A Model of Ocean-Atmosphere Flux of Carbon Dioxide in Response to Tropical Dynamics

Candidate Number: 341065
Supervisor: Dr L. Zanna
Word Count : 5805

April 28, 2014

Abstract

I propose a low order model to describe the carbon dioxide flux in the equatorial Pacific in response to the El Niño Southern Oscillation. I assume one contribution due to the carbon flux from the temperature of the water and the wind speed at the surface of the ocean, and a second contribution due to water from the deep ocean being drawn up to the surface. Using this approach, I am able to construct a solution which agrees well with observation on the magnitude of the carbon flux, the difference in flux between the east and the west of the ocean and the inter-annual variation in flux. I am also able to provide an estimate of the carbon flux based on a dynamical model of El Niño Southern Oscillation. The model is not however able to capture the north, south variation in flux in this area of the ocean, probably as the model ignores advective currents.

1 Introduction

In this project, I aim to construct a simple model which describes the flux of carbon dioxide between the ocean and the atmosphere in the equatorial Pacific. In particular I aim to investigate how the CO_2 flux is affected by the El Niño Southern Oscillation (hereafter ENSO).

An El Niño event is defined as anomalously warm water in the equatorial Pacific for an extended period of time, typically several months. The temperature anomaly is usually strongest in the east or the centre of the Pacific basin. The corresponding event with colder than average water is named La Niña. An example of each of these events is shown in Figure 1. ENSO is the term given to the irregular cycle of El Niño and La Niña event, and has a time period of between 2 and 7 years. [1]

The following idealised model of the processes which dominate the dynamics in the equatorial Pacific helps to explain why ENSO occurs.

The ocean in the equatorial Pacific can be thought of as two layer ocean. This is a simple model in which the ocean is described as having two distinct bodies of water, an upper, mixed layer, and a lower, deep ocean. For a given point in space and time, the properties of each layer are considered to be constant over the entire depth of that layer. Only the mixed layer is able to interact with the atmosphere, and the properties of the deep ocean change only very slowly. On the time period of interest (the time period of ENSO of a few years) we take the properties of the deep ocean to be constant.

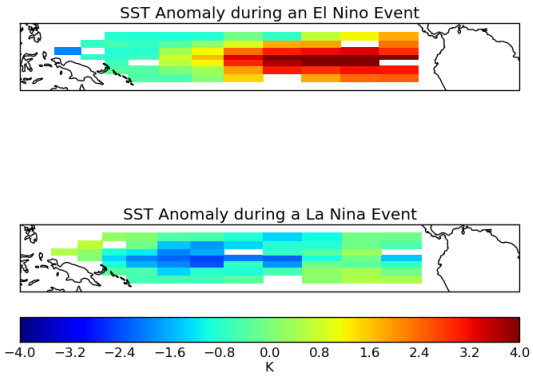

Figure 1: A comparison of the anomaly in Surface Sea Temperatures in the equatorial Pacific during an El Niño Event (1998) and a La Niña Event (1999). Data from [3]

Under normal conditions, (neither El Niño nor La

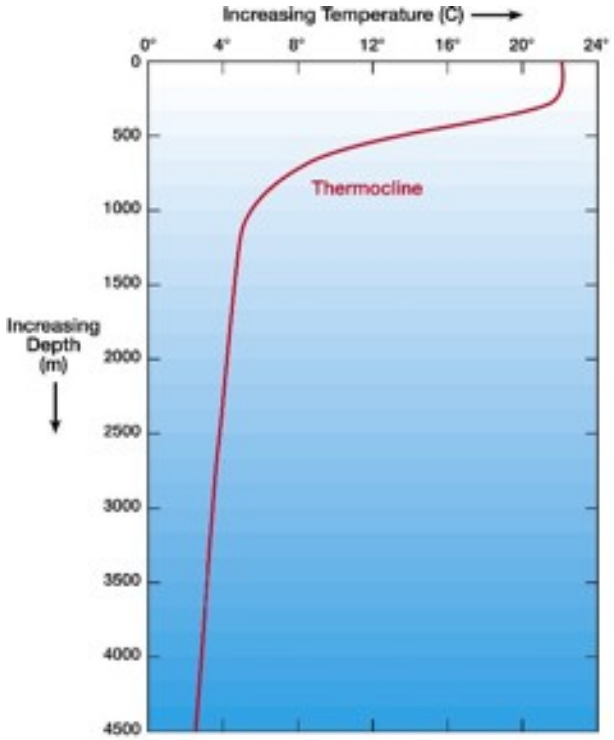

Figure 2: A typical low latitude ocean temperature profile as a function of depth. Image from [2]

Niña) the easterly trade winds create a mean westerly flow in the water at the surface of the ocean. In the east of the basin, this causes water from the deep ocean to be drawn to the surface to replace the water moving to the west. I shall refer to this process of deep ocean water being drawn up to the surface as upwelling. This deep ocean water is much cooler, (as shown is Figure 2), resulting in a cooler sea surface temperature (hereafter SST) in the east of the basin than in the west. A further effect of this upwelled water can be seen by examining the depth of the thermocline along the equator. The thermocline is a layer in the ocean in which the temperature of the water changes rapidly with depth. In a two layer ocean, this corresponds to the surface which marks the boundary between the mixed layer and the deep ocean. The thermocline depth is affected by the amount of upwelling, as the more upwelling is present, the smaller the volume of warm, mixed layer water, giving a shallower thermocline. Figure 3 confirms that under normal conditions, the thermocline is much shallower in the east of the basin where the upwelling is strongest. During an El Niño event, the profile of the thermocline is much flatter, making it much deeper than normal in the east. This corresponds to warmer than average SST here. The reverse is true during a La Niña event, with a steeper profile indicating more upwelling in the east and so lower than average SST. It is therefore possible to characterise ENSO using the concept of thermocline depth.

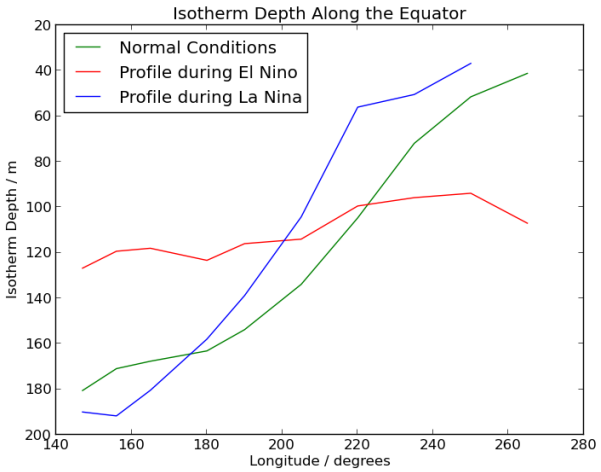

Figure 3: Comparison of the isotherm depth along the equator, showing the mean depth, (normal conditions), the depth during the El Niño event of 1998, and the depth during the La Niña event of 1999. Data from [3].

If the system is perturbed away from this mean state, for example by creating a positive SST anomaly in the centre of the Pacific, then we weaken the easterly trade winds (the Bjerkness hypothesis [4]). A reduction in the trade winds should reduce the flow of water to the west, and so reduce the amount of upwelling. The mechanism for this effect is the creation of waves in the depth of the thermocline. The anomaly in the centre of the basin creates an eastwards moving downwelling equatorial Kelvin wave causing a deeper than normal thermocline to the east of the anomaly. At the same time, a westwards propagating upwelling Rossby wave is created near the equator and causes a shallower than normal thermocline to the west of the anomaly. This is now an El Niño event, with the thermocline deeper than normal in the east, and shallower than normal in the west. The event is maintained by the strong influence of the SST in the east of the Pacific on the winds above it, but eventually the westwards moving Rossby waves will reach the western edge of the Pacific, and are reflected as upwelling Kelvin

waves which cancel the downwelling Kelvin waves and end the event. Frequently, the returning waves will cause the system to overshoot the mean state, triggering a La Niña Event. (Further knowledge of Kelvin and Rossby waves is not required here, except that they propagate with different, well known speeds).

ENSO is the largest observed source of variation in inter annual global SST, and as a result of this, we expect it to also have a large impact on the variability of the CO_2 flux between the ocean and the atmosphere. Understanding ENSO, and how it affects the exchange of CO_2 is therefore an important step in understanding how CO_2 is stored in the ocean. However, ENSO has proved to be a feature that is very difficult to model accurately, with the current climate models struggling to reproduce the correct combination of amplitude, location and power spectrum of ENSO. As a result, predictions of the carbon flux in this region are generally unreliable. Furthermore, due to the complex, numerical nature of these models, it is often difficult to gain any physical insight into the processes occurring.

In this project I therefore take a step back from these complex models, and instead attempt to build a low order, physically intuitive model of how the carbon flux varies during ENSO. The main goals of this project are to create a model which is as simple as possible while still capturing the main effects, gives an answer which is easy to interpret physically, and is driven by observational data.

To do this, I begin by constructing a model of the carbon flux in terms of measurable quantities, and use observational data to test it. I then look to parametrise this model such that it depends on only one 'El Niño Strength' variable. Finally, I use this to provide an estimate of the carbon flux between the ocean and the atmosphere based on a dynamical model of the El Niño Southern Oscillation.

2 Experimental Method

In constructing the model, I assume that the two factors which have largest effect on the carbon flux in this region are the surface temperature of the water, and the rate of upwelling. Two notable factors that I have chosen to ignore in simplifying the system are the biological life in the ocean and the horizontal advective currents. I also assume that it is possible to treat the flux due to the temperature of the water and the flux due to the upwelling as independent effects.

2.1 Flux due to SST

The SST is expected to have an effect on the amount of flux observed as both the solubility of a gas in water and the partial pressure of the gas depend on the temperature of the water. As the water heats up, the solubility decreases, the partial pressure increases, and so a more positive carbon flux is expected at higher temperatures.

It is intuitive that the flux of carbon dioxide out of the water will be proportional to the difference in concentration of CO_2 between the ocean and the atmosphere, giving

$$F_{SST} = k \cdot \Delta[CO_2]. \tag{1}$$

The constant of proportionality here is the gas transfer velocity, which is typically taken to be a function of the wind speed at the surface of the ocean only. We use the form suggested in [6] of;

$$k(v) = 87.6 \cdot (0.31v^2 - 0.71v + 7.76). \tag{2}$$

It is convenient to present the carbon flux in units of $mol{\cdot}m^{-2}{\cdot}yr^{-1}$, requiring k in units of $m{\cdot}yr^{-1}$ and $\Delta[CO_2]$ in $mol \cdot m^{-3}$. (The factor of 87.6 in Equation 2 is due to a unit conversion from $cm \cdot hour^{-1}$.) From here, Henry's constant $H(T)$ is used, which relates for a given temperature, the concentration of gas in a liquid to its partial pressure via

$$H(T) = \frac{[CO_2]}{pCO_2}. \tag{3}$$

$H(T)$ is a constant for a given temperature, so a value for it can be obtained by looking at a simple system with still water and gaseous CO_2 at a pressure of one atmosphere above it. The partial pressure of CO_2 is then one atmosphere (approximately the surface atmospheric pressure P) and the concentration of CO_2 in the water is then simply the solubility of the water at this temperature. The solubility of CO_2 in water is shown in Figure 4, which motivates an approximation of the solubility in this temperature range as

$$S(t) = \frac{1000}{44} \cdot (22.4 - 0.07T). \tag{4}$$

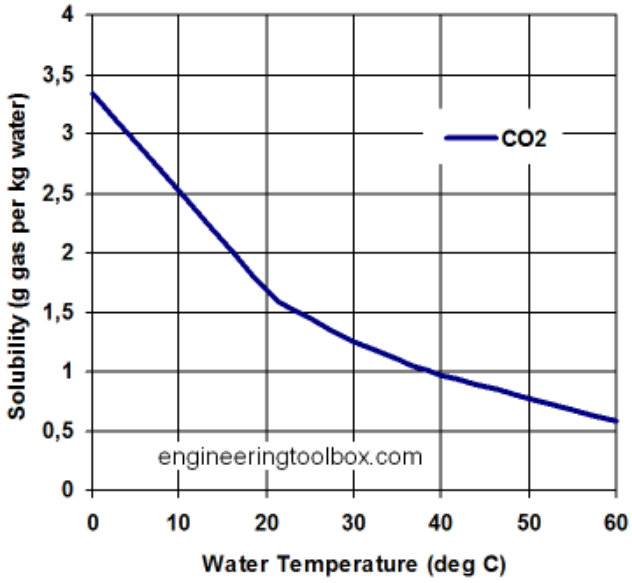

Figure 4: The solubility of carbon dioxide in water as a function of temperature. Image from [5]

(The numerical pre-factor is a unit conversion from $g \cdot kg^{-1}$ into $mol \cdot m^{-3}$ assuming a constant density of water of $1000 kgm^{-3}$.)

Combining Equations 3 and 4 gives

$$H(T) = \frac{1000 \cdot (22.4 - 0.07)}{44 \cdot P}. \tag{5}$$

Using Henry's constant, Equation 1 can be rewritten as

$$F_{SST} = H(T)k(v)\{pCO_2^{ocean} - pCO_2^{air}\}, \tag{6}$$

where pCO_2 is the partial pressure of CO_2. In order to progress further, I make use of the results of [7] to parametrise the pCO_2^{ocean} as

$$pCO_2^{ocean} = Ae^{0.0423T}, \tag{7}$$

where A is a parameter to be chosen later.

Finally, assuming a constant mole fraction of CO_2 in the atmosphere, f, gives

$$pCO_2^{air} = fP, \tag{8}$$

where P here is the atmospheric pressure at the surface of the ocean.

This gives a carbon flux due to SST of the form

$$F_{SST} = Hk\{Ae^{0.0423T} - fP\}. \tag{9}$$

2.2 Flux due to Upwelling

There are several mechanisms by which CO_2 is drawn down into the deep ocean. These include regions of the ocean where water is drawn down from the surface, bringing with it any dissolved CO_2, and biological life, which has a net effect of absorbing carbon during its life, and carrying it down to the deep ocean on death. For reasons such as this, the deep ocean is richer in CO_2 (and many other nutrients) than the mixed layer. Upwelling is therefore expected to have an effect on the CO_2 flux because as this water is brought to the surface, it warms up, the solubility decreases, and so some of the CO_2 contained within it is released.

The amount of CO_2 released per unit area in this process is calculated as being equal to the change of concentration of CO_2, multiplied by the upwelling velocity, w. I roughly estimate the concentration of CO_2 in the deep ocean water as being equal to the concentration that would exist if water at the temperature of the deep ocean was in equilibrium with the atmosphere, and do the same for the mixed layer concentration.

As before, this gives,

$$[CO_2] = \frac{f \cdot 1000 \cdot (22.4 - 0.07T)}{44}, \tag{10}$$

which suggests a change in the concentration of carbon dioxide in the water as it rises from the deep ocean to the mixed layer of

$$\Delta[CO_2] = \frac{70 \cdot f \cdot \Delta T}{44}, \tag{11}$$

where ΔT is a parameter representing the change in temperature experienced by the water as it rises.

Finally, this results in a flux due to an upwelling velocity, w, of

$$F_{upwelling} = \frac{70 \cdot f \cdot \Delta T w}{44}. \tag{12}$$

2.3 Total Carbon Flux

Adding the contribution from SST, (Equation 9), and the contribution from upwelling (Equation 12), gives the following equation for the total carbon flux

$$F_{total} = Hk \cdot \{Ae^{\alpha T} - fP\} + \frac{70 f \Delta T w}{44}. \tag{13}$$

2.4 Estimation of Upwelling Rate

Equation 13 serves as a clear and physically intuitive description of the carbon flux due to the processes considered, however, this form of the equation is difficult to use in practice as the upwelling rate w is very small and so is almost never measured. This lack of data makes it necessary to construct a simple model to approximate the value of w from variables for which data is more available.

The region of interest is a narrow strip close to the equator, and therefore all of the data is taken from points with similar latitudes. Additionally, near the equator currents predominantly run along the equator rather than across it. These two factors motivate the assumption that if there was no upwelling anywhere in the basin, the temperature profile with depth should be identical anywhere in the basin. This is not what is observed, and we make the approximation that the deviations from the expected profile are due to upwelling cold water.

A useful variable to use here is the heat content of the water per unit area (calculated as an integral of the heat content over the top $300m$ of the ocean). The model of the dynamics in the Pacific basin presented in the introduction suggests that there should be very little upwelling in the west of the basin, and so I make the assumption that the heat content here is what we would expect to find in the absence of upwelling. This will be our reference heat content ($Heat_{ref}$). At all other points, a heat difference is then calculated, finding that the heat content is lower in the east of the basin. Figure 5 illustrates a rough method of converting this heat difference into an upwelling rate.

$$\begin{aligned} Heat_{ref} &= \rho c d T_{hot} + \rho c (300-d) T_{cold}, \\ Heat_{actual} &= \rho c (d-x) T_{hot} + \rho c (300-d+x) T_{cold}, \\ \Delta Heat &= Heat_{ref} - Heat_{actual}, \\ &= mcx\Delta T, \end{aligned} \tag{14}$$

where ρ is the density of water, c, is the specific heat capacity of water, and I have defined $\Delta T = T_{hot} - T_{cold}$.

Rearranging to find the extra depth of cold water, x,

$$x = \frac{\Delta Heat}{\rho c \Delta T}. \tag{15}$$

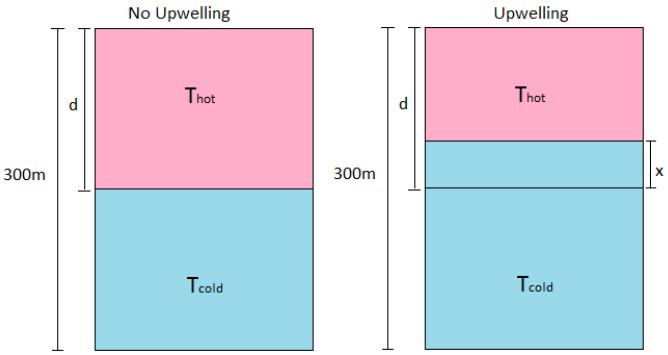

Figure 5: On the left is an idealised model of the west of the basin, where there is no upwelling. There are two layers in the ocean, with a depth d of warm water, and the remainder of the $300m$ is colder, deep water. On the right, representing the east of the basin, where the heat content is lower, we attribute the heat difference to an extra depth x of cold water.

In order to obtain an upwelling velocity from the additional depth of cold water, I introduce a parameter, R, the replenishment rate, which represents the frequency with which that depth of cold water is replaced.

From this, the upwelling rate, w, can be calculated as

$$w = \frac{R \cdot \Delta Heat}{\rho c \Delta T}. \tag{16}$$

2.5 Fitting Model Parameters to Data

Combining Equation 13 and Equation 16 produces a final estimate for the carbon flux.

$$F_{total} = Hk \cdot \{ A e^{0.0423T} - fP \} + \frac{70 f R \Delta Heat}{44 \rho c}. \tag{17}$$

Finally, values must be obtained for the model parameters.

- f: This is the mole fraction of CO_2 in the atmosphere. As the atmosphere is well mixed, f is assumed to be a constant in space and time, and has a value of 3.8×10^{-4} (atmospheric concentration of CO_2 is about 380ppm).
- $Heat_{ref}$: This is the expected heat content in the absence of upwelling. I selected the grid point (0^oN, 156W) to be the reference point,

and allow for a seasonal cycle. The reference value will therefore be the average for that month of the year.

- A: This is chosen by picking a temperature and pressure for which the carbon flux due to SST is zero. By examining Figure 10, I select a region in the west of the equatorial Pacific as our zero point, and by looking at mean SST and surface pressure in this region, obtained a value of 1.085×10^{-4}.

- R: As there is no precise data for the upwelling velocity, this was chosen to be 5000 by tuning the model to best fit the observed carbon flux. The corresponding value for w peaks at about $5 \times 10^{-4} cms^{-1}$ (assuming a ΔT of $10K$).

2.6 Effect of ENSO on Carbon Flux

I would now like to use this model to calculate how the carbon flux varies with time during ENSO. In order to do this, it is necessary to relate the variables used in Equation 17 to some parameter representing the strength of ENSO at a given time. For our purposes, we choose the thermocline depth as our ENSO parameter, as there is a large amount of data for this, and it is a common diagnostic to describe ENSO.

For this paper, I use data taken from [3]. The TAO data is gathered from a system of buoys in the Pacific which cover a region from 8^oN to 8^oS, and from 137^oE to 110^oW. These buoys gather data daily, and records go back to about 1960, although gaps in the data are frequent, especially early in the time series. The data set does not include data for the thermocline depth as such, but does include data for the 20^oC isotherm depth, which is a surface within the thermocline layer, so I use this as a substitute. Due to the frequent gaps in the data, I first convert the data to a monthly mean, thus reducing the impact of individual missing days or weeks. I then calculate a mean value for each of the variables used (SST, heat content, surface atmospheric pressure, surface wind speed and 20^oC isotherm depth). It is important to be able to separate the effect of ENSO from the effect of the annual cycle, so I calculate a separate mean for each month of the year. From this, I am able to obtain a time series of the anomaly in each of the variables, calculated by subtracting the expected value for that month from the actual value recorded. A scatter plot of the anomaly in each of these variables against the 20^oC isotherm depth anomaly is then created, and the Pearson Product-Moment Correlation Coefficient is calculated along with a least squares line of best fit. A selection of the results of this can be found in Table 1, Figure 6 and Figure 7.

Variable	PMCC	Gradient
SST	0.89	0.053
Heat Content	0.97	0.0054
Surface Pressure	-0.55	-0.018
Surface Wind Speed	0.63	0.022

Table 1: The Pearson Product-Moment Correlation Coefficient and the gradient of the line of best fit for the scatter plots between the anomaly in the variables shown and the anomaly in $20o$C isotherm depth at (0^oN, 110^oW). (Surface Pressure refers to the atmospheric pressure at the surface of the ocean).

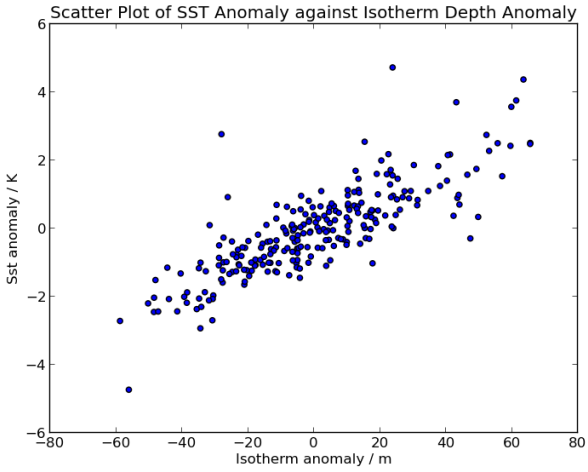

Figure 6: Scatter plot of the anomaly in SST against the anomaly in isotherm depth.

There is a strong correlation between the anomaly in SST and the isotherm depth anomaly, and an even better correlation between the anomaly in heat content and the isotherm depth anomaly. The correlations with the pressure anomaly and wind speed anomaly were much weaker, but are still considered good enough to proceed.

The gradient of the line of best fit, in combination with the expected value of a given variable allows me to provide an approximation of the value of any

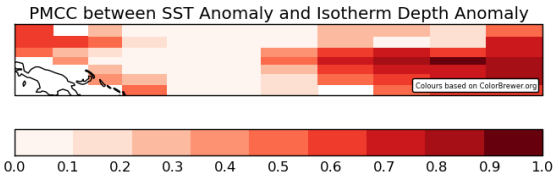

Figure 7: A plot of the correlation between SST anomaly and isotherm depth anomaly.

of the variables, simply by knowing the anomaly in the depth of the 20^{o}C isotherm. This provides a method of estimating the carbon flux using the thermocline depth as the only parameter.

2.7 A Simple Model of ENSO

I next wish to generate a prediction of the carbon flux based on a dynamical model of ENSO. In this case, we shall use the simple model of ENSO proposed in [8]. This model is excellent for our purposes as it is one of the simplest which still exhibits the low order chaotic behaviour that we observe, has a time period which approximately agrees with observation, and shows seasonal locking (a feature of ENSO where the peaks of the cycles are tied to particular times of year). It also characterises ENSO in terms of the thermocline depth alone.

The model is based around the idea of a delayed oscillator. This uses the description of ENSO that we put forward in the introduction, where we have an eastwards propagating Kelvin wave, which reaches the east of the basin a time τ_1 after being created. At this point, the coupling between the ocean and the atmosphere allows for this wave to have an impact on the centre of the basin via the wind. Along with the Kelvin wave, a westwards propagating Rossby wave is also created, and this is reflected off the western boundary of the basin and then travels eastwards as a Kelvin wave. This wave eventually reaches the east of the basin after a time τ_2 at which point it also has an effect on the anomaly at the centre of the basin.

By adding a seasonal cycle to this concept, it is possible to construct a simple equation for the rate of change of thermocline depth

$$\frac{dh}{dt} = aA[h(t-\tau_1)] - bA[h(t-\tau_2)] + c\cdot\cos(\omega t), \quad (18)$$

where a, b, and c are parameters, h is the anomaly in thermocline depth, and $A[h]$ is a function defining the coupling between the ocean and the atmosphere.

τ_1 and τ_2 can be easily calculated as the time taken for the waves to propagate the required distance;

$$\tau_1 = \frac{L}{2C_{Kelvin}} \quad (19)$$

$$\tau_2 = \frac{L}{2C_{Rossby}} + \frac{L}{C_{Kelvin}} \quad (20)$$

Here, L is the width of the Pacific basin, and $C_{Kelvin} = \frac{L}{2.3months}$ and $C_{Rossby} = \frac{C_{Kelvin}}{3}$ are the speeds of the Kelvin and Rossby waves respectively.

For the form of $A[h]$, I follow the suggestion of [9],

$$A[h] = \begin{cases} b_+ + \frac{b_+}{a_+}\{\tanh[\frac{\kappa a_+}{b_+}(h-h_+)] - 1\} & h_+ < h \\ \kappa h & h_- \le h \le h_+ \\ -b_- - \frac{b_-}{a_-}\{\tanh[\frac{\kappa a_-}{b_-}(h-h_-)] - 1\} & h < h_- \end{cases} \quad (21)$$

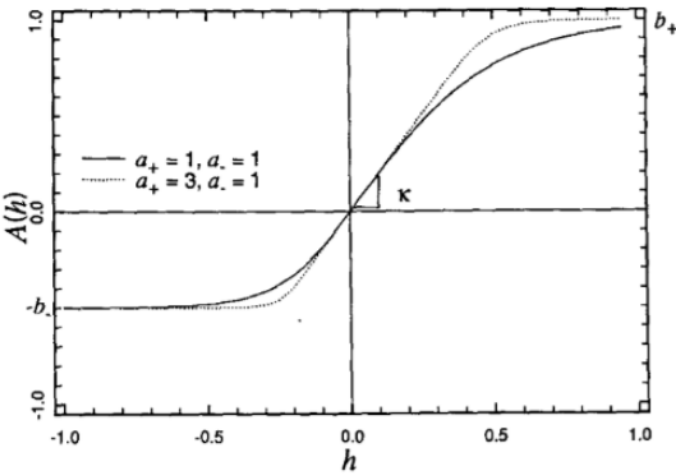

Figure 8: Form of $A[h]$ given by Equation 21. Image from [9]

The predicted form of ENSO varies greatly depending on the choice of parameters. For a full discussion of how the parameters affect the model, see [8] and [9], for our purposes, the parameters are as in [8] with $\kappa = 2.0$. This choice of parameters

gives the best possible mix of period of oscillation, seasonal locking, and chaotic behaviour. Figure 9 shows a plot of the time series of this model against the observed data.

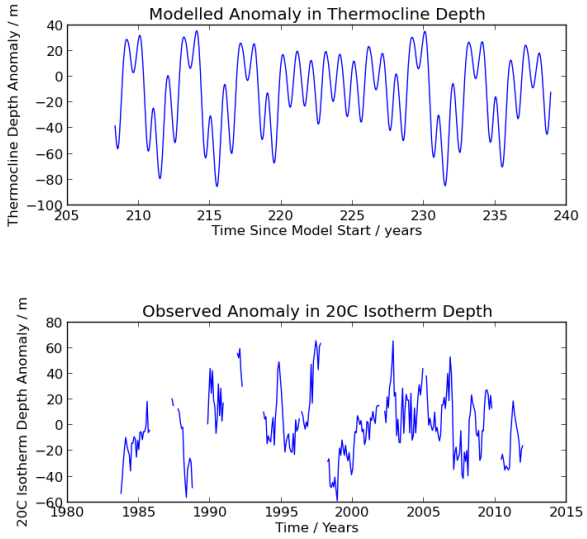

Figure 9: Comparison of the output of the Thermocline depth model and observational data of isootherm depth.

3 Results

Although in principle surface atmospheric pressure is an easily measurable variable, there are a large number of gaps in the data in both space and time. For this reason, I was not able to use the data for surface atmospheric pressure in the model. Instead, I noted that as the atmosphere is well mixed, the variance in the surface pressure is low enough that the ΔpCO_2 in Equation 6 is dominated by the pCO_2^{ocean} term, and so the differences in pressure have very little impact on the results. I have therefore chosen to input the pressure as a constant in space and time, with a value of 101kPa.

Figure 10 shows the observed global carbon flux, which acts as the truth against which the model is tested. In Figure 11, I present a plot of the time averaged carbon flux over the whole of the basin, comparing observation with the output of the model. This plot is created using Equation 17 with observational data from [3] to produce a carbon flux map for each month since 1960. The time average is then calculated for each grid point as a mean of all of the months for which there is data. In all of the plots shown, we have defined a positive value to refer to a flux of carbon dioxide out of the ocean.

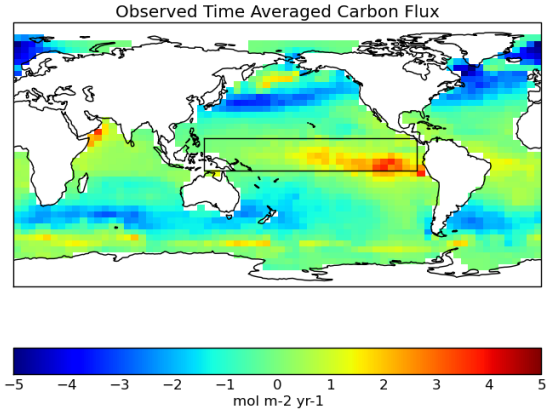

Figure 10: Observed time averaged carbon flux. Data from [3]. The highlighted box represents our region of interest for this project.

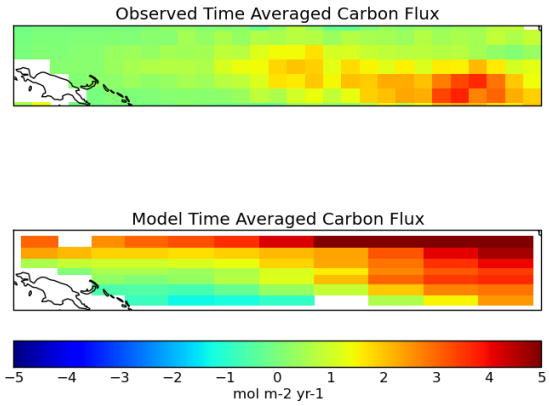

Figure 11: A comparison of the observed carbon flux and the carbon flux given by Equation 17.

It is clear that the model predicts a much larger flux than expected in the north of the basin, but in the south of the basin, the prediction matches the observational data well. Figure 12 illustrates this by plotting the time averaged carbon flux along a slice through the data at a latitude of 2^oS.

In order to better examine how the carbon flux varies with time, I select a single grid point, in this

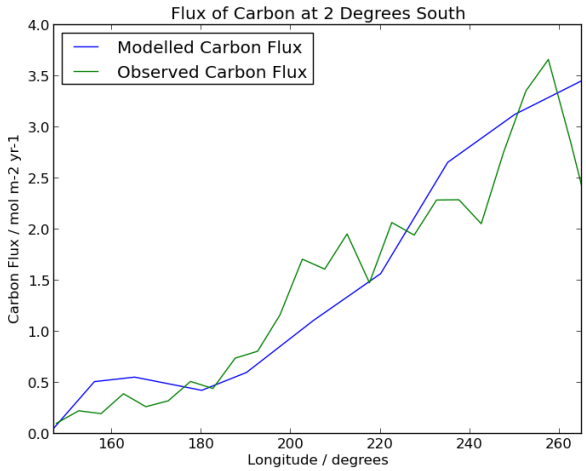

Figure 12: A comparison of the carbon flux across the basin at 2^oS shown in the data and predicted by Equation 17.

case (0^oN, 110^oW), and plot the time series of the carbon flux at that point. I have chosen this grid point because there are stronger correlations between the variables in the east of the basin than in the west, and therefore many of our assumptions will be most reliable in this region. It is also a grid point in which the data is amongst the most complete. Figure 13 shows the plot of the carbon flux as a function of time, detailing the individual components of the flux as well as the total.

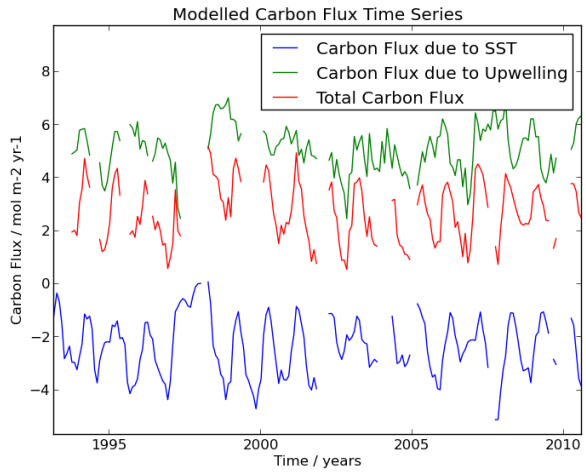

Figure 13: The carbon flux predicted by the Equation 17. The total flux is comprised of the flux due to SST, and the flux due to upwelling.

It is also possible to compare how the flux varies in different parts of the basin. Figure 14 compares the flux in the west of the basin to the flux in the east of the basin.

Finally, I show that I am able to make a prediction of the carbon flux based on a model of how the thermocline depth evolves with time. In order to do this, all of the variables have been parametrised in terms of the isotherm depth anomaly. To test the validity of this approach, Figure 15 compares a plot of the carbon flux at (0^oN, 110^oW) when calculated using the data for each of the variables, and when using the historic isotherm depth to parametrise all other variables.

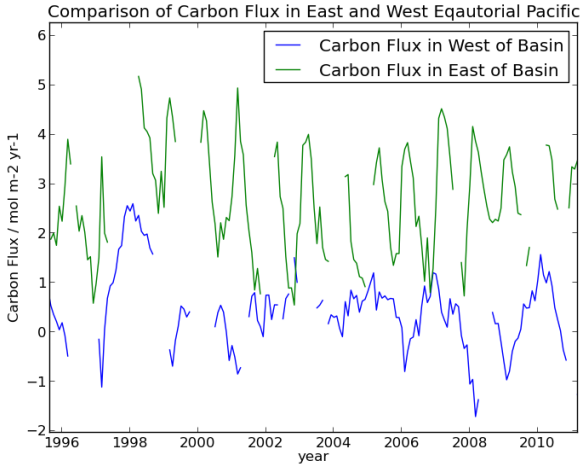

Figure 14: A comparison of the carbon flux in the east and west of the basin.

Figure 15 shows that approximation works well in this region of the basin, and so I proceed with this method. I am now able to plot the predicted carbon flux based on the model of the thermocline anomaly. I have scaled the thermocline anomaly by a factor of 80 so that the magnitude of the fluctuations are in line with the observed values. Figure 16 shows the predicted carbon flux from the dynamical model of thermocline depth.

4 Analysis

From Figure 12, we can see that the model captures the east, west pattern very well. Both the magnitude of the flux and the variation of the flux between the east and west of the ocean agree quite well with observation. This suggests that the proposed model is succeeding in capturing much of the

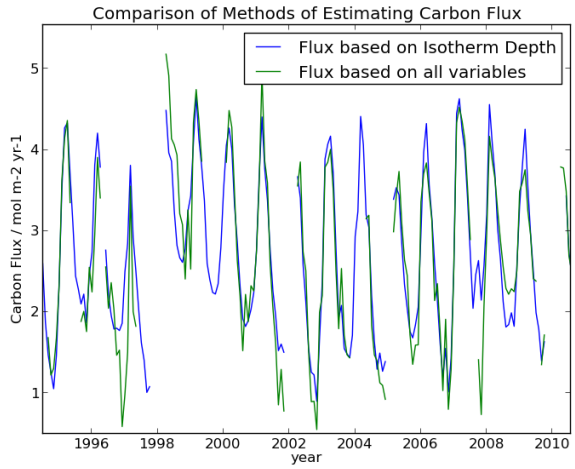

Figure 15: Time series of the carbon flux at (0^oN, 110^oW) based on (a) model with all variables from data and (b) all variables parametrised in terms of isotherm height.

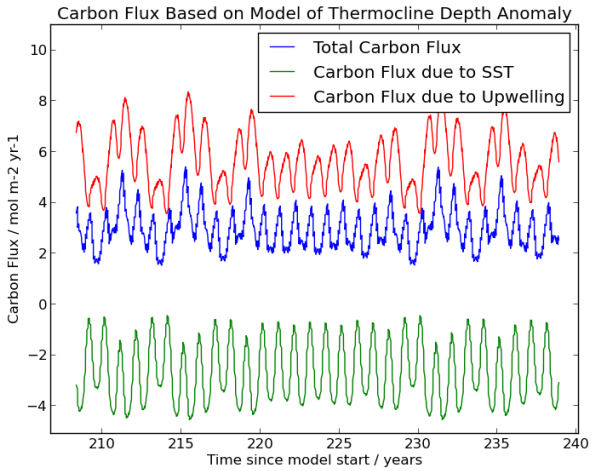

Figure 16: The flux at the grid point (0^oN, 110^oW) based on a model for the thermocline height. Plotted are lines representing the total carbon flux, the contribution due to SST,and the contribution due to the upwelling.

observed behaviour and is an encouraging result. However, the model shows significant a north-south gradient which does not agree with the data. A possible explanation of this is the ocean currents. My weakest assumption appears to be in the estimation of the upwelling velocity, where I have had to attribute all of difference in heat to upwelling. In the north-east of the basin, this is not the case, as there is the additional effect of cooler water being brought down from north of the equator (Figure 18). As a result of this, the water in this region is cooler than elsewhere before the effect of upwelling, and so I have over estimated the upwelling rate, giving the false result. This effect is less significant south of the equator as the shape of the coastline means that the currents are not symmetric and the current from the south does not reach as far north. In favour of this explanation, at about 2^oS, where Figure 18 indicates that the approximation of a purely westwards flow is best, we obtain the results which best match observation.

In Figures 13 and 16, it is shown that our total flux in the east is comprised of a strongly positive flux due to the upwelling, and a slightly negative flux due to the SST. It is also clear that in general, the anomaly in F_{SST} has the opposite sign to the anomaly in $F_{upwelling}$. This makes sense, as an increase in upwelling results in a decrease in the SST.

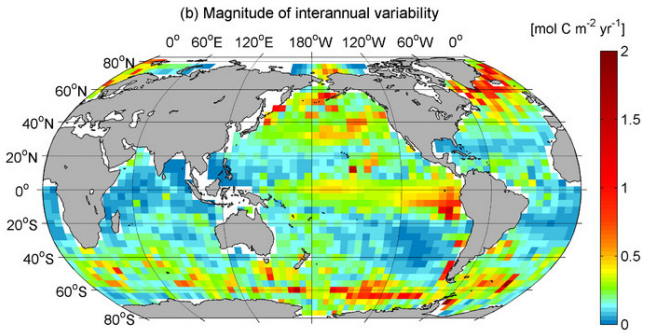

Figure 17: Inter-annual variability in the ocean-atmosphere carbon flux. [11]

When the inter-annual variability in the CO_2 flux that is obtained from our model is compared with observation, (Figure 17) the variation in the east of the basin roughly agrees with the known values, both having a value of about 1 $molm^{-2}yr^{-1}$, but our variability in the west of the basin does not drop off as it does in the observed data. This could be due to the fact that the thermocline is so much deeper in this section of the ocean. This extra depth will mean that the properties of this region will be much slower to change, and could damp the effect of ENSO more than suggested in the model.

Figure 16 shows a predicted form of the carbon flux based on the dynamical model which does not quite match the form of Figure 13 where observation data has been used to make the estimate. How-

ever, it does share many of the features. Notably, the magnitude, and the variation of the flux within each year agree well. Many of the differences can be put down to imperfections in the dynamical model for the thermocline depth anomaly, which Figure 9 shows to be only a rough approximation to reality.

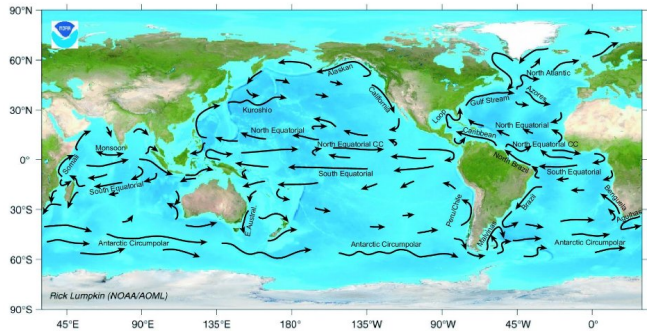

Figure 18: Map of the ocean surface currents. [12]

Another effect that I have ignored in this model is that of biology. This would have the effect of decreasing the carbon flux, as the biological life has the net effect of absorbing CO_2 and carrying it down to the deep ocean when it dies. The impact of ignoring this effect is not as obvious in our results, and this is possibly because the amount of biological life is also correlated to the upwelling rate. This is because as well as being rich in CO_2, the deep ocean is also rich in other nutrients that help to sustain life. To a first approximation, it may well be reasonable to approximate the contribution from biology as being equal to $const + K_{bio}w$. If this is the case, then the impact of biology would only be to change the values of the parameters used. The *const* term would effect the value of A, and K_{bio} would effect the value of R. This would suggest that our values for both F_{SST} and $F_{upwelling}$ are too low, but would be unlikely to effect the final result greatly. Alternately, the effect of biology could simply be very small in comparison to the other factors.

A third large assumption made in this model is in the estimation of the concentration of CO_2 in the upwelling water. I currently assume that this water has a concentration equal to water at this temperature in equilibrium with the atmosphere. However, water from the deep ocean interacts very little with the atmosphere, and so this is a poor assumption. A potential error in this calculation would result in an error in the amount of CO_2 released per unit area per unit upwelling velocity. As the model is calibrated to give the correct carbon flux, this error would be likely to only effect the size of the replenishment rate R.

A final significant challenge that affects all studies of ENSO is the relatively small amount of data. In particular, since data started to be taken, there have not yet been a sufficient number of cycles of ENSO to analyse its behaviour in much detail. In this project, the lack of data puts a limit on the accuracy with which I was able to determine the mean values of the variables, and the gradient of the relationships between them. However, given the low order of this model and the other approximations that have been made, these errors, and the errors on the individual data points are not significant.

5 Conclusion

I have shown that it is possible to reproduce many of the features seen in the air-sea carbon dioxide flux in the equatorial Pacific by considering only the flux due to the temperature of the water and the flux due to the upwelling of water from the deep ocean. In particular, the model is able to produce a profile of the carbon flux along a given latitude which agrees well with observation. This result is especially strong for latitudes just south of the equator.

I am also able to produce an estimate of the carbon flux based on a dynamical model of the thermocline depth. In order to achieve this, all of the variables are parametrised in terms of the thermocline depth. This approach is only valid in the east of the basin, where the correlations between the anomaly in thermocline depth are best correlated with the anomaly in the other variables, but we were able to show that within this limit, the approximation works well.

These successes suggest that the complex dynamics in the equatorial Pacific can be well estimated by a process based, low order model. However, there are some features in the data which are not well represented by this model. Most notably, the model shows a clear north-south gradient in the carbon flux, which disagrees with observation. A possible explanation of this is that in the north-east of the basin, a current of cold water from the north is reducing the temperature of the water. In our model we have assumed that any cooling is due to upwelling, and so by ignoring this current, we have

overestimated the amount of upwelling, and so overestimated the carbon flux.

The susceptibility of the model to ocean currents highlights that this model is likely only to be applicable in the equatorial Pacific. In addition to this, in other areas of the ocean there may be other dynamical effects which are not taken account of here. For example in many areas of the ocean, there is downwelling water, which we are unable to capture in this model.

In this project, I have shown that a low order linear model captures enough of the dynamics in the equatorial Pacific to make further investigation interesting. The most obvious cause of error appears to be that I have ignored the horizontal advective currents. It would therefore be desirable to find a method of parametrising the effects of the currents. However, it is not clear how this would be easily achieved. A factor that could be introduced given more time is the effect of biological life in the ocean. As discussed in the Analysis however, it may not affect the end result significantly. This would however allow for a more accurate estimation of the upwelling rate. If this were to become desirable, further study would be needed to obtain a better estimation of how the concentration of carbon dioxide changes with depth.

Finally, it would be interesting to examine the possibility of using this low order model in conjunction with a more complex climate model, either to help analyse the output of the climate model, or to help increase the speed of the model by making use of some of the relationships discovered here.

References

[1] The UK Met Office : El Nino, La Nina and the Southern Oscillation. Available at http://www.metoffice.gov.uk/research/climate-/seasonal-to-decadal/gpc-outlooks/el-nino-la-nina/enso- d escription (Accessed 26 April 2014)

[2] Hurricanes: Science and Society. Available at http://www.hurricanescience.org/science-/observation/ships/ships/ (Accessed 26 April 2014)

[3] Tropical Ocean Atmosphere Project. Available at http://www.pmel.noaa.gov/tao/ (Accessed 30 March 2014)

[4] J. Bjerknes, Atmospheric Teleconnerctions from the Equatorial Pacific, *Monthly Weather Review*, **97**, 163-172 (1969).

[5] The Engineering Toolbox. Available at http://www.engineeringtoolbox.com/gases-solubility-water-d_1148.html. (Accessed 30 March 2014)

[6] C.D. Jeffery, I.S. Robinson, D.K. Woolf, Tuning a Physically Based Model of the Air-Sea Gas Transfer Velocity, *Ocean Modelling*, **31**, 28-35. (2009)

[7] T. Takahashi, Surface Variation of CO_2 and Nutrients in the High Latitude Surface Oceans: A Comparative Study, *Global Biochemical Studies*, **7**, 843-878 (1993)

[8] E. Tziperman, L. Stone, M.A. Cane, H Jarosh, El Nino Chaos: Overlapping of Resonances Between the Seasonal Cycle and the Pacific Ocean Atmosphere Oscillator, *Science*, **264** 72-74 (1994)

[9] M. Munnich, M.A. Cane, S.E. Zebiak, A Study of Self-Excited Oscillations of the Tropical Ocean Atmosphere System. Part II: Nonlinear Cases. *J. Atmos. Sci.* **48**, 1238-1248 (1991)

[10] Iris and Cartopy : Open Source Python Libraries created by the Met Office. Available at http://scitools.org.uk/.

[11] PMEL Carbon Program. Available at http://www.pmel.noaa.gov/co2/file/CO2+Flux+Map, (Accessed 26 April 2014)

[12] Ocean Explorer. Available at http://oceanexplorer.noaa.gov/facts/climate.html (Accessed 26 April 2014)

[13] M.J.Suarez, P.S.Schopf, A Delayed Acion Oscillator for ENSO, *Journal of the Atmosoheric Sciences* **45** 3283-3287 (1988)

6 Appendix

In the Appendix I include the code used in the project

6.1 Library of Functions

```
'''
This is a library of useful functions that are used frequently in the project.

It contains the following functions;

1 - extract_data()
1.1 - get_data()
1.2 - make_cube()
1.3 - check_masked()
2 - get_anomaly()
'''

import iris
import numpy as np
import iris.analysis.maths as math
import scipy.stats
import iris.quickplot as qplt
import matplotlib.pyplot as plt
import scipy.interpolate
from scipy.fftpack import rfft, fftfreq

start_year = 1960

'''
--------------------------------------------------------------------------------
1. extract_data() - Assembles a cube from the data files
--------------------------------------------------------------------------------
'''

def extract_data(folder, mask, column):

    # State the grid over which the data is given.
    lat_points = (8, 5, 2, 0, -2, -5, -8)
    lon_points = (137, 147, 156, 165, 180, 190, 205, 220, 235, 250, 265)

    # We now can now loop through all of the grid points. We then merge all
    # cubes at the same latitude, for each latitude, and then finally merge all
    # resulting cubes to obtain the full dataset.

    data_cubelist_2d = iris.cube.CubeList()
    anomaly_cubelist_2d = iris.cube.CubeList()
    monthly_mean_cubelist_2d = iris.cube.CubeList()
```

```
    for latitude in lat_points:

        data_cubelist_1d = iris.cube.CubeList()
        anomaly_cubelist_1d = iris.cube.CubeList()
        monthly_mean_cubelist_1d = iris.cube.CubeList()

        for longitude in lon_points:

            data, time_points = get_data(latitude, longitude, mask, column, folder)
            anomaly, monthly_mean = get_anomaly(data, mask)

            data_cube = make_cube(data, time_points, latitude, longitude)
            anomaly_cube = make_cube(anomaly, time_points, latitude, longitude)
            monthly_mean_cube = make_cube(monthly_mean, range(12), latitude, longitude)

            data_cubelist_1d.append(data_cube)
            anomaly_cubelist_1d.append(anomaly_cube)
            monthly_mean_cubelist_1d.append(monthly_mean_cube)

        merged_data = data_cubelist_1d.merge()[0]
        merged_anomaly = anomaly_cubelist_1d.merge()[0]
        merged_monthly_mean = monthly_mean_cubelist_1d.merge()[0]

        data_cubelist_2d.append(merged_data)
        anomaly_cubelist_2d.append(merged_anomaly)
        monthly_mean_cubelist_2d.append(merged_monthly_mean)

    data_cube = data_cubelist_2d.merge()[0]
    anomaly_cube = anomaly_cubelist_2d.merge()[0]
    monthly_mean_cube = monthly_mean_cubelist_2d.merge()[0]

    return [data_cube, anomaly_cube, monthly_mean_cube]

'''
--------------------------------------------------------------------------------
1.1 - get_data() - Fetches data from an individual file.
--------------------------------------------------------------------------------
'''

def get_data(latitude, longitude, mask, column, folder):

    end_year = 2012
    tolerance = 0.1
    months = ('01', '02', '03', '04', '05', '06',
              '07', '08', '09', '10', '11', '12')

    data = []

    filename = folder + '/' + str(latitude) + '/' + str(longitude) + '.txt'
```

```
    for year in xrange(start_year, end_year):
        for month in months:
            current_file = open(filename, 'r')
            date = ' ' + str(year) + month
            monthly_data = []
            for line in current_file:
                if line.startswith(date):
                    values = line.split()
                    value = float(values[column])
                    if not check_masked(value, mask, tolerance):
                        monthly_data.append(value)
            if len(monthly_data) == 0:
                monthly_average = mask
            else:
                monthly_average = np.mean(monthly_data)
            data.append(monthly_average)

    array = np.array(data)
    data_array = np.ma.masked_values(array, mask, atol=tolerance)

    number_of_points = (end_year - start_year) * 12
    time_points = range(number_of_points)

    print 'scanned ' + filename

    return data_array, time_points

'''
--------------------------------------------------------------------------------
1.2 - make_cube() - Constructs a 1D cube from an array and a time_points
--------------------------------------------------------------------------------
'''

def make_cube(data, time_points, latitude, longitude):

    time = iris.coords.DimCoord(time_points, long_name='time')
    cube = iris.cube.Cube(data, dim_coords_and_dims=[(time, 0)])

    latitude = iris.coords.AuxCoord([latitude], long_name = 'latitude', units='degrees')
    longitude = iris.coords.AuxCoord([longitude], long_name = 'longitude', units='degrees')

    cube.add_aux_coord(latitude)
    cube.add_aux_coord(longitude)

    cube.rename('unknown')

    return cube

'''
--------------------------------------------------------------------------------
```

```
1.3 - check_masked() - Checks if the value matches the mask value.
-------------------------------------------------------------------------------
'''

def check_masked(value, mask, tolerance):
    if value > (mask - tolerance) and value < (mask + tolerance):
        return True
    else:
        return False

'''
-------------------------------------------------------------------------------
2 - get_anomaly() - converts the data cube into an anomaly cube.
-------------------------------------------------------------------------------
'''

def get_anomaly(data_array, mask):

    tolerance = 0.1
    month_totals = ([],[],[],[],[],[],[],[],[],[],[],[])
    month_averages = []
    anomaly_data = []

    for (index, data_point) in enumerate(data_array):
        for counter in xrange(12):
            if (index+counter)%12 == 0:
                if data_point is not np.ma.masked:
                    month_totals[counter].append(data_point)

    for counter in xrange(12):
        if len(month_totals[counter]) != 0:
            ave = np.mean(month_totals[counter])
            month_averages.append(ave)
        else:
            month_averages.append(mask)

    for (index, data_point) in enumerate(data_array):
        for counter in xrange(12):
            if (index+counter)%12 == 0:
                if data_point is np.ma.masked:
                    value = mask
                else:
                    value  = data_point - month_averages[counter]
                anomaly_data.append(value)

    anomaly_array = np.array(anomaly_data)
    anomaly_array = np.ma.masked_values(anomaly_array, mask, atol=tolerance)

    month_averages = np.array(month_averages)
    month_averages = np.ma.masked_values(month_averages, mask, atol=tolerance)
```

```
    return anomaly_array, month_averages

'''
--------------------------------------------------------------------------------
3 - calculate_flux_full_data() - calculates the carbon flux from the data given.
--------------------------------------------------------------------------------
'''

def calculate_flux_full_data(data, const):

    # We first unpack the data

    sst = data[0]
    heat = data[1]
    wind = data[2]
    pressure = const['pressure_const']
    f = const['f']
    A = const['A']
    R = const['R']
    a = const['a']
    heat_ref = const['heat_ref']
    rho = const['rho']
    c = const['c']

    sst.units=None
    heat.units=None
    wind.units=None
    heat_ref.units=None

    # We now use the following equation to estimate the carbon flux;
    #
    # Flux = H*k*{Aeˆ(0.0423*T) - fP} + (70/44*rho*c)*R*(heat_ref - heat)
    #
    # where H = (1000/44*P)*(22.4 - 0.07T)
    #
    # and k = 87.6*(0.31vˆ2 - 0.71v + 7.76

    H = (1000./(44.*pressure)) * (-0.07*sst + 22.4 )
    k = 87.6*(0.31*wind**2 - 0.71*wind + 7.76)
    flux_sst = H*k*(A*math.exp(a*sst) - f*pressure)

    flux_upwelling = (70. * f * R * (-1.*heat + heat_ref)) / (44. * rho * c)

    flux = flux_sst + flux_upwelling

    return [flux, flux_sst, flux_upwelling]
```

```
'''
--------------------------------------------------------------------------------
4 - calculate_flux_iso_data() - calculates the carbon flux from the data given.
--------------------------------------------------------------------------------
'''

def calculate_flux_iso_data(isotherm, correlations, const):

    # We first unpack the data

    gradient_sst = correlations['gradient_sst']
    gradient_heat = correlations['gradient_heat']
    gradient_pressure = correlations['gradient_pressure']
    gradient_wind = correlations['gradient_wind']

    means_sst = correlations['means_sst']
    means_heat = correlations['means_heat']
    means_pressure = correlations['means_pressure']
    means_wind = correlations['means_wind']

    means_sst.units = None
    means_heat.units = None
    means_pressure.units = None
    means_wind.units = None

    f = const['f']
    A = const['A']
    R = const['R']
    a = const['a']
    heat_ref = const['heat_ref']
    pressure_const = const['pressure_const']
    rho = const['rho']
    c = const['c']

    # We now estimate the value of the variables from the value of the isotherm
    # depth using the expected value and the gradient obtained from the
    # correlations.

    mask = -999.

    sst_data = []
    heat_data = []
    pressure_data = []
    wind_data = []

    heat_ref.units = None

    for time, datum in enumerate(isotherm.data):
        for counter in xrange(12):
```

```
            if (time+counter)%12 == 0:
                if datum is not np.ma.masked:
                    sst_data.append((means_sst[counter] + gradient_sst*datum).data)
                    heat_data.append((means_heat[counter] + gradient_heat*datum).data)
                    pressure_data.append((means_pressure[counter] + gradient_pressure*datum).data
                    wind_data.append((means_wind[counter] + gradient_wind*datum).data)
                else:
                    sst_data.append(mask)
                    heat_data.append(mask)
                    pressure_data.append(mask)
                    wind_data.append(mask)

    time_points = isotherm.coord('time').points
    latitude = const['selected_lat']
    longitude = const['selected_lon']

    sst_data = np.ma.masked_values(sst_data, mask, atol=0.1)
    heat_data = np.ma.masked_values(heat_data, mask, atol=0.1)
    pressure_data = np.ma.masked_values(pressure_data, mask, atol=0.1)
    wind_data = np.ma.masked_values(wind_data, mask, atol=0.1)

    sst = make_cube(sst_data, time_points, latitude, longitude)
    heat = make_cube(heat_data, time_points, latitude, longitude)
    pressure = make_cube(pressure_data, time_points, latitude, longitude)
    wind = make_cube(wind_data, time_points, latitude, longitude)

    # We now use the following equation to estimate the carbon flux;
    #
    # Flux = H*k*{Ae^(0.0423*T) - fP} + (70/44*rho*c)*R*(heat_ref - heat)
    #
    # where H = (1000/44*P)*(22.4 - 0.07T)
    #
    # and k = 87.6*(0.31v^2 - 0.71v + 7.76

    H = (1000./(44.*pressure_const)) * (-0.07*sst + 22.4 )
    k = 87.6*(0.31*wind**2 - 0.71*wind + 7.76)
    pco2_ocean = A*math.exp(a*sst)
    pco2_ocean.units = None
    flux_sst = H*k*(pco2_ocean - f*pressure_const)

    flux_upwelling = (70. * f * R * (-1.*heat + heat_ref.data)) / (44. * rho * c)

    flux = flux_sst + flux_upwelling

    return [flux, flux_sst, flux_upwelling]

'''
--------------------------------------------------------------------------------
4.1 - calculate_flux_iso_model() - calculates the carbon flux from the data given.
--------------------------------------------------------------------------------
```

```
'''

def calculate_flux_iso_model(isotherm, correlations, const):

    # We first unpack the data

    gradient_sst = correlations['gradient_sst']
    gradient_heat = correlations['gradient_heat']
    gradient_pressure = correlations['gradient_pressure']
    gradient_wind = correlations['gradient_wind']

    means_sst = correlations['means_sst']
    means_heat = correlations['means_heat']
    means_pressure = correlations['means_pressure']
    means_wind = correlations['means_wind']

    means_sst.units = None
    means_heat.units = None
    means_pressure.units = None
    means_wind.units = None

    f = const['f']
    A = const['A']
    R = const['R']
    a = const['a']
    heat_ref = const['heat_ref']
    pressure_const = const['pressure_const']
    rho = const['rho']
    c = const['c']

    # We now estimate the value of the variables from the value of the isotherm
    # depth using the expected value and the gradient obtained from the
    # correlations.

    mask = -999.

    sst_data = []
    heat_data = []
    pressure_data = []
    wind_data = []

    heat_ref.units = None

    day = 0
    month = 0
    for time, datum in enumerate(isotherm.data):
        if datum is not np.ma.masked:
            sst_data.append((means_sst[month] + gradient_sst*datum).data)
            heat_data.append((means_heat[month] + gradient_heat*datum).data)
            pressure_data.append((means_pressure[month] + gradient_pressure*datum).data)
```

```
            wind_data.append((means_wind[month] + gradient_wind*datum).data)
        else:
            sst_data.append(mask)
            heat_data.append(mask)
            pressure_data.append(mask)
            wind_data.append(mask)

        day += 1
        if day == 30:
            day = 0
            month += 1
            if month == 12:
                month = 0


time_points = isotherm.coord('time').points
latitude = const['selected_lat']
longitude = const['selected_lon']

sst_data = np.ma.masked_values(sst_data, mask, atol=0.1)
heat_data = np.ma.masked_values(heat_data, mask, atol=0.1)
pressure_data = np.ma.masked_values(pressure_data, mask, atol=0.1)
wind_data = np.ma.masked_values(wind_data, mask, atol=0.1)

sst = make_cube(sst_data, time_points, latitude, longitude)
heat = make_cube(heat_data, time_points, latitude, longitude)
pressure = make_cube(pressure_data, time_points, latitude, longitude)
wind = make_cube(wind_data, time_points, latitude, longitude)

# We now use the following equation to estimate the carbon flux;
#
# Flux = H*k*{Ae^(0.0423*T) - fP} + (70/44*rho*c)*R*(heat_ref - heat)
#
# where H = (1000/44*P)*(22.4 - 0.07T)
#
# and k = 87.6*(0.31v^2 - 0.71v + 7.76

H = (1000./(44.*pressure_const)) * (-0.07*sst + 22.4 )
k = 87.6*(0.31*wind**2 - 0.71*wind + 7.76)
pco2_ocean = A*math.exp(a*sst)
pco2_ocean.units = None
flux_sst = H*k*(pco2_ocean - f*pressure_const)

flux_upwelling = (70. * f * R * (-1.*heat + heat_ref.data)) / (44. * rho * c)

flux = flux_sst + flux_upwelling

return [flux, flux_sst, flux_upwelling]
```

```
'''
--------------------------------------------------------------------------------
5 - get_correlations - returns the correlations between the variables.
--------------------------------------------------------------------------------
'''

def get_correlations(x, y):

    # We look for a correlation of the form y = mx + c
    #
    # We assume that c = 0 as we are dealing with perturbations from the mean.

    tol = 0.1
    mask = -999.

    if x.shape != y.shape:
        print 'Fatal Error: Variables do not have the same shape!'
        exit()

    lat_points = x.coord('latitude').points
    lon_points = x.coord('longitude').points
    time_points = x.coord('time').points

    pmcc_data = np.zeros((len(lat_points), len(lon_points)))
    gradient_data = np.zeros((len(lat_points), len(lon_points)))
    standard_error_data = np.zeros((len(lat_points), len(lon_points)))

    for lat_index in xrange(len(lat_points)):
        for lon_index in xrange(len(lon_points)):
            reduced_x = x[lat_index, lon_index]
            reduced_y = y[lat_index, lon_index]
            x_data = []
            y_data = []
            for index in xrange(len(time_points)):
                if reduced_x.data[index] is not np.ma.masked:
                    if reduced_y.data[index] is not np.ma.masked:
                        x_data.append(reduced_x.data[index])
                        y_data.append(reduced_y.data[index])
            if len(x_data) == 0 or len(y_data) == 0:
                pmcc = mask
                gradient = mask
                standard_error = mask
            else:
                gradient, _, pmcc, _, standard_error = scipy.stats.linregress(x_data, y_data)

            pmcc_data[lat_index, lon_index] = pmcc
            gradient_data[lat_index, lon_index] = gradient
            standard_error_data[lat_index, lon_index] = standard_error

    pmcc_array = np.array(pmcc_data)
```

```
    pmcc_array = np.ma.masked_values(pmcc_array, mask, atol=tol)

    gradient_array = np.array(gradient_data)
    gradient_array = np.ma.masked_values(gradient_array, mask, atol=tol)

    standard_error_array = np.array(standard_error_data)
    standard_error_array = np.ma.masked_values(standard_error_array, mask, atol=tol)

    latitude = iris.coords.DimCoord(lat_points, long_name = 'latitude', units = 'degrees')
    longitude = iris.coords.DimCoord(lon_points, long_name = 'longitude', units = 'degrees')

    pmcc_cube = iris.cube.Cube(pmcc_array, long_name = 'pmcc', dim_coords_and_dims=[(latitude, 0)
(longitude, 1)])
    pmcc_cube.rename('pmcc')

    gradient_cube = iris.cube.Cube(gradient_array, long_name = 'gradient',
dim_coords_and_dims=[(latitude, 0), (longitude, 1)])
    gradient_cube.rename('gradient')

    standard_error_cube = iris.cube.Cube(standard_error_array, long_name = 'standard_error',
dim_coords_and_dims=[(latitude, 0), (longitude, 1)])
    standard_error_cube.rename('standard_error')

    return [pmcc_cube, gradient_cube, standard_error_cube]
```

6.2 Dynamical Model of ENSO

```
"""
This script is a re-write of the script written by Eli Galanti based
on the simple model detailed in the paper;

El Nino Chaos: Overlapping of Resonances Between the Seasonal Cycle and the
Pacific Ocean-Atmospheric Oscillator.
Eli Tziperman, Lewi Stone, Mark A. Cane, Hans Jarosh

Based on code written by Eli Galanti

In the following code, we will be looking to solve the following equation;

dh(t)/dt = a*A(h{t - [L/(2Ck)]}) - b*A(h{t - [L/Ck + L/2Cr]}) + c*cos(w*t)

where we have defined that;

h is the deviation in thermocline depth from seasonal values at the eastern
boundary

t is time
```

```
L is the basin width

w is the annual frequency of the seasonal forcing

Ck and Cr are the speeds of the Kelvin and Rossby waves respectivly.

A(h) relates the wind stress to SST and SST to thermocline depth.
"""

import matplotlib.pyplot as plt
import numpy as np
import iris
import iris.quickplot as qplt
import iris.plot as iplt
from scipy.fftpack import rfft, fftfreq

import masters_library as lib


# We begin by creating a 1D array h, representing h at each time step.
h = []
AmN1_array = []
AmN2_array = []
seasonal = []

# We give an intial deviation
dt = 1
start_time = int(400/dt)
end_time = int(100000/dt)
for _ in xrange(start_time):
    h.append(1.0e-4)
    AmN1_array.append(0)
    AmN2_array.append(0)
    seasonal.append(0)
for _ in xrange(end_time - start_time):
    h.append(0.0)


# We now construct a hyperbolic tangent function to smooth out the step function
kappa = 2.0
a_plus = 1.
a_minus = 1.
b_plus = 1.5
b_minus = - b_plus / 5.
h_plus = (b_plus / (kappa*a_plus)) * (a_plus - 1)
h_minus = (b_minus / (kappa*a_minus)) * (a_minus - 1)


# parameters for the delayed equation
```

```
month = 30.
w = 2.0*np.pi/(12.0*month)
a = 1./180.
b = 1./120.
c = 1./138.

L = 1.
Ck = 1./(2.3*month)
Cr = Ck/3
seasonality_phase_months = 3.0
seasonality_phase = 2.0*np.pi*seasonality_phase_months/12.0

# parameters concerning delay times
tau1 = L/(2*Ck)
N1 = int(tau1 / dt)
tau2 = L/Ck + L/(2*Cr)
N2 = int(tau2 / dt)

# solve the equation
for index in xrange(start_time, end_time):
    # calculate A[h(index - N1)]
    delayed_contribution = h[index - N1]
    if delayed_contribution > h_plus:
        AmN1 = b_plus + (b_plus/a_plus) * (np.tanh((kappa*a_plus/b_plus) * (delayed_contribution
h_plus)) - 1.0)
    elif delayed_contribution <= h_plus and delayed_contribution > h_minus:
        AmN1 = kappa * delayed_contribution
    else:
        AmN1 = b_minus + (b_minus/a_minus) * (np.tanh((kappa * a_minus/b_minus) *
(delayed_contribution - h_minus)) - 1.0)

    # calculate A[h(index - N2)]
    delayed_contribution = h[index - N2]
    if delayed_contribution > h_plus:
        AmN2 = b_plus + (b_plus/a_plus) * (np.tanh((kappa*a_plus/b_plus) * (delayed_contribution
h_plus)) -1.0)
    elif delayed_contribution <= h_plus and delayed_contribution > h_minus:
        AmN2 = kappa * delayed_contribution
    else:
        AmN2 = b_minus + (b_minus/a_minus) * (np.tanh((kappa * a_minus/b_minus) *
(delayed_contribution - h_minus)) -1.0)

    time_now = dt * index
    # we now propogate h in time.
    h[index] = h[index - 1] + dt*(a*AmN1 - b*AmN2 + c*np.cos(w*dt*index + seasonality_phase))
    AmN1_array.append((a/b) * AmN1)
    AmN2_array.append((-1)*AmN2)
    seasonal.append((c/b) * np.cos(w*dt*index + seasonality_phase))

time_points = [(index*dt)/360.0 for index in xrange(1, end_time+1)]
```

```
time = iris.coords.DimCoord(time_points, long_name='time', units='years')
cube = iris.cube.Cube(h, long_name='kappa = ' + str(kappa) + ', dt = ' + str(dt) + ', b+ = ' +
str(b_plus), units='m', dim_coords_and_dims=[(time, 0)])

iris.save(cube, 'enso_model.nc')
```