NUEN 618

Homework Assignment #2

September 27, 2019

Robert Turner

NUEN 618

Homework #2

Operator Splitting

The idea behind this assignment is to (1) mimic how OS is currently implemented in reactor transient analysis, and to (2) understand some of the flaws associated with the simplest OS schemes.

We will use a 4-equation PRKE model (2 equations for neutronics, one for fuel, and one for coolant) and perform a rod ejection simulation. At time $t = 0$, the reactor is critical (without any external source present). The rod is ejected between $t = T$ and $t = 2T$ during which time the external reactivity grows linearly in time from a value of 0 to a value of 1.45β. Use $T = 0.05$ sec and let the end of the transient be $t = 15T$.

Main Goal: Solve this time-dependent nonlinear problem using OS as described in the class notes. Crank-Nicolson (CN) must be used as the time integrator:

$$F_p(x_p^{n+1}) = x_p^{n+1} - x_p^n - \frac{\Delta t}{2}\left(f_p(t^{n+1}, x_p^{n+1}, o_p^{n+?}) + f_p(t^n, x_p^n, o_p^n)\right) = 0,$$

where p is the index of one of the 3 physics, x_p is the current physics, o_p are the other physics (the full solution vector is $x = [x_p, o_p]$), and f_p is the RHS (steady state residual) of physics p ($\frac{dx_p}{dt} = f_p$). At each time step, 3 single-physic components should be solved ($F_p\left(x_p^{n+1}\right) = 0$, with $p =$ neutronics, fuel, coolant).

Tasks:

1. Write a **general** Crank-Nicolson function F_p that advances a single physic component over $[t_n, t_{t+1}]$.
   - This CN function F_p should receive a function handle for f_p pertaining to the physic component being solved and should receive appropriate OS data o_p from the other physics. The zero of the F_p function should be the values at the end of the time step for that given physics, i.e. x_p^{n+1}.
   - Note that finding the zero of $F_{\text{neutronics}}$ is equivalent to solving a 2×2 linear system while finding the zero of F_{fuel} or F_{coolant} is equivalent to finding the root of a scalar nonlinear function.
   - However, you may streamline solving $F_p(x_p^{n+1}) = 0$ by always calling Python's nonlinear `fsolve`.
2. Your OS implementation should
   - allow for the ordering of physic solves over $[t_n, t_{n+1}]$ to be user-defined (i.e. arbitrary ordering),
   - allow for either simultaneous or staggered OS schemes (whether o_p always contains the latest values or not),
   - allow for the OS scheme to either resolve lagged nonlinearities by iterating over the time step (iterated OS) or to do a single pass through all of the physic components.

Results to present:

- The total power as a function of time for some representative choices of Δt using iteration OS / single pass OS (with simultaneous or staggered updates),
- The number of iterations per time step versus time,
- The convergence rates for the peak power that should occur around $t = 0.11$ s for all four cases of simultaneous/staggered OS and iterated/non-iterated OS.

Data:

Parameter	Value
λ	$0.1\ \text{s}^{-1}$
β	600×10^{-5}
Λ	$10^{-5}\ \text{s}^{-1}$
α_{fuel}	$-3.5 \times 10^{-5}\ \text{K}^{-1}$
α_{mod}	$-55 \times 10^{-5}\ \text{K}^{-1}$
R_{fuel}	0.0041 m
R_{gap}	0.00411 m
R_{clad}	0.00475 m
fuel height	4 m
initial power in a single pin	59,506 W
pin pitch	0.0126 m
gap conductance	10^4 W/(m^2.K)
clad conductivity	17 W/(m.K)
fuel conductivity	given in attached file
ρ_{fuel}	given in attached file
$C_{p_{fuel}}$	given in attached file
fluid axial velocity	5 m/s
convective heat transfer coefficient	$h_{conv} = 0.023 Re^{0.8} Pr^{0.4} \frac{k_{mod}}{D_{hy}}$
fluid conductivity	$k_{mod} = 0.54$ W/(m.K)
fluid viscosity	90×10^{-6} Pa.s
ρ_{mod}	given in attached file
$C_{p_{mod}}$	given in attached file
inlet fluid temperature	290 K

And

$$\Omega_{pow} = \frac{\text{initial power in a single pin}}{(\text{fuel height})\pi R_{fuel}^2}$$

$$D_{hy} = \frac{4A_{flow}}{P_{wet}}$$

$$A_{fuel} = \pi R_{fuel}^2$$

$$A_{flow} = (\text{pin pitch})^2 - \pi R_{clad}^2$$

$$P_{wet} = 2\pi R_{clad}$$

Finally, use the standard definitions for the Reynolds and Prandtl numbers and all of of the right-hand sides for the 4 equations are fully determined.

Code Development:

The key to the performance of this code is both the generality of the CN function and the data passage scheme. As the problem statement requires, the CN function takes as input a Python callable, which evaluates the residual of the physic function that is taken as an argument. Along with the function is passed a data object filled with the neutronics (2×1 list), fuel temperature (scalar), coolant temperature (scalar), and time index data (integer) at the appropriate time step.

If the "stagger" flag is True, the data object that is taken as input into the fsolve function (which takes as input the CN function) is updated with the most recent physic solve within the time step. If the "stagger" flag is False, a temporary data object stores each physic solution and only updates the true data object after all physics have been solved within the time step. If the "iterate" flag is True, a "data_old" object holds the solution until the solutions have been determined in the following iterate, then is reset.

Results:

The total power is plotted along with a figure of the number of iterations within each time step for the iterated solves for time steps of 1E-3 s, 5E-4 s, 1E-4 s, and 5E-5 s in Figures 1, 2, 3, and 4, respectively. For all staggered solves, fuel temperature is solved first, followed by coolant temperature and reactor kinetics. While the simultaneous solves result in less accurate solutions, iterating the simultaneous solution within each time step results in a seemingly identical solution to that of an iterated staggered solve. This can be seen most clearly in Figure 1 where the time domain is coarsest. However, iterating on simultaneous physic advances requires about 50% more iterations per time step than when iterating on staggered advances. Finally, the number of iterations needed per time step to reach the tolerance is proportional to the absolute value of the gradient of the power, as the magnitude of the gradient of the power during the transient is highest amongst all of the four physics.

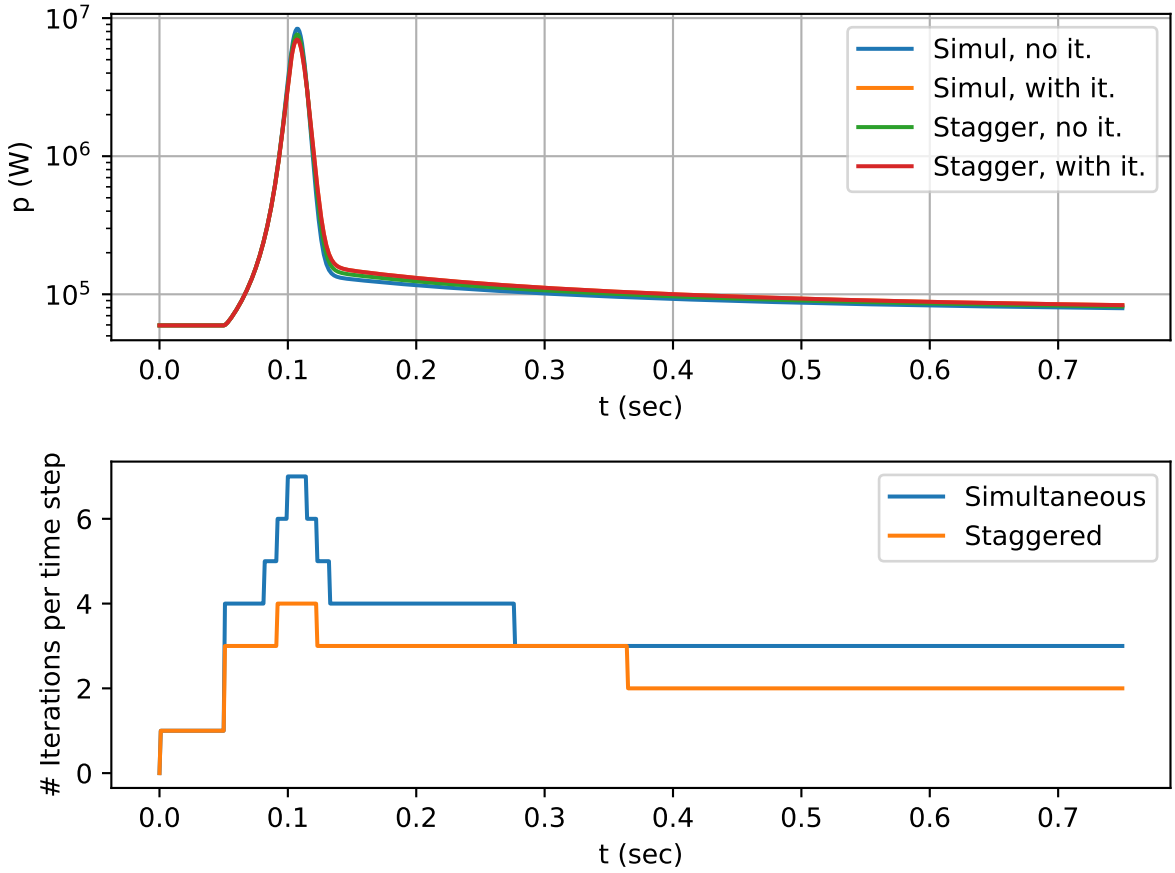

Figure 1: (above) Total power with the four combinations of simultaneous / staggered updates and iterated / non-iterated time steps for a time step size of 1E-3 s. (below) Number of time steps needed per time step for the iterated relative differences of each physic solution with the previous iterate solutions to reach a tolerance of 1E-7.

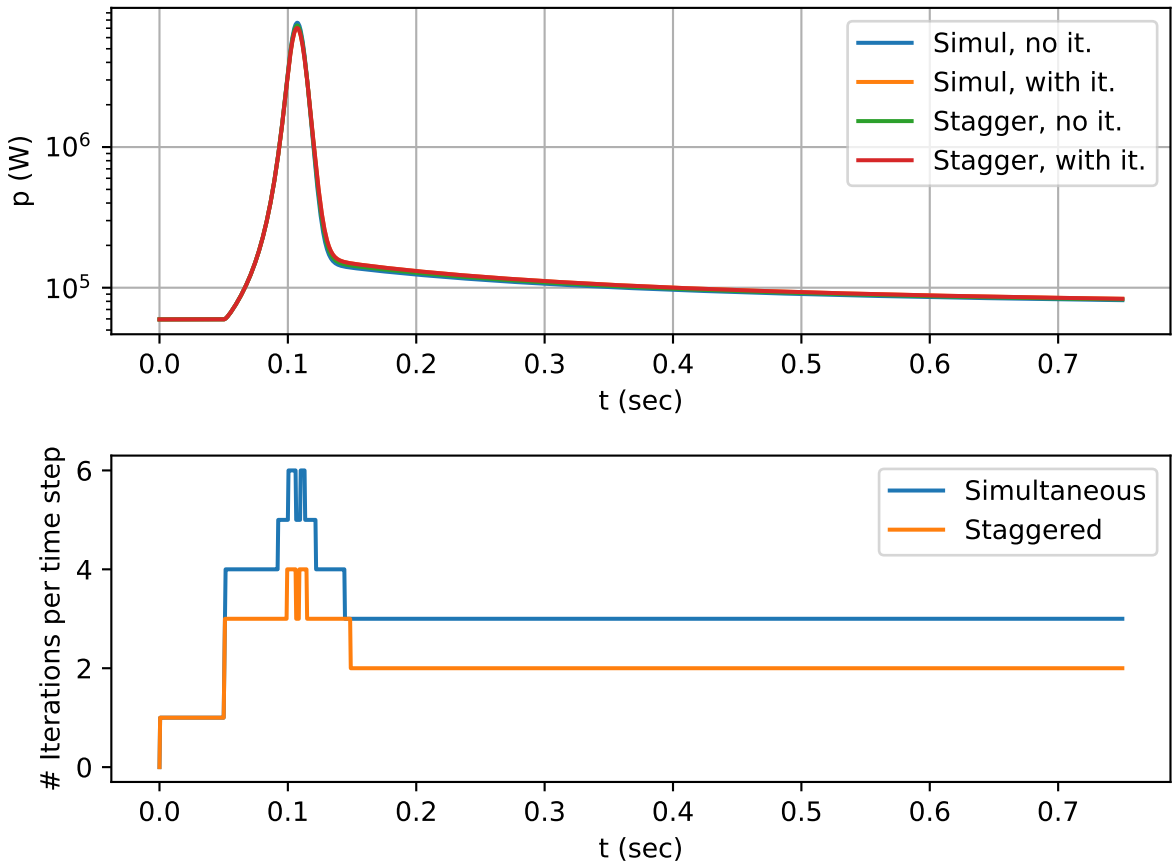

Figure 2: (above) Total power with the four combinations of simultaneous / staggered updates and iterated / non-iterated time steps for a time step size of 5E-4 s. (below) Number of time steps needed per time step for the iterated relative differences of each physic solution with the previous iterate solutions to reach a tolerance of 1E-7.

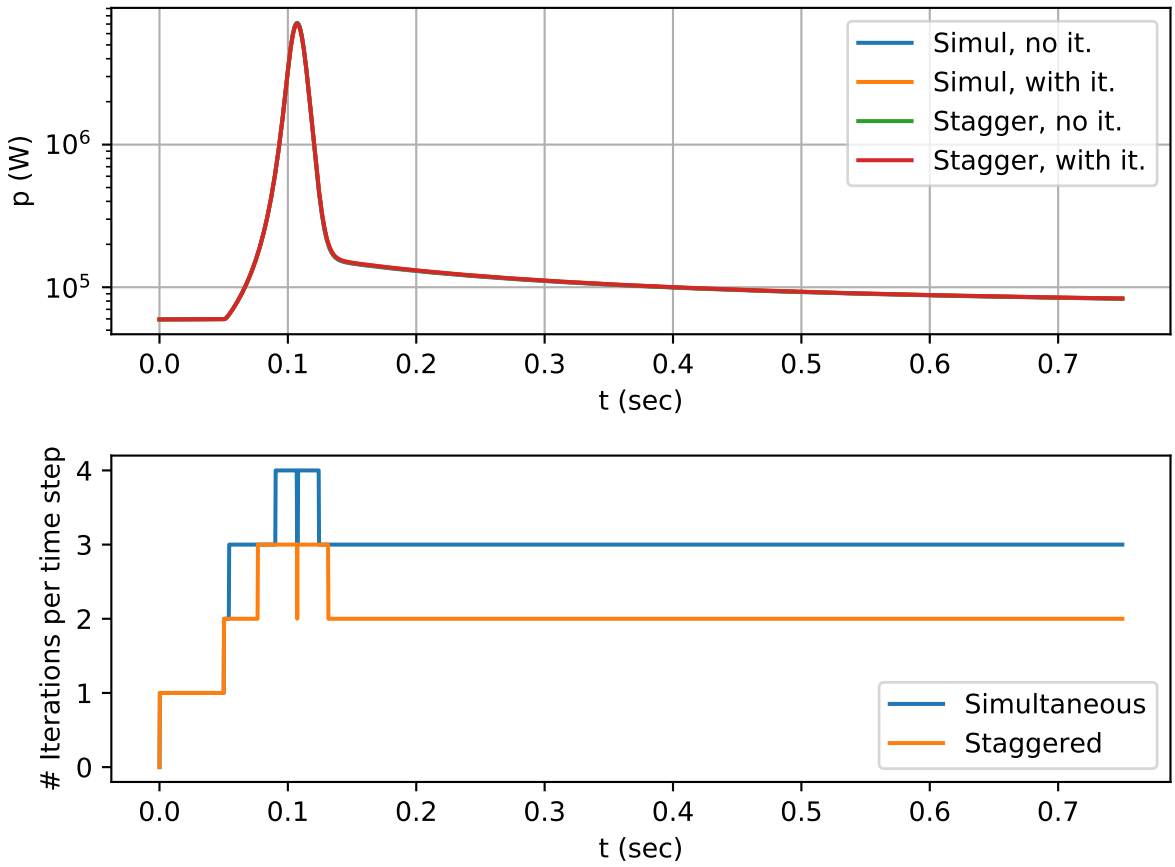

Figure 3: (above) Total power with the four combinations of simultaneous / staggered updates and iterated / non-iterated time steps for a time step size of 1E-4 s. (below) Number of time steps needed per time step for the iterated relative differences of each physic solution with the previous iterate solutions to reach a tolerance of 1E-7.

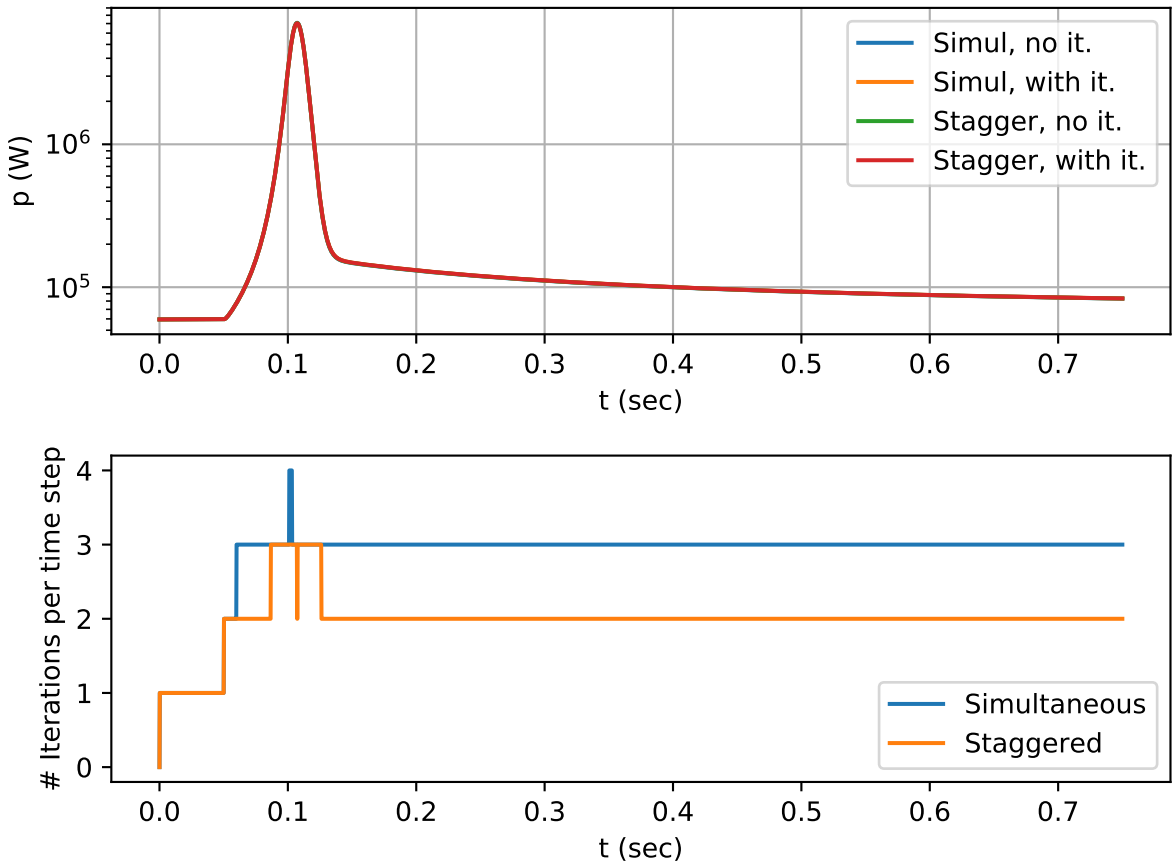

Figure 4: (above) Total power with the four combinations of simultaneous / staggered updates and iterated / non-iterated time steps for a time step size of 5E-5 s. (below) Number of time steps needed per time step for the iterated relative differences of each physic solution with the previous iterate solutions to reach a tolerance of 1E-7.

Figures 5 and 6 show convergence rates when performing simultaneous solves within time steps and when performing staggered solves, respectively. For these figures, a solve with a time step of 1E-6 s is used as the reference solution. Because the error between the coarse solutions and the reference solution is only evaluated at the time step where the maximum of the power occurs, a second order convergence rate is expected, as the Crank Nicolson scheme used is locally second order accurate. However, as the figures show, this second order convergence is only attained when iterating within time steps.

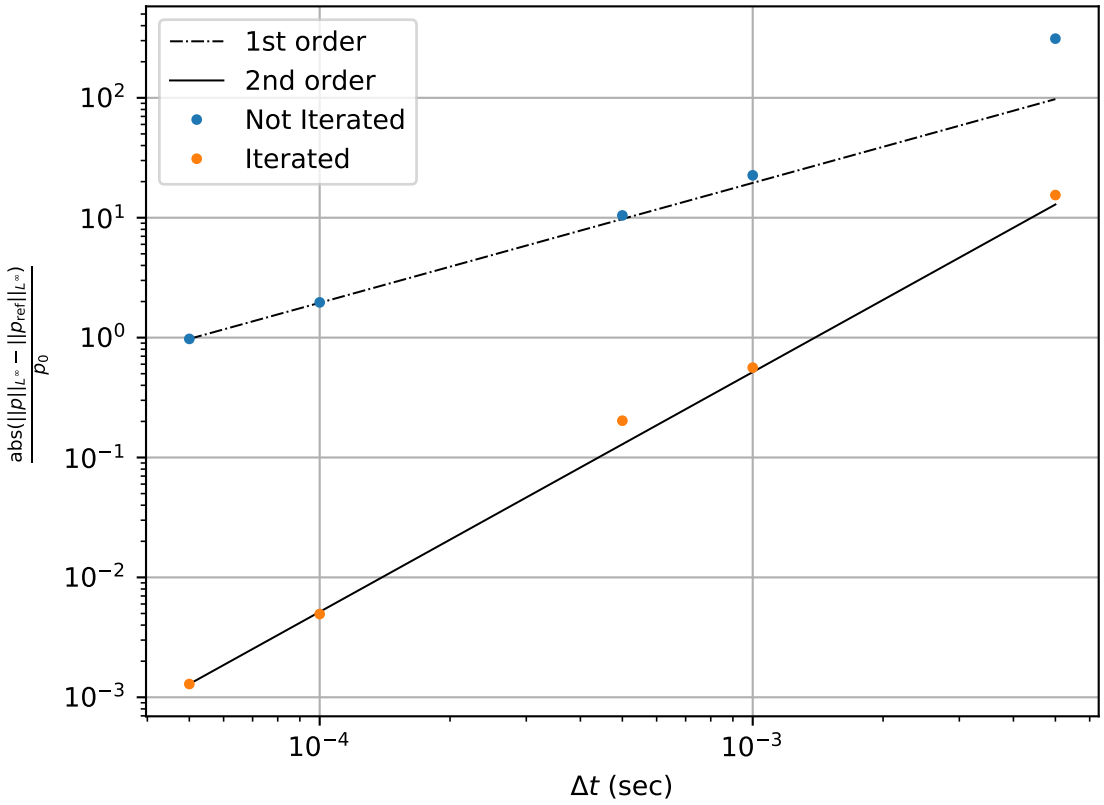

Figure 5: Convergence rates when advancing all physics simultaneously and with a tolerance of 1E-7 for the cases when iteration is performed within each time step.

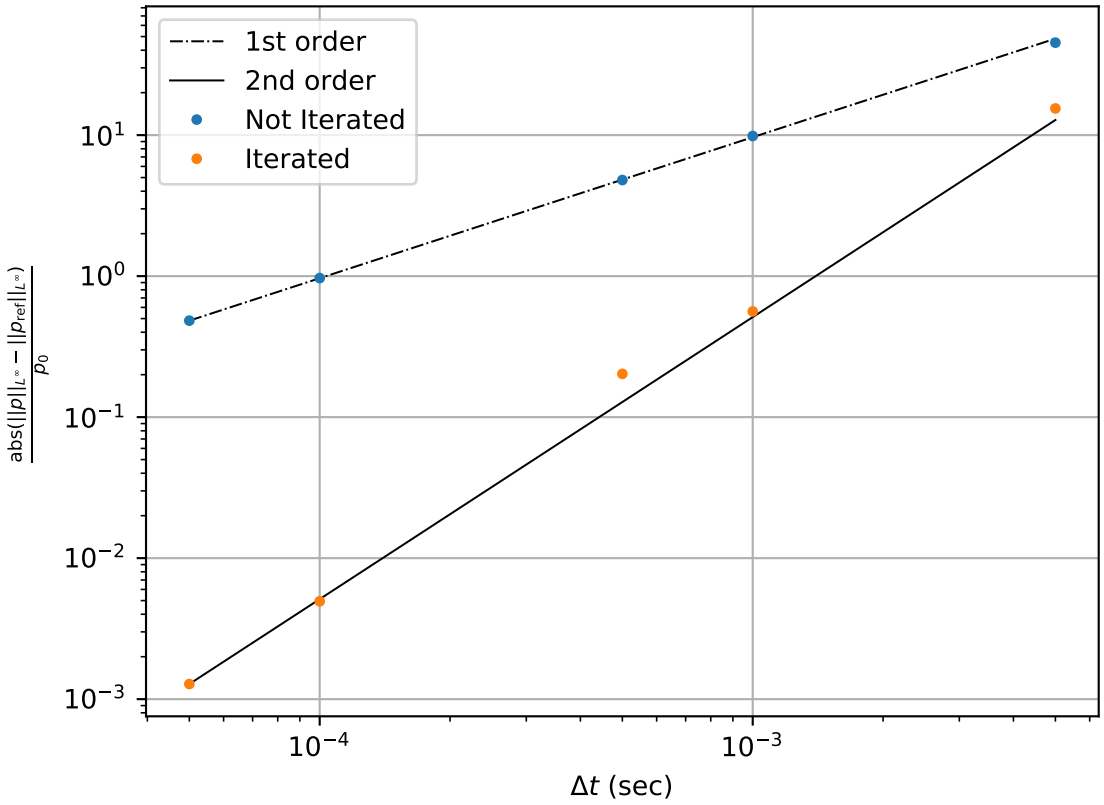

Figure 6: Convergence rates when staggering physics and with a tolerance of 1E-7 for the cases when iteration is performed within each time step.