Sparse Parameter Learning for Model Selection via Projected Block Proximal Gradient Descent

MATH 6490: Optimization Methods for Data

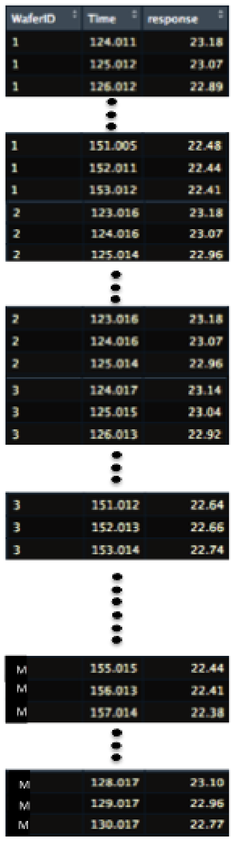
Andrés F. Vargas

1 Introduction

In this report, I describe an R implementation of block projected proximal gradient descent to sparsely learn the parameters of multiple data-generating functions, the prior distribution of these data generating functions, and then use the sparse representation for model selection. The data utilized in this project was collected during a real-world semiconductor manufacturing process, and so is proprietary. As such, although the results in Section 6 were obtained using this data, the raw data itself could not legally be included in the source files.

2 Data Structure and Assumptions

Figure 1: Structure of the Data. First column is "WaferID", second columns is "Time", third column is "response". We see that each unique wafer id has its own time series.



WaferID	Time	response
1	124.011	23.18
1	125.012	23.07
1	126.012	22.89
⋮		
1	151.005	22.48
1	152.011	22.44
1	153.012	22.41
2	123.016	23.18
2	124.016	23.07
2	125.014	22.96
⋮		
2	123.016	23.18
2	124.016	23.07
2	125.014	22.96
3	124.017	23.14
3	125.015	23.04
3	126.013	22.92
⋮		
3	151.012	22.64
3	152.013	22.66
3	153.014	22.74
⋮		
M	155.015	22.44
M	156.013	22.41
M	157.014	22.38
⋮		
M	128.017	23.10
M	129.017	22.96
M	130.017	22.77

To understand the data structure, figure 1 says a thousand words. In the figure, we see that each unique wafer id has its own time series. I represent the time series for the l^{th} wafer mathematically as $\{t, z_l\}_{t \in t_l}$, for all $l \in \{1, ..., M\}$, where z_l is a vector containing the responses in wafer l and t_l is the set of times in wafer l. Moving toward regression, we think of z_l as a realization of a multivariate random variable $Z_l | S_l \sim N(\overrightarrow{\alpha}(t_l, S_l), \sigma^2)$, where $S_l \sim N(\mu_S, diag(\sigma_S)^2)$, σ_S is a vector containing the standard deviations of the elements of S_l for all l, and $\vec{\alpha}$ is defined, for any vector of parameters s and set of times for N observations $t = \{t_1, t_2, ..., t_N\}$, by

$$\vec{\alpha}(t,s) := \begin{pmatrix} \alpha(t_1, s) \\ \alpha(t_2, s) \\ \vdots \\ \alpha(t_N, s) \end{pmatrix} \; ; \tag{1}$$

where α is a deterministic function of time and the parameters contained in s.

In other words, we assume the relationship between "Time" and "response" in every wafer can be modeled by the class of functions α, but the parameters of the function, which is contained in the second argument s, will change from wafer to wafer. We then assume that these parameters all come from the same underlying $N(\mu_S, diag(\sigma_S)^2)$ distribution (this is the prior) and that the error between the predicted responses and the true responses is distributed as $N(0, \sigma^2)$. We assume that each wafer is independent of all possible interactions between the other wafers ; mathematically, these independences are encoded by the statement

$$(Z_l, S_l) \perp \bigcap_{j \in B} (Z_j, S_j) \; , \forall B \subseteq A_l, \forall l \in \{1, ..., M\} \; .$$

Here, A_l is the power set of $\{1, 2, ..., l-1, l+1, l+2, ..., M\}$ and $\perp$ means "is independent of".

3 Task

We seek to learn σ, μ_S, and σ_S, and α and its parameters. We put s_l^* in lowercase because it is a realization of the random variable S_l. If α is known, we can use non-linear bayesian regression to maximize the joint probability $P(z_1, ..., z_M, s_1, ..., s_M) = P(z_1, s_2 | z_3, ..., z_M, s_1, ..., s_M) P(z_3, ..., z_M, s_1, ..., s_M)$. The first term in the product is equal to $P(z_1, s_1)$ due to the independence assumptions and can then be decomposed $P(z_1, s_1) = P(z_1|s_1)P(s_1)$ by Bayes Theorem. Similar manipulations lead to the second term decomposing into $\prod_{l=2}^{M} P(z_l|s_l)P(s_l)$. We have thus decomposed the joint probability into $\prod_{l=1}^{M} P(z_l|s_l)P(s_l)$. We equivalently minimize the negative logarithm of this quantity, enforcing the physical constraints $\omega_l \geq 0$ and $\phi_l \in [-\pi/2, \pi/2] \; , \forall l \in \{1, ..., M\}$. This can be formulated as the optimization problem

$$\begin{array}{ll} \min_{s_1,...,s_M,\sigma,\mu_S,\sigma_S} & -\sum_{l=1}^{M} \ln P(z_l|s_l;\sigma) - \sum_{l=1}^{M} \ln P(s_l;\mu_S,\sigma_S) \\ \text{subject to} & \omega_l \geq 0, l = 1, ..., M \\ & \phi_l \in [-\pi/2, \pi/2], l = 1, ..., M \end{array} \tag{2}$$

Note that I have used semi-colons to include the parameters σ, σ_S, and μ_S in their respective pdfs, since they are being optimized over. The explicit forms of the log-pdfs in the objective is

$$\begin{array}{l} -\ln P(z_l|s_l;\sigma) - \ln P(s_l;\mu_S,\sigma_S) = \\ \frac{\ln(\sigma^2)}{2}|t_l| + \frac{1}{2\sigma^2}\sum_{t \in t_l}(z_{l,t} - \alpha(t, s_l))^2 + \sum_{d=1}^{7} \ln((\sigma_S)_d) + \frac{1}{2}||s_l - \mu_S||^2_{diag(\sigma_S)^{-1}} . \end{array} \tag{3}$$

Here, $||\cdot||_A^2 := (\cdot)^T A(\cdot)$.

4 Determining α

The issue of determining α still remains. Based on domain knowledge of the manufacturing process, we know that α must be one of seven different functions, which can be arranged hierarchically as shown in figure 2. We call these seven different functions "basis shapes" and we see that shapes lower in the hierarchy are obtained by setting the parameters of higher-up shapes to zero. Previously, my approach was to choose the shape that maximized the bayesian information criteria, but this approach involved solving the optimization program from the previous section seven times (once for each shape), which is computationally expensive. The new approach I am proposing is to impose L1 (sparse) regularization on the optimization problem with α hard coded to the most complex basis shape, the damped linearly driven oscillator. Since L1-regularization encourages parameter sparsity, this implicitly performs model selection, since setting certain parameters equal to zero corresponds to choosing one of the less complex models.

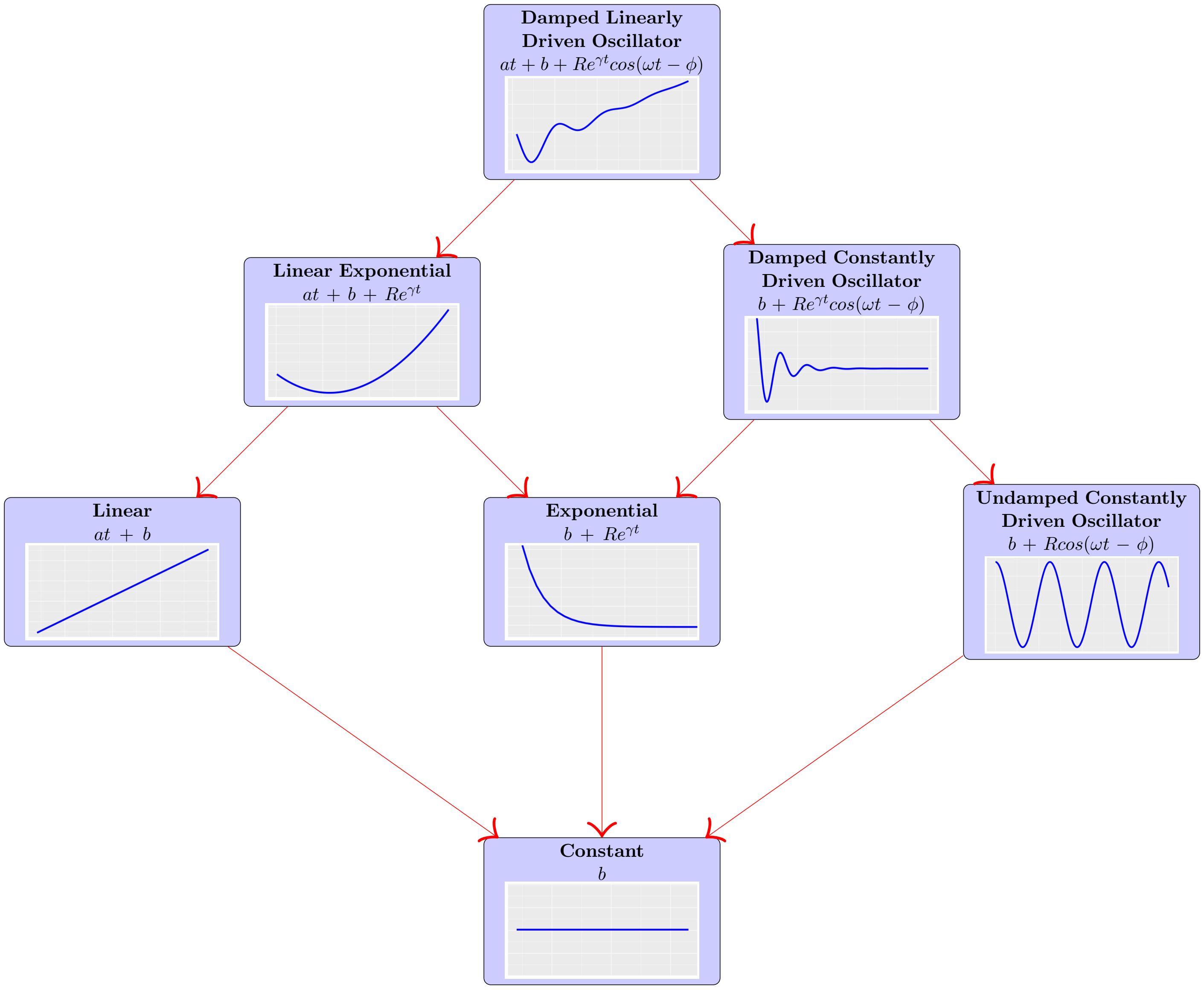

Damped Linearly Driven Oscillator
$at + b + Re^{\gamma t}cos(\omega t - \phi)$
Linear Exponential
$at + b + Re^{\gamma t}$
Damped Constantly Driven Oscillator
$b + Re^{\gamma t}cos(\omega t - \phi)$
Linear
$at + b$
Exponential
$b + Re^{\gamma t}$
Undamped Constantly Driven Oscillator
$b + Rcos(\omega t - \phi)$
Constant
b

Figure 2: Hierarchy of Basis Shapes

In the optimization formulation, we simply add an L1-term to the objective function from the previous section and set $\alpha = \alpha_{dld}$, where "dld" stands for "damped linearly driven". Note that this now means that $s_l = (\gamma_l, R_l, \omega_l, y_l, \phi_l, c_l, x_l)$ and so $\alpha(t, s_l) = R_l e^{-\gamma_l t} cos(\omega_l t - \phi_l) + c_l t + y_l$. Also note that each parameter encodes very different type of information: γ_l encodes how quickly the data for wafer l settles to the asymptote, R_l is the amplitude, ω_l controls the rapidness of oscillations, ϕ_l controls the phase shift, c_l controls the steepness of the asymptote, and y_l is the vertical shift. Since these parameters have completely different meanings, each one should have its own regularization parameter. Thus, we now define the regularization vector $\lambda := (\lambda_1, \lambda_2, \lambda_3, 0, \lambda_5, \lambda_6, \lambda_7)$, and the new objective function is

$$\begin{array}{ll} \min_{s_1,\dots,s_M,\sigma,\mu_S,\sigma_S} & -\sum_{l=1}^{M}\{\ln P(z_l|s_l;\sigma) + \ln P(s_l;\mu_S,\sigma_S) - \sum_{d=1}^{6}\lambda_d|s_{ld}|\} \\ \text{subject to} & \omega_l \geq 0, l = 1, \dots, M \\ & \phi_l \in [-\pi/2, \pi/2], l = 1, \dots, M \end{array} \tag{4}$$

The explicit objective function of (4) is

$$\ln(\sigma)\sum_{l=1}^{M}|t_l| + \frac{1}{2\sigma^2}\sum_{l=1}^{M}\sum_{t\in t_l}(z_{l,t}-\alpha_{dld}(t,s_l))^2 + M\sum_{d=1}^{6}\ln(\sigma_{S_d}) + \frac{1}{2}\sum_{l=1}^{M}||s_l-\mu_S||^2_{diag(\sigma_S)^{-1}} + \sum_{d=1}^{6}\lambda_d\sum_{l=1}^{M}|s_{ld}|\ .$$

Note that λ_4 is hard coded to zero. This is because it corresponds to the parameter y, which is present in all the models, and thus should not be regularized. Upon termination of the algorithm, we will have learned the parameter vectors s_l, for $l = 1, ..., M$. To choose α, take the median of each parameter over all the s_l, as this will again encourage sparsity.

5 Optimization Method

I used block coordinate proximal gradient descent to solve (4). The block coordinates are $\{s_1, ...s_M\}$, $\{\sigma, \mu_S\}$, and $\{\sigma_S\}$. To simplify notation, define $f(s, z, (\sigma, \mu_S, \sigma_S)) := -\ln P(z|s;\sigma) - \ln P(s;\mu_S,\sigma_S) + \sum_{d=1}^{6}\lambda_d|s_d|$. Note that by the independence of the S_i, the optimizations to learn the different s_i's can be performed individually. Thus the pseudocode is as follows:

```
1: function BLOCK1($z_1, ..., z_M$, $(\sigma, \mu_S, \sigma_S)$)
2:     for $l$ in $1 : M$ do
3:         $s_l \leftarrow \text{argmin}_s f(s, z_l(\sigma, \mu_S, \sigma_S))$          ▷ Projected Proximal Gradient Descent
4:     end for
5: return $(s_1, ..., s_M)$
6: end function
7:
8: function BLOCK2AND3($z_1, ..., z_M$, $s_1, ..., s_M$, $t_1, ..., t_M$)
9:     $\sigma \leftarrow \sqrt{\frac{1}{\sum_{l=1}^{M}|t_l|-1}\sum_{l=1}^{M}\sum_{t\in t_l}(\vec{z}_{l,t} - \alpha(t, s_l))^2}$          ▷ Analytical Solution for $\sigma$, no iterative procedure needed.
10:    $\mu_S \leftarrow \frac{1}{M}\sum_{l=1}^{M} s_l$          ▷ Analytical solution for $\mu_S$, no iterative procedure needed
11:    for $d$ in $1 : 7$ do
12:        $(\sigma_S)_d \leftarrow \sqrt{\frac{1}{M-1}\sum_{l=1}^{M}[(s_l)_d - (\mu_S)_d]^2}$          ▷ Analytical solution for $\sigma_S$, given independence of elements of $S$.
13:    end for
14: return $(\sigma, \mu_S, \sigma_S)$
15: end function
16:
17: function OUTEROPTIMIZE($z_1, ..., z_M$, $t_1, ..., t_M$)
18:    $(\sigma, \mu_S, \sigma_S) \leftarrow (\infty, \mathbb{1}_7, \infty_7)$          ▷ $\mathbb{1}_d$ is a vector of $d$ ones, $\infty_d$ is a vector of $d$ infinities.
19:    while Not Converged do
20:        $(s_1, ..., s_M) \leftarrow$ Block1($z_1, ..., z_M$,$(\sigma, \mu_S, \sigma_S)$)
21:        $(\sigma, \mu_S, \sigma_S) \leftarrow$ Block2and3($z_1, ..., z_M$, $s_1, ..., s_M$, $t_1, ..., t_M$)
22:    end while
23: return $(s_1, ..., s_M, (\sigma, \mu_S, \sigma_S))$
24: end function
```

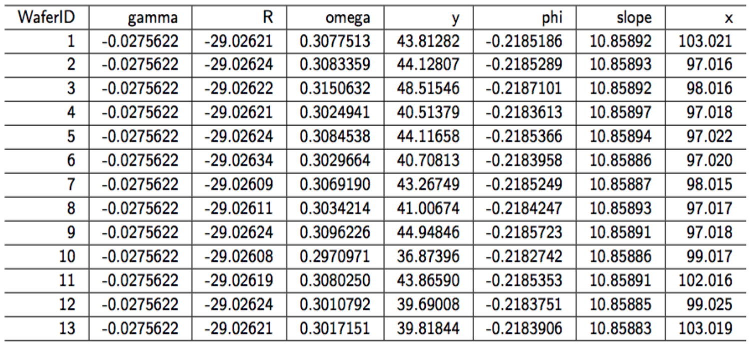
Note that proximal gradient descent is used in block 1 to minimize the function f for each wafer. f is non convex and non-differential with constraints, so projected proximal gradient descent is used. In proximal gradient, a gradient step is taken and then the soft threshold proximity operator is computed. Then, I project onto the feasible region.

6 Results

In my data set, there are many different subsets of data for which this optimization procedure can be performed. I have performed it on two such subsets and I display the results in this section. First, we see in the tables below that the regularized algorithm was able to achieve sparse sparse solutions with a sum of squared residuals comparable to that of the non-regularized algorithm. If we were to use the model selection procedure stated in the sections above (take the median of each parameter), this would mean that, for the first data set, the algorithm decided that the ϕ and γ were zero, hence resulting in the basis shape undamped constantly driven oscillator. For the second data set, the selected basis shape would be exponential.

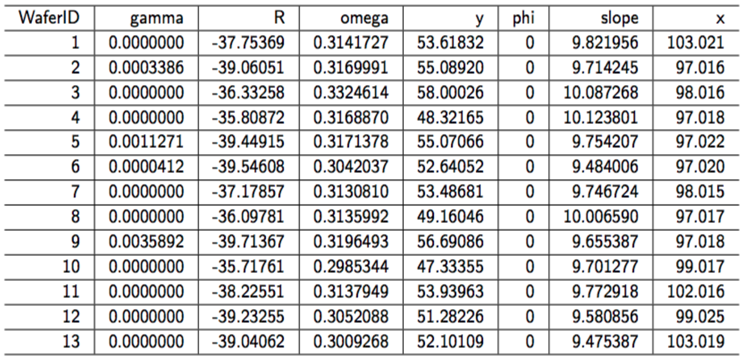
In figure 5, the difference between the sparse solution and the non-sparse solution is imperceptible to the human eye. In figure 8, we display just two of the solutions, since showing any more would be too cluttered. In this case, we see that, in finding sparse solutions, the algorithm was able to prevent overfitting in the frequency domain. Without regularization, there was no way of discouraging aliasing, and so the algorithm picked up on incorrect frequencies.

Figure 3: Dataset 1, No L1 regularization. Sum of Squared Residuals: 13.1767.



WaferID	gamma	R	omega	y	phi	slope	x
1	-0.0275622	-29.02621	0.3077513	43.81282	-0.2185186	10.85892	103.021
2	-0.0275622	-29.02624	0.3083359	44.12807	-0.2185289	10.85893	97.016
3	-0.0275622	-29.02622	0.3150632	48.51546	-0.2187101	10.85892	98.016
4	-0.0275622	-29.02621	0.3024941	40.51379	-0.2183613	10.85897	97.018
5	-0.0275622	-29.02624	0.3084538	44.11658	-0.2185366	10.85894	97.022
6	-0.0275622	-29.02634	0.3029664	40.70813	-0.2183958	10.85886	97.020
7	-0.0275622	-29.02609	0.3069190	43.26749	-0.2185249	10.85887	98.015
8	-0.0275622	-29.02611	0.3034214	41.00674	-0.2184247	10.85893	97.017
9	-0.0275622	-29.02624	0.3096226	44.94846	-0.2185723	10.85891	97.018
10	-0.0275622	-29.02608	0.2970971	36.87396	-0.2182742	10.85886	99.017
11	-0.0275622	-29.02619	0.3080250	43.86590	-0.2185353	10.85891	102.016
12	-0.0275622	-29.02624	0.3010792	39.69008	-0.2183751	10.85885	99.025
13	-0.0275622	-29.02621	0.3017151	39.81844	-0.2183906	10.85883	103.019

Figure 4: Dataset 1, L1 regularization. All elements of λ set to 1e-3 except for vertical shift (y). Sum of Squared Residuals: 14.76592



WaferID	gamma	R	omega	y	phi	slope	x
1	0.0000000	-37.75369	0.3141727	53.61832	0	9.821956	103.021
2	0.0003386	-39.06051	0.3169991	55.08920	0	9.714245	97.016
3	0.0000000	-36.33258	0.3324614	58.00026	0	10.087268	98.016
4	0.0000000	-35.80872	0.3168870	48.32165	0	10.123801	97.018
5	0.0011271	-39.44915	0.3171378	55.07066	0	9.754207	97.022
6	0.0000412	-39.54608	0.3042037	52.64052	0	9.484006	97.020
7	0.0000000	-37.17857	0.3130810	53.48681	0	9.746724	98.015
8	0.0000000	-36.09781	0.3135992	49.16046	0	10.006590	97.017
9	0.0035892	-39.71367	0.3196493	56.69086	0	9.655387	97.018
10	0.0000000	-35.71761	0.2985344	47.33355	0	9.701277	99.017
11	0.0000000	-38.22551	0.3137949	53.93963	0	9.772918	102.016
12	0.0000000	-39.23255	0.3052088	51.28226	0	9.580856	99.025
13	0.0000000	-39.04062	0.3009268	52.10109	0	9.475387	103.019

Figure 5: Dataset1: L1 Regularization Finds Sparse Representation. The black curve is the prior.

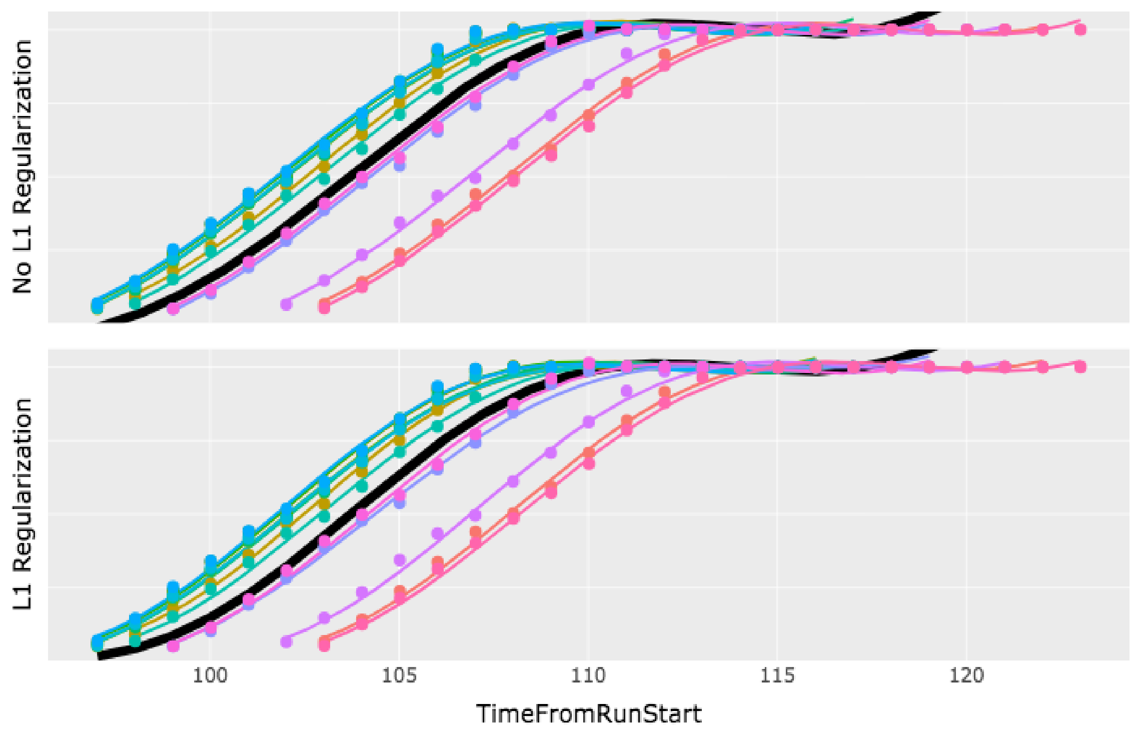
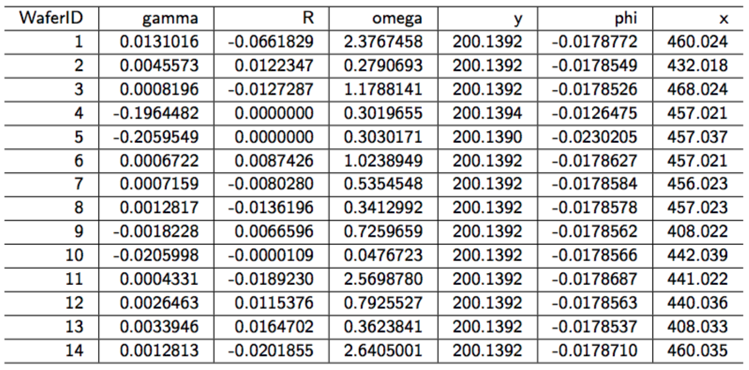
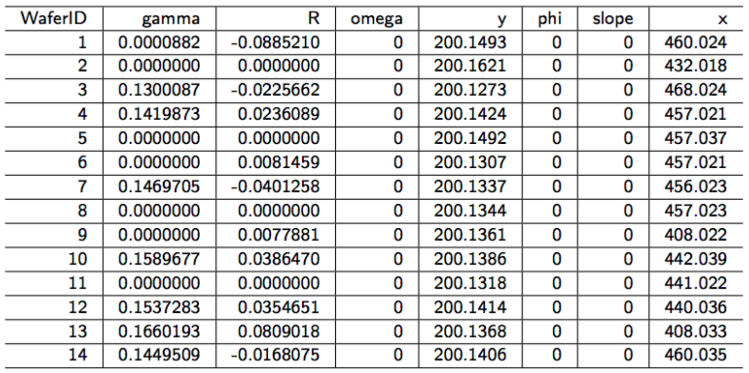


Figure 6: Dataset 2, No L1 regularization. Sum of Squared Residuals: 0.004541178



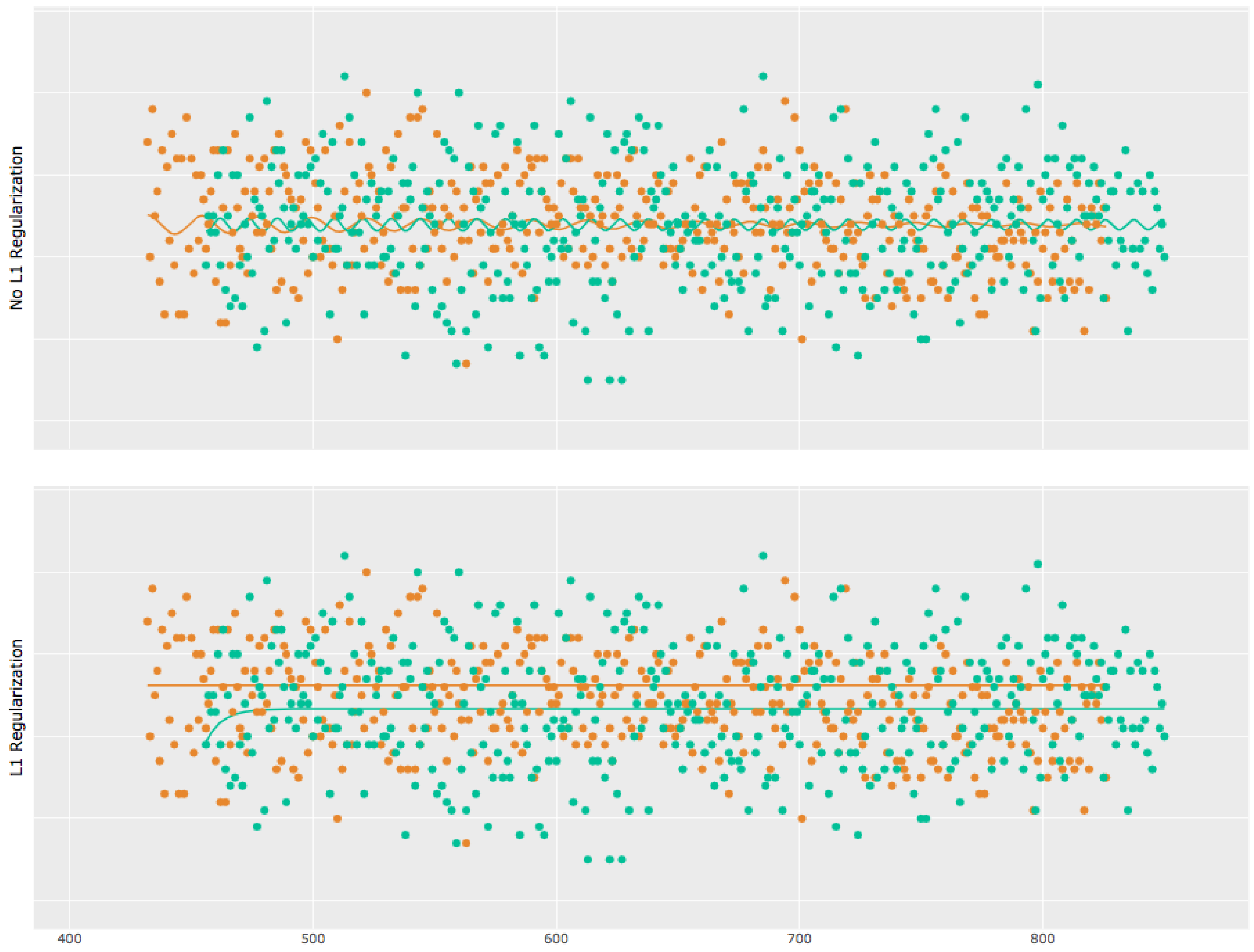
WaferID	gamma	R	omega	y	phi	x
1	0.0131016	-0.0661829	2.3767458	200.1392	-0.0178772	460.024
2	0.0045573	0.0122347	0.2790693	200.1392	-0.0178549	432.018
3	0.0008196	-0.0127287	1.1788141	200.1392	-0.0178526	468.024
4	-0.1964482	0.0000000	0.3019655	200.1394	-0.0126475	457.021
5	-0.2059549	0.0000000	0.3030171	200.1390	-0.0230205	457.037
6	0.0006722	0.0087426	1.0238949	200.1392	-0.0178627	457.021
7	0.0007159	-0.0080280	0.5354548	200.1392	-0.0178584	456.023
8	0.0012817	-0.0136196	0.3412992	200.1392	-0.0178578	457.023
9	-0.0018228	0.0066596	0.7259659	200.1392	-0.0178562	408.022
10	-0.0205998	-0.0000109	0.0476723	200.1392	-0.0178566	442.039
11	0.0004331	-0.0189230	2.5698780	200.1392	-0.0178687	441.022
12	0.0026463	0.0115376	0.7925527	200.1392	-0.0178563	440.036
13	0.0033946	0.0164702	0.3623841	200.1392	-0.0178537	408.033
14	0.0012813	-0.0201855	2.6405001	200.1392	-0.0178710	460.035

Figure 7: Dataset 2, L1 regularization. $\lambda_\gamma = 0.001, \lambda_R = 0.001, \lambda_\omega = 0.01, \lambda_y = 0, \lambda_\phi = 0.01, \lambda_c = 0.001$. Sum of Squared Residuals: 0.005132517



WaferID	gamma	R	omega	y	phi	slope	x
1	0.0000882	-0.0885210	0	200.1493	0	0	460.024
2	0.0000000	0.0000000	0	200.1621	0	0	432.018
3	0.1300087	-0.0225662	0	200.1273	0	0	468.024
4	0.1419873	0.0236089	0	200.1424	0	0	457.021
5	0.0000000	0.0000000	0	200.1492	0	0	457.037
6	0.0000000	0.0081459	0	200.1307	0	0	457.021
7	0.1469705	-0.0401258	0	200.1337	0	0	456.023
8	0.0000000	0.0000000	0	200.1344	0	0	457.023
9	0.0000000	0.0077881	0	200.1361	0	0	408.022
10	0.1589677	0.0386470	0	200.1386	0	0	442.039
11	0.0000000	0.0000000	0	200.1318	0	0	441.022
12	0.1537283	0.0354651	0	200.1414	0	0	440.036
13	0.1660193	0.0809018	0	200.1368	0	0	408.033
14	0.1449509	-0.0168075	0	200.1406	0	0	460.035

Figure 8: Dataset2: L1 Regularization Prevents Aliasing. The orange curve is wafer 2 and the turquoise curve is wafer 7 in the tables below.



7 Conclusions

I have shown my implementation of block proximal gradient descent for L1-regularized non-linear bayesian regression. We have seen examples where the regularization succeeds in finding sparser representations that do a similar job of fitting the data as the non-regularized algorithm. We can use the sparse representations for model selection. We showed how the sparse representations can also, at times, prevent aliasing in the frequency domain.