UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Faculty of Science and Technology
Department of Physics and Technology

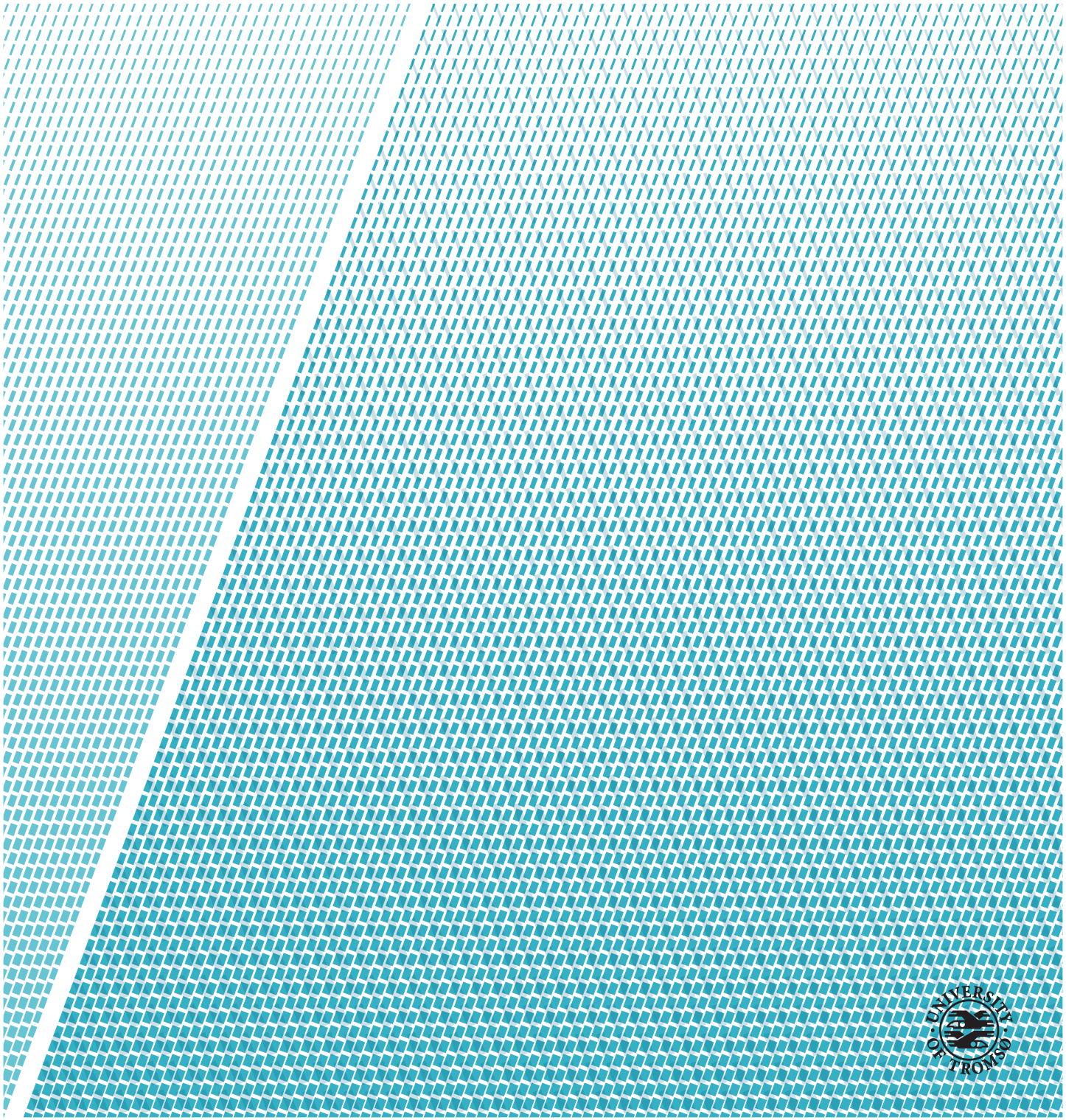
On the relationship between global mean temperature in historical runs of Earth system models and equilibrium climate sensitivity

—

Sofie Flyvholm Haug

EOM-3901 Master's Thesis in Energy, Climate and Environment, 30 SP

June 2019



“Once you stop learning, you start dying.”
–Albert Einstein

Abstract

Equilibrium climate sensitivity (ECS) measures the long-term global surface temperature response due to a doubling of CO_2 in the atmosphere. Estimates of ECS is not well constrained, 1.5-4.5°C [Pachauri et al., 2014], and there is a wide spread between different Earth system models (ESMs). Recently it has been suggested that ECS can be constrained using an observed relationship between the statistical properties of the unforced temperature fluctuations extracted from historical runs of energy system models (ESMs), and the Gregory estimates of ECS in these models [Cox et al., 2018]. In this thesis I derive general fluctuation-response relations for linear stochastic climate models and investigate the claimed relation over an ensemble of ESMs. My findings are consistent with the existence of a fluctuation-response relation, but uncertainties are large, and I find it unlikely that they can be used to constrain ECS. My conclusion is that the time period 1850-present is too short for estimation of ECS, and that we ultimately have to rely on longer reconstructed temperature time series or satellite measurements of Earth's energy imbalance.

Acknowledgements

I've never been quite sure what I wanted to be when I grow up. Maths and physics early became my favourite topics at school, so I decided to study civil engineering at UiT. I admit that I was a bit scared to move so far North alone, far away from everything I knew. I'm not much of an outdoor person and have never been much interested in hikes and skiing. When I moved all the way form Oslo to Tromsø, my mom hoped that I would be more of an outdoor person. Well, that never happened. These five years have been an amazing journey and I've learned so much, not only through my studies, but about life in general.

I would like to thank my supervisors, Hege-Beate Fredriksen and Martin Rypdal, for all their help and guidance through this last year. There would not have been a thesis without you.

A huge thanks to all my classmates from 'Energi, Klima og Miljø' class of 2014, it's been a journey! Without you as my classmates, I don't know how I would survive these hard academically years. A special thanks to Tuomas Heiskanen, who have helped us through everything we didn't understand. Thanks to family that have supported me through these five years, especially to both my Grandmothers, that always believed in me and supported me. Sadly, I lost them both to cancer during these five years, but they've been with me all the way. Thanks to all my teammates from Tromsø Atletklubb, who have done my life outside school a lot more fun.

And then, Arthur, my dear boyfriend, who is always is there for me. Backing me up, saying supportive words and even moving to Tromsø (at least for 2,5 years) just for me. I could literally not have done this without you and all the sacrifices you have made for me.

Sofie Flyvholm Haug
May 2019

Contents

List of Figures

List of Tables

1

Introduction

The concentration of greenhouse gases, especially CO_2, in the atmosphere is increasing significantly compared to pre-industrial levels. Since 1850, the temperature have been systematically recorded and we have seen an increasing trend in the global average surface temperature. Compared to pre-industrial values, we are on our way to double the concentrations of CO_2 in the atmosphere, and these changes are human made [Pachauri et al., 2014, Myhre et al., 2017]. The equilibrium climate sensitivity (ECS) is defined as the change in average global surface temperature due to a doubling of CO_2 in the atmosphere after reaching a new equilibrium state. The change in temperature is defined as

$$\Delta T = T_2 - T_1,$$

where T_1 is temperature before the forcing (the doubling of CO_2) was introduced, and T_2 is the temperature after the climate have reached a new equilibrium due to this forcing. The reason why ECS has become such an important number to estimate is that even though we know the temperature is rising, but we do not know how much the temperature will increase. What we do know is that if the temperature continues increasing there will be serious consequences for all living creatures on the planet. Large amounts of ice melting due to the rise in temperature, causing the sea level to rise. Areas that are warm and dry today, will become even warmer and drier. It is not only

affecting humans, but many species will most likely be in danger of eradication [Andrews and Jelley, 2013, Andrews et al., 2012].

There are different types of experiments that are done to obtain estimates of ECS. Some of the most common experiments that are done with climate models, are abrupt and spontaneous doubling or quadrupling of the concentration of CO_2 in the atmosphere compared to pre-industrial levels. There is no method today which allow us to measure ECS directly, but in principle it should be possible to make estimates. This can be done by either quantifying feedbacks in comprehensive climate models, palaeoclimate records, analysis of the post-industrial observed warming of the ocean and atmosphere in response to forcing, the short term climate response to forcing (volcanic eruptions etc.) or inter-annual temperature variations [Knutti et al., 2017]. The ECS can be used to assess how the climate will change in the future, which is highly interesting for us all. The values of ECS vary from 1.5°C to 4.5°C [Pachauri et al., 2014], which is a very wide interval. The results of the increasing temperature and its consequences will vary a lot within this interval. Many scientists try to narrow the interval, such that we would have a clearer image of the results and its consequences. One example is Cox et al. (2018), who used a fluctuation-response relation to try to constrain the estimates of ECS, but their method had some errors [Rypdal et al., 2018b]. IPCC estimates are found using the Gregory method, which we can not apply to historical data. Another method was presented by Rypdal et al (2018), still in the linear framework and assuming linear and stationary response. They use the convolution of the forcing with a response function. This describes the relationship between the global surface temperature and the global radiative forcing [Rypdal et al., 2018a].

In this thesis we will discuss standard methods for estimation of the ECS. These methods, such as the Gregory method, are methods where we can not use historical data to find an estimate, usually using $4\times CO_2$-data. Therefore we look at how linear-response theory can be used to study temperature response to forcing. Then we develop some theoretical fluctuation-response relations that we will apply to historical data to test if we can use the fluctuations to make an estimate of ECS. We also take a look at both CMIP5 and CMIP6 and the differences between them, and use CMIP6 to test linearity.

2

Theory and Background

Radiative forcing is the difference between incoming and outgoing radiation at the top of Earth's atmosphere, before the temperature responds to this imbalance. A forcing will either have a warming (positive forcing) or a cooling (negative forcing) effect on the Earth's surface temperature. A forcing can both be human-made or natural. The Earth seeks a balance. Such that if a forcing is introduced, the Earth will try to adapt to the change, counteract, so it can be in an equilibrium again. If we have more incoming than outgoing radiation, an imbalance, which leads to an increase in the surface temperature, the Earth will heat up and then be able to emit more radiation such that it obtains a radiation balance again. If we had more outgoing than incoming radiation, the Earth would cool down such that it emits less outgoing radiation, and once again obtains an equilibrium [Andrews and Jelley, 2013].

We will also have **feedback** due to the forcing. A feedback is something that is happening because of the forcing and will either amplify (positive feedback) or impair (negative feedback) the tempearture response to the forcing. A positive forcing, which will lead to a higher surface temperature, will lead to more melting of ice, such that more radiation will be absorbed instead of reflected, which leads to even more heating, a positive feedback. While a negative feedback would be a response, which in this case would lead to a cooling effect. The feedbacks are indirect changes that occur in a climate system as a response to the forcing [Sherwood et al., 2015]. In the latest generations of climate models, idealised experiments are commonly used to study the long-term temperature

responses to forcing.

The standard method of estimating ECS in these models is the so-called **Gregory plots**, where we plot the change in radiation at the top of atmosphere (N) against the change in global mean surface temperature (ΔT) [Gregory et al., 2004]. After the climate has reached equilibrium after the doubling of CO_2 in the atmosphere, the change in global mean surface temperature will be the equilibrium climate sensitivity, $\Delta T_{\text{eqm}} = \text{ECS}$. We can make an expression for the net heat flux N, where F is the imposed forcing (positive downwards) and H is the radiative response caused by climate change (positive upwards), both measured in Wm^{-2},

$$N = F - H.$$

This tells us the rate of change of heat in and out of the system. Such that if $F > H$ we will get $N > 0$, which means that heat is added to the system, while $F < H$ means that more heat is leaving the system than heat added, $N < 0$. Here N is the net heat flux downwards. If $F = H$, it means that $N = 0$, and the system has reached a steady-state. Most of Earth's heat capacity is in the ocean. We can therefore say that we have a steady-state when the heat storage is not changing on interannual timescales [Gregory et al., 2004]. The radiative response is often assumed to depend linearly on the temperature change

$$H = \lambda \Delta T,$$

where λ is the feedback parameter, assumed constant, that contains the strength of the net feedback to the climate system. Even though the parameter is assumed to be constant, it is a different constant for each general circulation model (GCM). It is found roughly independent of both climate state and forcing in any given GCM [Gregory et al., 2004]. For a steady-state where $N = 0$ and $F = H$ we get that, $F = \lambda \Delta T$. The ECS can then be expressed as,

$$\text{ECS} = \Delta T^{\text{eqm}}_{2\times CO_2} = \frac{F_{2\times CO_2}}{\lambda}.$$

$F_{2\times CO_2}$ is the forcing we get due to a doubling of CO_2 in the atmosphere. Using a linear framework and Gregory-plot, we can find an estimate for ECS by using

linear regression. By plotting $\Delta N(t)$ against $\Delta T(t)$, the intercept ($\Delta T = 0$) will be our forcing F, while the slope is $-\lambda$ and the ECS is the value on the x-axis (temperature change) when $N = 0$, given by $\Delta T = F/\lambda$.

CMIP is designed to improve knowledge of climate change, both the past, present and future [CMI, 2019]. Using $4\times CO_2$-data from CMIP5, we can make a Gregory-plot and find estimates for ECS, F and λ (figure 2.1). In table 2.1 we see the results using Gregory plot for all the different models. Using $4\times CO_2$-data, we divide the forcing and equilibrium temperature by 2 to make estimates due to $2\times CO_2$. These results is a reproduction of Andrews et al. (2012).

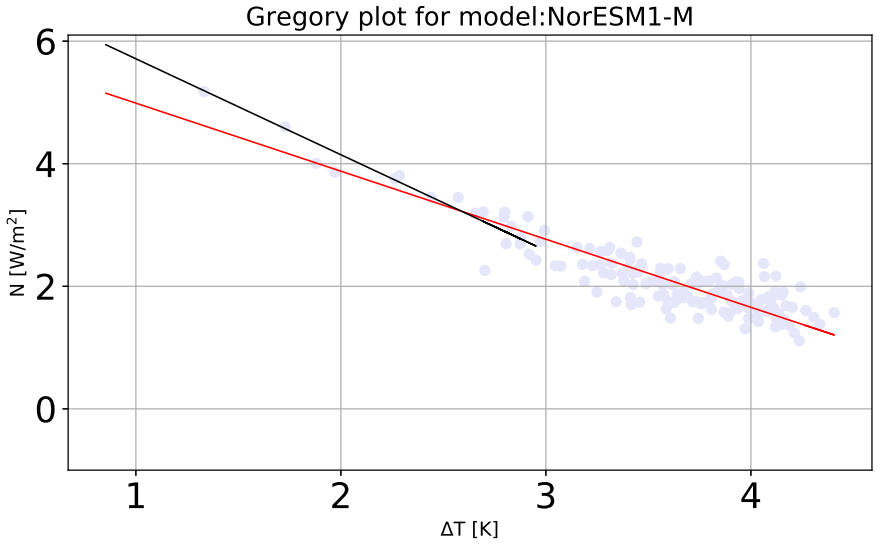


Figure 2.1: Gregory plot, plotting the change in heat flux (ΔN) against the change in temperature (ΔT). The red line is linear regression to all the data points (150 years) while the black line is linear regression to the first twenty years. Both lines are plotted together with the data that are used to make the linear regression. This plot is made using the climate model NorESM1-M.

This method assumes that the feedback parameter is constant. But we know that the climate system has slow feedbacks as well, which may result in a reduced feedback parameter, on long timescales [Rypdal et al., 2018a]. By using the same method, we can choose to only include the first 20 years of data, and see how this will affect the results. The ECS only including the first twenty years is usually a bit smaller than the ECS for the 150 years. Figure 2.1 shows the plot where we use all the data and only the first twenty years, and table 2.1 shows an overview over the different parameters you get by using both methods. The equilibrium climate sensitivity using historical runs is said to be in the lower range of the IPCC estimates [Proistosescu and Huybers, 2017]. This can be related to the fact that we are only able to see the shortest time

scales in historical data, which may correspond to using some of the early years (e.g. the first twenty years) in a Gregory-plot to make an estimate for ECS. In the following chapter we will look at alternative ways of estimating ECS, that can also be used for historical data. If we can use this to estimate ECS, it should be discussed whether we are estimating ECS or ECS^{20}.

Table 2.1: Parameters using Gregory plot, both for all years and only the first 20 years. The ones that are denoted with 20, such as ECS^{20}, indicate which parameters that are found only using the first twenty years. At the bottom we see the mean and the standard deviation of each parameter. We see a small difference between the ECS estimates that we found and the IPCC, using the same method. This is most likely because we have used the temperature at the surface, while IPCC is based on the temperature of the air 2 meters above the surface.

	IPCC ECS	ECS	ECS^{20}	forcing	$forcing^{20}$	feedback	$feedback^{20}$
ACCESS1.0	3.8	3.77	3.04	2.95	3.61	−0.78	−1.19
ACCESS1.3	n.a.	3.49	2.89	2.87	3.41	−0.82	−1.18
CanESM2	3.7	3.70	3.38	3.79	4.16	−1.03	−1.23
CNRM-CM5	3.3	3.21	3.33	3.67	3.54	−1.15	−1.07
CSIRO-Mk3.6.0	4.1	4.05	2.81	2.58	3.51	−0.64	−1.25
GFDL-CM3	4.0	3.84	3.10	2.95	3.47	−0.77	−1.12
GFDL-ESM2G	2.4	2.30	2.22	3.00	3.36	−1.31	−1.51
GFDL-ESM2M	2.4	2.33	2.28	3.27	3.43	−1.40	−1.51
GISS-E2-H	2.3	2.25	2.15	3.75	4.00	−1.67	−1.86
GISS-E2-R	2.1	2.05	1.80	3.71	4.66	−1.81	−2.58
HadGEM2-ES	4.6	4.51	3.94	2.88	3.26	−0.64	−0.83
INM-CM4	2.1	2.01	1.99	2.91	2.95	−1.45	−1.48
IPSL-CM5A-LR	4.1	4.05	3.72	3.08	3.32	−0.76	−0.89
IPSL-CM5B-LR	2.6	2.57	2.31	2.61	2.94	−1.02	−1.27
MIROC5	2.7	2.70	2.65	4.09	4.24	−1.52	−1.60
MIROC-ESM	4.7	4.67	4.08	4.23	4.70	−0.91	−1.15
MPI-ESM-LR	3.6	3.47	3.16	4.04	4.56	−1.16	−1.44
MPI-ESM-MR	n.a.	3.30	2.99	4.03	4.59	−1.22	−1.53
MPI-ESM-P	3.5	3.30	3.00	4.24	4.88	−1.28	−1.63
MRI-CGCM3	2.6	2.65	2.40	3.20	3.59	−1.21	−1.50
NorESM1-M	2.8	2.75	2.32	3.05	3.64	−1.11	−1.57
Mean	3.23	3.25	2.90	3.34	3.75	−1.10	−1.37
SD	0.826	0.845	0.672	0.492	0.522	0.331	0.389

3

Linear Theory of Global Climate Response

One way to try and make a model of the Earth is to make a so-called box model. A box model divides the surface into boxes where the box only have one characteristic, such that one box is either land or ocean. A small grid size will be more accurate than a larger grid size. We will now look at simple linear box-models and we see that we get an expression for the temperature, $T = \int_{-\infty}^{t} G(t-s)F(s)ds$, which is a convolution integral. $G(t)$ is the Green's function and $F(s)$ is forcing. The forcing can also be a sum of forcings, such that the the temperature response is equal to the sum of temperature responses from each forcing.

The simplest type of box model is a one-box model, where we only have one box in the vertical layer. Since we only have one box in the vertical layer, we lose the effect from deep water. We often use a general linear vector equation to explain the climate system, where the change in surface temperature often is expressed as

$$\mathbf{x}'(t) = -\mathbf{A} \cdot \mathbf{x}(t) + \mathbf{F}(t), \tag{3.1}$$

where $\mathbf{F}(t)$ is a vector with forcing terms and $\mathbf{A}$ is a matrix with constants due to

the fact that we assume a linear response. Using equation 3.1, we can formulate the simplest form where we only have a one-box energy balance model. The change in global temperature for a one-box model can be expressed as

$$C\frac{dx}{dt} = -\lambda x(t) + F(t), \tag{3.2}$$

where C is the heat capacity and λ is called the feedback parameter. To be able to look at this kind of energy climate model, we will first look at some relations using Fourier transform that we will be using while looking into the one-box model. By looking at first the one-box model, and then two-box and N-box model, we will try to find an relation such that they can tell us something about the equilibrium climate sensitivity.

3.1 One-Box Model

3.1.1 General Response

For a function $x(t)$, we can use the Fourier transform on the function to change from time domain to frequency domain,

$$\hat{x}(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi i f t}dt \tag{3.3}$$

[Kaper and Engler, 2013]. If we differentiate $\hat{x}$ with respect to time, and try to rewrite the integral by using integration by parts where the first term is zero. Then using the definition of a Fourier transform equation from time domain to frequency domain, we get a relation between $d\hat{x}/dt$ and $\hat{x}$ in frequency domain,

$$\frac{d\hat{x}}{dt} = 2\pi i f \cdot \hat{x}. \tag{3.4}$$

For an exponential function with a step-function, $x(t) = e^{-at}\theta(t)$, which is 0 for $t < 0$ and 1 for $t > 0$, we can get an expression for $\hat{x}(f)$ by inserting it into equation 3.3,

$$\hat{x}(f) = \int_{-\infty}^{\infty} e^{-at} \cdot \theta(t) \cdot e^{-2\pi i f t}dt$$

Because of the step function, the expression is equal to zero for all $t < 0$, such that we can then rewrite the integral limits. Integrating the remaining expression we get

$$\hat{x}(f) = \frac{1}{a + 2\pi i f}. \tag{3.5}$$

We also need to look at the definition of convolution between two functions and the definition of the convolution between the functions g and h is

$$(g * h)(t) \equiv \int_{-\infty}^{\infty} g(t - s)h(s)ds. \tag{3.6}$$

The Fourier transform of the convolution between two functions in time domain is equal to the Fourier transform of the functions multiplied in frequency domain, $\widehat{g * h}(t) = \hat{g}(f) \cdot \hat{h}(f)$ and the Fourier transform of two functions multiplied in time domain is equal to the Fourier transform of the functions and the convolution between them, $\widehat{g \cdot h}(t) = \hat{g}(f) * \hat{h}(f)$ (see appendix B for proofs) [Almeida, 1997]. Taking the Fourier transform of equation 3.2, such that $\hat{x}' = -a\hat{x} + \hat{F}$. Use equation 3.4 and substitute for $\hat{x'}$, such that $2\pi i f \hat{x} = -a\hat{x} + \hat{F}$. This gives us an expression for $\hat{x}$ in terms of the forcing, the coefficient a and the frequency,

$$\hat{x} = \frac{\hat{F}}{2\pi i f + a}$$

We can define the functions $\hat{G}(f) = 1/(a + 2\pi i f)$ and $\hat{F}(f) = \hat{F}$, such that $\hat{x} = \hat{G}(f)\hat{F}(f)$.

Comparing this to equation 3.5, if some function of time is equal to $e^{-at}\theta(t)$, then the Fourier transform of this function can be expressed as $1/(a + 2\pi i f)$. If we then want to move back again, from frequency domain to time domain, we can take the inverse Fourier transform and use the relations we know (equation B.6) such that

$$x(t) = (G * F)(t) = \int_{-\infty}^{\infty} G(t-s)F(s)ds,$$

and since $\hat{G}(f) = 1/(a + 2\pi i f)$, we know from equation 3.5 that $G(t) = e^{-at}\theta(t)$,

$$x(t) = \int_{-\infty}^{\infty} e^{-a(t-s)}\theta(t-s)F(s)ds.$$

The step function $\theta(t-s)$ is 0 for all $t > s$ and 1 otherwise, so can we redefine the limits in the integral such that

$$x(t) = \int_{-\infty}^{t} e^{-a(t-s)}F(s)ds. \tag{3.7}$$

3.1.2 Calculations with Stochastic Forcing

We will now take a closer look at the forcing function $F(s)$, and consider it as a stochastic process $F(s)ds = \sigma dB(s)$, where $dB(s)$ is the white-noise random measure with mean $\langle dB(t)\rangle = 0$ and correlation $\langle dB(t)dB(s)\rangle = 0$ for all s and t, except when $t = s$, [Rypdal et al., 2018a]. If we had normalized white noise in discrete time, this would give us 1 for the case where $t = s$, i.e.

$$\langle \Delta B(s)\Delta B(t)\rangle = \begin{cases} 0 & \text{for } t \neq s \\ 1 & \text{for } t = s. \end{cases}$$

In continuous time we can substitute this with a Dirac delta function [Hassani, 2009]:

$$\langle dB(s)dB(t)\rangle \propto \delta(t-s).$$

Now we use equation 3.7, and substitute for the forcing we just found an expression for,

$$x(t) = \int_{-\infty}^{t} e^{-a(t-s)}F(s)ds = \int_{-\infty}^{t} e^{-a(t-s)}\sigma dB(s). \tag{3.8}$$

This expression is called an Ornstein-Uhlenbeck process [Rypdal et al., 2018a]. It is a stochastic integral, and we can then use the definition of an Itô integral to write this as limits of a sum. Itô integral is written as $\int_a^b f(t)dB(t) = \lim_{\Delta t \to 0} \sum_{i=1}^{n} f(t_{i-1})\Delta B_i$. Where ΔB_i is a step of Brownian motion across the interval. An Itô integral differs from a Riemann integral, where the Itô integral has B_t as an integrator, and also that the input in the function f in the definition of an Riemann integral can be any chosen point in the interval, while the same point for an Itô integral is required to be the endpoint on left side of the interval, t_{i-1} [Sauer, 2011],

$$x(t) = \int_{-\infty}^{t} e^{-a(t-s)}\sigma dB(s) = \lim_{\Delta s \to 0} \sum_{s_i} e^{-a(t-s_i)}\sigma \left(B(s_{i+1}) - B(s_i)\right)\Delta s.$$

The increments $B(s_{i+1}) - B(s_i)$ of a Brownion motion is a white noise. We want to take a closer look at the Ornstein-Uhlenbeck processes and its characteristics. The mean of such a process denoted by brackets $\langle\ \rangle$ is always equal to zero [Gillespie, 1996],

$$\langle x(t)\rangle = \left\langle \int_{-\infty}^{t} e^{-a(t-s)}\sigma dB(s) \right\rangle = \int_{-\infty}^{t} e^{-a(t-s)} \langle \sigma dB(s)\rangle = 0.$$

The correlation can be expressed as

$$\langle x(t)x(s)\rangle = \left\langle \int_{-\infty}^{t} e^{-a(t-\gamma_1)}\sigma dB(\gamma_1) \int_{-\infty}^{s} e^{-a(s-\gamma_2)}\sigma dB(\gamma_2) \right\rangle.$$

Where the only stochastic terms are $dB(\gamma_1)$ and $dB(\gamma_2)$, and since dB is white noise, we will get zero out from this expression except when $\gamma_1 = \gamma_2$, and we can then use the Dirac delta function to express the correlation between white noise,

$$\langle x(t)x(s)\rangle = \sigma^2 \int_{-\infty}^{t} \int_{-\infty}^{s} e^{-a(t-\gamma_1)} e^{-a(s-\gamma_2)} \delta(\gamma_1 - \gamma_2) d\gamma_1 d\gamma_2.$$

Then we assume that $s > t$, such that we can rewrite the expression so we only get one integral. There are several terms that can go outside the integral, such that

$$\langle x(t)x(s)\rangle = \sigma^2 e^{-at} e^{-as} \int_{-\infty}^{t} e^{2a\gamma_1} d\gamma_1.$$

Integrating the remaining terms we get an expression for the correlation of the Ornstein-Uhlenbeck process,

$$\langle x(t)x(s)\rangle = \sigma^2 e^{-at} e^{-as} \left[\frac{1}{2a} e^{2a\gamma_1}\right]_{-\infty}^{t} = \frac{\sigma^2}{2a} e^{a(t-s)}. \tag{3.9}$$

The variance ($\tau = t - s = 0$) for an Ornstein-Uhlenbeck process can then be written as

$$\langle x(t)^2\rangle = \frac{\sigma^2}{2a}.$$

3.1.3 Fluctuation-response relation

We use the one-box energy balance model for the global temperature, equation **??**, where $a = \lambda/C$ and write the correlation for this model as

$$\langle x(t)x(t+\tau)\rangle = e^{-\lambda\tau/c} \cdot \frac{C\sigma^2}{2\lambda},$$

and the variance ($\tau = 0$) :

$$\langle x(t)^2\rangle = \frac{C\sigma^2}{2\lambda}.$$

We see that this expression is similar to the one mentioned above for the variance for an Ornstein-Uhlenbeck process, just that we have inserted the expression for $a = \lambda/C$.

If we look at a spontaneous doubling of CO_2 in the atmosphere such that the forcing term $F(t) = Q\theta(t)$, where Q is now equal to $F_{2\times CO_2}$, which is the forcing corresponding to a doubling of CO_2 in the atmosphere. We want to express temperature as a function of time, and add the forcing and want to see how the temperature changes due to a spontaneous doubling of CO_2. Using equation 3.8 and inserting our forcing term $F(t)$,

$$x(t) = \int_{-\infty}^{t} e^{-a(t-s)} Q\theta(s) ds.$$

Since we have a spontaneous doubling and this occurs at $t = 0$, the forcing term will be equal to zero for all $t < 0$, such that we can change the integration limits and solve the integral,

$$x(t) = Q \int_0^t e^{-a(t-s)} ds = Qe^{-at} \int_0^t e^{as} ds = Qe^{-at} \left[\frac{e^{as}}{a} \right]_{s=0}^{s=t}$$

$$x(t) = \frac{Q}{a}(1 - e^{at}) \tag{3.10}$$

The function for $x(t)$ expressed in equation 3.10, tells us something about how the temperature will increase if we have a spontaneous doubling of CO_2. If we plot the forcing F as a function of time t, we will have a step-function, that is 0 for $t < 0$ and Q for $t \geq 0$. The temperature as a function of time will start to increase at $t = 0$ and after a while it will stabilize, where the difference between the temperature at $t = 0$ and the temperature after it has stabilized will be what we call equilibrium climate sensitivity (ECS). The climate has reached an equilibrium when the temperature do not change any more [Gregory et al., 2004]. This can be found by taking the limit as $t \to \infty$,

$$\text{ECS} = \frac{Q}{a} \tag{3.11}$$

The integral of the covariance function for $x(t)$, $C(\tau) = \langle x(t)x(t+\tau) \rangle$ is

$$\int_0^\infty \langle x(t)x(t+\tau)\rangle d\tau = \int_0^\infty \frac{\sigma^2}{2a} e^{-a\tau} d\tau = \frac{\sigma^2}{2a^2}$$

The equilibrium climate sensitivity ECS $= Q/a$ is also related to a^2 by

$$\mathrm{ECS}^2 = \frac{Q^2}{a^2} \tag{3.12}$$

Now we want to use the expression for the auto-variance and ECS^2 to find an expression for the relation between them. We start with the expression we just found for the integration over the variance function and rewrite this so we get an expression for a^2,

$$a^2 = \frac{2}{\sigma^2} \int_0^\infty \langle x(t)x(t+\tau)\rangle d\tau = \frac{2}{\sigma^2} \int_0^\infty C(\tau) d\tau.$$

We substitute this expression for a^2 into the expression for ECS^2. Since the correlation function is a symmetric function [Kaper and Engler, 2013], we can use this to get rid of the factor 2, and instead rewrite the integration limits we get that,

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2} \int_{-\infty}^\infty C(\tau) d\tau.$$

The power spectral density (PSD), related to the Fourier transform of the covariance function by the Wiener-Kinchin theorem [Rypdal et al., 2018a]:

$$S(f) = \int_{-\infty}^\infty C(\tau) e^{-2\pi i f \tau} d\tau.$$

If we then take the limit of the power spectral density as the frequency f goes to zero, we see that the low-frequency limit of the power spectral density is proportional to the equilibrium climate sensitivity squared [Rypdal et al., 2018a],

$$\lim_{f\to 0} S(f) = \int_{-\infty}^{\infty} C(\tau)d\tau = \frac{\sigma^2}{Q^2}\mathrm{ECS}^2,$$

i.e.

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2}S(0), \tag{3.13}$$

where $S(0)$ is the power spectral density $S(f)$ evaluated when $f = 0$. This relation is shown for a model, such as the one-box energy climate model.By using a one-box model we assume that the box has a constant temperature in the vertical layer which means that it does not include any heat change with the deep ocean. We can include this by adding another box with a higher heat capacity. We want to look at a two-box energy climate model.

3.2 Two-Box Model

3.2.1 Two-Box Model Explained

Instead of only one box in the vertical layer, we add one more such that we have two vertical boxes laying on top of each other. Each box has its own characteristics, but the boxes interact with each other.

$$C_1\frac{dx_1}{dt} = -\lambda x_1(t) + k(x_2(t) - x_1(t)) + F(t) \tag{3.14}$$

$$C_2\frac{dx_2}{dt} = -k(x_2(t) - x_1(t)) \tag{3.15}$$

Equation 3.14 and 3.15 is obtained by assuming that the energy exchange between the two boxes is proportional to the temperature difference between the two boxes [Fredriksen and Rypdal, 2017]. C_1 and C_2 are the average heat capacities per square meter for the upper box and lower box respectively. x_1 and x_2 are the temperatures in the boxes, λ is the feedback parameter, $F(t)$ is the forcing term and k is the coefficient of heat transfer between the two boxes. First we need to find en expression for x_1 and x_2, as equation 3.8 that was used for the one-box model. Then we can rewrite equation 3.14 and 3.15

in matrix form,

$$\boldsymbol{C}\frac{d}{dt}\boldsymbol{x} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{F}, \tag{3.16}$$

where

$$\boldsymbol{C} = \begin{pmatrix} C_1 & 0 \\ 0 & C_2 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \boldsymbol{K} = \begin{pmatrix} -(\lambda + k) & k \\ k & -k \end{pmatrix} \quad \boldsymbol{F} = \begin{pmatrix} F(t) \\ 0 \end{pmatrix}.$$

The solution to the linear system will have the form $\boldsymbol{x} = \int_{-\infty}^{t} e^{-A(s-t)}F(s)ds$, where $e^{-A(s-t)} = \Phi(t)\Phi^{-1}(s)$, and $\Phi(t)$ is the fundamental matrix [Fredriksen and Rypdal, 2017]. The fundamental matrix for this system will be a 2×2 matrix where each column vector is $v_i e^{\lambda_i t}$, where λ_i is the eigenvalue and v_i is the corresponding eigenvector to the matrix $\boldsymbol{A} = \boldsymbol{C}^{-1}\boldsymbol{K}$. We will just call the eigenvalues of this problem λ_1 and λ_2. Using this method, called variations of parameters, we get that the eigenvectors are

$$\boldsymbol{v}_1 = \begin{pmatrix} \frac{C_1}{k}\lambda_1 + \frac{C_1}{C_2} \\ 1 \end{pmatrix},$$
$$\boldsymbol{v}_2 = \begin{pmatrix} \frac{C_1}{k}\lambda_2 + \frac{C_1}{C_2} \\ 1 \end{pmatrix}.$$

As we are going to use this expression further, we put

$$w_1 = \frac{C_1}{k}\lambda_1 + \frac{C_1}{C_2}, \qquad \boldsymbol{v}_1 = \begin{pmatrix} w_1 \\ 1 \end{pmatrix}$$

and

$$w_2 = \frac{C_1}{k}\lambda_2 + \frac{C_1}{C_2}, \qquad \boldsymbol{v}_2 = \begin{pmatrix} w_2 \\ 1 \end{pmatrix}.$$

To obtain the final expression for x_1 and x_2, we need to calculate $\Phi(t)\Phi^{-1}(s)\boldsymbol{C}^{-1}\boldsymbol{F}$. Where

$$\Phi(t)\Phi^{-1}(s) = \frac{1}{w_1 - w_2}\begin{bmatrix} w_1 e^{\lambda_1(t-s)} - w_2 e^{\lambda_2(t-s)} & w_1 w_2 \left(e^{\lambda_2(t-s)} - e^{\lambda_1(t-s)}\right) \\ e^{\lambda_1(t-s)} - e^{\lambda_2(t-s)} & w_1 e^{\lambda_2(t-s)} - w_2 e^{\lambda_1(t-s)} \end{bmatrix},$$

and

$$\boldsymbol{C}^{-1}\boldsymbol{F} = \begin{bmatrix} F(t)/C_1 \\ 0 \end{bmatrix}.$$

This gives us that

$$\boldsymbol{x} = \Phi(t)\Phi^{-1}(s)\boldsymbol{C}^{-1}\boldsymbol{F} = \frac{1}{w_1 - w_2}\begin{bmatrix} \left(w_1 e^{\lambda_1(t-s)} - w_2 e^{\lambda_2(t-s)}\right) F(t) \\ \left(e^{\lambda_1(t-s)} - e^{\lambda_2(t-s)}\right) F(t) \end{bmatrix}.$$

Then we get the expression for x_1 and x_2,

$$x_1(t) = \frac{1}{w_1 - w_2}\int_{-\infty}^{t}\left[w_1 e^{\lambda_1(t-s)} - w_2 e^{\lambda_2(t-s)}\right] F(s)ds, \tag{3.17}$$

$$x_2(t) = \frac{1}{w_1 - w_2}\int_{-\infty}^{t}\left[e^{\lambda_1(t-s)} - e^{\lambda_2(t-s)}\right] F(s)ds. \tag{3.18}$$

3.2.2 Calculations with Stochastic Forcing

We will now find an expression for the covariance, which we will relate to the ECS in the next section. We are only interested in looking at the covariance for x_1 because what we want to obtain is an expression for the change in temperature for the upper box. Using equation 3.17 and inserting for the forcing, $F(s) = \sigma dB(s)$, as for the one-box model, we can get an expression for the covariance.

$$\begin{aligned}\langle x_1(t)x_1(s)\rangle = \Bigg\langle & \frac{1}{w_1 - w_2}\int_{-\infty}^{t}\left(w_1 e^{\lambda_1(t-\gamma_1)} - w_2 e^{\lambda_2(t-\gamma_1)}\right)\sigma dB(\gamma_1) \\ & \frac{1}{w_1 + w_2}\int_{-\infty}^{s}\left(w_1 e^{\lambda_1(s-\gamma_2)} - w_2 e^{\lambda_2(s-\gamma_2)}\right)\sigma dB(\gamma_2)\Bigg\rangle\end{aligned}$$

As for the one-box model, the same yields for the two-box model, such that dB is the only stochastic processes, and that $\langle dB(\gamma_1)dB(\gamma_2)\rangle$ will be zero for all γ_1 and γ_2, except when $\gamma_1 = \gamma_2$, and we can then use the Dirac delta function instead. We also write out the multiplication, such that we get four terms,

$$\begin{aligned}\langle x_1(t)x_1(s)\rangle =& \frac{\sigma^2}{(w_1-w_2)^2}\left[\int_{-\infty}^{t}\int_{-\infty}^{s} w_1^2 e^{\lambda_1(t+s-\gamma_1-\gamma_2)}\delta(\gamma_1-\gamma_2)d\gamma_1 d\gamma_2\right.\\ &-\int_{-\infty}^{t}\int_{-\infty}^{s} w_1 w_2 e^{\lambda_1(t-\gamma_1)}e^{\lambda_2(s-\gamma_2)}\delta(\gamma_1-\gamma_2)d\gamma_1 d\gamma_2\\ &-\int_{-\infty}^{t}\int_{-\infty}^{s} w_1 w_2 e^{\lambda_1(s-\gamma_1)}e^{\lambda_2(t-\gamma_1)}\delta(\gamma_1-\gamma_2)d\gamma_1 d\gamma_2\\ &\left.+\int_{-\infty}^{t}\int_{-\infty}^{s} w_2^2 e^{\lambda_2(t+s-\gamma_1-\gamma_2)}\delta(\gamma_1-\gamma_2)d\gamma_1 d\gamma_2\right]\end{aligned}$$

We also assume that $s > t$, such that we can rewrite it to one integral. Solving the integrals,

$$\begin{aligned}&\langle x_1(t)x_1(s)\rangle =\\ &\frac{\sigma^2}{(w_1-w_2)^2}\left[w_1^2 e^{\lambda_1(t+s)}\left(-\frac{1}{2\lambda_1}\right)e^{-2\lambda_1 t} - w_1 w_2 e^{\lambda_1 t}e^{\lambda_2 s}\left(-\frac{1}{\lambda_1+\lambda_2}\right)e^{-t(\lambda_1+\lambda_2)}\right.\\ &\left.-w_1 w_2 e^{\lambda_1 s}e^{\lambda_2 t}\left(-\frac{1}{\lambda_1+\lambda_2}\right)e^{-t(\lambda_1+\lambda_2)} + w_2^2 e^{\lambda_2(t+s)}\left(-\frac{1}{2\lambda_2}\right)e^{-2\lambda_2 t}\right].\end{aligned}$$

We can substitute $\tau = s-t$, such that the correlation can be expressed as,

$$\langle x_1(t)x_1(t+\tau)\rangle = \frac{\sigma^2}{2(w_1-w_2)^2}\left[e^{\lambda_1\tau}\left(\frac{2w_1w_2}{\lambda_1+\lambda_2}-\frac{w_1^2}{\lambda_1}\right)+e^{\lambda_2\tau}\left(\frac{2w_1w_2}{\lambda_1+\lambda_2}-\frac{w_2^2}{\lambda_2}\right)\right]. \tag{3.19}$$

3.2.3 Fluctuation-response relation

We now look at a spontaneous doubling of CO_2, as we did for the one-box model, where $F(s) = Q\theta(s)$, inserted in equation 3.17. The doubling occurs

at $t = 0$, so $Q\theta(t) = 0$ for all $t < 0$, such that we can change the integral limits,

$$x_1(t) = \frac{Q}{w_1 - w_2}\left(w_1 e^{\lambda_1 t}\int_0^t e^{-\lambda_1 s}ds - w_2 e^{\lambda_2 t}\int_0^t e^{-\lambda_2 s}ds\right).$$

Solving the integral we get that,

$$x_1(t) = \frac{Q}{w_1 - w_2}\left[\frac{w_1}{\lambda_1}\left(e^{\lambda_1 t} - 1\right) - \frac{w_2}{\lambda_2}\left(e^{\lambda_2 t} - 1\right)\right]. \tag{3.20}$$

The equilibrium climate sensitivity (ECS) is found as the climate is in steady-state after a forcing has occurred. Such that we take the limit of the temperature, equation 3.20, as time goes to infinity,

$$\lim_{t\to\infty} x_1(t) = \mathrm{ECS} = \frac{Q}{w_1 - w_2}\left(\frac{w_2}{\lambda_2} - \frac{w_1}{\lambda_1}\right).$$

This can also be written as

$$\mathrm{ECS}^2 = \frac{Q^2}{(w_1 - w_2)^2}\left(\frac{w_2}{\lambda_2} - \frac{w_1}{\lambda_1}\right)^2. \tag{3.21}$$

Now we integrate over the auto-variance function for $x_1(t)$, $C(\tau) = \langle x_1(t)x_1(t+\tau)\rangle$, and use equation 3.19 to express the variance,

$$\int_0^\infty C(\tau)d\tau = \int_0^\infty \frac{\sigma^2}{2(w_1 - w_2)^2}\left[e^{\lambda_1\tau}\left(\frac{2w_1w_2}{\lambda_1 + \lambda_2} - \frac{w_1^2}{\lambda_1}\right) + e^{\lambda_2\tau}\left(\frac{2w_1w_2}{\lambda_1 + \lambda_2} - \frac{w_2^2}{\lambda_2}\right)\right]d\tau,$$

Solving the integral on the right-hand side, we get an expression for the covariance:

$$\int_0^\infty C(\tau)d\tau = \frac{\sigma^2}{2}\cdot\frac{1}{(w_1 - w_2)^2}\cdot\left(\frac{w_1}{\lambda_1} - \frac{w_2}{\lambda_2}\right)^2.$$

To make the expression a bit simpler, we can define $D^2 = \left(\frac{w_1}{\lambda_1} - \frac{w_2}{\lambda_2}\right)^2 / (w_1 - w_2)^2$, and make an expression for D^2,

$$D^2 = \frac{2}{\sigma^2} \int_0^\infty \langle x_1(t) x_1(t+\tau) \rangle d\tau. \tag{3.22}$$

We insert this in equation 3.21 for ECS^2, and because of symmetry, we can get rid of the factor 2 as we change the integral limits [Kaper and Engler, 2013],

$$\mathrm{ECS}^2 = Q^2 D^2 = \frac{Q^2}{\sigma^2} \int_{-\infty}^\infty C(\tau) d\tau.$$

As we did for the one-box model, we find an expression for the low frequency PSD by taking the limits of the power spectral density as the frequency f goes to zero,

$$\lim_{f \to 0} S(f) = \lim_{f \to 0} \int_{-\infty}^\infty C(\tau) e^{-2\pi i f \tau} d\tau = \int_{-\infty}^\infty C(\tau) d\tau.$$

We obtain a proportional relation between the power spectral density and the square of the equilibrium climate sensitivity,

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2} S(0). \tag{3.23}$$

We see that we get the same expression for ECS^2 for the two-box model as for the one-box model by equation 3.13. The two-box model only divides in to two boxes, is that enough to make a good approximation? Now we want to look at the N-box model, and see if this relation holds for that model as well.

3.3 N-Box Model

3.3.1 N-Box Model Explained

We have now looked at both one-box and two-box. We could move on like this or we can define a N-box model, such that we have N boxes on top of each other, where N is a positive integer. Each box will have their own heat capacity C_N, and there is no overlap of the boxes. Box $N-1$ is only in contact with box N and $N-2$, and can then only exchange heat with those boxes. This will give us N-equations, one for each box:

$$\begin{aligned} C_1 \frac{dx_1}{dt} &= -\lambda x_1 + k_2(x_2 - x_1) + F(t) \\ C_2 \frac{dx_2}{dt} &= -k_2(x_2 - x_1) - k_3(x_2 - x_3) \\ \vdots & \qquad \vdots \\ C_n \frac{dx_n}{dt} &= -k_n(x_n - x_{n-1}) \end{aligned}$$

We can write this on on matrix form:

$$\boldsymbol{C}\frac{d}{dt}\boldsymbol{x} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{F}$$

Where

$$\boldsymbol{C} = \begin{pmatrix} C_1 & 0 & 0 & \cdots & 0 \\ 0 & C_2 & 0 & \cdots & 0 \\ 0 & 0 & C_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & 0 & C_N \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$$

$$\boldsymbol{K} = \begin{pmatrix} -(\lambda + k_2) & k_2 & 0 & 0 & \cdots & 0 \\ k_2 & -(k_2 + k_3) & k_3 & 0 & \cdots & 0 \\ 0 & k_3 & -(k_3 + k_4) & k_4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & 0 & & k_N \end{pmatrix} \quad \boldsymbol{F} = \begin{pmatrix} F(t) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

$\boldsymbol{C}$ is a diagonal $N \times N$-matrix, where we have the heat capacities on the diagonal. $\boldsymbol{x}$ and $\boldsymbol{F}$ are both N-dimensional column vectors, where $\boldsymbol{F}$ contain only zeros, except from the first element, because the forcing will only have an

effect on the very upper box. $\boldsymbol{K}$ contains the coefficients for the heat transfer between the difference boxes. Since the boxes are located on top of each other with no overlap, the coefficients on row 2 gives us the heat transport for box 2, which only depends on the temperature in box 1 and box 3. The solution to this system is given as:

$$\boldsymbol{x}(t) = \int_{-\infty}^{t} e^{(t-s)\boldsymbol{A}} \boldsymbol{C}^{-1} \Delta \boldsymbol{F}(s) ds,$$

where $\boldsymbol{A} = \boldsymbol{C}^{-1}\boldsymbol{K}$. And as we did for the two-box model, we are only interested in the temperature in box 1, because this will tell us the surface temperature. For x_1, the response function is a sum of N exponentially decaying functions, $R(t) = \left(e^{t\boldsymbol{A}}\boldsymbol{C}^{-1}\right)_{11} = \sum_{k=1}^{N} b_k e^{-t/\tau_k}$. Where $\tau_k = -1/\lambda_k$ and λ_k are the eigenvalues for matrix $\boldsymbol{A}$, which are real and negative because of the matrix $-\boldsymbol{K}$ is symmetric and positive definite [Fredriksen and Rypdal, 2017]. We can then express $x_1(t)$:

$$x_1(t) = \int_{-\infty}^{t} R(t-s)\Delta F(s) ds, \tag{3.24}$$

where $\Delta F(s) = F(s)/C_1$.

3.3.2 Fluctuation-response relation

As we did for both the one-box and two-box model, we will now look at the auto-variance of x_1, using equation 3.24,

$$\langle x_1(t)x_1(s)\rangle = \left\langle \int_{-\infty}^{t} R(t-\gamma_1)\frac{F(\gamma_1)}{C_1} d\gamma_1 \int_{-\infty}^{s} R(s-\gamma_2)\frac{F(\gamma_2)}{C_2} d\gamma_2 \right\rangle.$$

We use that $F(s)ds = \sigma dB(S)$. $dB(\gamma_1)$ and $dB(\gamma_2)$ are uncorrelated, and their expectation value can then be expressed using the Dirac delta function, $\delta(\gamma_1 - \gamma_2)$. Assuming $s > t$, we can write this as

$$\langle x_1(t)x_1(s)\rangle = \frac{\sigma^2}{C_1^2} \int_{-\infty}^{t} R(t-\gamma_1)R(s-\gamma_1) d\gamma_1$$

$$= \frac{\sigma^2}{C_1^2} \int_{-\infty}^{t} \left(\sum_{k=1}^{N} b_k e^{-(t-\gamma_1)/\tau_k} \right) \left(\sum_{k=1}^{N} b_k e^{-(s-\gamma_1)/\tau_k} \right) d\gamma_1.$$

We can rewrite this such that we can split up the parentheses and gather the sums such that we get an expression where we can solve the integral,

$$\langle x_1(t)x_1(s)\rangle = \frac{\sigma^2}{C_1^2} \sum_{k=1}^{N} \sum_{j=1}^{N} b_k b_j e^{-t/\tau_k} e^{-s/\tau_j} \int_{-\infty}^{t} e^{\gamma_1/\tau_k} e^{\gamma_1/\tau_j} d\gamma_1.$$

$$= \frac{\sigma^2}{C_1^2} \sum_{k=1}^{N} \sum_{j=1}^{N} b_k b_j e^{-t/\tau_k} e^{-s/\tau_j} \left(\frac{1}{\frac{1}{\tau_k} + \frac{1}{t_j}} e^{t/\tau_k} e^{t/t_j} \right).$$

We can make a substitute where we set $\tau = s - t$. This gives us an expression for the correlation depending on τ,

$$C(\tau) = \langle x(t)x(t+\tau)\rangle = \frac{\sigma^2}{C_1^2} \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{b_k b_j \tau_j \tau_k}{\tau_k + \tau_j} e^{-\tau/\tau_j}. \tag{3.25}$$

Now we can integrate over the correlation function, using equation 3.25, and solve the integral on the right-hand side, such that

$$\int_0^{\infty} C(\tau) d\tau = \frac{\sigma^2}{C_1^2} \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{b_k b_j \tau_k \tau_j^2}{\tau_k + \tau_j}. \tag{3.26}$$

We want to take a look at $x_1(t)$, using equation 3.24, to find an expression for the equilibrium climate sensitivity. We can insert $\Delta F(s) = F(s)/C_1$ and $F(s) = Q\theta(s)$, where $\theta(s)$ is a step-function which is 0 for all $t < 0$ and 1 for $t \geq 0$.

$$x_1(t) = \frac{Q}{C_1} \int_{-\infty}^{t} R(t-s)\theta(s) ds = \frac{Q}{C_1} \int_0^{t} R(t-s) ds.$$

Inserting the expression for $R(t-s)$ where $R(t) = \left(e^{t\boldsymbol{A}}\boldsymbol{C}^{-1}\right)_{11} = \sum_{k=1}^{N} b_k e^{-t/\tau_k}$ and integrating the expression we get that

$$x_1(t) = \frac{Q}{C_1}\sum_{k=1}^{N} b_k e^{-t/\tau_k}\int_0^t e^{s/\tau_k}ds = x_1(t) = \frac{Q}{C_1}\sum_{k=1}^{N} b_k\tau_k\left(1-e^{-t/\tau_k}\right).$$

After a long time, we will reach a new equilibrium due to a doubling of CO_2. To make an expression for this equilibrium climate sensitivity, we let $t \longrightarrow \infty$,

$$\mathrm{ECS} = \lim_{t\to\infty} x_1(t) = \frac{Q}{C_1}\sum_{k=1}^{N} b_k\tau_k.$$

We can also make an expression for the square of the equilibrium climate sensitivity,

$$\mathrm{ECS}^2 = \frac{Q^2}{C_1^2}\left(\sum_{k=1}^{N} b_k\tau_k\right)^2 = \frac{Q^2}{C_1^2}\sum_{k=1}^{N}\sum_{j=1}^{N} b_k b_j\tau_k\tau_j. \tag{3.27}$$

Now we want to take a look at equation 3.26, and show that it is possible to rewrite this expression a bit such that it is easier to compare with equation 3.27. We leave the constants σ^2/C_1^2, and want to look at how we can rewrite the sums. We start by adding and subtract τ_k from one of the τ_j terms,

$$\sum_{k=1}^{N}\sum_{j=1}^{N}\frac{b_j b_k\tau_j^2\tau_k}{\tau_k+\tau_j} = \sum_{k=1}^{N}\sum_{j=1}^{N}\frac{b_j b_k\tau_j(\tau_j+\tau_k-\tau_k)\tau_k}{\tau_k+\tau_j}.$$

We can this expression into two fractions, such that we get two separate terms,

$$\sum_{k=1}^{N}\sum_{j=1}^{N}\frac{b_j b_k\tau_j^2\tau_k}{\tau_k+\tau_j} = \sum_{k=1}^{N}\sum_{j=1}^{N}\frac{b_j b_k\tau_j\tau_k(\tau_j+\tau_k)}{\tau_k+\tau_j} - \sum_{k=1}^{N}\sum_{j=1}^{N}\frac{b_j b_k\tau_j\tau_k^2}{\tau_k+\tau_j}.$$

We can cancel the $\tau_k + \tau_j$ from the numerator and the denominator in the first fraction on the right hand side. j and k are only indexes and it could have

been chosen the other way around. This means that the left-hand side and the second term on the right-hand side is equal just with opposite signs. We can then write this as

$$\sum_{k=1}^{N}\sum_{j=1}^{N}\frac{2b_j b_k \tau_j^2 \tau_k}{\tau_k+\tau_j} = \sum_{k=1}^{N}\sum_{j=1}^{N} b_j b_k \tau_j \tau_k.$$

We can insert this into equation 3.26, such that our expression for the integral over the covariance function looks like

$$\int_0^\infty C(\tau)d\tau = \frac{\sigma^2}{2C_1^2}\sum_{k=1}^{N}\sum_{j=1}^{N} b_k b_j \tau_k \tau_j. \tag{3.28}$$

To simplify both equation 3.28 and 3.27, we can define $B^2 = (1/C_1^2)\sum_{k=1}^{N}\sum_{j=1}^{N} b_k b_j \tau_k \tau_j$, and insert this in equation 3.28 and rewrite the expression such that it gives us an expression for B^2,

$$B^2 = \frac{2}{\sigma^2}\int_0^\infty C(\tau)d\tau.$$

If we now take a look at equation 3.27, which can be written as $\mathrm{ECS}^2 = Q^2B^2$ and insert the expression we just found for B^2. Here we can also use what we know of the symmetry of the correlation function [Kaper and Engler, 2013] , such that we can get rid of the factor 2. We get that

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2}\int_{-\infty}^{\infty} C(\tau)d\tau. \tag{3.29}$$

We want to make an expression for the low-frequency power spectral density. We take the limits of the PSD as the frequency goes towards zero,

$$\lim_{f\to 0} S(f) = \lim_{f\to 0}\int_{-\infty}^{\infty} C(\tau)e^{-2\pi i f\tau}d\tau = \int_{-\infty}^{\infty} C(\tau)d\tau = \frac{\sigma^2}{Q^2}\mathrm{ECS}^2,$$

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2}S(0) \tag{3.30}$$

where $S(0)$ is the power spectral density, $S(f)$, evaluated when $f = 0$. This relation is now showed for one-box model, two-box model and for N-box model, where we see that equation 3.13, equation 3.23 and equation 3.30 give the same relation. The relation connects the ECS to the power spectral density evaluated in the zero-frequency, where the power spectral density can be found in different ways. Later we will look at two methods, especially one where we use the fluctuations in historical temperature data.

3.4 General Case

We have shown the equilibrium climate sensitivity relation for both one-box, two-box and N-box models. Now we want to generalize it, so $G(t)$ is now any function while $F(t)$ is still a stochastic process where $F(s)ds = \sigma dB(s)$. We start with the auto-covariance function, $C(\tau) = \langle x(t)x(t+\tau)\rangle$, where

$$x(t) = \int_{\infty}^{t} G(t-s)\sigma dB(s).$$

When we insert $x(t)$ and $x(t+\tau)$ into the correlation function, both includes a variable s, but not necessarily the same s, so we will call one of them s_1 and the other s_2 such that we are able to separate them,

$$C(\tau) = \sigma^2 \int_{-\infty}^{t}\int_{-\infty}^{t+\tau} G(t-s_1)G(t+\tau-s_2)\left\langle dB(s_1)dB(s_2)\right\rangle .$$

Then we assume that $\tau > 0$, such that we get rid of the double integrals, and only get one integral. As mentioned before, the $\langle dB(s1)dB(s2)\rangle$ term will give zero for all $s_1 \neq s_2$, we can replace this with the Dirac delta function,

$$C(\tau) = \sigma^2 \int_{-\infty}^{t} G(t-s)G(t+\tau-s)ds.$$

We can do a change of variables, such that $t' = t - s$, $ds = -dt$, and we also have to change the limits of the integral as well, where it will now go from ∞ to o. But we will get a minus sign in front our expression for the correlation. We can flip the integral limits, which gives us another minus sign. The two minus signs becomes a plus sign instead. The auto-correlation will now be expressed as

$$C(\tau) = \sigma^2 \int_{0}^{\infty} G(t')G(t'+\tau)dt' = \sigma^2 \int_{0}^{\infty} G(t)G(t+\tau)dt.$$

Now we have an expression for the auto-covariance with the a generalized $G(t)$. Then we want to find an expression for the temperature $x(t)$, using equation

3.8 but with $G(t)$ instead of the exponential function. The forcing function $F(t) = Q\theta(t)$, where $\theta(t)$ is still a step function due to a spontaneous doubling of CO_2. Since we have no forcing for $t < 0$, we can change the integral limits such that it goes from zero to infinity,

$$x(t) = \int_0^t QG(t-s)ds.$$

We make a substitution where we set $u = t - s$ and $ds = -du$. Using the same trick here as before, by flipping the integral limits to get rid of the minus sign we get from the substitution,

$$x(t) = Q \int_0^t G(u)du.$$

The equilibrium climate sensitivity can be found from this expression if we let $t \longrightarrow \infty$, to find the change in temperature after the climate has stabilized due to an induced forcing because of doubling of CO_2 . We can then change the upper integration limits to achieve an expression for the equilibrium climate sensitivity,

$$\text{ECS} = Q \int_0^\infty G(t)dt.$$

This can be also be written as

$$\text{ECS}^2 = Q^2 \int_0^\infty \int_0^\infty G(t)G(s)dtds.$$

We then do a change of variables, where $\tau = s - t$ and $ds = d\tau$. Then we see that we get an expression that looks very alike the one we made for the covariance function, such that we get

$$\text{ECS}^2 = \frac{Q^2}{\sigma^2} \int_{-\infty}^\infty C(\tau).$$

We have achieved the same expression for the equilibrium climate sensitivity for a generalized $G(t)$ as for the other functions, and as we have shown for the other cases as well. If we want to take a look at the spectral density function for the low frequencies, we take the limit as the frequency goes towards zero, with gives us that the expression for $S(0)$ is equal to the integral from minus infinity to infinity over the correlation function, $C(\tau)$. So we can say that this yields for every response function $G(t)$,

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2} S(0). \tag{3.31}$$

3.5 Choice of Temperature Response Function

In this thesis, we will use the case where we have a three-box model such that the response function will be a sum of three exponentials

$$G(t) = \sum_{k=1}^{3} c_k e^{-t/\tau_k},$$

where τ_k, $k = 1, 2, 3$ are three different time scales. We have some responses that are happening on a short time scale, while others responses will occur after a long time (longer time scales). The reason that we chose to use three time scales is that we see from the fit we do to $4\times CO_2$-data that three timescales describes the response well enough. Choosing too many can lead to too much overfitting, which we will try to avoid. The surface temperature response is still defined as

$$x(t) = \int_{-\infty}^{t} G(t-s) \cdot F(s) ds, \tag{3.32}$$

where $G(t-s)$ is the response function that we have set to be the sum of the three exponentials, and the forcing term $F(s)$, is a constant equal to a doubling of CO_2 in the atmosphere and occurs at $t = 0$. Inserting this response function into equation 3.32,

$$x(t) = \int_{0}^{t} \sum_{k=1}^{3} c_k e^{-(t-s)/\tau_k} \cdot F_{2\times CO_2} ds.$$

Solving this integral and inserting the limits gives us this expression for the temperature response

$$x(t) = F_{2\times CO_2} \sum_{k=1}^{3} c_k \tau_k \left(1 - e^{-t/\tau_k}\right).$$

We want to find an estimate for the equilibrium climate sensitivity, which is the temperature after the climate has reached a new equilibrium due to the forcing that was introduced. We take the limit as time goes to infinity,

$$ECS = \lim_{t\to\infty} T(t) = \lim_{t\to\infty} F_{2\times CO_2} \sum_{k=1}^{3} c_k \tau_k \left(1 - e^{-t/\tau_k}\right)$$

$$ECS = F_{2\times CO_2} \sum_{k=1}^{3} c_k \tau_k. \tag{3.33}$$

3.6 Power Spectral Density and ECS

The previous section showed us that both in one-box model, two-box model, N-box model and a general case we got the same relation for ECS^2, equation 3.13, 3.23, 3.30 and 3.31. Such that we see that $\text{ECS}^2 \propto S(0)$. Equation 3.27 we also get that $\text{ECS} \propto \sum_k b_k \tau_k$. This relation can also be shown using N-box model, where we focus on the upper box and the spectral density of that box.

The temperature fluctuations in box number one, the one on the surface, in a energy balance model of N vertically distributed boxes is given by

$$x_1(t) = \sum_{k=1}^{N} b_k x_{1,k}(t)$$

[Fredriksen and Rypdal, 2017], where

$$x_{1,k}(t) = \int_{-\infty}^{t} e^{\lambda_k (t-s)} dF(s) \qquad \text{and} \qquad dF(s) = \sigma dB(s).$$

Using the Fourier transform on $x_{1,k}(t)$, using equation 3.5 we get that

$$x_{1,k}(f) = \frac{F(f)}{iw + w_k},$$

where

$$w = 2\pi f \qquad \text{and} \qquad w_k = -\frac{1}{\tau_k}.$$

The power spectral density of $x_1(t)$ becomes

$$S_1(f) = \lim_{T\to\infty} \frac{1}{T}\langle |x_1(f)|^2 \rangle.$$

Inserting the expression for $x_1(t)$ and $x_{1,k}$,

$$\begin{aligned} S_1(f) &= \lim_{T\to\infty} \frac{1}{T} \left\langle \left| \sum_{k=1}^{N} b_k \frac{F(f)}{iw + w_k} \right|^2 \right\rangle \\ &= \lim_{T\to\infty} \frac{1}{T} \left\langle |F(f)|^2 \right\rangle \left(\sum_{k=1}^{N} \frac{b_k}{(w_k + iw)} \cdot \sum_{j=1}^{N} \frac{b_j}{(w_j - iw)} \right). \end{aligned}$$

If we take a look at the last parenthesis and write out the denominator and multiply it with its complex conjugate,

$$\sum_{k=1}^{N} \sum_{j=1}^{N} \frac{b_k b_j \left(w_k w_j + w^2 - iw(w_j - w_k) \right)}{\left(w_k^2 + w^2 \right) \left(w_j^2 - w^2 \right)}$$

$$= \sum_{k=1}^{N} \sum_{j=1}^{N} \left(\frac{b_k b_j \left(w_k w_j + w^2 \right)}{\left(w_k^2 + w^2 \right) \left(w_j^2 - w^2 \right)} - \frac{b_k b_j iw \left(w_j - w_k \right)}{\left(w_k^2 + w^2 \right) \left(w_j^2 - w^2 \right)} \right)$$

The last term, including $(w_j - w_k)$ can be cancelled because of symmetry, such that we get

$$S_1(f) = \lim_{T\to\infty} \frac{1}{T} \left\langle |F(f)|^2 \right\rangle \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{b_k b_j \left(w_k w_j + w^2\right)}{\left(w_k^2 + w^2\right)\left(w_j^2 - w^2\right)}.$$

If we evaluate this spectrum in the lower frequencies, where $f \to 0$, we get that $w = 2\pi f = 0$. The spectrum evaluated in $f = 0$ is

$$S_1(0) = \lim_{T\to\infty} \frac{1}{T} \left\langle |F(0)|^2 \right\rangle \sum_{k=1}^{N} \sum_{j=1}^{N} \frac{b_k b_j w_k w_j}{w_k^2 w_j^2}. \tag{3.34}$$

If we insert the expression for w_k and w_j, we get that $S(0) \propto \sum_k \sum_j b_k b_j \tau_k \tau_j$. From equation 3.27 we have that $\mathrm{ESC}^2 \propto S(0) \propto \sum_k \sum_j b_k b_j \tau_k \tau_j$, which is consistent with $\mathrm{ECS} \propto \sum_k b_k \tau_k$.

4

Methods and Results

Using data from the models in Coupled Model Intercomparison Project Phase 5 (CMIP5) [Taylor et al., 2012], we will compare the response to $2\times CO_2$ using $4\times CO_2$-runs and historical runs. The $4\times CO_2$-data comes from experiments where the concentration of CO_2 in the atmosphere is quadrupled instantaneously. We will also study the power spectral density of the fluctuations in both scenarios. We will discuss if we are able to use historical data to predict the future. We can use Gregory-plots to make estimates for ECS using $4\times CO_2$-data directly, but we can not use this method for historical data. Therefore when we look at historical data we need to try to estimate the response due to historical forcing.

4.1 Adjust response to 4xCO_2

Instead of using Gregory-plot, we will now look at another method where we find a fit to the change in temperature using a sum of three exponentials as the response function (chapter 3.5). Then, the temperature response to a doubling of CO_2 can be written as

$$T(t) = F_{2\times CO_2} \sum_{k=1}^{3} c_k \tau_k \left(1 - e^{-t/\tau_k}\right) \tag{4.1}$$

where τ_k are the time scales and c_k are constants. Assuming a linear response, we expect 4×CO_2-data to be well approximated by 2× equation 4.1. Using 4×CO_2-data from climate models (CMIP5), we want to make a fit using the response function in equation 4.1. We assume that $\tau = (0.7, 9, 354)$ is constant, such that we only need to estimate the product $2 \cdot F_{2\times CO_2} \cdot c_k \cdot \tau_k$ (equation 4.1). The model is linearly dependent on these parameters, such that they can be estimated using linear regression. To avoid negative estimates we use non-negative least squares (figure 4.1). The choice of time scale and how the results change due to the choice will be discussed later in this chapter, but for now, we choose to use $\tau = (0.7, 9, 354)$. This is the mean of the time scales estimated for different general circulation models (GCMs) [Proistosescu and Huybers, 2017].

Using CMIP5-data for both control run and 4×CO_2 run, we make a plot of the change in surface temperature due to a quadrupling of CO_2 in the atmosphere. Using these fixed time scales, we make a fit to the plot of temperature change, which is shown in figure 4.1. With three different exponential responses, we also allow to have up to three different feedback parameters. However, we estimate only one value of the ECS. To investigate the time scale dependence of the feedback parameter, we would need additional radiation data, an in Proistosescu and Huybers(2017).

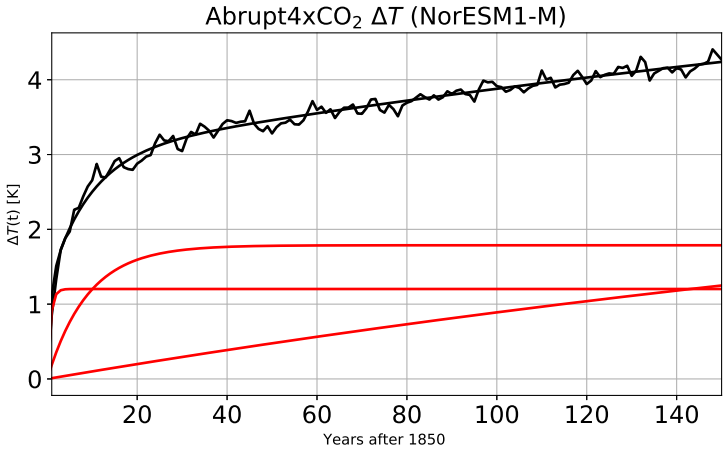


Figure 4.1: Adjusted response to $4\times CO_2$ data using climate model NorESM1-M. The red lines are the fits for each time scale, and the smooth black line is the combination of the three of them and a fit to the $4\times CO_2$ data. The time scales used are $\tau = (0.7, 9, 354)$.

4.2 The response on longer time scales

Earlier we looked at an expression for the equilibrium climate sensitivity,

$$\text{ECS} = F_{2\times CO_2} \sum_{k=0}^{N} c_k \tau_k,$$

where τ_k are time scales and c_k are the constants. We will in this thesis use $N = 3$, so that we have three time scales. But when looking at historical runs we might not able to see the response from longer time scales. We will therefore check how well we can predict the response with three time scales using only two time scales. To look at the relation between them we use $4\times CO_2$-data from CMIP5. The only uncertainty in using the two first time scales, is that we are missing information about the response on longer time scales. Figure 4.2 shows the estimates of ECS using the first two time scales and using all three of them. Gives a scatter plot, and using a linear fit, we get a red line that is a fit to the estimates. The correlation is good enough such that it can be used to predict ECS when we miss information about the longest response. The equation for the red line is found to be

$$\text{ECS}_2 = 0.02782 \cdot \text{ECS}_3 + 0.8601.$$

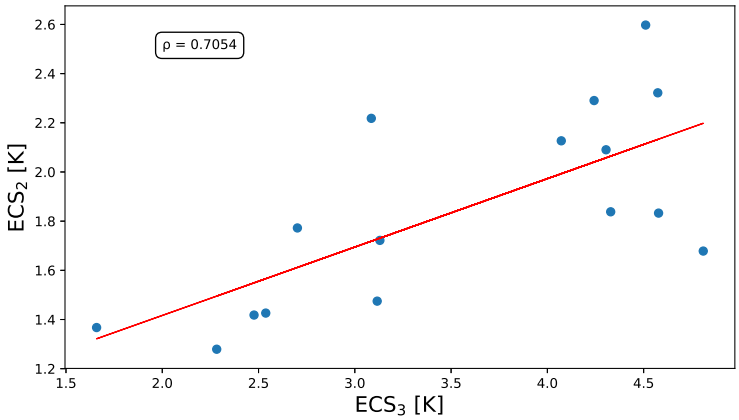
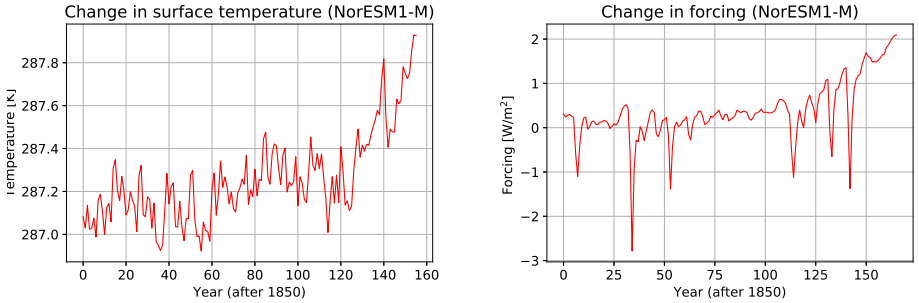


Figure 4.2: Using the expression for ECS containing the sums, ECS_3 is the sum of all the three time scales and ECS_2 is the sum only containing the first two time scales. Here using $\tau = (0.7, 9, 354)$. We will talk about the choice of time scale later. The estimates are made for 16 CMIP5 models.

4.3 Adjust response to historical data

Using historical data we want to estimate the temperature response to historical forcing. Figure 4.3a shows the historical run temperature for the NorESM1-M model. We also make a plot of the historical forcing, using the forcing that was computed by Rypdal et al (2018)(figure 4.3b). This forcing was computed by adjusting the low-frequency part of the Hansen forcing to the low-frequency part of Forster forcing [Forster et al., 2013] for each model. As mentioned, when we want to look at historical data we need to consider the response due to historical forcing (therefore using the Hansen forcing). Four of the models (GFDL-CM3, GFDL-ESM2G, GFDL-ESM2M and HadGEM2-ES) didn't have any record of the forcing for the first ten years. Looking at the plot of the change in surface temperature, figure 4.3a, we see that the temperature in overall is, as we know, increasing. We see that we have some valleys in the temperature plot, years where the surface temperature have been a bit lower, but the trend is increasing. The forcing plot (figure 4.3b) is showing the change in forcing, where we see that the forcing overall is increasing. There are some valleys in the forcing plot as well. These valleys in the forcing are mainly due to volcanic eruptions and have a net cooling effect on the surface temperature [Gregory et al., 2016].

To compute the temperature responses (T_k) to the forcing ($F(t)$) we use



(a) Historical surface temperature **(b)** Modified Hansen forcing

Figure 4.3: Historical average global surface temperature and modified Hansen forcing (historical), both for climate model NorESM1-M.

$$T_k = c_k e^{-t/\tau_k} * F(t),$$

where * denotes convolution and $k = 1, 2, 3$. $F(t)$ is the historical forcing, and $G(t)$ is the response function. We get a total response

$$T = \int_{-\infty}^{t} G(t-s)F(s)ds,$$

where $G(t) = \sum_{k=1}^{3} c_k e^{-t/\tau_k}$ (from chapter 3.5). Using historical data there are two ways we can calculate the temperature response. We can use the parameters we found using $4\times CO_2$ (divided by the forcing from $4\times CO_2$ and τ_k such that we only get out the constants c_k) to estimate the historical temperature response to the modified Hansen forcing. The other method is to use least squares to make a fit to the historical temperature (figure 4.3a) such that we get new estimates for the constants (figure 4.5). Using the last method we find the temperature responses to the modified Hansen forcing.

We use linear regression because we assume that the total temperature response in linearly depending on c_k (figure 4.4). Also in this case we use non-negative least squares to avoid negative constants c_k.

Using the historical runs from CMIP5, and comparing to the temperature responses we found using the fixed time scales and the modified forcing, we see that the linear response describes the main structure of the modelled temperature (figure 4.5). But there are some fluctuation that the linear response

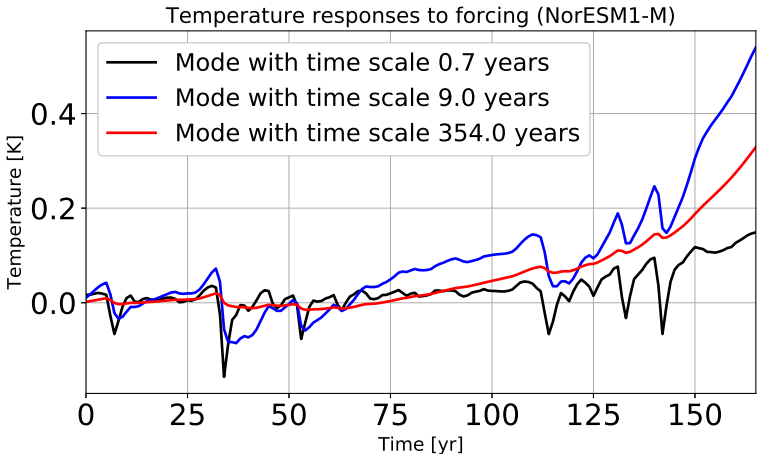


Figure 4.4: Temperature response to modified Hansen forcing. The different color represent responses to different time scales. Using climate model NorESM1-M.

does not describe. The difference between the the linear response to modified forcing and modelled temperature (historical temperature), $T - \Delta T$, will be further analysed in chapter 4.5.

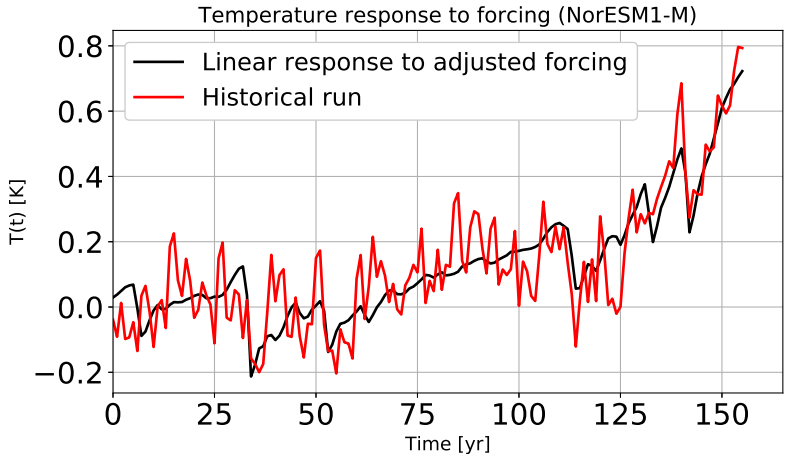


Figure 4.5: Temperature responses to forcing, where we both see the linear response to modified forcing and the modelled temperature. Done by using non-negative least squares to find new constants c_k and not using the ones we found from $4\times CO_2$-data.

4.4 Comparing models

The parameters we found from the fit to historical data together with the modified Hansen forcing is used to predict the temperature response due to a doubling of CO_2. This is then plotted together with the temperature response using $4\times CO_2$-data from CMIP5. Assuming linearity, such that the fit to $4\times CO_2$-data will be a fit to $2\times CO_2$ if we divide it by 2, we are able to compare the two methods.

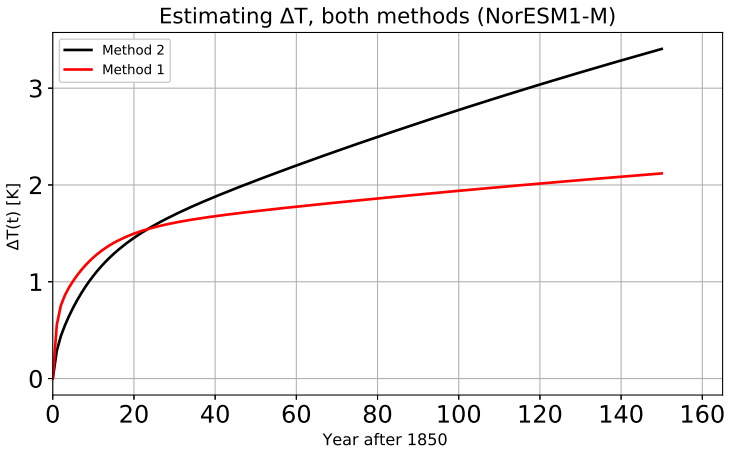
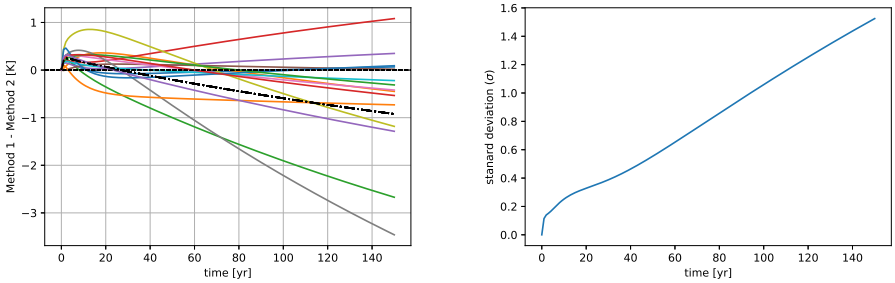


Figure 4.6: Temperature change to a doubling of CO_2 by two different methods. Method 1 is fitting three exponentials to $4\times CO_2$-data (and divide it by 2 for $2\times CO_2$-data). Method 2 is fitting linear response to modified forcing and finding the temperature response to $2\times CO_2$.

For each of the 16 models we use, we compute the difference between the temperature responses estimated by the two methods. The results are shown in figure 4.7a, and the standard deviation of these curves for each time step are shown in figure 4.7b. As we see from figure 4.6, method 2 (from historical temperatures) is giving us a much larger increase in temperature than method 1 (from $4\times CO_2$-data). This is not always the case for all the models, as we see in figure 4.7a, some of the models are above the $x = 0$ line, which means that method 1 gives a higher increase in surface temperature change.

The average for all models, given by the dotted line in figure 4.7a, tells us that using historical temperature data will in most models give an overestimation of the $2\times CO_2$-data (given linearity). But is the overestimating systematic? Figure 4.7a shows that there is a huge spread between the models. If the difference for all models was the same, such that all the curves in figure 4.7a would look the same and would lay on top of each other, we would be able to correct

for the bias. But this is not the case. By looking at the plot we see that after 150 years the spread is around 5 degrees K. The standard deviation in figure 4.7b is increasing with time and will give us a huge error in estimates of the equilibrium climate sensitivity.



(a) The coloured lines show the difference between the two methods, and the black dashed line is the mean value of all models at each time step.

(b) The standard deviation of the curves in (a) for each time step.

Figure 4.7: The difference between the two methods and the standard deviation of the curves for each time step.

4.5 Relation between power spectral density and the equilibrium climate sensitivity

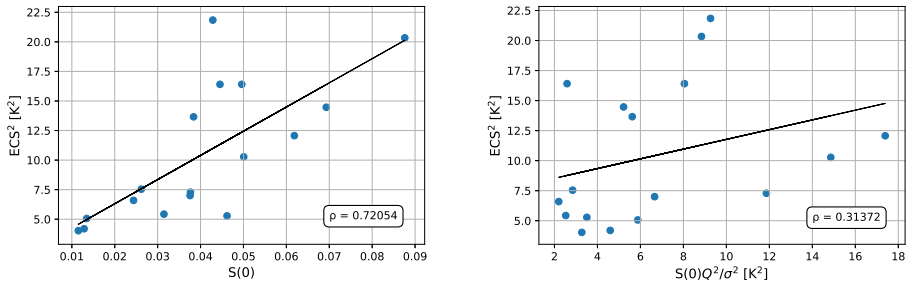
The relation we found between ECS^2 and the power spectral density, using both one-box (equation 3.13), two-box (equation 3.23), N-box (equation 3.30) and the general case (equation 3.31), is

$$\mathrm{ECS}^2 = \frac{Q^2}{\sigma^2} S(0).$$

We will try to find estimates for all the parameters such that we can make an estimate for ECS. Because of the uncertainties in the estimates, we might not be able to detect the relation even though it exists, such that we will also look at the correlation. When we look at historical runs, there is no direct way to make an estimate of ECS, but we can find estimates for these other parameters. First we will do this for the control run of temperature to try to compute a so-called unforced variability. Then we will look at $4\times CO_2$ and historical data. The reason we do this for the control run first is because we get an extra uncertainty using $4\times CO_2$ and historical data because we have to estimate the unforced variability.

The estimate of $S(0)$ is found in two ways. The first is to average over $S(f)$-values for the lower frequencies. The second method is to fit a function to the estimate of the spectrum on the form $S(f) = \sum_{k=1}^{N} \frac{c_k}{w_k^2 + w^2}$. We will talk more about the second method when we use this. Using data from the CMIP5 archive, we can find an estimate for Q and ECS directly from Gregory-plot. As mentioned, we assume a linear framework and the change in average global surface temperature is defined as $N = -\lambda T + F$, where λ is the feedback parameter found by linear regression using Gregory-plot, T is the temperature from control run and N is the net radiation. Rewriting it such that we get an expression for the forcing, $F = N + \lambda T$, we can find the variance (σ^2) by taking the variance of F.

To make an estimate for $S(0)$, we will use method 1, where we estimate the power spectral density of the control run (T) and use that $S(0) \approx \exp(\text{mean}(\log(S(1:k))))$, where k is an integer low enough such that it represent the lower frequencies. However, it must also be high enough to give an estimate with low uncertainty. The power spectral density is estimated using the windowed periodogram with a Hann-window . This is used for all power spectral densities in the thesis. We test different values for k, and plot the values for ECS^2 against the estimates of $S(0)$ for all models (figure 4.8a) and the same for ECS^2 against $S(0)Q^2/\sigma^2$ (figure 4.8b). To get a decent correlation we had to include the first 25 points, such that $k = 25$. Using the first 5 points only gave a correlation of 0.28 between ECS^2 and $S(0)$ and 0.07 between ECS^2 and $S(0)Q^2/\sigma^2$. We also see that ECS^2 and $S(0)$ has the highest correlation (for any k), and if we check the correlation between ECS^2 and $S(0)Q^2$ or $S(0)/\sigma^2$, the correlation decreases. There are uncertainties in all parameters, and the product of all parameters appears to have the greatest uncertainty. So in future analysis we will only study the correlation between ECS^2 and $\hat{S}(0)$.



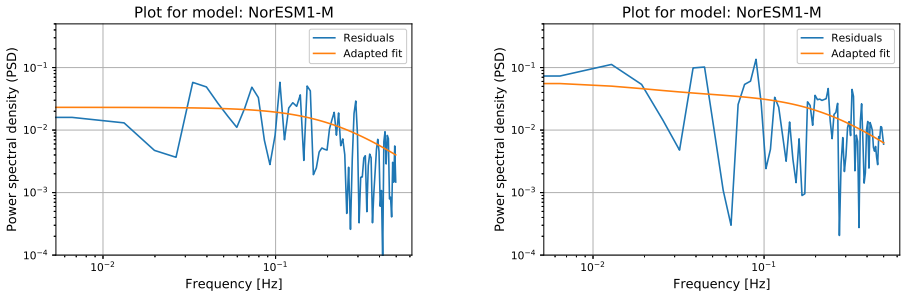
(a) **(b)**

Figure 4.8: For $k = 25$ and using Pearson correlation, we find the correlation between ECS^2 and $S(0)$ and between ECS^2 and $Q^2S(0)/\sigma^2$, where the correlation are shown in the plots.

Now we study the residuals from 4×CO_2 (figure 4.1) and historical data (figure 4.5). The residuals are computed by taking the difference between the data and the fitted curve. Using these residuals we can take the power spectral density and make a plot of the PSD of the residuals together with a fitted function $S(f)$, where

$$S(f) = \sum_{k=1}^{2} \frac{c_k^2}{w_k^2 + w^2}.$$

c_k are two parameters that are found by fitting $S(f)$ to the PSD of the residuals. $w_k = 1/\tau_k$, where τ_k are the first two time scales and $w = 2\pi f$. By taking out the first element of $S(f)$ we get an estimate for $S(0)$ and we can then compare to the Gregory-ECS values. Here we used $N = 2$, so that the sum only includes two terms. We do have three different timescales, but here we only use the two first because we only have data for 150 years while the last time scale operates at even longer time scales such that it will not have any impact on these 150 years.



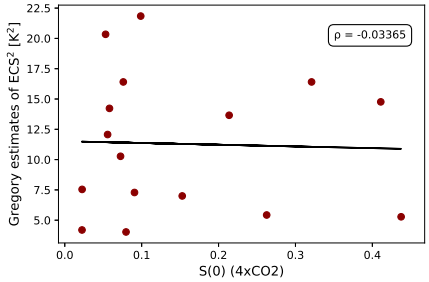
(a) PSD for the residuals between the data and the fit for the adjusted response to 4×CO_2 (figure 4.1).

(b) PSD for the residuals between the data and the fit for the forcing response using historical data (figure 4.5).

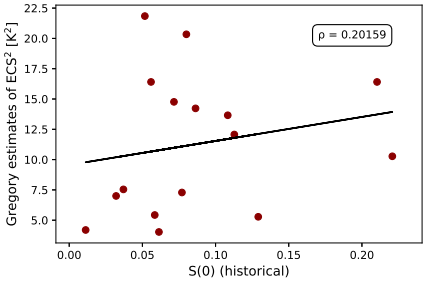
Figure 4.9: Plot of the power spectral density for the residuals and a fit to this using logarithmic axis and $\tau = (0.7, 9)$. This is done for model NorESM1-M.

Using this method, method 2, we use information about the PSD for all frequencies to make an estimate of $S(0)$. It is hard to tell whether this method is better than method 1 or not. Figure 4.11 shows us the result from using method 1 and we can then compare it to figure 4.10.

For the historical data, we mentioned another method to estimate residuals, using the constants c_k found from 4×CO_2-data instead of estimating new constants. We also took a look at what results that would give us. The residuals



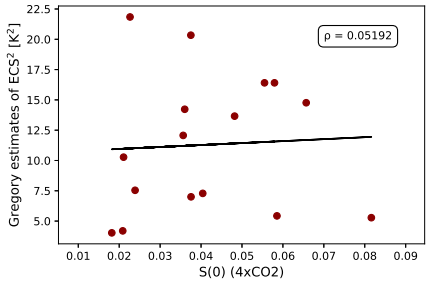
(a) S(0) from 4×CO_2-data and ECS^2 (from Gregory)



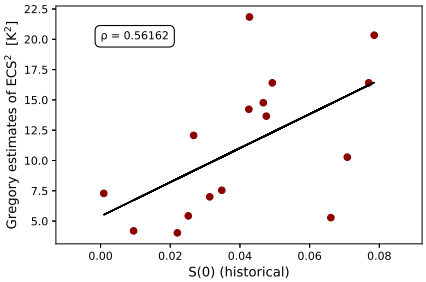
(b) $S(0)$ using historical data and ECS^2 (from Gregory)

Figure 4.10: Using Pearson correlation we find that the correlation between ECS^2 and S(0) from 4×CO_2-data and the correlation between ECS^2 and S(0) using historical data. This is done using method 2, where we fit a function to the plot of the PDS.

using this method are slightly different, such that also the estimates of the PSD of the residuals were a bit different with even larger peaks and valleys, while the fit to the PSD (yellow curve in figure 4.9b) looked almost the same. Making estimates for $S(0)$ gave approximately the same plot using method 2, where we used the fitted curve, with the same correlation at 0.20 (figure 4.10b). Using method 1, where we took the mean of the 25 first points of the PSD, we got a plot that was a bit different from figure 4.11b with a correlation of 0.46. So these two methods of estimating the constants c_k gave a slightly different result, but which method is better than the other is hard to determine.



(a) S(0) from 4×CO_2-data and ECS^2 (from Gregory)

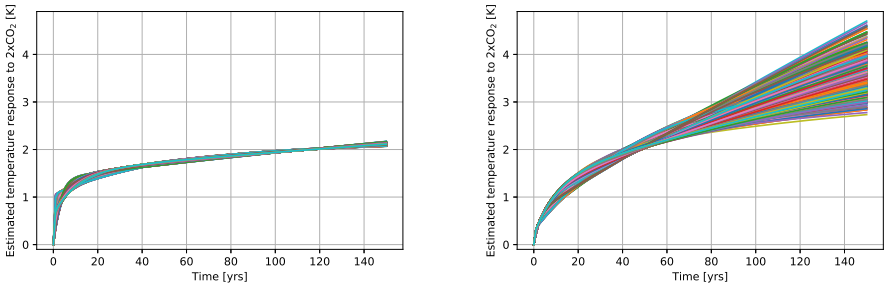


(b) $S(0)$ using historical data and ECS^2 (from Gregory)

Figure 4.11: Using Pearson correlation we find that the correlation between ECS^2 and S(0) from 4×CO_2-data and the correlation between ECS^2 and S(0) using historical data. This is done using method 1, where we take the average of the 25 first data points of the PSD.

4.6 Time scales

Now we will take a look at the sensitivity to the choice of time scales. We use data from Proistosescu and Huybers(2017) extra material, where they have given three different time scales for each model. From their time scales we make 1000 random samples of $\tau = (\tau_1, \tau_2, \tau_3)$, and repeat our analysis. We will just look at the results from one model, while the results from all the models are included in appendix A.



(a) 4×CO_2-data from CMIP5 for different time scales

(b) Historical data from CMIP5 for different time scales

Figure 4.12: The same plot as in figure 4.6 (response to 2×CO_2), the fitted lines for both methods but in two different plots here. Using 1000 different iterations, where each iteration has a different time scale.

Figure 4.12a shows the response to 2×CO_2 using 4×CO_2-data. We see that the results do not vary much due to 1000 different choices of time scales. Figure 4.12b shows the response to 2×CO_2 using historical data. Here we see that the responses are sensitive to the choice of time scales, especially for the last 100 years of the total of 150 years. For each iteration we also find the difference between the two methods. Figure 4.13 shows the difference between the the two methods for all the iterations. Also here, we see that we get a huge spread since our method is sensitive to the time scales and the other method is not.

As we just showed, the historical data is much more sensitive to the choice of time scales. For each of the 1000 samples, we also construct a residual and study how sensitive estimates of $S(0)$ are to the choice of time scales. So for each of the 16 models that are included, we make a 1000 different $S(0)$-values using method 2 from the difference between the fitted temperature response and the historical temperature data. From these 1000 values for $S(0)$, we can find the mean and standard deviation, and make a plot between ECS^2 and $S(0)$ with error bars for the $S(0)$-values.

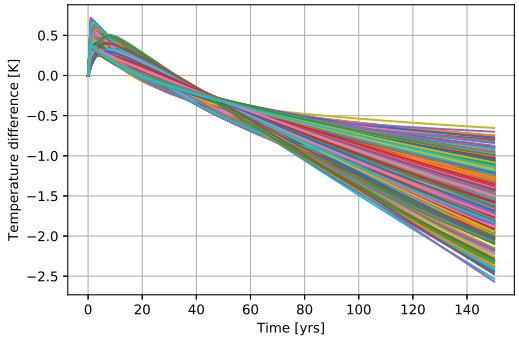


Figure 4.13: The difference between the two methods (figure 4.12a and figure 4.12b) for each of the 1000 iterations, where each iteration has a different choice of time scale.

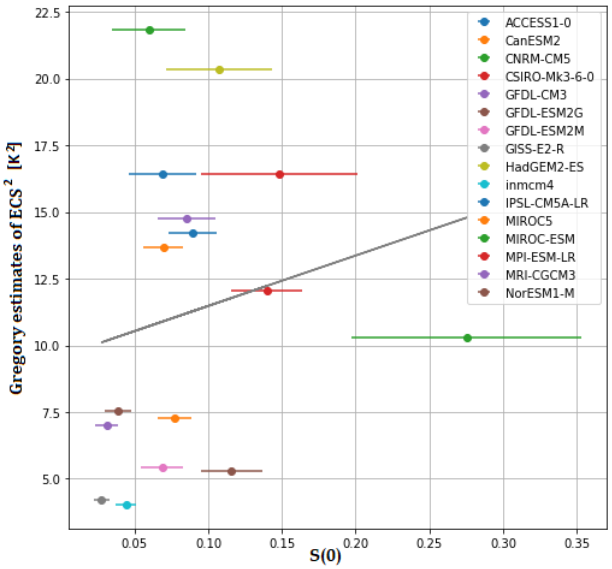


Figure 4.14: Relation between ECS^2 and $S(0)$ (using method 2 to make an estimate), with error bar and a fitted line to the mean values represented by the dots. The correlation between ECS^2 and S(0) is approximately 0.20.

5

Results for CMIP6

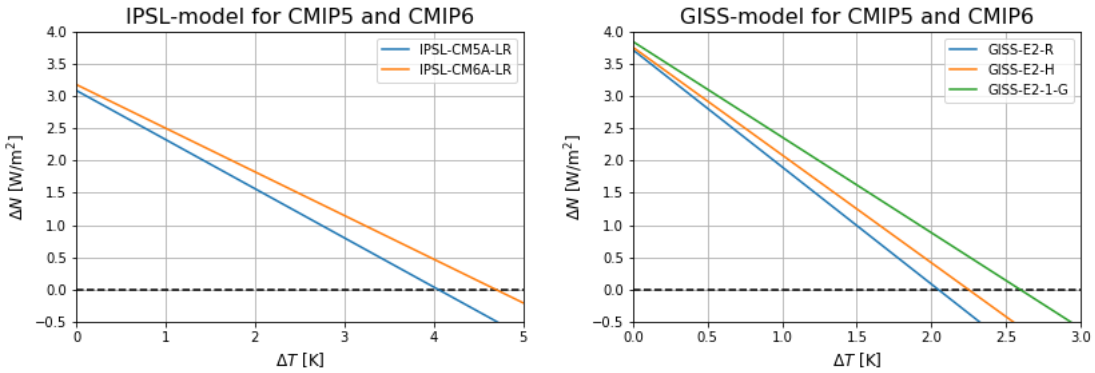
The first data from Coupled Model Intercomparison Project Phase 6 (CMIP6) are now available. We study results from four different models and two of them include other experiments than $4\times CO_2$, like $2\times CO_2$ and $0.5\times CO_2$. Using data from CMIP6 we want to check if Gregory plots for CMIP6 show different results than CMIP5. In addition, we use these data to test the linearity hypothesis.

Using $4\times CO_2$-data we make a Gregory plot and compare some of the models from CMIP5 and the models we got from CMIP6. REsults are shown in table 5.1. The Gregory plots are only for $4\times CO_2$ data. The IPSL-model has one from CMIP5 and one from CMIP6, the CNRM-model has one from CMIP5 and one similar to CMIP6, but also an Earth system model from CMIP6. The GISS-model has one new model from CMIP6 and two models from CMIP5. The model GISS-E2-1-G (from CMIP6) is a Atmosphere General Circulation Model (AGCM), while the other models are Atmosphere and Ocean General Circulation Model (AOGCM).

We can also look at the differences between the models from CMIP5 and CMIP6 by including both in a Gregory-plot. This is done for the IPSL-models and the GISS-models, but we see from table 5.1 that all the models from CMIP6 have a higher estimate for ECS than the models from CMIP5. The forcing is not necessarily lower or higher in the CMIP6 models, but the feedback parameter is lower in the CMIP6 models compared to the CMIP5 models.

Table 5.1: Comparing models from CMIP5 and CMIP6 using Gregory-plot.

	ECS	$F_{2\times CO_2}$	- Feedback parameter
IPSL-CM5A-LR (CMIP5)	4.05	3.08	−0.76
IPSL-CM6A-LR (CMIP6)	4.69	3.18	−0.68
CNRM-CM5 (CMIP5)	3.70	3.79	−1.03
CNRM-CM6-1 (CMIP6)	4.79	3.70	−0.77
CNRM-ESM2-1 (CMIP6)	4.74	2.96	−0.63
GISS-E2-H (CMIP5)	2.25	3.75	−1.67
GISS-E2-R (CMIP5)	2.05	3.71	−1.81
GISS-E2-1-G (CMIP6)	2.60	3.84	−1.48



(a) Gregory plot for the IPSL-models **(b)** Gregory plot for the GISS-models

Figure 5.1: Using $4\times CO_2$-data for the models and comparing the models from CMIP5 and CMIP6 in Gregory-plot. Where ΔT and ΔN is divided by 2 such that we get the result for $2\times CO_2$.

Since two of the models (IPSL-CM6A-LR and GISS-E2-1-G) have different runs, not only $4\times CO_2$ as in CMIP5, we can use the Gregory method for all the different scenarios and see if they support the assumption of linearity. When we assume linearity, we say that the temperature change for $4\times CO_2$ is the double of the temperature change for $2\times CO_2$, and the negative temperature change for $0.5\times CO_2$ is the same as for $2\times CO_2$. This is based on linearity in the forcing as well, where we assume that $F_{4\times CO_2} = 2 \cdot F_{2\times CO_2}$, which comes from the assumption

$$F_{CO_2} = 5.35 \log\left(\frac{C}{C_0}\right),$$

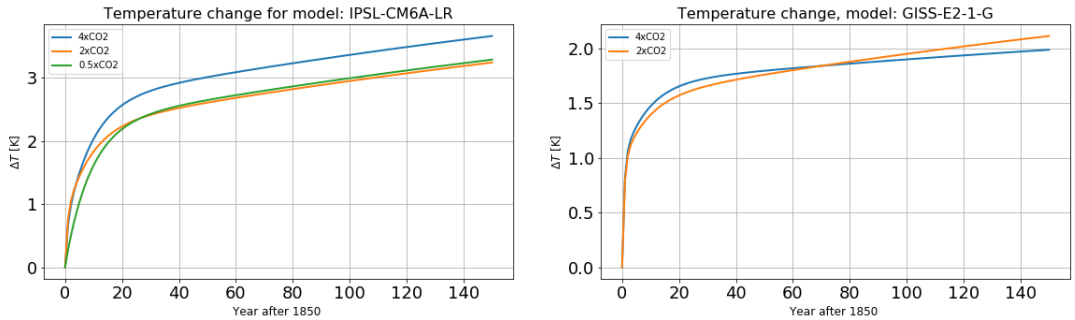
where C is the CO_2 concentration and C_0 is the reference level of CO_2 concentration, usually pre-industrial. The number 5.35 comes from the Third IPCC rapport (TAR), but varies between models as we can see in table 5.2. For the

IPSL-model we have data for $0.5\times CO_2$, $2\times CO_2$ and $4\times CO_2$. For the GISS-model we have data for $2\times CO_2$ and $4\times CO_2$.

Table 5.2: Comparing different types results using different kinds of data from CMIP6

Model, type of data	Temperature change	forcing	feedback parameter
IPSL-CM6A-LR, $0.5\times CO_2$	−3.96	−2.75	−0.695
IPSL-CM6A-LR, $2\times CO_2$	3.85	3.366	−0.873
IPSL-CM6A-LR, $4\times CO_2$	9.39	6.35	−0.676
GISS-E2-1-G, $2\times CO_2$	2.55	3.71	−1.46
GISS-E2-1-G, $4\times CO_2$	5.19	7.68	−1.48

In our comparison of the different experiments from the same models, we also compare the time evolutions of the temperature changes. As in figure 4.1, three exponential responses are fitted to the temperatures. We do this for all the different data and scale them such at all of them are a response to $2\times CO_2$. This difference between $4\times CO_2$ and the $2\times CO_2$ could be due to the estimates of forcing. But since the forcing estimates also contains uncertainties, we can not say for sure. In figure 5.2a we see that the 0.5 and the $2\times CO_2$ data give approximately the same response to $2\times CO_2$, while the $4\times CO_2$-data have a higher estimate, but lies with approximately the same distance to the two other curves at all times. In figure 5.2b we see that the responses using both $2\times CO_2$ and $4\times CO_2$ are not so far from each other, but since the difference between them is changing a bit, it is hard to tell how they would have looked if we looked at a longer time scale.



(a) Comparing response to 0.5, 2 and $4\times CO_2$-data for model IPSL-CM6A-LR

(b) Comparing response to 2 and $4\times CO_2$-data for model GISS-E2-1-G

Figure 5.2: Using data from CMIP6 and compare the response the different types of data available. If the assumption of linearity holds, we will get the same response from $0.5 \cdot 4\times CO_2$ and $-0.5\times CO_2$ as for $2\times CO_2$.

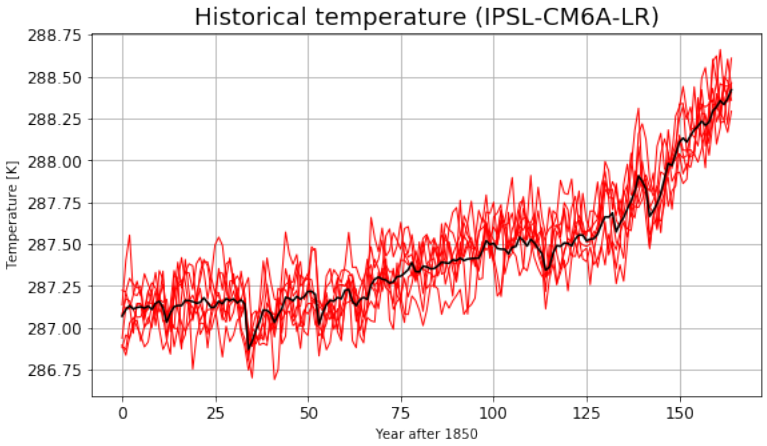


Figure 5.3: All the different runs of historical temperatures (red curves) plotted together with the mean (black curve) for model IPSL-CM6A-LR. The mean is found by taking the mean of the 31 different runs in each time step.

The CMIP6 model IPSL-CM6A-LR has 31 different historical runs. Since we don't have similar forcing estimates available as for CMIP5, we estimate the expected temperature response to historical forcing as the mean of these 31 runs. Subtracting the mean for each run, gives us 31 different residual vectors that we can make a power spectral density of. This gives us 31 estimates of S(0). S(0) is estimated using both methods mentioned earlier, and we get quite different results. First using method 1, the mean of the windowed periodogram is computed for the 25 lowest frequencies. We get that the estimates of S(0) vary from 0.04 to 0.20 with a mean value of all the estimates of 0.09 and standard deviation 0.04. Using method 2, where we fit a function to the PSD, we get 31 different estimates between 0.05 and 1.31, with a mean value of 0.44 and standard deviation of 0.35. Figure 4.14 shows the difference in estimates of $S(0)$ due to time scale. These uncertainties are much smaller compared to the difference in estimates we get looking at these different historical runs. So the greatest uncertainty in the estimates of $S(0)$ lies within the realisation of the historical run.

6

Discussion and Conclusion

In this thesis we have looked at two different ways to estimate the temperature response to a doubling of CO_2 in the atmosphere, using $4\times CO_2$ data and historical data from CMIP5. This was first done for one choice of time scales and we saw that using historical data gave a stronger response than using $4\times CO_2$-data (figure 4.6)for most models. By looking at 1000 different choices for time scales, we see that the uncertainties of the historical estimates are also very large (figure 4.13 and appendix A). When we use these two methods to try to say something about the ECS, we would need to know the response after it has reached a new equilibrium due to an abrupt doubling of CO_2 in the atmosphere. We only have data for 150 years using historical and $4\times CO_2$-data. The response due to $4\times CO_2$-data (figure 4.12a) looks like it is going to stabilize and have a much more flat curve than the response to historical data (figure 4.12b) which looks like it will increase even more before starting to flatten out.

Using the fluctuation-response relation we want to find estimates for the spectral density evaluated in the lower frequencies (S(0)), such that we may be able to use this to make an estimate for the ECS. We used two different methods to make an estimate for S(0): the method where we take the mean of the 25 first points (method 1), and the method where use the information from all frequencies to make a fit and take out the first element (method 2). Using the control run and method 1, we get a quite good correlation between ECS found by Gregory-plot squared (ECS^2) and S(0)(correlation ~ 0.72). When we use both method 1 and method 2 for the historical data, we find a correlation of

0.56 (method 1) and 0.20 (method 2) for the time scales $\tau = (0.7, 9, 354)$ years. For 4×CO_2-data we find a correlation of 0.05 (method 1) and −0.03 (method 2). We see that method 1 gives us some better correlation in general and especially for the historical run compared to using 4×CO_2-data. We took a brief look at the results we could get using the coefficients from 4×CO_2-data to estimate temperature response to historical forcing, instead of estimating the parameters from historical temperatures. In this way we got a different estimate of the residual. We got the same correlation as with historical estimates of c_k using method 2 (using the fitted curve to the PSD), but slightly lower for method 1 (the mean of the first 25 data points from the PSD). As we have seen previously, historical estimates of c_k have a large uncertainty. On the other hand , using estimated c_k from 4×CO_2 experiments assumes a linear response and there may be large uncertainties associated with this assumption as well. We have no arguments that say that one method is better than the other (for estimating c_k). Both methods will include uncertainties and it is hard to say which will have the greatest uncertainty.

We choose to only look at the correlation between ECS^2 and S(0) since we saw that the correlation decreased a lot when including other parameters. The correlation for the control run is much better than the ones we get from historical and 4×CO_2 data. This can be because of the choice of time scale that may affect the estimate of S(0), or maybe the linearity assumption does not hold. There is also a great uncertainty in the estimates of ECS. If we look at figure 4.14, where we include the 1000 different choices of time scales and look at the correlation between ECS^2 and the mean values of S(0) (for historical run, using method 2), we get approximately the same correlation, 0.200, which is still a pretty low correlation. From figure 4.14 we see that the estimates of S(0) changes with the choice of time scale and that for some models S(0)-values vary a lot more than for others. For the models in CMIP5 we only have one analysed historical run per model, but for the IPSL-model in CMIP6 we have 31 historical runs. By making a fit to all these runs (figure 5.3) and look at the difference between the fit and each of the 31 historical runs, we can get 31 different estimates for S(0) for this model. From this we get estimates that vary between 0.04 and 0.20 for method 1, and between 0.05 and 1.31 using method 2. We have just seen how much difference we can get in our results due to the choice of time scale. This shows us how much uncertainties there is in the historical runs, which leads to an even greater uncertainty in the S(0)-values. The uncertainties from using the different realizations are larger than the uncertainty from the choice of time scales.

The pattern of warming have a lot to say when we talk about climate sensitivity, and if the future warming pattern is different to the past, then we might not be able to predict it using historical records [Andrews et al., 2018]. The so-called

pattern effect is something that need to take into account. Such as sea surface temperature (SST) and sea ice will have an impact on the warming pattern. This are factors that we don't account for in this thesis. Andrews et al. (2018) used historical runs and included the change in sea ice and the sea surface temperature, and compare it to an abrupt $4\times CO_2$ for the corresponding models. By doing this they got a higher estimate from the forced abrupt $4\times CO_2$ than from using the historical records with adjustments, which is the opposite of what we got from our results. The effective climate sensitivity, the sensitivity on a timescale of about a century, for $4\times CO_2$-data, Andrews et al. (2018) found to be around 2.4-4.6K, which is not so far from the results we got looking at the response for $4\times CO_2$-data for all the models in appendix A. Using historical SST and sea ice changes, they got an effective climate sensitivity about 2K. Our results are very dependent on the time scale, but in most cases we estimate higher values. But we see that both using the $4\times CO_2$ and historical data gives different estimates for the effective climate sensitivity, and as they say in Andrews et al. (2018), "This is in contrast to decades of studies that explicitly or implicitly assume that the relationship between historical temperature change and energy budget variations provides a direct constraint on long-term climate sensitivity" (p.8490). We see from our results that the feedback from historical records and the feedback due to changes in the CO_2-concentration in the atmosphere may lead to different estimates for ECS. So maybe we can use historical records to estimate the future if we take into account the pattern effects. We see that our methods and results give very high uncertainties, such that we can not see any clear connection between ECS^2 and $S(0)$. Using $4\times CO_2$ and historical data gives results that differ a lot, and that the historical data may not be used directly such as we used the data.

Both Otto et al. (2013) and Dessler and Forster (2018) have also estimated ECS from the historical period, just with additional data. Using the same equation for energy balance, $N = -\lambda\Delta T + \Delta F$, where N is the heat uptake in the climate system such that $N = \Delta Q = E_{\text{in}} - E_{\text{out}}$. They rewrite λ using $F_{2x} = \text{ECS} \cdot \lambda$, where F_{2x} is the forcing due to a doubling of CO_2. This gives an expression for ECS

$$\text{ECS} = \frac{F_{2x}\Delta T}{\Delta F - \Delta Q}.$$

Using information about ΔQ they are able to say more about ECS than we are. They estimate the change in total heat uptake of the Earth system using data-based estimates for the main components of the system (ocean, contonent, ice and atmosphere) [Otto et al., 2013], or observations of radiation using satelite data [Dessler and Forster, 2018].

We assume linearity while doing this, but is the response really linear? Is the response to a $4\times CO_2$ the same as twice the response to $2\times CO_2$? Using CMIP6

and comparing the different types of data that are available for IPSL-CM6A-LR and GISS-E2-1-G, we can try to say something about this linearity assumption. The negative of $0.5\times CO_2$ and the $2\times CO_2$ for the IPSL-model give approximately the same response, while the $4\times CO_2 \cdot \frac{1}{2}$ data gives a slightly higher response. Looking at figure 5.2a, we observe that the difference between the response from $4\times CO_2 \cdot \frac{1}{2}$ and $2\times CO_2/-0.5\times CO_2$, is almost constant. This means that if there is some kind of bias to correct for, it might as well end up giving the same estimate as the two others. Looking at figure 5.2b, where we see $2\times CO_2$ and $\frac{1}{2} \cdot 4\times CO_2$ for the GISS-model, we see that the responses lay close to each other, but it is hard to tell how they will behave after the 150 years. They may both flatten out and give a quite similar results, or they may continue to grow apart from each other. This is only done for these two models, since that was the only available models with these experiments in CMIP6 at this time. If we were to use more models and maybe more tests per model, we might be able to draw a conclusion for the linearity assumption. It might look linear for some of these results, but we don't have enough test results to make a conclusion.

In table 5.1 we compare CMIP5 and CMIP6 parameters from Gregory plots. We see that the CMIP6 models give a higher estimate for ECS than the CMIP5 models. The difference is especially large between the two IPSL-models. The CMIP6 GISS-model is an AGCM model while the other GISS-models are AOGCM models. The AOGCM has an interactive ocean, and will therefore have a slower response than AGCM models. So far, the updated models (CMIP6) gives us a higher estimate for ECS.

6.1 Conclusion

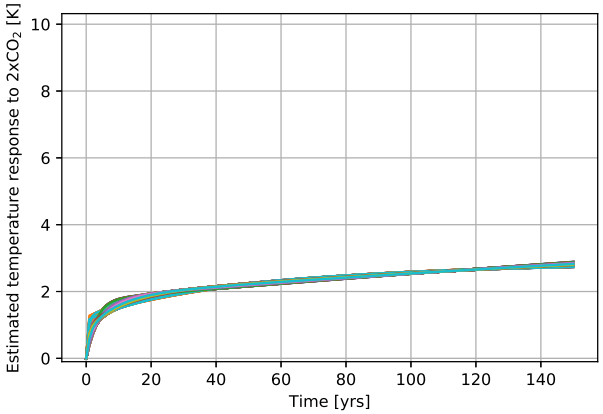
Using fluctuation-response relations for linear stochastic climate models and historical data (from 1850 to present) we have looked at the equilibrium climate sensitivity. Looking at both the response to historical forcing and the fluctuations around the response we have seen that we have great uncertainties in the relations, the choice of time scale and all the estimates that are done. Such that we cannot prove that we can use historical records to say something about the equilibrium temperature, using the relations and methods that we have used. We cannot say if it is the relation that doesn't hold or if it is the uncertainties that gives us this result.

Other methods that have smaller uncertainties will maybe show that we are able to use historical records. Also using additional data as Otto et al. (2013), might help to be able to use historical data to estimate the ECS. The time period 1850-present is too short for estimation of ECS, and we may ultimately have to rely on longer reconstructed temperature time series or satellite measurements of Earth's energy imbalance.

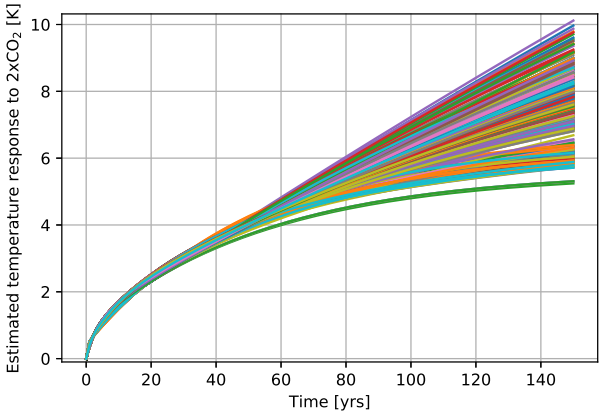
A

Additional plots for testing the different choice of time scales

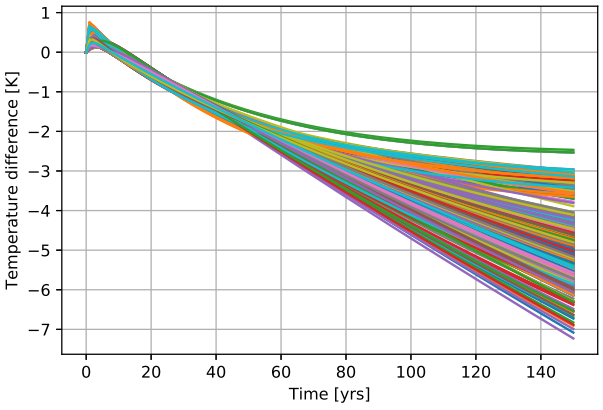
When Proistosescu and Huybers (2017) estimated linear responses functions for GCMs, they employed a Bayesian method also estimating the time scales. By doing sampling from this selection of time scales, we are able to see how the different models react to the different choice of time scales. We make 1000 different samples and compute responses as in figure 4.1 (adjusted response to $4\times CO_2$), figure 4.5 (response to forcing using historical data), and the difference between these two methods, such as shown in figure 4.6 and 4.7a. This is done for all the 16 climate models that are included in this thesis.



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

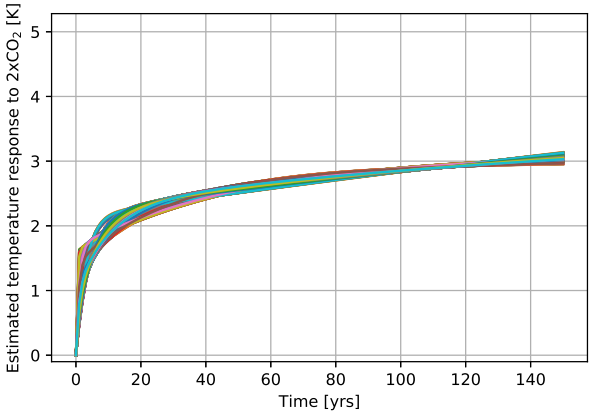


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

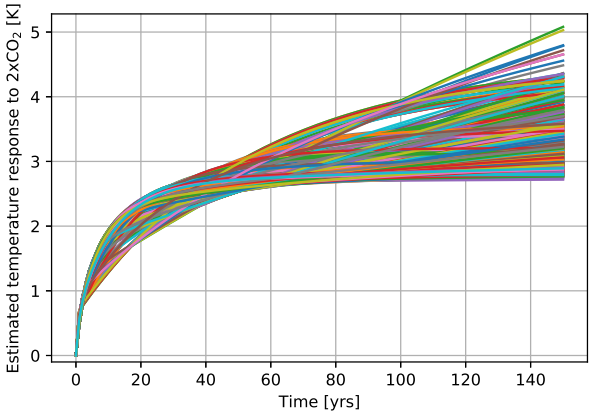


(c) Difference between the responses in (a) and (b).

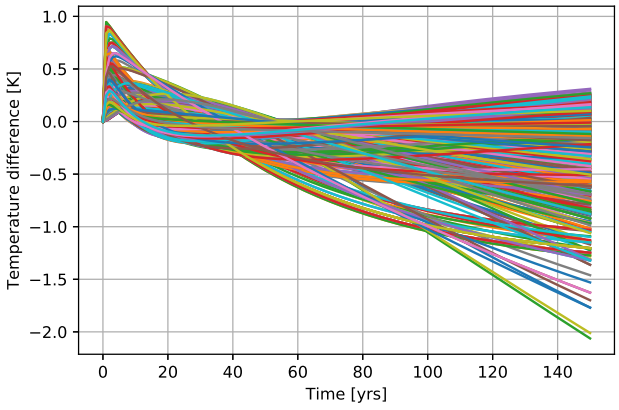
Figure A.1: Model ACCESS1-0



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

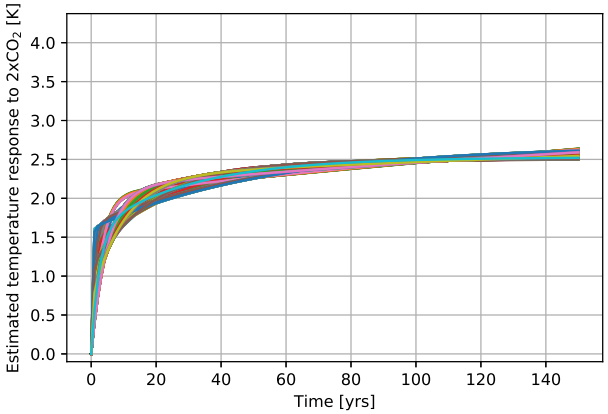


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

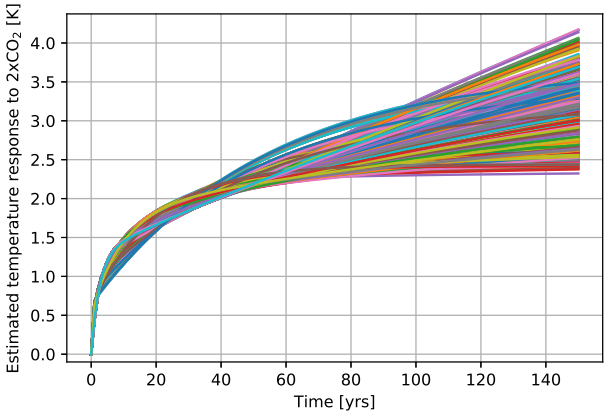


(c) Difference between the responses in (a) and (b).

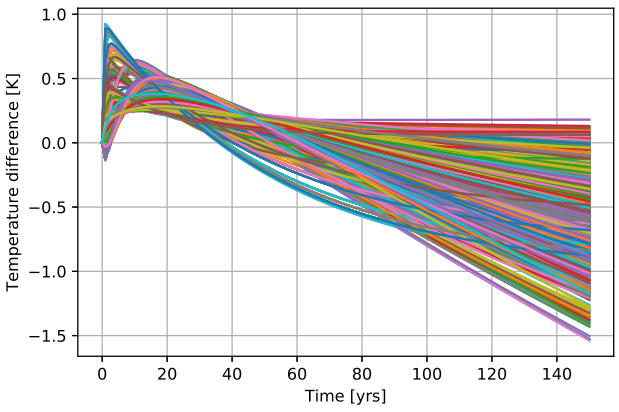
Figure A.2: Model CanESM2



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

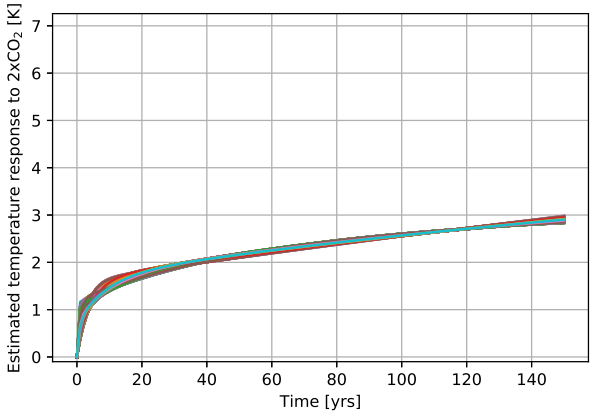


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

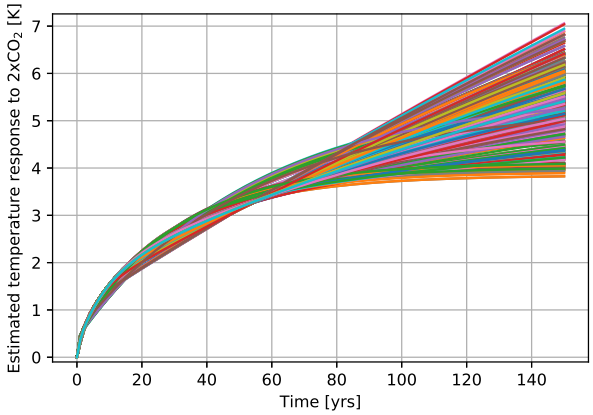


(c) Difference between the responses in (a) and (b).

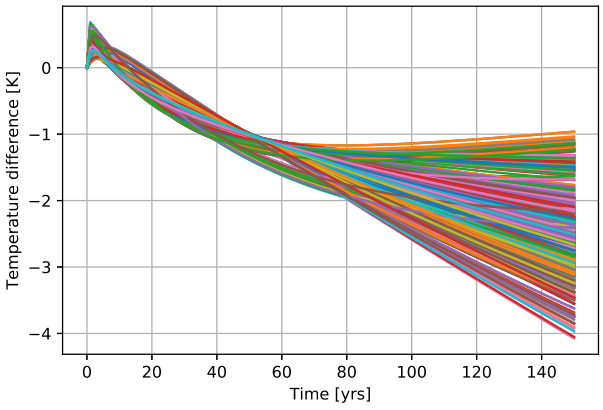
Figure A.3: Model CNRM-CM5



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

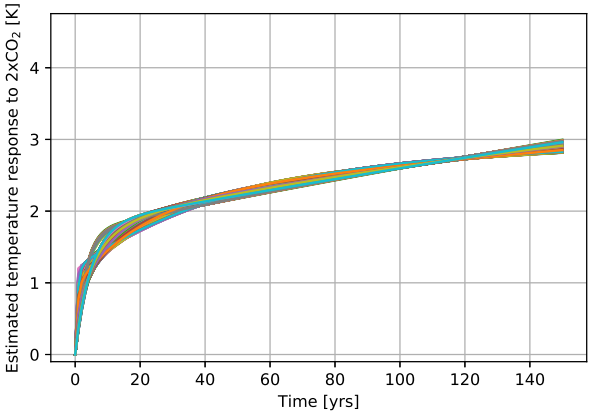


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

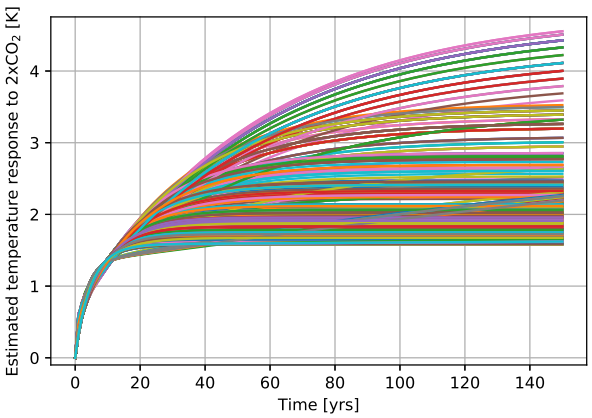


(c) Difference between the responses in (a) and (b).

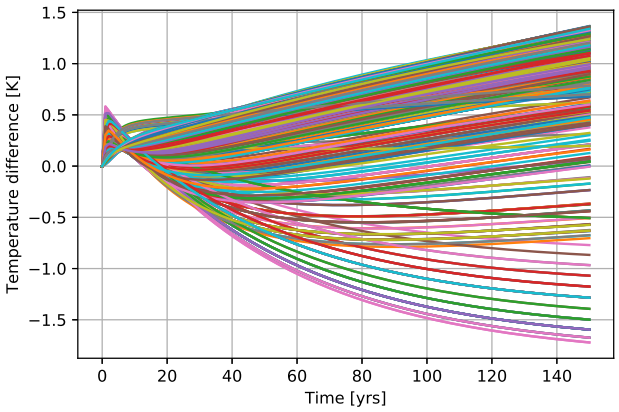
Figure A.4: Model CSIRO-Mk3-6-0



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

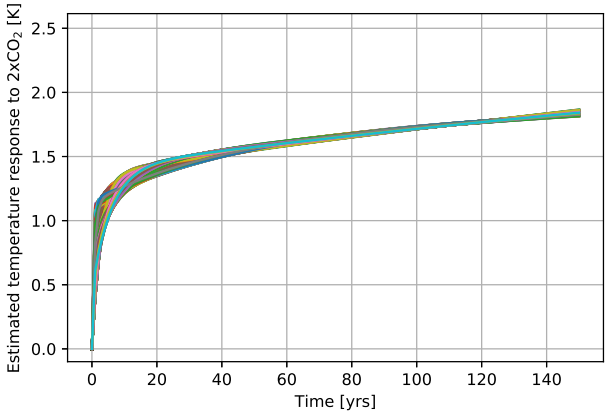


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

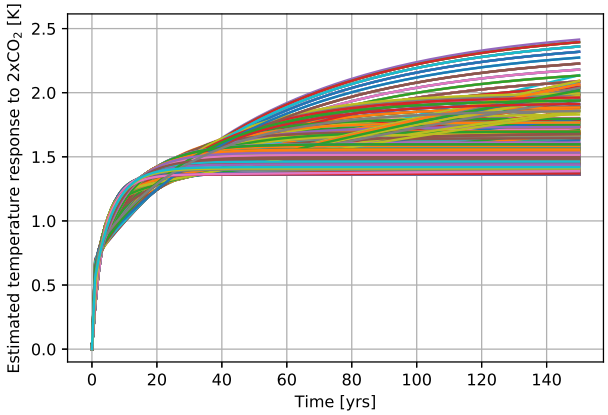


(c) Difference between the responses in (a) and (b).

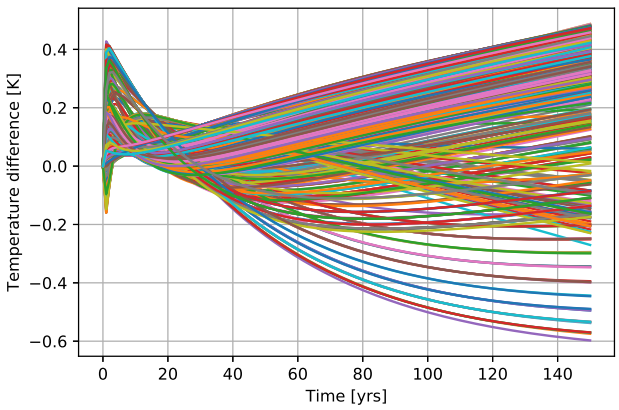
Figure A.5: Model GFDL-CM3



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

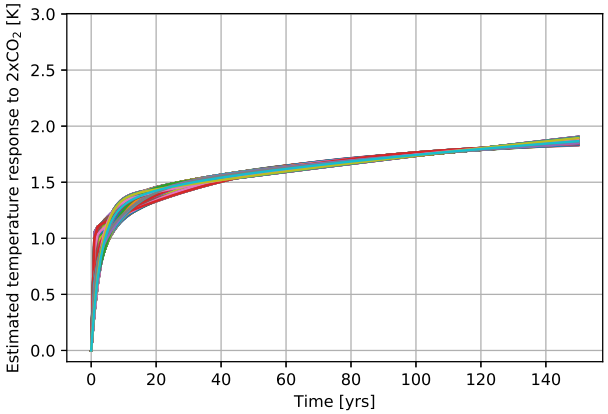


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

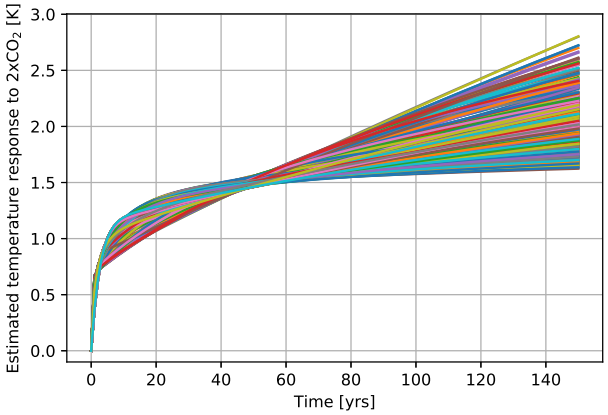


(c) Difference between the responses in (a) and (b).

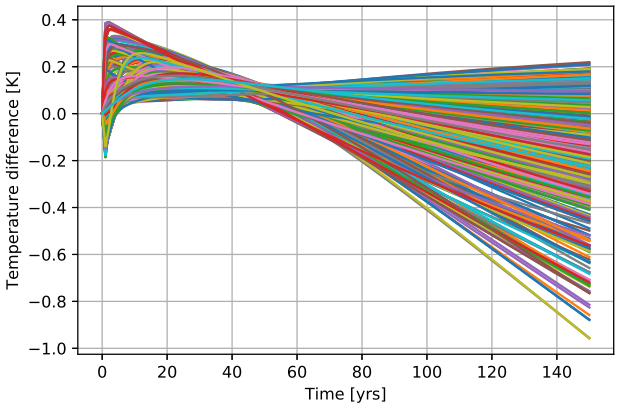
Figure A.6: Model GFDL-ESM2G



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

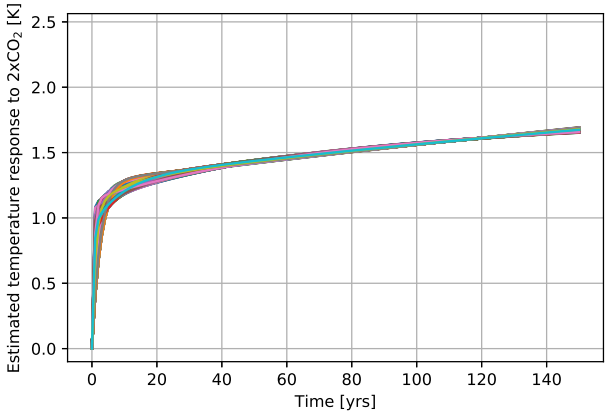


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

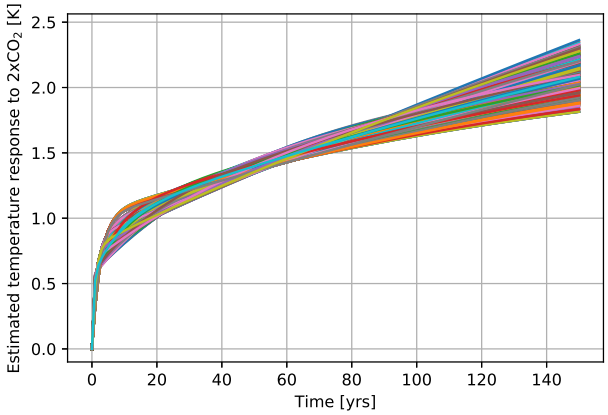


(c) Difference between the responses in (a) and (b).

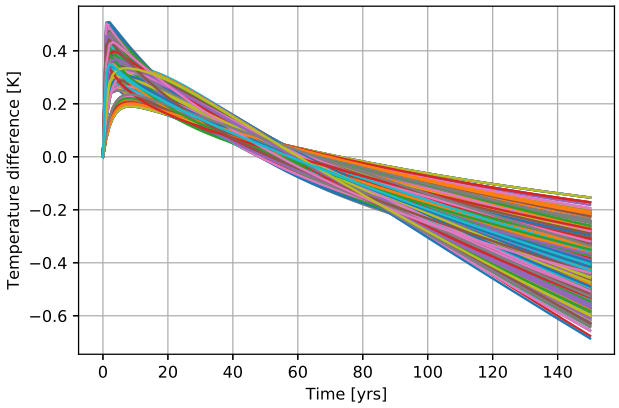
Figure A.7: Model GFDL-ESM2M



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

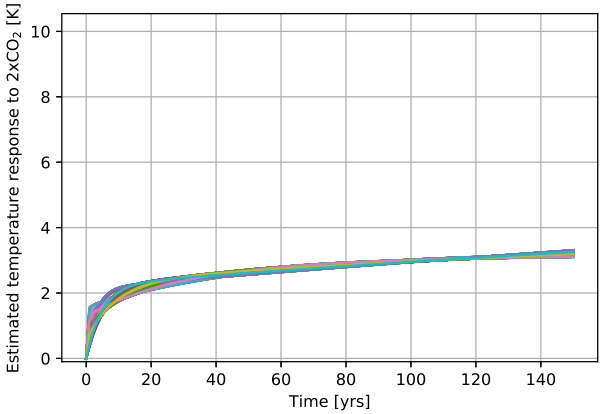


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

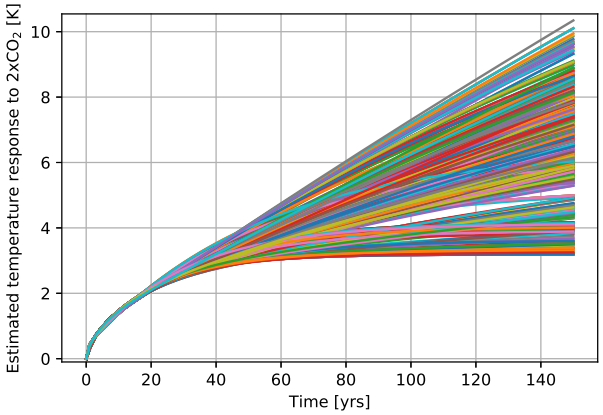


(c) Difference between the responses in (a) and (b).

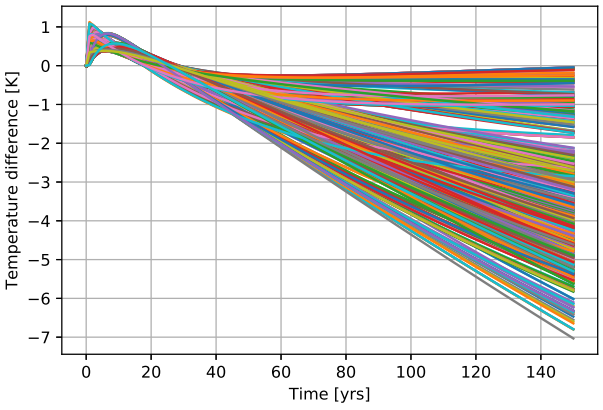
Figure A.8: Model GISS-E2-R



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

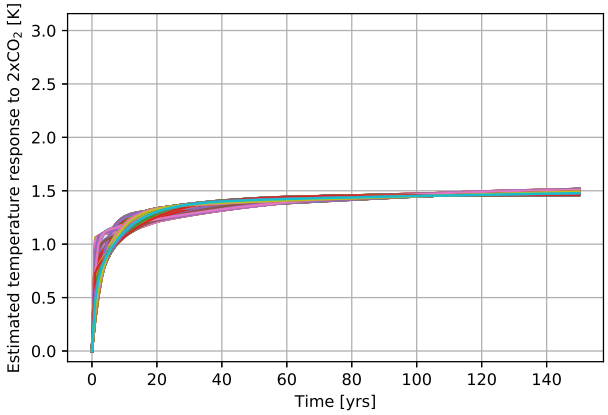


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

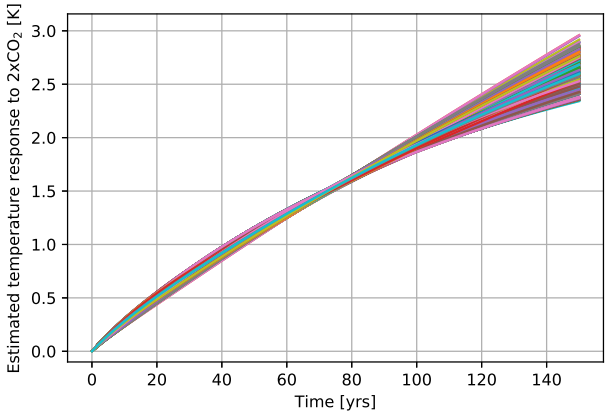


(c) Difference between the responses in (a) and (b).

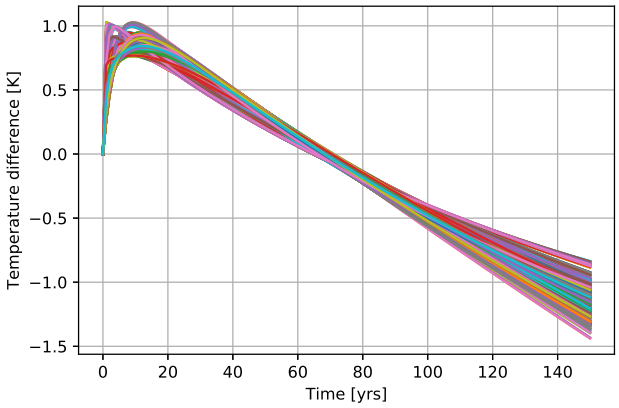
Figure A.9: Model HadGEM2-ES



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

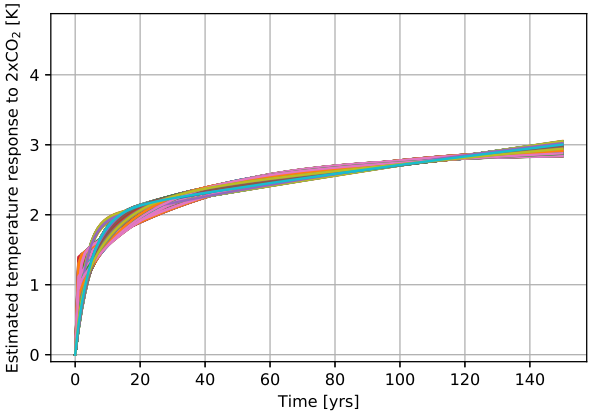


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

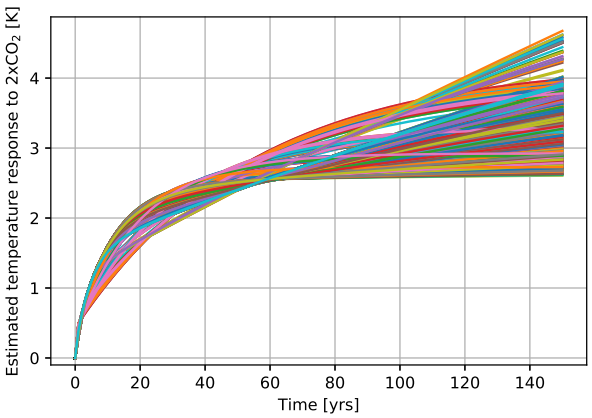


(c) Difference between the responses in (a) and (b).

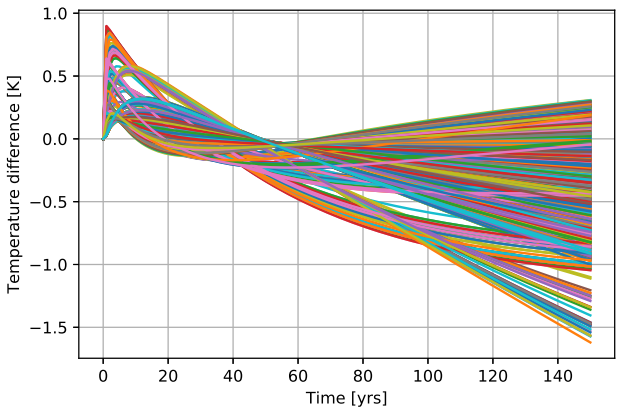
Figure A.10: Model inmcm4



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

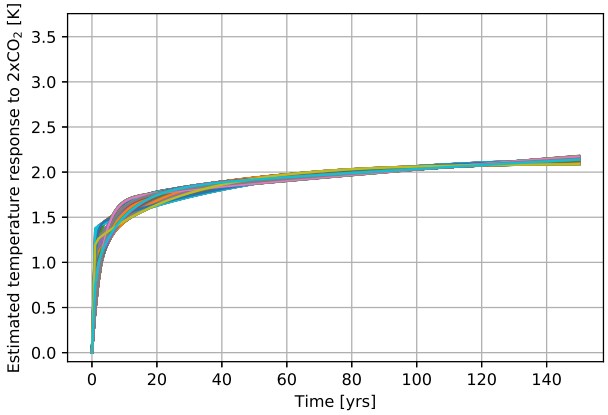


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

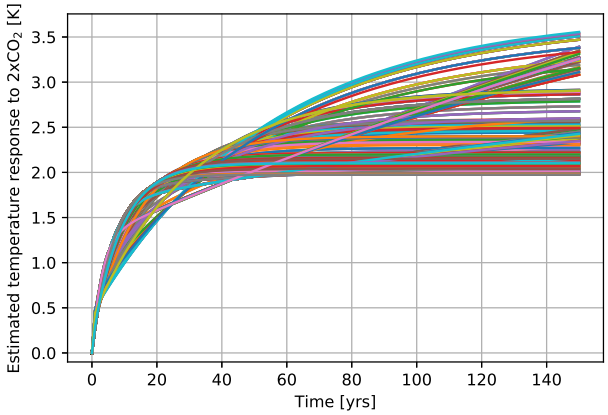


(c) Difference between the responses in (a) and (b).

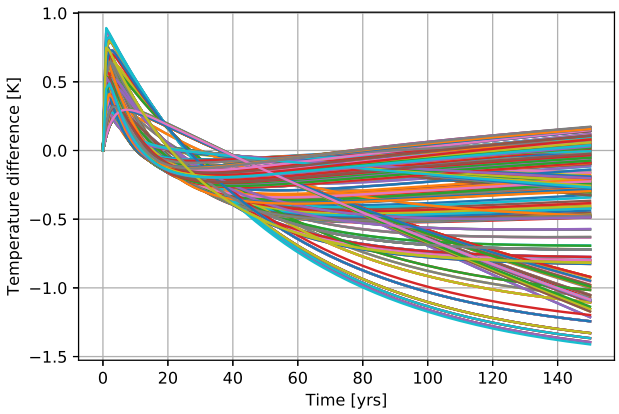
Figure A.11: Model IPSL-CM5A-LR



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

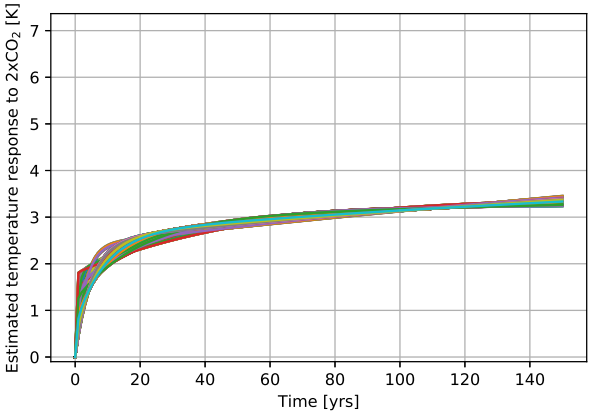


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

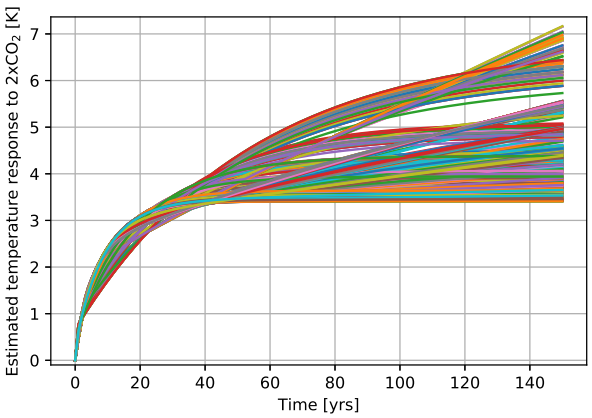


(c) Difference between the responses in (a) and (b).

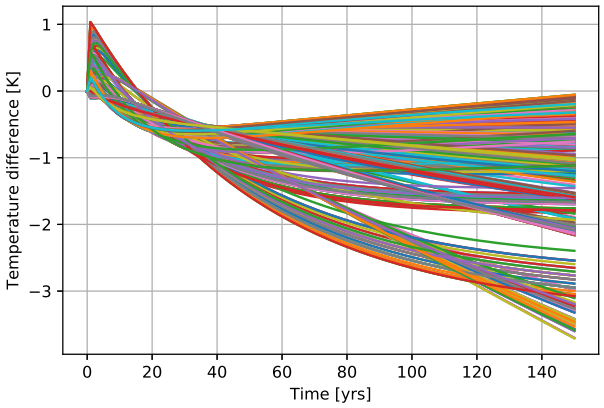
Figure A.12: Model MIROC5



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

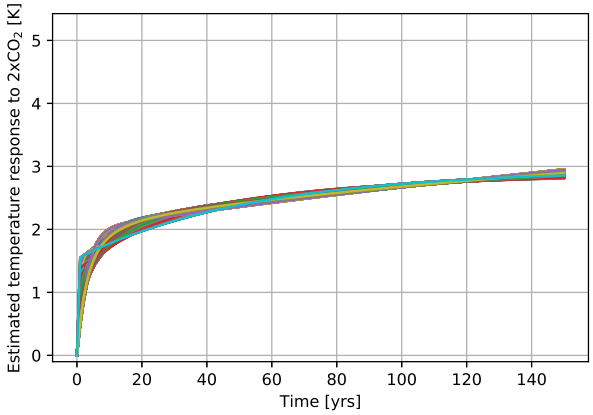


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

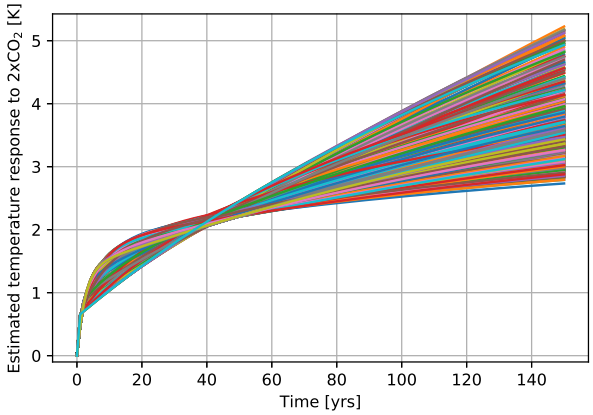


(c) Difference between the responses in (a) and (b).

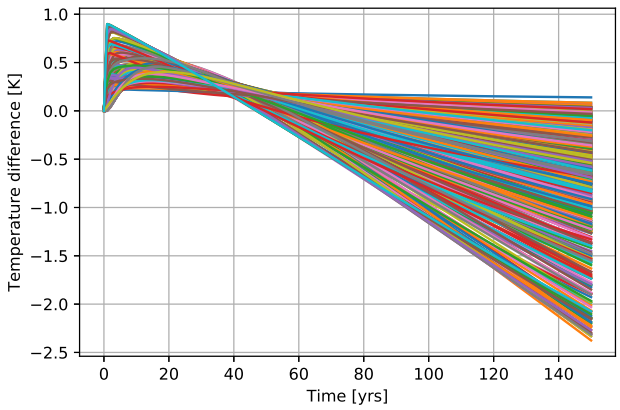
Figure A.13: Model MIROC-ESM



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

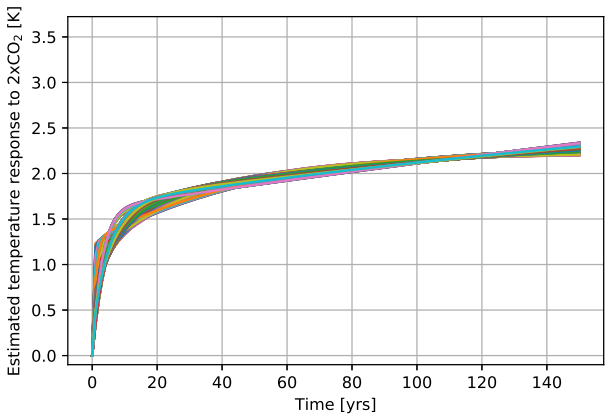


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

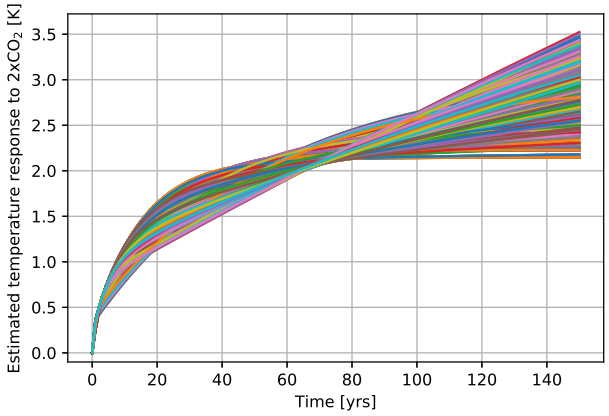


(c) Difference between the responses in (a) and (b).

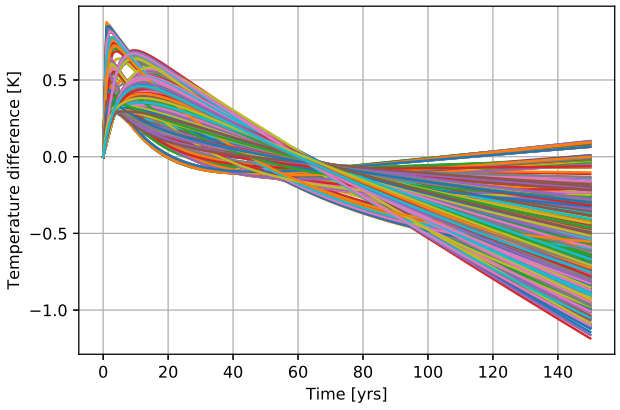
Figure A.14: Model MPI-ESM-LR



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.

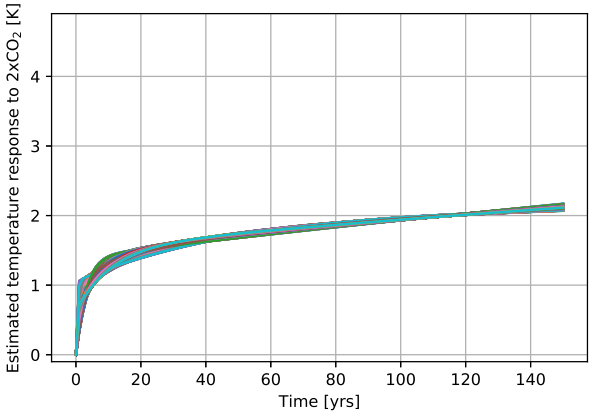


(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.

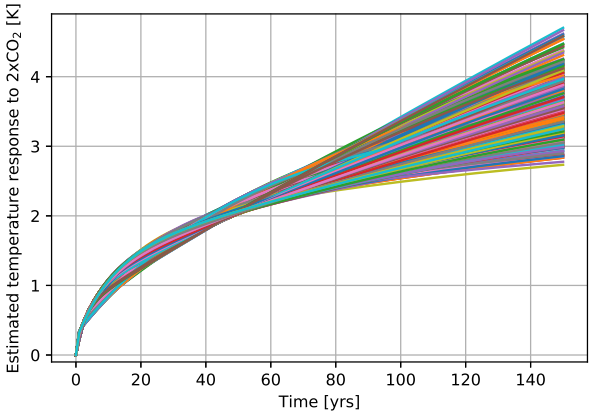


(c) Difference between the responses in (a) and (b).

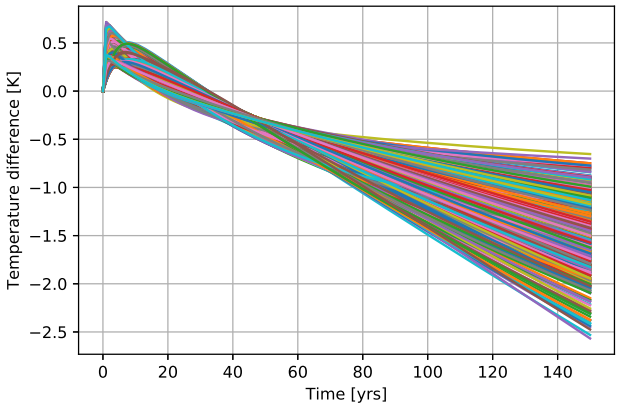
Figure A.15: Model MRI-CGCM3



(a) Adjusted response to $4 \times CO_2$, computed as in figure 4.1. Further this response is divided by 2, to convert it to the corresponding $2{\times}CO_2$ response.



(b) Response to forcing using estimated historical data as in figure 4.5, and then converted to the corresponding response to $2{\times}CO_2$, as in figure 4.6.



(c) Difference between the responses in (a) and (b).

Figure A.16: Model NorESM1-M

B

Fourier Transform

Here we look into some characteristics for the Fourier transform, and show these two equations:

$$\widehat{g * h} = \hat{g}(f) \cdot \hat{h}(f) \tag{B.1}$$

$$\widehat{g \cdot h} = \hat{g}(f) * \hat{h}(f) \tag{B.2}$$

where the hat represents the Fourier transform. The definition of the convolution between the functions g and h is

$$(g * h)(t) \equiv \int_{-\infty}^{\infty} g(t-s)h(s)ds. \tag{B.3}$$

By applying the Fourier transform, we change a function from the time domain to the frequency domain,

$$\hat{x}(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi i f t}dt \tag{B.4}$$

and from the frequency domain to the time domain using the inverse transform

$$x(t) = \int_{-\infty}^{\infty} \hat{x}(f) e^{2\pi i f t} df, \tag{B.5}$$

[Kaper and Engler, 2013].

To show the relation (B.1) we start with the convolution between two functions and then applying the definition of a Fourier transform, equation (B.4), and the definition of the convolution between to functions, equation (B.3), and do a change in variables:

$$\widehat{g * h} = \int_{-\infty}^{\infty} (g * h) e^{-2\pi i f t} dt = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(t-s) h(s) e^{-2\pi i f t} ds dt.$$

We define $p = t - s$ such that $g(t-s) = g(p)$ and $t = s + p$. Using this substitution, we can separate the variable such that we get two integrals multiplied, one where we integrate over p and the other where we integrate over s, such that

$$\widehat{g * h} = \int_{-\infty}^{\infty} g(p) e^{-2\pi i f p} dp \cdot \int_{-\infty}^{\infty} h(s) e^{-2\pi i f s} ds,$$

$$\widehat{g * h} = \hat{g}(f) \cdot \hat{h}(f). \tag{B.6}$$

The Fourier transform of two functions multiplied in time domain is equal to the Fourier transform of the convolution between the functions in frequency domain, $\widehat{gh} = \hat{g} * \hat{h}$, [Almeida, 1997]. This can be shown by taking the Fourier transform of the two functions multiplied, and then using equation (B.5) to express one of the functions.

$$\widehat{g \cdot h} = \int_{-\infty}^{\infty} g(t) h(t) e^{-2\pi i f t} dt = \int_{-\infty}^{\infty} g(t) \left(\int_{-\infty}^{\infty} \hat{h}(f') e^{2\pi i f' t} df' \right) e^{-2\pi i f t} dt.$$

We can change the order of the integrals. Using equation (B.5), we can rewrite the inner integral such that the expression changes from time domain to frequency domain. Then we get that

$$\widehat{g \cdot h} = \int_{-\infty}^{\infty} \hat{h}(f') \left(\int_{-\infty}^{\infty} g(t) e^{-2\pi i (f - f')t} dt \right) df' = \int_{-\infty}^{\infty} \hat{h}(f') \hat{g}(f - f') df',$$

$$\widehat{g \cdot h} = \hat{g}(f) * \hat{h}(f) \tag{B.7}$$

C

Python Code

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

filedir1 = '/Users/Bruker/Desktop/Forcingpaperdata'

#model = 'CanESM2'
#model = 'CNRM-CM5'
#model = 'CSIRO-Mk3-6-0'
#model = 'GFDL-CM3'
#model = 'GFDL-ESM2G'
#model = 'GFDL-ESM2M'
#model = 'GISS-E2-H'
#model = 'GISS-E2-R'
#model = 'HadGEM2-ES'
#model = 'inmcm4'
#model = 'IPSL-CM5A-LR'
#model = 'IPSL-CM5B-LR'
#model = 'MIROC-ESM'
#model = 'MIROC5'
#model = 'MPI-ESM-LR'
#model = 'MRI-CGCM3'
model = 'NorESM1-M'

realm = 'Amon'
ensemble = 'r1i1p1'

if model == 'ACCESS1-0':
```

```
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '030001-079912'
    rcptimeperiod = '200601-210012' #rcp45, rcp85
    # rcp26 and rcp60 not available
    control_branch_yr = 300
elif model == 'ACCESS1-3':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '025001-074912'
    rcptimeperiod = '200601-210012' #rcp45, rcp85
    # rcp26 and rcp60 not available
    control_branch_yr = 250
elif model == 'bcc-csm1-1':
    historicaltimeperiod = '185001-201212'
    print(model + 'has no control run')
    #rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-210012' #rcp60
    rcptimeperiod = '200601-209912' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = NaN
elif model == 'bcc-csm1-1-m':
    historicaltimeperiod = '185001-201212'
    print(model + 'has no control run')
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = NaN
elif model == 'CanESM2':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '201501-301012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-230012' #rcp26, rcp45
    #rcp60 not available
    control_branch_yr = 2321
elif model == 'CCSM4':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '025001-130012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    #rcptimeperiod = '200601-230012' #rcp26, rcp60, rcp85
    #rcptimeperiod = '200601-229912' #rcp45
    control_branch_yr = 937
elif model == 'CNRM-CM5':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-269912'
    #rcptimeperiod = '200601-210012' #rcp26
    rcptimeperiod = '200601-230012' #rcp45, rcp85
    #rcp60 not available
    control_branch_yr = 2250
elif model == 'CSIRO-Mk3-6-0':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 81
elif model == 'FGOALS-s2':
    print(model + 'has no historical run')
    controltimeperiod = '185001-235012'
    print(model + 'has no rcp runs for r1')
    control_branch_yr = NaN
```

```
elif model == 'GFDL-CM3':
    historicaltimeperiod = '186001-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 1
elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    historicaltimeperiod = '186101-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 162
elif model == 'GISS-E2-H':
    historicaltimeperiod = '185001-200512'
    print(model + 'has control run for two different periods')
    #controltimeperiod = '118001-141912'
    controltimeperiod = '241001-294912'
    #rcptimeperiod = '200601-210012' #rcp60
    rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    control_branch_yr = 2410
elif model == 'GISS-E2-R':
    historicaltimeperiod = '185001-200512'
    print(model + ' has two different control runs')
    #controltimeperiod = '333101-363012'
    controltimeperiod1 = '398101-453012'
    controltimeperiod2 = '398101-920512'
    #rcptimeperiod = '200601-210012' #rcp60
    rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    control_branch_yr = 3981
elif model == 'HadGEM2-ES':
    historicaltimeperiod = '186001-200511'
    controltimeperiod = '186001-243511'
    rcptimeperiod = '200601-229912' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-209911' #rcp60
    control_branch_yr = 1860
elif model == 'inmcm4':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-234912'
    rcptimeperiod = '200601-210012' # rcp45, rcp85
    #rcp26 and rcp60 not available
    control_branch_yr = 1850
elif model == 'IPSL-CM5A-LR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '180001-279912'
    rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-210012' #rcp60
    control_branch_yr = 1850
elif model == 'IPSL-CM5B-LR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '183001-212912'
    rcptimeperiod = '200601-210012' #rcp45, rcp85
    #rcp26 and rcp60 not available
    control_branch_yr = 1850
elif model == 'MIROC-ESM':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '180001-242912'
```

```
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    #rcptimeperiod = '200601-230012' #rcp45
    control_branch_yr = 1880
elif model == 'MIROC5':
    historicaltimeperiod = '185001-201212'
    controltimeperiod = '200001-286912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 2411

elif model == 'MPI-ESM-LR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-284912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    # rcp60 not available
    control_branch_yr = 1880
elif model == 'MPI-ESM-MR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-284912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp85
    # rcp60 not available
    control_branch_yr = 1850
elif model == 'MPI-ESM-P':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-300512'
    print(model + 'has no rcp runs')
    control_branch_yr = NaN
elif model == 'MRI-CGCM3':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185101-235012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 1950
elif model == 'NorESM1-M':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '070001-120012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    #rcptimeperiod = '200601-230012' #rcp45
    control_branch_yr = 700

###### load control run data ######
exp = 'piControl'

var = 'ts' # temperatures
if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod1

strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]

controlyears = np.arange(0,len(controltemp))
```

```
# subtract linear trend
p1 = np.polyfit(controlyears, controltemp, deg = 1)
lintrendT = np.polyval(p1,controlyears)

T = controltemp - lintrendT


# Compare with periodogram estimate for the PSD (more noise expected
    )

from scipy import signal

#pf,pS = signal.periodogram(T, window = 'hamming')
pf,pS = signal.welch(T, nperseg=256)
# This also computes the value for f = 0, which is approx 0 (= mean)

# create figure
fig, ax = plt.subplots(figsize = [10,6])
ax.plot(pf[1:],pS[1:]) # exclude f=0 from plot
ax.set_xscale('log')
ax.set_yscale('log')
ax.set_xlabel('f (1/years)',fontsize = 18)
ax.set_ylabel('S(f)',fontsize = 18)
ax.set_title('PSD for control temperature',fontsize = 18)
ax.grid()
ax.tick_params(axis='both',labelsize=18)


#for i=5
mean_explog=np.exp(np.mean(np.log(pS[1:5])))
mean_ps=np.mean(pS[1:5])

dp = pd.read_csv('mean_val.csv', index_col = 0)
dp.meanexp_5[model] = mean_explog
dp.mean_5[model]= mean_ps
dp.to_csv('mean_val.csv')

#for i=10
mean_explog=np.exp(np.mean(np.log(pS[1:10])))
mean_ps=np.mean(pS[1:10])

dp = pd.read_csv('mean_val.csv', index_col = 0)
dp.meanexp_10[model] = mean_explog
dp.mean_10[model]= mean_ps
dp.to_csv('mean_val.csv')

#for i=15
mean_explog=np.exp(np.mean(np.log(pS[1:15])))
mean_ps=np.mean(pS[1:15])

dp = pd.read_csv('mean_val.csv', index_col = 0)
dp.meanexp_15[model] = mean_explog
dp.mean_15[model]= mean_ps
dp.to_csv('mean_val.csv')
```

```
#for i=20
mean_explog=np.exp(np.mean(np.log(pS[1:20])))
mean_ps=np.mean(pS[1:20])

dp = pd.read_csv('mean_val.csv', index_col = 0)
dp.meanexp_20[model] = mean_explog
dp.mean_20[model]= mean_ps
dp.to_csv('mean_val.csv')

#for i =25
mean_explog=np.exp(np.mean(np.log(pS[1:25])))
mean_ps=np.mean(pS[1:25])

dp = pd.read_csv('mean_val.csv', index_col = 0)
dp.meanexp_25[model] = mean_explog
dp.mean_25[model]= mean_ps
dp.to_csv('mean_val.csv')


df = pd.read_csv('ECS2_est.csv', index_col=0)
#p1 = np.polyfit((dq.ECS)**2,est1,deg = 1)
#p2 = np.polyfit((dq.ECS)**2,est2,deg = 1)

dr = pd.read_csv('correlation.csv', index_col = 0)

i ='5'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_5)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_5)[1,0]
dr.S_exp[i] = corr1
dr.S[i] = corr2

i='10'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_10)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_10)[1,0]
dr.S_exp[i] = corr1
dr.S[i] = corr2

i='15'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_15)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_15)[1,0]
dr.S_exp[i] = corr1
dr.S[i] = corr2

i ='20'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_20)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_20)[1,0]
dr.S_exp[i] = corr1
dr.S[i] = corr2

i='25'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_25)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_25)[1,0]
dr.S_exp[i] = corr1
```

```
dr.S[i] = corr2

dr.to_csv('correlation.csv')



#S(0)*Q^2
dr=pd.read_csv('correlation.csv', index_col = 0)
dq = pd.read_csv('ECS2_est.csv', index_col=0)

i='5'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_5*(dq.Q)**2)[1,0]
corr2=np.corrcoef((dq.ECS)**2,dp.mean_5*(dq.Q)**2)[1,0]
dr.SQ_exp[i] = corr1
dr.SQ[i] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_10*(dq.Q)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_10*(dq.Q)**2)[1,0]
dr.SQ_exp['10'] = corr1
dr.SQ['10'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_15*(dq.Q)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_15*(dq.Q)**2)[1,0]
dr.SQ_exp['15'] = corr1
dr.SQ['15'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_20*(dq.Q)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_20*(dq.Q)**2)[1,0]
dr.SQ_exp['20'] = corr1
dr.SQ['20'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_25*(dq.Q)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_25*(dq.Q)**2)[1,0]
dr.SQ_exp['25'] = corr1
dr.SQ['25'] = corr2

dr.to_csv('correlation.csv')

#S(0)/sigma**2
dr=pd.read_csv('correlation.csv', index_col = 0)

i='fem'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_5/(df.sigma)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_5/(df.sigma)**2)[1,0]
dr.Ssigma_exp[i]=corr1
dr.Ssigma[i]=corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_10/(df.sigma)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_10/(df.sigma)**2)[1,0]
dr.Ssigma_exp['10']=corr1
dr.Ssigma['10']=corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_15/(df.sigma)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_15/(df.sigma)**2)[1,0]
```

```
dr.Ssigma_exp['15']=corr1
dr.Ssigma['15']=corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_20/(df.sigma)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_20/(df.sigma)**2)[1,0]
dr.Ssigma_exp['20']=corr1
dr.Ssigma['20']=corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_25/(df.sigma)**2)[1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_25/(df.sigma)**2)[1,0]
dr.Ssigma_exp['25']=corr1
dr.Ssigma['25']=corr2

dr.to_csv('correlation.csv')

#S(0)*Q^2/sigma^2
dr=pd.read_csv('correlation.csv', index_col = 0)

i='fem'
corr1=np.corrcoef((df.ECS)**2,dp.meanexp_5*(df.Q)**2/(df.sigma)**2)
    [1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_5*(df.Q)**2/(df.sigma)**2)
    [1,0]
dr.SQsigma_exp[i] = corr1
dr.SQsigma[i] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_10*(df.Q)**2/(df.sigma)**2)
    [1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_10*(df.Q)**2/(df.sigma)**2)
    [1,0]
dr.SQsigma_exp['10'] = corr1
dr.SQsigma['10'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_15*(df.Q)**2/(df.sigma)**2)
    [1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_15*(df.Q)**2/(df.sigma)**2)
    [1,0]
dr.SQsigma_exp['15'] = corr1
dr.SQsigma['15'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_20*(df.Q)**2/(df.sigma)**2)
    [1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_20*(df.Q)**2/(df.sigma)**2)
    [1,0]
dr.SQsigma_exp['20'] = corr1
dr.SQsigma['20'] = corr2

corr1=np.corrcoef((df.ECS)**2,dp.meanexp_25*(df.Q)**2/(df.sigma)**2)
    [1,0]
corr2=np.corrcoef((df.ECS)**2,dp.mean_25*(df.Q)**2/(df.sigma)**2)
    [1,0]
dr.SQsigma_exp['25'] = corr1
dr.SQsigma['25'] = corr2
```

```
dr.to_csv('correlation.csv')

x_val = dp.meanexp_25
p1 = np.polyfit(x_val,(df.ECS)**2,deg = 1)

linfit = np.polyval(p1,x_val)
plt.plot(x_val,linfit,linewidth=1,color = "black")
corr = np.corrcoef(x_val,df.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(x_val,(df.ECS**2))
plt.xlabel('S(0)', fontsize=12)
plt.ylabel('ECS$^2$ [K$^2$]',fontsize=12)
plt.grid()
plt.savefig('ESC_S.eps')


x_val = (dp.meanexp_25)*((df.Q)**2)/(df.sigma)**2
p2 = np.polyfit(x_val,(df.ECS)**2, deg = 1)
corr = np.corrcoef(x_val,df.ECS**2)[1,0]
corr = round(corr, 5)

linfit = np.polyval(p2,x_val)
plt.plot(x_val,linfit,linewidth=1,color = "black")

plt.scatter(x_val,(df.ECS)**2)
plt.xlabel('S(0)$Q^2$/$\sigma^2$ [K$^2$]',fontsize=12)
plt.ylabel('ECS$^2$ [K$^2$]', fontsize=12)
plt.grid()
plt.savefig('ECS_SQs.eps')


x_val = (df.Q)**2
p2 = np.polyfit(x_val,(df.ECS)**2, deg = 1)

linfit = np.polyval(p2,x_val)
plt.plot(x_val,linfit,linewidth=1,color = "black")
corr = np.corrcoef(x_val,df.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(x_val,(df.ECS)**2)
plt.xlabel('Q^2')
plt.ylabel('ECS^2')
plt.title('Plot',fontsize = 14)
plt.grid()

x_val = 1/(df.sigma)**2
p2 = np.polyfit(x_val,(df.ECS)**2, deg = 1)

linfit = np.polyval(p2,x_val)
plt.plot(x_val,linfit,linewidth=1,color = "black")

plt.scatter(x_val,(df.ECS)**2)
plt.xlabel('1/sigma^2')
```

```
plt.ylabel('ECS^2')
plt.title('Plot',fontsize = 14)
plt.grid()
```

```
#Functions that will be used

import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from matplotlib import pyplot
import statsmodels.api as sm


def temp_plot(model):
    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
    realm = 'Amon'
    ensemble = 'r1i1p1'
    exp = 'historical'

    period = '185001-200512'

    if model == 'bcc-csm1-1' or model == 'bcc-csm1-1-m' or model ==
    'MIROC5':
        period = '185001-201212'
    elif model == 'GFDL-CM3':
        period = '186001-200512'
    elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
        period = '186101-200512'
    elif model == 'HadGEM2-ES':
        period = '186001-200511'

    var = 'ts'
    strings = [var, realm, model, exp, ensemble, period]
    filename = 'glannual_' + "_".join(strings) + '.txt'
    file = os.path.join(filedir1, model, exp, filename)
    datatable = pd.read_table(file, header=None,sep=" ")
    temp=datatable.iloc[:,0]

    return temp


def forcing_plot(model):

    if model=='ACCESS1-3' or model =='GISS-E2-H' or model=='IPSL-
    CM5B-LR' or model== 'MPI-ESM-P':
        try:
            print('No forcing data for this model. Model:' + model )

        except ValueError:
            print('No forcing')


    forcing = pd.read_excel('C://Users/Bruker/Desktop/
```

```
    nyjustertforcingmedrcp85.xls')
    forcing = forcing[model].values


    if model == 'GFDL-CM3' or model == 'GFDL-ESM2G' or model == '
    GFDL-ESM2M' or model == 'HadGEM2-ES':
        forcing = forcing[10:]
    fig, ax = plt.subplots(figsize = [6,4])
    plt.plot(forcing,linewidth=1,color = "red")
    ax.set_xlabel('Year (after 1850)',fontsize = 12)
    ax.set_ylabel('Forcing [W/m$^2$] ',fontsize = 12)
    ax.set_title('Change in forcing (' + str(model) +')' , fontsize
    = 15)
    ax.grid()
    ax.tick_params(axis='both',labelsize=12)
    return print(len(forcing))



def exp_4CO2_function(model, taulist):

    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'


    realm = 'Amon'
    ensemble = 'r1i1p1'

    if model == 'ACCESS1-0':
        abrupt4xco2timeperiod = '030001-044912'
        controltimeperiod = '030001-079912'
        control_branch_yr = 300
    elif model == 'ACCESS1-3':
        abrupt4xco2timeperiod = '025001-040012'
        controltimeperiod = '025001-074912'
        control_branch_yr = 250
    elif model == 'CanESM2':
        abrupt4xco2timeperiod = '185001-199912'
        controltimeperiod = '201501-301012'
        control_branch_yr = 2321
    elif model == 'CNRM-CM5':
        abrupt4xco2timeperiod = '185001-199912'
        controltimeperiod = '185001-269912'
        control_branch_yr = 1850
    elif model == 'CNRM-CM6-1':
        ensemble = 'r1i1p1f2'
        abrupt4xco2timeperiod = '185001-199912'
        controltimeperiod = '185001-234912'
        control_branch_yr = 1850
    elif model == 'CSIRO-Mk3-6-0':
        abrupt4xco2timeperiod = '000101-015012'
        controltimeperiod = '000101-050012'
        control_branch_yr = 104
    elif model == 'GFDL-CM3':
        abrupt4xco2timeperiod = '000101-015012'
```

```
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    abrupt4xco2timeperiod = '000101-030012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GISS-E2-H':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + 'has control run for two different periods')
    #controltimeperiod = '118001-141912'
    controltimeperiod = '241001-294912'
    control_branch_yr = 2660
elif model == 'GISS-E2-R':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + ' has two different control runs')
    #controltimeperiod = '333101-363012'
    controltimeperiod1 = '398101-453012'
    controltimeperiod2 = '398101-920512'
    control_branch_yr = 4200
elif model == 'HadGEM2-ES':
    abrupt4xco2timeperiod = '186001-201012'
    controltimeperiod = '186001-243511'
    control_branch_yr = 1860
elif model == 'inmcm4':
    abrupt4xco2timeperiod = '209001-223912'
    controltimeperiod = '185001-234912'
    control_branch_yr = 2090
elif model == 'IPSL-CM5A-LR':
    abrupt4xco2timeperiod = '185001-210912'
    controltimeperiod = '180001-279912'
    control_branch_yr = 1850
elif model == 'IPSL-CM5B-LR':
    abrupt4xco2timeperiod = '185001-200912'
    controltimeperiod = '183001-212912'
    control_branch_yr = 1850
elif model == 'MIROC-ESM':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '180001-242912'
    control_branch_yr = 1880
elif model == 'MIROC5':
    abrupt4xco2timeperiod = '210001-225012'
    controltimeperiod = '200001-286912'
    control_branch_yr = 2100
elif model == 'MPI-ESM-LR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1880
elif model == 'MPI-ESM-MR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1850
elif model == 'MPI-ESM-P':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-300512'
```

```
    control_branch_yr = 1866
elif model == 'MRI-CGCM3':
    abrupt4xco2timeperiod = '185101-200012'
    controltimeperiod = '185101-235012'
    control_branch_yr = 1891
elif model == 'NorESM1-M':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '070001-120012'
    control_branch_yr = 700


######  load abrupt4xco2 data ######
exp = 'abrupt4xco2'

var = 'ts' # temperatures
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]


if len(temp)>150:
    temp = temp[0:150]


###### load control run data ######
exp = 'piControl'

var = 'ts' # temperatures
if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod1

strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]

if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod2

years = np.arange(1,150+1)
controlyears = np.arange(0,len(controltemp))

branchindex = control_branch_yr - int(controltimeperiod[0:4])
#print(branchindex)

p1 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controltemp[branchindex:(branchindex + len(temp))], deg =
1)
lintrendT = np.polyval(p1,controlyears[branchindex:(branchindex
+ len(temp))])
```

```
    deltaT = temp - lintrendT

    # for deltaT we also have the information that deltaT(0) = 0.
    Include this:
    deltaT0 = np.concatenate(([0],deltaT))
    years0 = np.concatenate(([0],years))

    # compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is a
    convolution and F is a constant, so we can compute T_n
    analytically)
    dim = len(taulist)


    A = np.zeros((len(years)+1,dim))
    for i in range(0,dim): # compute the predictors in the linear
    model for deltaT
        A[:,i] = (1 - np.exp((-np.arange(0,151))/taulist[i]))

    # find parameters par1 in the linear model: deltaT = \sum_i par1
    [i]*(1 - np.exp((-t/tau[i]))
    #par1, resT, rankT, sT = np.linalg.lstsq(A, deltaT0,rcond=None)
    # least squares for deltaT
    par1, rnorm1 = optimize.nnls(A,deltaT0)
    Ti = np.array([A[:,i]*par1[i] for i in range(0,dim)]) # compute
    components
    Tsum = A@par1 # sum of all components


    # create figure
    fig, ax = plt.subplots(figsize = [9,5])

    var = deltaT0; label = '$\Delta T$'
    ax.plot(years0,var,linewidth=2,color = "black")
    ax.plot(years0,Tsum,linewidth=2,color = "black")
    for i in range(0,dim):
        ax.plot(years0,Ti[i,:],linewidth=2,color = "red")

    ax.set_xlabel('Years after 1850 ',fontsize = 11)
    ax.set_ylabel(label + '(t) ',fontsize = 11)
    ax.set_title('Abrupt4xco2 $\Delta T$, model: ' +str(model),
    fontsize = 18)
    ax.grid()
    ax.set_xlim(min(years),max(years))
    ax.tick_params(axis='both',labelsize=18)
    return par1, rnorm1, deltaT0, A, Tsum



def lintrend_func(model):
    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
    realm = 'Amon'
    ensemble = 'r1i1p1'
```

```
if model == 'ACCESS1-0':
    abrupt4xco2timeperiod = '030001-044912'
    controltimeperiod = '030001-079912'
    control_branch_yr = 300
elif model == 'ACCESS1-3':
    abrupt4xco2timeperiod = '025001-040012'
    controltimeperiod = '025001-074912'
    control_branch_yr = 250
elif  model == 'CanESM2':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '201501-301012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-230012' #rcp26, rcp45
    #rcp60 not available
    control_branch_yr = 2321
elif model == 'CNRM-CM5':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-269912'
    #rcptimeperiod = '200601-210012' #rcp26
    rcptimeperiod = '200601-230012' #rcp45, rcp85
    #rcp60 not available
    control_branch_yr = 2250
elif model == 'CSIRO-Mk3-6-0':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 81
elif model == 'GFDL-CM3':
    historicaltimeperiod = '186001-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 1
elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    historicaltimeperiod = '186101-200512'
    controltimeperiod = '000101-050012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 162
elif model == 'GISS-E2-H':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + 'has control run for two different periods')
    #controltimeperiod = '118001-141912'
    controltimeperiod = '241001-294912'
    control_branch_yr = 2660
elif model == 'GISS-E2-R':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + ' has two different control runs')
    #controltimeperiod = '333101-363012'
    controltimeperiod1 = '398101-453012'
    controltimeperiod2 = '398101-920512'
    control_branch_yr = 4200 #

elif model == 'HadGEM2-ES':
    historicaltimeperiod = '186001-200511'
```

```
    controltimeperiod = '186001-243511'
    rcptimeperiod = '200601-229912' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-209911' #rcp60
    control_branch_yr = 1860
elif model == 'inmcm4':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-234912'
    rcptimeperiod = '200601-210012' # rcp45, rcp85
    #rcp26 and rcp60 not available
    control_branch_yr = 1850
elif model == 'IPSL-CM5A-LR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '180001-279912'
    rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-210012' #rcp60
    control_branch_yr = 1850
elif model == 'IPSL-CM5B-LR':
    abrupt4xco2timeperiod = '185001-200912'
    controltimeperiod = '183001-212912'
    control_branch_yr = 1850
elif model == 'MIROC-ESM':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '180001-242912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    #rcptimeperiod = '200601-230012' #rcp45
    control_branch_yr = 1880
elif model == 'MIROC5':
    historicaltimeperiod = '185001-201212'
    controltimeperiod = '200001-286912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 2411
elif model == 'MPI-ESM-LR':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-284912'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp85
    #rcptimeperiod = '200601-230012' #rcp26, rcp45, rcp85
    # rcp60 not available
    control_branch_yr = 1880
elif model == 'MPI-ESM-P':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185001-300512'
    print(model + 'has no rcp runs')
    control_branch_yr = NaN
elif model == 'MPI-ESM-MR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1850
elif model == 'MRI-CGCM3':
    historicaltimeperiod = '185001-200512'
    controltimeperiod = '185101-235012'
    rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
    control_branch_yr = 1950
elif model == 'NorESM1-M':
    historicaltimeperiod = '185001-200512'
```

```
        controltimeperiod = '070001-120012'
        rcptimeperiod = '200601-210012' #rcp26, rcp45, rcp60, rcp85
        #rcptimeperiod = '200601-230012' #rcp45
        control_branch_yr = 700

    exp = 'piControl'
    var = 'ts'

    if model == 'GISS-E2-R':
        controltimeperiod = controltimeperiod1

    strings = [var, realm, model, exp, ensemble, controltimeperiod]
    filename = 'glannual_' + "_".join(strings) + '.txt'
    file = os.path.join(filedir1, model, exp, filename)
    datatable = pd.read_table(file, header=None,sep=" ")
    controltemp=datatable.iloc[:,0]

    controlyears = np.arange(0,len(controltemp))

    p1 = np.polyfit(controlyears, controltemp, deg = 1)
    lintrendT = np.polyval(p1, controlyears)

    #plt.plot(controlyears, controltemp)
    #plt.plot(controlyears, lintrendT[0:len(controlyears)])

    return p1, lintrendT


def gregory_func(model,n):

    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
    realm = 'Amon'
    ensemble = 'r1i1p1'

    if model == 'ACCESS1-0':
        abrupt4xco2timeperiod = '030001-044912'
        controltimeperiod = '030001-079912'
        control_branch_yr = 300
    elif model == 'ACCESS1-3':
        abrupt4xco2timeperiod = '025001-040012'
        controltimeperiod = '025001-074912'
        control_branch_yr = 250
    elif  model == 'CanESM2':
        abrupt4xco2timeperiod = '185001-199912'
        controltimeperiod = '201501-301012'
        control_branch_yr = 2321
    elif model == 'CNRM-CM5':
        abrupt4xco2timeperiod = '185001-199912'
        controltimeperiod = '185001-269912'
        control_branch_yr = 1850
    elif model == 'CSIRO-Mk3-6-0':
        abrupt4xco2timeperiod = '000101-015012'
        controltimeperiod = '000101-050012'
        control_branch_yr = 104
```

```
elif model == 'GFDL-CM3':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    abrupt4xco2timeperiod = '000101-030012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GISS-E2-H':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + 'has control run for two different periods')
    #controltimeperiod = '118001-141912'
    controltimeperiod = '241001-294912'
    control_branch_yr = 2660
elif model == 'GISS-E2-R':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + ' has two different control runs')
    #controltimeperiod = '333101-363012'
    controltimeperiod1 = '398101-453012'
    controltimeperiod2 = '398101-920512'
    control_branch_yr = 4200
elif model == 'HadGEM2-ES':
    abrupt4xco2timeperiod = '186001-201012'
    controltimeperiod = '186001-243511'
    control_branch_yr = 1860
elif model == 'inmcm4':
    abrupt4xco2timeperiod = '209001-223912'
    controltimeperiod = '185001-234912'
    control_branch_yr = 2090
elif model == 'IPSL-CM5A-LR':
    abrupt4xco2timeperiod = '185001-210912'
    controltimeperiod = '180001-279912'
    control_branch_yr = 1850
elif model == 'IPSL-CM5B-LR':
    abrupt4xco2timeperiod = '185001-200912'
    controltimeperiod = '183001-212912'
    control_branch_yr = 1850
elif model == 'MIROC-ESM':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '180001-242912'
    control_branch_yr = 1880
elif model == 'MIROC5':
    abrupt4xco2timeperiod = '210001-225012'
    controltimeperiod = '200001-286912'
    control_branch_yr = 2100
elif model == 'MPI-ESM-LR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1880
elif model == 'MPI-ESM-P':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-300512'
    control_branch_yr = 1866
elif model == 'MRI-CGCM3':
```

```
    abrupt4xco2timeperiod = '185101-200012'
    controltimeperiod = '185101-235012'
    control_branch_yr = 1891
elif model == 'MPI-ESM-MR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1850
elif model == 'NorESM1-M':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '070001-120012'
    control_branch_yr = 700

exp = 'abrupt4xco2'

var = 'ts' # temperatures
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]

var = 'rlut' # rlut
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rlut=datatable.iloc[:,0]

var = 'rsut' # rsut
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rsut=datatable.iloc[:,0]

var = 'rsdt' # rsdt
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rsdt=datatable.iloc[:,0]

if len(temp)>150:
    temp = temp[0:150]
    rlut = rlut[0:150]
    rsut = rsut[0:150]
    rsdt = rsdt[0:150]


###### load control run data ######
```

```
exp = 'piControl'

var = 'ts' # temperatures

if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod1

strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]


if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod2

var = 'rlut' # rlut
strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrlut=datatable.iloc[0:len(controltemp),0]

var = 'rsut' # rsut
strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrsut=datatable.iloc[0:len(controltemp),0]

var = 'rsdt' # rsdt
strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrsdt=datatable.iloc[0:len(controltemp),0]

controlyears = np.arange(0,len(controltemp))

nettoarad = rsdt - rsut - rlut
controlnettoarad = controlrsdt - controlrsut - controlrlut

branchindex = control_branch_yr - int(controltimeperiod[0:4])
#print(branchindex)


p1 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controltemp[branchindex:(branchindex + len(temp))], deg =
1)
lintrendT = np.polyval(p1,controlyears[branchindex:(branchindex
+ len(temp))])

p2 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
```

```
    ))], controlnettoarad[branchindex:(branchindex + len(temp))],
    deg = 1)
    lintrendN = np.polyval(p2,controlyears[branchindex:(branchindex
    + len(temp))])

    deltaT = temp - lintrendT
    deltaN = nettoarad - lintrendN

    # Gregory plot

    fig, ax = plt.subplots(figsize = [9,5])
    plt.scatter(deltaT,deltaN,linewidth=1,color = "lavender")
    ax.set_xlabel('$\Delta$T [K]',fontsize = 12)
    ax.set_ylabel('$\Delta$N [W/m$^2$]',fontsize = 12)
    ax.set_title('Gregory plot for model:' +str(model),fontsize =
    16)
    ax.grid()
    ax.tick_params(axis='both',labelsize=22)

    # find linear fit to these points:
    p1 = np.polyfit(deltaT,deltaN, deg = 1)

    linfit = np.polyval(p1,deltaT)
    ax.plot(deltaT,linfit,linewidth=1,color = "r")

    p2 = np.polyfit(deltaT[0:n], deltaN[0:n], deg = 1)
    linfit2 = np.polyval(p2,deltaT[0:n])
    ax.plot(deltaT[0:n],linfit2,linewidth=1,color = "black")

    ax.set_ylim(-1,p1[1])
    fig.savefig(model+'-gregory.png')

    print('ECS_'+str(n)+' = ' + str(-p2[1]/(2*p2[0])))
    print('ECS = ' + str(-p1[1]/(2*p1[0])))

    ECS = (-p1[1]/(2*p1[0]))
    ECS_20 = (-p2[1]/(2*p2[0]))
    Feedback = p1[0]
    Feedback_20 = p2[0]
    Forcing = p1[1]/2
    Forcing_20 = p2[1]/2


    return ECS, ECS_20, Feedback, Feedback_20, Forcing, Forcing_20




def ECS_sum3(model, amplitudes):

    ds = pd.read_csv('gregory.csv', index_col = 0)

    tau_list =  np.array([0.7, 9, 354])
```

```
    forcing = ds.forcing[model]
    forcing_20 = ds.forcing_20[model]

    ECS = forcing*sum(amplitudes*tau_list)

    ECS_20 = forcing_20*sum(amplitudes*tau_list)

    return ECS, ECS_20




def ECS_sum_2(model, amplitudes):

    ds = pd.read_csv('gregory.csv', index_col = 0)

    tau_list =  np.array([0.7, 9])
    forcing = ds.forcing[model]
    forcing_20 = ds.forcing_20[model]

    ECS = forcing*sum(amplitudes*tau_list)
    ECS_20 = forcing_20*sum(amplitudes*tau_list)

    return ECS, ECS_20



def spec(freq, par1, par2):

    #par1 = amplitudes[0]
    #par2 = amplitudes[1]
    w1 = 1/taulist[0]
    w2 = 1/taulist[1]
    omega = 2*np.pi*freq

    spec_val = (par1**2/(w1**2 + omega**2))+(par2**2/(w2**2+omega
    **2))

    return spec_val




def psd_par_est(res, amplitudes, taulist):

    f, S = signal.welch(res)
    #f, S = signal.welch(res, nperseg=len(res), noverlap= len(res) /
     2)
    par1 = amplitudes[0]
```

```
    par2 = amplitudes[1]
    w1 = 1/taulist[0]
    w2 = 1/taulist[1]

    S1 = par1**2/(w1**2 + (2*np.pi*f)**2)
    S2 = par2**2/(w2**2 + (2*np.pi*f)**2)

    plt.plot(f,S, label = 'Residuals')
    plt.plot(f, S2, label = 'Second term in sum')
    plt.plot(f, S1, label= 'First term in sum')
    plt.plot(f, spec(f, par1, par2), label = 'The sum')
    plt.title('Plot for model: ' + str(model))
    pyplot.yscale('log')
    pyplot.xscale('log')
    plt.ylim(0.0001,0.5)
    plt.legend()
    plt.grid()

    def func(f, a, b):
        w1 = 1/taulist[0]
        w2 = 1/taulist[1]
        omega = 2*np.pi*f
        return (a**2/(w1**2 + omega**2))+(b**2/(w2**2+omega**2))


    popt, pcov = curve_fit(func, f, S)
    est_par1= popt[0]
    est_par2 = popt[1]

    return est_par1, est_par2, spec(f, par1, par2)



def psd_func(res, taulist):

    f, S = signal.welch(res)
    #f, S = signal.welch(res, nperseg=len(res), noverlap= len(res) /
     2)
    w1 = 1/taulist[0]
    w2 = 1/taulist[1]

    def func(f, a, b):
        w1 = 1/taulist[0]
        w2 = 1/taulist[1]
        omega = 2*np.pi*f
        return (a**2/(w1**2 + omega**2))+(b**2/(w2**2+omega**2))

    popt, pcov = curve_fit(func, f, S)
    a= popt[0]
    b= popt[1]

    S1 = a**2/(w1**2 + (2*np.pi*f)**2)
    S2 = b**2/(w2**2 + (2*np.pi*f)**2)
```

```
    S_sum = a**2/(w1**2 + (2*np.pi*f)**2) + b**2/(w2**2 + (2*np.pi*f
    )**2)

    #plt.plot(f,S, label = 'Residuals')
    #plt.plot(f, S2, label = 'Second term in sum')
    #plt.plot(f, S1, label= 'First term in sum')
    #plt.plot(f, S, label = 'The sum', linestyle= 'dashed')
    #plt.title('Plot for model: ' + str(model))
    #pyplot.yscale('log')
    #pyplot.xscale('log')
    #plt.ylim(0.0001,0.5)
    #plt.legend()
    #plt.grid()
    #plt.savefig(model+'-psd.png')


    return S_sum



def psd_plotfunc(res, taulist):

    f, S = signal.welch(res)
    #f, S = signal.welch(res, nperseg=len(res), noverlap= len(res) /
     2)
    w1 = 1/taulist[0]
    w2 = 1/taulist[1]

    def func(f, a, b):
        w1 = 1/taulist[0]
        w2 = 1/taulist[1]
        omega = 2*np.pi*f
        return (a**2/(w1**2 + omega**2))+(b**2/(w2**2+omega**2))

    popt, pcov = curve_fit(func, f, S)
    a= popt[0]
    b= popt[1]

    S1 = a**2/(w1**2 + (2*np.pi*f)**2)
    S2 = b**2/(w2**2 + (2*np.pi*f)**2)
    S_sum = a**2/(w1**2 + (2*np.pi*f)**2) + b**2/(w2**2 + (2*np.pi*f
    )**2)

    plt.plot(f,S, label = 'Residuals')
    plt.plot(f, S2, label = 'Second term in sum')
    plt.plot(f, S1, label= 'First term in sum')
    plt.plot(f, S_sum, label = 'The sum', linestyle= 'dashed')
    pyplot.yscale('log')
    pyplot.xscale('log')
    plt.legend()
    plt.grid()
```

```
    return S_sum, S
```

```
import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from matplotlib import pyplot
import statsmodels.api as sm


def temp_plot(model):
    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
    realm = 'Amon'
    ensemble = 'r1i1p1'
    exp = 'historical'

    period = '185001-200512'

    if model == 'bcc-csm1-1' or model == 'bcc-csm1-1-m' or model ==
    'MIROC5':
        period = '185001-201212'
    elif model == 'GFDL-CM3':
        period = '186001-200512'
    elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
        period = '186101-200512'
    elif model == 'HadGEM2-ES':
        period = '186001-200511'

    var = 'ts'
    strings = [var, realm, model, exp, ensemble, period]
    filename = 'glannual_' + "_".join(strings) + '.txt'
    file = os.path.join(filedir1, model, exp, filename)
    datatable = pd.read_table(file, header=None,sep=" ")
    temp=datatable.iloc[:,0]

    fig, ax = plt.subplots(figsize = [9,5])
    plt.plot(temp,linewidth=1,color = "red")
    ax.set_xlabel('Years (after 1850)',fontsize = 14)
    ax.set_ylabel('Temperature [K]',fontsize = 14)
    ax.set_title('Change in surface temperature, model: ' + str(
    model) , fontsize = 15)
    ax.grid()
    ax.tick_params(axis='both',labelsize=12)

    fig.savefig(model+'-temp.png')
    return temp


def exp_4CO2_function(model, taulist):

    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
```

```
realm = 'Amon'
ensemble = 'r1i1p1'

if model == 'ACCESS1-0':
    abrupt4xco2timeperiod = '030001-044912'
    controltimeperiod = '030001-079912'
    control_branch_yr = 300
elif model == 'ACCESS1-3':
    abrupt4xco2timeperiod = '025001-040012'
    controltimeperiod = '025001-074912'
    control_branch_yr = 250
elif model == 'CanESM2':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '201501-301012'
    control_branch_yr = 2321
elif model == 'CNRM-CM5':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-269912'
    control_branch_yr = 1850
elif model == 'CNRM-CM6-1':
    ensemble = 'r1i1p1f2'
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-234912'
    control_branch_yr = 1850
elif model == 'CSIRO-Mk3-6-0':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 104
elif model == 'GFDL-CM3':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    abrupt4xco2timeperiod = '000101-030012'
    controltimeperiod = '000101-050012'
    control_branch_yr = 1
elif model == 'GISS-E2-H':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + 'has control run for two different periods')
    #controltimeperiod = '118001-141912'
    controltimeperiod = '241001-294912'
    control_branch_yr = 2660
elif model == 'GISS-E2-R':
    abrupt4xco2timeperiod = '185001-200012'
    print(model + ' has two different control runs')
    #controltimeperiod = '333101-363012'
    controltimeperiod1 = '398101-453012'
    controltimeperiod2 = '398101-920512'
    control_branch_yr = 4200
elif model == 'HadGEM2-ES':
    abrupt4xco2timeperiod = '186001-201012'
    controltimeperiod = '186001-243511'
```

```
    control_branch_yr = 1860
elif model == 'inmcm4':
    abrupt4xco2timeperiod = '209001-223912'
    controltimeperiod = '185001-234912'
    control_branch_yr = 2090
elif model == 'IPSL-CM5A-LR':
    abrupt4xco2timeperiod = '185001-210912'
    controltimeperiod = '180001-279912'
    control_branch_yr = 1850
elif model == 'IPSL-CM5B-LR':
    abrupt4xco2timeperiod = '185001-200912'
    controltimeperiod = '183001-212912'
    control_branch_yr = 1850
elif model == 'MIROC-ESM':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '180001-242912'
    control_branch_yr = 1880
elif model == 'MIROC5':
    abrupt4xco2timeperiod = '210001-225012'
    controltimeperiod = '200001-286912'
    control_branch_yr = 2100
elif model == 'MPI-ESM-LR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1880
elif model == 'MPI-ESM-MR':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-284912'
    control_branch_yr = 1850
elif model == 'MPI-ESM-P':
    abrupt4xco2timeperiod = '185001-199912'
    controltimeperiod = '185001-300512'
    control_branch_yr = 1866
elif model == 'MRI-CGCM3':
    abrupt4xco2timeperiod = '185101-200012'
    controltimeperiod = '185101-235012'
    control_branch_yr = 1891
elif model == 'NorESM1-M':
    abrupt4xco2timeperiod = '000101-015012'
    controltimeperiod = '070001-120012'
    control_branch_yr = 700


######  load abrupt4xco2 data ######
exp = 'abrupt4xco2'

var = 'ts' # temperatures
strings = [var, realm, model, exp, ensemble,
abrupt4xco2timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]
```

```
if len(temp)>150:
    temp = temp[0:150]


###### load control run data ######
exp = 'piControl'

var = 'ts' # temperatures
if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod1

strings = [var, realm, model, exp, ensemble, controltimeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]

if model == 'GISS-E2-R':
    controltimeperiod = controltimeperiod2

years = np.arange(1,150+1)
controlyears = np.arange(0,len(controltemp))


branchindex = control_branch_yr - int(controltimeperiod[0:4])
print(branchindex)

p1 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controltemp[branchindex:(branchindex + len(temp))], deg =
1)
lintrendT = np.polyval(p1,controlyears[branchindex:(branchindex
+ len(temp))])

deltaT = temp - lintrendT


# for deltaT we also have the information that deltaT(0) = 0.
Include this:
deltaT0 = np.concatenate(([0],deltaT))
years0 = np.concatenate(([0],years))


# compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is a
convolution and F is a constant, so we can compute T_n
analytically)
dim = len(taulist)


A = np.zeros((len(years)+1,dim))
for i in range(0,dim): # compute the predictors in the linear
model for deltaT
    A[:,i] = (1 - np.exp((-np.arange(0,151))/taulist[i]))
```

```
    # find parameters par1 in the linear model: deltaT = \sum_i par1
    [i]*(1 - np.exp((-t/tau[i]))
    #par1, resT, rankT, sT = np.linalg.lstsq(A, deltaT0,rcond=None)
    # least squares for deltaT
    par1, rnorm1 = optimize.nnls(A,deltaT0)
    Ti = np.array([A[:,i]*par1[i] for i in range(0,dim)]) # compute
    components
    Tsum = A@par1 # sum of all components

    # create figure
    fig, ax = plt.subplots(figsize = [9,5])

    var = deltaT0; label = '$\Delta T$'
    ax.plot(years0,var,linewidth=2,color = "black")
    ax.plot(years0,Tsum,linewidth=2,color = "black")
    for i in range(0,dim):
        ax.plot(years0,Ti[i,:],linewidth=2,color = "red")

    ax.set_xlabel('Year after 1850 ',fontsize = 15)
    ax.set_ylabel(label + '(t) [K]',fontsize = 15)
    ax.set_title('Abrupt4xCO$_2$, model: ' +str(model),fontsize =
    18)
    ax.grid()
    ax.set_xlim(min(years),max(years))
    ax.tick_params(axis='both',labelsize=18)
    fig.savefig(model+'-fit_4CO2.png')


    return par1, rnorm1, deltaT0, A, Tsum


def psd_func(res, taulist):

    f, S = signal.welch(res)
    #f, S = signal.welch(res, nperseg=len(res), noverlap= len(res) /
     2)
    w1 = 1/taulist[0]
    w2 = 1/taulist[1]

    def func(f, a, b):
        w1 = 1/taulist[0]
        w2 = 1/taulist[1]
        omega = 2*np.pi*f
        return (a**2/(w1**2 + omega**2))+(b**2/(w2**2+omega**2))

    popt, pcov = curve_fit(func, f, S)
    a= popt[0]
    b= popt[1]

    S1 = a**2/(w1**2 + (2*np.pi*f)**2)
    S2 = b**2/(w2**2 + (2*np.pi*f)**2)
    spect = a**2/(w1**2 + (2*np.pi*f)**2) + b**2/(w2**2 + (2*np.pi*f
    )**2)
```

```
    plt.plot(f,S, label = 'Residuals')
    #plt.plot(f, S2, label = 'Second term in sum')
    #plt.plot(f, S1, label= 'First term in sum')
    plt.plot(f, spect, label = 'Adapted fit')
    plt.title('Plot for model: ' + str(model), fontsize=14)
    plt.xlabel('Frequency [Hz]', fontsize=12)
    plt.ylabel('Power spectral density (PSD)', fontsize=12)
    pyplot.yscale('log')
    pyplot.xscale('log')
    plt.ylim(0.0001,0.5)
    plt.legend()
    plt.grid()
    #plt.savefig(model+'-psd.png')


    return a, b, spect, S

import ipynb.fs.full.Functions as eom


#models included in nyjustertforcingmedrcp85 file and gregory file

#model = 'ACCESS1-0'
#model = 'CanESM2'
#model = 'CNRM-CM5'
#model = 'CSIRO-Mk3-6-0'
#model = 'GFDL-CM3'
#model = 'GFDL-ESM2G'
#model = 'GFDL-ESM2M'
#model = 'GISS-E2-R'
#model = 'HadGEM2-ES'
#model = 'inmcm4'
#model = 'IPSL-CM5A-LR'
#model = 'MIROC5'
#model = 'MIROC-ESM'
#model = 'MPI-ESM-LR'
#model = 'MRI-CGCM3'
model = 'NorESM1-M'


ECS, ECS_20, fb, fb_20, F, F_20 = eom.gregory_func(model, 20)


#ds = pd.read_csv('gregory.csv', index_col = 0)
#ds.ECS[model] = ECS
#ds.ECS_20[model] = ECS_20
#ds.forcing[model] = F
#ds.forcing_20[model] = F_20
#ds.feedback_par[model] = fb
#ds.feedback_par_20[model] = fb_20
#ds.to_csv('gregory.csv')
```

```
histemp =eom.temp_plot(model)

eom.forcing_plot(model)


taulist = np.array([0.7, 9, 354]) #parameter

#Try to find my own fit to deltaT(t) for a fixed set of time scales:
par_4CO2, rnorm, deltaT, A, Tsum = eom.exp_4CO2_function(model,
    taulist)
#(from this, the table parameter is made (in file
    linresponse_estimation), file=parameter)

res_4CO2 = Tsum - deltaT


a_4CO2, b_4CO2, spec_4CO2, S_4CO2 = psd_func(res_4CO2, taulist)
#plt.savefig(model+'-psd_4CO2.png')

SS4_met1 = pd.read_csv('S0_values_4CO2_met1.csv', index_col=0)
#SS4_met1.S0[model] = np.mean(S_4CO2[0:25])
#SS4_met1.to_csv('S0_values_4CO2_met1.csv')

spec_4CO2
SS4_met1

SS4 = pd.read_csv('S0_values_4CO2.csv', index_col=0)
#SS4.S0[model] = spec_4CO2[0]
#SS4.to_csv('S0_values_4CO2.csv')
SS4

ds = pd.read_csv('gregory_short.csv', index_col = 0)
p_1 = np.polyfit(SS4.S0,ds.ECS**2, deg = 1)
linfit_1 = np.polyval(p_1,SS4.S0)
corr = np.corrcoef(SS4.S0,ds.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(SS4, ds.ECS**2, color='darkred')
plt.plot(SS4, linfit_1 , color='black')
plt.ylabel('Gregory estimates of ECS$^2$ [K$^2$]', fontsize=12)
plt.xlabel('S(0) (4xCO2)', fontsize=12)
plt.savefig('S0_ECS_correlation_4CO2')


print('Correlation: ' +str(np.corrcoef(ds.ECS**2, SS4.S0)[0,1]))

p_1 = np.polyfit(SS4_met1.S0,ds.ECS**2, deg = 1)
linfit_1 = np.polyval(p_1,SS4_met1.S0)
corr = np.corrcoef(SS4_met1.S0,ds.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(SS4_met1, ds.ECS**2, color='darkred')
plt.plot(SS4_met1, linfit_1 , color='black')
```

```
plt.ylabel('Gregory estimates of ECS$^2$ [K$^2$]', fontsize=12)
plt.xlabel('S(0) (4xCO2)', fontsize=12)
plt.savefig('S0_ECS_correlation_4CO2_met1')

dr = pd.read_csv('parameter.csv', index_col = 0)
dp = pd.read_csv('gregory.csv', index_col = 0)

parameterlist = (dr.par1[model]/2, dr.par2[model]/2, dr.par3[model
    ]/2)
forcing = pd.read_excel('C://Users/Bruker/Desktop/
    nyjustertforcingmedrcp85.xls')
forcing = forcing[model].values
if model == 'GFDL-CM3' or model == 'GFDL-ESM2G' or model == 'GFDL-
    ESM2M' or model == 'HadGEM2-ES':
        forcing = forcing[10:]

F2x = dp.forcing[model]
amplitudes = parameterlist/(taulist*F2x)


# compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is a
    convolution)
dim = len(taulist)
lf = len(forcing)
predictors = np.full((lf,dim),np.nan)

# compute exact predictors by integrating greens function
for k in range(0,dim):
    intgreensti = np.full((lf,lf),0.)   # remember dot after 0 to
    create floating point number array instead of integer
    for t in range(0,lf):
        # compute one new contribution to the matrix:
        intgreensti[t,0] = taulist[k]*(np.exp(-t/taulist[k]) - np.
    exp(-(t+1)/taulist[k]))

        # take the rest from row above:
        if t > 0:
            intgreensti[t,1:(t+1)] = intgreensti[t-1,0:t]
    # compute discretized convolution integral by this matrix
    product:
    predictors[:,k] = intgreensti@np.array(forcing)


Y = histemp
X = predictors[0:len(histemp)]
X = sm.add_constant(X)

model1, rnorm1 = optimize.nnls(X,Y)
#model1 = sm.OLS(Y,X)
b_tmp = model1
b = np.array([b_tmp[0],b_tmp[1],b_tmp[2],b_tmp[3]])

Tn = b[1:]*predictors
```

```
residual = Y - X@b
T_hege = amplitudes*predictors


#Plot temperature responses to forcing

years = np.arange(0,len(Tn))
fig, ax = plt.subplots(figsize = [9,5])
plt.plot(years,Tn[:,0],linewidth=2,color = "black",label = 'Mode
    with time scale ' + str(taulist[0]) + ' years')
plt.plot(years,Tn[:,1],linewidth=2,color = "blue",label = 'Mode with
     time scale ' + str(taulist[1]) + ' years')
plt.plot(years,Tn[:,2],linewidth=2,color = "red",label = 'Mode with
    time scale ' + str(taulist[2]) + ' years')
ax.set_xlabel('Time [yr]',fontsize = 14)
ax.set_ylabel('Temperature [K]',fontsize = 14)
ax.set_title('Temperature responses to forcing ('+ str(model) +')',
    fontsize = 16)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18})
#fig.savefig(model+'-response.png')

# Plot the estimated temp with historical

p1, lin = eom.lintrend_func(model)


period = '185001-200512'

if model == 'bcc-csm1-1' or model == 'bcc-csm1-1-m' or model == '
    MIROC5':
    period = '185001-201212'

elif model == 'GFDL-CM3':
    period = '186001-200512'

elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
    period = '186101-200512'

elif model == 'HadGEM2-ES':
    period = '186001-200511'

filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
strings = ['ts', 'Amon', model, 'historical', 'r1i1p1', period]
filename = 'glannual_' + "_".join(strings) + '.txt'

file = os.path.join(filedir1, model, 'historical', filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]

deltaT = temp - lin[0:len(temp)]
Tn_1 = Tn[0:len(temp)]
```

```
Tn_156 = np.sum(Tn_1,axis=1)

years_1 = np.arange(0,len(deltaT))

fig, ax = plt.subplots(figsize = [9,5])
ax.plot(years_1,Tn_156,linewidth=2,color = "black",label = 'linear
    response to adjusted forcing')
ax.plot(years_1,deltaT,linewidth=2,color = "red", label = 'modelled
    temperature')
ax.set_xlabel('t',fontsize = 18)
ax.set_ylabel('T(t)',fontsize = 18)
ax.set_title('Temperature response to forcing, model: ' +str(model),
    fontsize = 18)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18});


Tn_mean= np.mean(Tn_156[0:20])
deltaT_mean = np.mean(deltaT[0:20])
print(Tn_mean)
print(deltaT_mean)
factor = Tn_mean - deltaT_mean


new_deltaT = factor+deltaT
new_Tn = Tn_156

print(factor)


fig, ax = plt.subplots(figsize = [9,5])
ax.plot(years_1,new_Tn,linewidth=2,color = "black",label = 'linear
    response to adjusted forcing')
ax.plot(years_1,new_deltaT,linewidth=2,color = "red", label = '
    modelled temperature')
ax.set_xlabel('t',fontsize = 18)
ax.set_ylabel('T(t)',fontsize = 18)
ax.set_title('Temperature response to forcing, model: ' +str(model),
    fontsize = 18)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18});
#fig.savefig(model+'-linearresponsetoforcing.png')

#Residuals

res = Tn_156 - deltaT
mean_res = np.mean(res)

res_justert = new_Tn - new_deltaT
```

```
print(mean_res)
print(np.mean(res_justert))


fig, ax = plt.subplots(figsize = [9,5])

ax.plot(res, label='residual')
ax.hlines(y=0, xmin=0, xmax=156, linewidth=1, color='black',
    linestyle ='--', label='x=0')
ax.hlines(y=mean_res, xmin=0, xmax=156, linewidth=1, color='red',
    linestyle ='--', label = 'mean')
ax.set_title('Residuals', fontsize= 18)
print('The mean :',  mean_res)
print('Maximum value :', max(abs(res)))
print('Minimum value :', min(abs(res)))
ax.legend()
plt.show()


print(mean_res)
print(np.mean(res_justert))


fig, ax = plt.subplots(figsize = [9,5])

ax.plot(res_hege, label='residual')
ax.hlines(y=0, xmin=0, xmax=156, linewidth=1, color='black',
    linestyle ='--', label='x=0')
ax.hlines(y=mean_res, xmin=0, xmax=156, linewidth=1, color='red',
    linestyle ='--', label = 'mean')
ax.set_title('Residuals', fontsize= 18)
print('The mean :',  mean_res)
print('Maximum value :', max(abs(res)))
print('Minimum value :', min(abs(res)))
ax.legend()
plt.show()


#da = pd.read_excel('res_justert.xlsx')
#da[model]=res_justert
#da.to_excel('res_justert.xlsx')
#da

#dd = pd.read_excel('res.xlsx', index_col =0)
#dd[model]=res
#dd.to_excel('res.xlsx')
#dd

#estimating parameters and making the fit to residuals
a, b, spect_hist, S_hist = psd_func(res, taulist)
#plt.savefig(model+'-psd_historical.png')

SSh_met1 = pd.read_csv('S0_values_hist_met1.csv', index_col=0)
#SSh_met1.S0[model] = np.mean(S_hist[0:25])
```

```
#SSh_met1.to_csv('S0_values_hist_met1.csv')
SSh_met1

ds = pd.read_csv('gregory_short.csv', index_col = 0)
SSh = pd.read_csv('S0_values_hist.csv', index_col=0)
#SSh.S0[model] = spect_hist[0]
#SSh.to_csv('S0_values_hist.csv')
SSh

p_2 = np.polyfit(SSh.S0,ds.ECS**2, deg = 1)
linfit_2 = np.polyval(p_2,SSh.S0)
corr = np.corrcoef(SSh.S0,ds.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(SSh, ds.ECS**2, color='darkred')
plt.plot(SSh, linfit_2 , color='black')
plt.ylabel('Gregory estimates of ECS$^2$ [K$^2$]', fontsize=12)
plt.xlabel('S(0) (historical)', fontsize=12)
plt.savefig('S0_ECS_correlation_historical')

print('Correlation: ' +str(np.corrcoef(ds.ECS**2, SSh.S0)[0,1]))


p_2 = np.polyfit(SSh_met1.S0,ds.ECS**2, deg = 1)
linfit_2 = np.polyval(p_2,SSh_met1.S0)
corr = np.corrcoef(SSh_met1.S0,ds.ECS**2)[1,0]
corr = round(corr, 5)

plt.scatter(SSh_met1, ds.ECS**2, color='darkred')
plt.plot(SSh_met1, linfit_2 , color='black')
plt.ylabel('Gregory estimates of ECS$^2$  [K$^2$]', fontsize=12)
plt.xlabel('S(0) (historical)', fontsize=12)
plt.savefig('S0_ECS_correlation_historical_met1')


dm = pd.read_csv('parameterestimation.csv', index_col = 0)

dm.par1[model] = amplitudes[0]
dm.par2[model] =  amplitudes[1]
dm.a[model] = a
dm.b[model] = b
dm.diff1[model] = amplitudes[0] - a
dm.diff2[model] =  amplitudes[1]- b

dm.to_csv('parameterestimation.csv')


dk = pd.read_csv('ECS_sum.csv', index_col = 0)
#dk.ECS_sum3[model] = ECS_sum
#dk.ECS20_sum3[model] = ECS20_sum
#dk.ECS_sum2[model] = ECS_sum2
#dk.ECS20_sum2[model] = ECS20_sum2

#dk.to_csv('ECS_sum.csv')
```

```
x = np.linspace(0,5, num=len(dk.ECS_sum3))
y = p1[1]+x*p1[0]

p1 = np.polyfit(dk.ECS_sum3,dk.ECS_sum2, deg = 1)
linfit1 = np.polyval(p1,dk.ECS_sum3)
corr_ECS = np.corrcoef(dk.ECS_sum3, dk.ECS_sum2)[1,0]
corr_ECS = round(corr_ECS, 5)

fig, ax = plt.subplots(figsize = [9,5])
ax.scatter(dk.ECS_sum3, dk.ECS_sum2)
ax.plot(dk.ECS_sum3, linfit1 ,linewidth=1, color = 'r')

ax.set_xlabel('ECS$_3$ [K]',fontsize = 16)
ax.set_ylabel('ECS$_2$ [K]',fontsize = 16)
#ax.set_title('',fontsize = 18)
plt.savefig('ECS_sum_correlation')
plt.show()


dy = pd.read_csv('gregory_short.csv', index_col = 0) #short becasue
    there are som models not included compared to gregory.csv file
p2 = np.polyfit(dk.ECS_sum3,dy.ECS, deg = 1)
linfit2 = np.polyval(p2,dk.ECS_sum3)

fig, ax = plt.subplots(figsize = [9,5])
ax.scatter(dk.ECS_sum3,dy.ECS)
ax.plot(dk.ECS_sum3, linfit2 ,linewidth=1, color = 'r')
ax.plot(dk.ECS_sum3, dk.ECS_sum3)

ax.set_xlabel('ECS_sum3',fontsize = 18)
ax.set_ylabel('ECS (gregory)',fontsize = 18)
ax.set_ylim(1.5,5)
ax.set_xlim(1.5,5)
plt.show()

#hat_ECS = dy.forcing[model]*(taulist[0]*(dm.a[model]/sigma[model])
    + taulist[1]*(np.abs(dm.b[model])/sigma[model]))
#hat_ECS


dd = pd.read_csv('hat_ECS.csv', index_col = 0)
#dd.hat_ECS[model] = hat_ECS
#dd.to_csv('hat_ECS.csv')

p3 = np.polyfit(dd.hat_ECS,dk.ECS_sum3, deg = 1)
linfit3 = np.polyval(p3,dd.hat_ECS)

fig, ax = plt.subplots(figsize = [9,5])
ax.scatter(dd.hat_ECS, dk.ECS_sum3)
ax.plot(dd.hat_ECS, linfit3 ,linewidth=1, color = 'r')

ax.set_xlabel('hat_ECS',fontsize = 18)
```

```
ax.set_ylabel('ECS_sum3',fontsize = 18)
#ax.set_title('',fontsize = 18)
plt.show()


p4 = np.polyfit(dd.hat_ECS,dk.ECS_sum2, deg = 1)
linfit4 = np.polyval(p4,dd.hat_ECS)

fig, ax = plt.subplots(figsize = [9,5])
ax.scatter(dd.hat_ECS, dk.ECS_sum2)
ax.plot(dd.hat_ECS, linfit4,linewidth=1, color = 'r')

ax.set_xlabel('hat_ECS',fontsize = 18)
ax.set_ylabel('ECS_sum2',fontsize = 18)
#ax.set_title('',fontsize = 18)
plt.show()

p5 = np.polyfit(dd.hat_ECS,dy.ECS, deg = 1)
linfit5 = np.polyval(p5,dd.hat_ECS)

fig, ax = plt.subplots(figsize = [9,5])
ax.scatter(dd.hat_ECS, dy.ECS)
ax.plot(dd.hat_ECS, linfit5,linewidth=1, color = 'r')

ax.set_xlabel('hat_ECS',fontsize = 18)
ax.set_ylabel('ECS (gregory)',fontsize = 18)
#ax.set_title('',fontsize = 18)
plt.show()
```

```
import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from matplotlib import pyplot
import statsmodels.api as sm

import ipynb.fs.full.Functions as eom

#model = 'ACCESS1-0'
#model = 'CanESM2'
#model = 'CNRM-CM5'
#model = 'CSIRO-Mk3-6-0'
#model = 'GFDL-CM3'
#model = 'GFDL-ESM2G'
#model = 'GFDL-ESM2M'
#model = 'GISS-E2-R'
#model = 'HadGEM2-ES'
#model = 'inmcm4'
#model = 'IPSL-CM5A-LR'
#model = 'MIROC5'
#model = 'MIROC-ESM'
#model = 'MPI-ESM-LR'
#model = 'MRI-CGCM3'
model = 'NorESM1-M'


# ### Punkt 1   (model 1)

taulist = np.array([0.7, 9, 354])
par, rnorm, deltaT1, A, Tsum = eom.exp_4CO2_function(model, taulist)

# ### Punkt 3 (model 2)

dr = pd.read_csv('parameter.csv', index_col = 0)
dp = pd.read_csv('gregory.csv', index_col = 0)

parameterlist = (dr.par1[model]/2, dr.par2[model]/2, dr.par3[model
    ]/2)
forcing = pd.read_excel('C://Users/Bruker/Desktop/
    nyjustertforcingmedrcp85.xls')
forcing = forcing[model].values
if model == 'GFDL-CM3' or model == 'GFDL-ESM2G' or model == 'GFDL-
    ESM2M' or model == 'HadGEM2-ES':
        forcing = forcing[10:]

F2x = dp.forcing[model] #gregory forcing
amplitudes = parameterlist/(taulist*F2x)

# compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is a
```

```
    convolution)
dim = len(taulist)
lf = len(forcing)
predictors = np.full((lf,dim),np.nan)

# compute exact predictors by integrating greens function
for k in range(0,dim):
    intgreensti = np.full((lf,lf),0.)   # remember dot after 0 to
    create floating point number array instead of integer
    for t in range(0,lf):
        # compute one new contribution to the matrix:
        intgreensti[t,0] = taulist[k]*(np.exp(-t/taulist[k]) - np.
    exp(-(t+1)/taulist[k]))

        # take the rest from row above:
        if t > 0:
            intgreensti[t,1:(t+1)] = intgreensti[t-1,0:t]
    # compute discretized convolution integral by this matrix
    product:
    predictors[:,k] = intgreensti@np.array(forcing)

histemp = eom.temp_plot(model)

Y = histemp
X = predictors[0:len(histemp)]
X = sm.add_constant(X)

model1, rnorm1 = optimize.nnls(X,Y)
#model1 = sm.OLS(Y,X)
b_tmp = model1
b = np.array([b_tmp[0],b_tmp[1],b_tmp[2],b_tmp[3]])

Tn = b[1:]*predictors

residual = Y - X@b

years = np.arange(0,len(Tn))
fig, ax = plt.subplots(figsize = [9,5])
plt.plot(years,Tn[:,0],linewidth=2,color = "black",label = 'Mode
    with time scale ' + str(taulist[0]) + ' years')
plt.plot(years,Tn[:,1],linewidth=2,color = "blue",label = 'Mode with
     time scale ' + str(taulist[1]) + ' years')
plt.plot(years,Tn[:,2],linewidth=2,color = "red",label = 'Mode with
    time scale ' + str(taulist[2]) + ' years')
ax.set_xlabel('t',fontsize = 18)
ax.set_ylabel('T(t)',fontsize = 18)
ax.set_title('Temperature responses to forcing, model: '+ str(model)
    ,fontsize = 18)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18})

# plot data
```

```
years = np.arange(0,len(Tn))
fig, ax = plt.subplots( figsize = [9,5])
ax.plot(years[0:len(histemp)],histemp,linewidth=2,color = "black")
ax.plot(years[0:len(histemp)],X@b,linewidth=2,color = "blue")
ax.set_ylabel('T(t)',fontsize = 18)
ax.set_title('Global temperature',fontsize = 18)
ax.set_xlabel('Year',fontsize = 18)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=20)

#trekker fra linear trend
p1, lin = eom.lintrend_func(model)

deltaT = histemp - lin[0:len(histemp)]
Tn_1 = Tn[0:len(histemp)]
Tn_156 = np.sum(Tn_1,axis=1)

years_1 = np.arange(0,len(deltaT))

fig, ax = plt.subplots(figsize = [9,5])
ax.plot(years_1,Tn_156,linewidth=2,color = "black",label = 'linear
    response to adjusted forcing')
ax.plot(years_1,deltaT,linewidth=2,color = "red", label = 'modelled
    temperature')
ax.set_xlabel('Time [yr]',fontsize = 14)
ax.set_ylabel('T(t) [K]',fontsize = 14)
ax.set_title('Temperature response to forcing (' +str(model) +')',
    fontsize = 16)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18});

#justerer plot
Tn_mean= np.mean(Tn_156[0:20])
deltaT_mean = np.mean(deltaT[0:20])
factor = Tn_mean - deltaT_mean

new_deltaT = factor+deltaT
new_Tn = Tn_156

print(factor)

fig, ax = plt.subplots(figsize = [9,5])
ax.plot(years_1,new_Tn,linewidth=2,color = "black",label = 'Linear
    response to adjusted forcing')
ax.plot(years_1,new_deltaT,linewidth=2,color = "red", label = '
    Historical run')
ax.set_xlabel('Time [yr]',fontsize = 14)
ax.set_ylabel('T(t) [K]',fontsize = 14)
ax.set_title('Temperature response to forcing (' +str(model) +')',
    fontsize = 16)
ax.grid()
```

```
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=22)
ax.legend(loc=2, prop={'size': 18});

# ### Punkt 4
w = b[1:]
w = w*(taulist*F2x)
Tsum2 = A@w # sum of all components

years0 = np.arange(151)
# create figure
fig, ax = plt.subplots(figsize = [9,5])

Tsum=Tsum/2

ax.plot(years0,Tsum2,linewidth=2,color = "black", label = 'Method 2'
    )
ax.plot(years0,Tsum, linewidth=2, color= "red", label='Method 1')
ax.set_xlabel('Year after 1850 ',fontsize = 12)
ax.set_ylabel('$\Delta$T(t) [K] ',fontsize = 12)
ax.set_title('Estimating $\Delta$T, both methods (' +str(model) +')'
    ,fontsize = 16)
ax.grid()
ax.set_xlim(min(years),max(years))
ax.tick_params(axis='both',labelsize=18)
ax.legend()
plt.show()

diff = Tsum - Tsum2

plt.plot(diff)

dd =  pd.read_excel('diff.xlsx', index_col = 0)
dd[model] = diff
dd.to_excel('diff.xlsx')

mean_vec = np.zeros(151)
for i in range(0,151):
    mean_vec[i] = np.mean(dd.iloc[i,:])

for i in range(0,15):
    differ = dd.iloc[:,i]
    plt.plot(differ)
    plt.grid()
    plt.axhline(linewidth=1, color='black', linestyle='--')
    #plt.axhline(y=np.mean(dd.iloc[150,:]), color='black', linestyle
    ='--')
    plt.plot(mean_vec, linewidth=1.5, color='black', linestyle='-.')
plt.xlabel('time [yr]')
plt.ylabel('Method 1 - Method 2 [K]')

end_point = dd.iloc[150,:]
print(max(end_point))
min(end_point)
```

```
max(end_point)-min(end_point)

reader = pd.read_excel('diff.xlsx')
vec_std = np.zeros(151)
for i in range(0,151):
    vec_std[i] = np.std(reader.iloc[i,])

plt.plot(vec_std)
plt.xlabel('time [yr]')
plt.ylabel('stanard deviation ($\sigma$)')
```

```
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas as pd
import statsmodels.api as sm
from scipy import optimize


#model = 'ACCESS1-0'
#model = 'CanESM2'
#model = 'CNRM-CM5'
#model = 'CSIRO-Mk3-6-0'
#model = 'GFDL-CM3'
#model = 'GFDL-ESM2G'
#model = 'GFDL-ESM2M'
#model = 'GISS-E2-R'
#model = 'HadGEM2-ES'
#model = 'inmcm4'
#model = 'IPSL-CM5A-LR'
#model = 'MIROC5'
#model = 'MIROC-ESM'
#model = 'MPI-ESM-LR'
#model = 'MRI-CGCM3'
model = 'NorESM1-M'


tau1_vec = [0.4, 0.4, 0.7, 1.0, 0.7, 0.7, 1.1, 1.5, 0.8, 0.6, 0.3,
    0.5, 0.5, 0.5, 0.8, 1.2, 1.0, 2.7, 2.5, 0.4, 0.2, 0.7, 0.9, 0.3]
plt.hist(tau1_vec)
plt.show()


tau2_vec = [5, 3, 8, 6, 6, 11, 15, 9, 8, 9, 5, 7, 7, 7, 9, 9, 20,
    54, 30, 11, 5, 10, 11, 9]
plt.hist(tau2_vec)
plt.show()


tau3_vec = [535, 229, 191, 407, 348, 209, 278, 442, 157, 271, 212,
    342, 380, 234, 255, 267, 528, 1010, 597, 434, 403, 453, 417,
    688]
plt.hist(tau3_vec)
plt.show()


n=1000
Tsum_matrix = np.zeros(shape=(151,n))
Tsum2_matrix = np.zeros(shape=(151,n))
diff_matrix = np.zeros(shape=(151,n))
tau_matrix = np.zeros(shape=(3,n))


import ipynb.fs.full.Functions as eom
```

```
for i in range(0,n,1):
    tau1_sample = random.sample(tau1_vec,1)[0]
    tau2_sample =  random.sample(tau2_vec,1)[0]
    tau3_sample = random.sample(tau3_vec,1)[0]
    #np.array([random.sample(tau1_vec,1), random.sample(tau2_vec,1),
     random.sample(tau3_vec,1)])
    taulist = np.array([tau1_sample, tau2_sample, tau3_sample])

    par, rnorm, deltaT1, A, Tsum = eom.exp_4CO2_function(model,
    taulist)
    dr = pd.read_csv('parameter.csv', index_col = 0)
    dp = pd.read_csv('gregory.csv', index_col = 0)

    parameterlist = (dr.par1[model]/2, dr.par2[model]/2, dr.par3[
    model]/2)
    forcing = pd.read_excel('C://Users/Bruker/Desktop/
    nyjustertforcingmedrcp85.xls')
    forcing = forcing[model].values
    if model == 'GFDL-CM3' or model == 'GFDL-ESM2G' or model == '
    GFDL-ESM2M' or model == 'HadGEM2-ES':
            forcing = forcing[10:]

    F2x = dp.forcing[model] #gregory forcing
    amplitudes = parameterlist/(taulist*F2x)

    dim = len(taulist)
    lf = len(forcing)
    predictors = np.full((lf,dim),np.nan)

    # compute exact predictors by integrating greens function
    for k in range(0,dim):
        intgreensti = np.full((lf,lf),0.)   # remember dot after 0
    to create floating point number array instead of integer
        for t in range(0,lf):
            # compute one new contribution to the matrix:
            intgreensti[t,0] = taulist[k]*(np.exp(-t/taulist[k]) -
    np.exp(-(t+1)/taulist[k]))

            # take the rest from row above:
            if t > 0:
                intgreensti[t,1:(t+1)] = intgreensti[t-1,0:t]
        # compute discretized convolution integral by this matrix
    product:
        predictors[:,k] = intgreensti@np.array(forcing)


    histemp = eom.temp_plot(model)

    Y = histemp
    X = predictors[0:len(histemp)]
    X = sm.add_constant(X)

    model1, rnorm1 = optimize.nnls(X,Y)
```

```
    #model1 = sm.OLS(Y,X)
    b_tmp = model1
    b = np.array([b_tmp[0],b_tmp[1],b_tmp[2],b_tmp[3]])

    Tn = b[1:]*predictors

    residual = Y - X@b

    p1, lin = eom.lintrend_func(model)

    deltaT = histemp - lin[0:len(histemp)]
    Tn_1 = Tn[0:len(histemp)]
    Tn_156 = np.sum(Tn_1,axis=1)

    Tn_mean= np.mean(Tn_156[0:20])
    deltaT_mean = np.mean(deltaT[0:20])
    factor = Tn_mean - deltaT_mean

    new_deltaT = factor+deltaT
    new_Tn = Tn_156

    w = b[1:]
    w = w*(taulist*F2x)
    Tsum2 = A@w
    Tsum=Tsum/2 #to correct for 4xCO2

    diff = Tsum - Tsum2

    tau_matrix[:,i]=taulist
    Tsum_matrix[:,i]=Tsum
    Tsum2_matrix[:,i]=Tsum2
    diff_matrix[:,i]=diff


if max(Tsum2_matrix[150,:])> max(Tsum_matrix[150,:]):
    val = max(Tsum2_matrix[150,:])
else:
    val = max(Tsum_matrix[150,:])

for i in range(0,n):
    Tsum_vec = Tsum_matrix[:,i]
    plt.plot(Tsum_vec)
plt.grid()
plt.xlabel('Time [yrs]')
plt.ylabel('Change in temperature [K]')
plt.ylim(-0.1, val+0.2)
plt.savefig(model+'tau-check-4CO2.eps')

for i in range(0,n):
    Tsum2_vec = Tsum2_matrix[:,i]
    plt.plot(Tsum2_vec)
plt.grid()
plt.xlabel('Time [yrs]')
plt.ylabel('Change in temperature [K]')
```

```
plt.ylim(-0.1, val+0.2)
plt.savefig(model+'tau-check-historical.eps')

diff_mean = np.mean(diff_matrix[150,:])
diff_mean = round(diff_mean, 2)
diff_mean

for i in range(0,n,1):
        if round(diff_matrix[150,i],2) == diff_mean:
           no_it=i
           print(i)


for i in range(0,n):
    diff_vec = diff_matrix[:,i]
    plt.plot(diff_vec)
#plt.plot(mean_vec, color='black', label='mean differance ')
#plt.plot(diff_matrix[:,no_it], color='darkblue', linestyle='-.')
plt.grid()
plt.xlabel('Time [yrs]')
plt.ylabel('Change in temperature [K]')
plt.savefig(model+'tau-check-differance.eps')

tau_matrix[:,no_it]
```

```
import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from matplotlib import pyplot


def abruptCO_func(model, taulist, expCO2):
    filedir1 = '/Users/Bruker/Documents/CMIP6'
    realm = 'Amon'
    ensemble = 'r1i1p1f1'
    if model=='GISS-E2-1-G':
        grid_label = 'gn'
    else:
        grid_label = 'gr'

    ## define time periods of data:
    if model == 'IPSL-CM6A-LR':
        abrupt_4xco2_timeperiod = '185001-214912'
        abrupt_2xco2_timeperiod = '185001-199912'
        abrupt_0p5xco2_timeperiod = '185001-199912'
        abrupt_solp4p_timeperiod = '185001-199912'
        piControl_timeperiod = '185001-234912'
        abrupt_exp_branch_yr = 1870

        if expCO2 == 'abrupt-4xCO2':
            abrupt_timeperiod = abrupt_4xco2_timeperiod
        elif expCO2 == 'abrupt-2xCO2':
            abrupt_timeperiod = abrupt_2xco2_timeperiod
        elif expCO2 == 'abrupt-0p5xCO2':
            abrupt_timeperiod = abrupt_0p5xco2_timeperiod

    elif model== 'GISS-E2-1-G':
        abrupt_4xco2_timeperiod = '185001-200012'
        abrupt_2xco2_timeperiod = '185001-200012'
        piControl_timeperiod = '415001-500012'
        abrupt_exp_branch_yr = 4150

        if expCO2 == 'abrupt-4xCO2':
            abrupt_timeperiod = abrupt_4xco2_timeperiod
        elif expCO2 == 'abrupt-2xCO2':
            abrupt_timeperiod = abrupt_2xco2_timeperiod


    ######  load abrupt-co2 data ######
    var = 'ts' # temperatures
    strings = [var, realm, model, expCO2, ensemble, grid_label,
    abrupt_timeperiod]
```

```
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, expCO2, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]


if len(temp)>150:
    temp = temp[0:150]


###### load control run data ######
exp = 'piControl'

var = 'ts' # temperatures

strings = [var, realm, model, exp, ensemble, grid_label,
piControl_timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]


years = np.arange(1,150+1)
controlyears = np.arange(0,len(controltemp))

branchindex = abrupt_exp_branch_yr - int(piControl_timeperiod
[0:4])
print(branchindex)

p1 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controltemp[branchindex:(branchindex + len(temp))], deg =
1)
lintrendT = np.polyval(p1,controlyears[branchindex:(branchindex
+ len(temp))])

deltaT = temp - lintrendT

if expCO2 == 'abrupt-0p5xCO2':
    deltaT=(-1)*deltaT

# for deltaT we also have the information that deltaT(0) = 0.
Include this:
deltaT0 = np.concatenate(([0],deltaT))
years0 = np.concatenate(([0],years))


# compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is a
convolution and F is a constant, so we can compute T_n
analytically)
dim = len(taulist)


A = np.zeros((len(years)+1,dim))
```

```
    for i in range(0,dim): # compute the predictors in the linear
    model for deltaT
        A[:,i] = (1 - np.exp((-np.arange(0,151))/taulist[i]))

    # find parameters par1 in the linear model: deltaT = \sum_i par1
    [i]*(1 - np.exp((-t/tau[i]))

    par1, rnorm1 = optimize.nnls(A,deltaT0)
    Ti = np.array([A[:,i]*par1[i] for i in range(0,dim)]) # compute
    components
    Tsum = A@par1 # sum of all components

    return years0, deltaT0, Tsum

def gregory(model, exp_type):

    filedir1 = '/Users/Bruker/Documents/CMIP6'
    realm = 'Amon'

    if model=='GISS-E2-1-G':
        grid_label = 'gn'
    else:
        grid_label = 'gr'

    ## define time periods of data:

    if model == 'IPSL-CM6A-LR':
        ensemble = 'r1i1p1f1'
        abrupt_4xco2_timeperiod = '185001-214912'
        abrupt_2xco2_timeperiod = '185001-199912'
        abrupt_0p5xco2_timeperiod = '185001-199912'
        abrupt_solp4p_timeperiod = '185001-199912'
        piControl_timeperiod = '185001-234912'
        abrupt_exp_branch_yr = 1870


    elif model == 'CNRM-CM6-1':
        ensemble = 'r1i1p1f2'
        abrupt_4xco2_timeperiod = '185001-199912'
        piControl_timeperiod = '185001-234912'
        abrupt_exp_branch_yr = 1850

    elif model == 'CNRM-ESM2-1':
        ensemble = 'r1i1p1f2'
        abrupt_4xco2_timeperiod = '185001-199912'
        piControl_timeperiod = '185001-234912'
        abrupt_exp_branch_yr = 1850

    elif model== 'GISS-E2-1-G':
        ensemble = 'r1i1p1f1'
        abrupt_4xco2_timeperiod = '185001-200012'
        abrupt_2xco2_timeperiod = '185001-200012'
        piControl_timeperiod = '415001-500012'
        abrupt_exp_branch_yr = 4150
```

```
###### load control run data ######
exp = 'piControl'
timeperiod = piControl_timeperiod
length_wanted = 500 # measured in years

var = 'ts' # temperatures
strings = [var, realm, model, exp, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controltemp=datatable.iloc[:,0]

var = 'rlut' # rlut
strings = [var, realm, model, exp, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrlut=datatable.iloc[0:len(controltemp),0]

var = 'rsut' # rsut
strings = [var, realm, model, exp, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrsut=datatable.iloc[0:len(controltemp),0]

var = 'rsdt' # rsdt
strings = [var, realm, model, exp, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
controlrsdt=datatable.iloc[0:len(controltemp),0]

if len(controltemp)>length_wanted:
    controltemp = controltemp[0:length_wanted]
    controlrlut = controlrlut[0:length_wanted]
    controlrsut = controlrsut[0:length_wanted]
    controlrsdt = controlrsdt[0:length_wanted]

##### load abrupt-exp data #####

if exp_type == 'abrupt-4xCO2':
    timeperiod = abrupt_4xco2_timeperiod
    length_wanted = 300 # measured in years
elif exp_type == 'abrupt-2xCO2':
    timeperiod = abrupt_2xco2_timeperiod
    length_wanted = 150
elif exp_type == 'abrupt-0p5xCO2':
```

```
    timeperiod = abrupt_0p5xco2_timeperiod
    length_wanted = 150
elif exp_type == 'abrupt-solp4p':
    timeperiod = abrupt_solp4p_timeperiod
    length_wanted = 150


var = 'ts' # temperatures
strings = [var, realm, model, exp_type, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp_type, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]

var = 'rlut' # rlut
strings = [var, realm, model, exp_type, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp_type, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rlut=datatable.iloc[:,0]

var = 'rsut' # rsut
strings = [var, realm, model, exp_type, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp_type, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rsut=datatable.iloc[:,0]

var = 'rsdt' # rsdt
strings = [var, realm, model, exp_type, ensemble, grid_label,
timeperiod]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp_type, filename)
datatable = pd.read_table(file, header=None,sep=" ")
rsdt=datatable.iloc[:,0]

if len(temp)>length_wanted:
    temp = temp[0:length_wanted]
    rlut = rlut[0:length_wanted]
    rsut = rsut[0:length_wanted]
    rsdt = rsdt[0:length_wanted]

#exp == 'piControl'

years = np.arange(0,len(temp));
if exp == 'piControl':
    years = np.arange(0,len(controltemp))

# plot temperature
var = temp[:]; label = 'temperature'
if exp == 'piControl':
```

```
    var = controltemp[:]

controlyears = np.arange(0,len(controltemp))

nettoarad = rsdt - rsut - rlut
controlnettoarad = controlrsdt - controlrsut - controlrlut

branchindex = abrupt_exp_branch_yr - int(piControl_timeperiod
[0:4])
#print(branchindex)

p1 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controltemp[branchindex:(branchindex + len(temp))], deg =
1)
lintrendT = np.polyval(p1,controlyears[branchindex:(branchindex
+ len(temp))])

p2 = np.polyfit(controlyears[branchindex:(branchindex + len(temp
))], controlnettoarad[branchindex:(branchindex + len(temp))],
deg = 1)
lintrendN = np.polyval(p2,controlyears[branchindex:(branchindex
+ len(temp))])

deltaN = nettoarad - lintrendN
deltaT = temp - lintrendT

# Gregory plot

fig, ax = plt.subplots(figsize = [9,5])
plt.scatter(deltaT,deltaN,linewidth=1,color = "lavender")
ax.set_xlabel('$\Delta$T',fontsize = 18)
ax.set_ylabel('$\Delta$N',fontsize = 18)
ax.set_title('Gregory plot for ' + str(model) + '  (' + str(
exp_type) + ')',fontsize = 18)
ax.grid()
ax.tick_params(axis='both',labelsize=22)

# find linear fit to these points:
p1 = np.polyfit(deltaT,deltaN,deg = 1)


#linfit = np.polyval(p1,deltaT)
deltaTextended = np.concatenate(([0],deltaT,[-p1[1]/p1[0]]))
linfit = np.polyval(p1,deltaTextended)
plt.plot(deltaTextended,linfit,linewidth=1,color = "black")

if p1[1]>0:
    ax.set_ylim(-1,p1[1]+1)
    ax.set_xlim(-0.3,-p1[1]/(p1[0])+1)
else:
    ax.set_ylim(p1[1],-1)
    ax.set_xlim(-p1[1]/(p1[0]),0)

return p1
```

```
# ## Making Gregory-plot

#model = 'IPSL-CM6A-LR'
#model = 'CNRM-CM6-1'
#model = 'CNRM-ESM2-1'
model = 'GISS-E2-1-G'

exp_type = 'abrupt-4xCO2'
#exp_type = 'abrupt-2xCO2'
#exp_type = 'abrupt-0p5xCO2'
#exp_type = 'abrupt-solp4p'

p1 = gregory(model, exp_type)


if exp_type == 'abrupt-2xCO2':
    print('F2xCO2 = '+ str(p1[1]))
    print('feedback parameter = ' + str(p1[0]))
    print('ECS / T2xCO2 = ' + str(-p1[1]/p1[0]))
elif exp_type == 'abrupt-4xCO2':
    convfactor = 1/2 # that converts to corresponding estimates for
    a doubling of CO2
    print('converted to measures for 2xCO2:')
    print('F2xCO2 = '+ str(p1[1]*convfactor))
    print('feedback parameter = ' + str(p1[0]))
    print('ECS / T2xCO2 = ' + str(-p1[1]*convfactor/p1[0]))
elif exp_type == 'abrupt-0p5xCO2':
    print('Forcing of 0p5xCO2 = ' + str(p1[1]))
    print('feedback parameter = ' + str(p1[0]))
    print('deltaT after 0p5CO2 = ' + str(-p1[1]/p1[0]))
elif exp_type == 'abrupt-solp4p':
    print('Forcing of solp4p = ' + str(p1[1]))
    print('feedback parameter = ' + str(p1[0]))
    print('deltaT after solp4p = ' + str(-p1[1]/p1[0]))

p1[1]

# ## Making different tabels

### making table for different ECS-estimates for IPSL-CM6A-LR ###
ipsl = pd.read_csv('IPSL-CM6A-LR.csv', index_col = 0)
if model=='IPSL-CM6A-LR':

    ipsl = pd.read_csv('IPSL-CM6A-LR.csv', index_col = 0)
    if exp_type == 'abrupt-4xCO2':
        ipsl.deltaT[exp_type]=-p1[1]/p1[0]
        ipsl.forcing[exp_type]=p1[1]
        ipsl.feedback[exp_type]=p1[0]
    else:
        ipsl.deltaT[exp_type]=-p1[1]/(p1[0])
        ipsl.forcing[exp_type]=p1[1]
        ipsl.feedback[exp_type]=p1[0]
```

```
    ipsl.to_csv('IPSL-CM6A-LR.csv')
ipsl


ipsl.forcing['abrupt-2xCO2'] - ipsl.forcing['abrupt-4xCO2']/2

ipsl.forcing['abrupt-2xCO2'] + ipsl.forcing['abrupt-0p5xCO2']


### making table for 4xCO2 for all models in CMIP6 ###
if exp_type== 'abrupt-4xCO2':
    df = pd.read_csv('cmip6_gregory.csv', index_col = 0)
    df.ECS[model]=-p1[1]/(p1[0]*2)
    df.forcing[model]=p1[1]*(1/2)
    df.feedback[model]=p1[0]

    df.to_csv('cmip6_gregory.csv')


# ## Compare IPSL-model in Gregory

x = np.arange(0,10)
greg6 = pd.read_csv('IPSL-gregory.csv', index_col= 0)
ipsl5 = greg6.feedback['IPSL-CM5A-LR']*x + greg6.forcing['IPSL-CM5A-
    LR']
ipsl6 = greg6.feedback['IPSL-CM6A-LR']*x + greg6.forcing['IPSL-CM6A-
    LR']

plt.plot(x,ipsl5, label='IPSL-CM5A-LR')
plt.plot(x,ipsl6, label='IPSL-CM6A-LR')
plt.hlines(0,0,5, linestyles='dashed')
plt.grid()
plt.xlim(0,5)
plt.ylim(-0.5,4)
plt.xlabel('$\Delta T$ [K]', fontsize=12)
plt.ylabel('$\Delta N $ [W/m$^2$]', fontsize=12)
plt.title('IPSL-model for CMIP5 and CMIP6', fontsize=16)
plt.legend()
print('ECS for IPSL-CM5A-LR:' + str(greg6.ECS['IPSL-CM5A-LR']))
print('ECS for IPSL-CM6A-LR:' + str(greg6.ECS['IPSL-CM6A-LR']))
plt.savefig('CMIP6-IPSL.png')


# ## Compare GISS-model in Gregory

x = np.arange(0,10)
giss6 = pd.read_csv('GISS-gregory.csv', index_col=0)
giss5_1 = giss6.feedback['GISS-E2-R']*x + giss6.forcing['GISS-E2-R']
giss5_2 = giss6.feedback['GISS-E2-H']*x + giss6.forcing['GISS-E2-H']
giss6 = giss6.feedback['GISS-E2-1-G']*x + giss6.forcing['GISS-E2-1-G
    ']

plt.plot(x,giss5_1, label='GISS-E2-R')
```

```
plt.plot(x,giss5_2, label='GISS-E2-H')
plt.plot(x,giss6, label='GISS-E2-1-G')
plt.hlines(0,0,5, linestyles='dashed')
plt.grid()
plt.xlim(0,3)
plt.ylim(-0.5,4)
plt.xlabel('$\Delta T$ [K]', fontsize=12)
plt.ylabel('$\Delta N $ [W/m$^2$]', fontsize=12)
plt.title('GISS-model for CMIP5 and CMIP6', fontsize=16)
plt.legend()
plt.savefig('CMIP6-GISS.png')


# ## Abrupt CO2, comparing different scenario

taulist = np.array([0.7, 9, 354])
model = 'IPSL-CM6A-LR'

exp4xCO2 = 'abrupt-4xCO2'
exp2xCO2 = 'abrupt-2xCO2'
exp0p5xCO2 = 'abrupt-0p5xCO2'

years_4xCO2, var_4xCO2, Tsum_4xCO2 = abruptCO_func(model, taulist,
    exp4xCO2)
years_2xCO2, var_2xCO2, Tsum_2xCO2 = abruptCO_func(model, taulist,
    exp2xCO2)
years_0p5xCO2, var_0p5xCO2, Tsum_0p5xCO2 = abruptCO_func(model,
    taulist, exp0p5xCO2)

fig, ax = plt.subplots(figsize = [9,5])

ax.plot(years_4xCO2,var_4xCO2,linewidth=2,color = "silver")
ax.plot(years_4xCO2,Tsum_4xCO2,linewidth=2,color = "slateblue",
    label='4xCO2')
ax.plot(years_2xCO2,var_2xCO2,linewidth=2,color = "silver")
ax.plot(years_2xCO2,Tsum_2xCO2,linewidth=2,color = "mediumvioletred"
    , label='2xCO2')
ax.plot(years_0p5xCO2,var_0p5xCO2,linewidth=2,color = "silver")
ax.plot(years_0p5xCO2,Tsum_0p5xCO2,linewidth=2,color = "darkblue",
    label='0.5xCO2')


ax.set_xlabel('Years after 1850 ',fontsize = 11)
ax.set_ylabel('$\Delta T$(t) ',fontsize = 11)
ax.set_title( '$\Delta T$, model: ' +str(model) ,fontsize = 18)
ax.grid()
ax.tick_params(axis='both',labelsize=18)
ax.legend()

fig, ax = plt.subplots(figsize = [9,5])

ax.plot(years_4xCO2,0.5*Tsum_4xCO2,linewidth=2, label='4xCO2')
ax.plot(years_2xCO2,Tsum_2xCO2,linewidth=2, label='2xCO2')
ax.plot(years_0p5xCO2,Tsum_0p5xCO2,linewidth=2, label='0.5xCO2')
```

```
ax.set_xlabel('Year after 1850 ',fontsize = 12)
ax.set_ylabel('$\Delta T$ [K] ',fontsize = 12)
ax.set_title( 'Temperature change for model: ' +str(model) ,fontsize
    = 16)
ax.grid()
ax.tick_params(axis='both',labelsize=18)
ax.legend()
fig.savefig('IPSL-comparing.png')

model = 'GISS-E2-1-G'
years_4xCO2, var_4xCO2, Tsum_4xCO2 = abruptCO_func(model, taulist,
    exp4xCO2)
years_2xCO2, var_2xCO2, Tsum_2xCO2 = abruptCO_func(model, taulist,
    exp2xCO2)

fig, ax = plt.subplots(figsize = [9,5])

ax.plot(years_4xCO2,var_4xCO2,linewidth=2,color = "silver")
ax.plot(years_4xCO2,Tsum_4xCO2,linewidth=2,color = "slateblue",
    label='4xCO2')
ax.plot(years_2xCO2,var_2xCO2,linewidth=2,color = "silver")
ax.plot(years_2xCO2,Tsum_2xCO2,linewidth=2,color = "mediumvioletred"
    , label='2xCO2')

ax.set_xlabel('Years after 1850 ',fontsize = 11)
ax.set_ylabel('$\Delta T$(t) ',fontsize = 11)
ax.set_title( '$\Delta T$, model: ' +str(model) ,fontsize = 18)
ax.grid()
ax.tick_params(axis='both',labelsize=18)
ax.legend()

fig, ax = plt.subplots(figsize = [9,5])

ax.plot(years_4xCO2,0.5*Tsum_4xCO2,linewidth=2, label='4xCO2')
ax.plot(years_2xCO2,Tsum_2xCO2,linewidth=2, label='2xCO2')

ax.set_xlabel('Year after 1850 ',fontsize = 12)
ax.set_ylabel('$\Delta T$ [K]',fontsize = 12)
ax.set_title( 'Temperature change, model: ' +str(model) ,fontsize =
    16)
ax.grid()
ax.tick_params(axis='both',labelsize=18)
ax.legend()
fig.savefig('GISS-comparing.png')


# ## Plotting Historical temperatures

temp = eom.temp_plot('CNRM-CM5')
model = 'CNRM-CM6-1'
filedir1 = '/Users/Bruker/Documents/CMIP6'
realm = 'Amon'
ensemble = 'r1i1p1f2'
```

```
exp = 'historical'
gr = 'gr'
period = '185001-201412'

var = 'ts'
strings = [var, realm, model, exp, ensemble, gr, period]
filename = 'glannual_' + "_".join(strings) + '.txt'
file = os.path.join(filedir1, model, exp, filename)
datatable = pd.read_table(file, header=None,sep=" ")
temp=datatable.iloc[:,0]

fig, ax = plt.subplots(figsize = [9,5])
plt.plot(temp,linewidth=1,color = "red")
ax.set_xlabel('Years (after 1850)',fontsize = 10)
ax.set_ylabel('Temperature (K)',fontsize = 10)
ax.set_title('Change in surface temperature, model: ' + str(model)
    , fontsize = 18)
ax.grid()
ax.tick_params(axis='both',labelsize=12)
```

```
import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
import random
from matplotlib import pyplot

model_list= np.array(['ACCESS1-0','CanESM2','CNRM-CM5', 'CSIRO-Mk3
    -6-0', 'GFDL-CM3', 'GFDL-ESM2G', 'GFDL-ESM2M', 'GISS-E2-R', '
    HadGEM2-ES',
                      'inmcm4', 'IPSL-CM5A-LR', 'MIROC5', 'MIROC-ESM
    ', 'MPI-ESM-LR', 'MRI-CGCM3', 'NorESM1-M'])
tau1_vec = [0.4, 0.4, 0.7, 1.0, 0.7, 0.7, 1.1, 1.5, 0.8, 0.6, 0.3,
    0.5, 0.5, 0.5, 0.8, 1.2, 1.0, 2.7, 2.5, 0.4, 0.2, 0.7, 0.9, 0.3]
tau2_vec = [5, 3, 8, 6, 6, 11, 15, 9, 8, 9, 5, 7, 7, 7, 9, 9, 20,
    54, 30, 11, 5, 10, 11, 9]
tau3_vec = [535, 229, 191, 407, 348, 209, 278, 442, 157, 271, 212,
    342, 380, 234, 255, 267, 528, 1010, 597, 434, 403, 453, 417,
    688]


import ipynb.fs.full.Functions as func
dg = pd.read_csv('gregory_short.csv', index_col = 0)
dr = pd.read_csv('parameter.csv', index_col = 0)
dp = pd.read_csv('gregory.csv', index_col = 0)


#model = 'NorESM1-M'
n=1000

#tau_matrix = np.zeros(shape=(3,n))
for i in range(0, len(model_list),1):
    spec_vec = np.zeros(n)
    model=model_list[i]


    for i in range(0,n,1):
        tau1_sample = random.sample(tau1_vec,1)[0]
        tau2_sample =  random.sample(tau2_vec,1)[0]
        tau3_sample = random.sample(tau3_vec,1)[0]
        taulist = np.array([tau1_sample, tau2_sample, tau3_sample])


        histemp = func.temp_plot(model)
        par_4CO2, rnorm, deltaT, A, Tsum = func.exp_4CO2_function(
    model, taulist)

        parameterlist = (dr.par1[model]/2, dr.par2[model]/2, dr.par3
    [model]/2)
```

```
    forcing = pd.read_excel('C://Users/Bruker/Desktop/
nyjustertforcingmedrcp85.xls')
    forcing = forcing[model].values
    if model == 'GFDL-CM3' or model == 'GFDL-ESM2G' or model ==
'GFDL-ESM2M' or model == 'HadGEM2-ES':
        forcing = forcing[10:]

    F2x = dp.forcing[model]
    amplitudes = parameterlist/(taulist*F2x)

    # compute components T_n(t) = exp(-t/tau_n)*F(t) (Here * is
a convolution)
    dim = len(taulist)
    lf = len(forcing)
    predictors = np.full((lf,dim),np.nan)

    # compute exact predictors by integrating greens function
    for k in range(0,dim):
        intgreensti = np.full((lf,lf),0.)   # remember dot after
 0 to create floating point number array instead of integer
        for t in range(0,lf):
            # compute one new contribution to the matrix:
            intgreensti[t,0] = taulist[k]*(np.exp(-t/taulist[k])
 - np.exp(-(t+1)/taulist[k]))

            # take the rest from row above:
            if t > 0:
                intgreensti[t,1:(t+1)] = intgreensti[t-1,0:t]
        # compute discretized convolution integral by this
matrix product:
        predictors[:,k] = intgreensti@np.array(forcing)


    Tn = amplitudes*predictors

    p1, lin = func.lintrend_func(model)


    period = '185001-200512'
    if model == 'bcc-csm1-1' or model == 'bcc-csm1-1-m' or model
 == 'MIROC5':
        period = '185001-201212'
    elif model == 'GFDL-CM3':
        period = '186001-200512'
    elif model == 'GFDL-ESM2G' or model == 'GFDL-ESM2M':
        period = '186101-200512'
    elif model == 'HadGEM2-ES':
        period = '186001-200511'

    filedir1 = '/Users/Bruker/Documents/Forcingpaperdata'
    strings = ['ts', 'Amon', model, 'historical', 'r1i1p1',
period]
    filename = 'glannual_' + "_".join(strings) + '.txt'
```

```
        file = os.path.join(filedir1, model, 'historical', filename)
        datatable = pd.read_table(file, header=None,sep=" ")
        temp=datatable.iloc[:,0]

        deltaT = temp - lin[0:len(temp)]
        Tn_1 = Tn[0:len(temp)]
        Tn_156 = np.sum(Tn_1,axis=1)
        res = Tn_156 - deltaT

        spect_hist = func.psd_func(res,taulist)
        spec_vec[i] = spect_hist[0]

    dd = pd.read_excel('hist_S0.xlsx')
    dd[model]=spec_vec
    dd.to_excel('hist_S0.xlsx')


Y = dg.ECS
X = np.mean(dd.iloc[:,])
plt.scatter(X,Y)


fig, ax = plt.subplots(figsize = [8,8])
ax.grid()
ax.set_ylabel('Gregory estimates of ECS$^2$ [K]', fontsize=12)
ax.set_xlabel('S(0)', fontsize=12)

for i in range(0,len(model_list),1):
    model = model_list[i]
    snitt = np.mean(dd.iloc[:,i])
    std = np.std(dd.iloc[:,i])

    Y = dg.ECS[model]
    X = snitt

    ax.errorbar(X, Y**2, fmt='o', xerr=std, label= str(model))
    ax.legend()


snitt_vec=np.zeros(len(model_list))
for p in range(0,len(model_list),1):
    snitt_vec[p]=np.mean(dd.iloc[:,p])

p1 = np.polyfit(snitt_vec, dg.ECS**2, deg= 1)
linfit1 = np.polyval(p1, snitt_vec)

plt.plot(snitt_vec, linfit1, color='grey')

print('Correlation: ' + str(np.corrcoef(dg.ECS**2, snitt_vec)[0,1]))


snitt_vec=np.zeros(len(model_list))
for p in range(0,len(model_list),1):
```

```
    snitt_vec[p]=np.mean(dd.iloc[:,p])

p1 = np.polyfit(snitt_vec, dg.ECS**2, deg= 1)
linfit1 = np.polyval(p1, snitt_vec)

plt.scatter(snitt_vec, dg.ECS**2)
plt.plot(snitt_vec, linfit1)
plt.xlabel('S(0)', fontsize=12)
plt.ylabel('ECS$^2$', fontsize=12)
print('Correlation: ' + str(np.corrcoef(dg.ECS**2, snitt_vec)[0,1]))
```

```
import os
import numpy as np
from numpy import NaN
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import signal
from scipy import optimize
from matplotlib import pyplot


model = 'IPSL-CM6A-LR'
filedir1 = '/Users/Bruker/Documents/CMIP6'
realm = 'Amon'

ensemble_vec = np.array(['r1i1p1f1', 'r2i1p1f1', 'r3i1p1f1', '
    r4i1p1f1', 'r5i1p1f1',
                         'r6i1p1f1', 'r7i1p1f1', 'r8i1p1f1', '
    r9i1p1f1', 'r10i1p1f1',
                         'r11i1p1f1', 'r12i1p1f1', 'r13i1p1f1', '
    r14i1p1f1', 'r15i1p1f1',
                         'r16i1p1f1', 'r17i1p1f1', 'r18i1p1f1', '
    r19i1p1f1', 'r20i1p1f1',
                         'r21i1p1f1', 'r22i1p1f1', 'r23i1p1f1', '
    r24i1p1f1', 'r25i1p1f1',
                         'r26i1p1f1', 'r27i1p1f1', 'r28i1p1f1', '
    r29i1p1f1', 'r30i1p1f1',
                         'r31i1p1f1'])

fig, ax = plt.subplots(figsize = [9,5])
ax.set_xlabel('Year after 1850',fontsize = 10)
ax.set_ylabel('Temperature [K]',fontsize = 10)
ax.set_title('Historical temperature (' + str(model) +')'  ,
    fontsize = 18)
ax.tick_params(axis='both',labelsize=12)
ax.grid()

ensemble = 'r1i1p1f1'
exp = 'historical'
gr = 'gr'
period = '185001-201412'
var = 'ts'

for i in range(0, len(ensemble), 1):
    ensemble = ensemble_vec[i]
    strings = [var, realm, model, exp, ensemble, gr, period]
    filename = 'glannual_' + "_".join(strings) + '.txt'
    file = os.path.join(filedir1, model, exp, filename)
    datatable = pd.read_table(file, header=None,sep=" ")
    temp=datatable.iloc[:,0]


    plt.plot(temp,linewidth=1,color = "red")
```

```
mean_temp = np.zeros(len(temp))
for i in range(0,len(temp),1):
    tmp = np.zeros(len(ensemble_vec))
    for t in range(0,len(ensemble_vec)):
        ensemble=ensemble_vec[t]
        strings = [var, realm, model, exp, ensemble, gr, period]
        filename = 'glannual_' + "_".join(strings) + '.txt'
        file = os.path.join(filedir1, model, exp, filename)
        datatable = pd.read_table(file, header=None,sep=" ")
        temp=datatable.iloc[:,0]

        tmp[t]=temp[i]
    mean_temp[i]=np.mean(tmp)

ax.plot(mean_temp, color='black')


res_matrix = np.zeros(shape=(len(temp),len(ensemble_vec)))

for i in range(0,len(ensemble_vec),1):
    ensemble = ensemble_vec[i]
    strings = [var, realm, model, exp, ensemble, gr, period]
    filename = 'glannual_' + "_".join(strings) + '.txt'
    file = os.path.join(filedir1, model, exp, filename)
    datatable = pd.read_table(file, header=None,sep=" ")
    temp=datatable.iloc[:,0]

    res = mean_temp - temp

    res_matrix[:,i]= res



import ipynb.fs.full.Functions as fun

taulist = np.array([0.7, 9, 354])
s, S_res = fun.psd_plotfunc(res, taulist)

taulist = np.array([0.7, 9, 354])
for i in range(0, len(ensemble_vec), 1):
    res = res_matrix[:,i]
    s, S_res = fun.psd_plotfunc(res, taulist)

    cc = pd.read_csv('IPSL-CM6A-LR_S0-values.csv', index_col=0)
    cc.S0[i+1] = s[0]
    cc.to_csv('IPSL-CM6A-LR_S0-values.csv')

    cc1 =  pd.read_csv('IPSL-CM6A-LR_S0-values_met1.csv', index_col
    =0)
    cc1.S0[i+1] = np.mean(S_res[1:25])
    cc1.to_csv('IPSL-CM6A-LR_S0-values_met1.csv')


cc = pd.read_csv('IPSL-CM6A-LR_S0-values.csv', index_col=0)
```

```
cc1 =  pd.read_csv('IPSL-CM6A-LR_S0-values_met1.csv', index_col=0)
print( 'The mean: ' +str(np.mean(cc.S0)))
print( 'Minimum: '+str(np.min(cc.S0)))
print( 'Maximum: ' + str(np.max(cc.S0)))
print( 'Standard deviation: ' + str(np.std(cc.S0)))
print( 'The mean: ' +str(np.mean(cc1.S0)))
print( 'Minimum: '+str(np.min(cc1.S0)))
print( 'Maximum: ' + str(np.max(cc1.S0)))
print( 'Standard deviation: ' + str(np.std(cc1.S0)))



plt.hist(cc.S0)
plt.show()
```

Bibliography

[CMI, 2019] (2019). Wcrp coupled model intercomparison project (cmip). [Online; accessed 15-February-2019].

[Almeida, 1997] Almeida, L. B. (1997). Product and convolution theorems for the fractional fourier transform. *IEEE Signal Processing Letters*, 4(1):15–17.

[Andrews and Jelley, 2013] Andrews, J. and Jelley, N. (2013). *Energy Science: Principles, Technologies, and Impacts*. OUP Oxford.

[Andrews et al., 2018] Andrews, T., Gregory, J. M., Paynter, D., Silvers, L. G., Zhou, C., Mauritsen, T., Webb, M. J., Armour, K. C., Forster, P. M., and Titchner, H. (2018). Accounting for changing temperature patterns increases historical estimates of climate sensitivity. *Geophysical Research Letters*, 45(16):8490–8499.

[Andrews et al., 2012] Andrews, T., Gregory, J. M., Webb, M. J., and Taylor, K. E. (2012). Forcing, feedbacks and climate sensitivity in cmip5 coupled atmosphere-ocean climate models. *Geophysical Research Letters*, 39(9).

[Cox et al., 2018] Cox, P., Huntingford, C., and S. Williamson, M. (2018). Emergent constraint on equilibrium climate sensitivity from global temperature variability. *Nature*, 553:319–322.

[Dessler and Forster, 2018] Dessler, A. E. and Forster, P. M. (2018). An estimate of equilibrium climate sensitivity from interannual variability. *Journal of Geophysical Research: Atmospheres*, 123(16):8634–8645.

[Forster et al., 2013] Forster, P. M., Andrews, T., Good, P., Gregory, J. M., Jackson, L. S., and Zelinka, M. (2013). Evaluating adjusted forcing and model spread for historical and future scenarios in the cmip5 generation of climate models. *Journal of Geophysical Research: Atmospheres*, 118(3):1139–1150.

[Fredriksen and Rypdal, 2017] Fredriksen, H.-B. and Rypdal, M. (2017). Long-range persistence in global surface temperatures explained by linear multi-

box energy balance models. *Journal of Climate*, 30(18):7157–7168.

[Gillespie, 1996] Gillespie, D. T. (1996). Exact numerical simulation of the ornstein-uhlenbeck process and its integral. *Phys. Rev. E*, 54:2084–2091.

[Gregory et al., 2016] Gregory, J. M., Andrews, T., Good, P., Mauritsen, T., and Forster, P. M. (2016). Small global-mean cooling due to volcanic radiative forcing. *Climate Dynamics*, 47(12):3979–3991.

[Gregory et al., 2004] Gregory, J. M., Ingram, W. J., Palmer, M. A., Jones, G. S., Stott, P. A., Thorpe, R. B., Lowe, J. A., Johns, T. C., and Williams, K. D. o. O. (2004). A new method for diagnosing radiative forcing and climate sensitivity.

[Hassani, 2009] Hassani, S. (2009). *Mathematical Methods: For Students of Physics and Related Fields*. Springer, New York, NY.

[Kaper and Engler, 2013] Kaper, H. and Engler, H. (2013). *Mathematics and Climate*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[Knutti et al., 2017] Knutti, R., A. A. Rugenstein, M., and Hegerl, G. (2017). Beyond equilibrium climate sensitivity. 10.

[Myhre et al., 2017] Myhre, G., Myhre, C. L., and Shine, K. P. (2017). Halfway to doubling of co2 radiative forcing. *Nature Geoscience*, 10(10):710–711.

[Otto et al., 2013] Otto, A., E. L. Otto, F., Boucher, O., Church, J., Hegerl, G., Forster, P., P. Gillett, N., Gregory, J., Johnson, G., Knutti, R., Lewis, N., Lohmann, U., Marotzke, J., Myhre, G., Shindell, D., Stevens, B., and R. Allen, M. (2013). Energy budget constraints on climate response. *Nature Geoscience*, 6:415–416.

[Pachauri et al., 2014] Pachauri, R. K., Allen, M. R., Barros, V. R., Broome, J., Cramer, W., Christ, R., Church, J. A., Clarke, L., Dahe, Q., Dasgupta, P., Dubash, N. K., Edenhofer, O., Elgizouli, I., Field, C. B., Forster, P., Friedlingstein, P., Fuglestvedt, J., Gomez-Echeverri, L., Hallegatte, S., Hegerl, G., Howden, M., Jiang, K., Cisneroz, B. J., Kattsov, V., Lee, H., Mach, K. J., Marotzke, J., Mastrandrea, M. D., Meyer, L., Minx, J., Mulugetta, Y., O'Brien, K., Oppenheimer, M., Pereira, J. J., Pichs-Madruga, R., Plattner, G.-K., Pörtner, H.-O., Power, S. B., Preston, B., Ravindranath, N. H., Reisinger, A., Riahi, K., Rusticucci, M., Scholes, R., Seyboth, K., Sokona, Y., Stavins, R., Stocker, T. F., Tschakert, P., van Vuuren, D., and van Ypserle, J.-P. (2014). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. IPCC, Geneva,

Switzerland.

[Proistosescu and Huybers, 2017] Proistosescu, C. and Huybers, P. J. (2017). Slow climate mode reconciles historical and model-based estimates of climate sensitivity. *Science Advances*, 3(7).

[Rypdal et al., 2018a] Rypdal, M., Fredriksen, H.-B., Myrvoll-Nilsen, E., Rypdal, K., and Sørbye, S. H. (2018a). Emergent scale invariance and climate sensitivity. *Climate*, 6(4).

[Rypdal et al., 2018b] Rypdal, M., Fredriksen, H.-B., Rypdal, K., and Steene, R. (2018b). Emergent constraints on climate sensitivity. *Nature*, 563.

[Sauer, 2011] Sauer, T. (2011). *Numerical Analysis*. Addison-Wesley Publishing Company, USA, 2nd edition.

[Sherwood et al., 2015] Sherwood, S. C., Bony, S., Boucher, O., Bretherton, C., Forster, P. M., Gregory, J. M., and Stevens, B. (2015). Adjustments in the forcing-feedback framework for understanding climate change. *Bulletin of the American Meteorological Society*, 96(2):217–228.

[Taylor et al., 2012] Taylor, K. E., Stouffer, R. J., and Meehl, G. A. (2012). An overview of cmip5 and the experiment design. *Bulletin of the American Meteorological Society*, 93(4):485–498.