Math 493 Project 2

Aneesh & JJ & Matt

October 22, 2018

Trajectory Matching with Markov Chain Monte Carlo

Since the ODE function given could be easily solved, the problem became estimating C and β in $Ce^{-\beta t}$.
We first initialized the Markov Chain Monte Carlo with the following parameters:

Variable	Value
C_0	0
β_0	0
Data Standard Deviation σ	0.03
Guess Jump	0.01
Burn Time	100
Iteration Limit	5000

Trajectory Matching with Markov Chain Monte Carlo

For each iteration step:

1. make a new guess for $[C_n, \beta_n] = [C_{n-1}, \beta_{n-1}] + D * \mathrm{randn}$
2. calculate the SSE using $[C_n, \beta_n]$
3. evaluate the ratio $e^{\frac{-\mathrm{SSE}_{n-1} + \mathrm{SSE}_n}{2\sigma^2}}$
4. if a random number between 0 and 1 is smaller than the ratio, the new guess is accepted

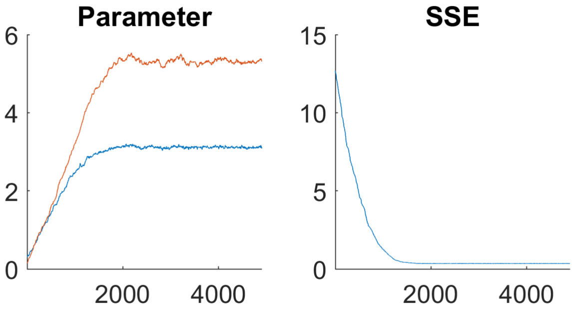
Trajectory Matching with Markov Chain Monte Carlo

With the smoother version of the data, we obtained the following result:

C	3.1792
β	5.5447

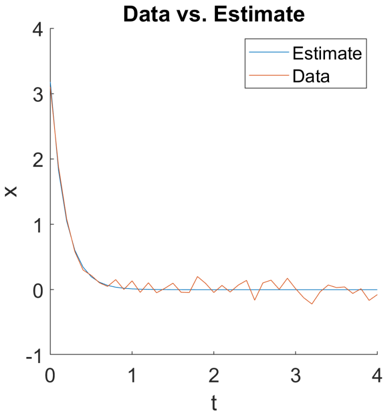
Trajectory Matching with Markov Chain Monte Carlo

We can also observe the convergence in the following figure:



Trajectory Matching with Markov Chain Monte Carlo

We can also see how our model compares with the data:



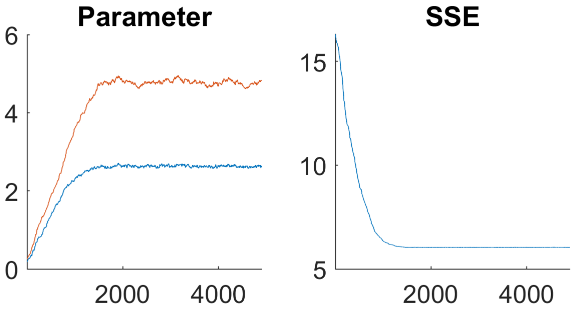
Trajectory Matching with Markov Chain Monte Carlo

With the noisy version of the data, we obtained the following result:

C	2.6331
β	4.8364

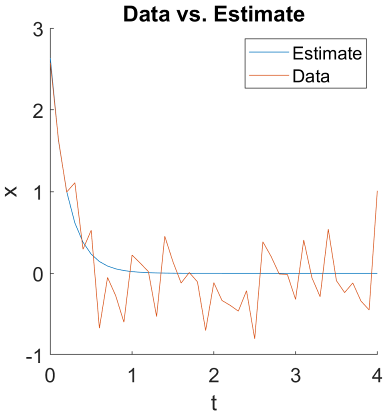
Trajectory Matching with Markov Chain Monte Carlo

We can also observe the convergence in the following figure:



Trajectory Matching with Markov Chain Monte Carlo

We can also see how our model compares with the data:



Trajectory Matching with Markov Chain Monte Carlo

We explored with the guess jump a little bit and obtained the following result:

Guess Jump	C	β	Converged Step	Converged SSE
0.005	3.1656	5.4653	4326	0.3466
0.01	3.1200	5.3525	2073	0.3446
0.02	3.0976	5.2793	755	0.3458
0.1	3.1783	5.5051	99	0.3482

Gradient Matching with Smoothing

We can smooth the data using collocation. The basis ϕ could be created using *spcol* function in matlab. Then the coefficients $\mathbf{c} = (\phi^T\phi)^{-1}(\phi y)$, where y is the data given.
The gradient matching method is essentially using Gauss-Newton method to minimize the integrated SSE:

$$\text{ISSE} = \sum_{i=1}^{n} W_i[Dx(\hat{t_i}) - f(x(\hat{t_i}), \theta)]^2$$

where $x(\hat{t_i}) = \phi\mathbf{c}$ is the smoothed data using collocation.
For this problem, the ISSE could be simplied to

$$\text{ISSE} = \sum_{i=1}^{n} W_i[D\phi(t_i)\mathbf{c} + \beta\phi(t_i)\mathbf{c}]^2$$

Gradient Matching with Smoothing

We set $W_i = 1$ for all $i = 1, \ldots, n$. Then the Jacobian becomes $\hat{x}$. For each iteration step, we perform the following:

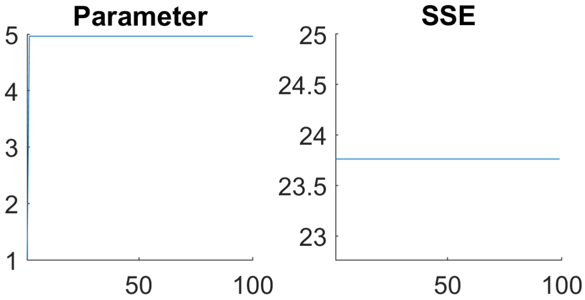
$$
\begin{aligned}
H &= J^T J \\
g &= J^T (D\hat{x} + \beta\hat{x}) \\
\beta_n &= \beta_{n-1} - H^{-1} g
\end{aligned}
\tag{1}
$$

Due to the fact that ISSE does not take $x(t_0)$ into account, we use the initial value of the smoothed data as $x(t_0)$ for the following parts.

Gradient Matching with Smoothing

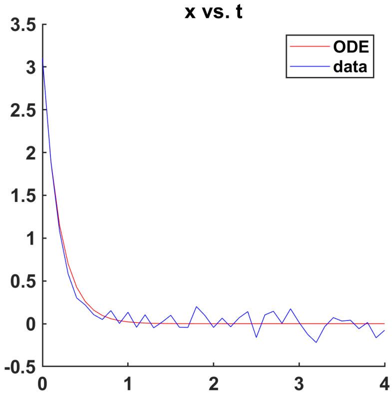
With the smoother version of the data, the estimated $\beta = 4.962410410929266$.

We can also observe the convergence in the following figure:



Gradient Matching with Smoothing

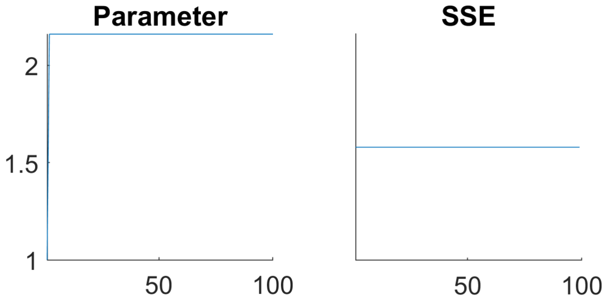
We can also see how our model compares with the data:



Gradient Matching with Smoothing

With the noisy version of the data, the estimated $\beta = 2.160787567688269$.

We can also observe the convergence in the following figure:



Gradient Matching with Smoothing

We can also see how our model compares with the data:

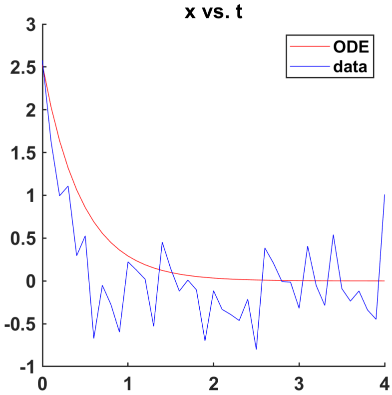


Figure: The model we obtained is the line in blue. We can see that our model captures the trend of the data.

Gradient Matching with Smoothing

When the method is used without smoothing, it fails catastrophically. The parameter blows up.

Gradient Matching with Smoothing

When we smooth the data with collocations, we try different number of polynomials allowed for the basis and obtained the following result:

Number of Polynomials	β
9	4.97277486579170
10	4.94682728787471
11	4.96241041092927
12	4.96995381483431
13	4.97336941885107

Therefore, the number of polynomials in the collocation basis does not affect the results drastically.

Integral Matching

Our goal is to fit the following system to the data given by y_i

$$x'(t) = -Bx$$
$$x(0) = C$$

Since we can solve this system directly, we see that

$$\mathrm{ISSE}(B, C) = \sum_{i=1}^{N} (Ce^{-Bt} - y_i)^2$$

Using the Gauss-Newton Method, we let

$$J(B, C) = \begin{pmatrix} -C\mathbf{t}e^{-B\mathbf{t}} \\ e^{-B\mathbf{t}} \end{pmatrix}$$

Integral Matching Results

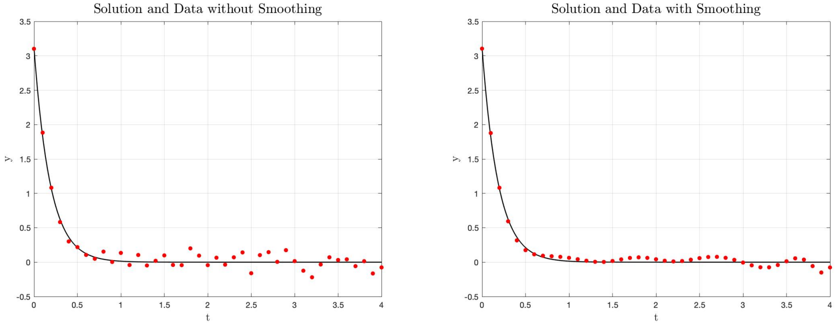


Figure: Integral matching fits the smoothed data in both cases

Convergence of Parameters

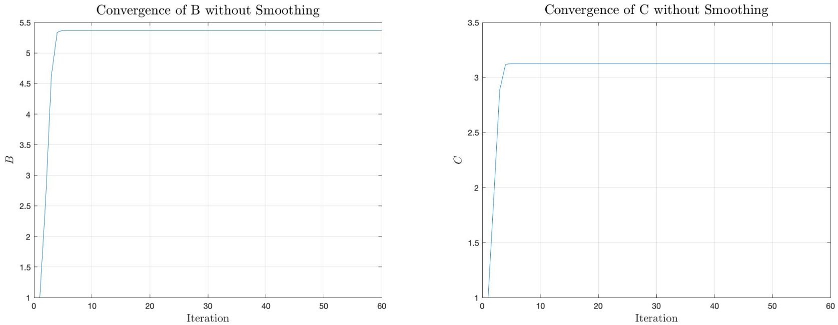


Figure: Convergence of Parameters without Smoothing. Both parameters converge in ≈ 6 iterations

Convergence of Parameters

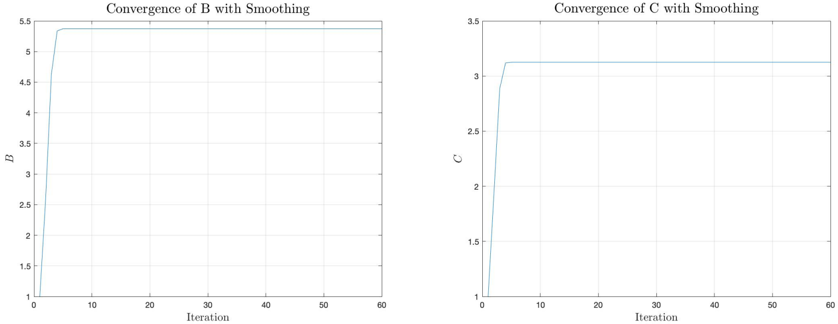


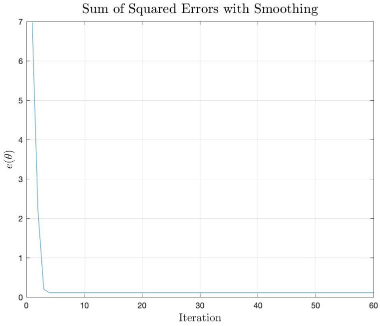
Figure: Convergence of Parameters with Smoothing. Both parameters converge in ≈ 6 iterations

SSE Error with Integral Matching

c	[1,...,1]
C	1
β	1

β	5.33
C	3.08

SSE Error with Integral Matching



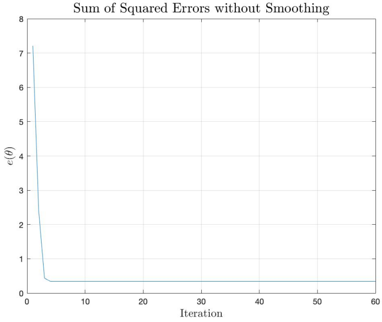


Figure: SSE Error Convergence

Integral Matching with Noisy Data

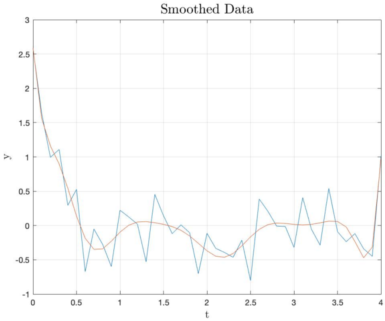


Figure: Noisy Data with Smoothing

Integral Matching with Noisy Data

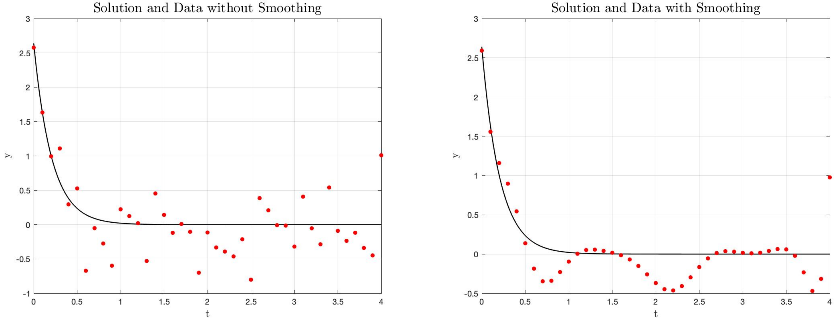


Figure: Integral Matching with Noisy Data

Convergence of Parameters

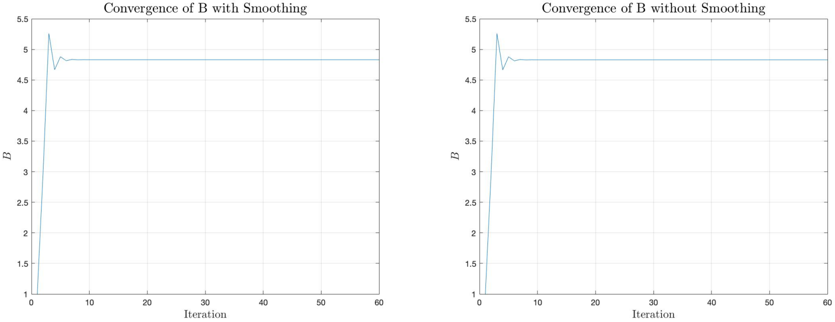


Figure: Convergence of the parameter B with noisy data

Convergence of Parameters

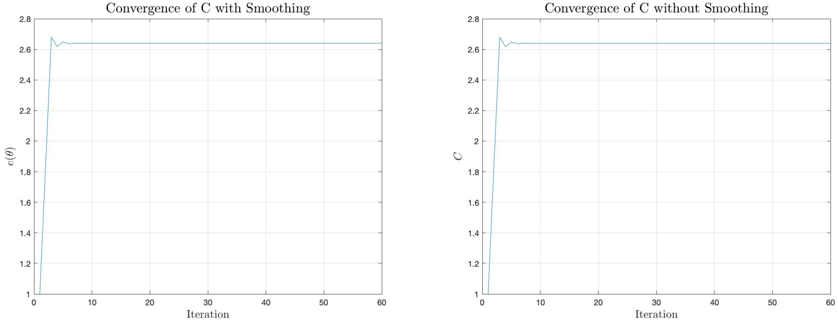


Figure: Convergence of the parameter C with noisy data

Integral Matching Error with Noisy Data

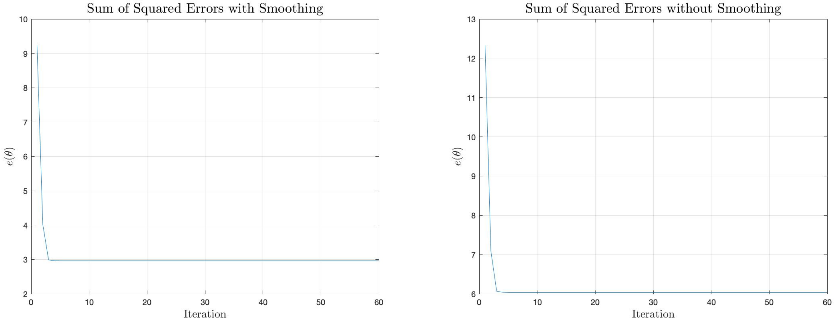


Figure: Convergence of the parameter C with noisy data

Parameter Cascading with Smoothing

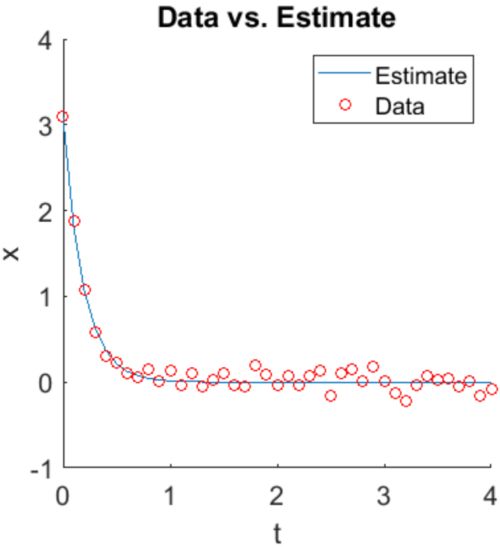


Figure: Cascading Parameter Estimation

Parameter Cascading with Smoothing

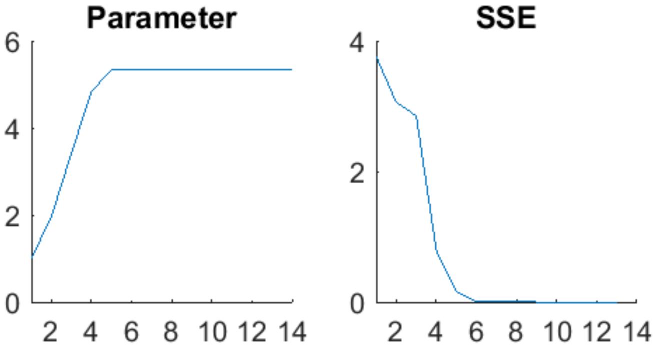


Figure: Cascading Parameter Convergence

Parameter Cascading with Smoothing

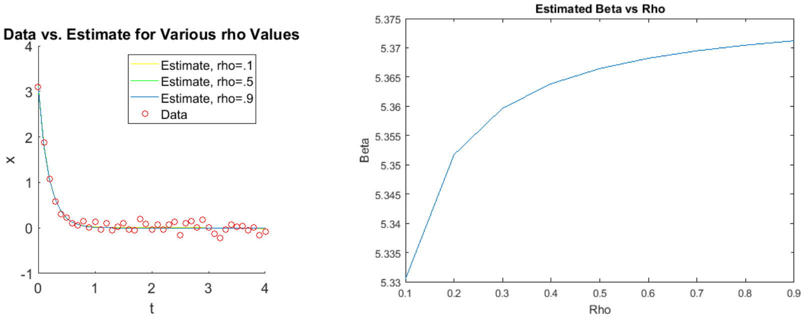


Figure: Cascading Parameter Convergence