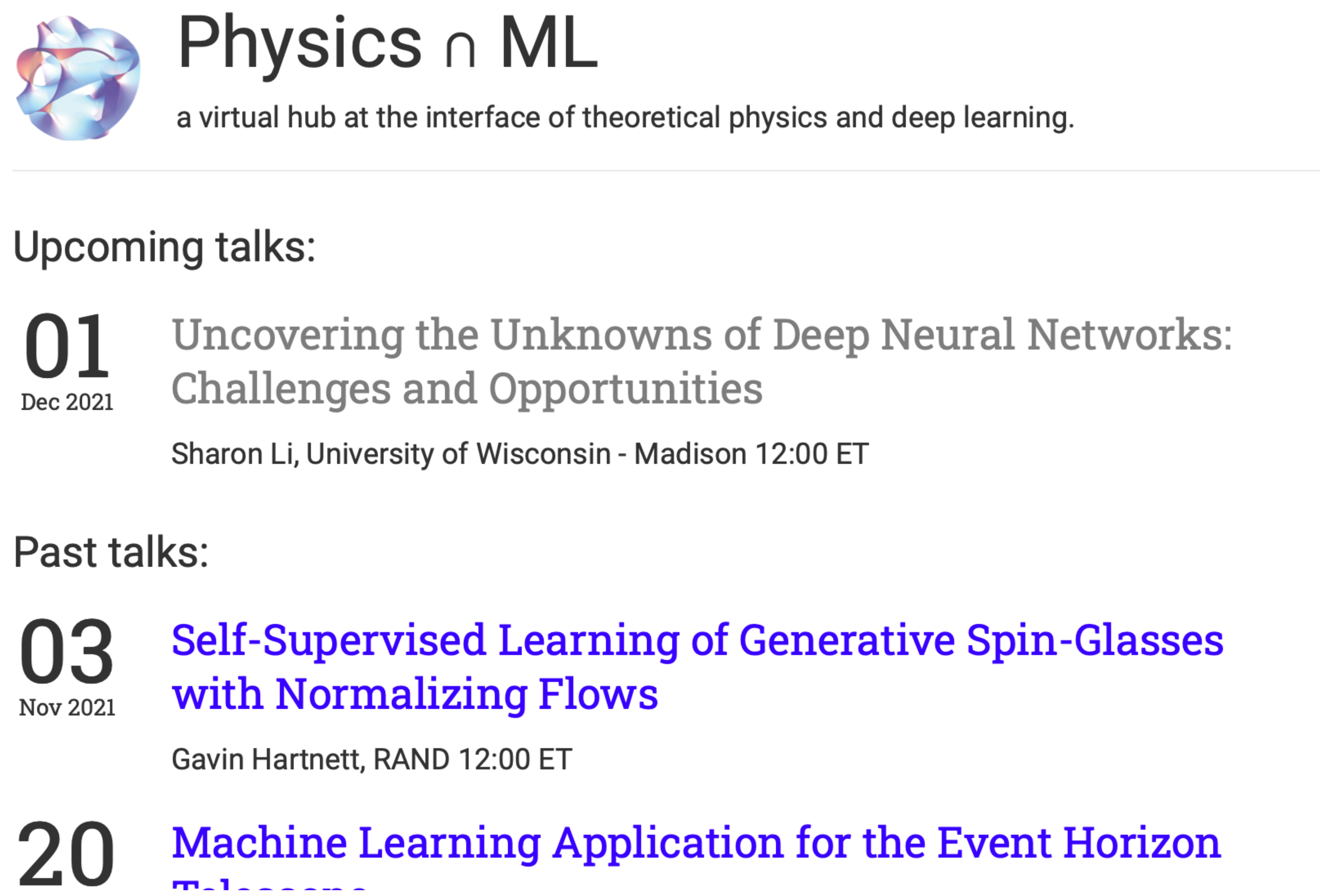
December 16, 2021

Based on:

[Larfors, Lukas, FR, Schneider: 2111.01436 & to appear]

[Anderson, Gray, Gerdes, Krippendorf, Raghuram, FR: 2012.04656]

If you are interested



Physics ∩ ML

a virtual hub at the interface of theoretical physics and deep learning.

Upcoming talks:

01 Dec 2021 — Uncovering the Unknowns of Deep Neural Networks: Challenges and Opportunities

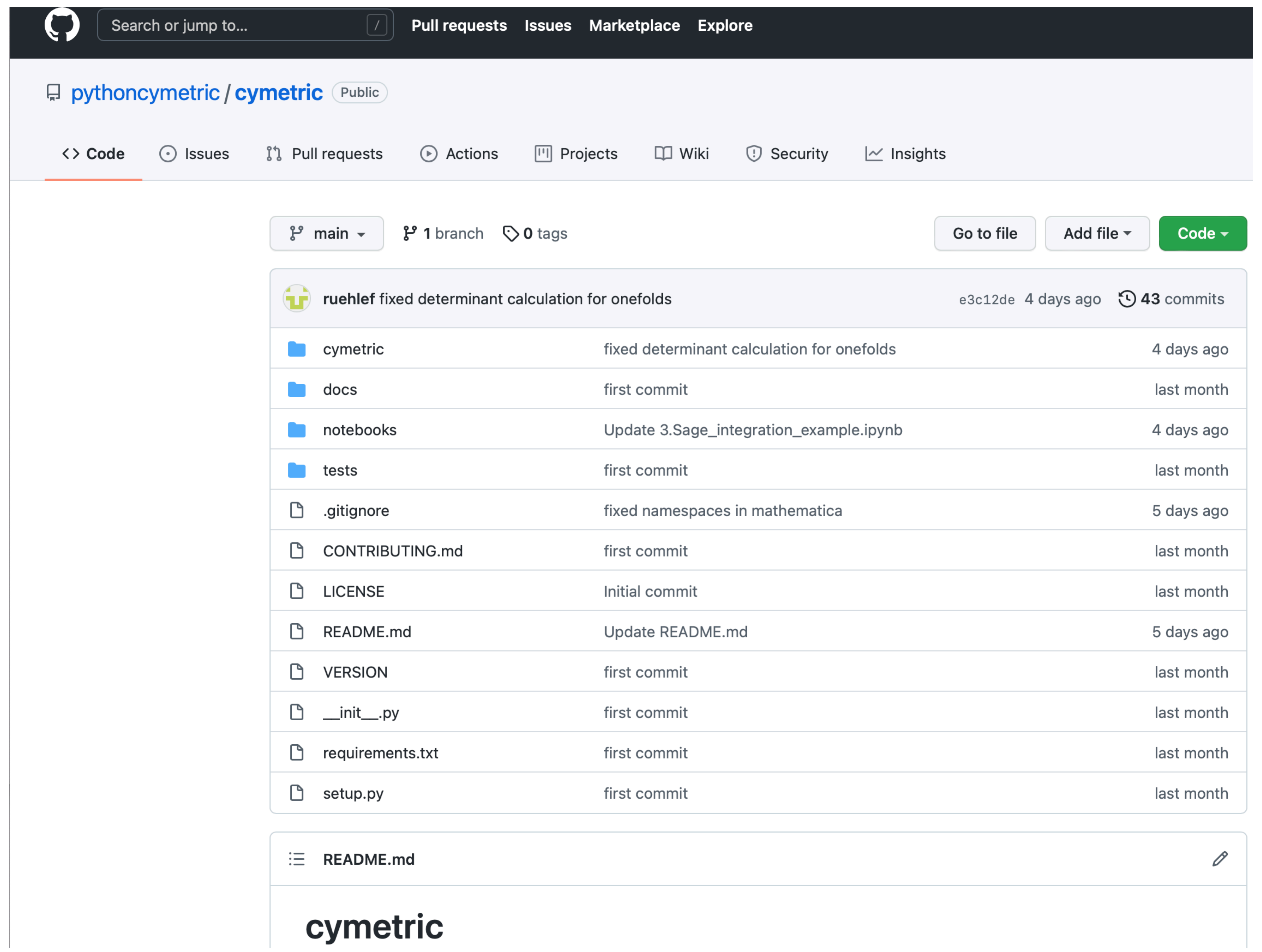
Sharon Li, University of Wisconsin - Madison 12:00 ET

Past talks:

03 Nov 2021 — Self-Supervised Learning of Generative Spin-Glasses with Normalizing Flows

Gavin Hartnett, RAND 12:00 ET

20 — Machine Learning Application for the Event Horizon



pythoncymetric / cymetric Public

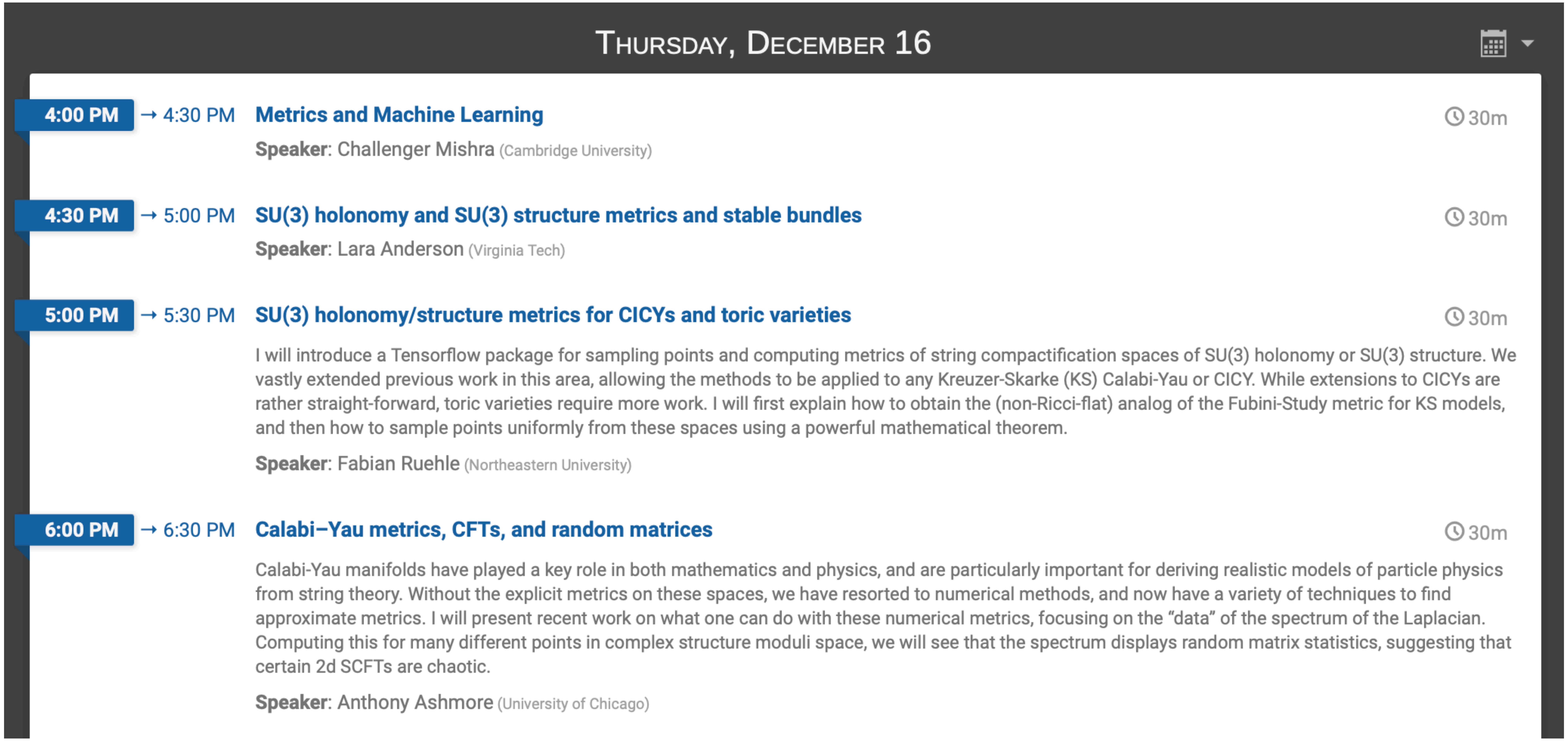
main · 1 branch · 0 tags

ruehlef fixed determinant calculation for onefolds — e3c12de 4 days ago — 43 commits

cymetric	fixed determinant calculation for onefolds	4 days ago
docs	first commit	last month
notebooks	Update 3.Sage_integration_example.ipynb	4 days ago
tests	first commit	last month
.gitignore	fixed namespaces in mathematica	5 days ago
CONTRIBUTING.md	first commit	last month
LICENSE	Initial commit	last month
README.md	Update README.md	5 days ago
VERSION	first commit	last month
__init__.py	first commit	last month
requirements.txt	first commit	last month
setup.py	first commit	last month

README.md

cymetric


Thursday, December 16

4:00 PM → 4:30 PM **Metrics and Machine Learning** 30m

Speaker: Challenger Mishra (Cambridge University)

4:30 PM → 5:00 PM **SU(3) holonomy and SU(3) structure metrics and stable bundles** 30m

Speaker: Lara Anderson (Virginia Tech)

5:00 PM → 5:30 PM **SU(3) holonomy/structure metrics for CICYs and toric varieties** 30m

I will introduce a Tensorflow package for sampling points and computing metrics of string compactification spaces of SU(3) holonomy or SU(3) structure. We vastly extended previous work in this area, allowing the methods to be applied to any Kreuzer-Skarke (KS) Calabi-Yau or CICY. While extensions to CICYs are rather straight-forward, toric varieties require more work. I will first explain how to obtain the (non-Ricci-flat) analog of the Fubini-Study metric for KS models, and then how to sample points uniformly from these spaces using a powerful mathematical theorem.

Speaker: Fabian Ruehle (Northeastern University)

6:00 PM → 6:30 PM **Calabi–Yau metrics, CFTs, and random matrices** 30m

Calabi-Yau manifolds have played a key role in both mathematics and physics, and are particularly important for deriving realistic models of particle physics from string theory. Without the explicit metrics on these spaces, we have resorted to numerical methods, and now have a variety of techniques to find approximate metrics. I will present recent work on what one can do with these numerical metrics, focusing on the “data” of the spectrum of the Laplacian. Computing this for many different points in complex structure moduli space, we will see that the spectrum displays random matrix statistics, suggesting that certain 2d SCFTs are chaotic.

Speaker: Anthony Ashmore (University of Chicago)

Motivation

Introduction to CY metrics

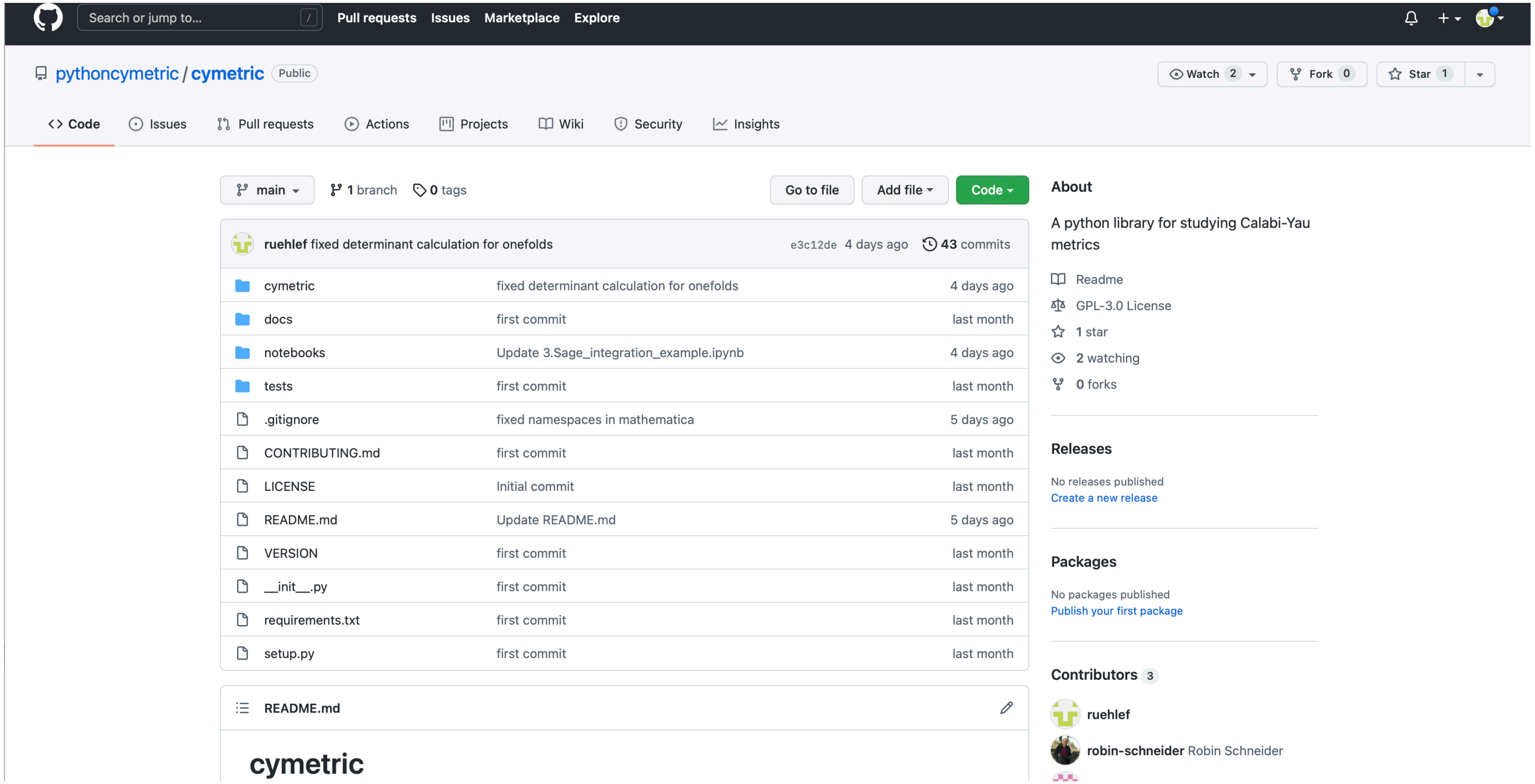
Challenger:
CY metrics

Lara:
CY + SU(3) structure metrics

Robin:
Introduction to cymetric package

Outline

- The cymetric package **[Robin's talk]**
- CY metrics for manifolds with arbitrary number of Kähler and CS moduli
  - Fixing the Kähler class
  - CY metrics for toric varieties and CICYs
    - Constructing the patches
    - Constructing the holomorphic top form
    - Constructing a canonical Kähler metric
    - Sampling points and finding weights
- Conclusion



ruehlef fixed determinant calculation for onefolds e3c12de 4 days ago 43 commits

cymetric	fixed determinant calculation for onefolds	4 days ago
docs	first commit	last month
notebooks	Update 3.Sage_integration_example.ipynb	4 days ago
tests	first commit	last month
.gitignore	fixed namespaces in mathematica	5 days ago
CONTRIBUTING.md	first commit	last month
LICENSE	Initial commit	last month
README.md	Update README.md	5 days ago
VERSION	first commit	last month
__init__.py	first commit	last month
requirements.txt	first commit	last month
setup.py	first commit	last month

README.md

cymetric

About

A python library for studying Calabi-Yau metrics

Readme
GPL-3.0 License
1 star
2 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

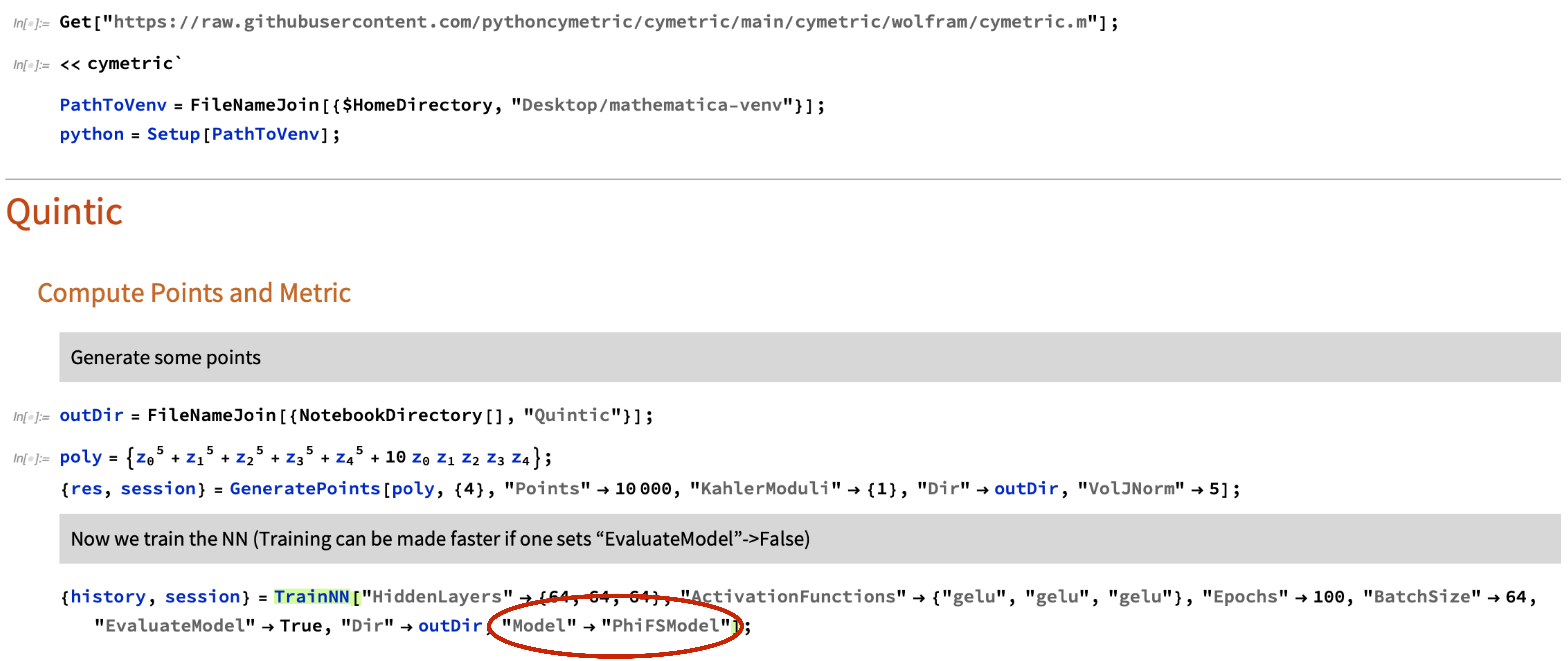
ruehlef
robin-schneider Robin Schneider

The cymetric package

Our goal

- Provide a package that
  - Is very fast and can deal with large number of complex structure parameters
  - Can deal with $h^{1,1} \geq 1$ in a **user-specified, fixed** Kähler class
  - Works in principle for **any** of the **CY** constructions typically explored:
    - The CICY list
    - The KS list
  - Integrates into current workflow (Mathematica, Sage)

The cymetric package in Mathematica



```
In[∘]:= Get["https://raw.githubusercontent.com/pythoncymetric/cymetric/main/cymetric/wolfram/cymetric.m"];

In[∘]:= << cymetric`

        PathToVenv = FileNameJoin[{$HomeDirectory, "Desktop/mathematica-venv"}];
        python = Setup[PathToVenv];
```

Quintic

Compute Points and Metric

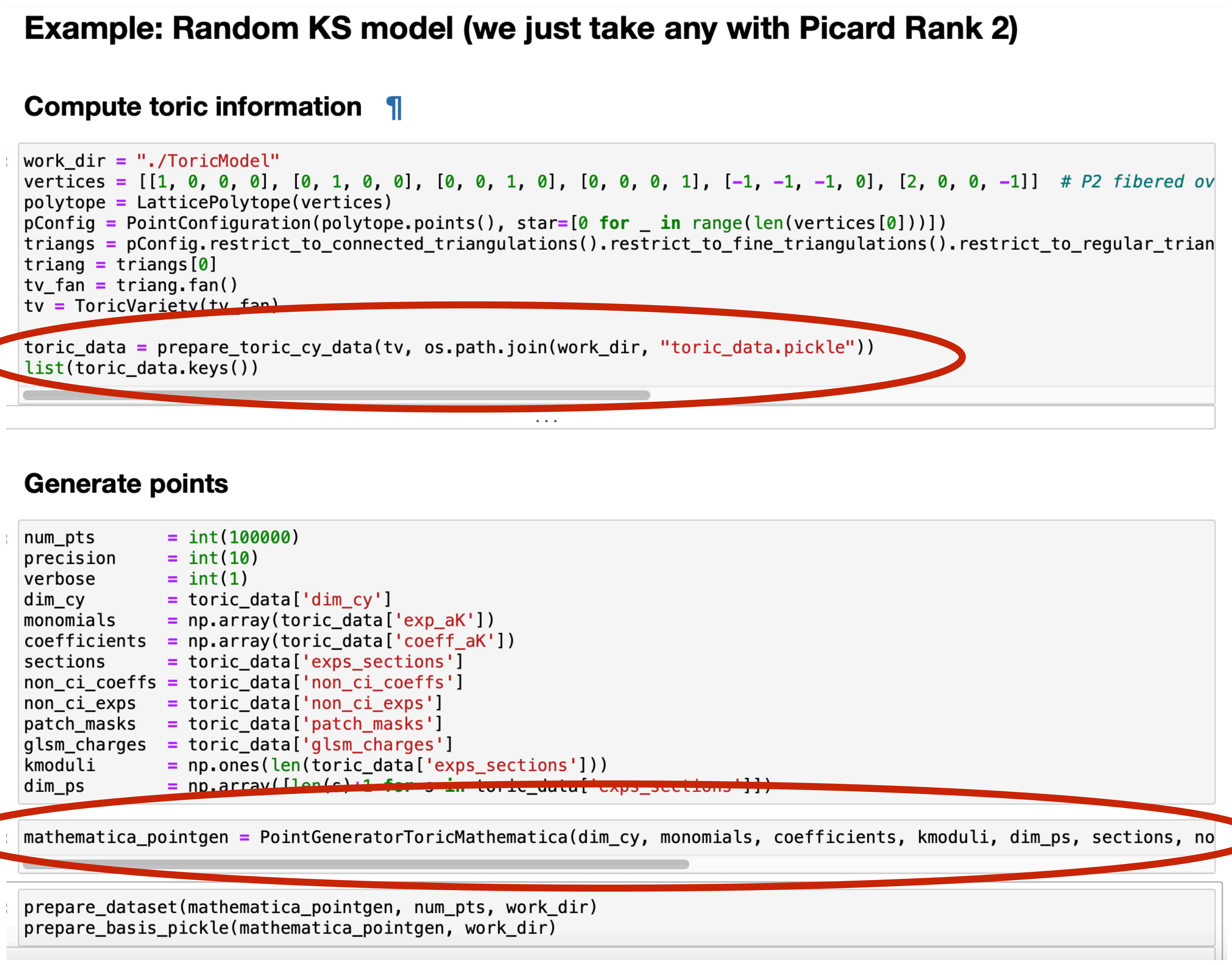
Generate some points

```
In[∘]:= outDir = FileNameJoin[{NotebookDirectory[], "Quintic"}];

In[∘]:= poly = {z0^5 + z1^5 + z2^5 + z3^5 + z4^5 + 10 z0 z1 z2 z3 z4};
        {res, session} = GeneratePoints[poly, {4}, "Points" → 10 000, "KahlerModuli" → {1}, "Dir" → outDir, "VolJNorm" → 5];
```

Now we train the NN (Training can be made faster if one sets “EvaluateModel”->False)

```
{history, session} = TrainNN["HiddenLayers" → {64, 64, 64}, "ActivationFunctions" → {"gelu", "gelu", "gelu"}, "Epochs" → 100, "BatchSize" → 64,
    "EvaluateModel" → True, "Dir" → outDir, "Model" → "PhiFSModel"];
```

The cymetric package in Sage



Example: Random KS model (we just take any with Picard Rank 2)

Compute toric information ¶

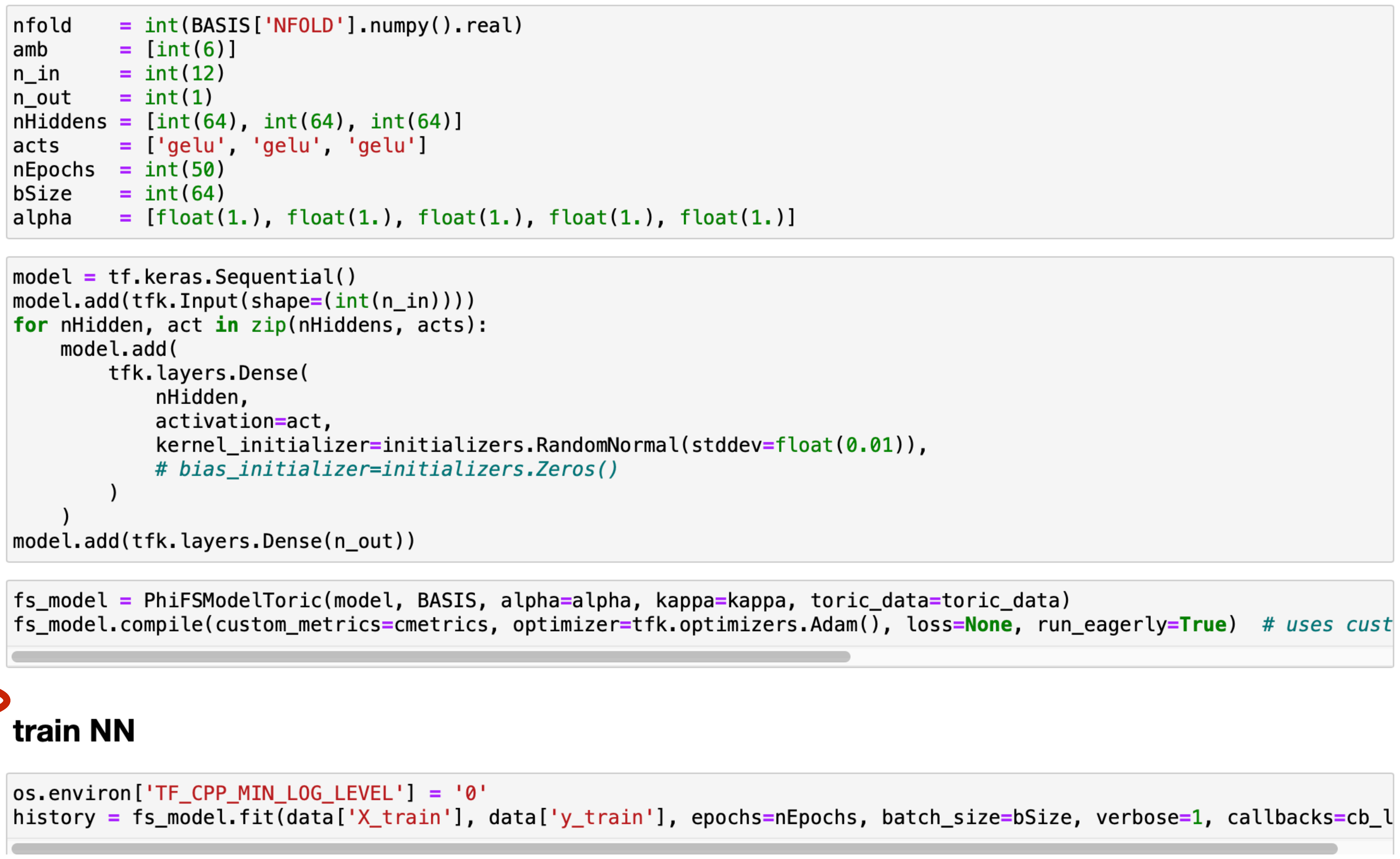
```
work_dir = "./ToricModel"
vertices = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1], [-1, -1, -1, 0], [2, 0, 0, -1]]  # P2 fibered ov
polytope = LatticePolytope(vertices)
pConfig = PointConfiguration(polytope.points(), star=[0 for _ in range(len(vertices[0]))])
triangs = pConfig.restrict_to_connected_triangulations().restrict_to_fine_triangulations().restrict_to_regular_trian
triang = triangs[0]
tv_fan = triang.fan()
tv = ToricVariety(tv_fan)

toric_data = prepare_toric_cy_data(tv, os.path.join(work_dir, "toric_data.pickle"))
list(toric_data.keys())
```

Generate points

```
num_pts        = int(100000)
precision      = int(10)
verbose        = int(1)
dim_cy         = toric_data['dim_cy']
monomials      = np.array(toric_data['exp_aK'])
coefficients   = np.array(toric_data['coeff_aK'])
sections       = toric_data['exps_sections']
non_ci_coeffs  = toric_data['non_ci_coeffs']
non_ci_exps    = toric_data['non_ci_exps']
patch_masks    = toric_data['patch_masks']
glsm_charges   = toric_data['glsm_charges']
kmoduli        = np.ones(len(toric_data['exps_sections']))
dim_ps         = np.array([len(s)-1 for s in toric_data['exps_sections']])
```

```
mathematica_pointgen = PointGeneratorToricMathematica(dim_cy, monomials, coefficients, kmoduli, dim_ps, sections, no
```

```
prepare_dataset(mathematica_pointgen, num_pts, work_dir)
prepare_basis_pickle(mathematica_pointgen, work_dir)
```



```
nfold    = int(BASIS['NFOLD'].numpy().real)
amb      = [int(6)]
n_in     = int(12)
n_out    = int(1)
nHiddens = [int(64), int(64), int(64)]
acts     = ['gelu', 'gelu', 'gelu']
nEpochs  = int(50)
bSize    = int(64)
alpha    = [float(1.), float(1.), float(1.), float(1.), float(1.)]
```

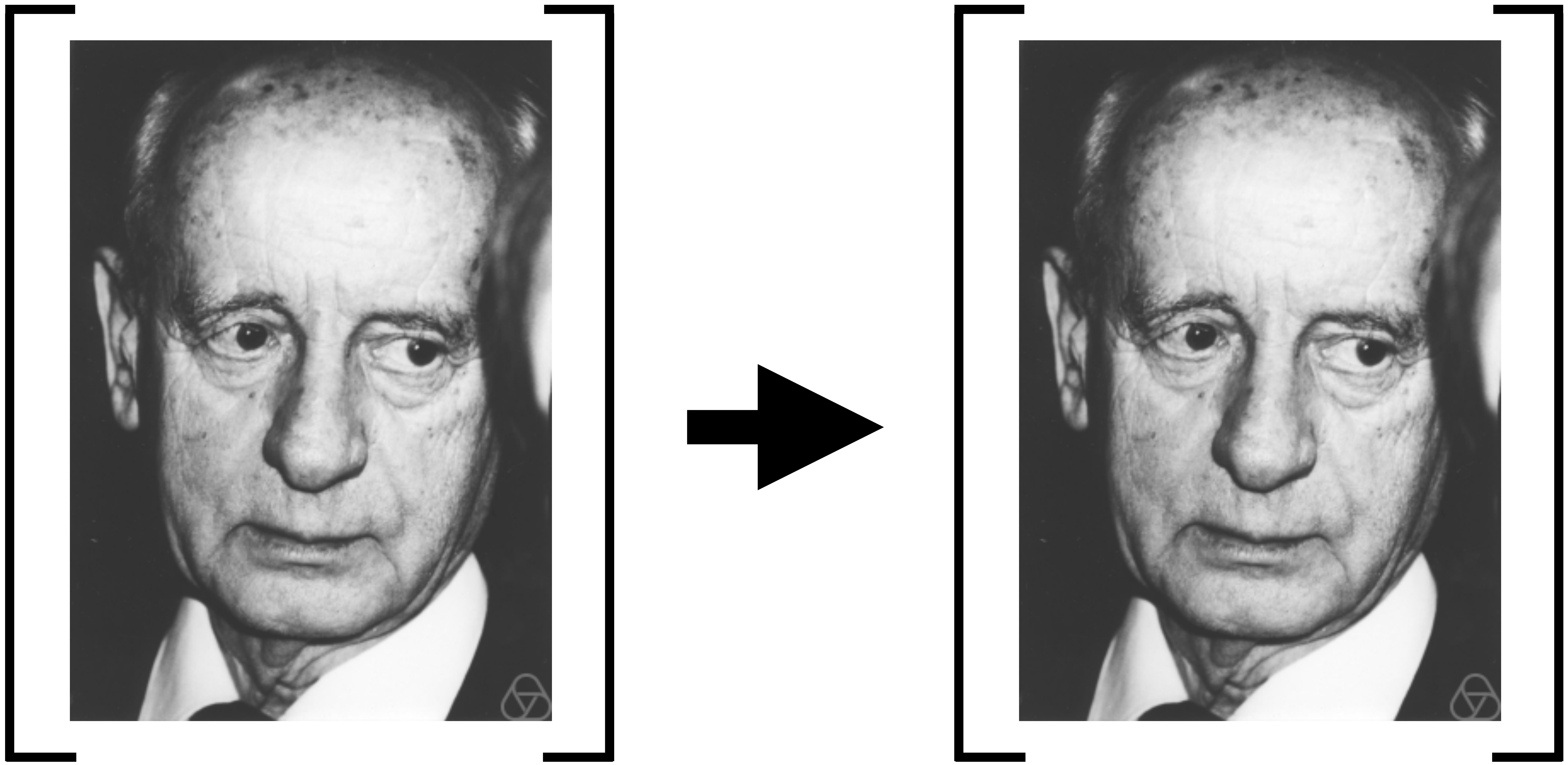
```
model = tf.keras.Sequential()
model.add(tfk.Input(shape=(int(n_in))))
for nHidden, act in zip(nHiddens, acts):
    model.add(
        tfk.layers.Dense(
            nHidden,
            activation=act,
            kernel_initializer=initializers.RandomNormal(stddev=float(0.01)),
            # bias_initializer=initializers.Zeros()
        )
    )
model.add(tfk.layers.Dense(n_out))
```

```
fs_model = PhiFSModelToric(model, BASIS, alpha=alpha, kappa=kappa, toric_data=toric_data)
fs_model.compile(custom_metrics=cmetrics, optimizer=tfk.optimizers.Adam(), loss=None, run_eagerly=True)  # uses cust
```

train NN

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '0'
history = fs_model.fit(data['X_train'], data['y_train'], epochs=nEpochs, batch_size=bSize, verbose=1, callbacks=cb_l
```

NN :



Fixing the Kähler class

Previous metric ansatze

- Optimizing $\mathrm{Ric}_{\mu\nu} = 0$ requires 4 derivatives $\Rightarrow$ look at **surrogate loss**
- The condition $J^3 = \kappa|\Omega|^2$ can be turned into a (Monge-Ampere) PDE

$$\boxed{g_{\mathrm{CY}} = g_{\mathrm{reference}} + \partial\bar{\partial}\Phi \quad \Rightarrow \quad \det(g_{\mathrm{reference}} + \partial\bar{\partial}\Phi) = \kappa\Omega \wedge \bar{\Omega}}$$

- …but actually we are in a peculiar situation here: We have a PDE in Φ, but we do not care about Φ; this would be the Kähler potential of the CY manifold, so it is not even unique
- Rather, we care about $g_{\mathrm{correction}} = \partial\bar{\partial}\Phi$, and learn this correction via a NN

Previous metric ansatze

- … so learn $g_{\mathrm{NN}} = \partial\bar{\partial}\Phi$, turning the surrogate loss into an algebraic equation
- Hence the MA loss in previous works **is not** even **a PDE** (let alone of MA type)
- Need to impose that the metric is Kähler, which requires taking one derivative

$$\begin{tikzcd}
\text{Ricci flat (4 derivatives)} \arrow[r] & \text{MA equation (2 derivatives)} \arrow[r] & \text{MA loss (0 derivatives)} \arrow[d] \\
 & & \text{Kähler loss (1 derivative)}
\end{tikzcd}$$

- Possibile ansatze:

$$g_{\mathrm{CY}} = g_{\mathrm{NN}}$$

$$g_{\mathrm{CY}} = g_{\mathrm{reference}} + g_{\mathrm{NN}}$$ **[Anderson, Gray, Gerdes, Krippendorf, Raghuram, FR: 2012.04656]**

$$g_{\mathrm{CY}} = g_{\mathrm{reference}}(\mathbb{1} + g_{\mathrm{NN}})$$ **[Lara's talk]**

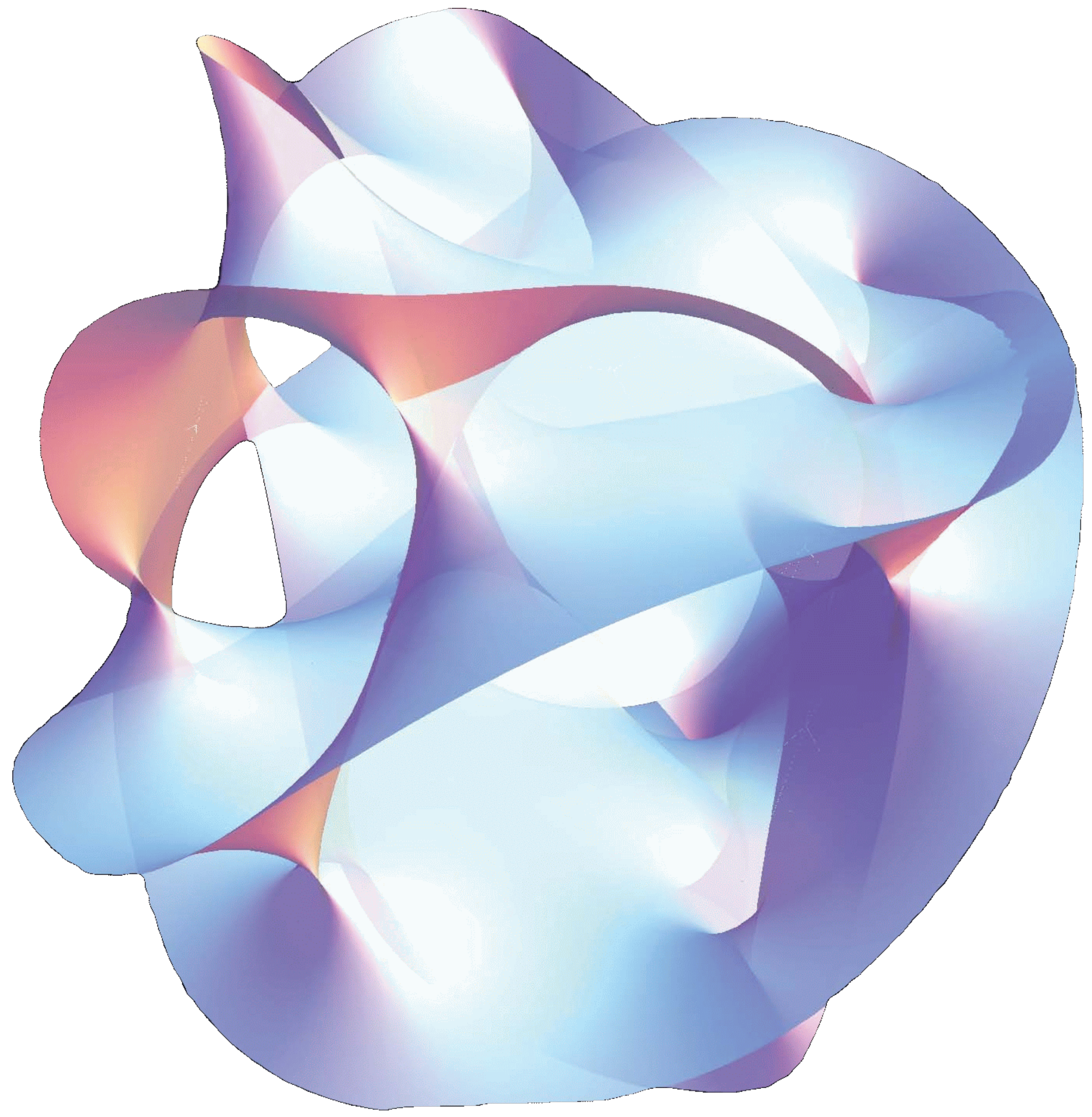
Fixing the Kähler class

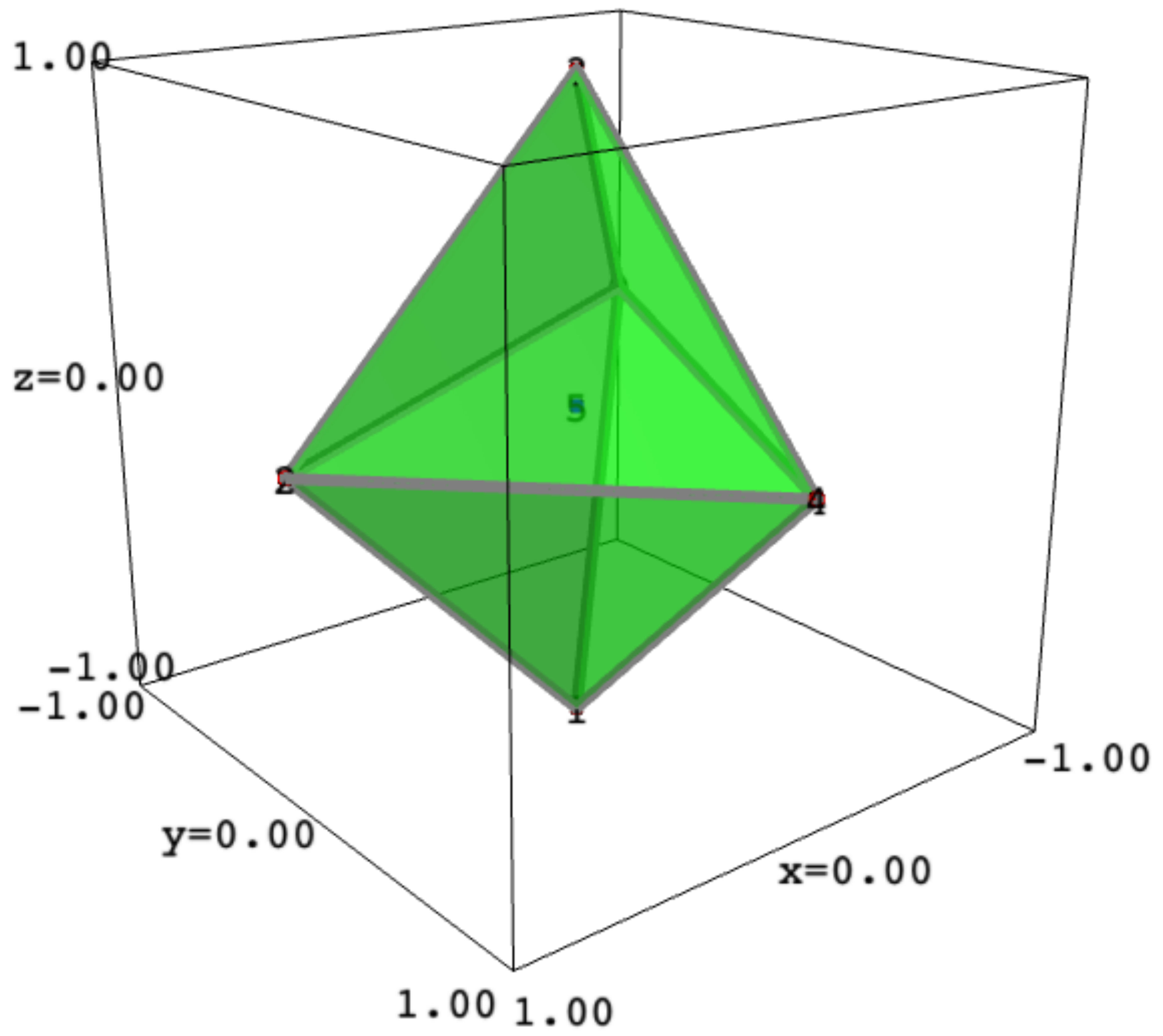
- But if we just learn $g_{\text{NN}} = \partial\bar{\partial}\Phi$ rather than Φ , we will (generically) change the Kähler class. So we have 2 options:
  - Since the Kähler class $[J]$ is fixed by

$$\int_C J = \text{vol}(C) \quad , \quad \int_D J \wedge J = \text{vol}(D) \quad , \quad \int_X J \wedge J \wedge J = \text{vol}(X)$$

  we could fix it by sampling points on curves and divisors and impose that the volume stays fixed during training (not yet implemented)

  - In contrast, learning Φ as an exact correction to $g_{\text{CY}} = g_{\text{reference}} + \partial\bar{\partial}\Phi$ the Kähler class cannot change. We call this the **Phi Network.**





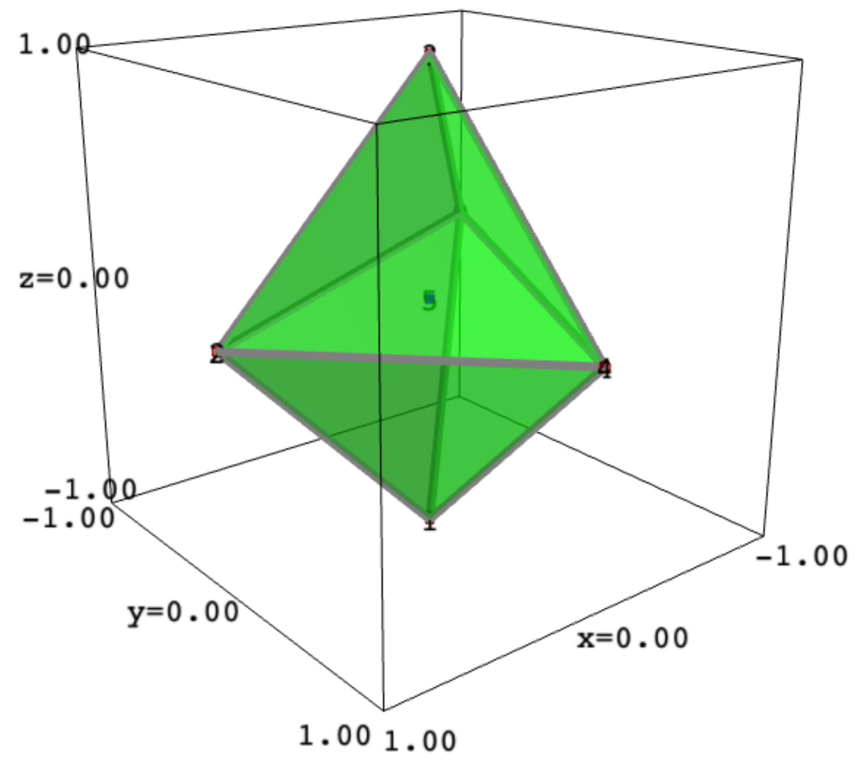
CY metrics for toric varieties and CICYs

Constructing the patches

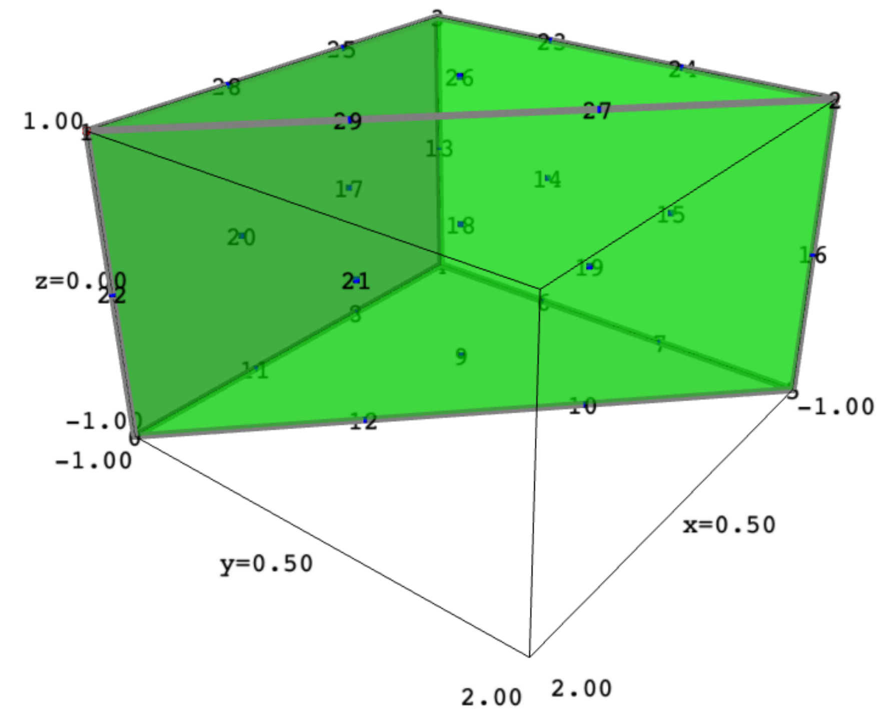
- For CICYs, this is easy enough: Just divide by one of the homogeneous coordinates of each projective ambient space factor
- For toric varieties, the patches can be constructed from the toric polytope and its dual

$$z_\mu = \prod_{i=1}^{n} x_i^{\langle v_i, w_\mu \rangle}$$

$v_i \in$



$w_\mu \in$



The transition functions are just the Jacobians of the coordinate change: $T_\mu^\nu = \dfrac{\partial z_\mu}{\partial z_\nu}$

Constructing the holomorphic (3,0) form

- This can be done in complete generality (i.e. for nef partitions in toric ambient spaces). But we focus here on
  - **KS** models (toric hypersurfaces):

$$\boxed{\Omega = \frac{dz_1 \wedge dz_2 \wedge dz_3}{\partial p / \partial z_4}}$$

  - **CICYs** in projective spaces:

$$\boxed{\Omega = \frac{dz_1 \wedge dz_2 \wedge dz_3}{\det(\partial p_\delta / \partial z_a)}}$$

Constructing a Kähler metric for CICYs

- For CICYs, this is easy enough:
  - Go to an affine patch
  - Construct the FS metric in each $\mathbb{P}^{n_i}$: $$\boxed{g_{\mathrm{FS}}^{(n_i)} = \frac{t_i}{2\pi} \partial\overline{\partial} \ln \sum_{\mu=1}^{n_i} |z_\mu|^2}$$
  - Put them into one big, block-diagonal matrix
  - Pull the ambient space metric back to the CY using the defining equations

Constructing a Kähler metric for KS

- For toric varieties, let me do something that might look overly complicated
- First, we compute the sections of the Kähler cone generators

$$\boxed{D = \sum_{i=1}^{n} c_i D_i \quad \to \quad \Gamma(D) \sim \left\{ \prod_{i=1}^{n} x_i^{\langle v_i, w\rangle + c_i} \;\middle|\; \langle v_i, w\rangle + c_i \geq 0\,, \quad w \in M \right\}}$$

- The basic point is that the sections of base-point-free divisors cannot all vanish in any given patch
- Moreover, they are all homogeneous (after all, they are all sections of the same line bundle)

Constructing a Kähler metric for KS

- This should look familiar (from just a standard $\mathbb{P}^n$):
  - All sections have the same scaling
  - In each patch, one section is always non-vanishing
- I can construct maps Φ_i from the toric ambient space coordinates into the projectivization $\mathbb{P}H^0(D_i) \simeq \mathbb{CP}^{h^0(D_i)-1} := \mathbb{CP}^{r_i}$

$$\boxed{\Phi_i : \ [x_0 : x_1 : \ldots : x_N] \to [s_0^{(i)}(x) : s_1^{(i)}(x) : \ldots : s_{r_i}^{(i)}(x)] \qquad , \ i = 1, 2, \ldots, h^{1,1}(\mathcal{A})}$$

- If the ambient space itself is $\mathbb{P}^n$: $J_i = H_i, \ s_i = x_i$, and this is the Kodaira embedding for $\mathcal{O}(1)$

Constructing a Kähler metric for KS

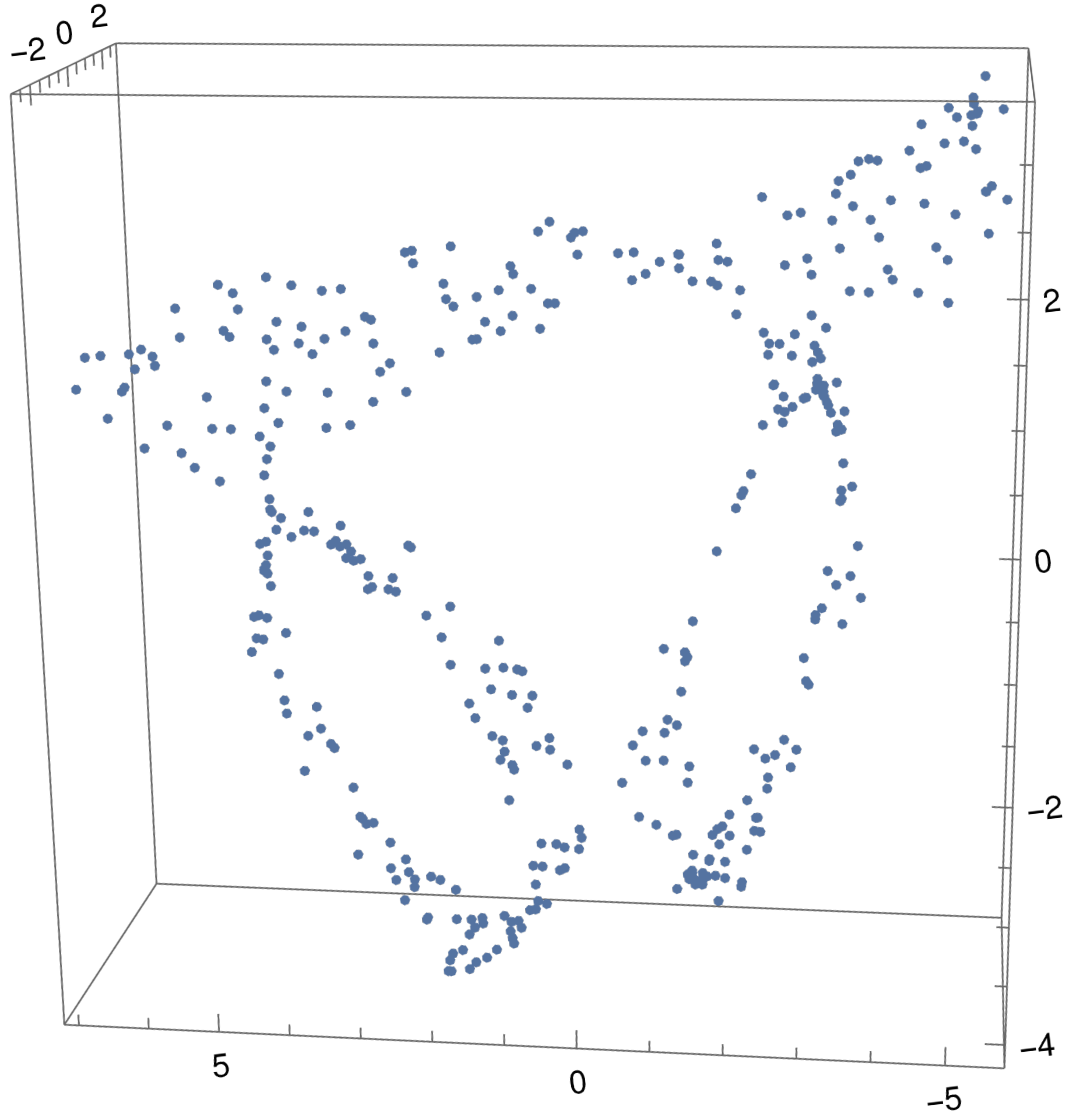
- Now I can write down a Kähler metric in this projectivized KC section space $\mathbb{P}^{r_i}$

$$K = \ln(s_j^\dagger\ H^{jk}\ s_k)\ , \qquad \vec{s} = (s_0^{(i)}, s_1^{(i)}, \ldots, s_{r_i}^{(i)}) \quad , \ H \text{ hermitian}$$

- This is well defined since not all s_j vanish
- For $H = \mathbb{1}$, get the standard FS metric in projectivized section space

This metric is also a well-defined (non-FS) Kähler metric for the toric ambient space (when written in terms of x rather than s)

- Construct this FS metric in $\mathbb{P}^{r_i}$ for each KC generator and write it as a (non-block-diagonal) metric in terms of the toric coordinates



Sampling points and computing weights

Sampling points - I

- To sample points, we want to use a theorem by Shiffman and Zelditch

> The zeros of random sections are distributed according to the FS measure.

[Shiffman,Zelditch `98; Douglas,Karp,Lukic,Reinbacher `06; Braun,Brelidze,Douglas,Ovrut `08]

- More concretely, we look at the projectivized KC sections $\mathbb{P}^{r_i}$
  - We build random sections $S_j^{(i)} = \sum_{j=0}^{r_i} a_j^{(i)} s_j^{(i)}$ with iid Gaussian $a_j^{(i)} \sim \mathcal{N}(0,1)$
  - Here, iid means w.r.t. the orthogonal sections as defined by H previously
  - So what we need are the zeros of these random sections intersected with the locus that defines the CY

Sampling points - II

- However, there are two problems
  - The CY hypersurface is expressed in terms of x_a and not in terms of $s_j^{(i)}$
  - There are more sections than toric coordinates $\Rightarrow$ relations among sections
- Can write the CY hypersurface in terms of sections of the Kähler cone generators
- Then we have a hypersurface in a product of projective spaces $\mathcal{A} = \otimes_i \mathbb{P}^{r_i}$
- But $\sum_i r_i \gg 4 \quad \Rightarrow$ one hypersurface will not cut out the original threefold
- We need to add the relation among the sections (which are higher syzygies), and the degree of the sections plus the degree of the hypersurface **does not** sum up to $r_i + 1 \Rightarrow$ we get a **non-complete intersection CY**

Sampling points - III

- How do we find the relation among sections?
- General (commutative algebra) way (luckily Sage interfaces to Singular):
  - Start with the coordinate ring of the toric variety, plus all sections
  - Define the quotient ring of the coordinate ring (which is already a quotient ring) modulo the ideal defining the sections, i.e. $K[V] \, / \, \langle s_i - \prod x^{\langle v, w \rangle + c} \rangle$
  - Find the defining ideal of the quotient ring
  - Find the primary decomposition of the defining ideal
  - Find the elimination ideal of the quotient ring
  - Express the ideals obtained from the primary decomposition in terms of s_i

Sampling points - IV

- Alternative (linear algebra) way
  - Want relations of the form $\prod_{i,j} \left(s_j^{(i)}\right)^{\alpha_{i,j}} = \prod_{a,b} \left(s_b^{(a)}\right)^{\beta_{a,b}} \quad \Rightarrow \quad \prod_{i,j} \left(s_j^{(i)}\right)^{\gamma_{i,j}} = 1\,, \quad \alpha, \beta \in \mathbb{Z}_{\geq 0}\,, \ \gamma \in \mathbb{Z}$
  - Now $s_j^{(i)} = \prod_a x_a^{\langle v_a, w_j \rangle + c_i} := \prod_a x_a^{M_{Ai}}\,, \quad A = \{a, j\}$
  - Hence, we want $M \cdot \gamma = 0$, i.e. a primitive basis of the integer kernel of the integer matrix $M_{A,i}$
- The first way requires Groebner basis techniques, the latter solving a shortest vector problem in a lattice. We implement both in cymetric and the user can choose
- My feeling is that before this becomes a computational issue, you run into different complexity problems with Sage

Computing weights

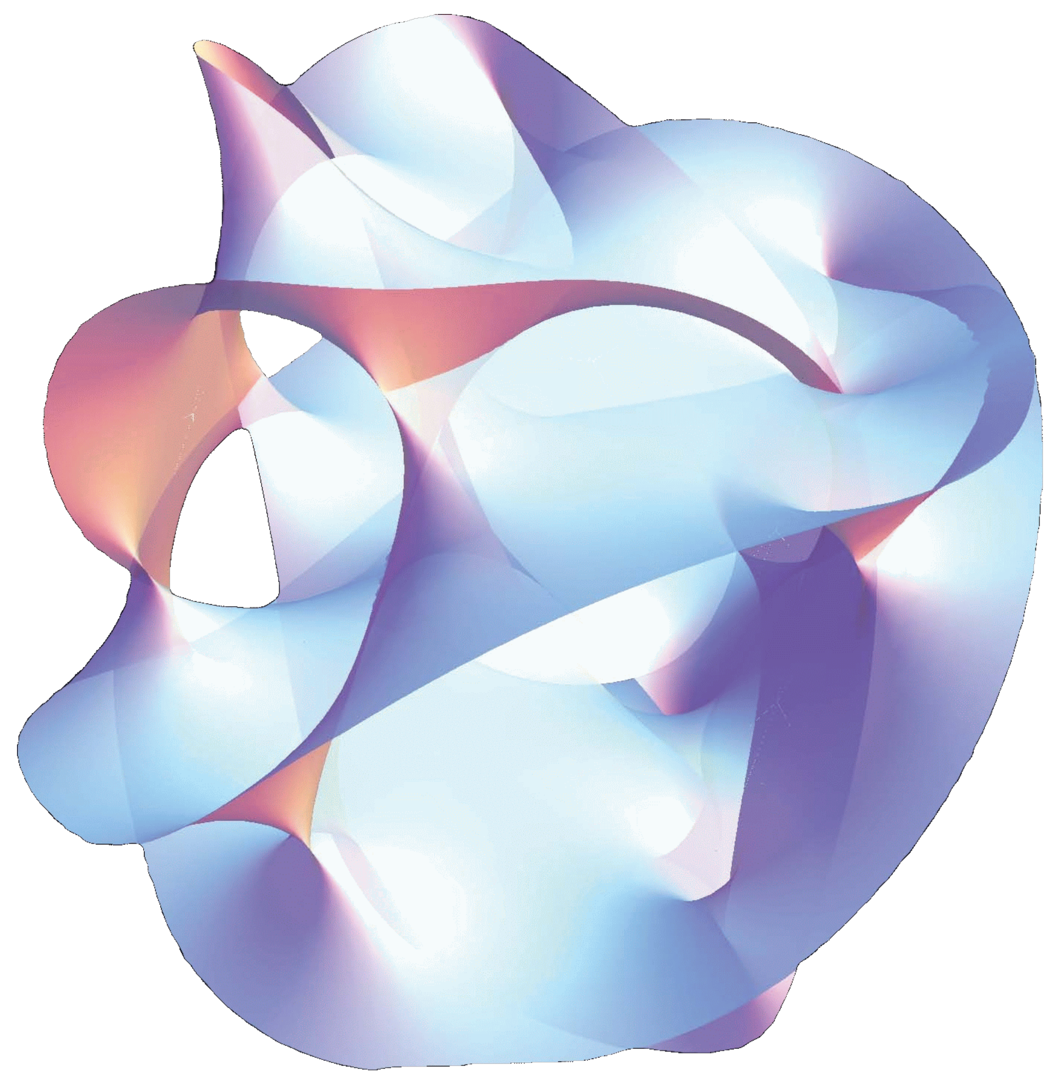
- Now that we have the equations defining the toric variety as a non-complete intersection in a product of projectivized KC generator section space, we can look at the simultaneous zeros of the non-CICY and the appropriate number (i.e. 3 for a CY threefold) of random sections in these spaces

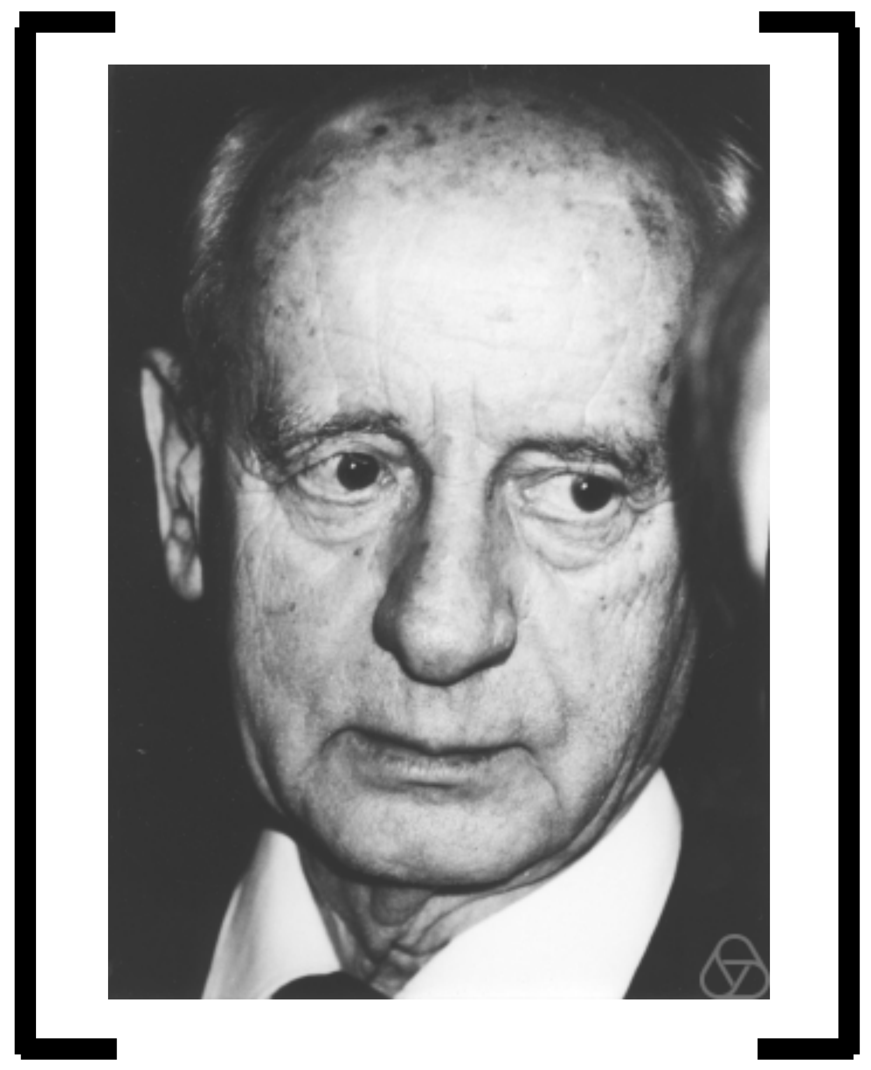
These zeros will be distributed according to the FS metric pulled back to the CY
But this pullback is the pullback of the canonical Kähler metric of the toric variety.

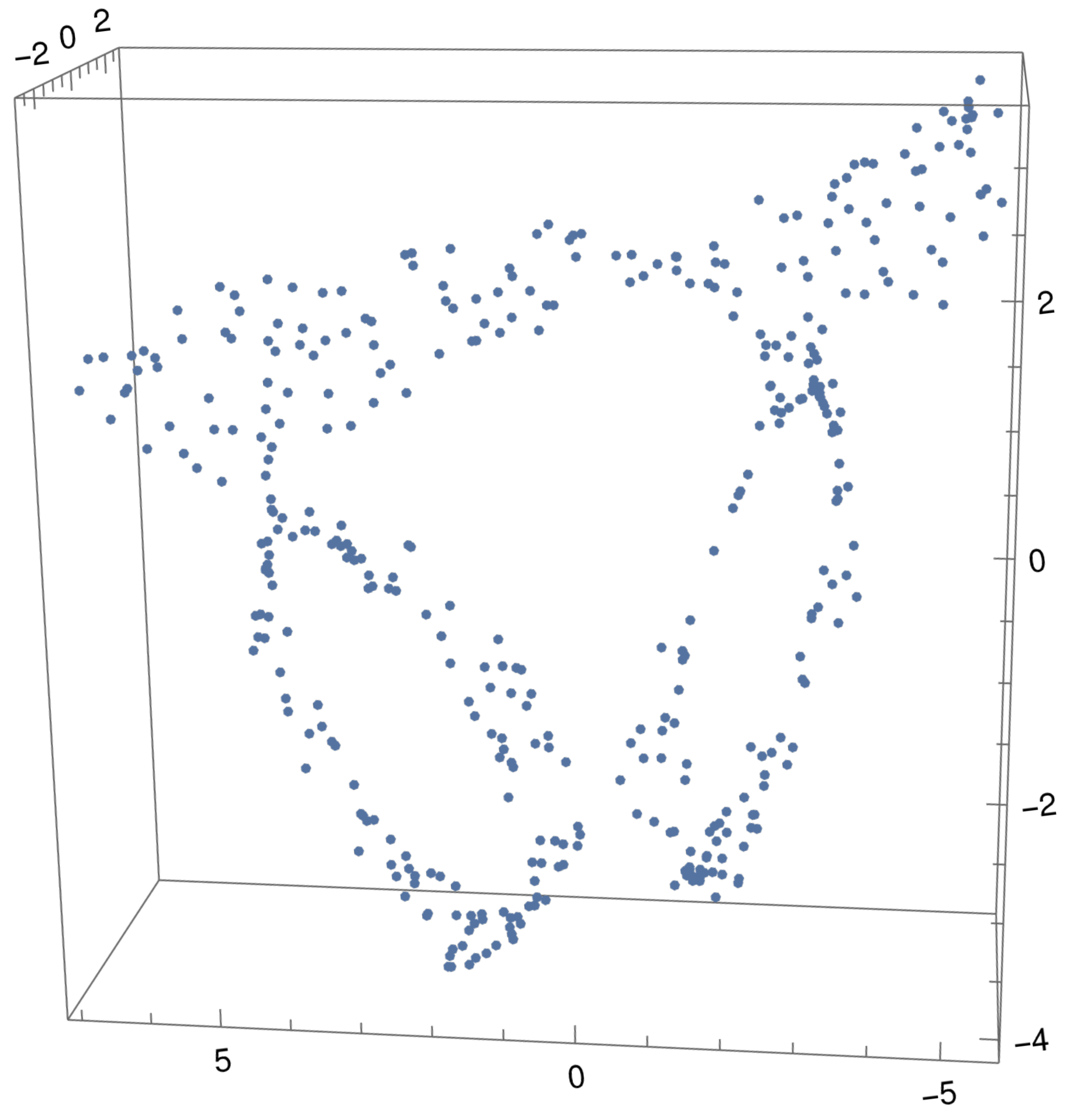
- So we can construct the weights using the pullbacks via the maps Φ_i

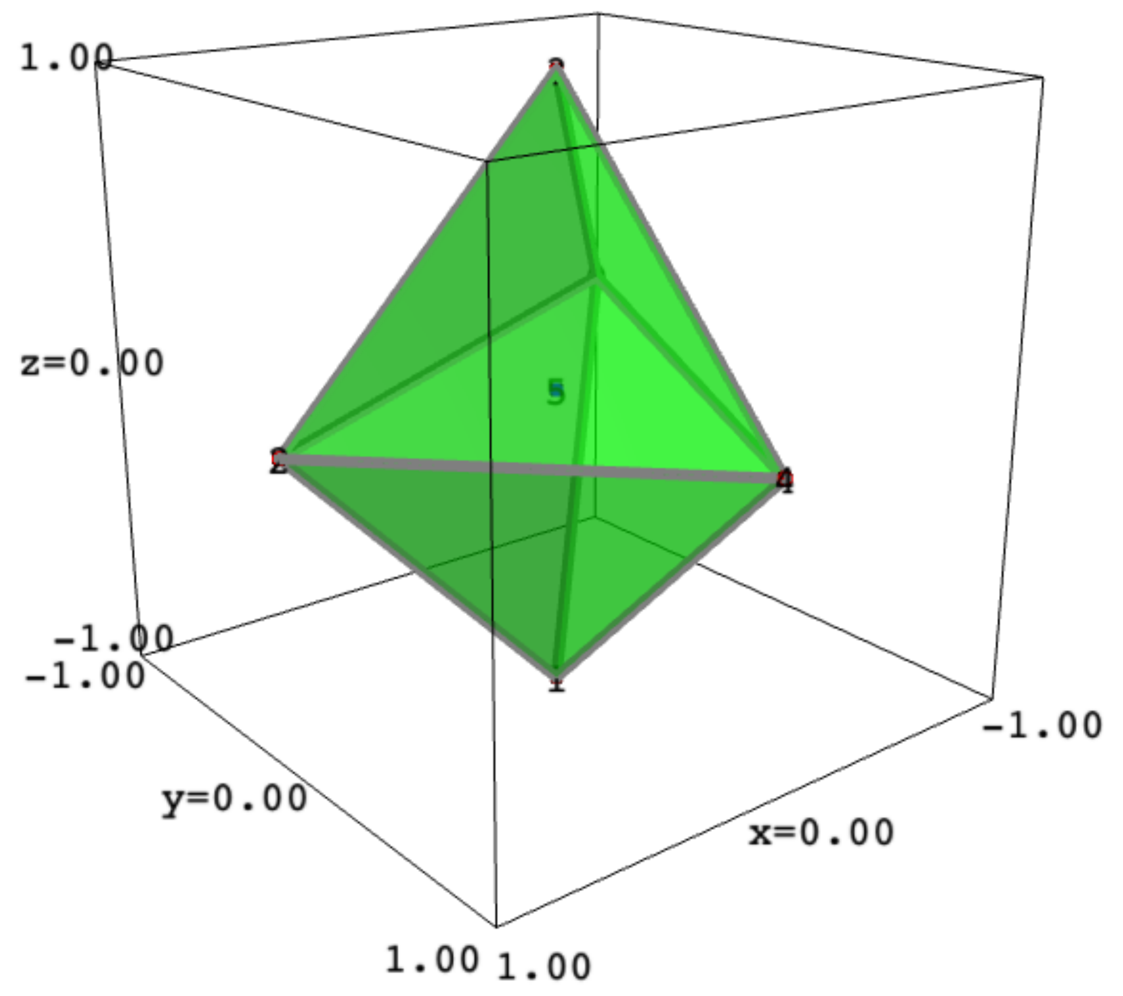
$$dA = \Phi_i^*(J) \wedge \Phi_j^*(J) \wedge \Phi_k^*(J)$$

where the (i,j,k) are the (not necessarily distinct) indices of the random KC generator sections in $\mathbb{P}^{r_i}$









Conclusions

Conclusions

- Provide a package to compute CY metrics in CICYs and toric varieties
- Introduce the "Phi NN" to learn CY metrics for a fixed Kähler class
- Explain how to generalize construction to toric case & CICYs
    - Construct sections of Kähler cone generator
    - Write CY as a non-complete intersection in the projectivized sections space
    - Construct weights using Shiffman-Zelditch and the pullbacks via the maps Φ_i from the toric coordinates into the sections

Thank You!