TOF_schema_temps

October 15, 2022

[1]:
```python
import sys
import os

lib_path = os.path.realpath(os.path.join(os.getcwd(), ".."))
if lib_path not in sys.path:
    sys.path = [lib_path] + sys.path
```

[2]:
```python
from src.main_discontinu import *
from src.plot_fields import *

%matplotlib inline
rc("figure", figsize=(10, 5))
rc("figure", dpi=100)
```

1 Test de l'opérateur Problem en upwind avec différents schémas en temps

[3]:
```python
n_lim = 10**8
t_fin_lim = 0.002
```

[4]:
```python
phy_prop = PhysicalProperties(
    Delta=0.02,
    v=0.2,
    dS=0.005**2,
    lda1=5.5 * 10**-2,
    lda2=15.5,
    rho_cp1=70278.0,
    rho_cp2=702780.0,
    diff=1.0,
    alpha=0.06,
    a_i=357.0,
)
phy_prop_ref = PhysicalProperties(
    Delta=0.02,
    v=0.0,
```

```python
    dS=0.005**2,
    lda1=5.5 * 10**-2,
    lda2=15.5,
    rho_cp1=70278.0,
    rho_cp2=702780.0,
    diff=1.0,
    alpha=0.06,
    a_i=357.0,
)
num_prop_euler = NumericalProperties(
    dx=3.9 * 10**-5,
    schema="upwind",
    time_scheme="euler",
    phy_prop=phy_prop,
    cfl=0.5,
    fo=1.0,
)
num_prop_rk3 = NumericalProperties(
    dx=3.9 * 10**-5,
    schema="upwind",
    time_scheme="rk3",
    phy_prop=phy_prop,
    cfl=0.5,
    fo=1.0,
)
num_prop_rk4 = NumericalProperties(
    dx=3.9 * 10**-5,
    schema="upwind",
    time_scheme="rk4",
    phy_prop=phy_prop,
    cfl=0.5,
    fo=1.0,
)
markers = BulleTemperature(phy_prop=phy_prop, x=num_prop_euler.x, n_bulle=1)
markers.shift(0.00001)
```

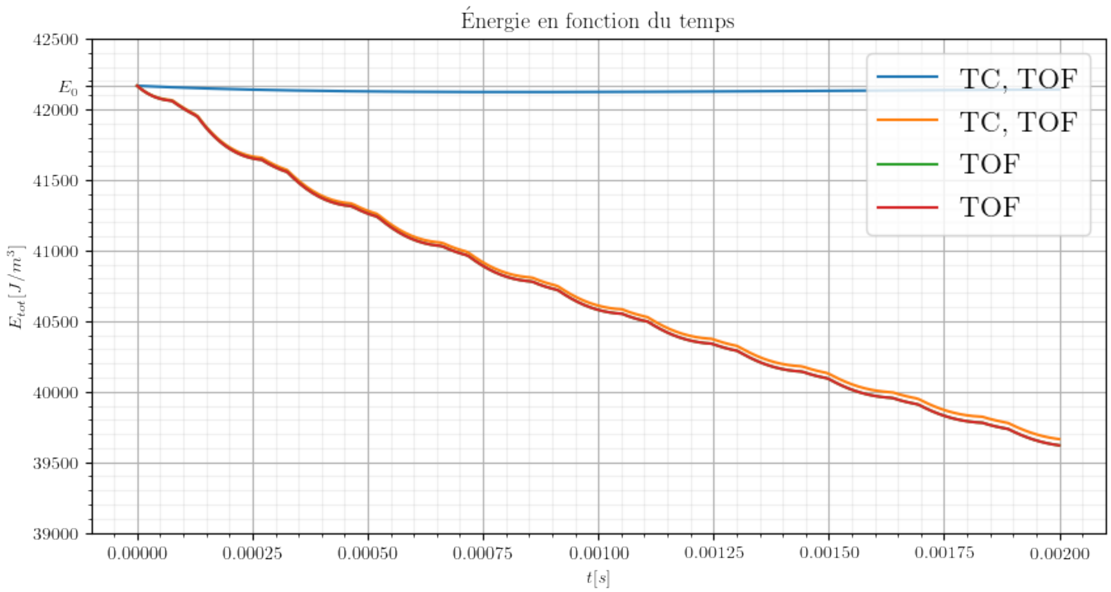
[5]:
```python
t_fin = 0.2
plot = Plotter("decale")
plot0 = Plotter("decale")
plot1 = Plotter("decale")
plot2 = Plotter("decale")
# plot5 = Plotter('decale')
fig1, ax1 = plt.subplots(1)
ax1.set_title("Énergie en fonction du temps")
ax1.set_xlabel(r"$t [s]$")
ax1.set_ylabel(r"$E_{tot} [J/m^3]$")
```

```
prob_ref = Problem(
    get_T_creneau, markers=markers, phy_prop=phy_prop_ref, num_prop=num_prop_rk4
)
E1 = prob_ref.energy
# print(prob_ref.name)
print("==========================")
t_ref, e_ref = prob_ref.timestep(
    t_fin=min(t_fin, t_fin_lim),
    n=n_lim,
    number_of_plots=1,
    plotter=[plot, plot0, plot1, plot2],
)
l = ax1.plot(t_ref, e_ref / (0.02 * 0.005 * 0.005), label=prob_ref.name)

n = len(e_ref)
i0 = int(n / 5)
dedt_adim = (
    (e_ref[-1] - e_ref[i0]) / (t_ref[-1] - t_ref[i0]) * prob_ref.dt / E1
)  # on a mult
print("dE*/dt* ref = %g" % dedt_adim)

prob0 = Problem(
    get_T_creneau, markers=markers, phy_prop=phy_prop, num_prop=num_prop_euler
)
E0 = prob0.energy
# print(prob0.name)
print("==========================")
t, e = prob0.timestep(
    t_fin=min(t_fin, t_fin_lim), n=n_lim, number_of_plots=1, plotter=[plot,␣
 ↪plot0]
)
l = ax1.plot(t, e / (0.02 * 0.005 * 0.005), label=prob0.name)

dedt_adim = (e[-1] - e[i0]) / (t[-1] - t[i0]) * prob0.dt / E0  # on a mult
print("dE*/dt* = %g" % dedt_adim)

prob1 = Problem(
    get_T_creneau, markers=markers, phy_prop=phy_prop, num_prop=num_prop_rk3
)
E0 = prob1.energy
# print(prob1.name)
print("==========================")
t, e = prob1.timestep(
    t_fin=min(t_fin, t_fin_lim), n=n_lim, number_of_plots=1, plotter=[plot,␣
 ↪plot1]
)
l = ax1.plot(t, e / (0.02 * 0.005 * 0.005), label=prob1.name)
```

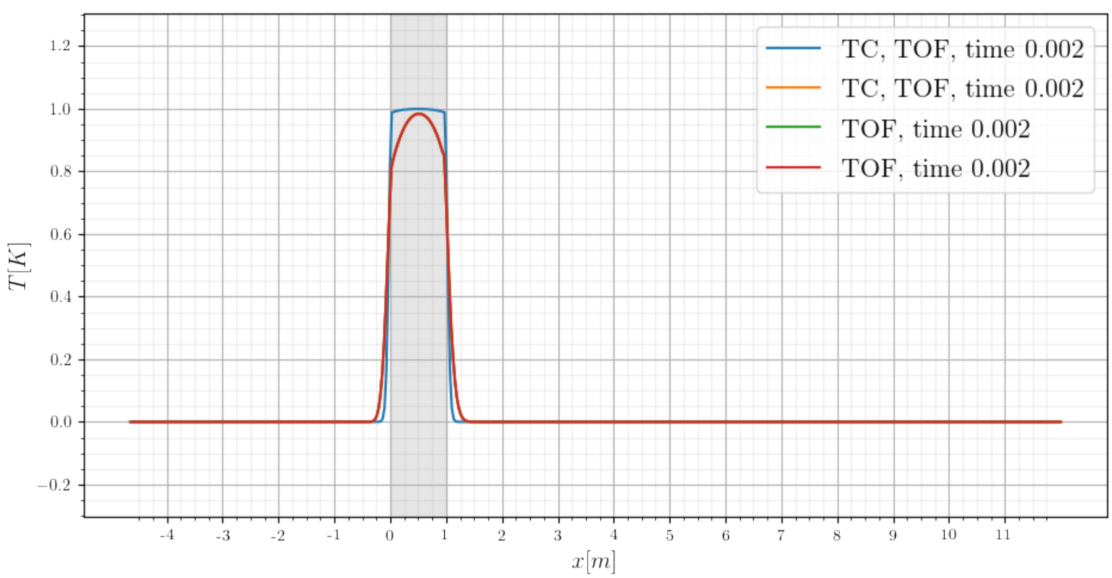
```
dedt_adim = (e[-1] - e[i0]) / (t[-1] - t[i0]) * prob1.dt / E0  # on a mult
print("dE*/dt* = %g" % dedt_adim)

prob2 = Problem(
    get_T_creneau, markers=markers, phy_prop=phy_prop, num_prop=num_prop_rk4
)
E0 = prob2.energy
# print(prob2.name)
print("===========================")
t, e = prob2.timestep(
    t_fin=min(t_fin, t_fin_lim), n=n_lim, number_of_plots=1, plotter=[plot,␣
 ↪plot2]
)
l = ax1.plot(t, e / (0.02 * 0.005 * 0.005), label=prob2.name)

dedt_adim = (e[-1] - e[i0]) / (t[-1] - t[i0]) * prob2.dt / E0  # on a mult
print("dE*/dt* = %g" % dedt_adim)

# Modif plot énergie

ax1.minorticks_on()
ax1.grid(b=True, which="major")
ax1.grid(b=True, which="minor", alpha=0.2)

fig1.canvas.draw()
labels = [item.get_text() for item in ax1.get_yticklabels()]
ticks = list(ax1.get_yticks())
ticks.append(E0 / (0.02 * 0.005**2))
labels.append(r"$E_0$")

ticks = ax1.set_yticks(ticks)
ticklab = ax1.set_yticklabels(labels)

handles, labels = ax1.get_legend_handles_labels()
labels[0] = "TC, " + labels[0]
labels[1] = "TC, " + labels[1]
ax1.legend(handles, labels)

# Modif plot température

handles, labels = plot.ax.get_legend_handles_labels()
labels[0] = "TC, " + labels[0]
labels[1] = "TC, " + labels[1]
plot.ax.legend(handles, labels)
plot.ax.set_xlabel(r"$x [m]$")
plot.ax.set_ylabel(r"$T [K]$")
```

```
TOF
===
dt fourier
6.918433404737903e-06
Db / dx = 30
==========================
dE*/dt* ref = 1.062e-06

TOF
===
dt fourier
6.918433404737903e-06
Db / dx = 30
==========================
dE*/dt* = -0.000176852

TOF
===
dt fourier
6.918433404737903e-06
Db / dx = 30
==========================
dE*/dt* = -0.000179679

TOF
===
dt fourier
6.918433404737903e-06
Db / dx = 30
==========================
dE*/dt* = -0.000179696
```

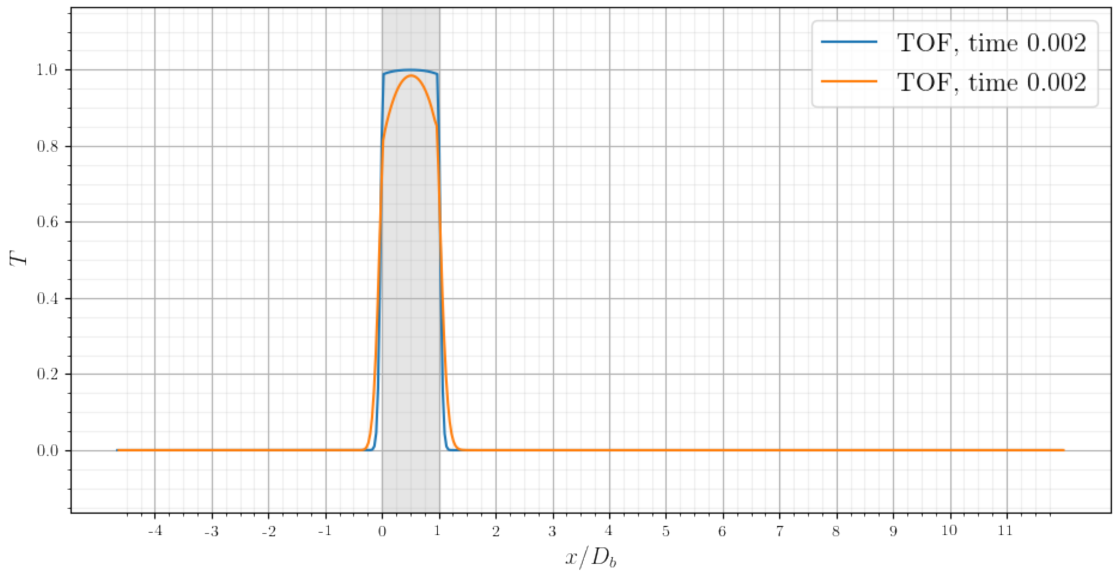
[5]:
```
Text(32.71132217265766, 0.5, '$T [K]$')
```



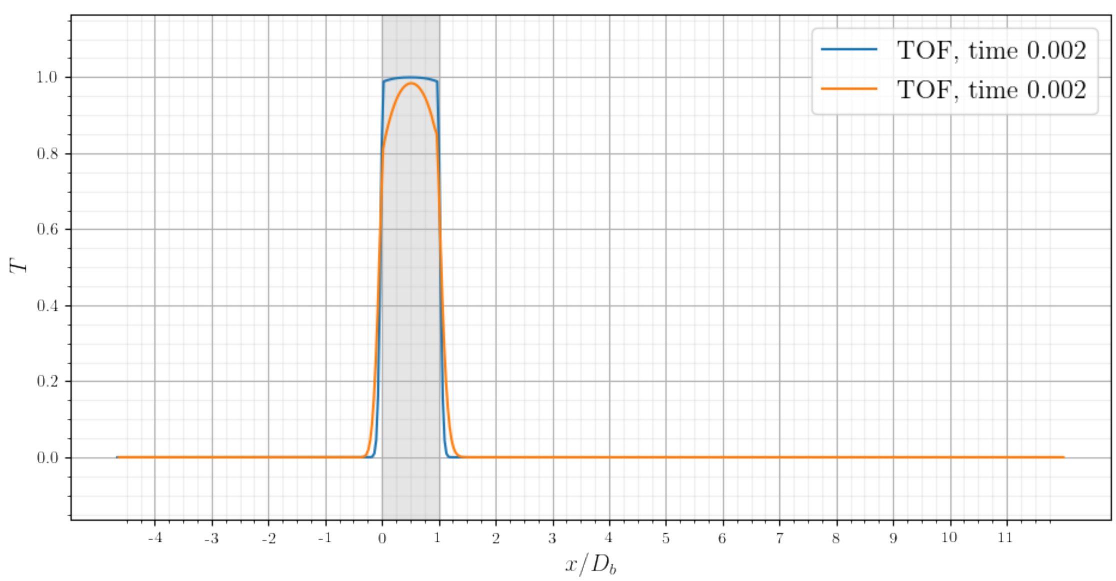
Énergie en fonction du temps
$E_{tot}[J/m^3]$
E_0
42500
42000
41500
41000
40500
40000
39500
39000
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
0.00200
$t[s]$
TC, TOF
TC, TOF
TOF
TOF



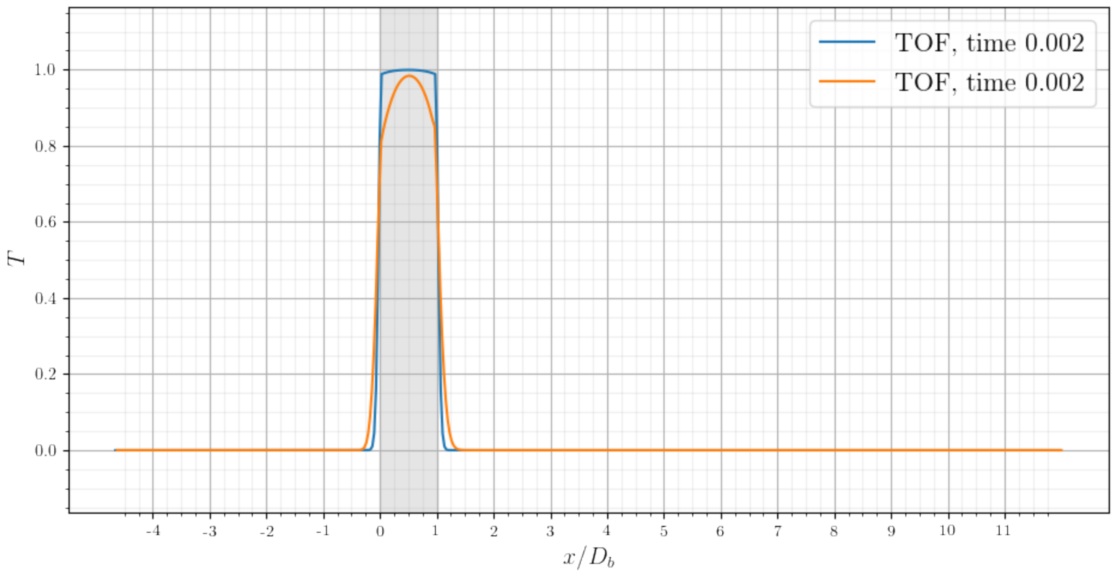
$T[K]$
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11
$x[m]$
TC, TOF, time 0.002
TC, TOF, time 0.002
TOF, time 0.002
TOF, time 0.002



TOF, time 0.002
TOF, time 0.002
T
1.0
0.8
0.6
0.4
0.2
0.0
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11
x/D_b



TOF, time 0.002
TOF, time 0.002
T
1.0
0.8
0.6
0.4
0.2
0.0
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11
x/D_b



TOF, time 0.002
TOF, time 0.002
1.0
0.8
0.6
0.4
0.2
0.0
T
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11
x/D_b