Exercise 3, TFY4235 Computational physics

Martin Johnsrud

Introduction

This is an implementation of [1].

Theory and implementation

The diffusion equation, can be written as

$$\Delta t \frac{\partial}{\partial t} C(z,t) = \Delta t \left(K(z) \frac{\partial^2}{\partial z^2} + \frac{\mathrm{d}K(z)}{\mathrm{d}z} \frac{\partial}{\partial z} \right) C(z,t) = \mathcal{D} C(z,t).$$

Discretizing the spatial part, and applying boundary conditions, gives

$$\Delta t \frac{\partial}{\partial t} C_n(t) = \mathcal{D}_{nm} C_n(t) + S_n(t),$$

where

$$\mathcal{D} = \begin{pmatrix} -4\alpha K_0 - 2\Gamma & 4\alpha K_0 & 0 & \dots & 0 \\ -\frac{\alpha}{2} K_1' + 2\alpha K_1 & -4\alpha K_1 & \frac{\alpha}{2} K_1' + 2\alpha K_1 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & -\frac{\alpha}{2} K_{N-1}' + 2\alpha K_{N-1} & -4\alpha K_{N-1} & \frac{\alpha}{2} K_{N-1}' + 2\alpha K_{N-1} \\ 0 & \dots & 0 & 4\alpha K_N & -4\alpha K_N \end{pmatrix},$$

$$S(t) = \begin{pmatrix} 2\Gamma C_{\mathrm{eq}}(t) & 0 & \dots & 0 \end{pmatrix}^T \quad \Gamma = 2\frac{\alpha k_w \Delta z}{K_0} \left(K_0 - \frac{1}{2}(-\frac{3}{2}K_0 + 2K_1 - \frac{1}{2}K_2) \right), \quad \alpha = \frac{\Delta t}{2\Delta z^2},$$

$$K_n' = K_{n+1} - K_{n-1}$$

The Cranck-Nichelson scheme then yields

$$C_n^{i+1} = C_n^i + \frac{1}{2}(\mathcal{D}_{nm} C_m^i + S_n^i) + \frac{1}{2}(\mathcal{D}_{nm} C_m^{i+1} + S_n^{i+1}),$$

so the equation to be solved to get the next timestep is

$$A_{nm} C_m^{i+1} = V_n^i,$$
$$V_n^i = \left(\delta_{nm} + \frac{1}{2}\mathcal{D}_{nm} \right) C_m^i + \frac{1}{2}(S_n^i + S_n^{i+1}), \quad A_{mn} = \left(\delta_{nm} - \frac{1}{2}\mathcal{D}_{nm} \right)$$

The implementation of this system of equation uses SciPy's sparse matrix library. After creating sparse realizations of A, SciPy's `splu` is used to generate the LU decomposition `LU` of A. This is an object with methods such as `.solve()`, which utilizes the LU decomposition. The wrapper `simulate` then loops over $N_t - 1$ steps, using `solve(V) = lambda V: LU.solve(V)`, where `V` is as given above.

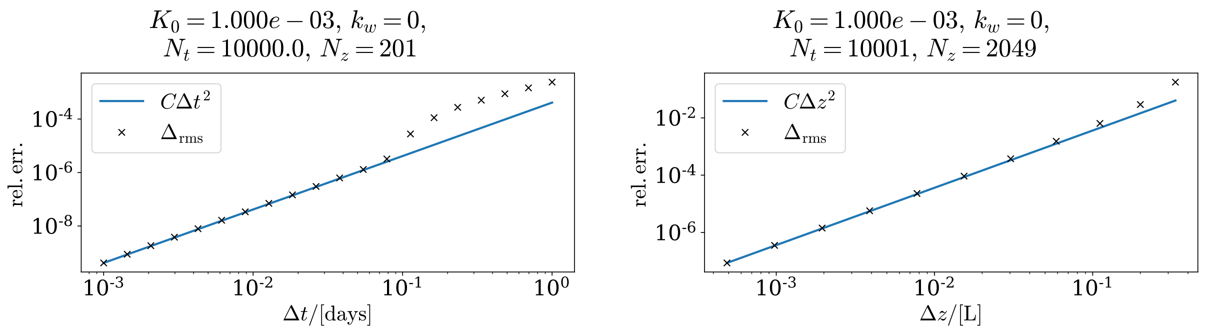


Figure 1: Error, measured as the root mean square deviation from a reference value, after 1 day simulation of an Gaussian initial concentration.

The results of the simulation of C is kept in a NumPy array. However, it is not necessary to keep all steps made in time. `simulate` can be passed a integer argument `save`,

To check convergence of the method, both in time and space, a way of measuring error is needed. In this project, relative root mean square (RMS) error is used. Given a concentration $C(z)$ and a reference $C_0(z)$ at a given time, sampled at a set of points $\{z_i\}_{i=1}^N \in [0, L]$, relative RMS error is given by

$$\Delta_{\mathrm{rms}} = \sqrt{\frac{1}{N}\sum_i \left(\frac{C(z_i) - C_0(z_i)}{C_0(z_i)}\right)^2}.$$

When comparing concentrations with different values for N_z, as is necessary to find the convergence of the method in space, one must make sure that the points of comparison are the same for both C and C_0. This is done by setting $N_z = 5 \cdot 2^N a + 1$ for C_0, and $N_z = 2^n a + 1$. Here, a and N are integers, and n is some integer less or equal to N. This ensures that the set of points $\{z_i\}_{i=1}^{2^n a+1}$ that $C(z)$ is simulated on, $C_0(z)$ is as well, and they can be compared. This is implemented by `get_rms` in `utillities.py`, which takes a list of arrays and compare each with the last element.

Tests

Make sure the implementation gives good answers, it is compared to known solutions. The method used in this implementation has quadratic convergence, both in time and space. A convergence test was implemented for a simple test case, to check this. Figure 1 shows the result of this.

A constant concentration of CO_2 should remain constant, regardless of $K(z)$, as long as it is positive. This test is shown in Figure 2, with a both a constant $K(z)$, and a smooth step function between two values of K. For both, the constant initial concentration is close to unchanged, save for numerical errors.

The systems should also, given $k_w = 0$, conserve mass. To test this, a initial distribution of two Gaussian functions were evolved in time, with a non-constant diffusivity. The result is shown in Figure 3, where the mass is conserved to a good approximation.

A sharply peaked Gaussian package should have a variance that increases linearly with time, then approach a steady state. This is shown in Figure 4.

Figure 5 shows the depletion of CO_2 from the ocean, given a zero partial pressure in the atmosphere. For a small Biot number, this should follow a exponential decay, as this test shows.

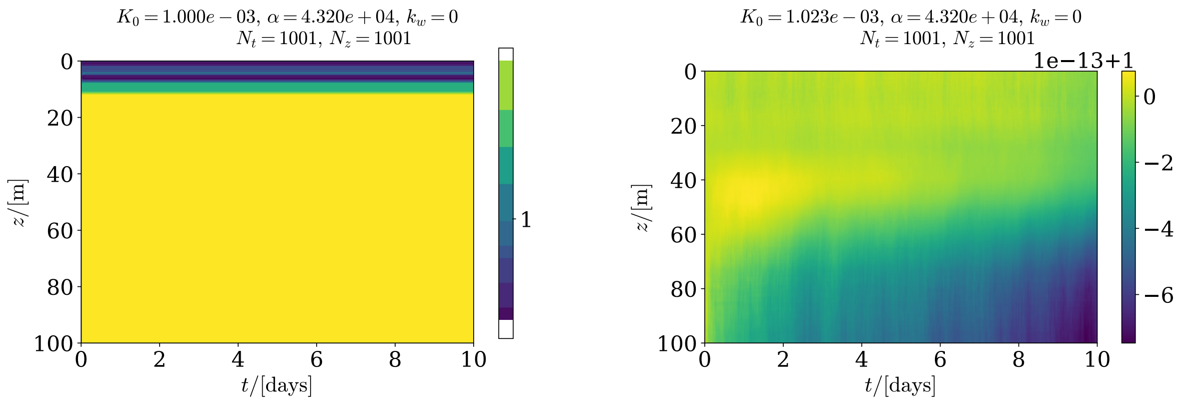


Figure 2: The time evolution of an constant concentration. The system on the left has a constant $K(z)$, while system of the right has an oscillatory K. The largest deviation of the system is of order 10^{-15} and 10^{-13}, respectively.

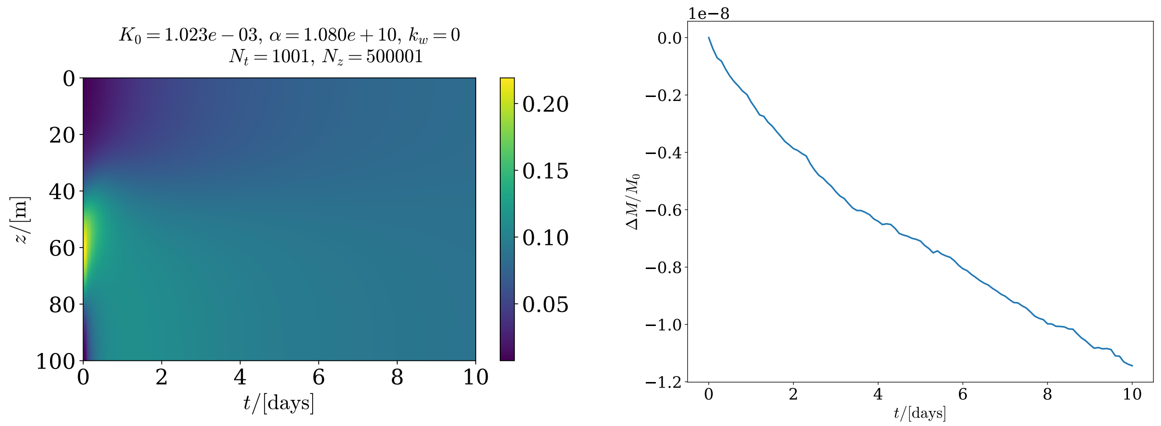


Figure 3: The evolution of a Gaussian distribution of CO_2 is shown on the left. On the right, the relative change in mass as a function of time is plotted.

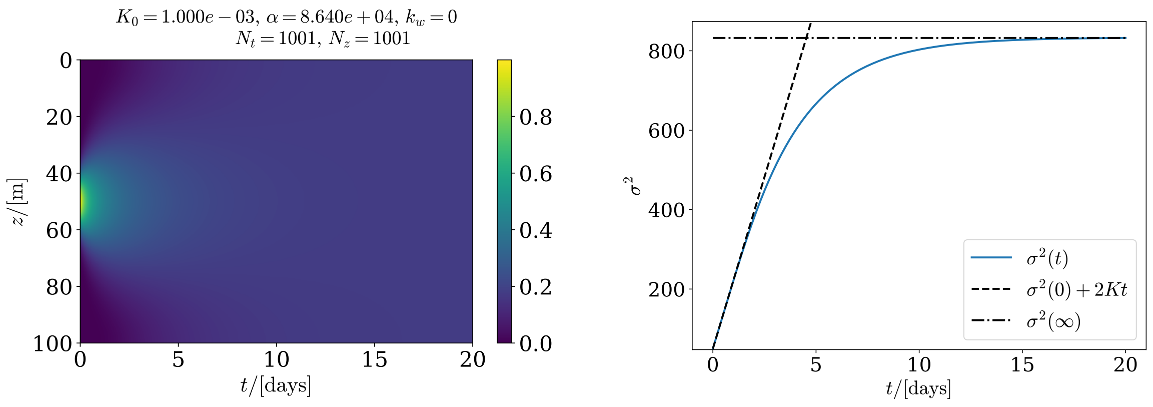


Figure 4: A system with an initial Gaussain distribution is simulated. On the right, the variance as a function of time is shown. The increase is initially constant, but then approaches a steady state.

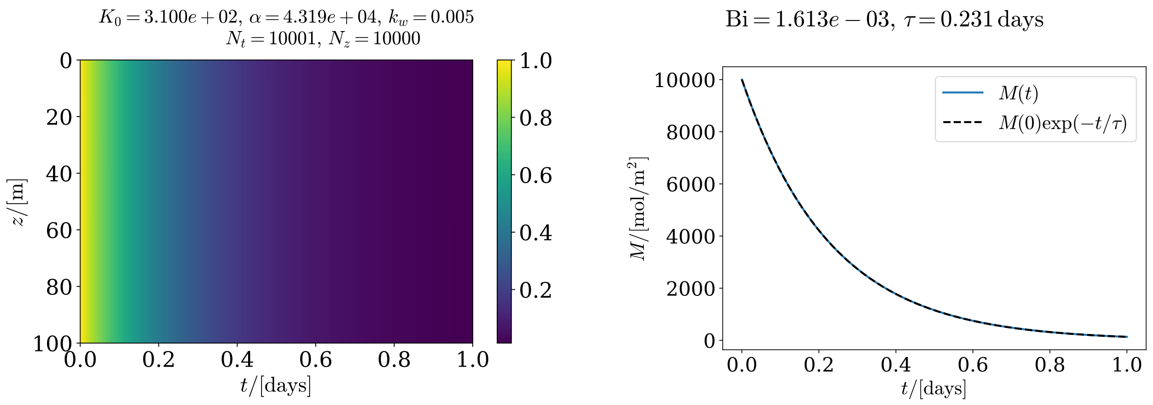


Figure 5: The slow removal of CO_2 from the ocean, when the atmosphere contains a partial pressure of 0.

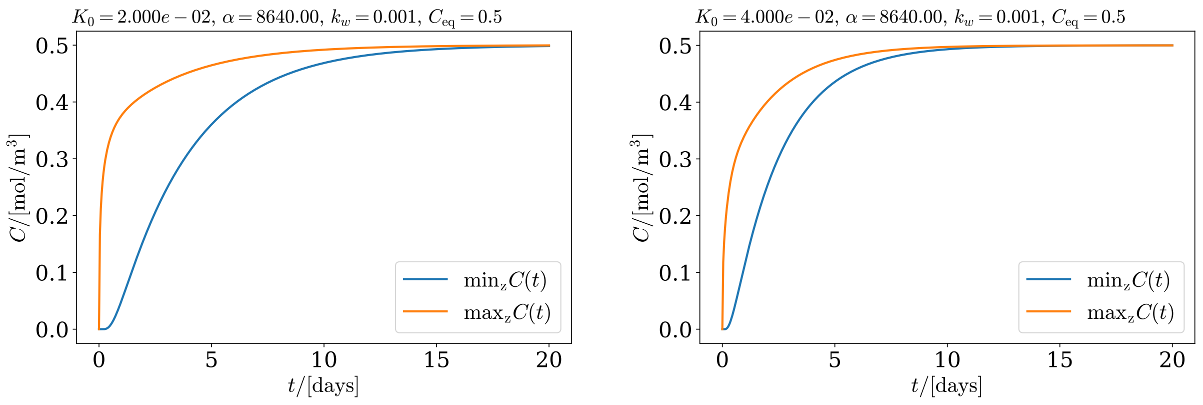


Figure 6: Equilibration of the ocean with a atmosphere with $C_{\mathrm{eq}} = 0.5$. The figure on the left is a system with constant diffusivity, the left side has a oscillating but always positive diffusivity.

Lastly, Figure 6 shows how the ocean reaches a equilibrium with the atmosphere, given a non-zero, positive mass-transfer coefficient k_w and either constant or non-constant diffusivity.

Results

Shallow waters

To simulate the uptake of CO_2 in shallow areas, $L = 100\,\mathrm{m}$, diffusivity $K(z)$ is used as given in equation (11) in [1]. Figure 7 illustrates the convergence of the simulation, which shows that the simulation still has quadratic convergence both in time and space. The convergence test are run for a time of 10 days.

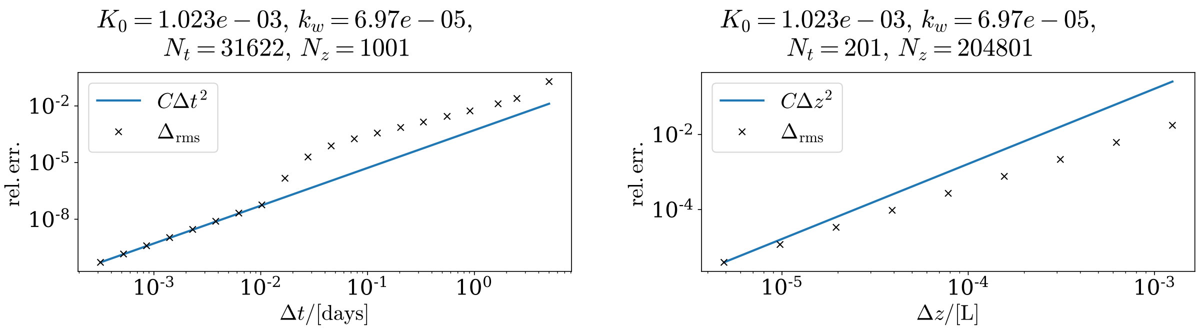


Figure 7: Error, as a function of step length in both time and space.

The simulation is then run for 180 days, with $N_t = N_z = 10000$. This corresponds to $\Delta t = 180/10000 = 1.8 \cdot 10^{-2}$days, and $\Delta z = L/10000 = 10^{-4}L$. The results are shown in Figure 8, together with the form of the diffusivity, $K(z)$. Figure 9 shows how the minimum and maximum concentration evolves as a function of time, together with some snap shots of the concentration as it evolves. The surface of the ocean, where wind dominates the diffusion, first absorbs the CO_2. The deeper parts of the ocean lacks behind some, as there low diffusivity around 40 meters are a bottle neck, but it does not take long before it penetrates down to the ocean floor.

At 90 days, the ocean is nearly in equilibrium with the atmosphere, and at 180 it is so close as does not make a difference. This gives confidence to an assumption that the carbon in the ocean is in sync with the atmosphere.

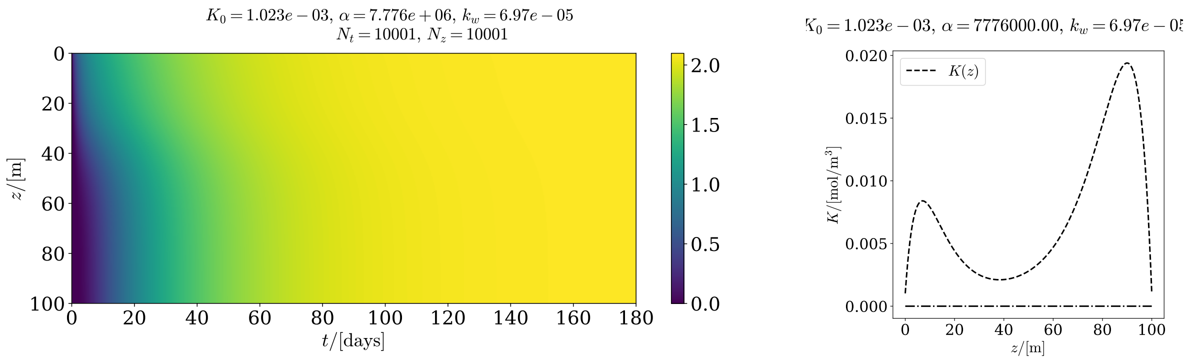


Figure 8: cap

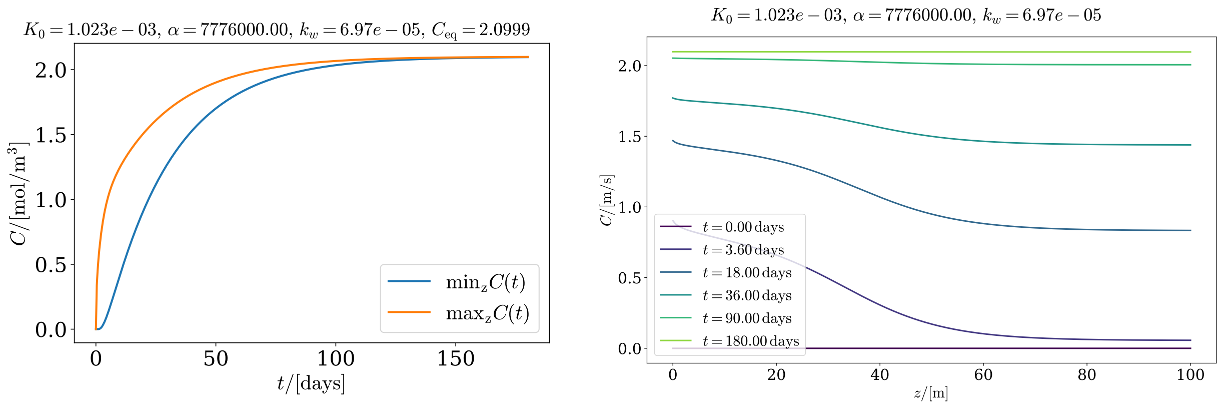


Figure 9: cap

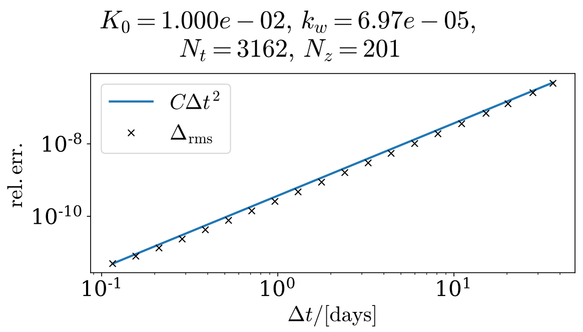
Deep waters

In this section, the ocean is modeled as 4000 m deep, and with a 2-layered diffusivity. This model is run for a longer time of several years, and therefore includes the increase in CO_2 in the atmosphere. Figure 10 shows the convergence of the simulation, confirming the simulation still converges quadratically.

Figure 11 shows the concentration of CO_2 as a function of time, as given by the simulation. The units of C in this exercise is $\mathrm{mol/m^{-3}}$, so the "mass" given by the formula $M = \int \mathrm{d}z C(z)$ is mol per square meter of ocean. Using $0.36 \cdot 10^{15}\mathrm{m}^2$ as the value of the surface area of the global oceans, and 24 g/mol as the molar weight of carbon in CO_2, the total wieght of the carbon stored in the ocean is plotted in Figure 12.

References

[1] NTNU, Institutt for Fysikk. Exercise 3, tfy4235 computational physics, 2021.



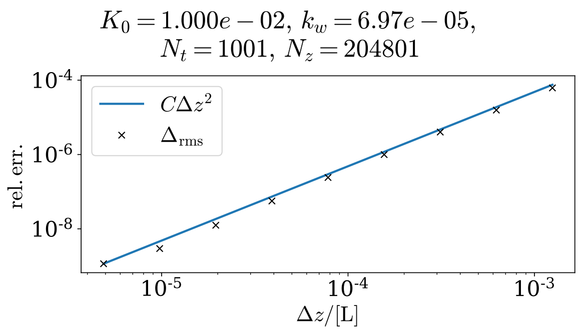


Figure 10: Error, as a function of step length in both time and space.

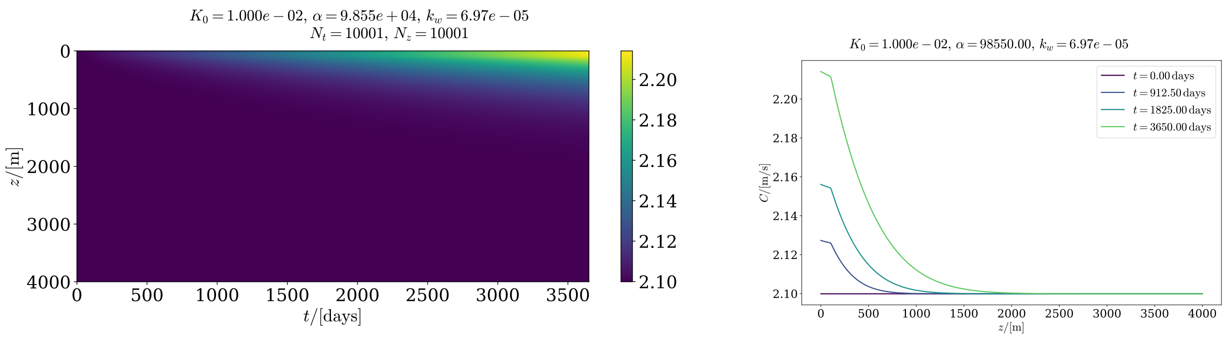


Figure 11: The concentration of CO_2 in as a function of time, over 10 years.

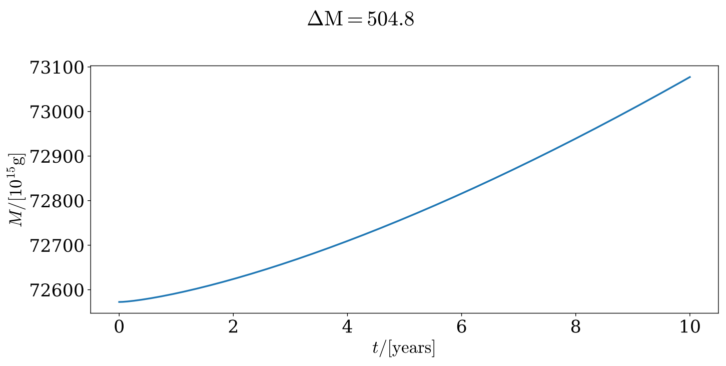


Figure 12: The concentration of CO_2 in as a function of time, over 10 years.