

# Bachelor's Thesis

- [Playing around with parameters](#)
- [Data.csv Structure](#)
- [Other implementations](#)
- [Explanation of L2 norm of an error](#)
- [Various cutoffs](#)
- [Notes](#)
- [Bilevel optimization for regression of HH onto FHN](#)

# Playing around with parameters

## Optimizer

### STLSQ

*"Sequentially thresholded least squares"*

They take the argument  $\lambda$ , which specifies a threshold "of sparsity". Should parameter  $p_i$  be smaller than  $\lambda$ , it will be removed (and its corresponding term)

Meaning that the bigger the threshold is, the **less** terms may be present in the final model. In other words, the bigger the threshold, the more terms will be removed.

For `1D-Neuron-Multiple` the value of  $\lambda$  seemed to be best around  $(10^{-1}, 10^{-3})$ , smaller values of  $\lambda$  included insignificant noise.

“ Default value is  $\lambda = 10^{-1}$

Also for the [basic example](#) they set the value to  $\lambda = 10^{-1}$ , which seems to agree.

TODO: STLSQ with vector of thresholds

## Metrics

### $L_2$ Norm error

$L_2$  (or  $l_2$ ) norm of the error gives the  $l_2$  of an error in each dimension

Of course the smaller the better

# TVDIFF

Most times around `300` iterations seems to be by far enough

## Regularization $\alpha$

The regularization parameter  $\alpha$  tells us how strongly should the derivative be regularized (think of it as smoothed)

The bigger the  $\alpha$  the less it oscillates, though the less "features" of the true derivative it really exhibits

It is mostly visible when there is a big spike in the derivative. Then the `tvdiff` is unable "to catch up" when strongly regularized doesn't handle the spike well (it simply doesn't feature nearly as big of a spike)

I'd recommend starting with a higher  $\alpha$  and slowly increasing it, until we find the derivative smooth enough.

With small  $\alpha$  always check if it more or less corresponds to data (it has tendency to oscillate when the function is too constant)

## Epsilon $\epsilon$

Using `tvdiff` with  `$\epsilon=1e-9$` , we obtain a strongly regularized result. Larger values of  `$\epsilon$`  improve conditioning and speed, while smaller values give more accurate results with sharper jumps.

## Scale and preconditioner

TO BE DONE

## Performance

In general the more data (and thus the derivation) varies in scale, the worse to model performs

# Collocations

Data collocation is only used when derivative is NOT supplied (and is surely better than forward diff)

TODO: Usage collocations on existing derivative?

# Data.csv Structure

File `data.csv` should follow structure, where each "run" is suffixed by `_` and the index of the run. For example the columns could be

- `time__1`
- `x__1`
- `y__1`
- `time__2`
- `x__2`
- `y__2`

whaz

For simple singular trajectories, it remains to be done per [BTHS-19 - Explore how many trajectories can be used as opposed to just one](#)

# Other implementations

Here are some implementations in other languages (and hopefully guides to use them)

- [matlab](#)
- [python](#)

# Explanation of L2 norm of an error

When solving for the sparsest possible set of DEs, it is likely our found model will **not** describe the data exactly - there will be an error

Therefore we can measure the error and give the user it's  $\ell^2$ -norm

- The error is a vector of errors at each time-step
- More information regarding the  $\ell^2$ -norm is [here](#)

When working with 2 or more dimensional data, the  $\ell^2$ -norm returned will be vector of  $\ell^2$  norms in each coordinate

# Various cutoffs

Should be

$$\text{Differential}(t)(V) = p_1 + V \cdot p_2 + W \cdot p_4 + p_3 \cdot (V^3)$$

$$\text{Differential}(t)(W) = p_5 + V \cdot p_6 + W \cdot p_7$$

Always the title is `cutoff` and `optimization method`

## Smooth Forward Df

### 2000 & STLSQ

Model ##Basis#388 with 2 equations

States : V W

Parameters : 6

Independent variable: t

Equations

$$\text{Differential}(t)(V) = p_1 + V \cdot p_2 + W \cdot p_4 + p_3 \cdot (V^2)$$

$$\text{Differential}(t)(W) = p_5 + W \cdot p_6$$

Linear Solution with 2 equations and 6 parameters.

Returncode: solved

L<sub>2</sub> Norm error : [74.78388648297391, 25.370167490831182]

AIC : [34110.303086665364, 25566.935128488396]

R<sup>2</sup> : [-1.7705432321293069, 0.48830612769746606]

Parameters:

[0.3, 0.8, -0.4, -0.3, 0.6, -0.31]

### 1000 & SR3



Model ##Basis#405 with 2 equations

States : V W

Parameters : 8

Independent variable: t

Equations

Differential(t)(V) =  $V \cdot p_1 + W \cdot p_4 + p_2 \cdot (V^2) + p_3 \cdot (V^3)$

Differential(t)(W) =  $p_5 + V \cdot p_6 + W \cdot p_8 + p_7 \cdot (V^2)$

Linear Solution with 2 equations and 8 parameters.

Returncode: solved

L<sub>2</sub> Norm error : [211.63117047913556, 1062.8083759316835]

AIC : [47690.18501396972, 62058.069854531444]

R<sup>2</sup> : [-3.0596412951792242, 0.6042583461776001]

Parameters:

[1.09, -0.8, 0.11, -0.15, 0.24, -0.13, 0.21, -0.14]

## 2000 & SR3

Model ##Basis#411 with 2 equations

States : V W

Parameters : 8

Independent variable: t

Equations

Differential(t)(V) =  $p_1 + V \cdot p_2 + p_3 \cdot (V^2) + p_4 \cdot (V^3)$

Differential(t)(W) =  $p_5 + V \cdot p_6 + p_7 \cdot (V^2) + p_8 \cdot (W^2)$

Linear Solution with 2 equations and 8 parameters.

Returncode: solved

L<sub>2</sub> Norm error : [288.2814774371565, 1582.4585434912008]

AIC : [44778.096916964794, 58235.30636382105]

R<sup>2</sup> : [-9.680058684080175, -30.916791253738456]

Parameters:

[-0.5, 1.8, -1.4, 0.29, 0.27, -0.25, 0.3, -0.18]

# Tvdiff Df

## 1000 & SR3

Model ##Basis#507 with 2 equations

States : V W

Parameters : 8

Independent variable: t

Equations

$\text{Differential}(t)(V) = V \cdot p_1 + W \cdot p_4 + p_2 \cdot (V^2) + p_3 \cdot (V^3)$

$\text{Differential}(t)(W) = p_5 + V \cdot p_6 + W \cdot p_8 + p_7 \cdot (V^2)$

Linear Solution with 2 equations and 8 parameters.

Returncode: solved

L<sub>2</sub> Norm error : [211.3101858827484, 1053.3627931231706]

AIC : [47676.671431685485, 61978.59180147926]

R<sup>2</sup> : [-3.054002039146713, 0.6099054364959031]

Parameters:

[1.09, -0.8, 0.11, -0.15, 0.24, -0.13, 0.21, -0.14]

## 1 & SR3

Model ##Basis#446 with 2 equations

States : V W

Parameters : p<sub>1</sub> p<sub>2</sub> p<sub>3</sub> p<sub>4</sub>

Independent variable: t

Equations

$\text{Differential}(t)(V) = 0$

$\text{Differential}(t)(W) = p_1 + V \cdot p_2 + W \cdot p_4 + p_3 \cdot (V^2)$

Linear Solution with 2 equations and 4 parameters.

Returncode: solved

L<sub>2</sub> Norm error : [155.0710066105855, 9291.229194306132]

AIC : [49952.53064617247, 90480.8524061853]

R<sup>2</sup> : [-0.30329307021503427, 0.8870442179710343]

Parameters:

[0.27, -0.28, 0.26, -0.11]

# With smoothing and without supplied derivative

## 1000 & SR3 & GaussianKernel

# Notes

```
$$ \xdef\scal#1#2{\langle #1, #2 \rangle} \xdef\norm#1{\left\| \right. #1 \right\|}
\xdef\dist{\rho} \xdef\and{\&} \xdef\AND{\quad \and \quad} \xdef\brackets#1{\left\{ #1 \right\}}
\xdef\parc#1#2{\frac {\partial #1} {\partial #2}} \xdef\mtr#1{\begin{pmatrix} #1 \end{pmatrix}}
\xdef\bm#1{\boldsymbol{#1}} \xdef\mc#1{\mathcal{#1}}
\xdef\vv#1{\mathbf{#1}} \xdef\vp#1{\pmb{#1}} \xdef\ve{\varepsilon} \xdef\l{\lambda}
\xdef\th{\vartheta} \xdef\alpha{\alpha} \xdef\vf{\varphi} \xdef\Tagged#1{(\text{#1})}
\xdef>tagged*#1{\text{#1}} \xdef>tagEqHere#1#2{\href{#2\#eq-#1}{(\text{#1})}}
\xdef>tagDeHere#1#2{\href{#2\#de-#1}{\text{#1}}} \xdef>tagEq#1{\href{\#eq-#1}{(\text{#1})}}
\xdef>tagDe#1{\href{\#de-#1}{\text{#1}}} \xdef\T#1{\htmlld{eq-#1}{#1}}
\xdef\D#1{\htmlld{de-#1}{\vv{#1}}} \xdef\conv#1{\mathrm{conv}}, #1
\xdef\cone#1{\mathrm{cone}}, #1 \xdef\aff#1{\mathrm{aff}}, #1 \xdef\lin#1{\mathrm{Lin}}, #1
\xdef\span#1{\mathrm{span}}, #1 \xdef\O{\mathcal O} \xdef\ri#1{\mathrm{ri}}, #1
\xdef\rd#1{\mathrm{r}\partial}, #1 \xdef\interior#1{\mathrm{int}}, #1 \xdef\proj{\Pi}
\xdef\epi#1{\mathrm{epi}}, #1 \xdef\grad#1{\mathrm{grad}}, #1
\xdef\gradT#1{\mathrm{grad}^T #1} \xdef\gradx#1{\mathrm{grad}_x #1}
\xdef\hess#1{\nabla^2, #1} \xdef\hessx#1{\nabla^2_x #1} \xdef\jacobx#1{D_x #1}
\xdef\jacob#1{D #1} \xdef\grads#1#2{\mathrm{grad}_{#1} #2} \xdef\subdif#1{\partial #1}
\xdef\co#1{\mathrm{co}}, #1 \xdef\iter#1{\wedge^{[#1]}} \xdef\str{\wedge^*} \xdef\spv{\mathcal V}
\xdef\civ{\mathcal U} \xdef\other#1{\hat{#1}} \xdef\prox{\mathrm{prox}}
\xdef\sign#1{\mathrm{sign}}, #1 \xdef\brackets#1{\left( #1 \right)} $$
```

Počítejme  $(Y - X \xi)^T (Y - X \xi) = (Y^T - \xi^T X^T) (Y - X \xi) = Y^T Y - Y^T X \xi - \xi^T X^T Y + \xi^T X^T X \xi$

a pak  $\frac{\partial}{\partial \xi}$  tohoto výrazu je

$0 - (Y^T X)^T - X^T Y + (X^T X + (X^T X)^T) \xi = 0 - X^T Y - X^T Y + (X^T X + X^T X) \xi = -2 X^T Y + 2 X^T X \xi$

“ Viz přednáška z [lineárních statistických modelů](#)

## Proximální operátor $\| \cdot \|_1$ -normy

Víme, že  $\arg \min_{\|x\|_1} \|x - v\|_2$  je řešení minimalizačního problému  $\min_{\|x\|_1} \left( \lambda \|x\|_1 + \frac{1}{2} \|x - v\|_2^2 \right)$

Uvědomme si, že  $\|\text{grads}\{\mathbf{x}\}\|_1 = \text{mtr}\{\text{parc}\{|x_1| \} \{x_1\} \ \vdots \ \text{parc}\{|x_n| \} \{x_n\} \}$ , proto  $\|\text{grads}\{\mathbf{x}\}\|_1 = \begin{cases} \text{sign } x_i & x_i \neq 0 \\ 0 & x_i = 0 \end{cases}$  a také  $\|\text{grads}\{\mathbf{x}\}\|_2^2 = \|\text{grads}\{\mathbf{x}\}\|_2^2 = 2 \text{mtr}\{(x_1 - v_1) \ \vdots \ (x_n - v_n) \}$  Tedy stacionární bod  $\hat{\mathbf{x}}$  našeho problému musí splňovat  $\lambda \text{mtr}\{\text{parc}\{|x_1| \} \{x_1\} \ \vdots \ \text{parc}\{|x_n| \} \{x_n\} \}$

# Bilevel optimization for regression of HH onto FHN

[illegible]

Vzpomeňme si, že SINDy metoda spočívala v optimalizaci 
$$\min_{\|v\|_1} \left\{ \frac{1}{2} \left\| \sum_{i=1}^n v_i \dot{X}^{(i)} - \sum_{i=1}^n v_i \Theta(X) v_i \right\|_2^2 + \mu \sum_{i=1}^n |v_i| \right\},$$
 což můžeme v případě LASSO regrese jakožto optimalizační metody napsat jako 
$$\min_{\|v\|_1} \left\{ \frac{1}{2} \left\| \sum_{i=1}^n v_i \dot{X}^{(i)} - \sum_{i=1}^n v_i \Theta(X) v_i \right\|_2^2 + \mu \sum_{i=1}^n |v_i| \right\}.$$

Předpokládejme, že máme "pevnou" trajektorii  $\{H\}$  HH modelu a derivace  $\{\dot{H}\}$  a obdobně pro FHN model trajektorie  $\{F\}$  a příslušné derivace  $\{\dot{F}\}$ .

Potom nalezení lineární transformace  $\mathbb{v} \mapsto \Lambda \mathbb{v}$  modelu HH na model FHN můžeme formulovat jako úlohu 
$$\min_{\Lambda} \left\{ \frac{1}{2} \|\mathbb{F} - \mathbb{H}(\Lambda)\|_F^2 + \alpha \|\Lambda\|_F^2 \right\} \quad \text{tag{T{LTR}}}$$

Jak jsme si řekli, tak uvažujeme, že trajektorie  $\mathbf{H}$  je "pevná", tedy že nemůžeme měnit parametry HH modelu. Naopak o modelu FHN předpokládáme, že jeho parametry měnit můžeme. Tedy bychom chtěli nalézt pro model FHN 
$$\frac{d \mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}; \mathbf{p}),$$
 kde funkce  $\mathbf{f}$  zadává FHN model,  $\mathbf{p}$  je vektor parametrů a  $\mathbf{u}$  je stav FHN systému,

takové parametry, že řeší úlohu  $\min_{\mathbf{p}} \underbrace{\frac{1}{2} \sum_{i=1}^N \|\mathbf{u}(t_i; \mathbf{p}) - \mathbf{H}_{i, \cdot} \mathbf{v} \mathbf{\Lambda}\|_2^2}_{\text{mcal } F_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p})}} \tag{T\{PRM.1\}}$ , kde  $\mathbf{H}_{i, \cdot}$  je  $i$ -té pozorování HH modelu (v čase  $t_i$ ) a  $\mathbf{u}(t_i; \mathbf{p})$  je pozorování FHN modelu v čase  $t_i$  za předpokladu parametrů  $\mathbf{p}$ . Označme  $\mathbf{u}(T; \mathbf{p}) = [\mathbf{u}(t_1; \mathbf{p}) \ \dots \ \mathbf{u}(t_N; \mathbf{p})]$ ,  $T = \{t_1, \dots, t_N\}$ . Potom můžeme tyto 2 části dát dohromady a formulovat úlohu  $\min_{\mathbf{p}} \overbrace{\frac{1}{2} \sum_{i=1}^N \|\mathbf{u}(t_i; \mathbf{p}) - \mathbf{H}_{i, \cdot} \mathbf{v} \mathbf{\Lambda}\|_2^2}^{\text{mcal } H_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p})}} \tag{T\{F2H.1\}}$  nebo také zkrácené  $\min_{\mathbf{p}} \mathbf{\Lambda} \text{mcal } H_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p}) \tag{T\{PRM.2\}}$  za podmínky  $\mathbf{v} \mathbf{\Lambda} = \argmin_{\mathbf{p}} \frac{1}{2} \sum_{i=1}^N \|\mathbf{u}(t_i; \mathbf{p}) - \mathbf{H}_{i, \cdot} \mathbf{v} \mathbf{\Lambda}\|_2^2 \tag{T\{F2H.1\}}$  nebo také zkrácené  $\min_{\mathbf{p}} \mathbf{\Lambda} \text{mcal } H_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p}) \tag{T\{PRM.2\}}$  za podmínky  $\mathbf{v} \mathbf{\Lambda} = \argmin_{\mathbf{p}} \mathbf{\Lambda} \text{mcal } F_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p})$

Pravděpodobně nemusíme řešit případ  $\mathbf{v} \mathbf{\Lambda} = \mathbf{0}$ ,  $\mathbf{v} \mathbf{\Lambda} = \mathbf{0}$ , neboť i pro  $\mathbf{v} \mathbf{\Lambda} = \mathbf{0}$  nepovoluje tvar FHN modelu konstantní nulové řešení, které by bylo best fitem pro  $\mathbf{v} \mathbf{\Lambda} = \mathbf{0}$ .

Ačkoliv by tento přístup byl jistě užitečný, dostáváme se do problému s nalezením podmínky stacionarity pro optimalizaci  $\mathbf{v} \mathbf{\Lambda} = \argmin_{\mathbf{p}} \mathbf{\Lambda} \text{mcal } F_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p})$ . Pokud bychom ji chtěli najít, museli bychom spočítat  $\frac{\partial \mathbf{\Lambda} \text{mcal } F_{\mathbf{v} \mathbf{\Lambda}}}{\partial \mathbf{p}}$ , avšak bez této podmínky nejsme schopni optimalizovat celou úlohu.

Toto platí v případě, že bychom úlohu optimalizovali metodou vyžadující gradient účelové funkce. Např. metoda "Nelder-Mead" se bez něj obejde

Proto raději použijme analogii SINDy metody, která nám umožní tento problém obejít. Proto místo úlohy  $\tag{T\{PRM.1\}}$  řešme  $\min_{\mathbf{p}} \underbrace{\frac{1}{2} \sum_{i=1}^N \|\mathbf{f}(\mathbf{H}_{i, \cdot} \mathbf{v} \mathbf{\Lambda}; \mathbf{p}) - \dot{\mathbf{H}}_{i, \cdot}\|_2^2}_{\text{mcal } F'_{\mathbf{v} \mathbf{\Lambda}}(\mathbf{p})}} \tag{T\{PRM.2\}}$