Exercise 3

Thus far, we have solved equations that are comparatively simple. We now attempt to solve a more complicated problem of interest namely the Convection-Diffusion-Reaction (CDR) equation. This exercise is broken up into 2 parts. The first part demonstrates the difference between convective and diffusive transport. The second part outlines how reactions can be modelled. In particular, we explore coupling the transport equations for multiple species.

Learning outcomes

After this exercise, the following learning outcomes are desired:

1. Be able to visualize how convection and diffusive transport occur
2. Appreciate the added complexity in modelling convection.
3. Implement a convection-diffusion-reaction system and observe the transient behavior of the system

Part 1

Problem statement

The one 1D convection-diffusion equation with a constant convective velocity U that we are solving in this exercise can be written as follows:

$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2}$$

where D is the diffusion coefficient and c is the concentration / mole fraction. We shall take it to be mole fraction in this case which makes it non-dimensional by definition. When considering the full convection-diffusion equation where velocity is not a constant, the 1D and full equation will take the following forms respectively:

$$\frac{\partial c}{\partial t} + u_x\frac{\partial c}{\partial x} = D\frac{\partial^2 c}{\partial x^2}$$

$$\frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla c = D\nabla^2 c$$

where u_x is the velocity in the x direction and $\mathbf{u}$ is the velocity vector. The difference when u_x or $\mathbf{u}$ is not a constant is that we need to supply information regarding the velocity. This could take the form of:

1. Assuming that u_x is constant with a magnitude U which gives us the equation we are interested in.
2. Supply information about the velocity e.g. providing a solution for the velocity profile e.g. laminar flow in a pipe
3. Supply additional equations which will be coupled to the convection-diffusion equation. This typically might be the Navier-Stokes equations or approximations of them.

Typically, the equations are scaled and solved in non-dimensional form which enables us to work in dimensionless groups instead. However, some solvers do not accommodate this so readily e.g. `OpenFOAM`.

We introduce the following scalings:

$$x = L\tilde{x}$$

$$t = \frac{L}{U}\tilde{t}$$

The model equation then becomes:

$$\frac{U}{L}\frac{\partial c}{\partial \tilde{t}} + \frac{U}{L}\frac{\partial c}{\partial \tilde{x}} = \frac{D}{L^2}\frac{\partial^2 c}{\partial \tilde{x}^2}$$

After a bit of cleanup:

$$\frac{\partial c}{\partial \tilde{t}} + \frac{\partial c}{\partial \tilde{x}} = \frac{1}{\mathrm{Pe}}\frac{\partial^2 c}{\partial \tilde{x}^2}$$

Where $\mathrm{Pe} = \frac{UL}{D}$ is the Peclet number and tracks the ratio of convective transport to diffusive transport. We will often see the Pe come up in transport related problems.

We now need to address the problems of the initial and boundary conditions. For this problem, we can introduce periodic boundary conditions or "Mario" boundary conditions. These boundary conditions have the effect of connecting one end of the simulation domain to the other end so as the particles are transported out of one end, they re-appear at the other end. Fortunately, `FiPy` makes it easy for us to set-up a domain with periodic boundary conditions. Mathematically, these periodic boundary conditions can be written as:

$$c(x = 0, t) = c(x = 4L, t)$$

In non-dimensional form:

$$c(\tilde{x} = 0, \tilde{t}\,) = c(\tilde{x} = 4, \tilde{t}\,)$$

For the initial boundary conditions, we specify a narrow gaussian distribution at the center of the domain:

$$c(\tilde{x}, \tilde{t}\, = 0) = \frac{0.05}{0.05\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{\tilde{x}-0.5}{0.05}\right)^2}$$

Numerical implementation

Theoretical discussion

Let us explore things from a more theoretical level. We introduce two specific dimensionless groups: the Courant Co and grid Peclet Pe_{Cell} numbers:

$$\mathrm{Co} = \frac{U\Delta t}{\Delta x}$$

$$\mathrm{Pe}_{Cell} = \frac{U\Delta x}{D}$$

Suppose we keep $D = 1.0$ as per the previous diffusion problems and we specify $\mathrm{Pe} = 100$, in terms of dimensionless variables ($L = 1.0$). From a stability perspective, it has been presented in the literature that $\mathrm{Pe}_{Cell} < 2.0$ is necessary. We shall keep $\mathrm{Pe}_{Cell} = 1.0$ which corresponds to $\Delta x = 0.01$.

To make sense of the Courant number, let us perform a similar discretisation as we did for the diffusion problem to derive the Fourier number. We consider the pure convective equation:

$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} = 0$$

Before progressing, let us revisit some basic concepts:

A derivative is written as follows:

$$\frac{d}{dx}c(x_0) = \lim_{\Delta x \to 0} \frac{c(x_0 + \Delta x) - c(x_0)}{\Delta x}$$

We can also write the Taylor series expansion for c:

$$c(x_0 + \Delta x) = c(x_0) + \Delta x \frac{\partial c}{\partial x}\bigg|_{x_0} + \frac{\Delta x^2}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{x_0} + \frac{\Delta x^3}{6}\frac{\partial^3 c}{\partial x^3}\bigg|_{x_0} + \dots$$

We can now evaluate the approximation order of the derivative:

$$\left(\frac{\partial c}{\partial x}\bigg|_{x_0}\right)_{DISCRETE} - \left(\frac{\partial c}{\partial x}\bigg|_{x_0}\right)_{EXACT} = \frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{x_0} + \frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3}\bigg|_{x_0} + \dots$$

$$\frac{c(x_0 + \Delta x) - c(x_0)}{\Delta x} - \frac{\partial c}{\partial x}\bigg|_{x_0} = \frac{\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{x_0} + \frac{\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3}\bigg|_{x_0} + \dots$$

Note that the approximate form of the derivative looks like the simple discretisation schemes we have used previously and will use subsequently. As shown in the RHS of the equation, the truncation error is of order Δx which is first order. Hence the Euler time-stepping and first order upwind schemes are called first order. The error of the approximations scales with $\mathcal{O}(\Delta x)$. It is possible to conceive of different schemes that are higher order, but that is significantly beyond the scope of this course.

Going back to our original problem, we apply an explicit forward Euler discretization scheme and a simple first order upwind scheme for the convective term:

$$\frac{c_i^{new} - c_i^{old}}{\Delta t} + U\frac{c_i^{old} - c_{i-1}^{old}}{\Delta x} = 0$$

This can be written as follows:

$$\frac{c(x_0, t_0 + \Delta t) - c(x_0, t_0)}{\Delta t} + U\frac{c(x_0, t_0) - c(x_0 - \Delta x, t_0)}{\Delta x} = 0$$

To evaluate the error, we employ the same approach as with the derivatives.

$$\left(\frac{c(x_0, t_0 + \Delta t) - c(x_0, t_0)}{\Delta t} + u_x\frac{c(x_0, t_0) - c(x_0 - \Delta x, t_0)}{\Delta x}\right) - \left(\frac{\partial c}{\partial t}\bigg|_{(x_0,t_0)} + U\frac{\partial c}{\partial x}\bigg|_{(x_0,t_0)}\right)$$
$$= \frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2}\bigg|_{(x_0,t_0)} - \frac{U\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)} + \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3}\bigg|_{(x_0,t_0)} - \frac{U\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3}\bigg|_{(x_0,t_0)} + \dots$$
$$= \mathcal{O}(\Delta x) + \mathcal{O}(\Delta t)$$

Another way of seeing the same problem is by considering the close relationship between the wave equation and the convection equation if Clairaut's theorem holds:

First considering the convection equation:

$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} = 0$$

We take the partial derivatives with respect to t and x separately:

$$\frac{\partial^2 c}{\partial t^2} + U\frac{\partial^2 c}{\partial t \partial x} = 0$$
$$\frac{\partial^2 c}{\partial x \partial t} + U\frac{\partial^2 c}{\partial x^2} = 0$$

Since Clairaut's theorem holds, we can combine these equations to give us the wave equation:

$$\frac{\partial^2 c}{\partial t^2} - U^2\frac{\partial^2 c}{\partial x^2} = 0$$

This gives us a convenient way to rejig the 2nd order derivatives from the error evaluation process:

$$\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2}\bigg|_{(x_0,t_0)} - \frac{U\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)} = U^2\frac{\Delta t}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)} - \frac{U\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)}$$

$$\frac{\Delta t}{2}\frac{\partial^2 c}{\partial t^2}\bigg|_{(x_0,t_0)} - \frac{U\Delta x}{2}\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)} = \frac{U}{2}\Delta x\left(\frac{U\Delta t}{\Delta x} - 1\right)\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)}$$

Let us examine the result we have just obtained. We note that there is a second order spatial derivative ($\nabla^2 c$) and there are a bunch of constants in front of it acting as a coefficient. This should immediately remind you of diffusion.

So lets stick this result back into the error analysis we previously evaluated:

$$\left(\frac{c(x_0, t_0+\Delta t) - c(x_0,t_0)}{\Delta t} + U\frac{c(x_0,t_0) - c(x_0-\Delta x, t_0)}{\Delta x}\right) - \left(\frac{\partial c}{\partial t}\bigg|_{(x_0,t_0)} + U\frac{\partial c}{\partial x}\bigg|_{(x_0,t_0)}\right)$$
$$+ \frac{U}{2}\Delta x\left(1 - \frac{U\Delta t}{\Delta x}\right)\frac{\partial^2 c}{\partial x^2}\bigg|_{(x_0,t_0)}$$
$$= \frac{\Delta t^2}{6}\frac{\partial^3 c}{\partial t^3}\bigg|_{(x_0,t_0)} - \frac{U\Delta x^2}{6}\frac{\partial^3 c}{\partial x^3}\bigg|_{(x_0,t_0)} + \dots$$

This way of seeing the problem gives us a scheme that is 2nd order accurate, but with added diffusion. This diffusion is not natural and is called numerical diffusion. We can now obtain an expression for our Courant number.

The "diffusion" coefficient or viscosity can be written by as follows:

$$\mu = \frac{U}{2}\Delta x\left(1 - \frac{U\Delta t}{\Delta x}\right)$$

And the courant number is given by::

$$\mathrm{Co} = \frac{U\Delta t}{\Delta x}$$

When $\mathrm{Co} > 1$, the problem becomes ill-posed since the viscosity is negative and the solution will continue to grow. It must be noted that this analysis is specifically relevant to the numerical schemes considered. However, the intuition gained here is useful going forward. We use more sophisticated schemes for the convection term and use backwards time-stepping which gives us more flexibility.

Implementation

We saw that stability and numerical dispersion are two things we should take note of when solving the problem. Generally speaking, we would like a finer mesh resolution as this gives us a more accurate solution as the physics is better resolved and numerical dispersion is mitigated. However, there is a trade off with the compute required with more time and memory being required.

The mesh with periodic boundary conditions is created as follows:

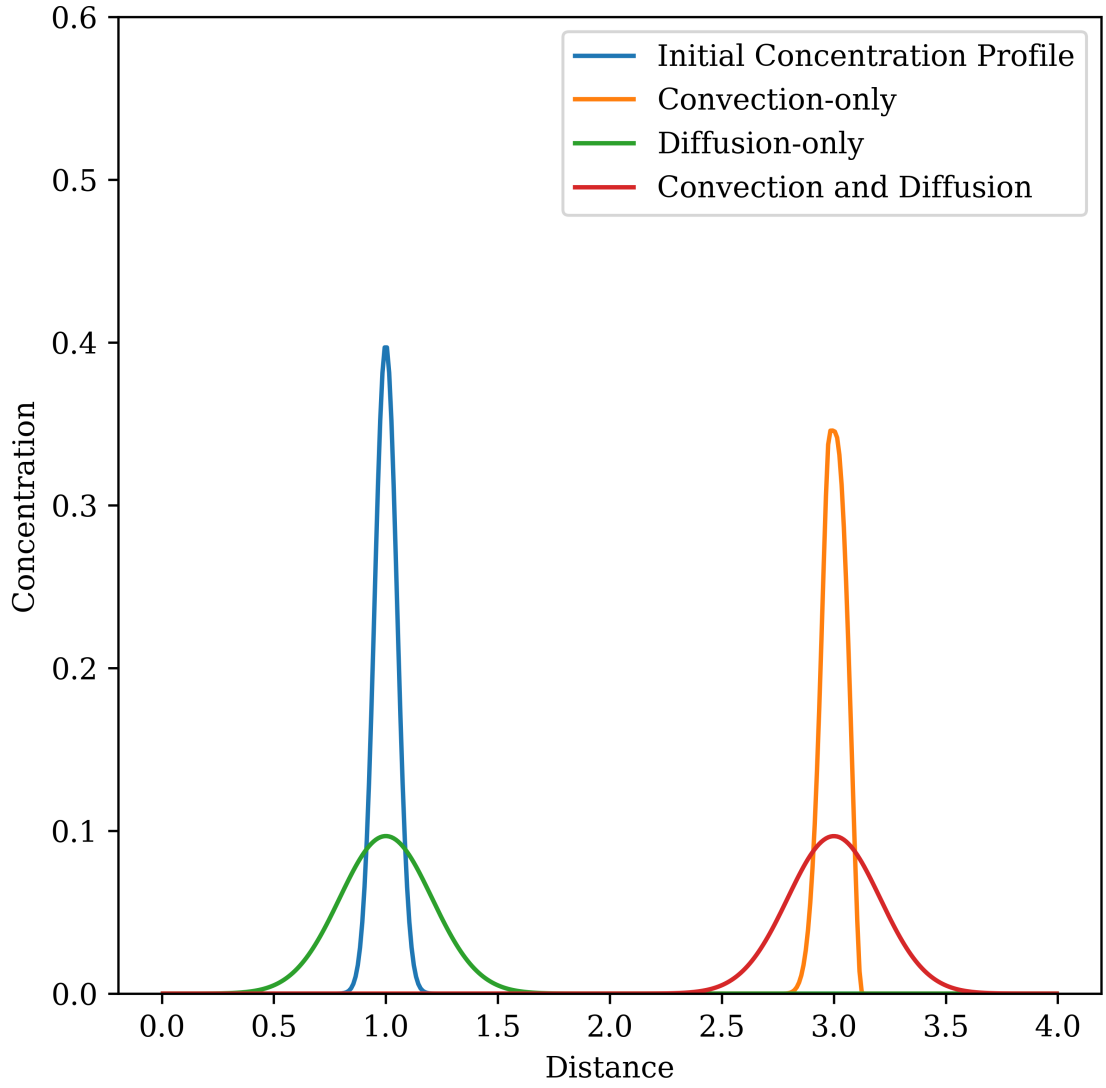
```
# Setting up a mesh of domain size 4L
nx = 400
dx = 0.01
L = 1.0

# The use of FiPy's PeriodicGrid1D / 2D gives us a mesh with the necessary BC implemented
mesh = PeriodicGrid1D(nx=nx, dx=dx)
```

Feel free to explore the impact of the mesh resolution i.e. changing `dx`. However, for the base conditions used in this exercise, I found that making the mesh coarser i.e. making `dx` larger results in unacceptable levels of numerical dispersion for the pure convection equation. Three different equation systems are implemented for you to explore with. Just comment out one equation and uncomment the other to explore. There are also various `ConvectionTerm` implemented in `FiPy`, but I found the `Van Leer` term to be most effective for this case.

1. Pure convection
2. Pure diffusion
3. Convection + diffusion (scaled by Pe)

```
# defining the equation
# We write three equations: pure convection, pure diffusion and convection-diffusion
eq = TransientTerm() + VanLeerConvectionTerm(coeff=convCoeff) == 0
# eq = TransientTerm() - DiffusionTerm(coeff=(1/peclet)) == 0
# eq = TransientTerm() + VanLeerConvectionTerm(coeff=convCoeff) - DiffusionTerm(coeff=(1.0/peclet)) == 0
```

Results



We can see quite a few things of interest here:

1. When there is only diffusive transport, the location of the highest concentration does not change, but it does spread out and the peak becomes broader which is to be expected.
2. When there is pure convective transport only, we see the peak simply being transported as is to its final position. However, due to the difficulties in modelling convection (false numerical diffusion) which we have discussed previously, the peak at the final time is slightly broader.
3. When there is both convective and diffusive transport, the two transport effects discussed in points 1 and 2 are combined.

Part 2

Problem statement

We now consider the following second order reaction:

$$A + B \rightarrow C$$

The equation system for all the species can be written as follows:

$$\frac{\partial a}{\partial t} + U\frac{\partial a}{\partial x} - D\frac{\partial^2 a}{\partial x^2} + r_A = 0$$
$$\frac{\partial b}{\partial t} + U\frac{\partial b}{\partial x} - D\frac{\partial^2 b}{\partial x^2} + r_B = 0$$
$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} - D\frac{\partial^2 c}{\partial x^2} + r_C = 0$$

where a, b and c are the concentrations of species A, B and C respectively, U is the convective velocity and is a constant, D is the diffusivity of the species as is assumed to be constant and identical for all three species. r_i is the rate of reaction for species A, B and C.

Based on the stoichiometry, we are able to write down a relationship between the rates of reaction:

$$r_A = r_B = -r_C$$

And the following expression for r_A can be written down:

$$r_A = \kappa a b$$

where κ is the rate constant. At this point, we see that the first two equations are strongly coupled since we need information about both a and b at the same time to solve for the rate of reaction. The last equation tracking C can be solved after solving for A and B since it is much more weakly coupled. We now impose the following boundary conditions:

At the inlet $x = 0$:

$$a(x = 0, t) = 1.0$$
$$b(x = 0, t) = 0.5$$
$$c(x = 0, t) = 0.0$$

At the outlet $x = L$, we need to provide information a boundary condition that reflects the condition at the outlet. This is not a trivial problem as specifying information at the outlet boundary in the form of Neumann or Dirichlet BCs can be somewhat unphysical. Nevertheless, we adopt the approach of specifying the zero diffusive flux boundary condition:

$$\frac{\partial a}{\partial x}(x = L, t) = 0$$
$$\frac{\partial b}{\partial x}(x = L, t) = 0$$
$$\frac{\partial c}{\partial x}(x = L, t) = 0$$

We set $a = b = c = 0$ as the initial conditions.

We use the following parameters:

$$D = 1.0$$
$$\kappa = 30.0$$
$$U = 3.0$$

It is possible to add more layers of complexity to this equation system, but that is beyond the scope of this exercise.

Numerical Implementation

This is the first time we encounter non-linearity in our equation system. This comes from the reaction term r_i. The reaction term also causes the equations to be coupled, however `FiPy` handles that with ease. The equations are defined as follows:

```
eq_a = (TransientTerm(var=a)) + VanLeerConvectionTerm(coeff=convCoeff, var=a) -
DiffusionTerm(coeff=D, var =a) + ImplicitSourceTerm(coeff = k*b, var=a) == 0

eq_b = (TransientTerm(var=b)) + VanLeerConvectionTerm(coeff=convCoeff, var=b) -
DiffusionTerm(coeff=D, var =b) + ImplicitSourceTerm(coeff = k*b, var=a) == 0

eq_c = (TransientTerm(var=c)) + VanLeerConvectionTerm(coeff=convCoeff, var=c) -
DiffusionTerm(coeff=D, var =c) + ImplicitSourceTerm(coeff = -k*b, var=a) == 0
```

Note the change in sign for the coefficient of the source term. As we have the non-linear term, we need to perform sweeps at each timestep. The at each timestep, the solver solves a linear system of equations which is perfectly fine for nice and linear PDEs. However, if we update the problem at each timestep with the results from the previous solution (sweep), we can solve the non-linear PDE. The process of enabling sweeping in `FiPy` requires the following steps:

1. Updating the `CellVariable` declaration:

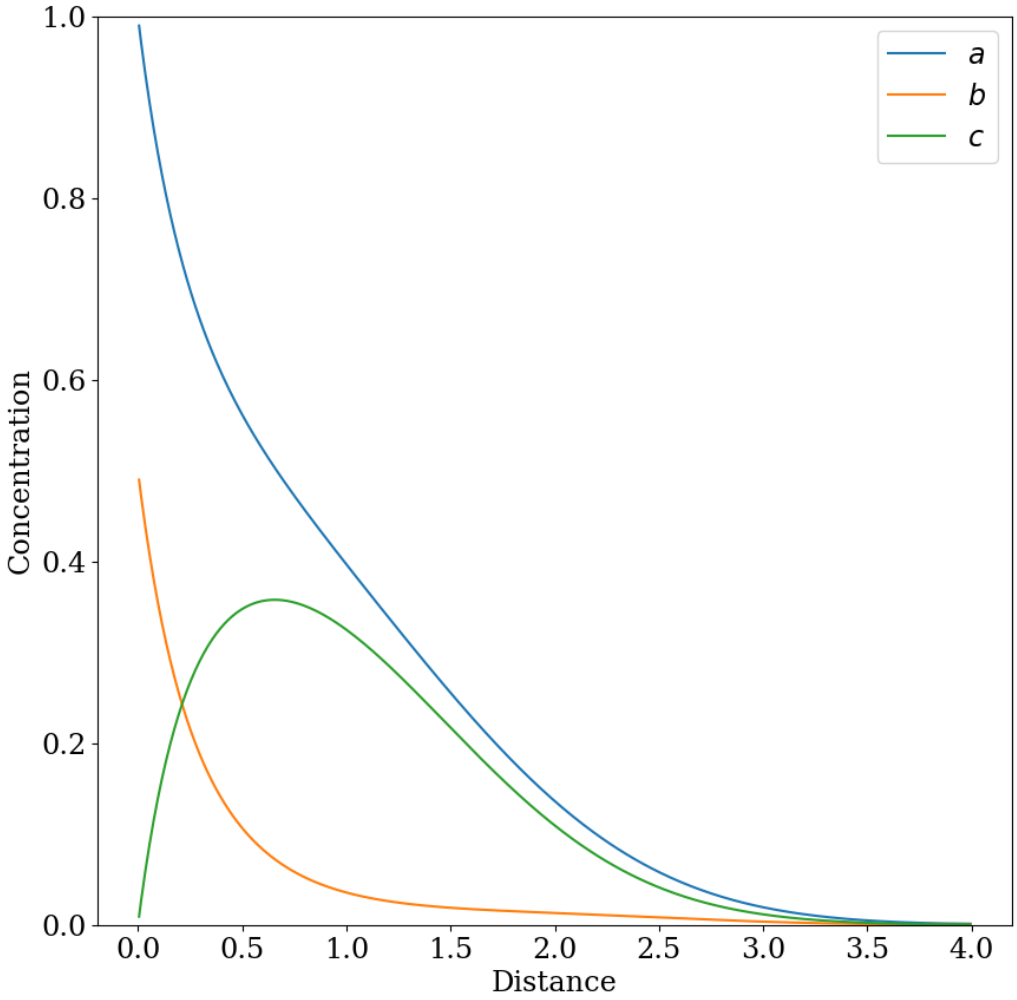
```
a = CellVariable(mesh=mesh, name=r"$a$", value = 0.0, hasOld=True)
b = CellVariable(mesh=mesh, name=r"$b$", value = 0.0, hasOld=True)
c = CellVariable(mesh=mesh, name=r"$c$", value = 0.0, hasOld=True)
```

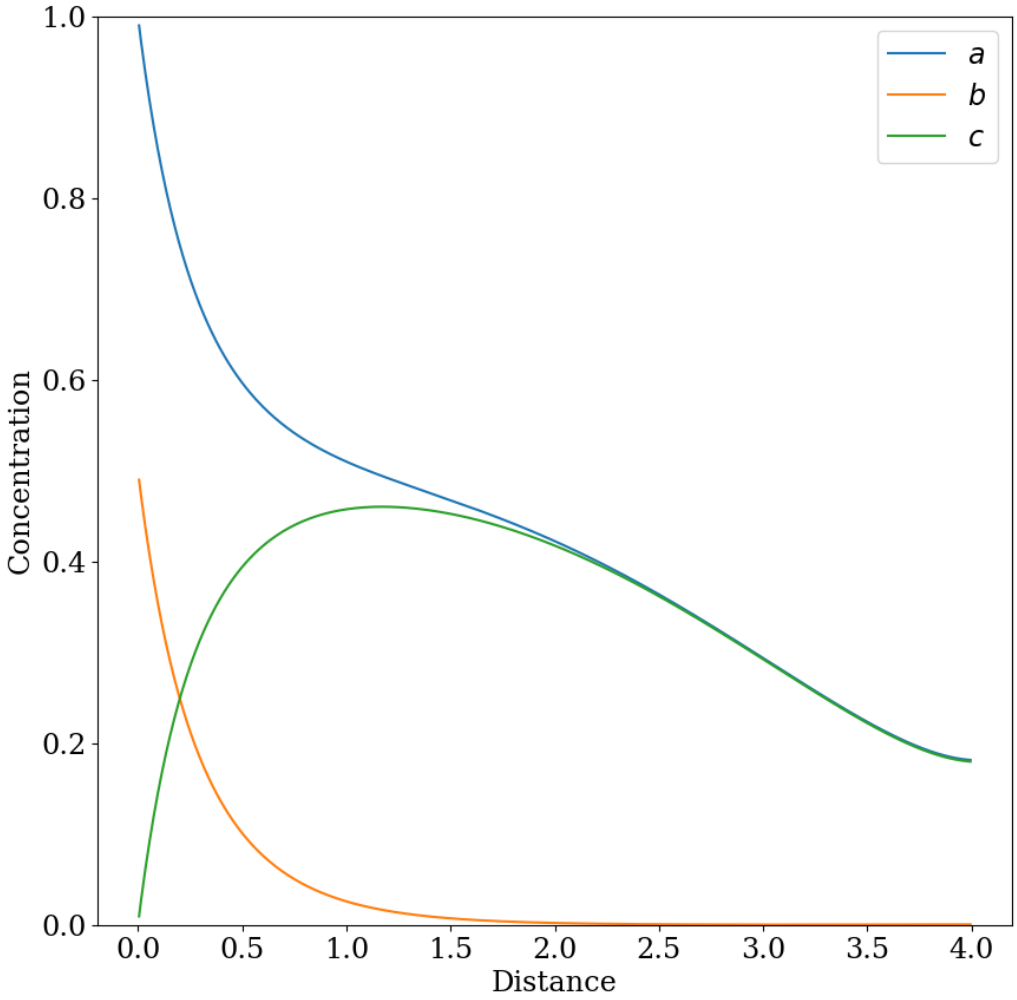
2. Adding the following bits to the time-stepping portion of the code:

```
while t < run_time:
    t += dt
    timestep += 1
    a.updateOld()
    b.updateOld()
    c.updateOld()
    res1 = 1e+10
    res2 = 1e+10
    while res1 > 1e-4 and res2 > 1e-4:
        res1 = eqn_main.sweep(dt=dt)
        res2 =  eq_c.sweep(dt=dt)
        print("sweep")
    if (timestep % time_stride ==0):
        print ("Beep")
        if __name__ == '__main__':
            viewer.plot()
if __name__ == '__main__':
    input("Press <return> to proceed...")
```

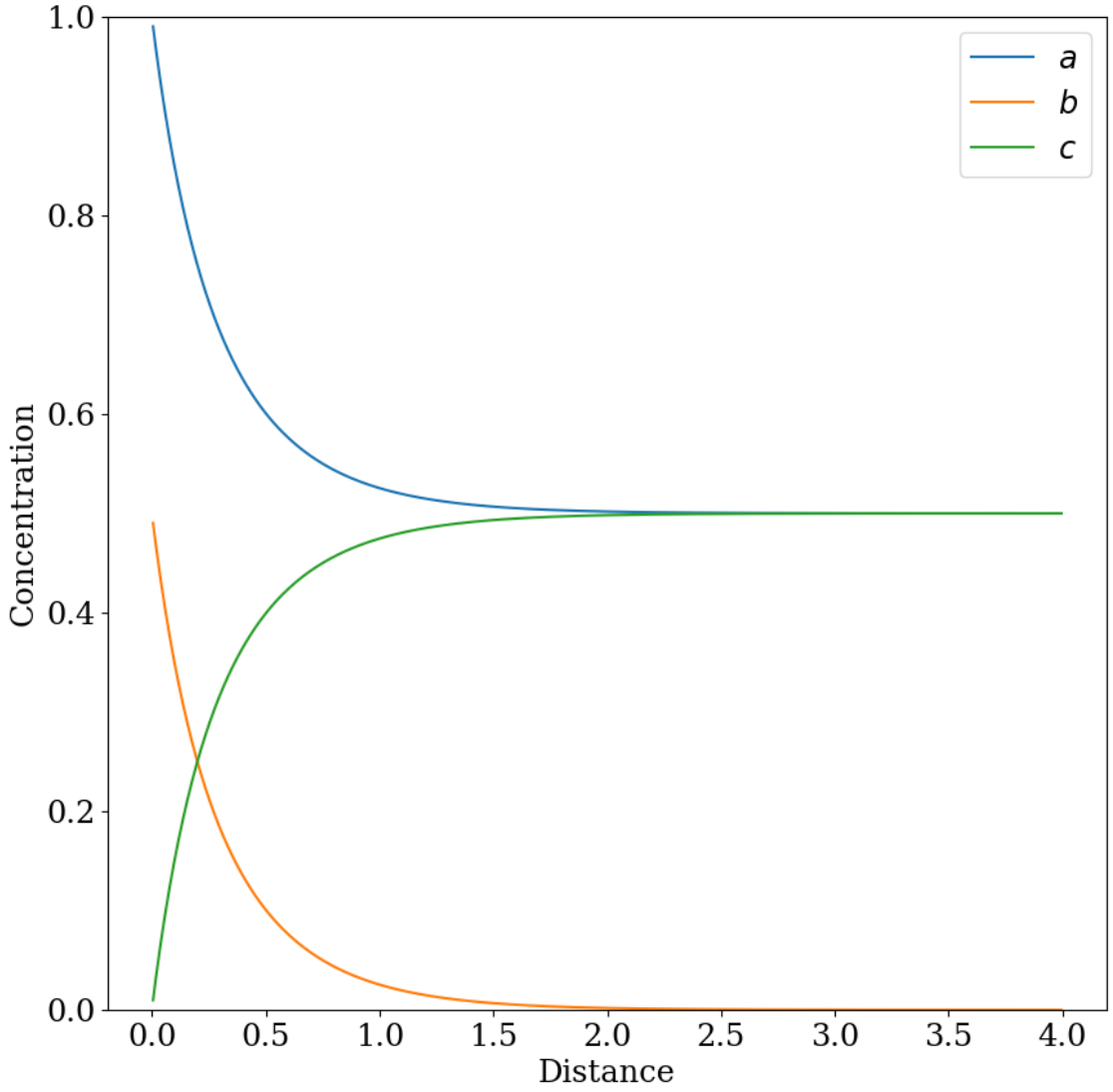
Results

This code takes a bit longer to run on a single processor. But parallelization is also beyond the scope of this course. Feel free to run the simulation with a larger time-step. I have attached some snapshots for your reference:


1.0
0.8
0.6
0.4
0.2
0.0
Concentration
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Distance
a
b
c



And our steady-state profile:


1.0
0.8
0.6
0.4
0.2
0.0
Concentration
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Distance
a
b
c