Matthew LeDuc
ECEN 5244
Fall 2023
HW 1

1 Question 1

I inserted a scan of work I did on paper, so it won't appear until the next page. Solutions to part A rely on the definition of the Gamma function

$$\Gamma(k) = \int_0^\infty x^{k+1} e^{-x} \mathrm{d}x$$

1 A) i) $\mathbb{E}[x^3] = \int_{\mathbb{R}} \frac{x^3}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{1}{2\sigma^2}x^2\right) dx$

$= \lim_{R\to\infty} \int_{-R}^{R} \frac{x^3 \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2\right)}{\sqrt{2\pi\sigma_x^2}} dx$. Since the integrand is an odd function this integral is 0 $\forall R$,

Therefore $\mathbb{E}[x^3] = \lim_{R\to\infty} \int_{-R}^{R} \frac{x^3}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{1}{2}\left(\frac{x}{\sigma_x}\right)^2\right) dx = 0$

ii) $\mathbb{E}[x^4] = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{\mathbb{R}} x^4 \exp\left[\frac{-1}{2\sigma_x^2}x^2\right] dx = \frac{2}{\sqrt{2\pi\sigma_x^2}} \int_0^\infty x^4 \exp\left(\frac{-1}{2\sigma_x^2}x^2\right) dx.$

Let $z = \frac{x^2}{2\sigma_x^2}$, then $dz = \frac{x}{\sigma_x^2} dx$ and the integral

becomes $\frac{2}{\sqrt{2\pi\sigma_x^2}} \int_0^\infty x^4 \left(\frac{\sigma_x^2}{x}\right) \exp(-z) dz$

$= 2\sqrt{\frac{\sigma_x^2}{2\pi}} \int_0^\infty x^3 \exp(-z) dy \quad \rightarrow x = \sqrt{2\sigma_x^2 z}$

$= 2\sqrt{\frac{\sigma_x^2}{2\pi}} \int_0^\infty (2\sigma_x^2)^{3/2} z^{3/2} \exp(-z) dz$

$= \frac{4}{\sqrt{\pi}} \sigma_x^4 \int_0^\infty z^{3/2} \exp(-z) dz = \frac{4}{\sqrt{\pi}} \sigma_x^4 \Gamma(5/2)$ so

$\boxed{\mathbb{E}[x^4] = 3\sigma_x^4}$ as desired, as $\Gamma(5/2) = \frac{3}{4}\sqrt{\pi}$

iii) $\mathbb{E}[x^6] = \frac{1}{\sqrt{2\pi\sigma_x^2}} \int_{\mathbb{R}} x^6 \exp\left(-\frac{x^2}{2\sigma_x^2}\right) dx = \sqrt{\frac{2}{\pi\sigma_x^2}} \int_0^\infty x^6 \exp\left(-\frac{x^2}{2\sigma_x^2}\right) dx$

Again let $z = \frac{x^2}{2\sigma_x^2}$, then $dz = \frac{x}{\sigma_x^2} dx$ and the integral

becomes $\sqrt{\frac{2\sigma_x^2}{\pi}} \int_0^\infty (2\sigma_x^2)^{5/2} z^{5/2} \exp(-z) dz$

$= \frac{8\sigma_x^6}{\sqrt{\pi}} \Gamma(7/2) = 15\sigma_x^6$ since $\Gamma(7/2) = \frac{15\sqrt{\pi}}{8}$

So $\boxed{\mathbb{E}[x^6] = 15\sigma_x^6}$

iv) $\mathbb{E}[X^2Y^4]$: We apply Isserlis's Theorem: $\mathbb{E}[X^2Y^4] = \sum \prod \mathbb{E}[X_iX_j]$. This sum has $5!! = 15$ terms. Only one of these terms will be of the form $\mathbb{E}[X^2]\,\mathbb{E}[Y^2]^2$, and then for each other term, we have the form $\mathbb{E}[XY]^2\,\mathbb{E}[Y^2]$. So, we can write that

$$\mathbb{E}[X^2Y^4] = \mathbb{E}[X^2]\,\mathbb{E}[Y^2]^2 + 14\,\mathbb{E}[XY]^2\,\mathbb{E}[Y^2]$$
$$= \sigma_X^2\sigma_Y^4 + 14\sigma_Y^2[\rho\sigma_X\sigma_Y]^2 = \sigma_X^2\sigma_Y^4(1+14\rho^2)$$

Thus $\boxed{\mathbb{E}[X^2Y^4] = \sigma_X^2\sigma_Y^4(1+14\rho^2)}$

1 B) Since X is zero-mean we need only the variance

C) We need three numbers: The variances of the real and imaginary parts, and the correlation coefficient

D) $M = \mathbb{E}[\exp(i\vec{v}\cdot\vec{x})] = 2\pi\,\mathcal{F}^{-1}[p_{xy}(x,y)]$

$$= \int_{\mathbb{R}^2} \exp(i\vec{v}\cdot\vec{x})\, p_{xy}(x,y)\, dx\,dy$$

$$= \int_{\mathbb{R}} \frac{\exp(i(v_x x + v_y y))}{2\pi\sqrt{|D|}} \exp\left(-\frac{1}{2D}\left(\sigma_y^2 x^2 - 2\sigma_x\sigma_y \rho_{xy} + \sigma_x^2 y\right)\right) dx\,dy$$

Now we let $z = \vec{v}\cdot\begin{pmatrix} x \\ y \end{pmatrix}$, and letting $\sigma_z^2 = \vec{v}^T \Sigma \vec{v}$, the integral becomes

$$\frac{1}{\sqrt{2\pi}\sigma_y} \int_{-\infty}^{\infty} \exp(iz) \exp\left(-\frac{1}{2}\frac{z^2}{\sigma_y^2}\right) dz$$

$$= \frac{1}{\sqrt{2\pi}\sigma_y} \int_{-\infty}^{\infty} \exp\left(-\tfrac{1}{2}\left[\frac{z^2}{\sigma_z^2} - 2iz\right]\right) dz$$ By completing the square we see $\frac{1}{\sigma_z^2}\frac{z^2}{} - 2iz = \frac{1}{\sigma_z^2}\left[z + \frac{i}{\sigma_z^2}\right]^2 + \frac{1}{\sigma_z^2}$

So the integral becomes

$$\frac{1}{\sqrt{2\pi}\sigma_z} \int_{-\infty}^{\infty} \exp\left(\frac{-1}{2\sigma_z^2}\left(z + \frac{i}{\sigma_z^2}\right)^2 + 1\right)$$

$$= \frac{1}{\sqrt{2\pi}\sigma_z} \exp\left(-\frac{1}{2}\sigma_z^{-2}\right) \int_{-\infty}^{\infty} \exp\left(\frac{-1}{2\sigma_z^2}\left(z + \frac{i}{\sigma_z^2}\right)\right) dz$$

$= \exp(-\frac{1}{2}\sigma_z^{-2})\, p_z(z)$ where $z \sim N(\frac{i}{\sigma_z^2}, \sigma_z^2)$. Thus, the integral is evaluated to $\sqrt{2\pi}\sigma_z$, and so

$\mathbb{E}[i\vec{v}\cdot\vec{x}] = \exp\left(-\frac{1}{2\sigma_z^2}\right) = \exp\left(-\frac{1}{2}\vec{v}^T\Sigma^{-1}\vec{v}\right)$ as

2 Question 2

(a) The scatterplot is shown in Figure 2.1. As we wanted, visual evidence of correlation between x_1 and x_3, and there appears to be positive correlation between x_1 and x_2, and negative between x_2 and x_3.

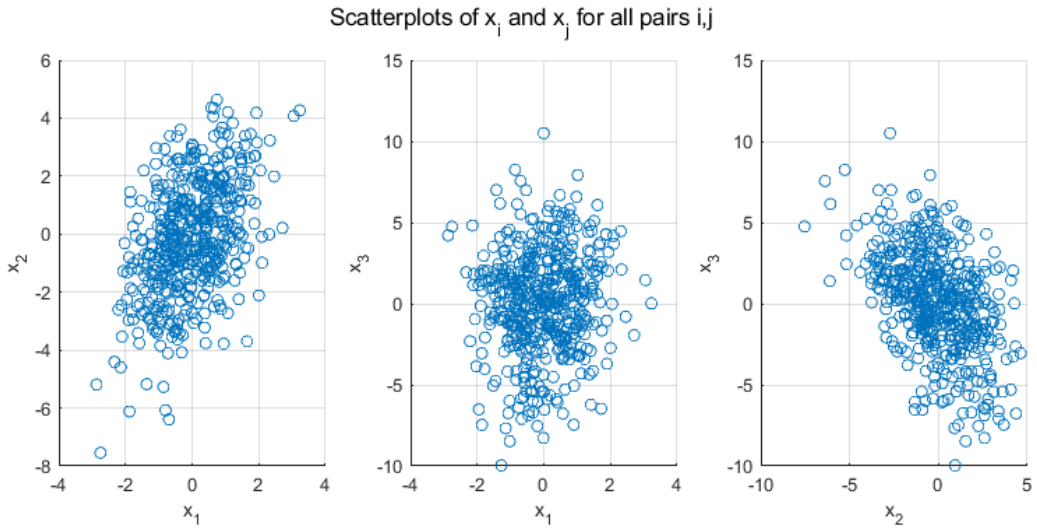


Figure 2.1: Scatterplot of simulated x_i

(b) The sample covariance matrix $\hat{R} = \frac{1}{N-1}\left[\vec{x}\vec{x}^T\right]$ was found to be

$$\hat{R} \approx \begin{pmatrix} 1.00 & 0.92 & 0.24 \\ 0.92 & 3.93 & -2.93 \\ 0.24 & -2.93 & 9.76 \end{pmatrix} \tag{2.1}$$

while the true covariance matrix R is given by

$$R = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 4 & -3 \\ 0 & -3 & 9 \end{pmatrix} \tag{2.2}$$

The two matrices are significantly different. This is because the sample covariance is not a robust estimator of the covariance, and so it is affected by outliers in the realization. In addition,

2.1 MATLAB Code

```
%% ECEN 5244 HW 1 Problem 2
clear variables;close all;
rng(452467365)
N = 500;%Number of points to simulate
```

```
R = diag([1,2,3]).^2;% Constructing covariance matrix
R(1,2) = 0.5*sqrt(R(1,1)*R(2,2));R(2,1)=R(1,2);
R(2,3) = -0.5*sqrt(R(2,2)*R(3,3)); R(3,2)=R(2,3);
%% Begin simulation
x = randn(3,N);
% L = chol(R, 'lower');
[E,Lambda] = eig(R);
z = E*sqrt(Lambda)*x;
figure;
subplotCounter = 1;
for ii=1:2
    for jj = ii+1:3
        subplot(1, 3, subplotCounter)
        scatter( z(ii,:), z(jj,:) )
        grid on
        xlabel(sprintf('x_{%d}', ii))
        ylabel(sprintf('x_{%d}', jj))
        subplotCounter = subplotCounter+1;
    end
end
sgtitle('Scatterplots of x_i and x_j for all pairs i,j')
% savefig( 'Q2_scatterplot.fig')
% saveas(gcf, 'Q2_scatterplot.png')

R_sample = z*z'/(N-1); %Sample covariance matrix

figure;imagesc( R );colorbar;caxis([-3,9])
figure;imagesc( R_sample );colorbar;caxis([-3, 9])
```

3 Question 3

Let $f(x)$ be a given function and let $a_1 < a_2 < a_3$ be three distinct points on the real line such that $f(a_2) < f(a_1), f(a_3)$. These three points define a parabola $\hat{f}(x) = Ax^2 + Bx + C$ where the coefficients A, B, C are given by solving the matrix equation

$$X \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} a_1^2 & a_1 & 1 \\ a_2^2 & a_2 & 1 \\ a_3^2 & a_3 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \end{pmatrix} = \begin{pmatrix} f(a_1) \\ f(a_2) \\ f(a_3) \end{pmatrix} \tag{3.1}$$

Since the a_i are distinct, we can calculate X^{-1} by first finding $\Delta = \det(X)$, then for each (i, j), $(X^{-1})_{i,j}$ is given by the determinant of the (i, j) minor of X divided by Δ, and then swapping the signs on the $(1, 2), (2, 1), (2, 3)$, and $(3, 2)$ entries. Therefore, we can show that

$$X^{-1} = \frac{1}{(a_1 - a_2)(a_2 - a_3)(a_1 - a_3)} \begin{pmatrix} a_2 - a_3 & a_3 - a_1 & a_1 - a_2 \\ a_3^2 - a_2^2 & a_1^2 - a_3^2 & a_2^2 - a_1^2 \\ a_2 a_3(a_2 - a_3) & a_1 a_3(a_3 - a_1) & a_1 a_2(a_1 - a_2) \end{pmatrix} \tag{3.2}$$

Since $\Delta = \frac{1}{(a_1 - a_2)(a_2 - a_3)(a_1 - a_3)}$, evaluating the matrix-vector multiplication shows that

$$\begin{pmatrix} A \\ B \\ C \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} (a_2 - a_3)f(a_1) + (a_3 - a_1)f(a_2) + (a_1 - a_2)f(a_3) \\ (a_3^2 - a_2^2)f(a_1) + (a_1^2 - a_3^2)f(a_2) + (a_2^2 - a_1^2)f(a_3) \\ a_2 a_3(a_2 - a_3)f(a_1) + a_1 a_3(a_3 - a_1)f(a_2) + a_1 a_2(a_1 - a_2)f(a_3) \end{pmatrix} \tag{3.3}$$

The vertex of this parabola is found at $x^* = -\frac{B}{2A}$, so from 3.3 we see that

$$x^* = -\frac{1}{2}\left(\frac{(a_3^2 - a_2^2)f(a_1) + (a_1^2 - a_3^2)f(a_2) + (a_2^2 - a_1^2)f(a_3)}{(a_2 - a_3)f(a_1) + (a_3 - a_1)f(a_2) + (a_1 - a_2)f(a_3)}\right) \tag{3.4}$$

We can rearrange and simplify this expression as follows:

$$\begin{aligned} x^* &= -\frac{1}{2}\left(\frac{(a_3^2 - a_2^2)f(a_1) + (a_1^2 - a_3^2)f(a_2) + (a_2^2 - a_1^2)f(a_3)}{(a_2 - a_3)f(a_1) + (a_3 - a_1)f(a_2) + (a_1 - a_2)f(a_3)}\right) \\ &= a_2 - \frac{1}{2}\left(\frac{(a_3^2 - a_2^2)f(a_1) + (a_1^2 - a_3^2)f(a_2) + (a_2^2 - a_1^2)f(a_3)}{(a_2 - a_3)f(a_1) + (a_3 - a_1)f(a_2) + (a_1 - a_2)f(a_3)}\right) - a_2 \end{aligned} \tag{3.5}$$

From here, we can multiply $-a_2$ by the denominator of the second term to find that

$$x^* = a_2 - \frac{1}{2}\frac{(a_3 - a_2)^2 f(a_1) - (a_2 - a_1)^2 f(a_3) + (a_1 - a_3)(a_1 + a_3 - 2a_2)f(a_2)}{(a_2 - a_3)f(a_1) + (a_3 - a_1)f(a_2) + (a_1 - a_2)f(a_3)} \tag{3.6}$$

Expanding the coefficient on $f(a_2)$ in the numerator, we can show that

$$\begin{aligned} (a_1 - a_3)(a_1 + a_3 - 2a_2) &= a_1^2 - 2a_1 a_2 + 2a_2 a_3 - a_3^2 \\ &= (a_1^2 - 2a_1 a_2 + a_2^2) - (a_3^2 - 2a_2 a_3 + a_2^2) \\ &= (a_2 - a_1)^2 - (a_2 - a_3)^2 \end{aligned} \tag{3.7}$$

and so we have that

$$\begin{aligned} x^* &= a_2 - \frac{1}{2}\frac{(a_3-a_2)^2 f(a_1) - (a_2-a_1)^2 f(a_3) + \left((a_2-a_1)^2 - (a_2-a_3)^2\right) f(a_2)}{(a_2-a_3)f(a_1) + (a_3-a_1)f(a_2) + (a_1-a_2)f(a_3)} \\ &= a_2 - \frac{1}{2}\frac{(a_3-a_2)^2(f(a_1)-f(a_2)) + (a_2-a_1)^2(f(a_2)-f(a_3))}{(a_2-a_3)f(a_1) + (a_3-a_1)f(a_2) + (a_1-a_2)f(a_3)} \\ &= a_2 - \frac{1}{2}\frac{(a_3-a_2)^2(f(a_1)-f(a_2)) + (a_2-a_1)^2(f(a_2)-f(a_3))}{(a_2-a_3)f(a_1) + (a_3-a_1)f(a_2) + (a_1-a_2)f(a_3) + a_2 f(a_2) - a_2 f(a_2)} \\ &= a_2 - \frac{1}{2}\frac{(a_3-a_2)^2(f(a_1)-f(a_2)) + (a_2-a_1)^2(f(a_2)-f(a_3))}{(a_2-a_1)(f(a_2)-f(a_3)) - (a_2-a_3)(f(a_2)-f(a_1))} \end{aligned} \tag{3.8}$$

Setting $a_2 = x^*$, we have obtained the desired iterative form of Brent's Method.

4 Question 4

4.1 Part A

The data appears to come from an exponentially damped cosine. Therefore, I propose that a suitable model would be that $\hat{y}(x|\vec{a}) = a_1 \cos(a_2 x) e^{a_3 x}$. In the model, we will have to enforce that $a_3 < 0$. For the Prony's Algorithm, I will use the ansatz that $\hat{y}(x) = a_1 e^{\alpha_1 x} + a_2 e^{\alpha_2 x}$.

4.2 Part B

For this problem, I implemented a steepest descent algorithm. However, during my work I found that $f(\vec{a})$ is locally very flat away from certain values of a_3, which made finding the global minimum difficult (see Figure 4.1). To accelerate the convergence, the first several steps were taken using a vanilla gradient descent algorithm of the form

$$\vec{a}_{n+1} = \vec{a}_n - \lambda \vec{\nabla} f(\vec{a}_n) \tag{4.1}$$

with initial guess $\vec{a} = (1, 1, -1)^T$,where λ controls the step size. I found the best results with $\lambda = 10^{-4}$. After the first few steps were taken using this algorithm I implemented the steepest descent algorithm as

$$\vec{a}_{n+1} = \vec{a}_n - \left[\text{Hess}(f(\vec{a}_n))\right]^{-1} \vec{\nabla} f(\vec{a}_n) \tag{4.2}$$

where $\text{Hess}(f(\vec{a}_n))$ is the Hessian of f evaluated at $\vec{a}_n$. Since the model was simple, I was able to implement an analytic Hessian and gradient. Following these steps on unfiltered data, I found that

$$\vec{a}^* \approx \begin{pmatrix} 2.933 \\ 2.010 \\ -0.276 \end{pmatrix} \tag{4.3}$$

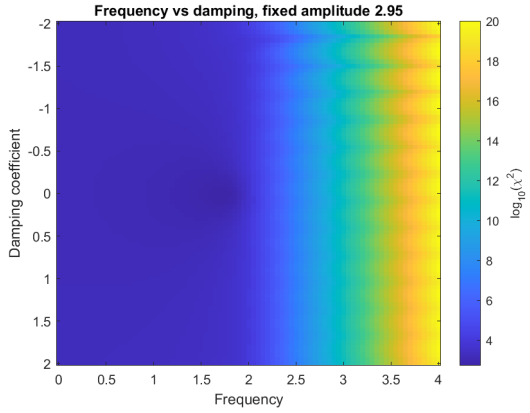
and that $\chi^2(\vec{a}^*) \approx 537.8$, indicating that this is a good model.

4.3 Part C

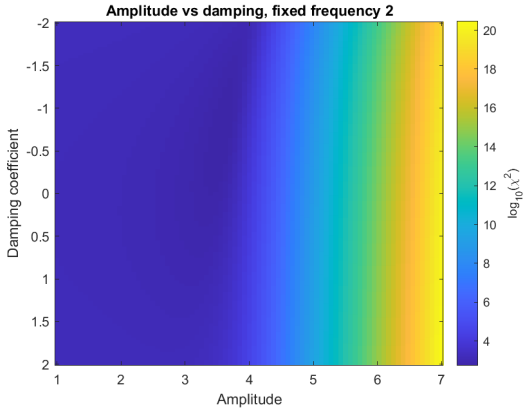
In order to apply simulated annealing, I discretized the parameter space $[0, 7] \times [0, 4] \times [-2, 0]$ into a $2000 \times 2000 \times 2000$ grid. My algorithm performed 10^5 iterations with a cooling schedule $T = t^{-1/2}$ and a maximum transition probability of $P_0 = 0.9$. After randomly initializing, I found that

$$\vec{a}^* \approx \begin{pmatrix} 2.936 \\ 2.011 \\ -0.276 \end{pmatrix} \tag{4.4}$$

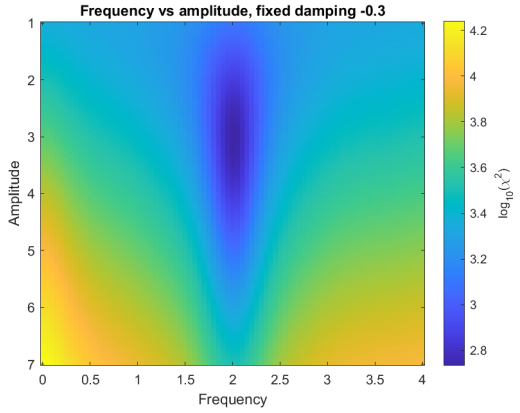
which is very similar to the value recovered via steepest descent shown in Equation 4.3. The final value for χ^2 was approximately 537.8, again very similar to the value found using steepest descent. The plot of χ^2 against time is shown in Figure 4.2.



(a) Fixed amplitude



(b) Fixed frequency



(c) Fixed damping

Figure 4.1: Plots of $\log_{10}(\chi^2)$ near $\vec{a}^*$

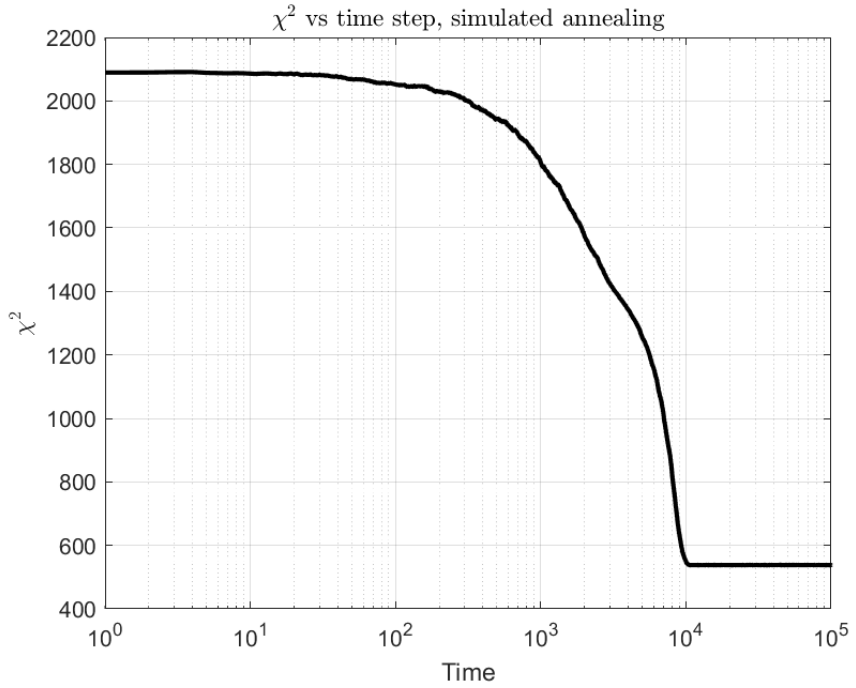


Figure 4.2: χ^2 vs time for simulated annealing.

4.4 Part D

In order to apply Prony's Algorithm, we require that $\hat{y}(x|\vec{a}) = a_1 e^{\alpha_1 x} + a_2 e^{\alpha_2 x}$. Before processing, the data is convolved with a triangle filter of half-length 51 points, which suppresses noise that could hide the oscillations. This reduces the standard deviation of the noise by a factor of $w_{ss}^{-1/2}$, where w_{ss} is the squared sum of the coefficients of the triangle filter, and this must be accounted for in the calculation of χ^2. After trimming the data to remove any filter effects near the start and end, I applied Prony's Algorithm and found that

$$\begin{aligned} a_1 &\approx 1.572 - 0.1429i \\ a_2 &\approx 1.572 + 0.1429i \\ \alpha_1 &\approx -0.289 + 2.003i \\ \alpha_2 &\approx -0.289 - 2.003i \end{aligned} \tag{4.5}$$

Since $a_1 e^{\alpha_1 x} = \overline{a_2 e^{\alpha_2 x}}$, this can be recast in the form of the models used before in the equivalent form

$$\hat{y}(x|\vec{a}) \approx 3.14 \cos(2.003x) e^{-0.289x} \tag{4.6}$$

which is similar to the results found in Parts B and C. In this case, comparing to the filtered data $\chi^2 \approx 519.1$, again suggesting that this is a good fit.

The results for all algorithms are shown in Figure 4.3. Since the results for simulated annealing and steepest descent are so similar, the models ended up plotted on top of each other.

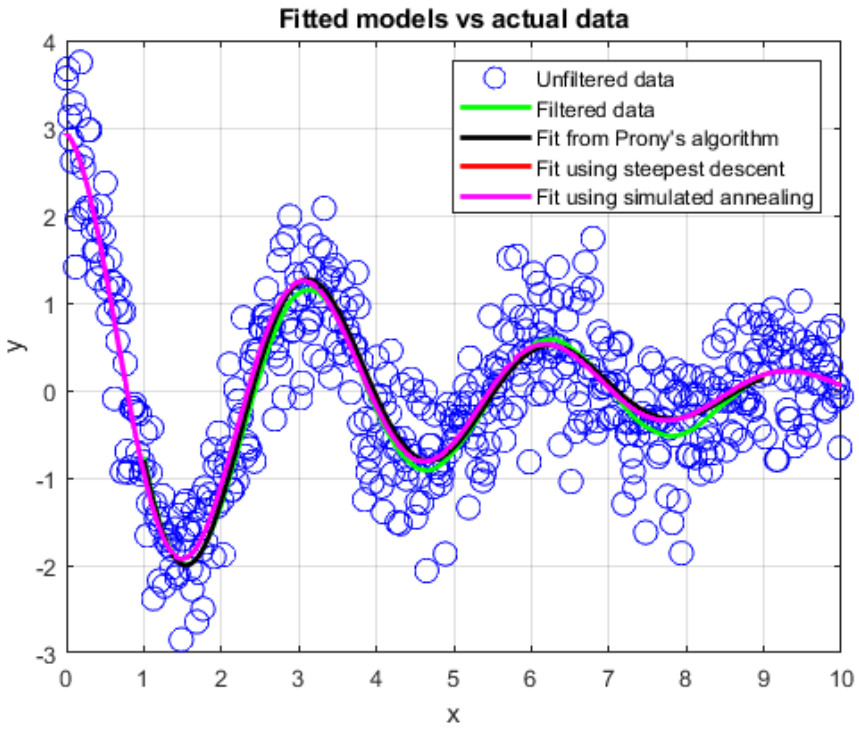


Figure 4.3: Models derived using all three algorithms.

4.5 MATLAB Code

```
%% ECEN 5244 HW 1 Problem 4
% dbstop if nan
clear variables;close all
rng(24572467)
data = load( 'hw1_4.mat' );
sigma = 0.5;
%%%%%Model is a1*cos(a2*x)*exp(a3*x)
%% First: Apply the steepest descent method
b = [1, 1, -1].';%[2.9, 2, -0.2].';
del = 1;
tol = 1e-3;
cnt=0;
while cnt<100 && del>tol
    %First: Find the feasible region with vanilla gradient descent
    cnt=cnt+1;
    gradf = getGradient( b, data )/sigma^2;
    db = -0.0001*gradf;
    b = b+db;
    del = max(abs(db));
end
tol = tol/1000;
cnt = 0;
hessMat = eye(3);
```

```
while cnt<500 && del>tol && cond(hessMat)<10000 %Now use big boy gradient descent
    cnt=cnt+1;
    gradf = getGradient( b, data )/sigma^2;
    hessMat = getHessian(b, data)/sigma^2;
    db = -hessMat\gradf;
    b = b+db;
    del = max(abs(db));
end
chi2_steep = getChi2(b, data, sigma);
%% Second: Apply simulated annealing
a = linspace(1, 7, 2000);%Amplitude
w = linspace(0, 4, 2000);%Frequency
d = linspace(-2, 0, 2000);%Damping
ind = randi( 2000,[1,3]);
z = [a(ind(1)), w(ind(2)), d(ind(3))];%Parameter vector
maxIters = 1e5;
t = 1:maxIters;
cooling = 1./sqrt(t); %Cooling schedule
pHop = 0.9; %Transition probability
chiVals = zeros([1, maxIters+1]);
chiVals(1) = getChi2(z, data, 0.5);
for jj = t
    delind = (randi(3,[1,3])-2);%Choosing a random neighbor
    if all(delind==0)
        delind = (randi(3,[1,3])-2);
    end
    ind2 = ind+delind;
    mask = (ind2>2000);
    ind2(mask) = 2000;
    chiVals(jj+1) = getChi2( [a(ind2(1)), w(ind2(2)), d(ind2(3))], data, 0.5 );
    changeProb = pHop*exp(-1/cooling(jj)*max(0, chiVals(jj+1)-chiVals(jj)));
    changeBool = rand(1)<changeProb;%Update chi2 and switch based on changeProb
    if changeBool
        ind = ind2;
    end
end

saf = figure;semilogx( t, chiVals(2:end), 'k', 'Linewidth', 2 );
grid on;
xlabel('Time')
ylabel('$\chi^2$', 'interpreter', 'latex')
title('$\chi^2$ vs time step, simulated annealing','interpreter', 'latex')
saveas(saf, 'sim_anneal.png')
savefig( saf, 'sim_anneal.fig' )
b_sa = [a(ind(1)), w(ind(2)), d(ind(3))];%Final fitted parameters
%% Last: Apply the Prony Method
%Prefilter the data. Sensitive to the filter!
filtLength = 50;
tFilt = bartlett(filtLength*2+1);%abs(sawtooth( linspace(-pi, pi, 2*filtLength+1 )));
wss = (sum(tFilt.^2));
tFilt = tFilt/wss;
% if ~iscola(tFilt, 2*filtLength)
%     keyboard
% end
```

```
data.y = conv(tFilt, data.y);
data.y = data.y(2*filtLength+1:end-2*(filtLength));
%M=4
dx = mean(diff(data.x));
M = 2;
A = [data.y(1:end-2),data.y(2:end-1)];
beta = -(A'*A)\A'*data.y(3:end);
uj = roots([1,flip(beta.')]);
alpha = log( uj )/dx;
X = exp(data.x(filtLength+1:end-filtLength)*alpha.');

amp = (X'*X)\X'*data.y(1:end);
expPart = [];
for ii = 1:M
    expPart(ii,:) = exp(alpha(ii)*data.x(filtLength+1:end-filtLength));
end
sigma_prony = sqrt(1/wss)*sigma ;

chi2_prony = 1/sigma_prony^2*sum(abs(data.y-amp(1)*exp(alpha(1)*data.x(filtLength+1:end-fil
    -amp(2)*exp( alpha(2)*data.x(filtLength+1:end-filtLength) )).^2);

d2 = load('hw1_4.mat');
f = figure;plot(d2.x, d2.y,'bo','markersize',10);hold on;
plot(data.x(filtLength+1:end-filtLength), data.y, 'g', 'linewidth',2);
plot(data.x(filtLength+1:end-filtLength),(sum((amp.'.*expPart.').') ),'k', 'linewidth',2);
plot(data.x, b(1)*cos(b(2)*data.x).*exp(b(3)*data.x), 'r', 'linewidth',2);
plot(data.x, b_sa(1)*cos(b_sa(2)*data.x).*exp(b_sa(3)*data.x), 'm', 'linewidth',2);
grid on;
xlabel('x')
ylabel('y')
title('Fitted models vs actual data')
legend('Unfiltered data', 'Filtered data', 'Fit from Prony''s algorithm', 'Fit using steepe

%% Functions
function gradF = getGradient( b, data )%Taking advantage of the known model here
x = data.x;
y = data.y;
gF = zeros( length(b), length(x) );
F = y-b(1)*exp(b(3)*x).*cos(b(2)*x);
gF(1,:) = -cos(b(2)*x).*exp(b(3)*x).*F;
gF(2,:) = b(1)*x.*sin( b(2)*x).*exp(b(3)*x).*F;
gF(3,:) = -b(1)*x.*cos( b(2)*x).*exp(b(3)*x).*F;
gradF = 2*sum( gF.' ).';
end
function chi2 = getChi2(b, data, sigma)
chi2 = sum( abs(data.y-b(1)*exp(b(3)*data.x).*cos(b(2)*data.x)).^2)/sigma^2;
end
function H = getHessian( b, data )
x=data.x;y=data.y;
F = y-b(1)*cos(b(2)*x).*exp(b(3)*x);
H = zeros(length(b),length(b),length(data.x));
H(1,1,:) = 2*cos(b(2)*x).^2.*exp(2*b(3)*x);
H(1,2,:) = 2*x.*sin(b(2)*x).*exp(b(3)*x).*F-2*x*b(1).*exp(2*b(3)*x).*cos(b(2)*x).*sin(b(2)
H(2,1,:) = H(1,2,:);
```

```
H(1,3,:) = -2*x.*cos( b(2)*x ).*exp(b(3)*x).*F+2*x*b(1).*cos(b(2)*x).^2.*exp(2*b(3)*x);
H(3,1,:) = H(1,3,:);
H(2,2,:) = 2*b(1)*x.^2.*cos(b(2)*x).*exp(b(3)*x).*F+2*b(1)^2*x.^2.*sin(b(2)*x).^2.*exp(2*b
H(2,3,:) = 2*b(1)*x.^2.*sin(b(2)*x).*exp(b(3)*x).*F-2*b(1)^2*x.^2.*sin(b(2)*x).*cos(b(2)*x)
H(3,2,:) = H(2,3,:);
H(3,3,:) = -2*b(1)*x.^2.*cos(b(2)*x).*exp(b(3)*x).*F+2*b(1)^2*x.^2.*cos(b(2)*x).^2.*exp(2*k
H = sum(H,3);
end
```