

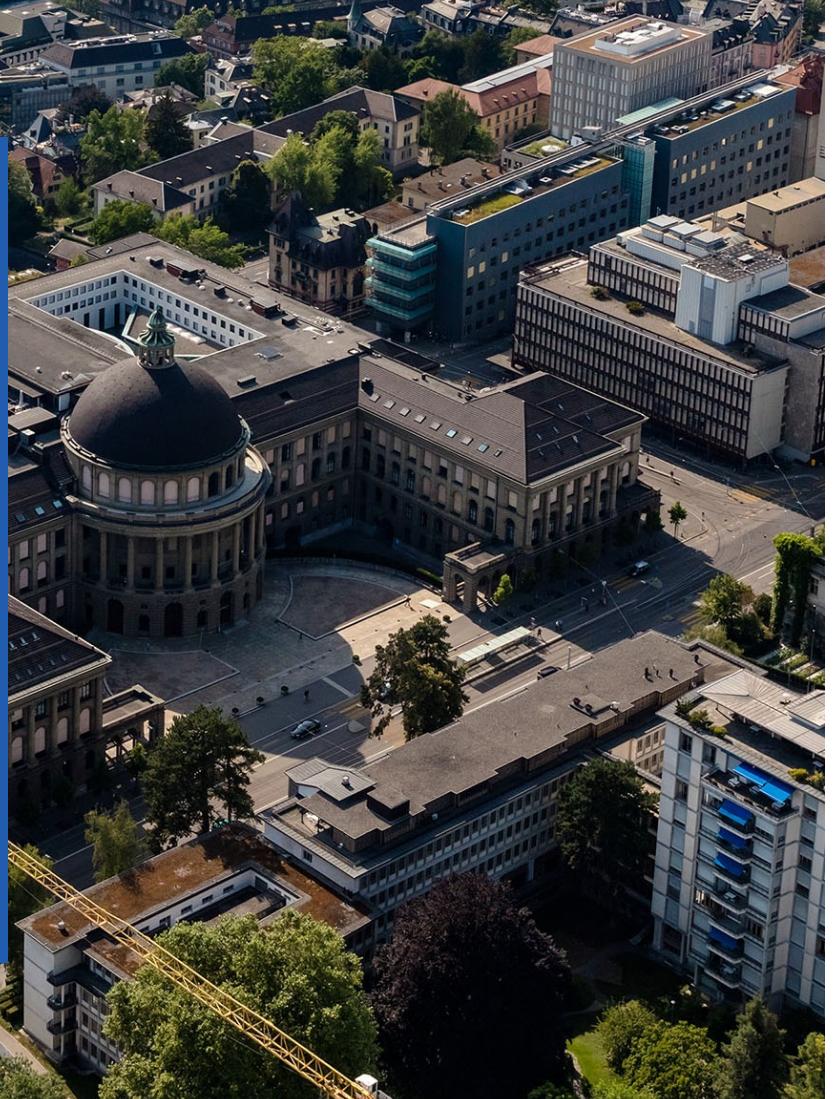


Data-Driven Identification of Spectral Submanifold-Based Reduced-Order Models

Bálint Kaszás, George Haller

Nonlinear Dynamics Group
Institute for Mechanical Systems

ETH Zurich

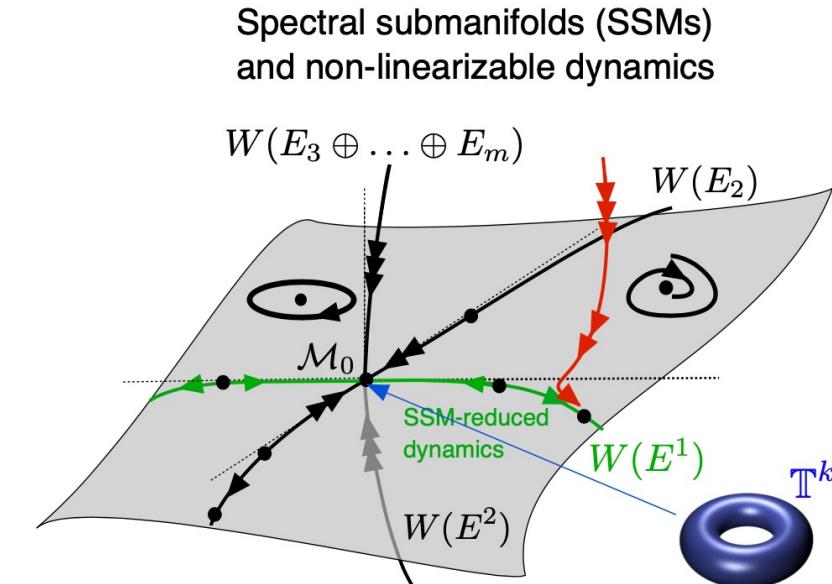
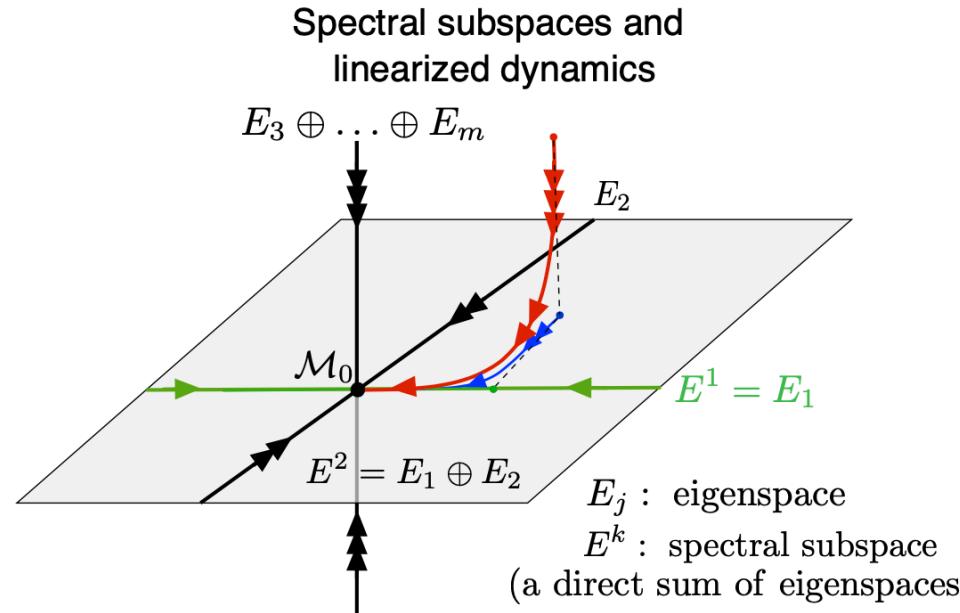


Spectral submanifolds

Reminder: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^n$
 $\text{Re } \lambda_1, \dots, \text{Re } \lambda_n < 0$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{f}_{\text{nl}}(\mathbf{x})$$

- 1.) Under **non-resonance** assumptions on the eigenvalues the spectral subspaces perturb into spectral submanifolds



- 2.) Slow SSMs act as attractors!

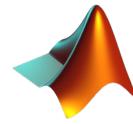
[1] M. Cenedese, J. Axås, B. Bäuerlein, K. Avila & G. Haller, **Data-Driven Modeling and Prediction of Non-Linearizable Dynamics via Spectral Submanifolds**, Nature Comm. (2022)

SSMLearn

SSMLearn



<https://github.com/haller-group>



Contributors 6



Joar
Axås



Mattia
Cenedese



B.K.
Xu



Zhenwei
Liu



Aihui
Liu



Leonardo
Bettini

Structure of SSMLearn (M. Cenedese, J. Axås, 2022)

Preprocessing

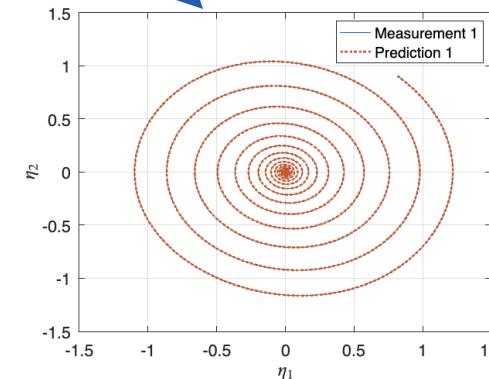
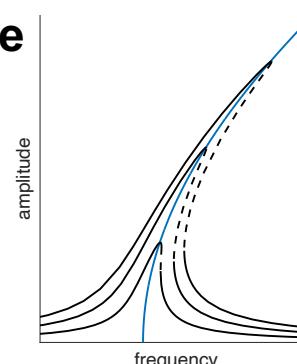
Geometry

Reduced dynamics

Normal form

Data-driven identification
of the **SSM** and the
reduced dynamics

The **outputs** are



Forced-response curves

Accurate
trajectory-predictions

Preprocessing

1. Ensure that the gathered **data approximates an SSM**
2. Find an anchor point or anchor trajectory with an attracting slow SSM
3. Center the data. The origin, $\mathbf{x} = \mathbf{0}$ is a fixed point
4. Infer the **dimension of the SSM (denoted d)** and ensure the data is close to it
For oscillatory systems, these can be deced by time-frequency analysis
The function `showSpectrogram()` constructs a spectrogram by a sliding-window fast Fourier transform
5. Compute the number of observables necessary **to embed the SSM [4]**
Use delay embedding if necessary
6. Sort the trajectories with a **train – test** split, but ideally **train-test-validation**. Arrange them in the Matlab **cell arrays**

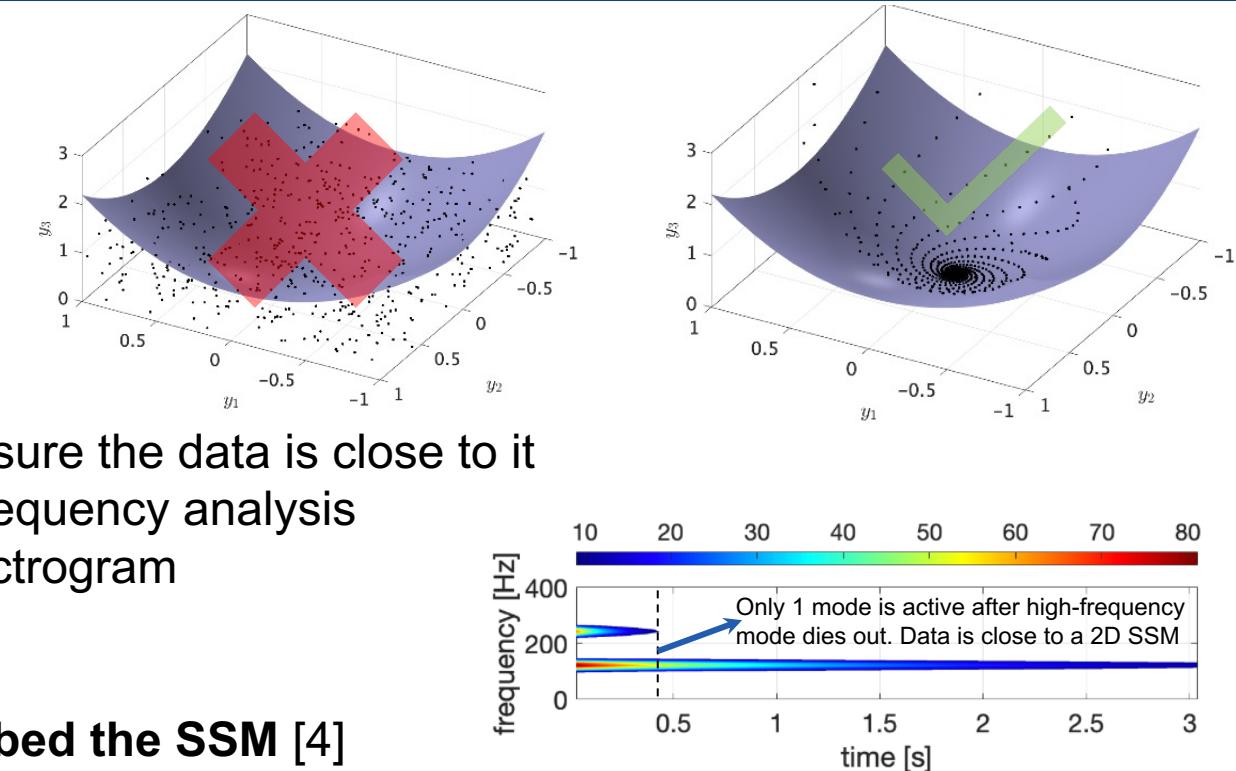
`xData`: raw data

`yData`: data in the observable space
(preprocessed and delay embedded)

```
>> yData
yData =
4x2 cell array
sampling times
{1x1201 double}
{1x1201 double}
{1x1201 double}
{1x1201 double}
```

p: number of observables
number of samples

{20x1201 double}
{20x1201 double}
{20x1201 double}
{20x1201 double}

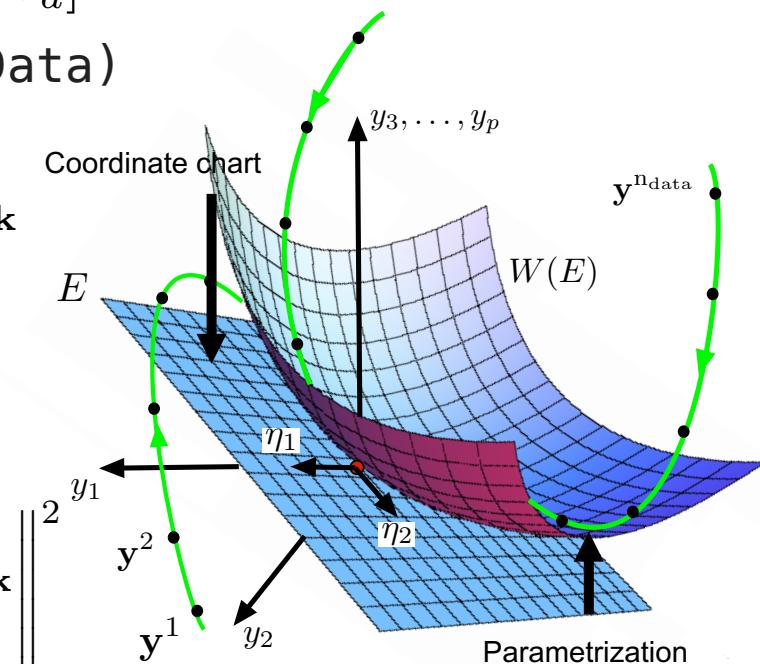


Geometry

1. Collect the observable data (from $yData$) as $\mathcal{Y} = [\mathbf{y}^1, \dots, \mathbf{y}^{n_{\text{data}}}]$, $\mathbf{y}^j \in \mathbb{R}^p$, $p > 2d$
2. The yet unknown orthonormal basis of the tangent space E is $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d]$
3. The reduced coordinates are given by $\boldsymbol{\eta} = \mathbf{V}^T \mathcal{Y}$ (collected in the cell etaData)
the coordinate chart is a projection
4. The SSM is parametrized as a graph over E $\mathbf{y} = \mathbf{w}(\boldsymbol{\eta}) = \mathbf{V}\boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{SSM}}} \mathbf{W}_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}}$
 where $\boldsymbol{\eta}^{\mathbf{k}} = \eta_1^{k_1} \cdots \eta_d^{k_d}$ are monomial terms
5. The basis vectors \mathbf{V} and the coefficients $\mathbf{W}_{\mathbf{k}}$ are determined

by **minimizing the loss function**

$$\mathcal{L}(\mathbf{V}, \mathbf{W}) = \left\| \mathcal{Y} - \mathbf{V}\boldsymbol{\eta} - \sum_{|\mathbf{k}|=2}^{n_{\text{SSM}}} \mathbf{W}_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}} \right\|^2 = \left\| \mathcal{Y} - \mathbf{V}\mathbf{V}^T \mathcal{Y} - \sum_{|\mathbf{k}|=2}^{n_{\text{SSM}}} \mathbf{W}_{\mathbf{k}} (\mathbf{V}^T \mathcal{Y})^{\mathbf{k}} \right\|^2$$



Two constraints are enforced:

- the basis is orthonormal $\mathbf{V}^T \mathbf{V} = \mathbf{I}_{d \times d}$
 - the coordinate chart is the inverse of the parametrization $\mathbf{V}^T \mathbf{y} = \mathbf{V}^T \mathbf{w}(\boldsymbol{\eta}) = \boldsymbol{\eta}$
- this is enforced by $\mathbf{V}^T \mathbf{W}_{\mathbf{k}} = \mathbf{0}$ for all \mathbf{k}

Geometry, optimization

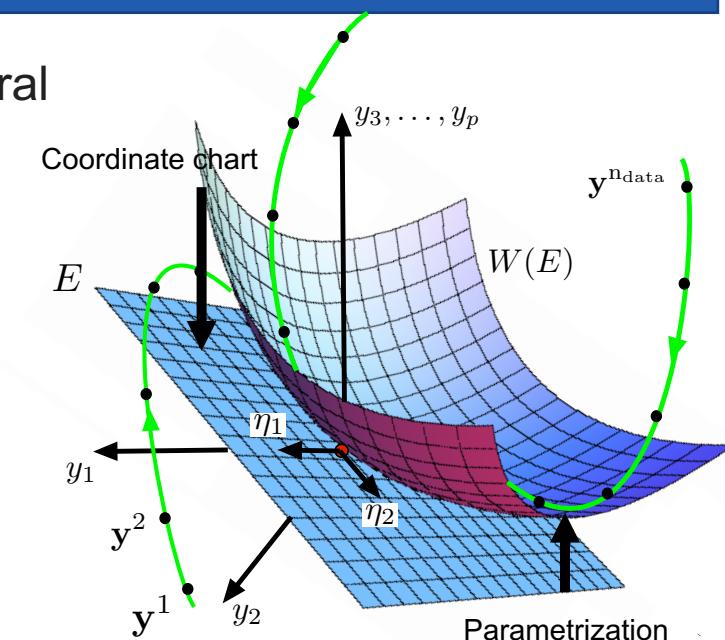
Minimizing $\mathcal{L}(\mathbf{V}, \mathbf{W})$ subject to the constraints is a **nonconvex problem** in general

1. Find an approximation for the tangent space spanned by $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d]$ using **Proper Orthogonal Decomposition (POD)** [5]. This is computed from a **Singular Value Decomposition (SVD)** of the data matrix \mathcal{Y}
The d modes containing the most variance [6] are collected in \mathbf{V}_0
2. Fixing \mathbf{V}_0 , compute the best-fitting nonlinear coefficients $\mathbf{W}_{0,k}$
this is a **constrained linear regression** problem with a closed-form solution
Ridge-type regularization can also be added
3. Use $(\mathbf{V}_0, \mathbf{W}_{0,k})$ as an initial guess to minimize $\mathcal{L}(\mathbf{V}, \mathbf{W})$
using the built-in function **IMGeometry()**

```
function [IMInfo, IMChart, IMPParam] = IMGeometry(yData, SSMDim, M, varargin)
    %
    % Minimization process
    z_opt = fmincon(fun,z_0,Aine,bine,Aequ,bequ,[],[],nonlincon, ...
        opt_options);
    %
    % Final output
    V_e = reshape(z_opt(1:n*k),n,k);
    H = reshape(z_opt(1+n*k:end),n,phi_dim);
    IMPParam = @(q) V_e * q + H * phi(q);
```

We also **compute the derivatives of the loss function** $\frac{\partial \mathcal{L}}{\partial \mathbf{V}}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ to speed up the iterations
the default setting of `fmincon` [7] is used, which is the **interior-point algorithm** [8]

4. (Optional) Cross validation: adjust the parameters p, n^{SSM} based on the **validation error**



Reduced dynamics, without normal form

1. Project to the tangent space to get the **reduced coordinates** $\boldsymbol{\eta} = \mathbf{V}^T \mathbf{y}$ (etaData)
 2. Approximate the **time-derivatives**, e.g., as $\dot{\boldsymbol{\eta}}(t_j) \approx \frac{\boldsymbol{\eta}(t_{j+1}) - \boldsymbol{\eta}(t_j)}{t_{j+1} - t_j}$
SSMLearn uses an **order-8 central difference** approximation [9]
 3. Approximate the reduced dynamics as a **polynomial of the reduced coordinates** $\dot{\boldsymbol{\eta}} = \mathbf{r}(\boldsymbol{\eta}) = \mathbf{R}_0 \boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}}$
by constructing a loss function based on the **observed time-derivatives**
- $$L_0(\mathbf{R}_0, \mathbf{R}_{\mathbf{k}}) = \left\| \dot{\boldsymbol{\eta}} - \mathbf{R}_0 \boldsymbol{\eta} - \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}} \right\|^2$$
4. (Optional) Additional constraints, such as fixed points $\mathbf{r}(\boldsymbol{\eta}^*) = \mathbf{0}$, and ridge-type regularizing terms can be added to the loss function
 5. Minimize $L_0(\mathbf{R}_0, \mathbf{R}_{\mathbf{k}})$ via (un-)constrained **linear regression** (closed form solution is available)
 6. If we don't want a normal-form style dynamics, we can stop here and optionally adjust the parameter of the approximation order n_{model} based on cross-validation with the validation set

Reduced dynamics, with normal form

If **normal form-style reduced dynamics** are desired

Only for **oscillatory dynamics with complex eigenvalues!**

7. Define the yet unknown maps $z = T^{-1}(\eta)$ and $\eta = T(z)$ transforming the reduced coordinates, such that the dynamics are in the **extended normal form [1]**

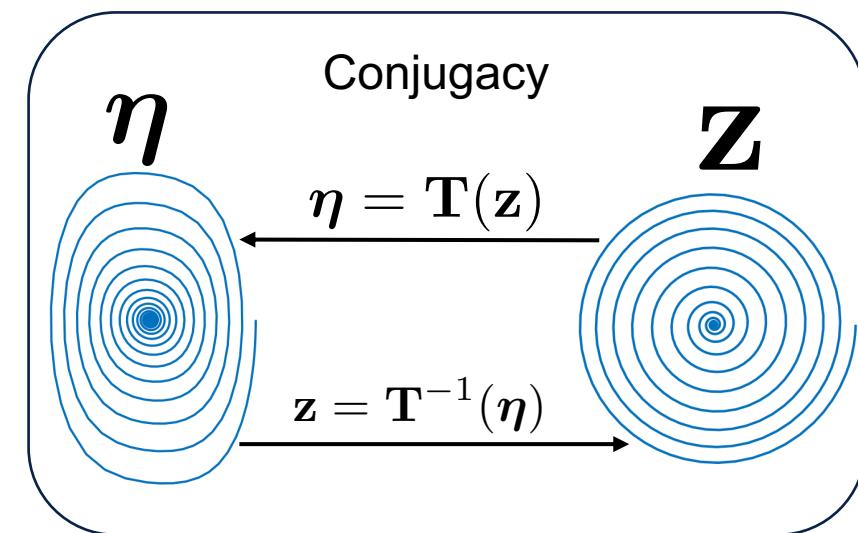
$$\dot{z} = R_0 z + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} R_k^{\text{norm}} z^k$$

8. Determine the sparsity structure, i.e., which R_k^{norm} coefficients are nonzero. This is based on eigenvalue-resonances [10] of R_0

9. Define the **loss function**, depending on R_k^{norm} and the coefficients of the near-identity map

$$L_1(R_k, T_k^{\text{inv}}) = \left\| \dot{z} - R_0 z - \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} R_k^{\text{norm}} z^k \right\|^2$$

```
function [RDInfo,R,iT,N,T] = IMDynamicsFlow(etaData,varargin)
    ...
    % Run unconstrained optimization
    z = fminunc(fun,mapsInfo.IC_opt,opt_options_fm); % Conj. Error is
    % Final output
    N_info = mapsInfo.N; iT_info = mapsInfo.iT;
    ...
```



10. Minimize the loss function using the built-in function `IMDynamicsFlow()`. The **initial guess is zero** by default and the **derivatives of the loss function** $\frac{\partial L_1}{\partial R^{\text{norm}}}, \frac{\partial L_1}{\partial T^{\text{inv}}}$ are also computed to speed up the iterations

11. Compute the coefficients of the transformation $\eta = z + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} T_k z^k$ by **linear regression**

12. (Optional) Cross validation: adjust the parameter of the approximation order n_{model} using the validation set

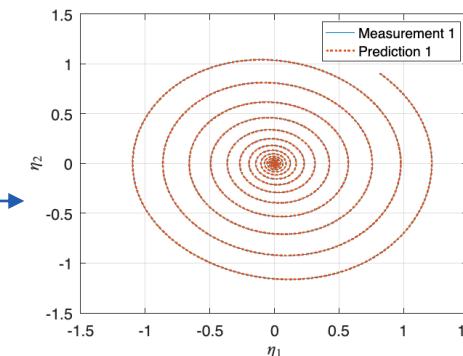
Postprocessing

1. The normal form-style dynamics can be transformed to polar coordinates as and **further analysis** and **validation** can be carried out

$$\mathbf{z} = \begin{pmatrix} z \\ \bar{z} \end{pmatrix} = \begin{pmatrix} \rho e^{i\theta} \\ \rho e^{-i\theta} \end{pmatrix}$$

E.g, using the `advect()` function

$$\begin{aligned}\dot{\rho} &= -0.0500\rho + 0.0778\rho^3 & \text{Prediction on the test set} \\ \dot{\theta} &= 0.9988 + 0.0084\rho^2\end{aligned}$$

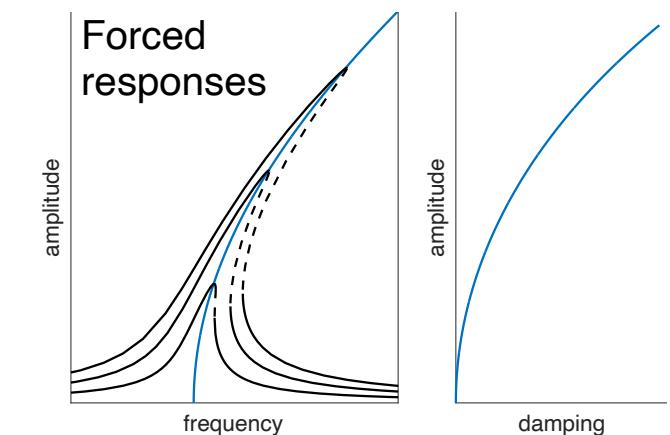


2. Modeling the effects of (quasi-)periodic forcing $\mathbf{g}(t)$

$$\dot{\mathbf{z}} = \mathbf{R}_0 \mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}}^{\text{norm}} \mathbf{z}^{\mathbf{k}} + \mathbf{V}^T \mathbf{g}(t)$$

If the forcing is **known**, it can simply be projected to the **tangent space**

Otherwise, it can be obtained via calibration. If the **response** is known for a single frequency-amplitude pair, then we can calculate how the forcing enters the normal form



Structs encoding the geometry and the dynamics

>> IMInfo.parametrization **Invariant Manifold Info**

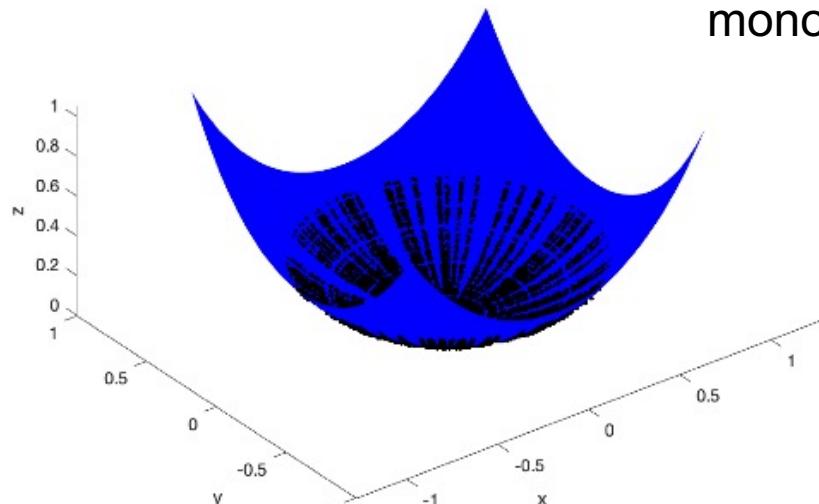
```
ans =
struct with fields:
    map: @(q)V_e*q+H*phi(q)
    polynomialOrder: 4
    dimension: 2
    tangentSpaceAtOrigin: [3x2 double]
    nonlinearCoefficients: [3x12 double]
    phi: [function_handle]
    exponents: [12x2 double]
    l: 0
    c1: 0
    c2: 0
```

$$y = w(\eta) = V\eta + \sum_{|\mathbf{k}|=2}^{n_{SSM}} W_k \eta^k$$

V

W_k

$\eta^k = \eta_1^{k_1} \cdots \eta_d^{k_d}$
monomial terms



>> RDInfo **Reduced Dynamics Info**

```
RDInfo =
struct with fields:
    reducedDynamics: [1x1 struct]
    inverseTransformation: [1x1 struct]
    conjugateDynamics: [1x1 struct]
    transformation: [1x1 struct]
    conjugacyStyle: 'normalform'
    dynamicsType: 'flow'
    eigenvaluesLinPartFlow: [2x1 double]
    eigenvectorsLinPart: [2x2 double]
```

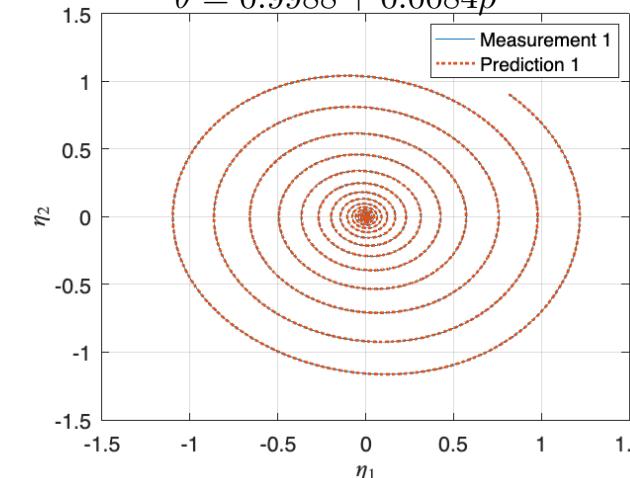
$$\dot{\eta} = r(\eta) = R_0\eta + \sum_{|\mathbf{k}|=2}^{n_{model}} R_k \eta^k$$

$$z = \eta + \sum_{|\mathbf{k}|=2}^{n_{model}} T_k^{-1} \eta^k$$

$$\dot{z} = R_0 z + \sum_{|\mathbf{k}|=2}^{n_{model}} R_k^{\text{norm}} z^k$$

$$\dot{\eta} = z + \sum_{|\mathbf{k}|=2}^{n_{model}} T_k z^k$$

$$\dot{\rho} = -0.0500\rho + 0.0778\rho^3$$

$$\dot{\theta} = 0.9988 + 0.0084\rho^2$$


Additional options

1. If the tangent space is known (e.g. PCA/POD), or if the reduced coordinates are available (physics / “domain knowledge”) We can stop at **Step 2** on **Slide 6**: no need for iterative optimization
2. Instead of a one-step optimization (Steps 7 – 10 on Slide 8), normalization of the dynamics can be **done analytically** using an interface with [SSMTool](#). This approach is called [fastSSM](#) [11] and greatly reduces the computational costs
3. The SSM is smooth in parameters of the underlying system, therefore **parameter-dependent** models can be constructed. In this case, the coefficients can be parameter-dependent

$$\dot{\eta} = \mathbf{r}(\eta, \kappa) = \mathbf{R}_0(\kappa)\eta + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}(\kappa)_k \eta^k$$

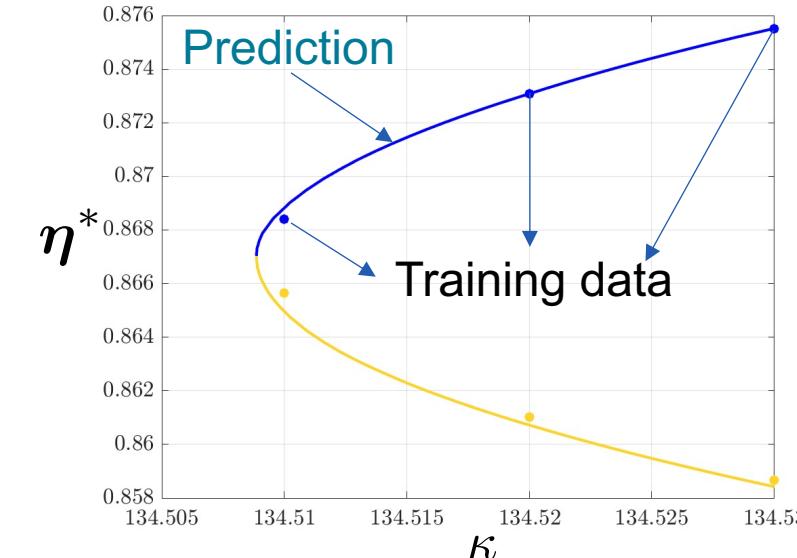
Constraints, such as fixed points, $\mathbf{r}(\eta^*, \kappa^*) = \mathbf{0}$ can be added

```
>> yData
yData =
3x3 cell array
```

{1x5001 double}	{20x5001 double}	{[0.0100]}
{1x3001 double}	{20x3001 double}	{[0.0100]}
{1x3001 double}	{20x3001 double}	{[0.0200]}

Third column of yData: parameter value

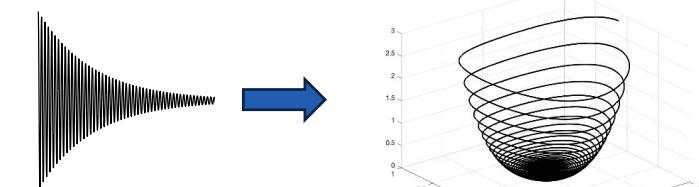
Example: Fixed points of plane Couette flow [12]



Summary: A typical use case

1. Preprocessing and embedding

```
[yData, opts_embed] = coordinatesEmbedding(xData, SSMDim, 'OverEmbedding', overEmbed);
```



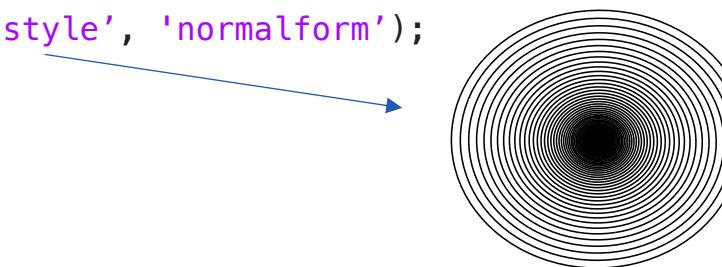
2. Fitting the SSM geometry

```
IMInfo = IMGeometry(yData(indTrain,:), SSMDim, SSMOrder);
plotSSMWithTrajectories(IMInfo, [1,2,3], yData);
```



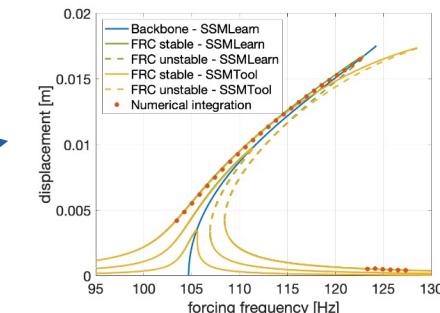
3. Fitting the reduced dynamics

```
etaData = projectTrajectories(IMInfo, yData);
RDInfo = IMDynamicsFlow(uData(indTrain,:), 'R_PolyOrd', ROMOrder, 'style', 'normalform');
```



4. Evaluation on test set

```
[yRec, uRec, wRec] = advect(IMInfo, RDInfo, yData);
errors = computeTrajectoryErrors(yRec, yData);
```



5. Postprocessing: Predict forced response

```
fRed = calibrateFRC(IMInfo, RDInfo, yCal, Omega);
[IMInfoF, RDInfoF] = forcedSSMROM(IMInfo, RDInfo, 'nForcingFrequencies', 1);
FRC = analyticalFRC(IMInfoF, RDInfoF, fRed, y0observable)
```

References

- [1] M. Cenedese, J. Axås, B. Bäuerlein, K. Avila, G. Haller, **Data-driven modeling and prediction of non-linearizable dynamics via spectral submanifolds**, *Nature Comm.* **13**, 872 (2022).
- [2] G. Haller, S. Ponsioen, **Nonlinear normal modes and spectral submanifolds: existence, uniqueness and use in model reduction**. *Nonlinear Dyn* **86**, 1493–1534 (2016).
- [3] P. Cabré, E. Fontich, R. de la Llave, **The parametrization method for invariant manifolds I: manifolds associated to non-resonant spectral subspaces**. *Indiana Univ. Mathe. J.* **52**, 283–328 (2003).
- [4] Takens, F. **Detecting strange attractors in turbulence**. In: D. Rand and L. Young, (editors), *Dynamical Systems and Turbulence*, Warwick, 366–381 (1980).
- [5] P. Holmes, J. L. Lumley, G. Berkooz, C. W. Rowley, **Turbulence, coherent structures, dynamical systems and symmetry**, *Cambridge Monographs on Mechanics*, (Cambridge University Press, 2012).
- [6] T. Hastie, R. Tibshirani, J. Friedman, **The elements of statistical learning: data mining, inference, and prediction**, *Springer Series in Statistics* (Springer New York, 2009).
- [7] <https://ch.mathworks.com/help/optim/ug/fmincon.html>
- [8] R. H. Byrd, M. E. Hribar, J Nocedal. **An interior point algorithm for large-scale nonlinear programming**, *SIAM J. Optim.*, **9**, 877–900 (1999).
- [9] B. Fornberg, **Generation of finite difference formulas on arbitrarily spaced grids**, *Math. Comput.* **51**, 699–706 (1988).
- [10] J. Guckenheimer, P. Holmes, **Nonlinear oscillations, dynamical systems, and bifurcations of vector fields**, *Applied Mathematical Sciences*, Vol. 42 (Springer, Berlin, 2013).
- [11] J. Axås, M. Cenedese, G. Haller, **Fast data-driven model reduction for nonlinear dynamical systems** *Nonlinear Dyn* **111**, 7941–7957 (2023).
- [12] B. Kaszás, M. Cenedese, G. Haller, **Dynamics-based machine learning of transitions in Couette flow** *Phys. Rev. Fluids* **7**, L082402 (2022).