Sparse regression

Filton workshop 2024

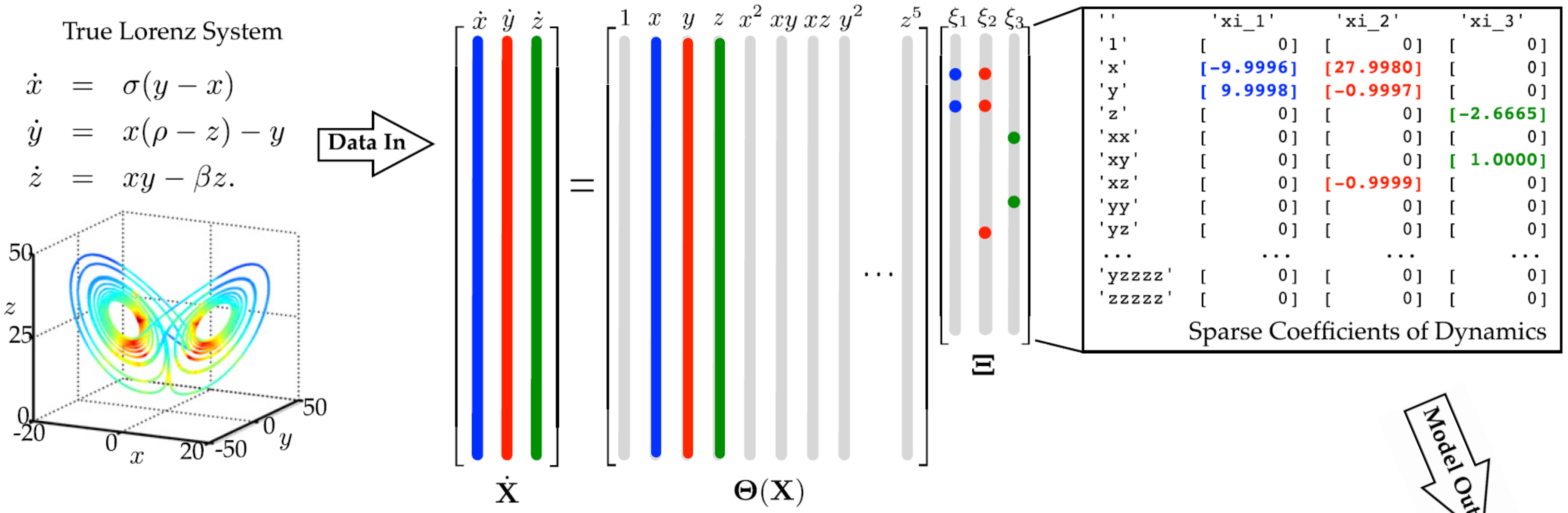
Urban Fasel

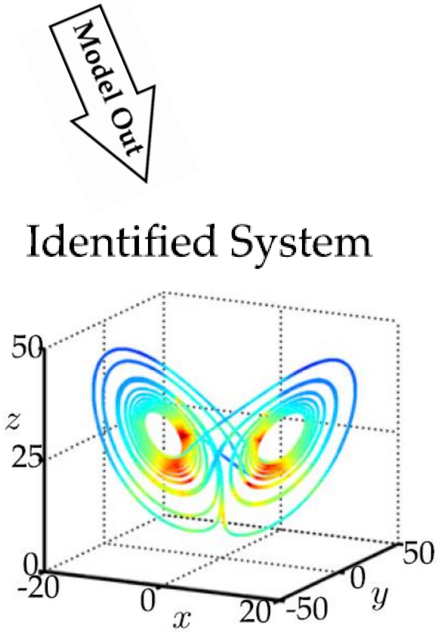
Imperial College London

Literature

- SL Brunton, JL Proctor, JN Kutz (2016) Discovering governing equations from data by sparse identification of nonlinear dynamical systems.
- G Sanchez, E Marzban (2020) All Models Are Wrong: Concepts of Statistical Learning.
- L Tibshirani (1996) Regression shrinkage and selection via the lasso.
- H Zou, T Hastie (2005) Regularization and variable selection via the elastic net.
- P Zheng, T Askham, SL Brunton, JN Kutz, AY Aravkin (2018) A Unified Framework for Sparse Relaxed Regularized Regression: SR3.
- T Blumensath, ME Davies (2009) Iterative hard thresholding for compressed sensing.

SINDy – sparse regression



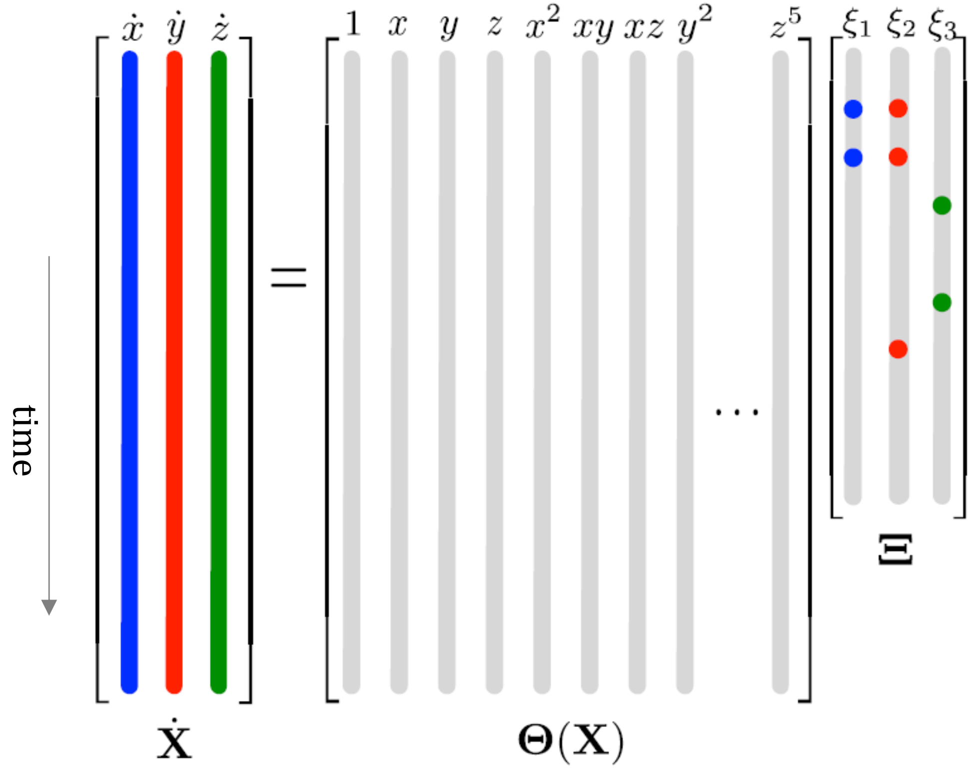


SINDy sparse regression

$$\rightarrow \quad \hat{\boldsymbol{\xi}}_k = \mathrm{argmin}_{\boldsymbol{\xi}_k} \underbrace{\left\|\dot{\mathbf{X}}_k - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\xi}_k\right\|_2^2}_{\text{least squares}} + \underbrace{\lambda\|\boldsymbol{\xi}_k\|_0}_{\ell_0\text{-penalized}}$$

Lecture outline

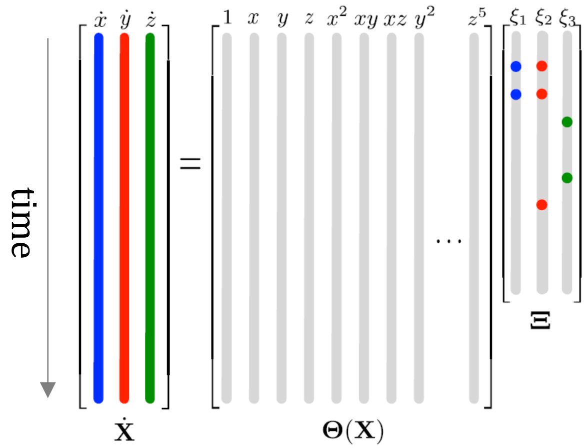
- Sparse regression – notation
- Ordinary least squares
- Ridge regression (ℓ_2)
- LASSO regression (ℓ_1)
- Geometric intuition for sparse solution
- Sequentially thresholded least squares
  - approximating ℓ_0
- MATLAB examples



Sparse regression – notation

SINDy system of ODEs: $\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}$

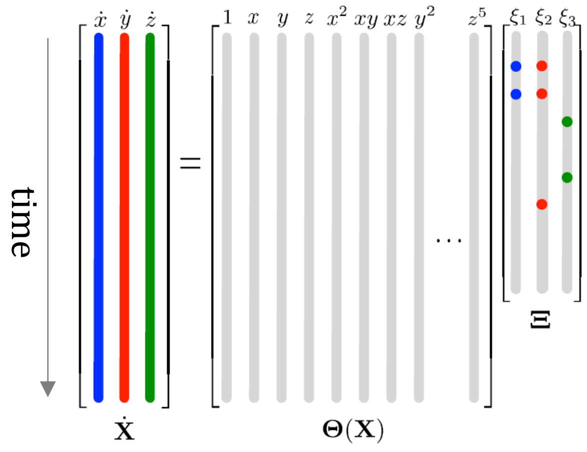
- Library terms: $\mathbf{\Theta}(\mathbf{X}) \in \mathbb{R}^{\mathrm{m \times D}}$
- Derivatives: $\dot{\mathbf{X}} \in \mathbb{R}^{\mathrm{m \times n}}$
- Coefficients: $\mathbf{\Xi} \in \mathbb{R}^{\mathrm{D \times n}}$ (e.g. $\mathbf{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3]$)



Sparse regression – notation

SINDy system of ODEs: $\dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$ $\quad (\mathbf{b} = \mathbf{A}\mathbf{x})$

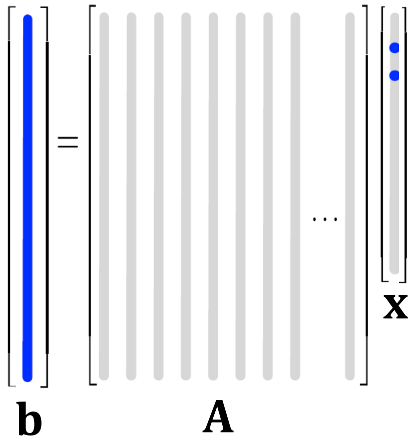
- Library terms (*features* $\mathbf{A}$): $\boldsymbol{\Theta}(\mathbf{X}) \in \mathbb{R}^{\mathrm{m}\times\mathrm{D}}$
- Derivatives (*response* $\mathbf{b}$): $\dot{\mathbf{X}} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$
- Coefficients (*loadings* $\mathbf{x}$): $\boldsymbol{\Xi} \in \mathbb{R}^{\mathrm{D}\times\mathrm{n}}$ (e.g. $\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3]$)



Change notation and n=1

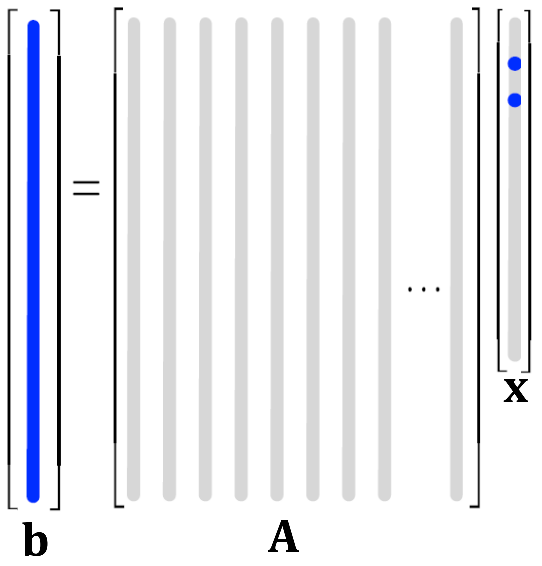
$\rightarrow \quad \dot{\mathbf{X}} = \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}$

$\rightarrow \quad \mathbf{b} = \mathbf{A}\mathbf{x}$



Sparse regression – notation

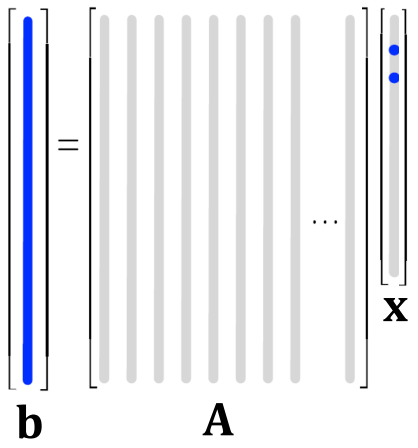
- Library terms (*features* $\mathbf{A}$): $\boldsymbol{\Theta}(\mathbf{X})$ $\in \mathbb{R}^{\mathrm{m}\times\mathrm{D}}$
- Derivatives (*response* $\mathbf{b}$): $\dot{\mathbf{X}}$ $\in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$
- Coefficients (*loadings* $\mathbf{x}$): $\boldsymbol{\Xi}$ $\in \mathbb{R}^{\mathrm{D}\times\mathrm{n}}$



Generally: $m > D$ $\rightarrow$ over-determined system

- Objective: find accurate & sparse model
- Solve least squares regression with regularization (ℓ_0, ℓ_1, (ℓ_2)):
- Optimization: $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \mathrm{L}(\mathbf{x})$
  - Find argument $\mathbf{x}$ that minimizes loss function $\mathrm{L}(\mathbf{x})$
  - Loss function: $\mathrm{L}(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$
- Accuracy $\rightarrow$ $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ & Sparsity $\rightarrow$ $\lambda\|\mathbf{x}\|_1$

Ordinary least squares: $\hat{\mathbf{x}} = \mathrm{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$



Minimize mean squared error: $\mathrm{MSE} = \frac{1}{n}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 = \frac{1}{n}(\mathbf{b} - \mathbf{A}\mathbf{x})^{\mathrm{T}}(\mathbf{b} - \mathbf{A}\mathbf{x})$

- $\frac{\partial}{\partial \mathbf{x}}\mathrm{MSE}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}}\left(\frac{1}{n}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \frac{2}{n}\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{b} + \frac{1}{n}\mathbf{b}^{\mathrm{T}}\mathbf{b}\right)$
  $= \frac{2}{n}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \frac{2}{n}\mathbf{A}^{\mathrm{T}}\mathbf{b} = 0$
- $\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{b} \quad \rightarrow \quad \hat{\mathbf{x}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b} \quad \rightarrow$ unique solution

Challenges / limitations

1. Multicollinearity: Two or more features (e.g. library terms) are highly correlated
   - $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ ill-conditioned → near singular
   - Leads to unstable, irregular estimates of $\mathbf{x}$ → fit is sensitive to small perturbations
2. Dense solutions: we don't perform feature (variable) selection …
   - possibly overfitting to data, difficult to interpret the results …

Approach: *regularization* → *Ridge regression, LASSO, STLS*

Ridge regression: $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_2^2$

Idea: impose restriction on **squared magnitude of sum** of coefficients $\mathbf{x}$

- ℓ_2-penalized least squares
- Loss function:
  - $\mathrm{L}(\mathbf{x}) = \frac{1}{n}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_\mathbf{2}^\mathbf{2} = \mathrm{MSE}(\mathbf{x}) + \frac{\lambda}{\mathrm{n}}\mathbf{x}^{\mathrm{T}}\mathbf{x}$
- Unique solution:
  - $\frac{\partial}{\partial \mathbf{x}}\mathrm{L}(\mathbf{x}) = \frac{2}{n}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \frac{2}{n}\mathbf{A}^{\mathrm{T}}\mathbf{b} + \frac{2\lambda}{n}\mathbf{x} = 0$
    - $\hat{\mathbf{x}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\mathbf{\Theta}^{\mathrm{T}}\mathbf{b}$ $\rightarrow$ *adding positive elements ($\lambda \geq 0$) to diagonal*

      $\rightarrow$ *increasing condition number, stabilizing solution*

How to select the parameter λ ? $\rightarrow$ model selection

- e.g. cross validation $\rightarrow$ *discussed in next lecture*

LASSO regression: $\hat{\mathbf{x}} = \operatorname{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1$

Ridge regression solution not sparse …

Idea: impose restriction on **magnitude of sum** of coefficients $\mathbf{x}$

- ℓ_1-penalized least squares
- Loss function:
  - $\mathrm{L}(\mathbf{x}) = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 = \mathrm{MSE}(\mathbf{x}) + \lambda \sum |x_j|$
- Loss function (ℓ_1 norm) not differentiable …
  - Variety of methods to solve optimization from convex analysis and optimization theory:
    - e.g. coordinate descent, proximal gradient method (soft thresholding), …

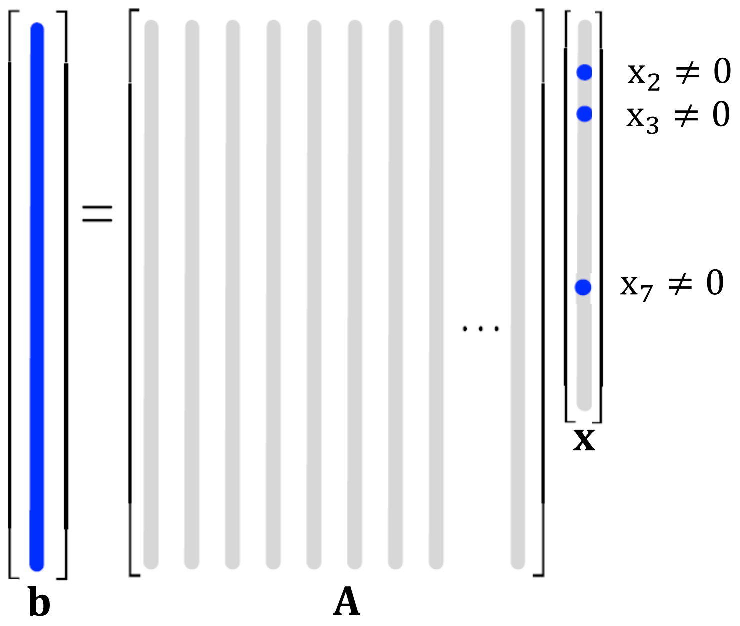
LASSO: Least Absolute Shrinkage and Selection Operator

- Shrinkage: constraining magnitude of coefficients → similar to ridge regression
- Selection: promoting sparsity → setting small coefficients to zero

LASSO implementation

MATLAB example: feature selection (comparing OLS, ridge, LASSO)

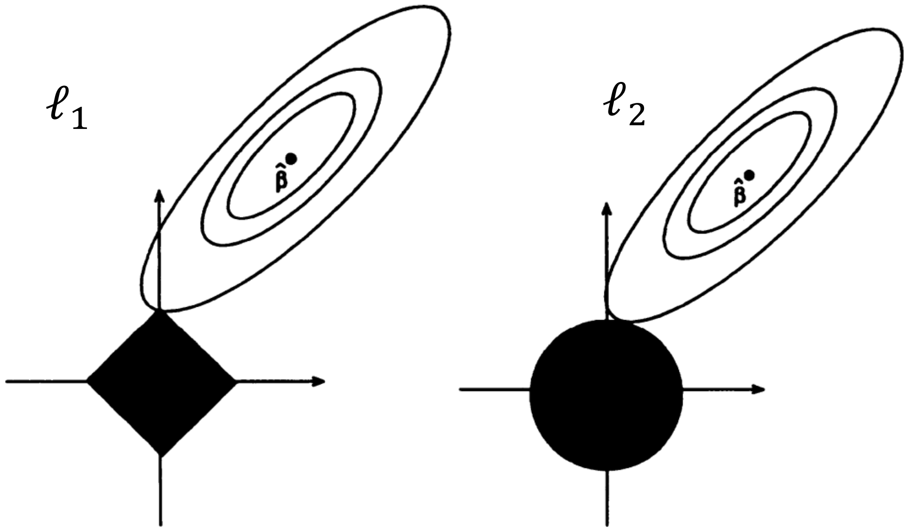
- Data set consisting of 100 observations of a response $\mathbf{b}$
  - *SINDy* → derivatives $\dot{\mathbf{X}}$
- Each outcome given by a combination of 3 of 10 candidate features $\mathbf{A}$ with loading $\mathbf{x}$
  - *SINDy* → time series data $\mathbf{X}$, library $\boldsymbol{\Theta}(\mathbf{X})$, and coefficients $\boldsymbol{\xi}$



Why is LASSO solution sparse?

Equivalent formulations of the LASSO optimization problem:

1. Regularized least squares
   - $\hat{\mathbf{x}} = \mathrm{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \; + \; \lambda\|\mathbf{x}\|_1$
2. Constrained optimization:
   - $\hat{\mathbf{x}} = \mathrm{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_1 \leq \mathrm{c}$
3. Constrained optimization 2:
   - $\hat{\mathbf{x}} = \mathrm{argmin}_{\mathbf{x}}\|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \leq \epsilon$



ℓ_1 vs ℓ_2 constrained optimization (for $D = 2$)

- ℓ_2: solution constrained to a circle around origin
- ℓ_1: solution constrained to a diamond around origin → *promoting sparse solution*

MATLAB example

L Tibshirani (1996) [Regression shrinkage and selection via the lasso](#).

Sequentially thresholded least squares – approx ℓ_0 penalty

Unfortunately: ℓ_1 penalized LS solutions for SINDy often not really sparse …

Solution: STLS $\rightarrow$ iteratively solving LS and hard thresholding small coefficients

- STLS approximates the solution to the ℓ_0-penalized least squares
  - ℓ_0 penalty: $\hat{\mathbf{x}} = \mathrm{argmin}_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_0$
  - ℓ_0-norm: counting all non-zero terms $\rightarrow$ non-convex, combinatorial search
- Improved performance over ℓ_1-penalized least squares (LASSO)

SINDY sparse regression $\Longrightarrow$

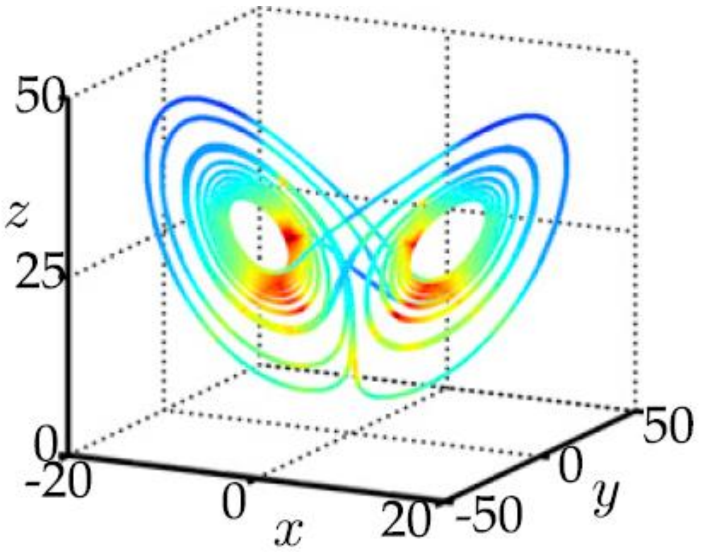
```
function Xi = sparsifyDynamics(Theta,dXdt,lambda,n)
% Compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % Initial guess: Least-squares

% Lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % Find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
% Regress dynamics onto remaining terms to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

LASSO vs STLS example

MATLAB example: SINDy Lorenz system

- Limitations of LASSO: false discovery occur early, already at small λ
  - False discoveries occur early in LASSO: Wu, Bogdam, Candes, 2015
  - SINDy uses STLS $\rightarrow$ approximate solution to ℓ_0-penalized LS



Coding examples and discussion

1. **MATLAB live scripts** (or implement SINDy in Python, Julia, …)
    - Test / break the method
    - Identify different dynamical systems
        - e.g. dysts database W Gilpin: Database, paper, PySINDy
            - 131 chaotic dynamical systems: fields such as astrophysics, climatology, and biochemistry
    - Test data requirements for different dynamical systems
    - Test custom libraries, e.g. combining polynomial with trig functions

2. **Explore PySINDy**
    - PySINDy lectures notebook by Alan Kaptanoglu
    - PySINDy feature overview