ETH zürich

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

Bálint Kaszás, George Haller

Nonlinear Dynamics Group
Institute for Mechanical Systems

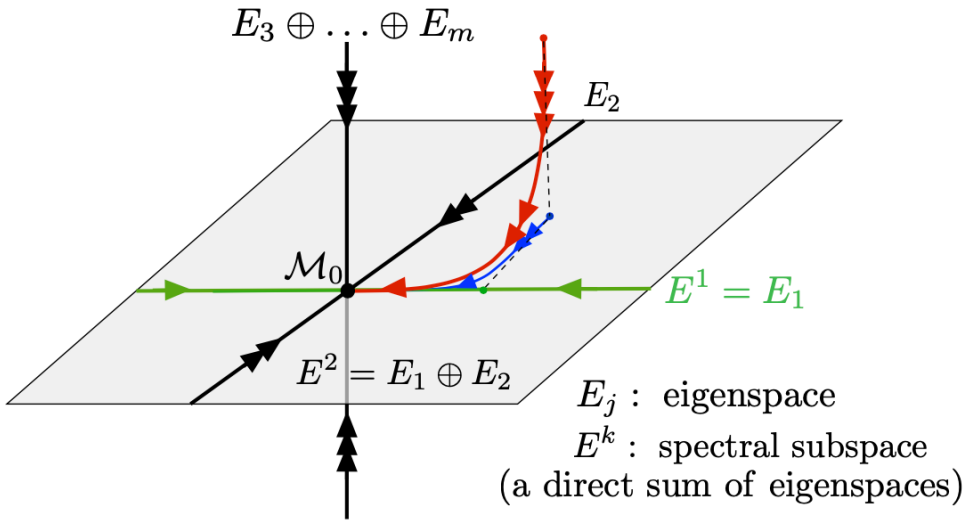
ETH Zurich

Spectral submanifolds

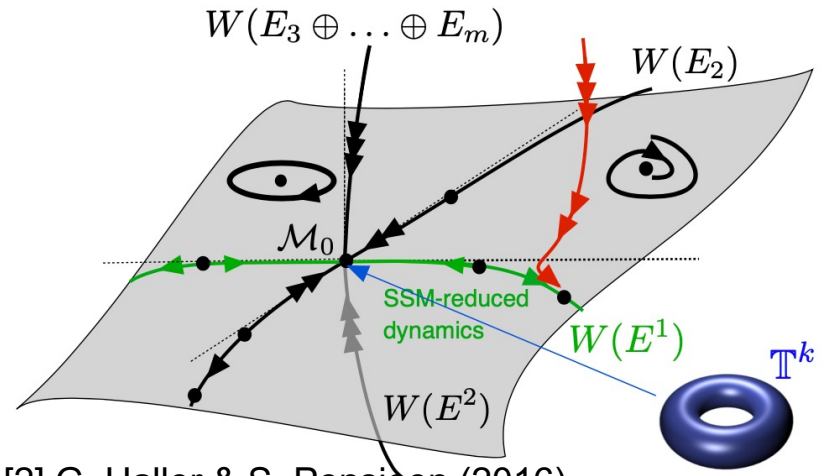
Reminder: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^n$, $\operatorname{Re}\lambda_1, \ldots, \operatorname{Re}\lambda_n < 0$ $\qquad \dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{f}_{\mathrm{nl}}(\mathbf{x})$

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

Spectral subspaces and linearized dynamics



Spectral submanifolds (SSMs) and non-linearizable dynamics



[2] G. Haller & S. Ponsioen (2016)
[3] Cabré et al. (2003)

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., Zhenwei Xu, Aihui Liu, Leonardo Bettini

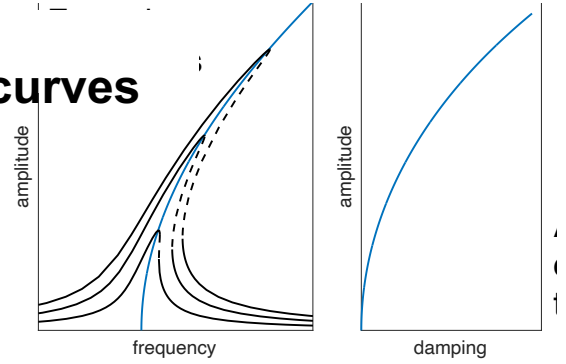
Structure of SSMLearn (M. Cenedese, J. Axås, 2022)

- Preprocessing
- Geometry
- Reduced dynamics
- Normal form

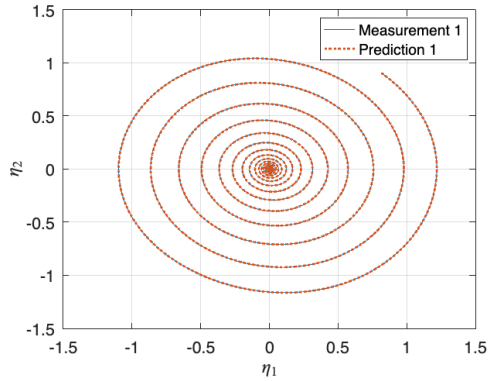
Data-driven identification of the **SSM** and the **reduced dynamics**

The **outputs** are

Forced response and damping curves

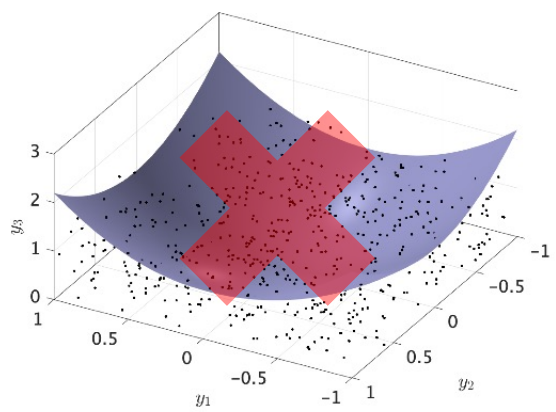


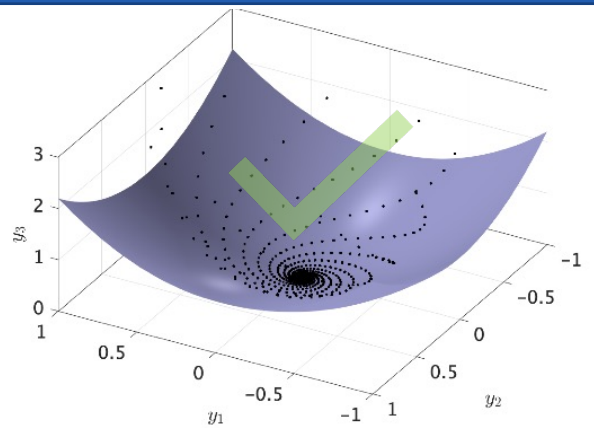
Accurate trajectory-predictions



Preprocessing

1. Ensure that the gathered **data approximates an SSM**

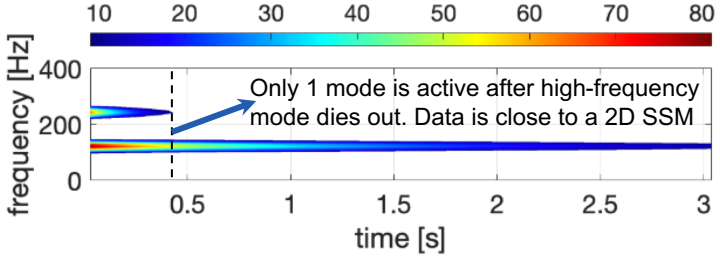




2. Find an anchor point or anchor trajectory with an attracting slow SSM

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, this is **twice the number of persistent modes**
   We use the function `showSpectrogram()` to construct a spctrogram by a sliding-window fast Fourier transform. The function `SSM_starttime()` automatically **finds the optimal truncation time**



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-validation-test.** Arrange them in the `Matlab cell arrays`
   - `xData:` raw data
   - `yData:` data in the observable space (preprocessed and delay embedded)

```
>> yData

yData =

  4×2 cell array
    sampling times
    {1×1201 double}    {20×1201 double}
    {1×1201 double}    {20×1201 double}
    {1×1201 double}    {20×1201 double}
```

p: number of observables

number of samples

Geometry

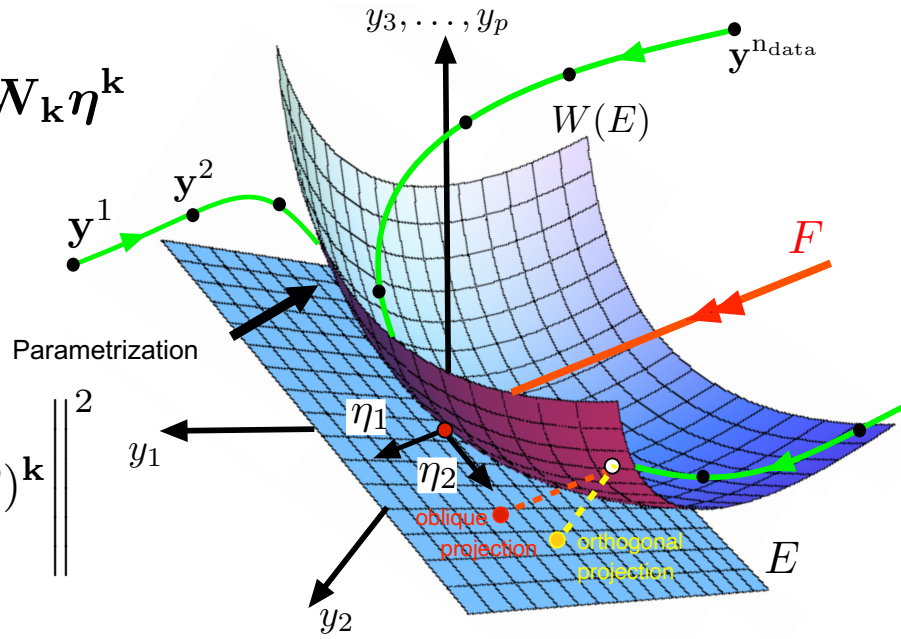
1. Collect the observable data (from `yData`) as $\boldsymbol{\mathcal{Y}} = [\mathbf{y}^1, \ldots, \mathbf{y}^{\mathrm{n_{data}}}], \quad \mathbf{y}^j \in \mathbb{R}^p, \quad p > 2d$
2. The yet unknown orthonormal basis of the tangent space E is $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ and the fast subspace is F
3. The reduced coordinates are given by $\boldsymbol{\eta} = \mathbf{V}^{\mathrm{T}}\boldsymbol{\mathcal{Y}}$ (collected in the cell array `etaData`)

   the **coordinate chart is a projection**
4. The SSM is parametrized as a graph over E $\mathbf{y} = \mathbf{w}(\boldsymbol{\eta}) = \mathbf{V}\boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\mathrm{SSM}}} \mathbf{W}_{\mathbf{k}}\boldsymbol{\eta}^{\mathbf{k}}$

   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**

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



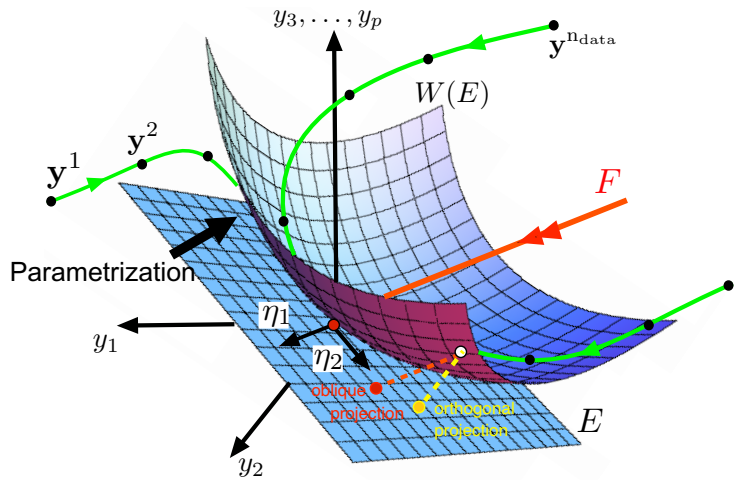
Two constraints are enforced:

- 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 $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ 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 $\mathbf{W}_{0,\mathbf{k}}$
   this is a **linear regression** problem with a closed-form solution
   Ridge-type regularization can also be added
3. Use $(\mathbf{V}_0, \mathbf{W}_{0,\mathbf{k}})$ as an initial guess to minimize $\mathcal{L}(\mathbf{V}, \mathbf{W})$ using the built-in function `IMGeometry()`

```
function [IMInfo, IMChart, IMParam] = 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);
IMParam = @(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) Adjust the parameters p, n^{SSM} to minimize the **validation error**

Simplification: Precomputing the tangent space

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. The loss function becomes

$$\mathcal{L}(\mathbf{W}) = \left\| \boldsymbol{\mathcal{Y}} - \mathbf{V}\boldsymbol{\eta} - \sum_{|\mathbf{k}|=2}^{n_{\mathrm{SSM}}} \mathbf{W}_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}} \right\|^2$$ and $\mathbf{W}_{\mathbf{k}}$ are obtained from **linear regression**

2. Specifically, it was shown by Axås, Haller (2023) [9] that if the observables are obtained by delay embedding with delay time Δt and the **eigenvalues** λ_j corresponding to the d-dimensional spectral subspace are known, the tangent space is given by the columns of the Vandermonde matrix $\mathbf{V}^{\mathbf{d}}$. This is computed using the `delayTangentSpace()` function

$$\mathbf{V}^{\mathbf{d}} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ e^{\lambda_1 \Delta t} & e^{\lambda_2 \Delta t} & \dots & e^{\lambda_d \Delta t} \\ \vdots & \vdots & \ddots & \vdots \\ e^{(p-1)\lambda_1 \Delta t} & e^{(p-1)\lambda_2 \Delta t} & \dots & e^{(p-1)\lambda_d \Delta t} \end{pmatrix}$$

3. Alternatively, for a 2D oscillatory SSM, we can find the **oblique projection** $\mathbf{P} : \mathbb{R}^p \to \mathbb{R}^p$ along the fast direction by **minimizing the oscillations of the backbone** curve [10] using the function `obliqueProjection()`. The reduced coordinates are then given as $\boldsymbol{\eta} = \mathbf{V}^{\mathrm{T}} \mathbf{P} \boldsymbol{\mathcal{Y}}$

Reduced dynamics, without normal form

1. Project to the tangent space to get the **reduced coordinates** $\boldsymbol{\eta} = \mathbf{V}^{\mathrm{T}}\boldsymbol{\mathcal{Y}}$ **or** $\boldsymbol{\eta} = \mathbf{V}^{\mathrm{T}}\mathbf{P}\boldsymbol{\mathcal{Y}}$ `(etaData)`
2. Approximate the **time-derivatives**, e.g., as $\dot{\boldsymbol{\eta}}(t_j) \approx \dfrac{\boldsymbol{\eta}(t_{j+1}) - \boldsymbol{\eta}(t_j)}{t_{j+1} - t_j}$
   SSMLearn uses an **order-8 central difference** approximation [11]
3. Approximate the reduced dynamics as a **polynomial of the reduced coordinates** $\dot{\boldsymbol{\eta}} = \mathbf{r}(\boldsymbol{\eta}) = \mathbf{R}_0\boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\mathrm{model}}} \mathbf{R}_{\mathbf{k}}\boldsymbol{\eta}^{\mathbf{k}}$
   by constructing a loss function based on the **observed time-derivatives**

$$L_0(\mathbf{R}_0, \mathbf{R}_{\mathbf{k}}) = \left\| \dot{\boldsymbol{\eta}} - \mathbf{R}_0\boldsymbol{\eta} - \sum_{|\mathbf{k}|=2}^{n_{\mathrm{model}}} \mathbf{R}_{\mathbf{k}}\boldsymbol{\eta}^{\mathbf{k}} \right\|^2$$

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 the normalized mean trajectory error (NMTE) on validation set

Reduced dynamics, with normal form

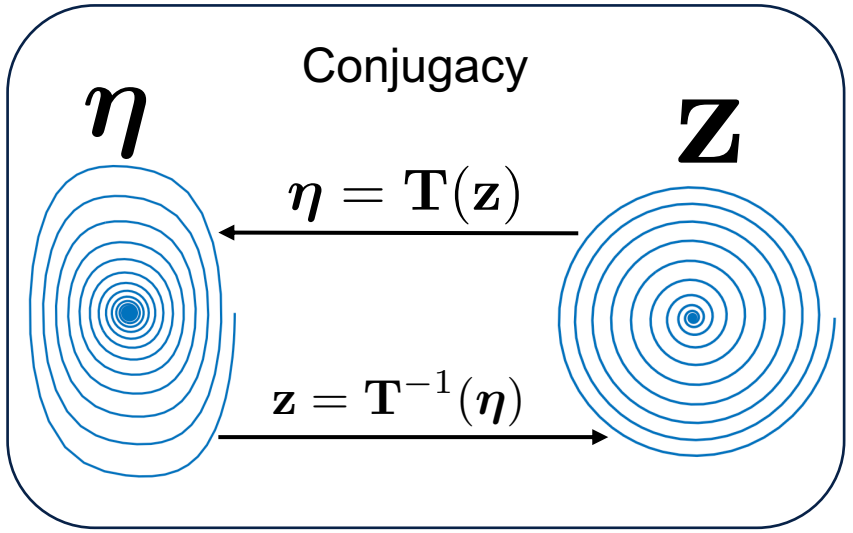
If **normal form-style reduced dynamics** are desired
Only for **oscillatory dynamics with complex eigenvalues!**

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

$$\dot{\mathbf{z}} = \mathbf{R}_0\mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}}^{\text{norm}}\mathbf{z}^{\mathbf{k}}$$

8. Determine the sparsity structure, i.e., which $\mathbf{R}_{\mathbf{k}}^{\text{norm}}$ coefficients are nonzero. This is based on eigenvalue-resonances [12] of $\mathbf{R}_0$
9. Define the **loss function**, depending on $\mathbf{R}_{\mathbf{k}}^{\text{norm}}$ and the coefficients of the near-identity map $\mathbf{z} = \boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{T}_{\mathbf{k}}^{\text{inv}}\boldsymbol{\eta}^{\mathbf{k}}$

$$L_1(\mathbf{R}_{\mathbf{k}}, \mathbf{T}_{\mathbf{k}}^{\text{inv}}) = \left\| \dot{\mathbf{z}} - \mathbf{R}_0\mathbf{z} - \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}}^{\text{norm}}\mathbf{z}^{\mathbf{k}} \right\|^2$$




```
function [RDInfo,R,iT,N,T] = IMDynamicsFlow(etaData,varargin)
⋮
% Run unconstrained optimization
z = fminunc(fun,mapsInfo.IC_opt,opt_options_fm); % Conj. Errror 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** $\frac{\partial L_1}{\partial \mathbf{R}^{\text{norm}}}, \frac{\partial L_1}{\partial \mathbf{T}^{\text{inv}}}$ are also computed to speed up the iterations
11. Compute the coefficients of the transformation $\boldsymbol{\eta} = \mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{T}_{\mathbf{k}}\mathbf{z}^{\mathbf{k}}$ by **linear regression**
12. (Optional) Cross validation: adjust the parameter of the approximation order n_{model} using the validation set

Postprocessing

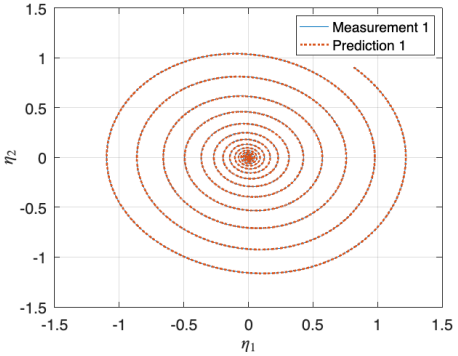
1. The normal form-style dynamics can be transformed to polar coordinates as and **further analysis** and **validation** can be carried out

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

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

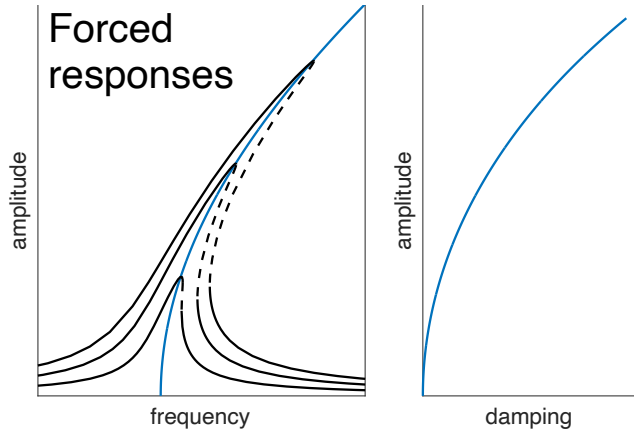
$$\dot{\rho} = -0.0500\rho + 0.0778\rho^3$$
$$\dot{\theta} = 0.9988 + 0.0084\rho^2$$

Prediction on the **test set**



2. Modeling the effects of (small) forcing $\epsilon \mathbf{g}(\mathbf{x}, t)$. Note: forcing is often **periodic or quasi-periodic**

$$\dot{\mathbf{z}} = \mathbf{R}_0 \mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}}^{\text{norm}} \mathbf{z}^{\mathbf{k}} + \epsilon \mathbf{V}^T \mathbf{P} \mathbf{g}(\mathbf{0}, t)$$



If the forcing is **known**, it can simply be projected to the **tangent space** (leading-order apporximation)

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

Invariant Manifold Info

```
>> IMInfo.parametrization

ans =

  struct with fields:

                    map: @(q)V_e*q+H*phi(q)
        polynomialOrder: 4
              dimension: 2
    tangentSpaceAtOrigin: [3×2 double]
    nonlinearCoefficients: [3×12 double]
                    phi: [function_handle]
              exponents: [12×2 double]
                      l: 0
                     c1: 0
                     c2: 0
```

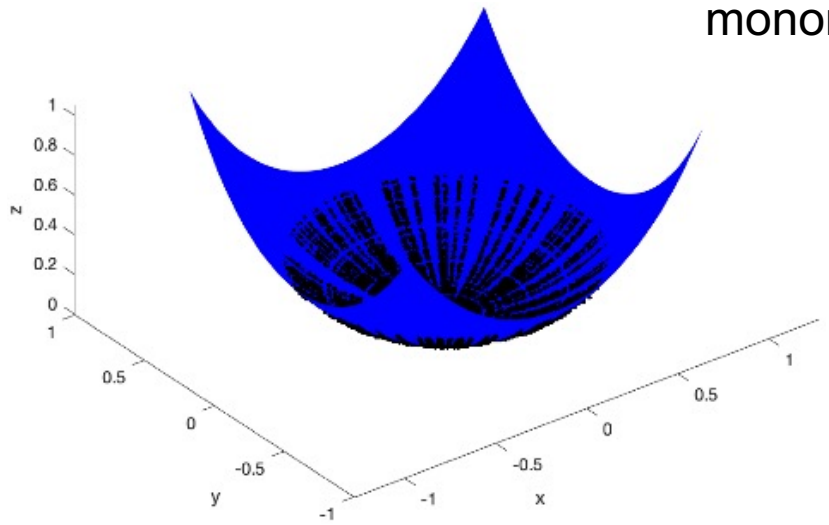
$$\mathbf{y} = \mathbf{w}(\boldsymbol{\eta}) = \mathbf{V}\boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{SSM}}} \mathbf{W}_{\mathbf{k}}\boldsymbol{\eta}^{\mathbf{k}}$$

$\mathbf{V}$

$\mathbf{W}_{\mathbf{k}}$

$\boldsymbol{\eta}^{\mathbf{k}} = \eta_1^{k_1} \cdots \eta_d^{k_d}$

monomial terms



Reduced Dynamics Info

```
>> RDInfo

RDInfo =

  struct with fields:

          reducedDynamics: [1×1 struct]
    inverseTransformation: [1×1 struct]
        conjugateDynamics: [1×1 struct]
           transformation: [1×1 struct]
           conjugacyStyle: 'normalform'
             dynamicsType: 'flow'
    eigenvaluesLinPartFlow: [2×1 double]
      eigenvectorsLinPart: [2×2 double]
```

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

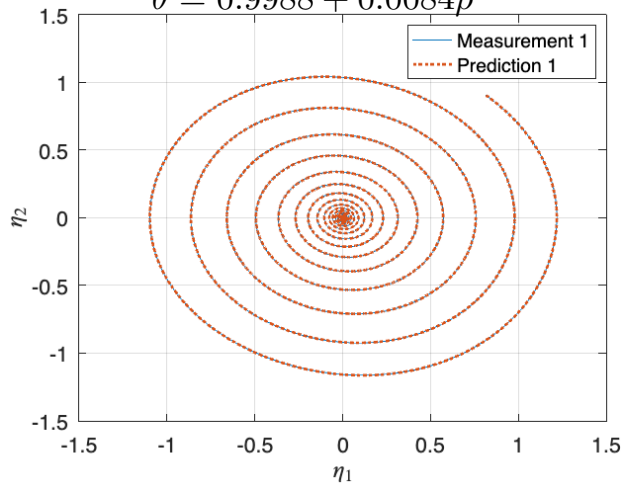
$$\mathbf{z} = \boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{T}_{\mathbf{k}}^{\text{inv}}\boldsymbol{\eta}^{\mathbf{k}}$$

$$\dot{\mathbf{z}} = \mathbf{R}_0\mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}_{\mathbf{k}}^{\text{norm}}\mathbf{z}^{\mathbf{k}}$$

$$\boldsymbol{\eta} = \mathbf{z} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{T}_{\mathbf{k}}\mathbf{z}^{\mathbf{k}}$$

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

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



Additional options

1. Instead of a one-step optimization (Steps 7 – 10 on Slide 9), normalization of the dynamics can be **done analytically** using an interface with `SSMTool`. This approach is called `fastSSM` [13] and greatly reduces the computational costs

2. 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

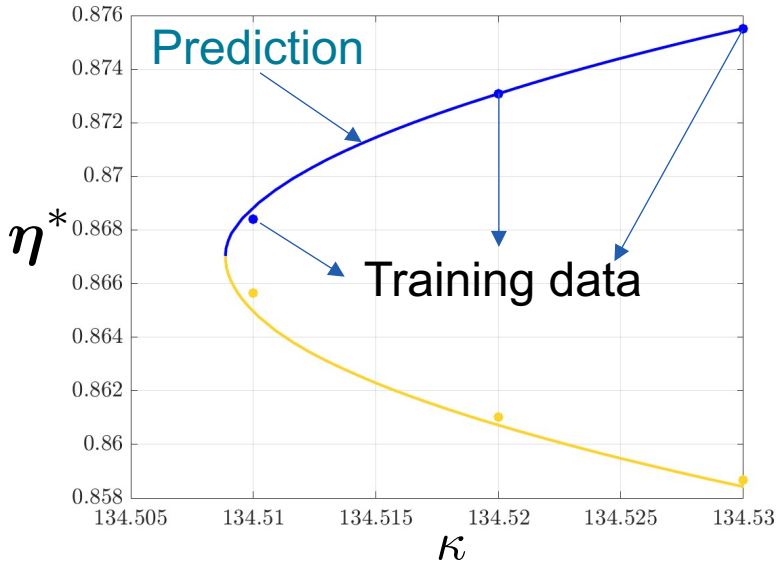
$$\dot{\boldsymbol{\eta}} = \mathbf{r}(\boldsymbol{\eta}, \kappa) = \mathbf{R}_0(\kappa)\boldsymbol{\eta} + \sum_{|\mathbf{k}|=2}^{n_{\text{model}}} \mathbf{R}(\kappa)_{\mathbf{k}} \boldsymbol{\eta}^{\mathbf{k}}$$

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

```
>> yData

yData =

  3×3 cell array

    {1×5001 double}    {20×5001 double}    {[0.0100]}
    {1×3001 double}    {20×3001 double}    {[0.0100]}
    {1×3001 double}    {20×3001 double}    {[0.0200]}
```

Third column of `yData`: parameter value

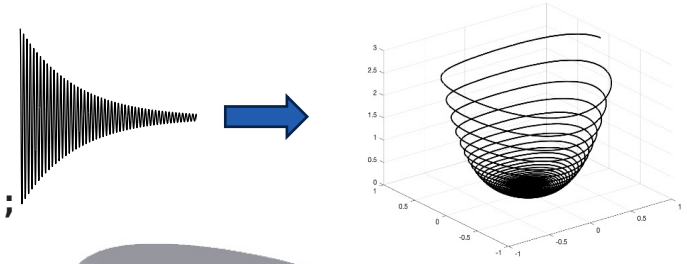
Example: Fixed points of plane Couette flow [14]



Summary: A typical use case

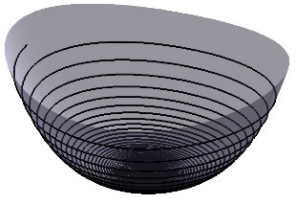
1. Preprocessing: find the dimension, truncate, and embed

```
[startTime, indStartTime, SSMDim] = SSM_startTime(xData(1,:),1);
xData_tr = sliceTrajectories(xData, [startTime, Inf]);
[yData, opts_embd] = coordinatesEmbedding(xData_tr, SSMDim, 'OverEmbedding', overEmbed);
```



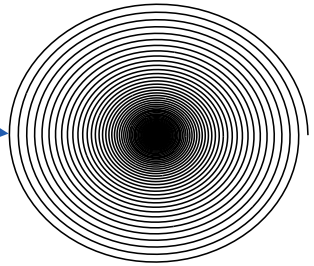
2. Fitting the SSM geometry

```
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');
```

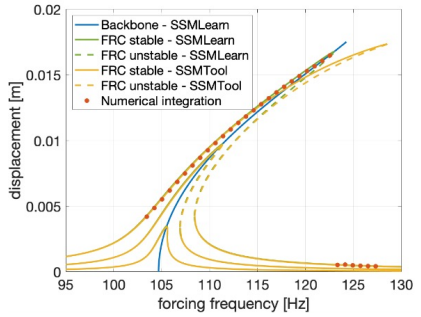


4. Evaluation on test set

```
[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, yObservable)
```



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

References

[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] J. Axås, G. Haller, **Model reduction for nonlinearizable dynamics via delay-embedded spectral submanifolds,** *Nonlinear Dyn* **111** 22079 (2023).

[10] L. Bettini, B. Kaszás, B. Zybach, J. Dual, and G. Haller. **Data-driven nonlinear model reduction to spectral submanifolds via oblique projection**, Chaos, (submitted) (2024).

[11] B. Fornberg, **Generation of finite difference formulas on arbitrarily spaced grids**, Math. Comput, **51**, 699–706 (1988).

[12] J. Guckenheimer, P. Holmes, **Nonlinear oscillations, dynamical systems, and bifurcations of vector fields**, Applied Mathematical Sciences, Vol. 42 (Springer, Berlin, 2013).

[13] J. Axås, M. Cenedese, G. Haller, **Fast data-driven model reduction for nonlinear dynamical systems,** Nonlinear Dyn **111,** 7941-7957 (2023).

[14] B. Kaszás, M. Cenedese, G. Haller, **Dynamics-based machine learning of transitions in Couette flow,** Phys. Rev. Fluids **7**, L082402 (2022).