It would be good idea to give The intuition behind That, i.e. The fact That This means Their approximating vectors are basically perpendicular since The inner product relates to length and angle (cos of The angle formed).

Providing The link between orthogonal functions and Fourier series might be good, I mean, The idea That any vector can be seen as The sum of two weighted orthonormal (might need to explain what That is) vectors. The jump into functions is clear.

Regression Analysis

5 Oct 2012

(I) PROBLEM

The problem is as follows: Given a set of N data points

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

find a function that best approximates the underlying function $f(x_i) = y_i$ which relates the data. The nature of the underlying function that defines the data or the details of how the data was generated is usually unknown. These data points might be totally random or may be related by a characteristic function.

(II) THEORY

In the absence of the underlying function, one could treat the data as a linear combination of some set of functions.

$$\hat{y} = \sum_{i=1}^{M} w_i \phi_i(x)$$
$$= \bar{w}^T \bar{\phi}(x), \qquad (1)$$

where M is the number of terms, $\bar{w} = [w_1 w_2 \ldots w_M]^T$ (superscript T refers to the matrix transpose), and $\bar{\phi}(x) = [\phi_1(x)\phi_2(x)\ldots\phi_M(x)]^T$. ~~Here,~~ w_i is the weight corresponding to the function ϕ_i.

I am not sure This makes sense at This point. It might be better to move it after II.II, where The meaning of ϕ_i is clearer.

(II.I) ORTHOGONAL FUNCTIONS

Consider two functions $\phi_1(x)$ and $\phi_2(x)$ defined over the range $[a, b]$. The *inner product* of these functions is defined as

$$< \phi_1, \phi_2 >= \int_a^b \phi_1(x)\phi_2(x)$$

ϕ_1 and ϕ_2 are said to be *orthogonal* if $< \phi_1, \phi_2 >= 0$. If there is a set of functions $\{\phi_1, \phi_2, \ldots, \phi_M\}$ defined over a range, then these functions form an orthogonal set if the inner product of any pair of these orthogonal functions $< \phi_i, \phi_j >= 0$ for $i \neq j$.

~~Providing The link~~

(II.II) FOURIER SERIES

A Fourier series is a decomposition of a periodic function into a sum of

point out They are orthogonal. In fact you might want to point out The reasons behind needing both, This will need a brief explanation about odd and even functions, and why using only sin or only cos is not enough.

infinite sine/cosine functions. Any periodic function $f(x)$ with fundamental period T ~~can be represented~~ using Fourier series as:

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n cos\frac{2n\pi x}{T} + b_n sin\frac{2n\pi x}{T}\right)$$

where ~~The~~ Fourier coefficients a_n and b_n can be determined from the following integrals:

$$a_0 = \frac{1}{T}\int_0^T f(x)dx$$
$$a_n = \frac{2}{T}\int_0^T f(x)cos\frac{2n\pi x}{T}dx$$
$$b_n = \frac{2}{T}\int_0^T f(x)sin\frac{2n\pi x}{T}dx$$

Intuition behind each coefficient?

Fourier series representation of a Sawtooth wave

Consider the sawtooth function below with a peak value of 1. The func-

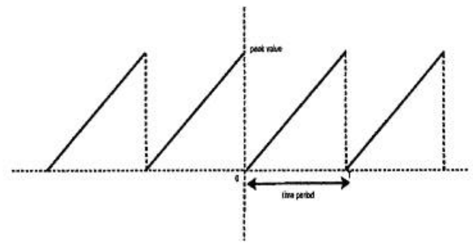

Figure 1: Sawtooth function

tional form of this wave is

$$f(x) = \frac{x}{T}$$

The above This needs a frac ~~[illegible]~~ or something like That to indicate it is The fractional part only, no?

The sawtooth wave is a periodic function (time period T) and hence can be decomposed using Fourier series representation. The components of the Fourier series are therefore given by:

$$a_0 = \frac{1}{T}\int_0^T \frac{x}{T}dx$$
$$= \frac{1}{2}$$
$$a_n = \frac{2}{T}\int_0^T \frac{x}{T}cos\left(\frac{2n\pi x}{T}\right)dx$$
$$= 0$$

Now that I have read it all, I see why you have used these two (you use them in your experiments). That's fine, but perhaps add one extra.

both functions have a_n as $\varnothing$. It would be better to have one that has only sin (done), only cosin, and both.

$$b_n = \frac{2}{T}\int_0^T \frac{x}{T} sin\left(\frac{2n\pi x}{T}\right)dx$$
$$= -\frac{1}{n\pi}$$

The Fourier series is therefore given by

$$f(x) = \frac{1}{2} - \frac{1}{\pi}\sum_{n=1}^{\infty}\frac{1}{n} sin\left(\frac{2n\pi x}{T}\right) \tag{2}$$

Fourier series representation of a Square wave
Consider the square wave shown below with a peak value of 1.

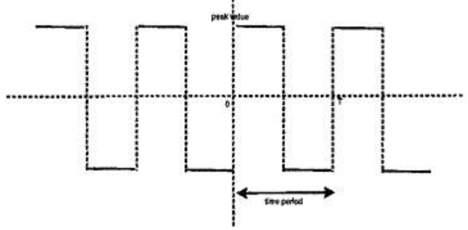

Figure 2: Square function

The functional form of this wave is

$$f(x) = \begin{cases} +1 & \text{if } 0 < x < \frac{T}{2}, \\ -1 & \text{if } \frac{T}{2} < x < T. \end{cases}$$

Same issue as before.

The components of the Fourier series are then given by

$$a_0 = \frac{1}{T}\int_0^T f(x)dx$$
$$= \frac{1}{T}\int_0^{\frac{T}{2}} 1dx + \frac{1}{T}\int_{\frac{T}{2}}^T -1dx$$
$$= 0$$
$$a_n = \frac{2}{T}\int_0^T f(x)cos\left(\frac{2n\pi x}{T}\right)dx$$
$$= \frac{2}{T}\int_0^{\frac{T}{2}}(1)cos\left(\frac{2n\pi x}{T}\right)dx + \frac{2}{T}\int_{\frac{T}{2}}^T(-1)cos\left(\frac{2n\pi x}{T}\right)dx$$
$$= 0$$
$$b_n = \frac{2}{T}\int_0^T f(x)sin\left(\frac{2n\pi x}{T}\right)dx$$

good to remind the reader what you are talking about.

$$= \frac{2}{T}\int_0^{\frac{T}{2}}(1)sin\left(\frac{2n\pi x}{T}\right)dx + \frac{2}{T}\int_{\frac{T}{2}}^T(-1)sin\left(\frac{2n\pi x}{T}\right)dx$$
$$= \begin{cases} \frac{2}{n\pi} & \text{if n is even} \\ 0 & \text{if n is odd} \end{cases}$$

The Fourier series is therefore given by

$$f(x) = \frac{4}{\pi}\sum_{n=1,3,5,\ldots}^{\infty}\frac{1}{n} sin\left(\frac{2n\pi x}{T}\right) \tag{3}$$

(II.III) Regression Model

(1) is a *linear model* for regression. Approximating the output values y_n using the linear model has an associated error. Minimizing the sum of their squared errors results in an optimal set of weights. The error in approximation is given by $(y_n - \hat{y_n})^2$ where $\hat{y_n}$ is the estimated y_n value. The combined error for all N data points can then be written as

Not sure we netter this phrase is needed at all

$$\mathcal{E} = \sum_{n=1}^{N}(y_n - \hat{y_n})^2 \tag{4}$$

The error given a M and $\bar{w}$ is

number of terms

vector of weights

$$\mathcal{E}(\bar{w}) = \sum_{n=1}^{N}(y_n - \sum_{i=1}^{M} w_i\phi_i(x_n))^2 \tag{5}$$

The optimal set of weights is one that minimizes $\mathcal{E}(\bar{w})$; That is one that...

$$\bar{w}^* = \underset{\bar{w}}{\text{argmin}}\,\mathcal{E}(\bar{w})$$

relatively clear from context, but it would be better to introduce all notation.

Differentiating $\mathcal{E}(\bar{w})$ with respect to $\bar{w}$, we have

$$\frac{d}{d\bar{w}}\mathcal{E}(\bar{w}) = \frac{d}{d\bar{w}}\sum_{n=1}^{N}(y_n - \bar{w}^T\bar{\phi}(x_n))^2$$
$$= 2\sum_{n=1}^{N}(y_n - \bar{\phi}(x_n)^T\bar{w})\bar{\phi}(x_n)$$

Not sure everyone will follow this step.

Now,

$$\frac{d}{d\bar{w}}\mathcal{E}(\bar{w}) = 0 \Rightarrow \sum_{n=1}^{N}(y_n - \bar{\phi}(x_n)^T\bar{w})\bar{\phi}(x_n) = 0$$
$$\therefore \sum_{n=1}^{N} y_n\bar{\phi}(x_n) = \sum_{n=1}^{N}\bar{\phi}(x_n)^T\bar{w}\bar{\phi}(x_n) \tag{6}$$

What experiment? You haven't talked about experiment yet, so you cannot use "This". You will have to say something like "In our experiments", or, "For the experiments we ~~will~~ perform in this paper", or something like it.

If $\bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}$ and $\Phi = \begin{bmatrix} \bar{\phi}(x_1)^T \\ \bar{\phi}(x_2)^T \\ \cdot \\ \cdot \\ \cdot \\ \bar{\phi}(x_N)^T \end{bmatrix}_{N\times M}$, ~~then~~ (??) can be expressed as

equation

$$\Phi\bar{w} = \bar{y}$$

$$(\Phi^T\Phi)\bar{w} = \Phi^T\bar{y} \tag{7}$$

$$\bar{w} = (\Phi^T\Phi)^{-1}\Phi^T\bar{y} \tag{8}$$

The point of this derivation is to show that the optimal set of weights can be computed analytically. (7) is a system of linear equations. This value of $\bar{w}$ corresponds to the regression fit for the given data samples. Given a basis set $\bar{\phi}$ and the data values, one can construct the matrix Φ. (7) can therefore be solved for $\bar{w}$ by using Gaussian Elimination (using pivoting) or LU-Decomposition. Alternatively, one can solve for $\bar{w}$ using (8) by computing the *pseudo inverse* $(\Phi^T\Phi)^{-1}$. Computing the inverse however is numerically unstable and hence, LU-Decomposition is usually preferred.

In this experiment, $\bar{\phi} = [1, sin\,x, cos\,x, sin\,2x, cos\,2x, \ldots, sin\,mx, cos\,mx]$ is an orthogonal basis set. Hence Φ will be of the following form:

lowercase.

$$\Phi = \begin{bmatrix} 1 \; sin\,x_1 \; cos\,x_1 \; sin\,2x_1 \; cos\,2x_1 \; \ldots \; sin\,mx_1 \; cos\,mx_1 \\ 1 \; sin\,x_2 \; cos\,x_2 \; sin\,2x_2 \; cos\,2x_2 \; \ldots \; sin\,mx_2 \; cos\,mx_2 \\ \cdot \\ \cdot \\ \cdot \\ 1 \; sin\,x_N \; cos\,x_N \; sin\,2x_N \; cos\,2x_N \; \ldots \; sin\,mx_N \; cos\,mx_N \end{bmatrix} \tag{9}$$

You use upper case later.

This is a bit confusing. In ~~p~~ you called this $\bar{\phi}(x)^T$. And in page 4, after equation (1) you called it $\bar{\phi}(x)$ but then put the "T" in the right hand side. Try to use always the same format for the notation or you will confuse the reader.

(III) Experiment

The experiment involves ~~coming up with~~ determining the optimal number of terms ~~in the orthogonal basis set~~ that ~~to~~ best approximate the underlying function. As the number of terms ~~is varied~~ changes, the hypothesis or the approximating function also changes. As one increases the number of terms, the regression fit ~~gets better~~ improves i.e., the sum of squared errors decreases. The question that one needs to address is whether one can afford to increase the number of terms unconditionally.

I don't think the point you are trying to make here is clear. I mean, ~~clearly, the best~~ The fact that we cannot have an infinite amount of terms and, therefore, we need to determine when we can stop while still being under a certain error bound. So it is not really the "optimal number of terms that best ....", but the "minimal number of terms that ensure..."

Proposition:- The Minimum Message Length (MML) framework can provide a direction in answering the above question. I am using MML ~~as a de-~~ termining criterion to predict the optimal number of terms to be used in the approximating function. The whole point of this exercise is to determine the hypothesis (the number of terms plus their corresponding weights). This involves choosing the hypothesis which results in the least overall message length required to encode the hypothesis and the data given the hypothesis.

citation.

You are going to need to introduce MML in some detail. In fact you do not need yet to talk about this. You just need to show with the graphs how things vary depending on the number of terms M provided. So I would change the title ("Experiments") and the first two paragraphs to simply explain the ~~proposition~~ (~~that~~ well known fact that the fit changes considerably with M and the rest. And only later, in section IV talk about MML and your aim.

To verify the above proposition, data points are generated from a known function and noise is added to them. For a given number of terms, the

N M

weights corresponding to the sum of least squares ~~is~~ are computed. This would be the hypothesis that is used to approximate the data. The message length for this setting is then evaluated. This process is repeated for increasing ~~value of~~ number of terms and the value at which the length of the encoding message is minimum is regarded as the optimal number of terms. It is important to note that, ~~usually,~~ in practice the function which is used to generate the data is often unknown.

For simulation purposes, the above tests are done by generating data from two functions, namely, the **sawtooth** and **square** functions shown in Figures 1 and 2 respectively. Each of these functions can be represented using an infinite Fourier series representation as detailed in (2) and (3) respectively.

Data Generation

- *Generating X's:* For a given range and the number of samples, the x values are generated randomly as per a uniform distribution.
- For a chosen sawtooth/square function, the function values $f(x)$ corresponding to the x values generated are then computed.
- *Generating Y's:* To the previously generated $f(x)$ values, *Gaussian noise* is added to account for any errors in the actual experiment conducted, as follows:
$$y = f(x) + \epsilon \quad \text{and} \quad \epsilon \sim N(\mu, \sigma)$$
where μ and σ are the parameters of the Normal distribution and ϵ is the amount of noise added.

Parameters

The parameters for a model are specified during runtime. They are as follows:

- Number of data samples (N)
- Number of terms in the Fourier series under consideration (M)
- Range (lower bound & upper bounds of the interval) from which the x values are generated
- The function according to which data is generated
- The mean (μ) and standard deviation (σ) of the Gaussian noise
- Time period of the chosen ~~wave form~~ function (T)

→ it is indeed a wave form but you never called it that.

Regression fit

- For a given ~~value of the number of terms~~ M ~~the matrix~~ Φ is evaluated as shown in (9). ~~This would be~~ a $N \times M$ rectangular matrix.
- The system of linear equations in (7) is constructed and solved for the weight vector $\bar{w}$.

→ two plots cannot really "verify" any thing, only show, suggest ... etc.

- $\bar{w}$ corresponds to the linear regression fit for the model. Using this $\bar{w}$, the predictions ($\hat{y}$) for the x values are computed using (1). The greater the number of terms, the better the fit in terms of minimizing the squared error (4).
- Figure 3 presents ~~represents~~ the regression fit using the first 3 terms $\{1, sin\,x, cos\,x\}$ from the orthogonal basis set while ~~and~~ Figure 4 ~~r~~epresents the regression fit using the first 9 terms $\{1, sin\,x, cos\,x, \ldots, sin\,4x, cos\,4x\}$. These two plots verify ~~the fact that~~ increasing the number of terms results in a better fit (minimized squared error).

→ show how

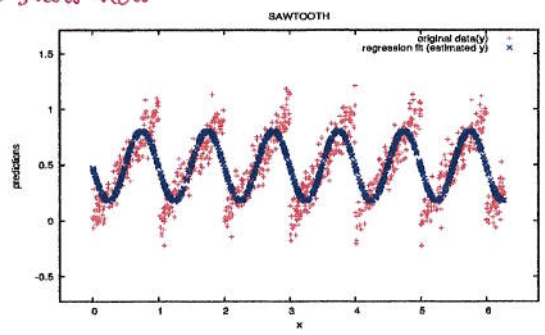

Figure 3: Regression fit for *sawtooth* ~~wave~~ using $M = 3$ terms and $\sigma = 0.1$ and μ? N? T?

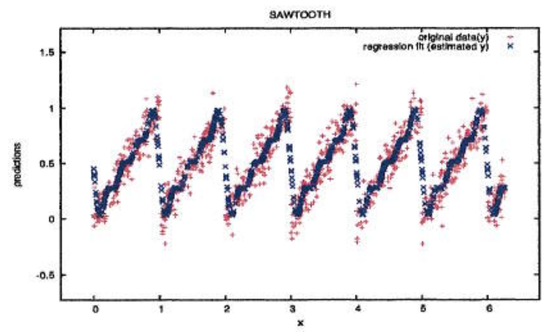

Figure 4: Regression fit for *sawtooth* ~~wave~~ using $M = 9$ terms and $\sigma = 0.1$

- Figure 5 presents the regression fit ~~shows the case~~ when the data is generated without any noise. When the regression is done with 9 terms for a sawtooth wave,

→ It is always a good idea to use exactly the same words when possible, so that the reader links things in their heads.

the weights (coefficients in the Fourier series representation (2)) are zero for the cosine terms and non-zero for the sine terms. Hence, the contribution is mainly due to 1, $sin\,x$, $sin\,2x$, $sin\,3x$, $sin\,4x$ terms. That is the reason why we see 5 intermediate peaks (sub-waves) ~~the blue curve~~ in the approximating (blue) function.

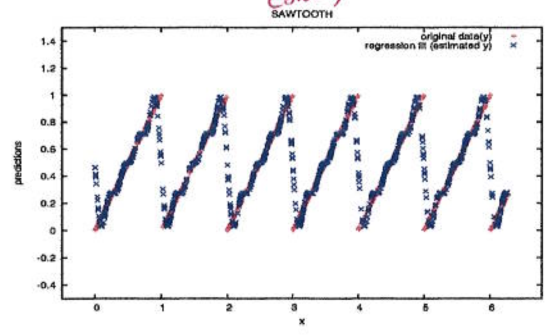

Figure 5: Regression fit for *sawtooth* ~~wave~~ using $M = 9$ terms and $\sigma = 0$

- Similar behaviour is observed for a square wave. Figure 6 represents the fit using 4 terms and Figure 7 represents a fit using 10 terms.

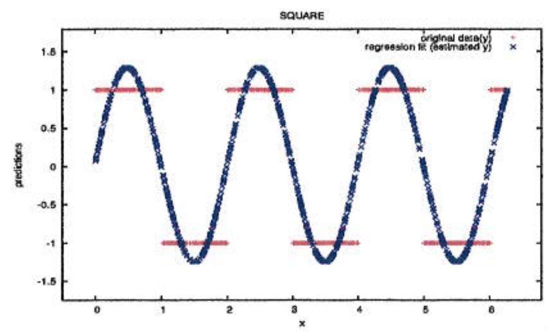

Figure 6: Regression fit for *square* ~~wave~~ using $M = 4$ terms and $\sigma = 0$

→ straight to the no-noise case?

- For a square function ~~wave~~, the coefficients of its infinte Fourier series representation (shown in (3)) are non-zero for odd sines. Therefore, the main contribution is due to the $sin\,x$, $sin\,3x$, and $sin\,5x$ (3 peaks) terms when

Make it into a new section IV and start with a brief intro of what you want to use MML for.

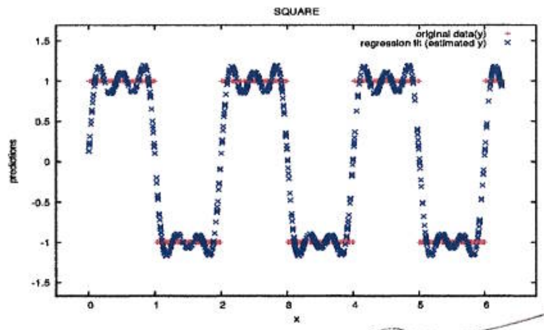

Figure 7: Regression fit for *square* wave using $M = 10$ terms and $\sigma = 0$

why now 10, not 9?

10 terms are used in the approximation. When 4 terms are used, the contribution is only due to the $sin\, x$ (just 1 peak) term. Hence in Figure 6, a distinct sine-like curve is observed.

I would put this first.

Computation of Message Length

No need for bullet points.

- Message Length can be thought of as the length of the ~~encryption~~ encrypted message a transmitter sends across. There is a receiver at the other end of the transmission channel and he decodes the encrypted message. Both the transmitter and the receiver adhere to a *codebook* which contains some basic things that both agree on. We are interested in lossless transmission and if the message is sent as per a Gaussian distribution, it is referred as a Gaussian channel.

  what transmission channel? You have not introduced this before!

- In to a MML framework, for a given hypothesis (in our case the number M of terms, and the weights $\bar{w}$), the message length is computed as the number of bits used in encoding the hypothesis and the data given the hypothesis. A message has two components. In this experiment, the hypothesis refers to the ~~model in consideration~~. ~~The model is defined by the~~ number of terms and the weights. This is the first component. The second part of the message encodes the data given a particular model. (The N points, μ and σ)

  what data? You have not talked about dataset. (need to link it to the data you generated before).

- To transmit a set of observations assuming that they are drawn from a Gaussian distribution, one needs to encode the Gaussian parameters (μ and σ), and then ~~need to~~ encode the observations using this distribution. This two part message when computed ~~turns out to be~~ has the following expression using the Wallace Freeman approach.

$$\frac{1}{2}\log\frac{N\sigma^2}{N-1}+\frac{1}{2}(N-1)-\frac{1}{2}N\log\frac{2\pi}{\epsilon^2}+\frac{1}{2}(2N^2)+\log(R_\mu R_\sigma)+1+\log(K_2) \tag{10}$$

The length

Are you going to explain this? You need at the very least to provide an intuition for every term, and for what it corresponds to (a single number? The entire message ....)

Start slower. Set the context; I mean talk about what MML is: an information-theoretic ~~technique~~ approach that states that the best explanation of observed data is the shortest. Thus, it can be used to compare different fits by .....

Not really, you are estimating them too in, and for example.

where N – number of observations, as before

ϵ – accuracy of measurement

R_μ– range of mean of normal distribution

R_σ– range of $\log(\sigma)$ of normal distribution

K_2– lattice constant → give an intuition

R_μ and R_σ are dependent on the type of problem one is dealing with. They are usually chosen based on the domain knowledge. As an example, if one is interested in encoding the heights of individuals, one can assume that average height to be between 5 and 6 feet. R_μ would then be $6-5=1$. One could similarly ~~come up with~~ find reasonable estimate for R_σ. ~~In the current~~ For my experiment, I ~~have~~ show how to estimate these parameters in (11) and (15).

- *Encoding weights*: The transmitter needs to send the weights across to the receiver. It is assumed that the weights correspond to values sampled from a normal distribution and, hence, the message length used to encode these weights is computed as per (10). To ~~calculate~~ do this ~~the message length~~, one needs to estimate the parameters of the distribution ~~and also compute~~ together with R_μ and R_σ ~~to be used in (10)~~.

From the infinite Fourier series representation of the sawtooth and square ~~waveforms~~ functions shown in (2) and (3), one can see that the coefficients (weights) of the sine and cosine terms are of the form $\pm\frac{1}{n\pi}$. This means the magnitude of the weights is always less than 1. This helps in determining the bounds for each weight w_i, since ~~Hence~~ $|w_i| < 1$ and thus $R_\mu = 2$.

To determine R_σ, one needs to estimate the bounds for σ. ~~Now~~ Clearly, σ cannot be zero as this would mean that the deviation from the mean is zero and all observations are the same as the mean. Hence σ is bounded from below by ϵ, the accuracy of measurement. The lower bound of σ is set to be a constant times ϵ. The magnitude of the weights decreases as the number of terms increases in the Fourier series representation. The upper bound of σ is taken to be 1.

you need to explain this: not being able to ... σ does not immediately mean being bounded by ϵ

what constant?

how does the second phrase follow the first? I mean, why 1 from the fact it decreases?

- *Encoding X*: In the experiment, the x values are sampled from a predefined range $[a, b]$. Further, they are sorted in increasing order. The first term of this sorted sequence is made 0 and ~~this~~ is part of the codebook, which means it does not need to be transmitted. The other x values are scaled accordingly.

  have you used upper case X before?

Instead of sending the x's, what is sent is the difference Δx between consecutive x values. This ~~will~~ enables the receiver to construct the current x value using the previous x value received and the difference Δx. Sending Δx's results in a compact message ~~as well~~. Hence, information is sent in an efficient manner. Δx's are sent over a Gaussian channel. The overall message length to encode Δx's is computed as per (10). However, we still need to estimate R_μ and R_σ.

Explain!

For a set of x's which are sampled from $[a, b]$, one can compute the sample mean and variance. Further, one can derive the bounds for the mean and standard deviation. The following is how I estimate R_μ and R_σ.

This is new. In the R_μ before you did not add was the weights

1. To estimate $R_{\mu_{\Delta x}}$

$$x \in [a,b] \Rightarrow a \leqslant x \leqslant b$$
$$\therefore a \leqslant x_i \leqslant b \quad \text{and} \quad -b \leqslant x_j \leqslant -a$$
$$\text{If} \quad \Delta x = x_i - x_j, \quad a - b \leqslant \Delta x \leqslant b - a$$
$$\therefore a - b \leqslant \mu_{\Delta x} \leqslant b - a \qquad (11)$$

what is this ?

didn't you said they were sorted in increasing order ? So why is $\Delta x \in [a-b, b-a]$: arent you always sendin x_{i+1} after x_i?

The above result means if a set of observations are sampled from a known range, the mean of those observations also lies in that range. $\Delta x \in [a-b, b-a]$, and hence $\mu_{\Delta x}$ also lies in that range. Hence $R_{\mu_{\Delta x}} = 2(b-a)$

2. To estimate $R_{\sigma_{\Delta x}}$

$$\sigma^2_{\Delta x} = \sum_{i=1}^{N-1} \frac{(\Delta x_i - \mu_{\Delta x})^2}{N-1}$$

The following exercise is done to determine the bounds of $\sigma_{\Delta x}$

$$\text{Consider} \quad (\Delta x_i - \mu_{\Delta x})^2 = \Delta x^2 + \mu^2_{\Delta x} - 2\Delta x \mu_{\Delta x}$$
$$a - b \leqslant \mu_{\Delta x} \leqslant b - a \quad \Rightarrow 0 \leqslant \Delta x^2 \leqslant (b-a)^2 \qquad (12)$$
$$\text{and} \quad a - b \leqslant \mu_{\Delta x} \leqslant b - a \quad \Rightarrow 0 \leqslant \mu^2_{\Delta x} \leqslant (b-a)^2 \qquad (13)$$
$$\text{Also} \quad -2(b-a)^2 \leqslant -2\Delta x \mu_{\Delta x} \leqslant 2(b-a)^2 \qquad (14)$$

I will have a look at the ~~th~~ when I understand

Adding (12), (13), (14) :-

$$-2(b-a)^2 \leqslant (\Delta x_i - \mu_{\Delta x})^2 \leqslant 4(b-a)^2$$
$$\therefore 0 \leqslant \frac{(\Delta x_i - \mu_{\Delta x})^2}{N-1} \leqslant \frac{4(b-a)^2}{N-1}$$
$$0 \leqslant \sigma^2_{\Delta x} \leqslant \frac{4(b-a)^2}{N-1}$$
$$\therefore 0 \leqslant |\sigma_{\Delta x}| \leqslant \frac{2(b-a)}{\sqrt{N-1}} \qquad (15)$$

why 3 ?

Equation (15) gives an upper bound on $\log(\sigma_{\Delta x})$. Since $\sigma_{\Delta x}$ cannot be zero as it is a measure of the deviation from the mean, a lower bound is assumed for $\sigma_{\Delta x}$ and is set to 3ϵ (a constant factor of the accuracy of measurement). Hence, $R_{\sigma_{\Delta x}} = \log \frac{2(b-a)}{\sqrt{N-1}} - \log(3\epsilon)$.

Using these values of $R_{\mu_{\Delta x}}$ and $R_{\sigma_{\Delta x}}$ in (10), the message length to encode Δx is computed.

That is true for $\sigma = 0$. But as soon as $\sigma \geq 0.1$ The plots are not that great. I mean, VERY soon the message length grows a lot. Have you had a look at actual results with figures like. Fig 4 and 5 to see that the results of the fit match the plots ?

again, uppercase ?

- *Encoding Y*: The receiver can decode and infer the weights and x values sent by the transmitter. Further these two can be used to construct $\hat{y}$ using (1). Since the transmitter knows that the receiver has access to this information, instead of sending the original y values, the difference $\Delta y = \hat{y} - y$ is transmitted. The receiver can then construct the original y on his side of the channel. Transmitting the differences in y values is an efficient form of transmission as the encoding/decoding is done using previous knowledge and there is no redundancy in the data transmitted.

To compute the length of encoding the Δy's, it is again assumed that they are derived from a Gaussian distribution and hence the meesage length follows (10). → You mean, have R_μ an R_σ computed as ? x? but they have different distribution, no ?

(IV) Observations

- The above experiment is run for different values of number of data points (N), and different values of Gaussian noise (σ). Data is generated in the range $[0, 2\pi]$ and time period of the wave forms set to 1. ε ? The below discussion is valid for both the sawtooth and the square ~~waves~~. functions. The is
- It should be expected that the message length continuously decreases as the number of terms (M) increases, reaches a minimum, and then starts to increase. ~~This is the ideal behaviour that one would expect.~~
- The theory is validated for $N = 1000$ and $N = 10000$ for varying values of σ. Please refer to the attached plots. The plots show the ideal behaviour. However the plots become erratic when the message lengths are plotted for $N = 100$.

where are these ? I see many plots in the directories, can you put them ALL in a text document and comment them ?

- **An interesting observation**: The optimal value of M is the one which results in the overall minimum message length. It was interesting to note that the weights corresponding to a value of M are the corresponding coefficients in the Fourier expansion. But this is observed as long as the graph behaves ideally. There is a strong correlation between the erratic behaviour and the divergence of the inferred weights from the Fourier coefficients – I think this is a potential indicator of strange behaviour! (more about this is in the discussion below)

<u>The N = 100 case</u>:-
The following discussion is with respect to the sawtooth function.

- For a given value of σ, the graph tapers off at the beginning and then steadily increases until a certain value of M. From then on, the ideality is lost and the graph is erratic. It doesn't seem to follow a pattern from then on. It behaves randomly and at times, the message length increases and decreases with no clear indication as to why.

- Now I investigated as to why this was happening. I thought I will eliminate the most obvious choice first – problem with my code. There might be a problem with the way I am solving for $\bar{w}$ in (7). I used three different techniques to do this.
  - I first tried the analytical solution, trying to compute $\Phi^T\Phi^{-1}$ and solving for the weights analytically as in (8). I implemented the matrix inverse using Gaussian elimination (with partial pivoting).
  - Secondly, I used the **boost** library implementation of LU-Decomposition to solve for the matrix inverse.
  - Thirdly, I used my own implementation of LU-Decomposition to solve for the linear system (7).
  - **All the three methods produce the same results.** So there was nothing wrong with the way I was solving the linear system.

Good.

  - I did a few other sanity checks just to make sure that my code is not playing up. I computed the determinant using my implementation and compared it with **boost** implementation. Both of them agree.

you mean, other parts of your code?

  - I checked whether the Φ matrix was being generated correctly. It seems to be the case.
  - I used this Φ matrix and used **MATLAB** to compute $\Phi^T\Phi$ and eventually computed the weights. These weights match the ones computed using my implementation. There however seem to be differences in $\Phi^T\Phi^{-1}$ but they differ in the fractional part of the matrix elements. I think it is because of computational errors that creep in when dealing with large matrices. The weights computed are The same nevertheless.
- For different values of M, I started printing out the corresponding weights and analysed them. Please refer to the plot for the sawtooth function ($N = 100, \sigma = 0$). The graph shows ideal behaviour until about $M = 55$ or so and then it starts behaving randomly. I printed the weights ranging from $M = 1$ to $M = 100$. As long as the ideal behaviour was exhibited, the weights approximately match the Fourier coefficients. But they begin to diverge from then on. Consider the following specific values of M:
  - $M = 50$ – well behaved, everything is normal
  - at about $M \sim 55$ – erratic behaviour begins
  - From $M = 79$ to $M = 80$, the message length decreases. There is a reduction in the length of encoded message. This is bizzare as the message increases suddenly decreases and increases.

I've included the values of weights for these three cases in `comparison_msglen.txt` file for you to refer to. If you see, the weights for $M = 50$ case match the Fourier coefficients. But when $M = 79$ or $M = 80$, they are clearly far apart from their corresponding Fourier coefficients.

whereas the data with the exact Fourier coefficients?

- I've also plotted the message length for part 1 (to encode weights) and part 2 (encoding data given weights). Please refer to the `comparison_msglen.eps` file. If you see there, the part 1 of the message is the one that is causing the problem. Part 1 of the overall message deals with encoding the weights. This forces me to think that there is something weird ~~that~~ is happening when the weights are being computed for higher values of M.
- Another important point to note is if you see the message length for part 2, it consistently keeps decreasing which points to the fact that the root mean squared error is steadily decreasing with increasing value of M. This is however expected because as the model complexity increases, the error of the regression fit decreases.
- Since this erratic behaviour is observed for $N = 100$ data points, I was curious whether I can reproduce the similar behaviour with $N = 1000$ data points. So I ran the experiment for 1000 data points (with $\sigma = 0.1$) and let it run until the $\Phi^T\Phi$ matrix becomes singular. As it turns out, this behaviour exists but for sufficiently large value of M. After about $M = 400$ or so, the same bizzare behaviour appears and continues until $M = 820$ when the matrix becomes singular and the program terminates. Please refer to this plot which I included in the folder.
- This forces me to think that the method breaks down for values of M which are comparable to the value of N.