# Comparison of temperature response for various climate gases

Sara Blichner, T. K. Berntsen

March 23, 2021

## 1 Plot temperature response over time

### 1.1 Standard case (not $CO_2$):

Radiative forcing:

$$RF(t) = R_0\tau(1 - e^{-t/\tau})$$

Instantaneous response function:

$$IRF(t) = \alpha_1 e^{-t/\tau_1} + \alpha_2 e^{-t/\tau_2}$$

$$\begin{aligned}
\Delta T(t) &= \int_0^t RF(t')IRF(t-t')dt' \\
&= \int_0^t \Big(R_0\tau(1 - e^{-t'/\tau})\Big) \cdot \Big(\alpha_1 e^{-(t-t')/\tau_1} + \alpha_2 e^{-(t-t')/\tau_2}\Big)dt' \\
&= \tau R_0\Big[\int_0^t \alpha_1 e^{-\frac{t-t'}{\tau_1}} + \alpha_2 e^{-\frac{t-t'}{\tau_2}}dt' - \int_0^t \alpha_1 e^{-t/\tau_1}e^{-t'(\frac{1}{\tau}-\frac{1}{\tau_1})} + \alpha_2 e^{-t/\tau_2}e^{-t'(\frac{1}{\tau}-\frac{1}{\tau_2})}dt'\Big]
\end{aligned}$$

Let

$$\beta_i = \Big(\frac{1}{\tau} - \frac{1}{\tau_i}\Big)^{-1} = \frac{\tau_i\tau}{\tau_i - \tau}$$

and then:

$$\begin{aligned}
\Delta T(t) &= \tau R_0\Big[\int_0^t \alpha_1 e^{-\frac{t-t'}{\tau_1}} + \alpha_2 e^{-\frac{t-t'}{\tau_2}}dt' - \int_0^t \alpha_1 e^{-t/\tau_1}e^{-t'/\beta_1} + \alpha_2 e^{-t/\tau_2}e^{-t'/\beta_2}dt'\Big] \\
&= R_0\tau\Big[I_1 - I_2\Big]
\end{aligned}$$

Thus $I_1$:

$$\begin{aligned}
I_1 &= \int_0^t \alpha_1 e^{-\frac{t-t'}{\tau_1}} + \alpha_2 e^{-\frac{t-t'}{\tau_2}}dt' \\
&= \alpha_1\tau_1(1 - e^{-t/\tau_1}) + \alpha_2\tau_2(1 - e^{-t/\tau_2})
\end{aligned}$$

Then for $I_2$

$$\begin{aligned}
I_2 &= \int_0^t \alpha_1 e^{-t/\tau_1} e^{-t'/\beta_1} + \alpha_2 e^{-t/\tau_2} e^{-t'/\beta_2} dt' \\
&= -\alpha_1\beta_1 e^{-t/\tau_1}(e^{-t/\beta_1} - 1) - \alpha_2\beta_2 e^{-t/\tau_2}(e^{-t/\beta_2} - 1) \\
&= -\alpha_1\beta_1 e^{-t/\tau_1}(e^{-\frac{t}{\tau}+\frac{t}{\tau_i}} - 1) - \alpha_2\beta_2 e^{-t/\tau_2}(e^{-t/\beta_2} - 1) \\
&= -\alpha_1\beta_1(e^{-t/\tau} - e^{-t/\tau_1}) - \alpha_2\beta_2(e^{-t/\tau} - e^{-t/\tau_2})
\end{aligned}$$

#### 1.1.1 Solution:

So, following this:

$$\begin{aligned}
\Delta T(t) &= R_0\tau\Big[I_1 - I_2\Big] \\
&= R_0\tau\Big[\alpha_1\tau_1(1 - e^{-t/\tau_1}) + \alpha_2\tau_2(1 - e^{-t/\tau_2}) + \big(\alpha_1\beta_1(e^{-t/\tau} - e^{-t/\tau_1}) + \alpha_2\beta_2(e^{-t/\tau} - e^{-t/\tau_2})\Big]
\end{aligned}$$

### 1.2 Standard case (not $CO_2$):

Radiative forcing:

$$RF(t) = R_0\tau(1 - e^{-t/\tau})$$

Instantaneous response function:

$$IRF(t) = \alpha_1 e^{-t/\tau_1} + \alpha_2 e^{-t/\tau_2}$$

#### 1.2.1 Constants climate IRF function

$$\begin{aligned}
\text{IRF}(t) =& 0.885 \cdot (\frac{0.587}{4.1} \cdot exp(\frac{-t}{4.1}) + \frac{0.413}{249} \cdot exp(\frac{-t}{249})) \\
\text{IRF}(t) =& \sum_{i=1}^{2} \frac{c_i}{\tau_i} \cdot exp(\frac{-t}{\tau_1})
\end{aligned}$$

with $c_1 = 0.587 \cdot 0.885$, $\tau_1 = 4.1$, $c_2 = 0.413 \cdot 0.885$ and $\tau_2 = 249$.

With new version:

$$\text{IRF}(t) = \sum_{i=1}^{2} \frac{q_1}{d_i} \cdot exp(\frac{-t}{d_1})$$

So:

With new version:

$$\text{IRF}(t) = \sum_{i=1}^{2} \frac{q_1}{d_i} \cdot exp(\frac{-t}{d_1})$$

So: $\tau_i = d_i$ and $c_i = q_i$

### 1.3 updated IRF:

[1]:
```python
import pandas as pd
```

[2]:
```python
fn_IRF_constants = 'recommended_irf_from_2xCO2_2021_02_25_222758.csv'
irf_consts = pd.read_csv(fn_IRF_constants).set_index('id')

ld1 = 'd1 (yr)'
ld2 = 'd2 (yr)'
lq1 = 'q1 (K / (W / m^2))'
lq2 = 'q2 (K / (W / m^2))'
median = 'median'
perc5 = '5th percentile'
perc95 = '95th percentile'
irf_consts  # [d1]
lalph1_irf = 'alpha1'
lalph2_irf = 'alpha2'
ltau1_irf = 'tau1'
ltau2_irf = 'tau2'
irf_consts[ltau1_irf] = irf_consts[ld1]
irf_consts[ltau2_irf] = irf_consts[ld2]
irf_consts[lalph1_irf] = irf_consts[lq1]/irf_consts[ld1]
irf_consts[lalph2_irf] = irf_consts[lq2]/irf_consts[ld2]
med_irf = irf_consts.loc['recommendation']
med_irf
```

[2]:
```
C (W yr / m^2 / K)            7.649789
C_d (W yr / m^2 / K)        147.168593
alpha (W / m^2 / K)           1.310000
eta (dimensionless)           1.027856
kappa (W / m^2 / K)           0.880636
d1 (yr)                       3.424102
d2 (yr)                     285.003478
q1 (K / (W / m^2))            0.443768
q2 (K / (W / m^2))            0.319591
efficacy (dimensionless)      1.027856
ecs (K)                       3.000000
tcr (K)                       1.801052
rf2xCO2 (W / m^2)             3.930000
tau1                          3.424102
tau2                        285.003478
alpha1                        0.129601
alpha2                        0.001121
Name: recommendation, dtype: float64
```

[3]:
```python
tau1_irf =med_irf['tau1']#4.1  # 8.4
tau2_irf = med_irf['tau2']#249.  # 409.5
```

```python
alpha1_irf = med_irf['alpha1']#c1_irf / tau1_irf
alpha2_irf =  med_irf['alpha2']#c2_irf / tau2_irf
print(tau1_irf, tau2_irf)
print(alpha1_irf,alpha2_irf)
```

```
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
```

[4]:
```python
#from useful_scit.imps import (np, plt, sns, pd)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# constants:


def delta_T(t, r0_rf, tau_rf, tau1_irfc=tau1_irf, tau2_irfc=tau2_irf,
            alpha1_irfc=alpha1_irf, alpha2_irfc=alpha2_irf):
    """
    computes analytic solution to
        delta_T=integral_0^t RF(t')*IRF(t-t') dt'
    :param t: time [years]
    :param r0_rf: R0 in RF(t)
    :param tau_rf: tau for RF
    :param tau1_irfc: tau1 for irf climate
    :param tau2_irfc: tau2 for irf climate
    :param alpha1_irfc: alpha1 for irf climate
    :param alpha2_irfc: alpha2 for irf climate
    :return:
    """
    # compute beta:
    print(tau1_irfc,tau2_irfc, )
    print(alpha1_irfc, alpha2_irfc)
    beta1 = beta(tau_rf, tau1_irfc)
    beta2 = beta(tau_rf, tau2_irfc)
    # compute integrals:
    I1 = alpha1_irfc * tau1_irfc * (1 - np.exp(-t / tau1_irfc)) + alpha2_irfc *␣
 ↪tau2_irfc * (1 - np.exp(-t / tau2_irfc))
    I2 = -(alpha1_irfc * beta1 * (np.exp(-t / tau_rf) - np.exp(-t / tau1_irfc))␣
 ↪+ alpha2_irfc * beta2 * (
            np.exp(-t / tau_rf) - np.exp(-t / tau2_irfc)))
    return r0_rf * tau_rf * (I1 - I2)


def RF(t, R0_rfco2, tau_rfco2):
    """
    Approximated radiative forcing from pulse emission of
```

```
     SLCFer with atmospheric lifetime tau and initial RF=R0_rf
    :param t: time [years]
    :param R0_rfco2: initial radiative forcing of SLCFer
    :param tau_rfco2: Atmospheric lifetime of component
    :return: Radiative forcing.
    """
    return R0_rfco2 * tau_rfco2 * (1 - np.exp(-t / tau_rfco2))


def beta(tau_rf, tau_i_irf):
    """
    Helping function
    beta_i = (1/rau_rf -1/tau_i_irf)**(-1)
    :param tau_rf: tau forcing agent
    :param tau_i_irf: tau climate irf
    :return: value
    """
    return (1 / tau_rf - 1 / tau_i_irf) ** (-1)
```

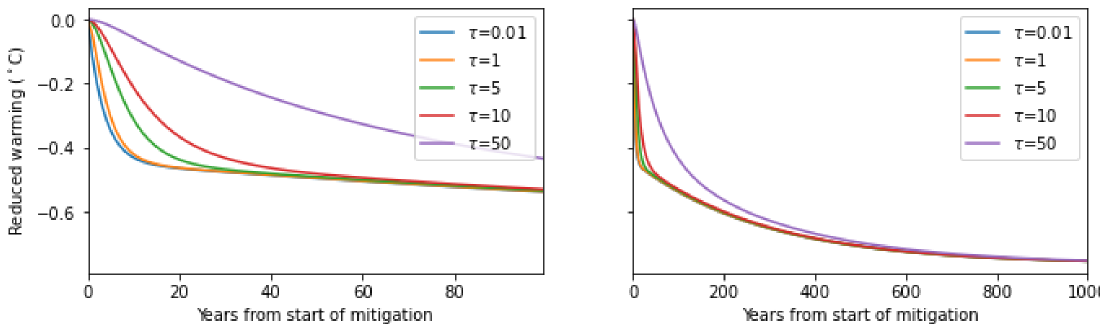
### 1.3.1 Reduced warming for SLCF with different atmospheric lifetimes

[5]:
```
#sns.reset_orig()
#sns.set_palette('deep')
#sns.set_style("whitegrid")
# define various atmospheric lifetimes:
tau_values = [0.01,  1, 5, 10, 50]
# time in years:
t_yrs = np.linspace(0, 1000, 1000)
# dataframe to hold results:
df = pd.DataFrame(index=t_yrs)
df.index.name = 'Years from start of mitigation'
# compute delta T for each gas:
for tau in tau_values:
    df['$\\tau$=%s' % tau] = delta_T(t_yrs, -1 / tau, tau)
# plot:
fig, axs = plt.subplots(1,2, figsize=[11, 3], sharey=True)
tmaxs=[100, 1000]
for ax, tmax in zip(axs, tmaxs):
    df.loc[0:tmax].plot.line(ax=ax)#, cmap=sns.palettes.deep)

    ax.set_ylabel('Reduced warming ($^\circ$C)')
```

```
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
```

```
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
3.4241020923110033 285.0034778419114
0.12960119672831902 0.0011213584204743735
```




## 2 Including ΔT for $CO_2$:

$$"\Delta E" = \Delta E\cdot$$

Gjeldende $CO_2$ nivÃě. IPCC – AR5. Chap 8 Supplementary. 8.SM.11.3.1.

- Sjekke responsfunksjonene og oppdatere! CMIP5/
- Side 16.

#### 2.0.1 Calculate $\Delta E$:

We choose to plot Δ T for $CO_2$ for an emission change giving us $\mathrm{RF}_{CO_2}(t = 100yrs) = -1\mathrm{W/m}^2$. Calculate $\Delta E$ such that $\mathrm{RF}_{CO_2}(t = 100yrs) = -1\mathrm{W/m}^2$

$$IRF_{CO_2} = a_0 + \sum_{i=1}^{3} a_i e^{-t/\tau_i}$$

Here we assume that the forcing can be approximated as a factor times the concentration (calculated in the integral), while in reality it would be a logarithmic relationship.

$$RF_{CO_2}(t) = \int_0^t \Delta E\big(IRF_{CO_2}(t')\big)dt'$$

Calculate the integral:

$$\int_0^t IRF_{CO_2}(t')d_t = \int_0^t a_0 + \sum_{i=1}^{3} a_i e^{-t'/\tau_i} dt'$$
$$= a_0 t + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-t/\tau_i})$$

$$RF_{CO_2}(t) = \Delta E\big(a_0 t + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-t/\tau_i}\big)$$

Let $RF_{CO_2}(t = 100) = -1\text{W}/\text{m}^2$ and solve for $\Delta E$:

$$RF_{CO_2}(100) = \int_0^{100} \Delta E\big(IRF_{CO_2}(t')\big)dt'$$
$$-1 = \Delta E\Big(a_0 \cdot 100 + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-100/\tau_i})\Big)$$
$$\Delta E = -\Big(a_0 \cdot 100 + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-100/\tau_i})\Big)^{-1}$$

Values taken from Joos et al (2013) (which are the same as AR5).

#### 2.0.2 Compute $\Delta E$

[6]:
```
a_i_irfco2 = [0.2173, 0.2240, 0.2824, 0.2763]
tau_i_irfco2 = [394.4, 36.54, 4.304]

def IRF_CO2(t, alphs_co2 = a_i_irfco2, taus_co2 = tau_i_irfco2):
    a_0 = alphs_co2[0]
    a_is = alphs_co2[1:]  # a1, a2, a3
    tau_is = taus_co2  # tau1, tau2, tau3
    sum_ = 0.
    for ai, taui in zip(a_is, tau_is):  # in a_vals[1:]:
        sum_ += ai *np.exp(-t / taui)
    return a_0 + sum_

def delta_E(RF_t = -1, t = 100., alphs_co2 = a_i_irfco2, taus_co2 =␣
 ↪tau_i_irfco2):
    """
    Calclulates the change in emissions that gives RF_t after t years.
    :param RF_t:
    :param t:
    :param alphs_co2:
    :param taus_co2:
```

```
    :return: d_E
    """
    a_0 = alphs_co2[0]
    a_is = alphs_co2[1:]  # a1, a2, a3
    tau_is = taus_co2  # tau1, tau2, tau3
    sum_ = 0.
    for ai, taui in zip(a_is, tau_is):  # in a_vals[1:]:
        sum_ += ai * taui * (1 - np.exp(-t / taui))
    return RF_t / (a_0 * t + sum_)
```

[7]:
```
delta_E()
```

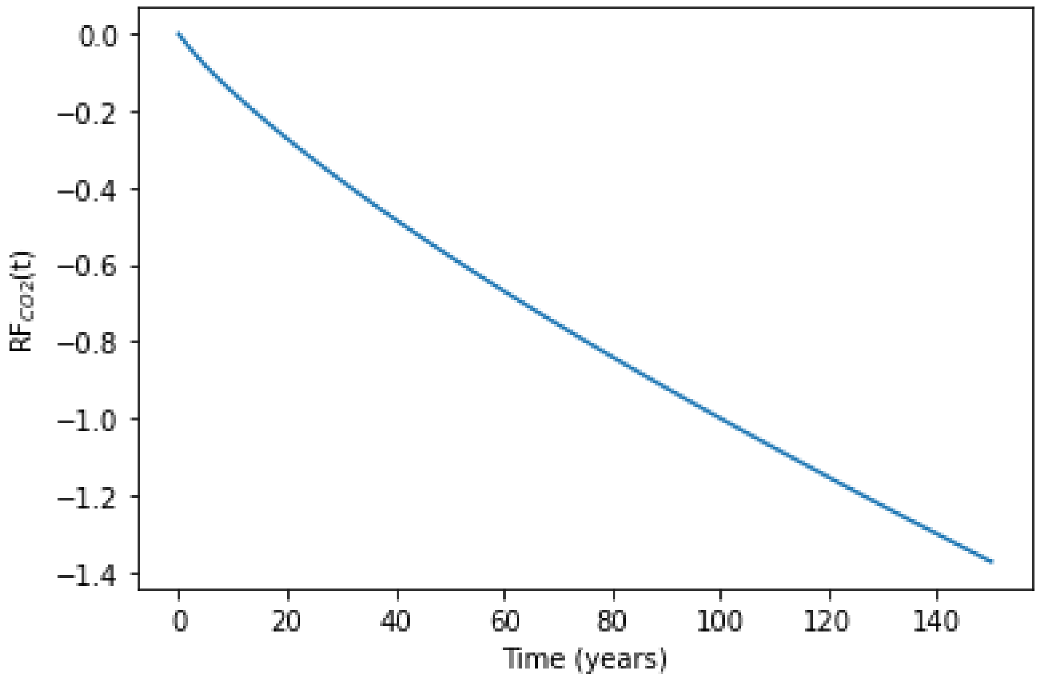
[7]: -0.019100230698874933

$$RF_{CO_2}(t) = \Delta E\big(a_0 t + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-t/\tau_i}\big)$$

**Plot of RF for CO2**

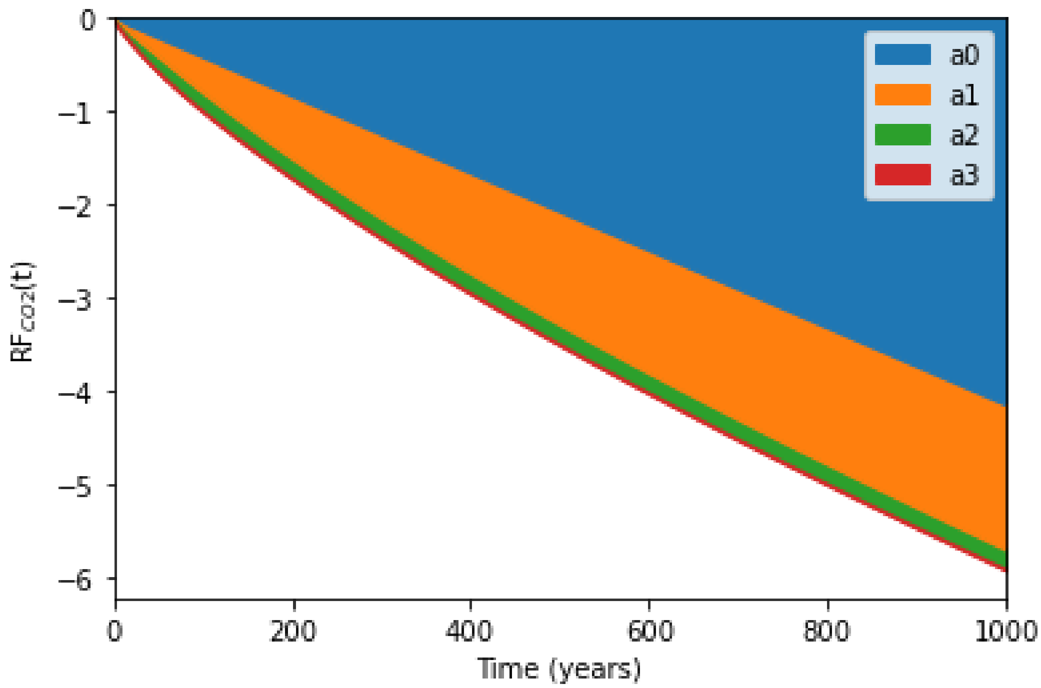
[8]:
```
def RF_CO2(t, dE, alphs_co2 = a_i_irfco2, taus_co2 = tau_i_irfco2):
    """
    Radiative forcing co2
    :param t: time [years]
    :param dE: change in emissions
    :param alphs_co2: alpha values co2 irf
    :param taus_co2: tau values co2 irf
    :return:
    """
    a0 = alphs_co2[0]
    ais = alphs_co2[1:]
    taus = taus_co2
    # sum:
    _s = 0
    for ai, taui in zip(ais, taus):
        _s += ai * taui * (1 - np.exp(-t / taui))
    return dE * (a0 * t + _s)


t_yrs = np.linspace(0, 150, 100)
plt.plot(t_yrs, RF_CO2(t_yrs, delta_E()))
plt.xlabel('Time (years)')
plt.ylabel('RF$_{CO2}$(t)')
plt.show()
```




### 2.1 Contribution of the terms in the IRF$_{CO_2}$

[9]:
```
t_yrs = np.linspace(0, 1000, 500)
_df = pd.DataFrame(index=t_yrs)
for i in range(4):
    a_vals = np.zeros(4)
    a_vals[i] = a_i_irfco2[i]
    _df['a%s' % i] = RF_CO2(t_yrs, delta_E(), alphs_co2=a_vals)
    # plt.plot(t_yrs, RF_CO2(t_yrs, delta_E(), a_vals=a_vals), label='a%s'%i)
_df[::-1].plot.area()  # stacked=True)
plt.xlabel('Time (years)')
plt.ylabel('RF$_{CO2}$(t)')
plt.legend()
plt.show()
```




## 2.2 Calculate $\Delta T$ for CO2

If we follow the same logic as above, we will now get: Instantanious response function:

$$IRF_{climate}(t) = \alpha_1 e^{-t/\tau_1} + \alpha_2 e^{-t/\tau_2}$$

and:

$$\begin{aligned}
\Delta T_{CO_2}(t) &= \int_0^t RF_{CO_2}(t') \cdot IRF_{Climate}(t-t')dt' \\
&= \int_0^t \Delta E\Big(a_0 t + \sum_{i=0}^{3} a_i \tau_i (1 - e^{-t'/\tau_i})\Big) \cdot \Big(\alpha_1 e^{-(t-t')/\tau_{1,c}} + \alpha_2 e^{-(t-t')/\tau_{c,2}}\Big)dt' \\
&= \Delta E \int_0^t a_0 t (\sum_{j=1}^{2} \alpha_j e^{-(t-t')/\tau_{j,c}})dt + \Delta E \int_0^t \Big(\sum_{i=1}^{3} a_i \tau_i (1 - e^{-t'/\tau_i})\Big) \cdot \Big(\sum_{j=1}^{2} \alpha_j e^{-(t-t')/\tau_{j,c}}\Big)dt \\
&= \Delta E\big(I_1 + I_2\big)
\end{aligned}$$

**Integral 1**

$$I_1 = \int_0^t a_0 t (\sum_{j=1}^{2} \alpha_j e^{-(t-t')/\tau_{j,c}})dt$$

Let $u = t$ and $v' = \sum_{j=1}^{2} \alpha_j e^{-(t-t')/\tau_{j,c}}$. Then by integration by parts

$$I_1 = \int_0^t a_0 t(\sum_{j=1}^{2} \alpha_j e^{-(t-t')/\tau_{j,c}})dt$$
$$= a_0\Big(t\sum_{j=1}^{2}\alpha_j\tau_{j,c} - \sum_{j=1}^{2}\alpha_j\tau_{j,c}^2(1-e^{-t/\tau_{j,c}})\Big)$$

**Integral 2:**

$$I_2 = \int_0^t \Big(\sum_{i=1}^{3} a_i\tau_i(1-e^{-t'/\tau_i})\Big)\cdot\Big(\sum_{j=1}^{2}\alpha_i e^{-(t-t')/\tau_{i,c}}\Big)dt$$
$$= \sum_{i=1}^{3}\sum_{j=1}^{2}\int_0^t a_i\tau_i(1-e^{-t'/\tau_i})\alpha_j e^{-(t-t')/\tau_{j,c}}dt$$
$$= \sum_{i=1}^{3}\sum_{j=1}^{2} a_i\tau_i\alpha_j\tau_{j,c}\Big(1-e^{-t/\tau_{j,c}}+\frac{\tau_i}{\tau_{j,c}-\tau_i}\big(e^{-t/\tau_i}-e^{-t/\tau_{j,c}}\big)\Big)$$

**Solution: $\mathbf{I_1}$+$\mathbf{I_2}$:**

$$\Delta T_{CO_2}(t) = \int_0^t RF_{CO_2}(t')\cdot IRF_{Climate}(t-t')dt'$$
$$= \Delta E\big(I_1+I_2\big)$$
$$= \Delta E\Big(a_0\big[t\sum_{j=1}^{2}\alpha_i\tau_{j,c} - \sum_{j=1}^{2}\alpha_j\tau_{j,c}^2(1-e^{-t/\tau_{j,c}})\big] + \sum_{i=1}^{3}\sum_{j=1}^{2}a_i\tau_i\alpha_j\tau_{j,c}\Big(1-e^{-t/\tau_{j,c}}+\frac{\tau_i}{\tau_{j,c}-\tau_i}\big(e^{-t/\tau_i}-e^{-t/\tau_{j,c}}\big)\Big)\Big)$$

### 2.2.1 Compute integral:

Values taken from the table below:

**Table 5.** Coefficients to fit multi-model mean responses to a pulse emission of 100 GtC following Equation 11 in the main text and for $0 < t < 1000$ yr. The mean relative error of the fit is given in percent. The error is calculated from annual values as the average of the absolute differences between fit ($f$) and multi-model mean ($m$) divided by the multi-model mean ($1/N\sum(m-f)/m$). Multiplication by $(12/(100\times 44\times 10^{12}))$ yields the change per kg-$CO_2$ for ocean and land carbon storage, surface air temperature (SAT), time-integrated SAT (iSAT), steric sea level rise (SSLR), and ocean heat content (OHC). The timescales $\tau_i$ are given in years and units of $a_i$ are indicated in parentheses in the first column.

| | rel. error | $a_0$ | $a_1$ | $a_2$ | $a_3$ | $\tau_1$ | $\tau_2$ | $\tau_3$ |
|---|---|---|---|---|---|---|---|---|
| $IRF_{CO_2}$ | 0.6 | 0.2173 | 0.2240 | 0.2824 | 0.2763 | 394.4 | 36.54 | 4.304 |
| Ocean (GtC) | 0.6 | 60.29 | −26.48 | −17.45 | −16.35 | 390.5 | 100.5 | 4.551 |
| Land (GtC) | 1.3 | 17.07 | 332.1 | −334.1 | −15.09 | 74.76 | 70.31 | 6.139 |
| SAT (°C) | 1.8 | 0.1383 | 0.05789 | −0.06729 | −0.1289 | 264.0 | 5.818 | 0.8062 |
| iSAT (°C yr) | 1.8 | 3934 | −4432 | 777.7 | −280.0 | 16 080 | 2294 | 1144 |
| SSLR (cm) | 1.5 | 5.259 | −3.789 | −0.9351 | −0.5350 | 581.7 | 75.71 | 5.963 |
| OHC ($10^{22}$ J) | 1.0 | 42.63 | −32.86 | −6.589 | −3.182 | 420.4 | 54.82 | 6.340 |

Ref: Values taken from [1]

### 2.2.2 Code for calculating $\Delta$ $T_{CO_2}$

[10]:
```python
alphas_irfco2 = [0.2173, 0.2240, 0.2824, 0.2763]
taus_irfco2 = [394.4, 36.54, 4.304]


# def delta_T(t, r0, tau, tau1=tau1, tau2=tau2, alpha1=c1/tau1, alpha2=c2/tau2):
def deltaT_co2(t, dE, alphs_co2=alphas_irfco2, taus_co2=taus_irfco2,␣
 ↪alphs_irfc=[alpha1_irf, alpha2_irf], taus_ifrc=[tau1_irf, tau2_irf]):
    """
    Calculates delta T for CO2 based on analytic solution
    :param t: time
    :param dE: change in emissions
    :param alphs_co2: alpha values co2 IRF
    :param taus_co2: tau values co2 IRF
    :param alphs_irfc: alpha values climate IRF
    :param taus_ifrc: tau values climate IRF
    :return:
    """
    a0_c = alphs_co2[0]
    int1 = I1(t, a0=a0_c, alphajs_k=alphs_irfc, taujs_k=taus_ifrc)
    int2 = I2(t, alphs_co2=alphs_co2, taus_co2=taus_co2, alphs_irfc=alphs_irfc,␣
 ↪taus_irfc=taus_ifrc)
    return dE * (int2 + int1)


def I2(t, alphs_co2 = alphas_irfco2, taus_co2 = taus_irfco2,
       alphs_irfc=[alpha1_irf, alpha2_irf], taus_irfc=[tau1_irf, tau2_irf]):
    """
    Integral part 2
    :param t: time
    :param as_co2: alpha values co2 IRF
    :param taus_co2: tau values co2 IRF
    :param alphs_irfc: alpha values climate IRF
    :param taus_irfc: tau values climate IRF
    :return: value
    """
    ais_c = alphs_co2[1:]
    tauis_c = taus_co2[:]
    alphajs_k = alphs_irfc[:]
    taujs_k = taus_irfc[:]  # t,a0 = as_co2[0], alphajs_k = [alpha1_k,␣
 ↪alpha2_k], taujs_k = [tau1_k, tau2_k]):
    _s1 = 0
    for taui, ai in zip(tauis_c, ais_c):
        for tauj, alphaj in zip(taujs_k, alphajs_k):
            _rat = taui / (tauj - taui)
```

```
            _d = 1 - np.exp(-t / tauj) + _rat * (np.exp(-t / taui) - np.exp(-t /␣
↪tauj))
            _s1 += ai * taui * alphaj * tauj * _d
    return _s1


def I1(t, a0=alphas_irfco2[0], alphajs_k=[alpha1_irf, alpha2_irf],␣
↪taujs_k=[tau1_irf, tau2_irf]):
    """
    Integral part 1
    :param t: years
    :param a0: a0 co2 irf
    :param alphajs_k: alpha values climate IRF
    :param taujs_k: tau  values climate IRF
    :return: value of integral
    """
    _s1 = 0
    for alphaj, tauj in zip(alphajs_k, taujs_k):
        _s1 += alphaj * tauj
    _s2 = 0
    for alphaj, tauj in zip(alphajs_k, taujs_k):
        _s2 += alphaj * tauj ** 2 * (1 - np.exp(-t / tauj))

    return a0 * (t * _s1 - _s2)
```

**Check correctness of integral**

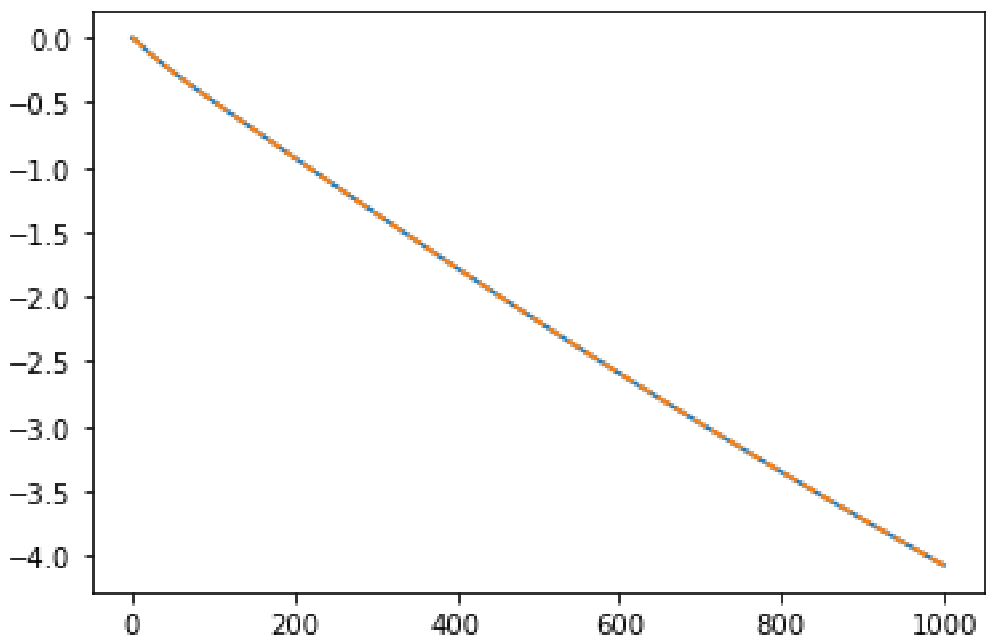
[11]:
```
from scipy.integrate import quad


def integrand(t, tn, dE, alphs_co2=alphas_irfco2, taus_co2=taus_irfco2,␣
↪a_k=[alpha1_irf, alpha2_irf], tau_k=[tau1_irf, tau2_irf]):
    """
    The integrand:
    RF_{CO_2}(t') * IRF_{Climate}(t-t')
    :param t: t'
    :param tn: t limit
    :param dE: delta Emissions
    :param alphs_co2: alpha values co2
    :param taus_co2: tau values co2
    :param a_k: alpha values irf climate
    :param tau_k: tau values irf climate
    :return: value of integrand
    """
    a0_c = alphas_irfco2[0]
    ais_c = alphs_co2[1:]
```

```
    tauis_c = taus_co2[:]
    alphajs_k = a_k[:]
    taujs_k = tau_k[:]
    _s = 0
    for taui, ai in zip(tauis_c, ais_c):
        _s += ai * taui * (1 - np.exp(-t / taui))
    fact1 = a0_c * t + _s
    fact2 = alphajs_k[0] * np.exp(-(tn - t) / taujs_k[0]) + alphajs_k[1] * np.
 ↪exp(-(tn - t) / taujs_k[1])
    return dE * (fact1 * fact2)


integ = []

for t in t_yrs:
    # Numerically integrate:
    integ.append(quad(integrand, 0, t, args=(t, delta_E()))[0])
plt.plot(t_yrs, integ)
y = deltaT_co2(t_yrs, delta_E())
plt.plot(t_yrs, y, linestyle='dashed')
plt.show()
```



# 3 Final combined plots:

[12]:
```
def set_fontsize(ax, SM=8, MED=10, BIG=12):
    # ax.title.set_fontsize(SM)
    for item in [ax.title, ax.xaxis.label, ax.yaxis.label]:
        item.set_fontsize(SM)
    ax.title.set_fontsize(SM)

    for item in (ax.get_xticklabels() + ax.get_yticklabels()):
        item.set_fontsize(SM)
```

[13]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import TransformedBbox, BboxPatch,␣
 ↪BboxConnector
import numpy as np
```

[14]:
```
def mark_inset(parent_axes, inset_axes, loc1a=1, loc1b=1, loc2a=2, loc2b=2,␣
 ↪**kwargs):
    rect = TransformedBbox(inset_axes.viewLim, parent_axes.transData)

    pp = BboxPatch(rect, fill=False, **kwargs)
    parent_axes.add_patch(pp)

    p1 = BboxConnector(inset_axes.bbox, rect, loc1=loc1a, loc2=loc1b, **kwargs,



                      )
    inset_axes.add_patch(p1)
    p1.set_clip_on(False)
    p2 = BboxConnector(inset_axes.bbox, rect, loc1=loc2a, loc2=loc2b, **kwargs)
    inset_axes.add_patch(p2)
    p2.set_clip_on(False)

    return pp, p1, p2

#mark_inset(ax, axins, loc1a=1, loc1b=4, loc2a=2, loc2b=3, fc="none",␣
 ↪ec="crimson")
```

[15]:
```
import matplotlib.pyplot as plt

from mpl_toolkits.axes_grid1.inset_locator import zoomed_inset_axes
#from mpl_toolkits.axes_grid1.inset_locator import mark_inset

import numpy as np
```

```
#fig, ax_short = plt.subplots(figsize=[5,2])
fig, ax_short = plt.subplots( figsize=[6, 6], dpi=150)



#ax.plot(x, y)

ax_long = ax_short.inset_axes([.0, -.6, .6, .4], facecolor='w', frameon=True)#[.
↪8,.6,.4,.4])#zoomed_inset_axes(ax_long, 20, loc='center right',␣
↪bbox_to_anchor=(1,1)) # zoom = 6

#ax_long = plt.axes([.65, .6, .3, .25], facecolor='w', frameon=True)  # ,␣
↪fontsize=10)

for ax in [ax_short, ax_long]:
    # plot standard agents
    df.plot.line(ax=ax, legend=False, cmap='viridis')#, linewidth=linewid[ax])
    y = deltaT_co2(t_yrs, delta_E())  # a_vals=_a, t=1e100))
    ax.plot(t_yrs, y, label='CO$_2$', c='k', linestyle='dashed')
    #, linewidth=linewid[ax])
#axins.plot(x, y)
ax_short.set_xlim([0, 100]) # Limit the region for zoom
ax_short.set_ylim([-.55, .2])
ax_long.set_xlim([-1,400]) # Limit the region for zoom
ax_long.set_ylim([-2, .22])

## draw a bbox of the region of the inset axes in the parent axes and
## connecting lines between the bbox and the inset axes area
mark_inset(ax, ax_short, loc1b=1, loc1a=4, loc2b=2, loc2a=3,
           fc="none",
           #linewidth=2,
           ec="0.5",
           zorder=-1)#, edgecolor='r')

ax_short.legend(title='Lifetime',frameon=False, ncol =2)#, bbox_to_anchor=(.6,.
↪8))

#plt.draw()
#ax_long.get_legend().remove()  # legend(visible=False)
#ax_short.set_ylabel('Reduced warming ($^\circ$C)')
ax_long.set_ylabel('($^\circ$C)')
ax_short.set_ylabel('($^\circ$C)')
ax_short.set_title('Reduced warming after abrupt decrease in SLCFs and CO$_2$␣
↪emissions')
ax_short.set_title('Cooling after abrupt reduction in SLCFs and CO$_2$␣
↪emissions')
```

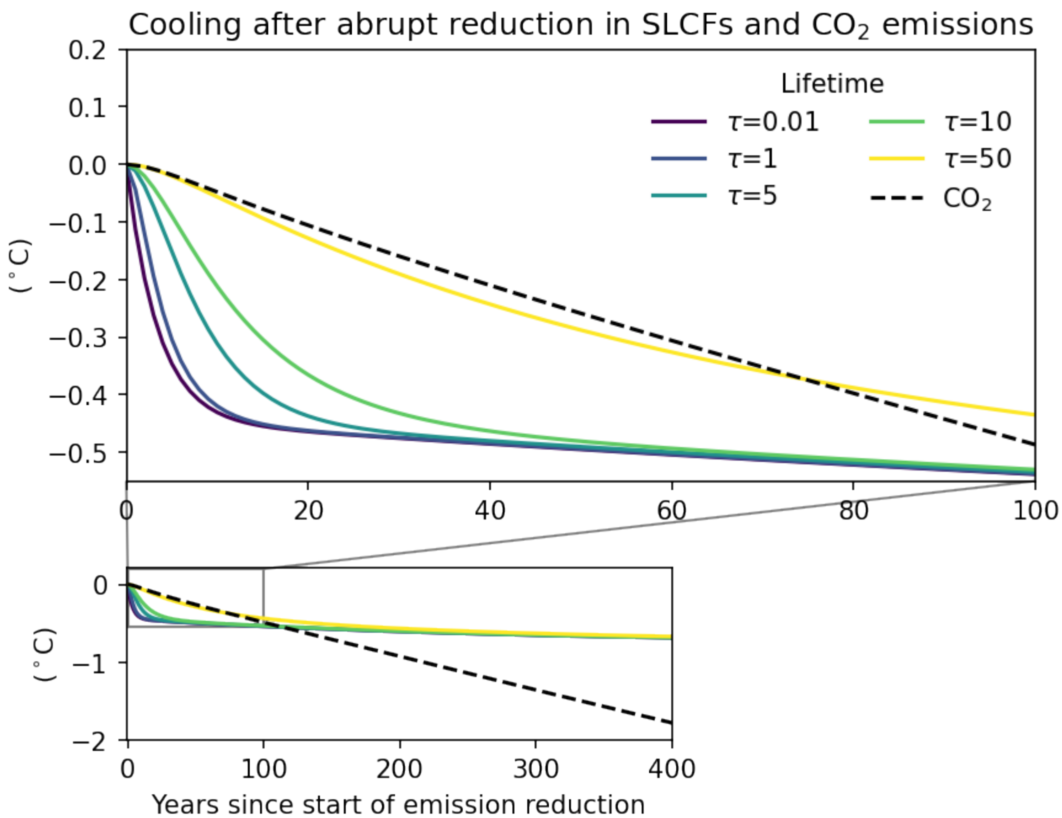
```
#fig.suptitlele('Reduced warming after abrupt decrease in SLCFs and CO$_2$␣
 ↪emissions')
#fig.suptitle('Reduced warming from mitigation of SLCFs and CO$_2$')
ax_long.set_xlabel('Years since start of emission reduction')
ax_short.set_xlabel('')#Years since start of emission decrease')
#ax_short.spines['right'].set_visible(False)
#ax_short.spines['top'].set_visible(False)
#ax_long.spines['right'].set_visible(False)
#ax_long.spines['top'].set_visible(False)
#mark_inset(ax, ax_short, loc1=2, loc2=4, fc="none", ec="0.5")
ax_long.patch.set_alpha(0.)
#ax_long.text(10,-1.5,'Long term future', alpha=.7)

fig.tight_layout()

fname = 'plots/embedded_long_time.'

#fig.savefig(fname, dpi=300)
#fig.subplots_adjust(bottom=.1)

fig.savefig(fname+'pdf', dpi=300,  bbox_inches = 'tight',␣
 ↪bbox_extra_artists=(ax_long,))
fig.savefig(fname+'png', dpi=300,  bbox_inches = 'tight',␣
 ↪bbox_extra_artists=(ax_long,))
plt.show()
```




# References


[1] F. Joos, R. Roth, J. S. Fuglestvedt, G. P. Peters, I. G. Enting, W. von Bloh, V. Brovkin, E. J. Burke, M. Eby, N. R. Edwards, T. Friedrich, T. L. FrÃűlicher, P. R. Halloran, P. B. Holden, C. Jones, T. Kleinen, F. T. Mackenzie, K. Matsumoto, M. Meinshausen, G.-K. Plattner, A. Reisinger, J. Segschneider, G. Shaffer, M. Steinacher, K. Strassmann, K. Tanaka, A. Timmermann, and A. J. Weaver. Carbon dioxide and climate impulse response functions for the computation of greenhouse gas metrics: a multi-model analysis. *Atmospheric Chemistry and Physics*, 13(5):2793–2825, March 2013.