Fitting dynamical models to time series data & application to the *Drosophila* segmentation network

Theodore J. Perkins

McGill University

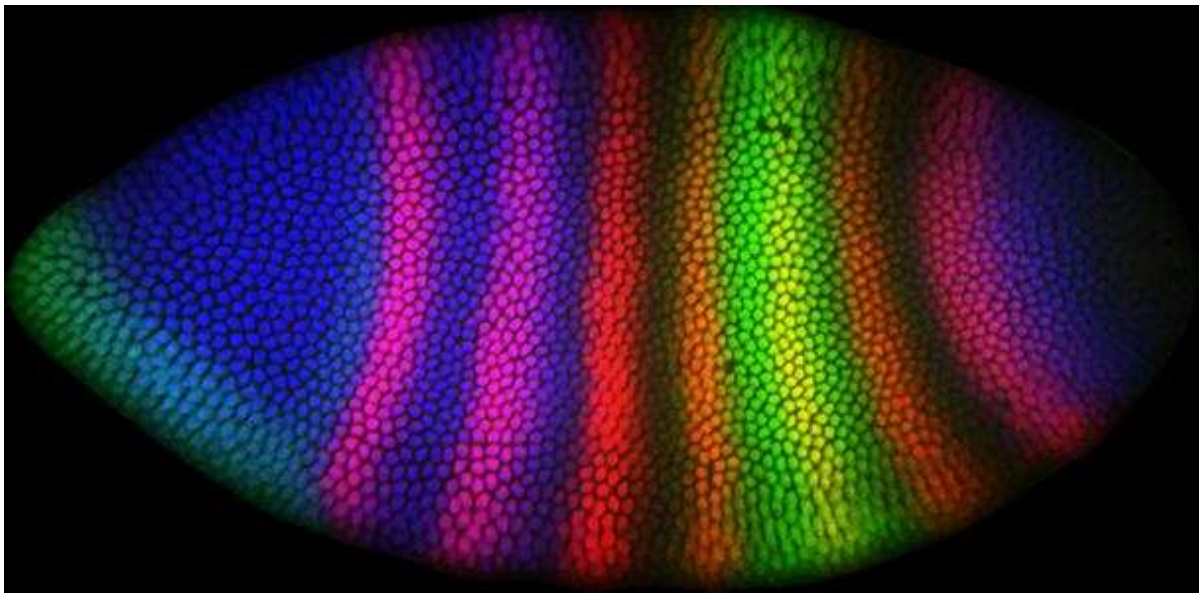
perkins@mcb.mcgill.ca www.mcb.mcgill.ca/∼perkins

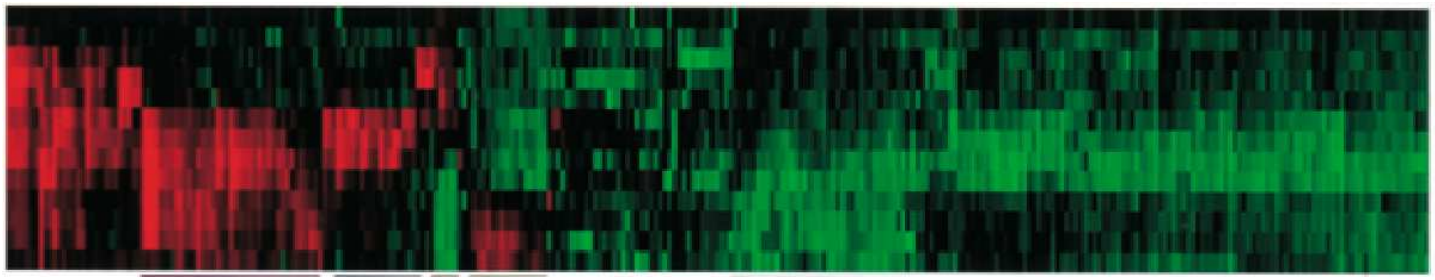
Outline

1. Methods for fitting dynamical models to time series data
   - Motivation for fitting dynamical models
   - Typical formulation of the problem (for ODEs)
     - Analytical intractability
     - Numerical methods & examples
   - The regression formulation of the problem
     - Functional data analysis
2. Application to the segmentation network of *Drosophila* melanogaster

Motivation – ever more quantitative data!

In situ hybridization imaging, 2D-PAGE, Mass spectrometry, microarrays, SAGE, protein arrays, . . . are generating enormous amounts of quantitive data on molecular presence/absence or abundance – including much time series data.





Why model it?

- This data can be hard for biologists to understand / interpret.
- Formulating and fitting dynamical models can:
  - Identify important system parameters (reaction rates)
  - Test the ability of models to explain the data
  - Identify the structure of the system (i.e., what regulates what)
  - Reveal detailed temporal (or spatial) properties of the data
  - . . .

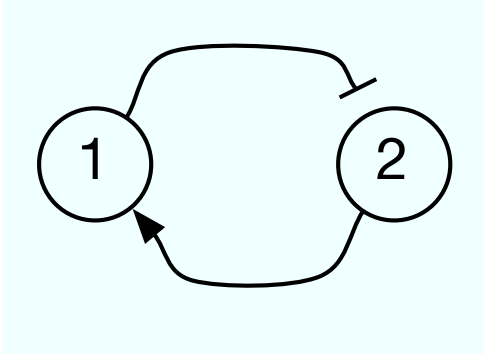
Problem formulation

- We observe a sequence of vectors $x(t_1), x(t_2), \ldots, x(t_T)$, where
  - $0 = t_1 < t_2 < \ldots < t_T$ are the observation times
  - each $x(t_i) \in \Re^M$
- The variables x may be: gene expression levels (mRNA or protein), metabolite concentrations, phosphorylation levels,. . .
- The observed variables may not include all important system variables
- The t_i need not be uniformly spaced

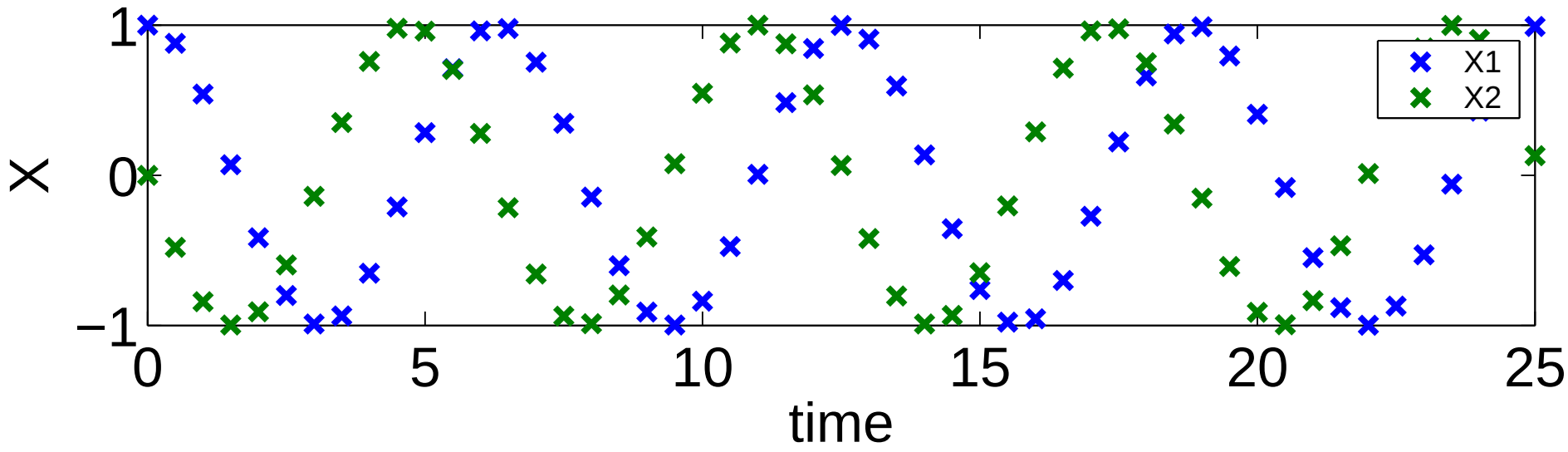
Aside: Most of what I will say can be generalized to multiple time series and/or spatio-temporal data.

Example time series

Consider a system of two genes, in which gene 2 activates gene 1, and gene 1 represses gene 2.



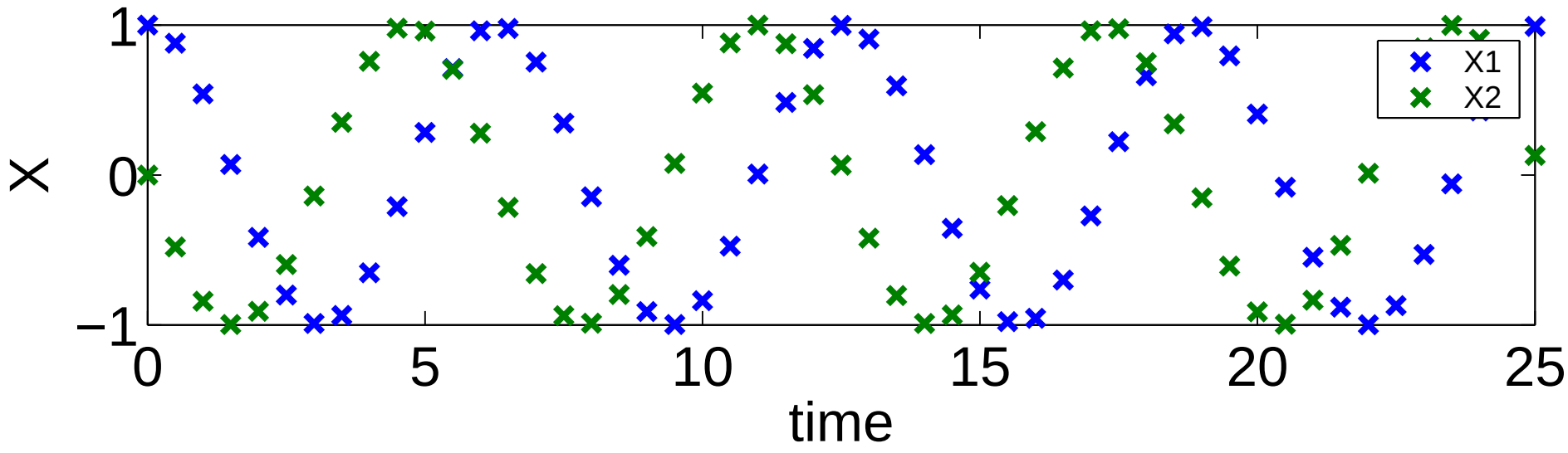
Suppose we measure the expression of two genes, 1 and 2:



Fitting ODE models

- Assume a model of the form $\dot{y} = f(y, \theta)$, where
  - $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$ is a vector or real parameters
  - The variables y are a superset of the observed variables x
  - Let y' be the modeled variables corresponding to x

First model for the example time-series



To start, we model both variables for a linear ODE:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & a \\ b & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Aside: Usually the mathematical form of our model is not right! And we may not know the correct structure!

Formulating the model-fitting problem

- Assume we know good initial conditions $y(0)$ for our model.
- For any θ, let $y(t|\theta)$ be the solution to $\dot{y} = f(y, \theta)$ from the initial conditions
- Intuitively, we want to find θ such that for all t_i

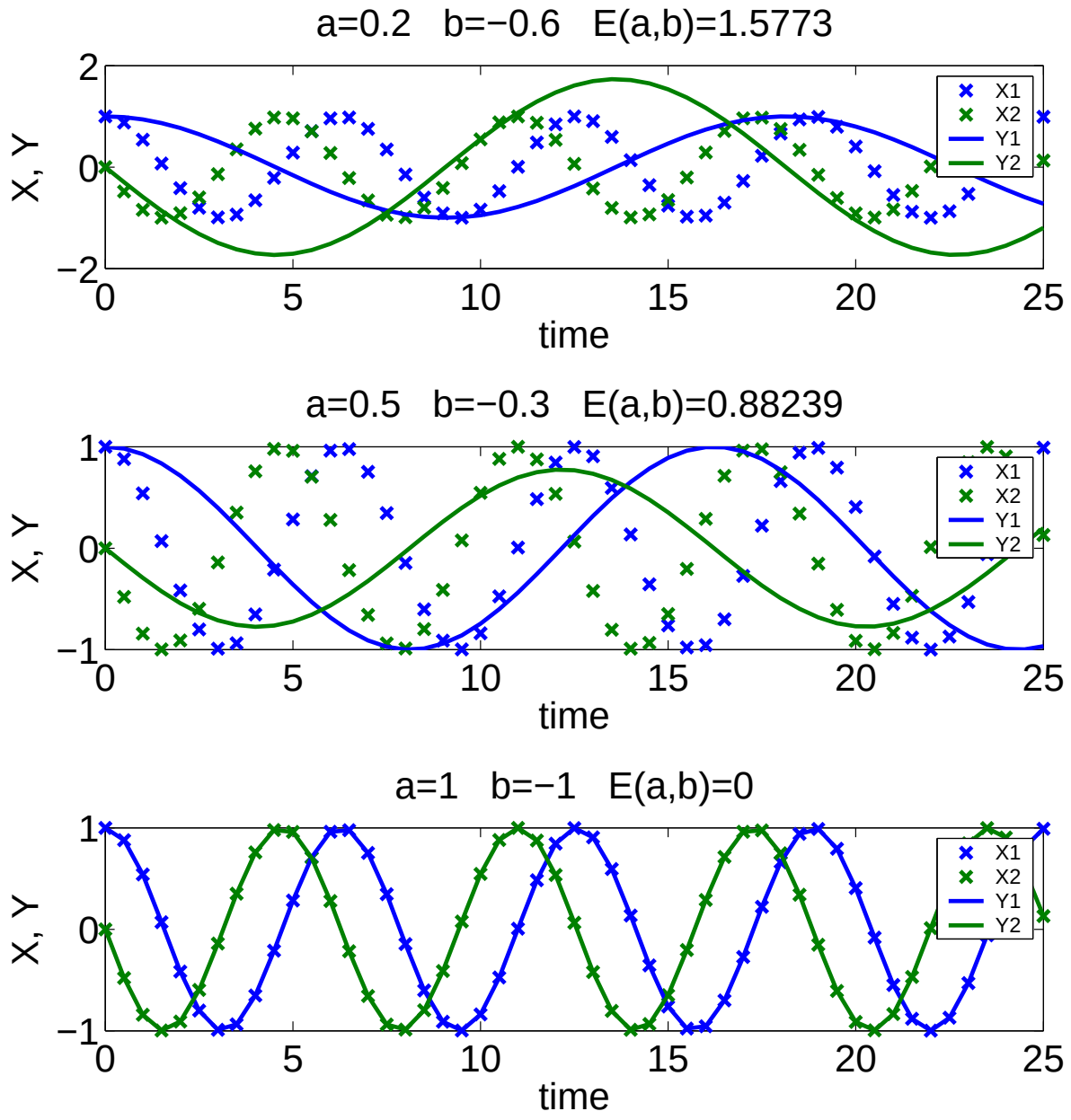
$$x(t_i) \approx y'(t_i|\theta)$$

- Typically, we seek θ that minimizes the mean squared error (MSE):

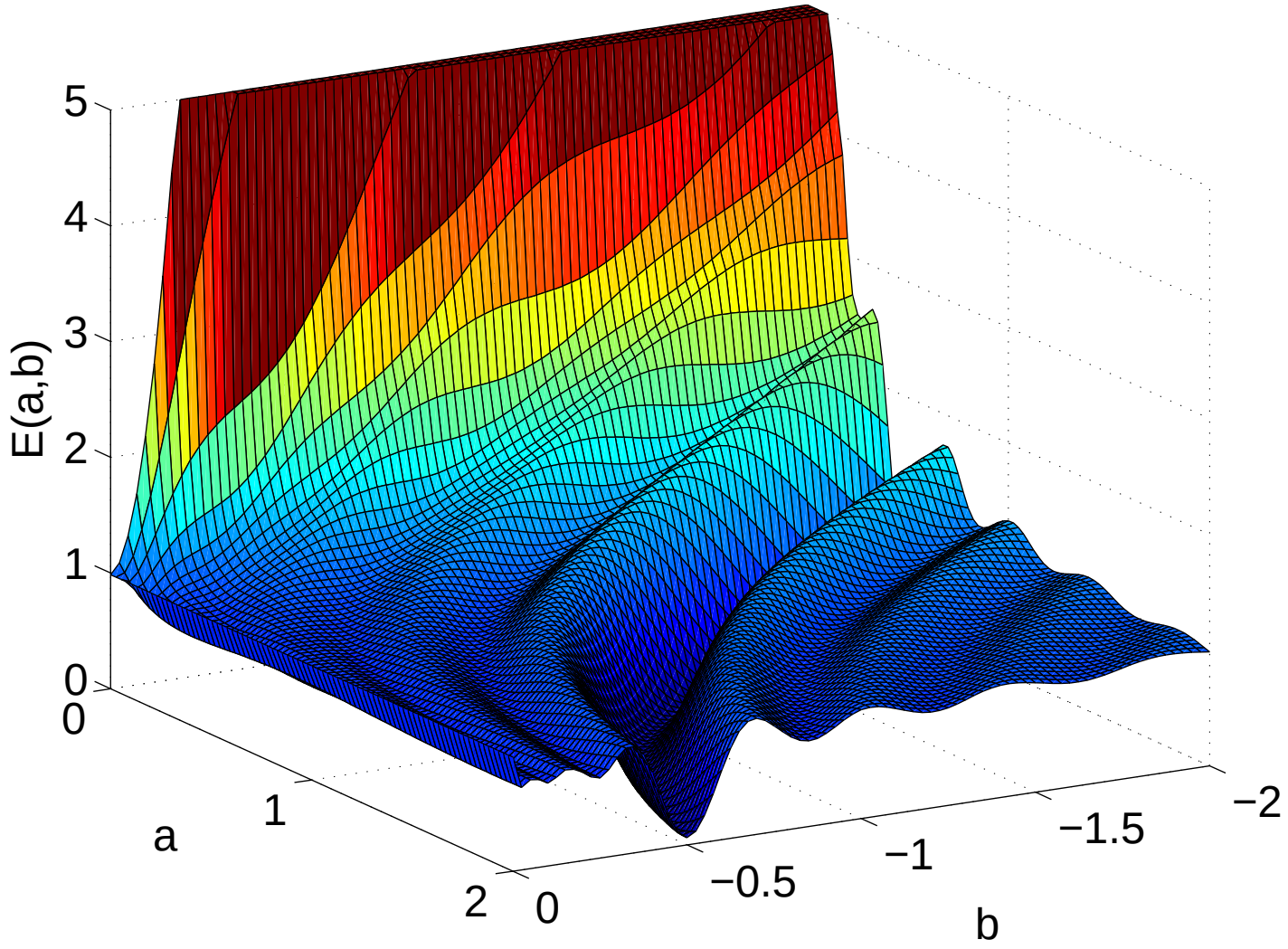
$$\begin{aligned} E(\theta) &= \frac{1}{TM} \sum_{i=1}^{T} \sum_{j=1}^{M} (x_j(t_i) - y'_j(t_i|\theta))^2 \\ &= \frac{1}{TM} \sum_{i=1}^{T} \|x(t_i) - y'(t_i|\theta)\|^2 \end{aligned}$$

Aside: The mean squared error can be justified as a maximum likelihood solution under an i.i.d. $N(0, \sigma)$ observation error model.

Trajectories and errors for several different parameter sets



The error surface



How to minimize MSE?

First principles solution by calculus:

- Write down partial derivatives:

$$\begin{aligned}\frac{\partial E(\theta)}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i}\frac{1}{TM}\sum_{i=1}^{T}\sum_{j=1}^{M}(x_j(t_i) - y'_j(t_i|\theta))^2 \\ &= \frac{1}{TM}\sum_{i=1}^{T}\sum_{j=1}^{M} 2(x_j(t_i) - y'_j(t_i|\theta))\frac{\partial}{\partial \theta_i} y'_j(t_i|\theta)\end{aligned}$$

- Solve system of equations $\frac{\partial E(\theta)}{\partial \theta_i} = 0$

How to minimize MSE?

First principles solution by calculus:

- Write down partial derivatives:

$$\begin{aligned}\frac{\partial E(\theta)}{\partial \theta_i} &= \frac{\partial}{\partial \theta_i}\frac{1}{TM}\sum_{i=1}^{T}\sum_{j=1}^{M}(x_j(t_i) - y'_j(t_i|\theta))^2 \\ &= \frac{1}{TM}\sum_{i=1}^{T}\sum_{j=1}^{M} 2(x_j(t_i) - y'_j(t_i|\theta))\frac{\partial}{\partial \theta_i} y'_j(t_i|\theta)\end{aligned}$$

- Solve system of equations $\frac{\partial E(\theta)}{\partial \theta_i} = 0$

Unfortunately. . .

- We often can't get $y'(t_i)$ in analytic form
- Even if we could, the system of equations may be impossible tos olve analytically / have many zeros

Numerical minimization of MSE

We can fall back to numerical minimization of MSE by standard methods:

- Gradient descent
- Newton's method
- Local search
- Simplex search (e.g. `fminsearch` in Matlab)
- Simulated annealing
- …

The gradient descent algorithm

- Choose initial parameter set θ^0
- Repeat until a stopping criterion reached:
  - Compute $\nabla_{\theta^i} E$
  - $\theta^{i+1} = \theta^i - \alpha_i \nabla_{\theta^i} E$

The α_i are called step sizes.

Under suitable conditions in E and the α_i (Robbins-Monroe), the θ^i converge to a local minimum of E.

How to compute the gradient?

One generic method for computing the partial derivatives of E with respect to the parameters θ_i is to estimate them by finite differences:

- Let $\theta^{i+\Delta} = (\theta_1, \theta_2, \ldots, \theta_i + \Delta, \ldots, \theta_N)$
- Then for suitably small Δ,

$$\frac{\partial E}{\partial \theta_i} \approx \frac{E(\theta^{i+\Delta}) - E(\theta)}{\Delta}$$

Aside: This takes $N + 1$ evaluations of E, which is not terribly efficient.

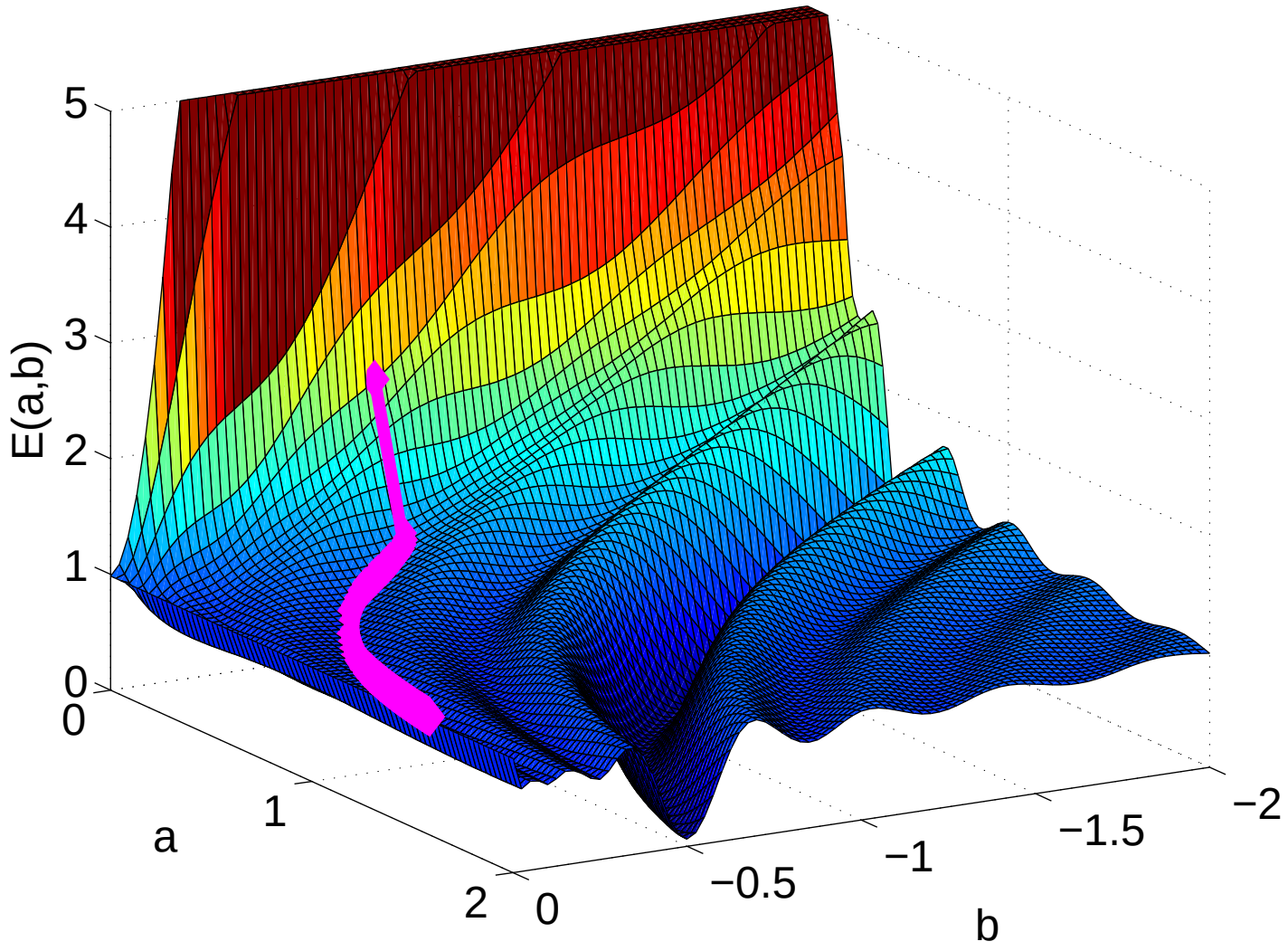
Aside: Sometimes the centered difference is preferred.

$$\frac{\partial E}{\partial \theta_i} \approx \frac{E(\theta^{i+\Delta}) - E(\theta^{i-\Delta})}{2\Delta}$$

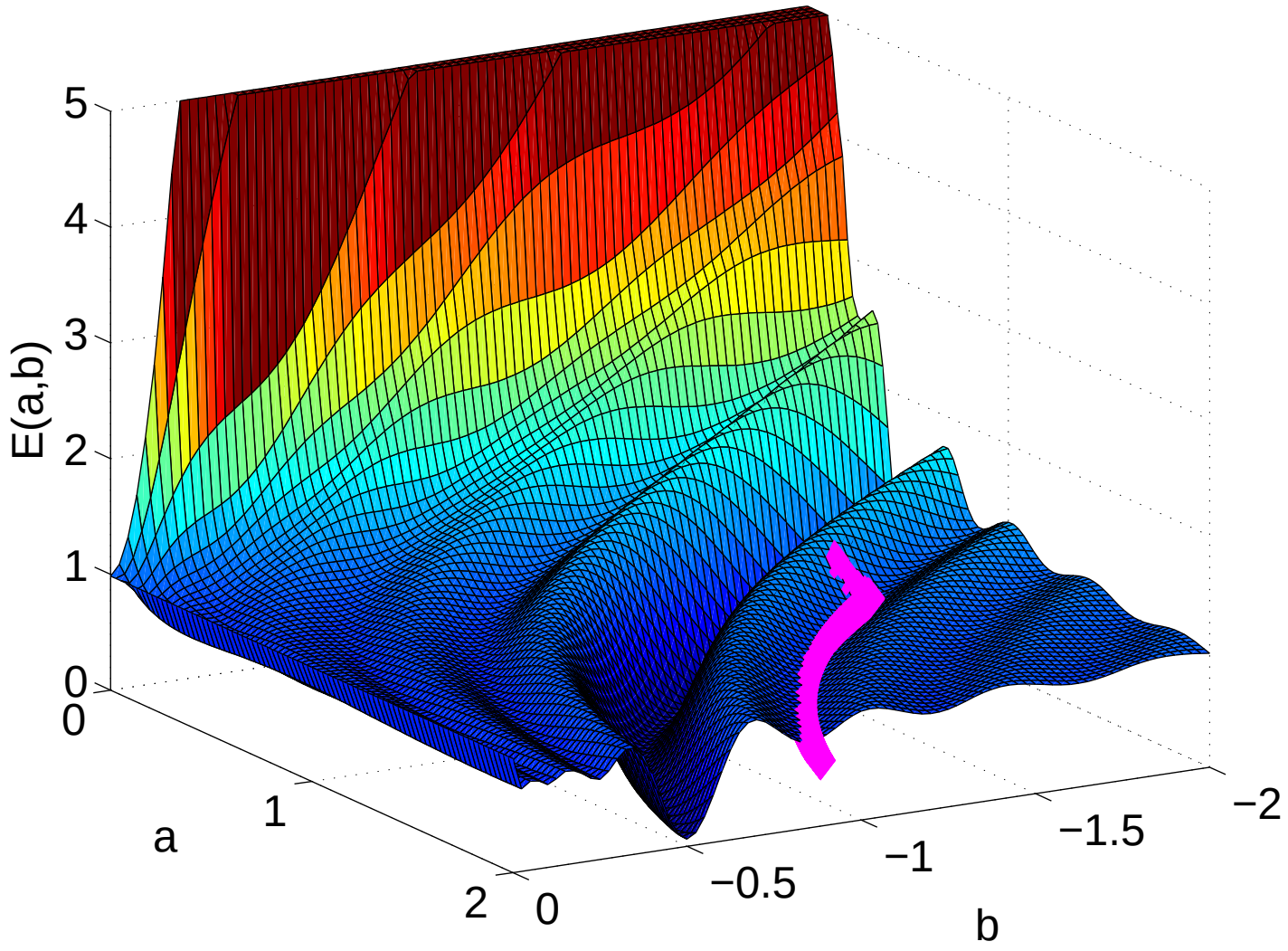
Gradient descent for our example

- Gradient computed by finite differences with $\Delta = 10^{-4}$.
- Step size constant at $\alpha_i = 10^{-2}$.
- 1000 gradient steps
- From several different initial conditions.

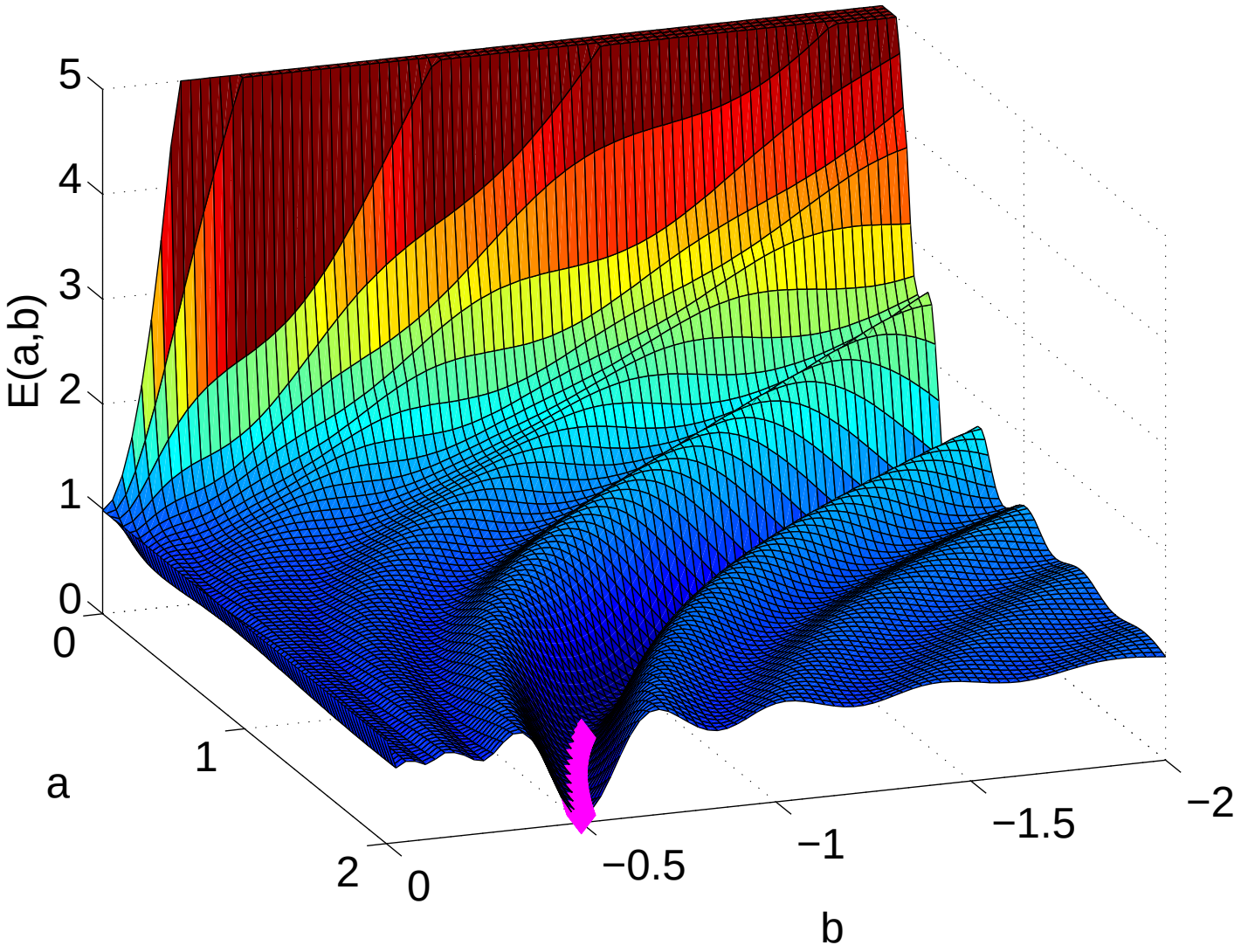
From starting point $a = 0.1, b = -0.7$



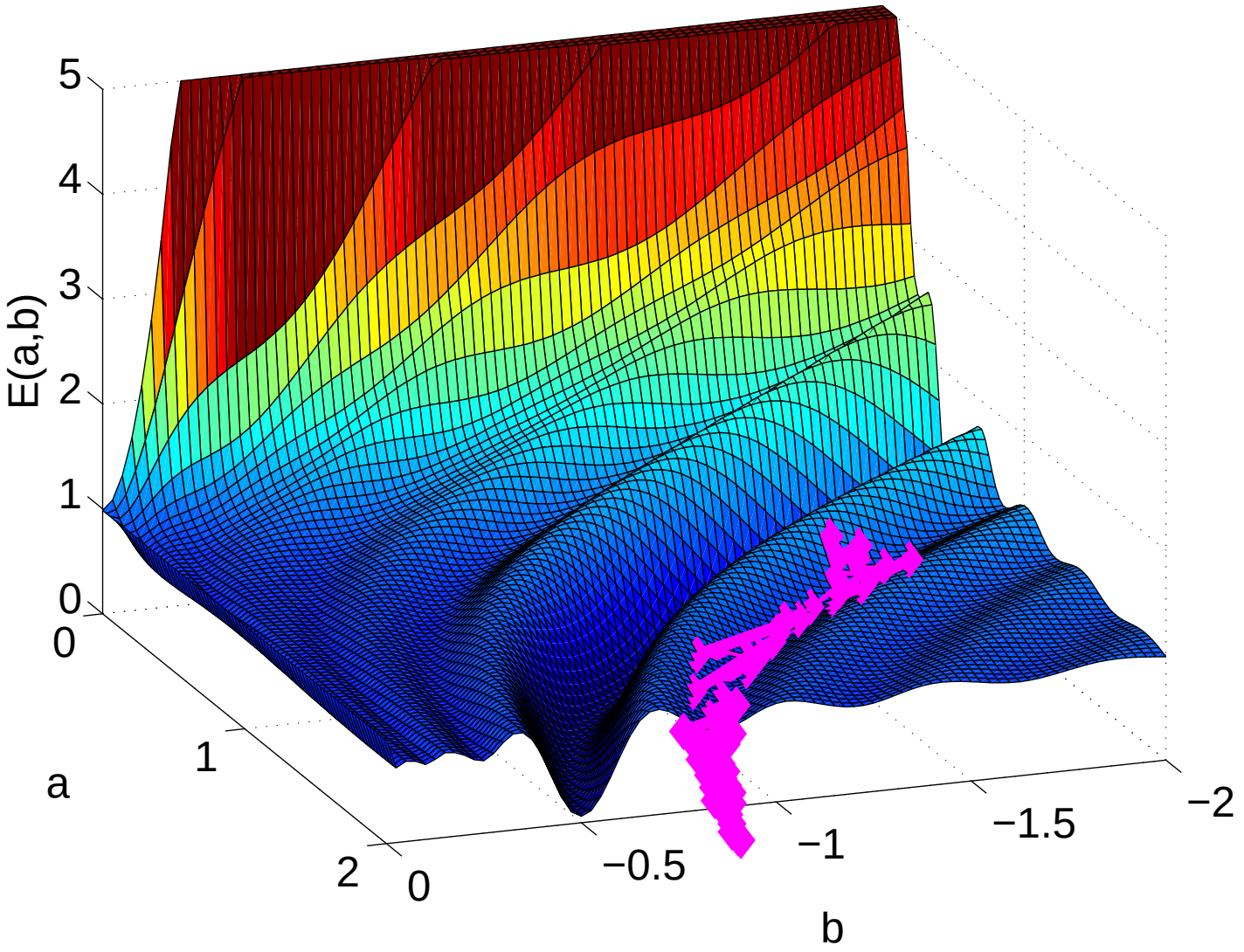
From starting point $a = 1, b = -1.5$



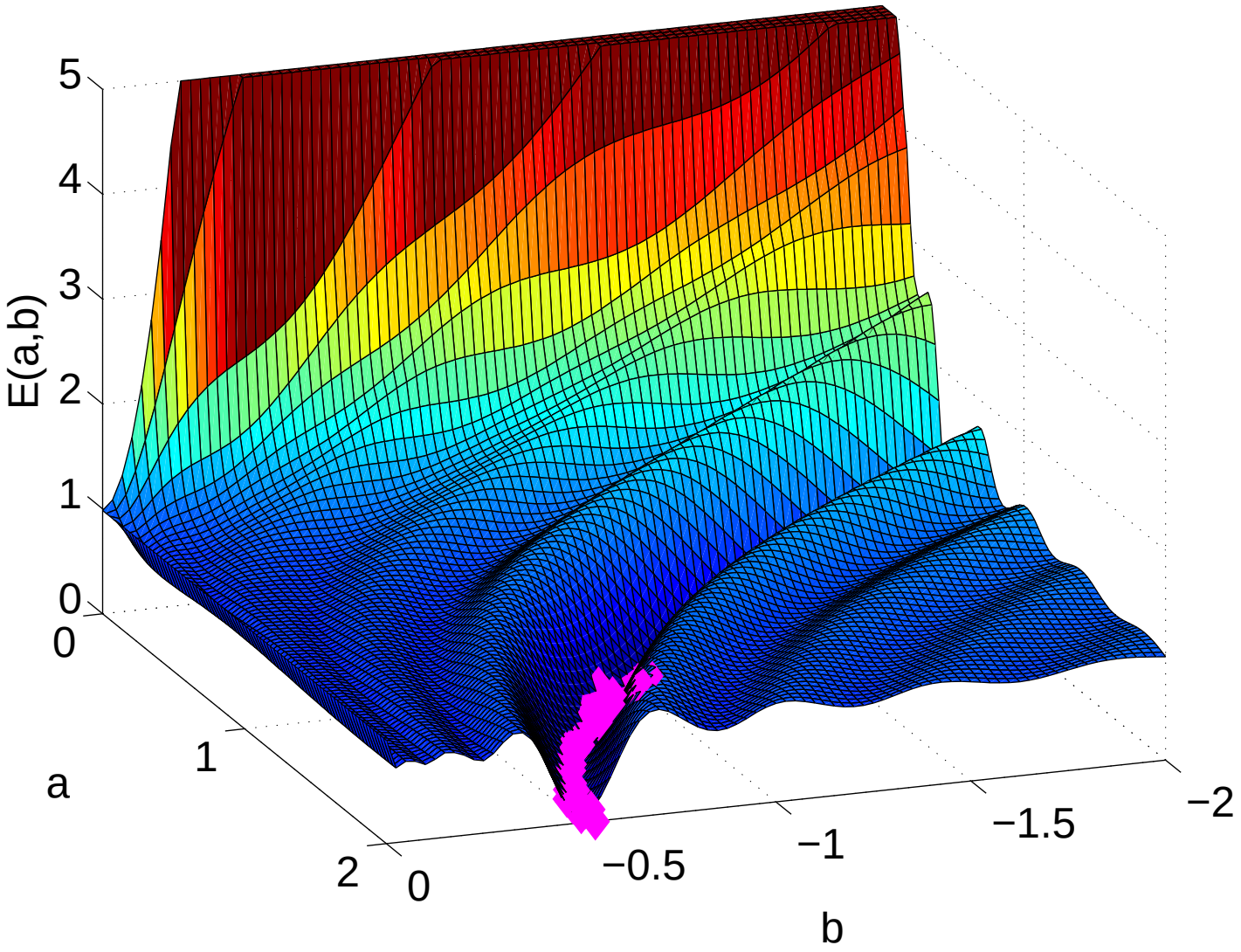
From starting point $a = 2, b = -0.5$



`fminsearch` from starting point $a = 1, b = -1.5$



`fminsearch` from starting point $a = 2, b = -0.5$



Motivating simulated annealing

- Gradient descent, Newton's method, `fminsearch`, local search may all converge to locally optimal solutions that are not globally optimal.
- Simulated annealing attempts to avoid this problem
  - It is a (randomized) local search algorithm
  - It allows "uphill" moves as well as "downhill", in order to escape local basins of attraction

Simulated annealing

We begin with initial parameters $\theta_{current}$.

At each step of the search:

- Generate a parameter set θ' in the "neighborhood" of $\theta_{current}$, often by a random perturbation
- If $E(\theta') < E(\theta_{current})$ then
  - $\theta_{current} = \theta'$

  else with probability $\exp\left(\frac{E(\theta_{current}) - E(\theta')}{T}\right)$
  - $\theta_{current} = \theta'$

T is called the temperature parameter.

The temperature parameter

The value of the temperature parameter affects the behavior of the search:

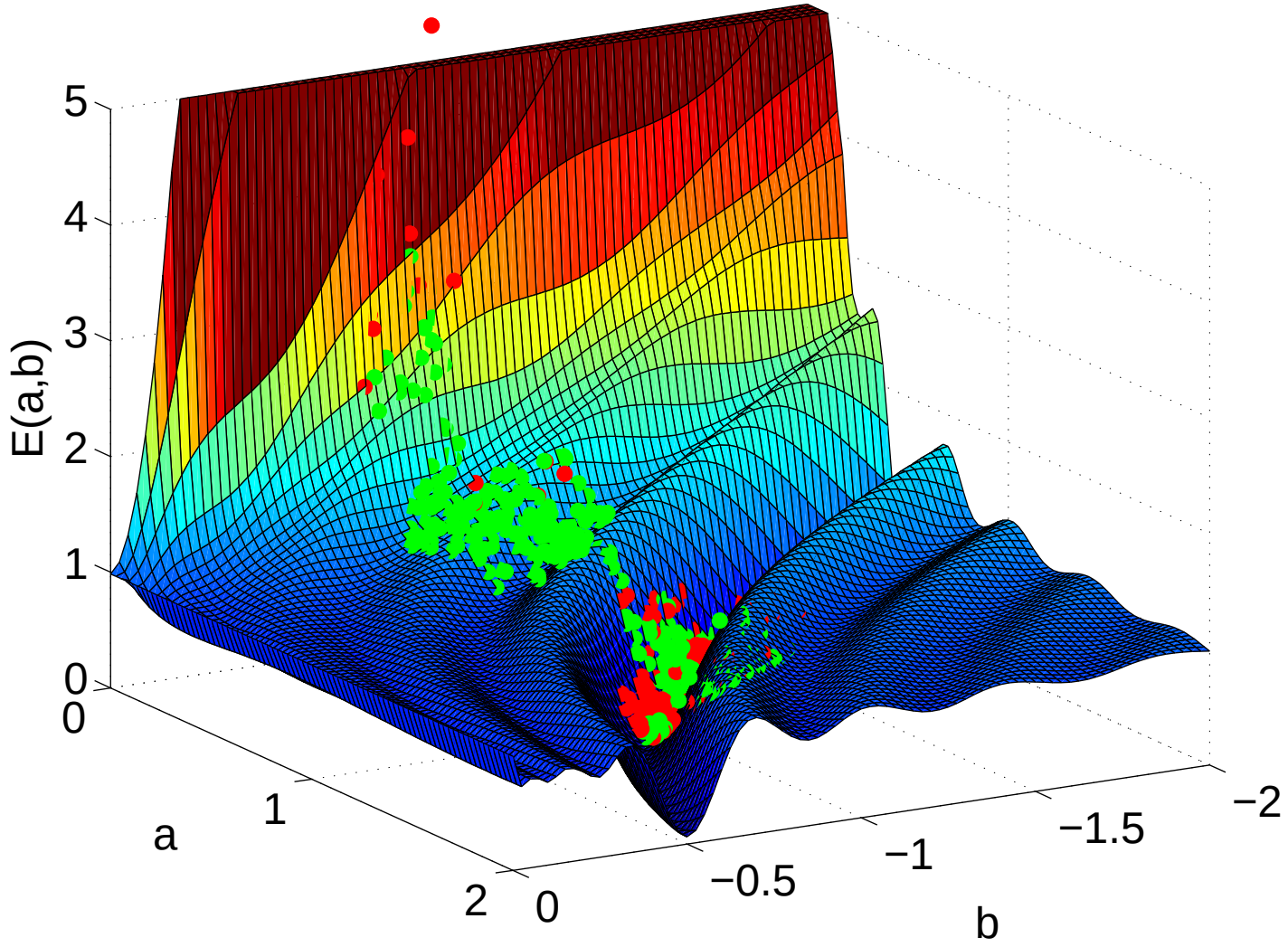
- If $T = +\infty$, then all moves are accepted, and $\theta_{current}$ evolves as a random walk.
- If $T = 0$, then only downhill moves are accepted, and $\theta_{current}$ evolves as with a standard local search.
- If $0 < T < +\infty$, then uphill moves are sometimes accepted, depending on how much worse the neighbor is.

Usually, T is scheduled to approach zero as the search progresses – allowing broad exploration at the start, and converging to local search at the end.

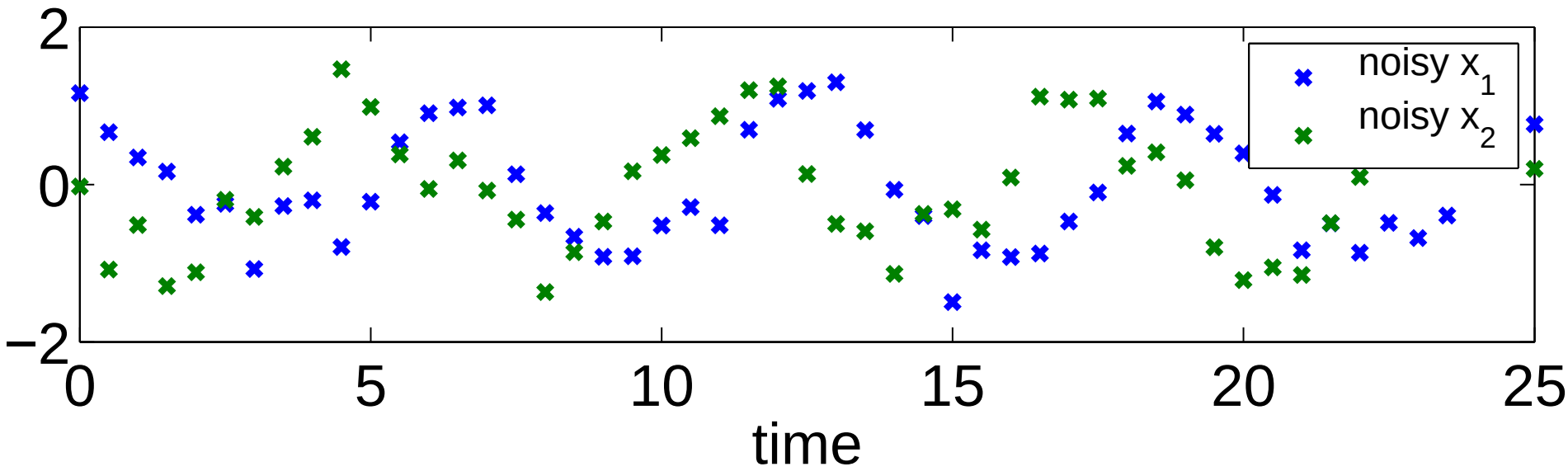
Simulated annealing for our example

- 1000 optimization steps
- Temperature $T_i = \left(\frac{(1000-i)}{1000}\right)^2$
- Random neighbor $\theta'_j = \theta_{current,j} + N(0, 0.03)$

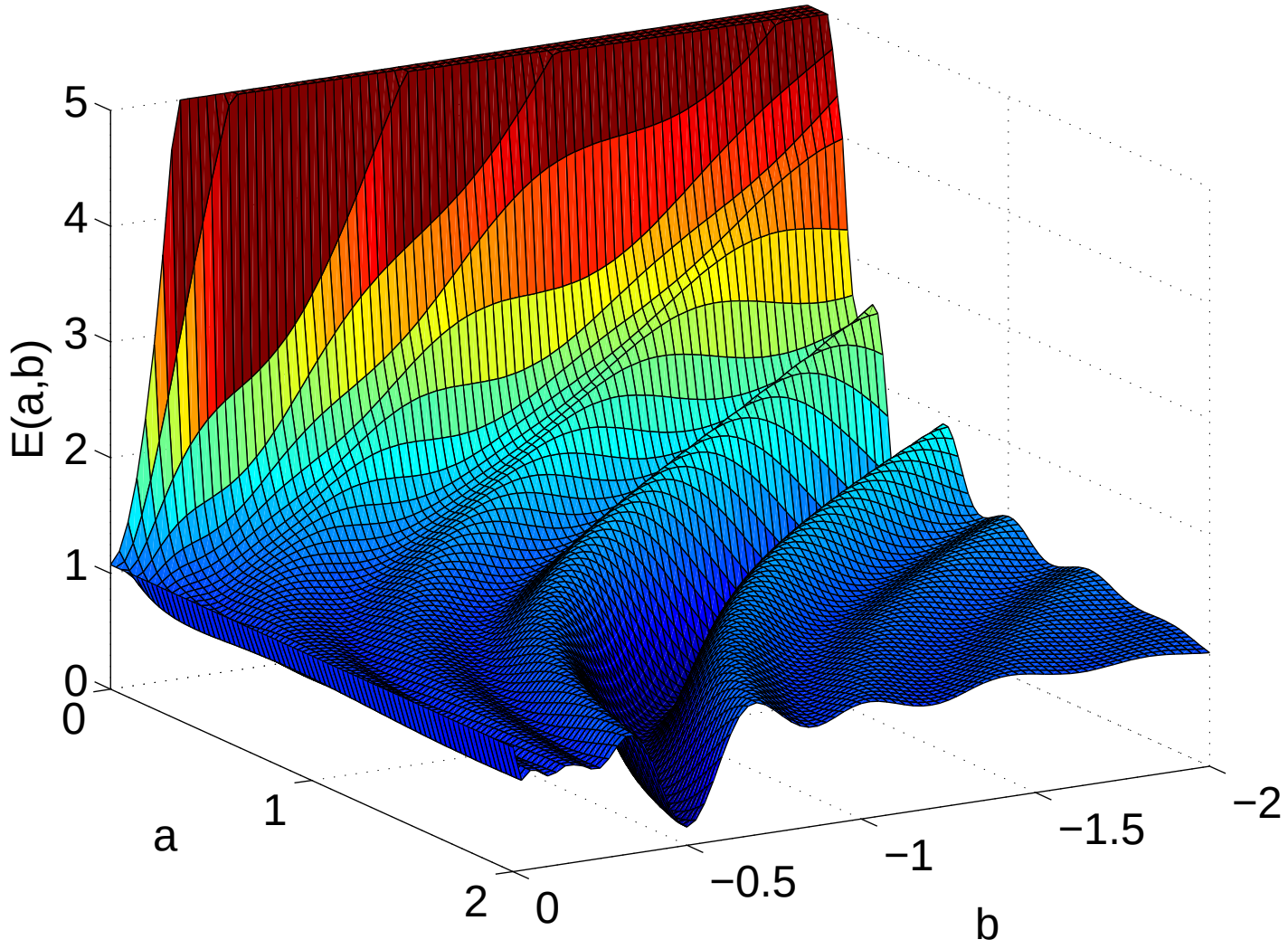
Simulated annealing results



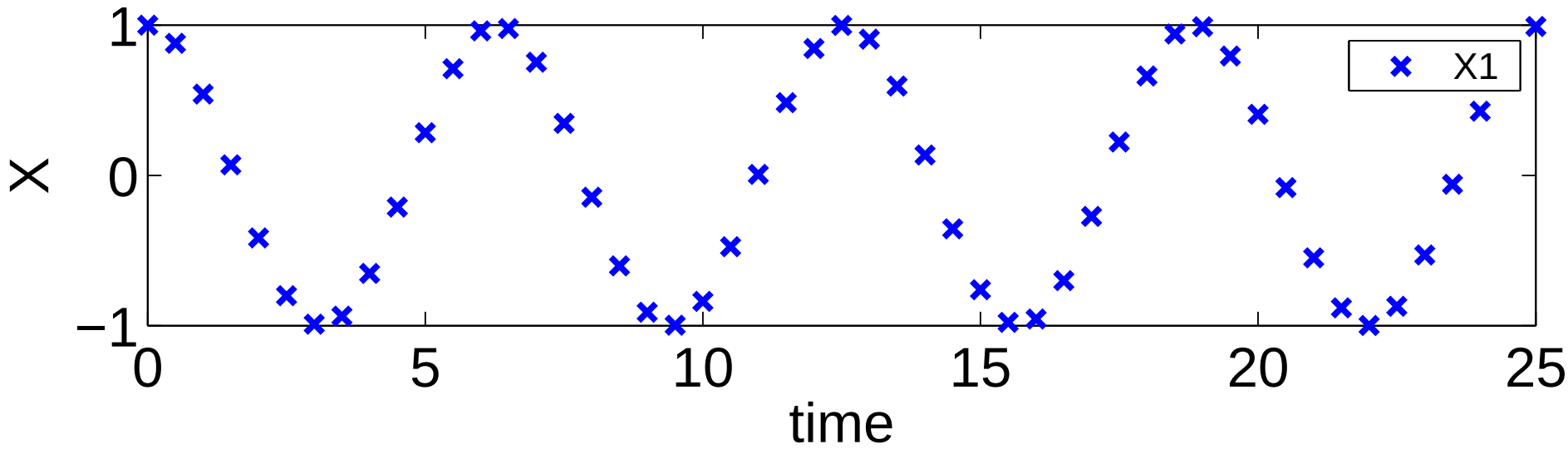
What if the data are noisy?



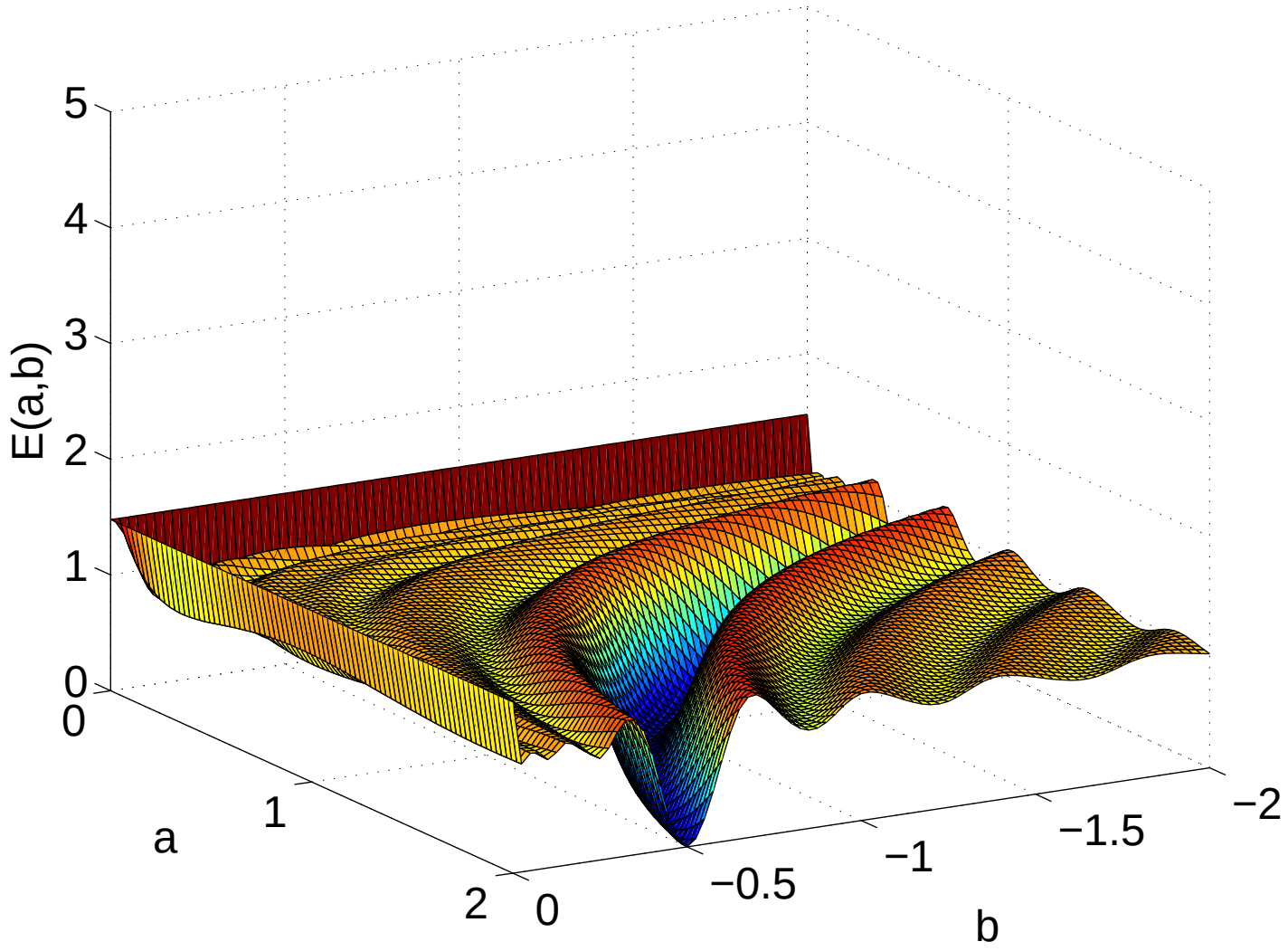
Error surface is very little changed!



What if we only observe x_1?



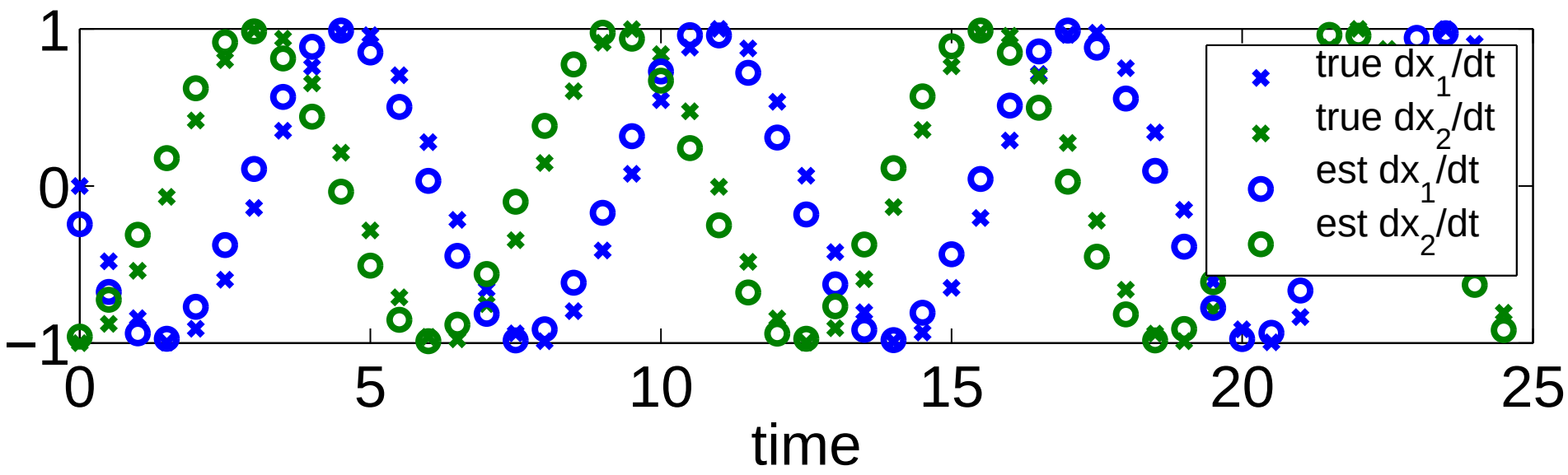
The error surface



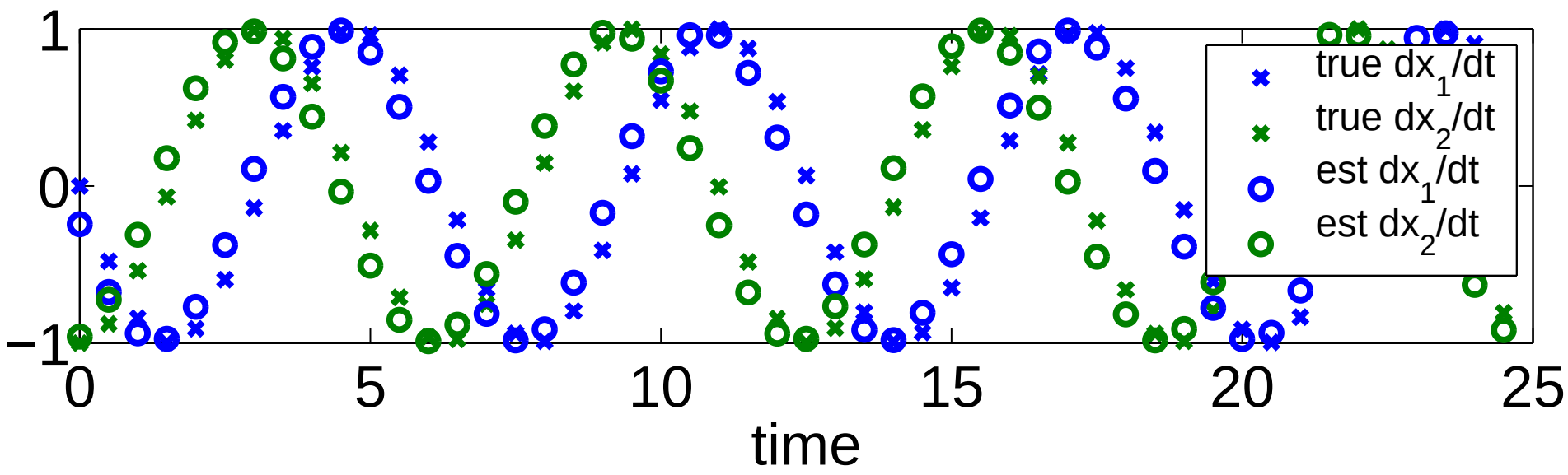
Structure Learning & Regression Approach

Done on blackboard!

Finite difference estimates of derivatives



Finite difference estimates of derivatives

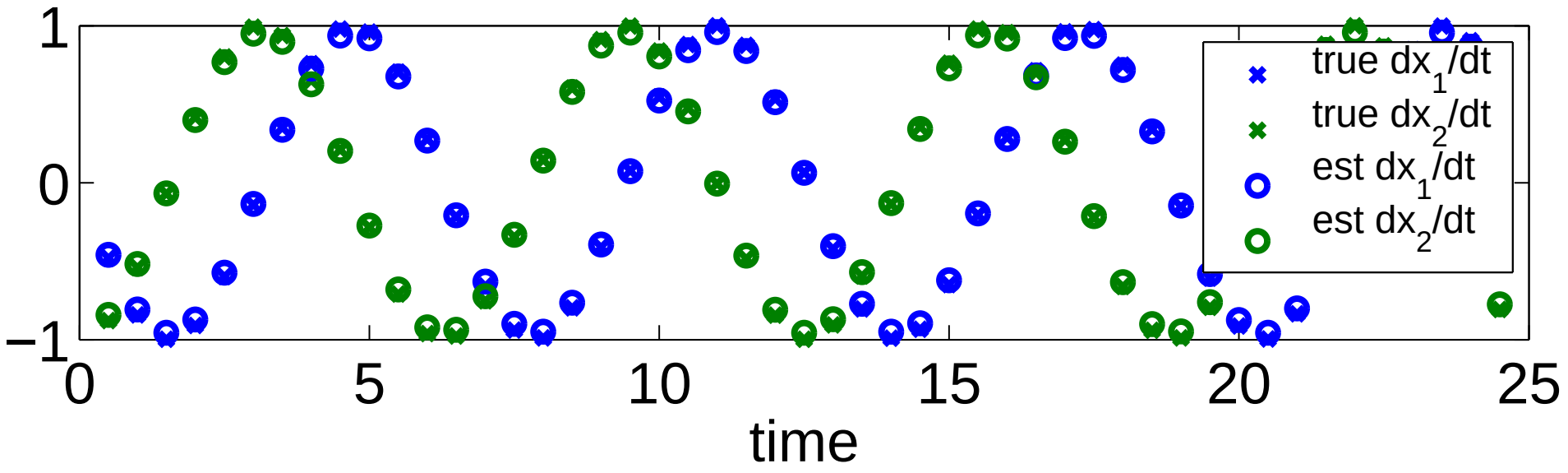


Minimizing

$$\frac{1}{TM}\sum_{i=1}^{T}\left\|\begin{bmatrix}\hat{\dot{x}}_1(t_i)\\ \hat{\dot{x}}_2(t_i)\end{bmatrix}-\begin{bmatrix}0 & a\\ b & 0\end{bmatrix}\begin{bmatrix}x_1(t_i)\\ x_2(t_i)\end{bmatrix}\right\|^2$$

can be done analytically, yielding $a = 0.9597$ and $b = -0.9581$.

Central difference estimates of derivatives

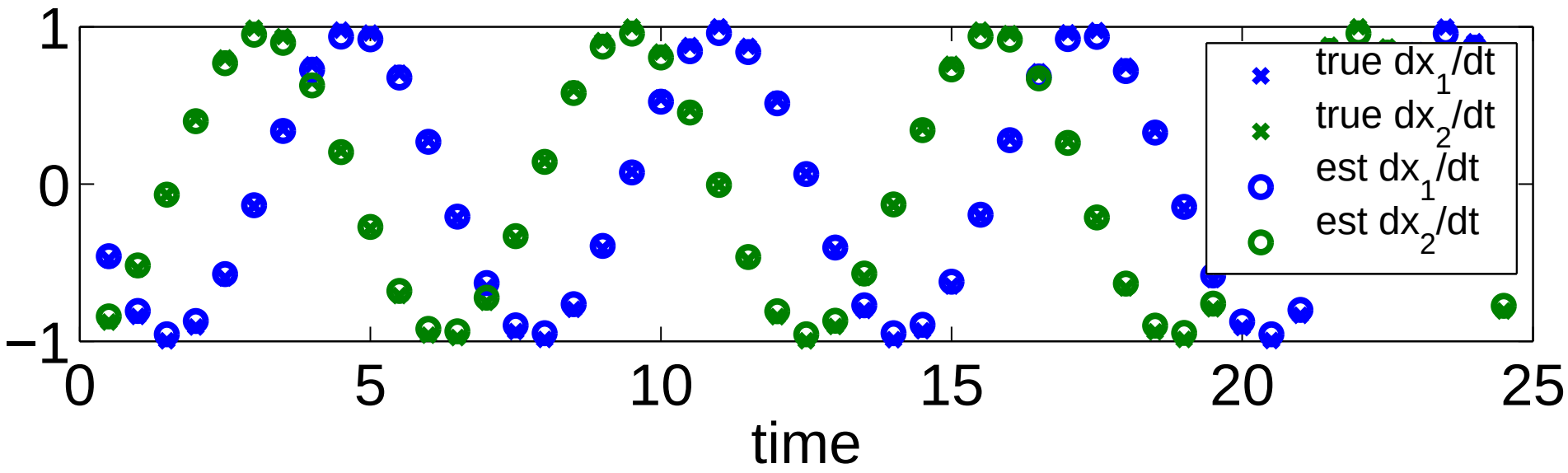


Minimizing

$$\frac{1}{TM}\sum_{i=1}^{T}\left\|\begin{bmatrix}\hat{\dot{x}}_1(t_i)\\ \hat{\dot{x}}_2(t_i)\end{bmatrix}-\begin{bmatrix}0 & a\\ b & 0\end{bmatrix}\begin{bmatrix}x_1(t_i)\\ x_2(t_i)\end{bmatrix}\right\|^2$$

yields $a = 0.9589$ and $b = -0.9589$.

Fitting full interconnect matrix

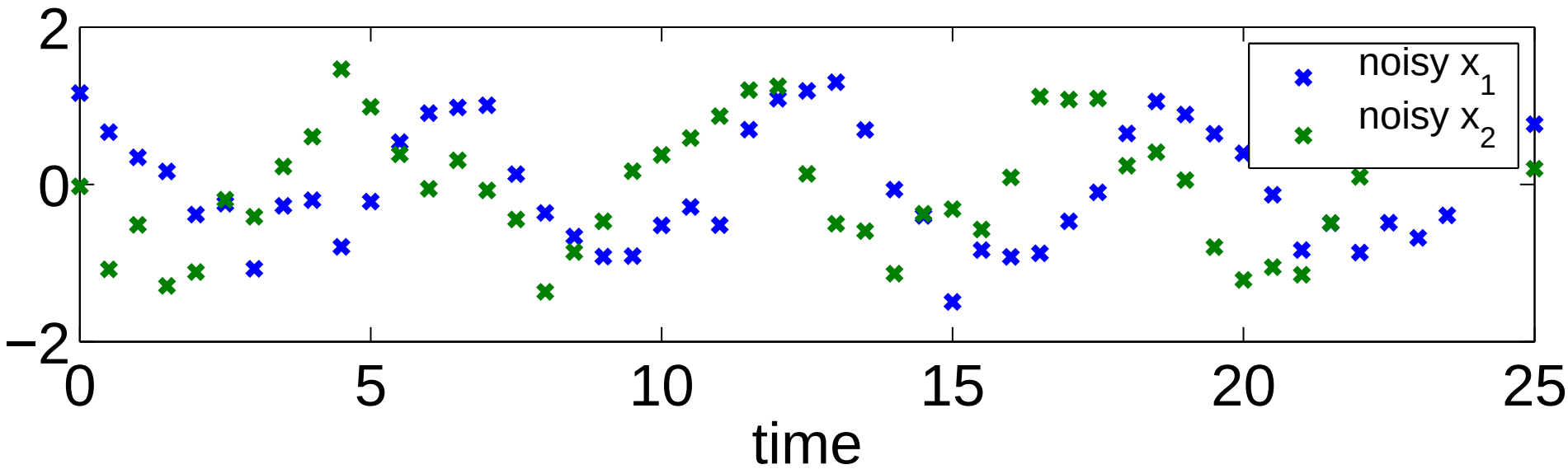


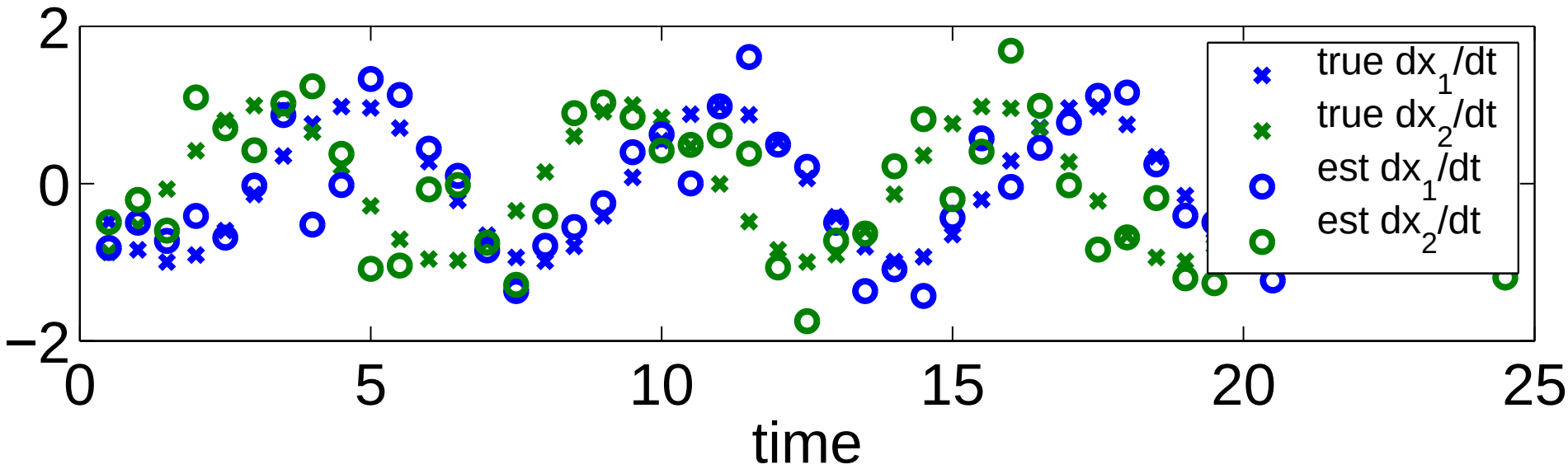
Minimizing

$$\frac{1}{TM}\sum_{i=1}^{T}\left\|\begin{bmatrix}\hat{\dot{x}}_1(t_i)\\ \hat{\dot{x}}_2(t_i)\end{bmatrix}-\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}x_1(t_i)\\ x_2(t_i)\end{bmatrix}\right\|^2$$

yields $A = \begin{bmatrix}-0.00003 & 0.9589\\ -0.9589 & -0.00002\end{bmatrix}$.

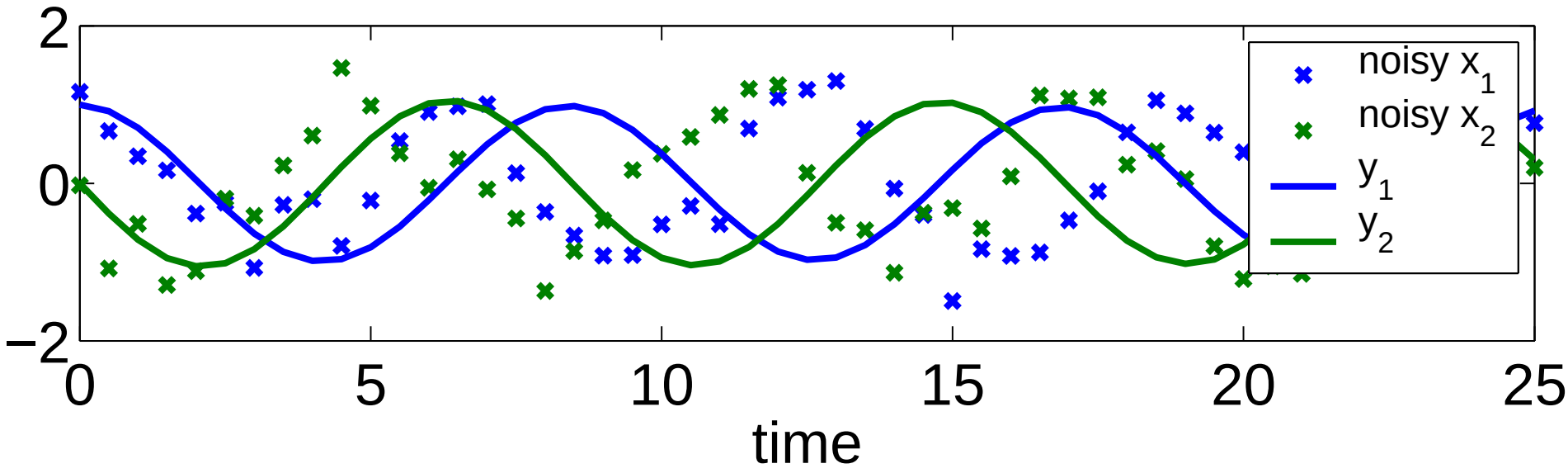
What if the data are noisy?





Fitting a full interconnect matrix

Yields $A = \begin{bmatrix} -0.0337 & 0.6994 \\ -0.7886 & 0.0298 \end{bmatrix}$



Functional data analysis

Done on board!

Conclusions

- Usually, we cannot solve parameter fitting problems analytically
- Numerical methods for fitting to trajectories may be subject to local optimality (grad. descent, Newton, fminsearch, local search) or computationally intensive (simulated annealing).
- However, they apply most generally, including when some model variables are not observed and/or data are noisy.
- Regression-based methods (including functional data analysis) are computationally efficient, if not analytically solvable.
- However, they apply only when all model variables are observed (noise allowed), and when the model is simulated, it may not match the observed trajectory well.