test_sans_saut

October 15, 2022

1 Test de la nouvelle classe TimeProblem sans saut de propriétés

[1]:
```
import sys
import os
from matplotlib.animation import FuncAnimation
from IPython.display import HTML

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

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

%matplotlib widget
rc("figure", figsize=(10, 5))
rc("figure", dpi=100)
rc("font", size=18)
rc("legend", fontsize=16)
rc("text.latex", preamble=r"\usepackage{siunitx}")
```

1.1 Test des 3 opérateurs à maillage constant

Ici on va réaliser une simulation sans saut de propriété avec différents opérateurs de convection.

La résolution se fait à chaque fois en WENO avec Euler explicite en temps.

[3]:
```
n_lim = 10**8
t_fin_lim = 0.2
```

[4]:
```
# d = 6./100*Delta/2.
# dx = 0.06 / 25.6 * 0.02
```

```
dx = 0.02 / 512
dx_fin = 0.06 / 60. * 0.02
cfl = 0.5

phy_prop_conv = PhysicalProperties(
    Delta=0.02,
    v=0.2,
    dS=0.005**2,
    lda1=5.5*10**-2,
    lda2=5.5*10**-2,
    rho_cp1=7.03*10**4,
    rho_cp2=7.03*10**4,
    diff=1.0,
    alpha=0.06,
    a_i=357.0,
)
phy_prop_no_conv = PhysicalProperties(
    Delta=0.02,
    v=0.0,
    dS=0.005**2,
    lda1=5.5*10**-2,
    lda2=5.5*10**-2,
    rho_cp1=7.03*10**4,
    rho_cp2=7.03*10**4,
    diff=1.0,
    alpha=0.06,
    a_i=357.0,
)
num_prop_ref = NumericalProperties(
    dx=dx_fin,
    schema="weno",
    time_scheme="rk3",
    phy_prop=phy_prop_no_conv,
    cfl=cfl,
    fo=0.5,
)
num_prop_weno = NumericalProperties(
    dx=dx,
    schema="weno",
    time_scheme="rk3",
    phy_prop=phy_prop_conv,
    cfl=cfl,
    fo=0.5,
)
num_prop_quick = NumericalProperties(
    dx=dx,
    schema="quick",
```

```
    time_scheme="rk3",
    phy_prop=phy_prop_conv,
    cfl=cfl,
    fo=0.5,
)
num_prop_upwind = NumericalProperties(
    dx=dx,
    schema="upwind",
    time_scheme="rk3",
    phy_prop=phy_prop_conv,
    cfl=cfl,
    fo=0.5,
)
# markers = Bulles(phy_prop=phy_prop_conv, x=num_prop.x, n_bulle=1)
markers = Bulles(phy_prop=phy_prop_conv, n_bulle=1)

print('dx : ', num_prop_quick.dx)
```

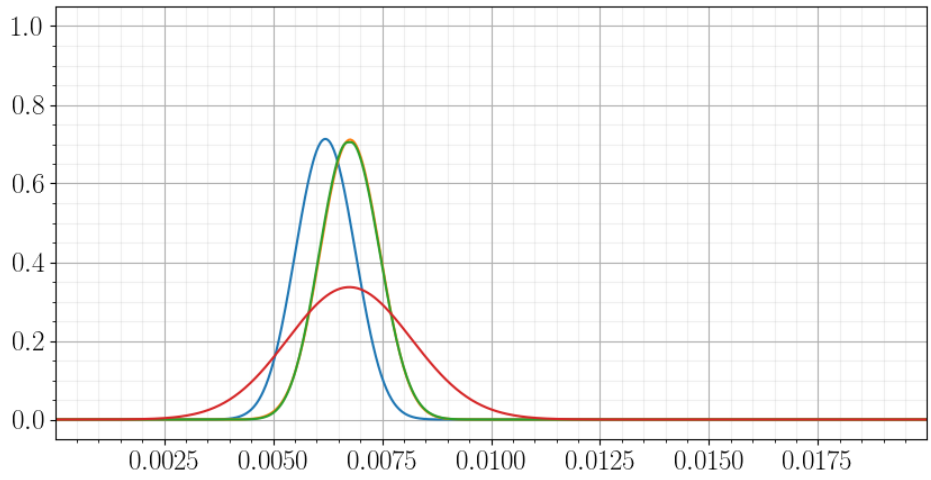
```
dx :  3.90625e-05
```

[16]:
```
prob_clean_weno_ref = TimeProblem(
    get_T_creneau, markers=markers, phy_prop=phy_prop_no_conv,␣
 ↪num_prop=num_prop_ref, plotter=[],
)
prob_clean_weno = TimeProblem(
    get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_weno, plotter=[],
)
prob_clean_quick = TimeProblem(
    get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_quick, plotter=[],
)
prob_clean_upwind = TimeProblem(
    get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_upwind, plotter=[],
)

list_prob = [prob_clean_weno_ref, prob_clean_weno, prob_clean_quick,␣
 ↪prob_clean_upwind]

fig, ax = plt.subplots()
n_parcours_domaine = 2
# n_frame = int(n_parcours_domaine * phy_prop_conv.Delta / phy_prop_conv.v / up.
 ↪dt) + 1
n_frame = 100
dt_max = max([pb.dt for pb in list_prob])
```

```
n_dt_per_frame = max(int(n_parcours_domaine * phy_prop_conv.Delta /␣
 ↪phy_prop_conv.v / dt_max / n_frame)+1, 1)
up = Compare(list_prob, ax, ylim=(-0.05, 1.05), n_dt_per_frame=n_dt_per_frame)
print()
print("Nombre de frames : ", n_frame)
anim = FuncAnimation(fig, up, frames=n_frame, interval=100, blit=True,␣
 ↪repeat=False, cache_frame_data=False)
HTML(anim.to_jshtml())
```

```
TOF
===
dt fourier
0.00025614840429655244
Db / dx = 59
Monofluid convection :  weno

TOF
===
dt cfl
9.765624999999999e-05
Db / dx = 30
Monofluid convection :  weno

TOF
===
dt cfl
9.765624999999999e-05
Db / dx = 30
Monofluid convection :  quick

TOF
===
dt cfl
9.765624999999999e-05
Db / dx = 30
Monofluid convection :  upwind

Nombre de frames :  100
```

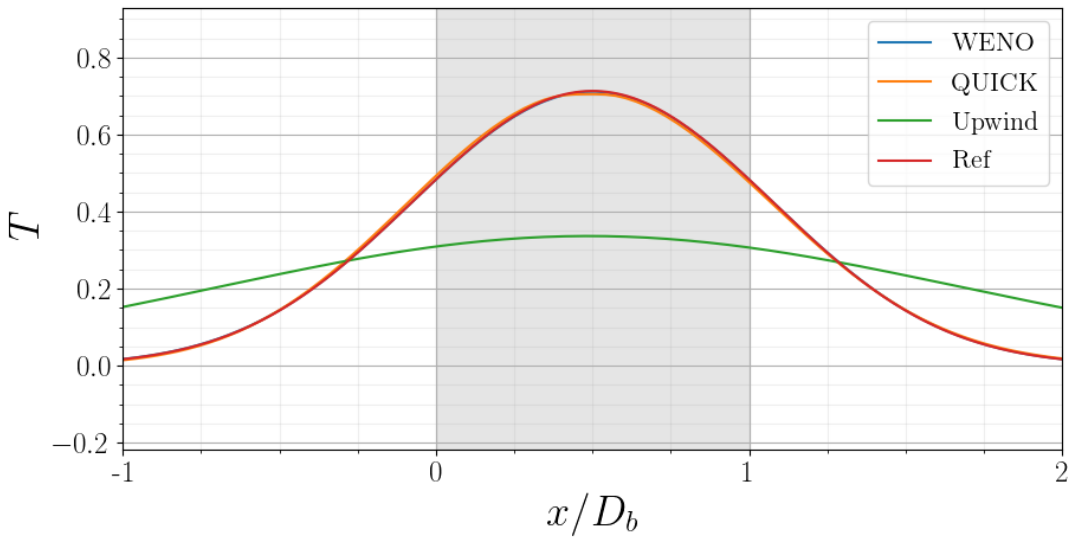
[16]: <IPython.core.display.HTML object>



[6]:
```
# t_fin = 10.0

# prob_clean_weno_ref = TimeProblem(
#     get_T_creneau, markers=markers, phy_prop=phy_prop_no_conv,␣
 ↪num_prop=num_prop_ref
# )
# t, e = prob_clean_weno_ref.timestep(
#     t_fin=min(t_fin, t_fin_lim),
#     n=n_lim,
#     number_of_plots=1,
#     plotter=Plotter("classic", ispretty=True),
# )
# prob_clean_weno = TimeProblem(
#     get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_weno
# )
# t, e = prob_clean_weno.timestep(
#     t_fin=min(t_fin, t_fin_lim),
#     n=n_lim,
#     number_of_plots=1,
#     plotter=Plotter("classic", ispretty=True),
# )

# prob_clean_quick = TimeProblem(
#     get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_quick
# )
# t, e = prob_clean_quick.timestep(
```

```
#     t_fin=min(t_fin, t_fin_lim),
#     n=n_lim,
#     number_of_plots=1,
#     plotter=Plotter("classic", ispretty=True),
# )

# prob_clean_upwind = TimeProblem(
#     get_T_creneau, markers=markers, phy_prop=phy_prop_conv,␣
 ↪num_prop=num_prop_upwind
# )
# t, e = prob_clean_upwind.timestep(
#     t_fin=min(t_fin, t_fin_lim),
#     n=n_lim,
#     number_of_plots=1,
#     plotter=Plotter("classic", ispretty=True),
# )
```

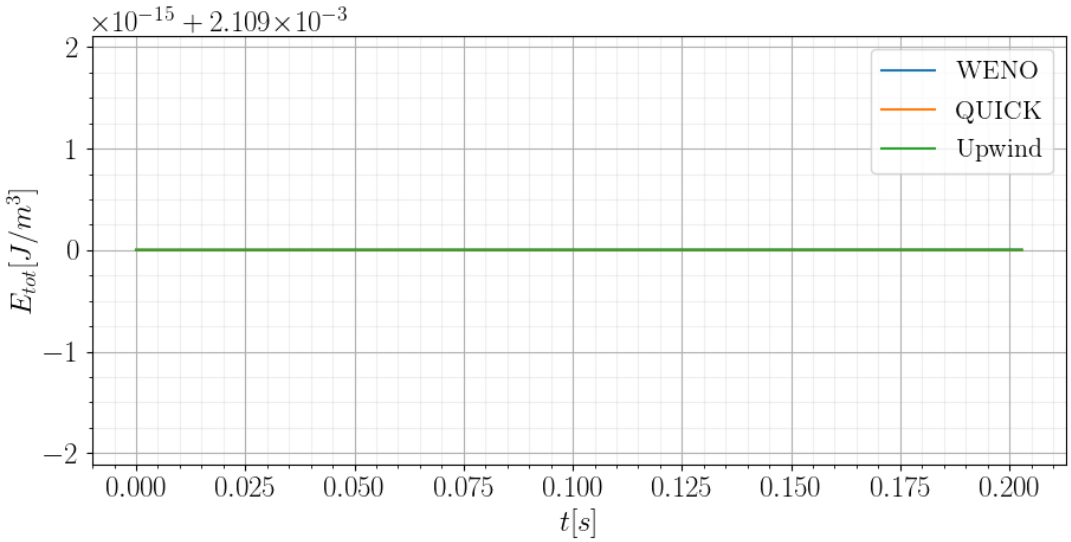
[7]:
```
plot = Plotter("decale", ispretty=True, zoom=(-1., 2.))
plot.plot(prob_clean_weno.problem_state, label='WENO')
plot.plot(prob_clean_quick.problem_state, label='QUICK')
plot.plot(prob_clean_upwind.problem_state, label='Upwind')
plot.plot(prob_clean_weno_ref.problem_state, label='Ref')
# plot.ax.set_ylim(0.1,0.4)
plot.fig.savefig(os.path.join(os.getcwd(), '../figures/sans_saut/profil_T.pdf'))
```



[8]:
```
plot_en = EnergiePlot()
plot_en.plot_tpb(prob_clean_weno, label='WENO')
```

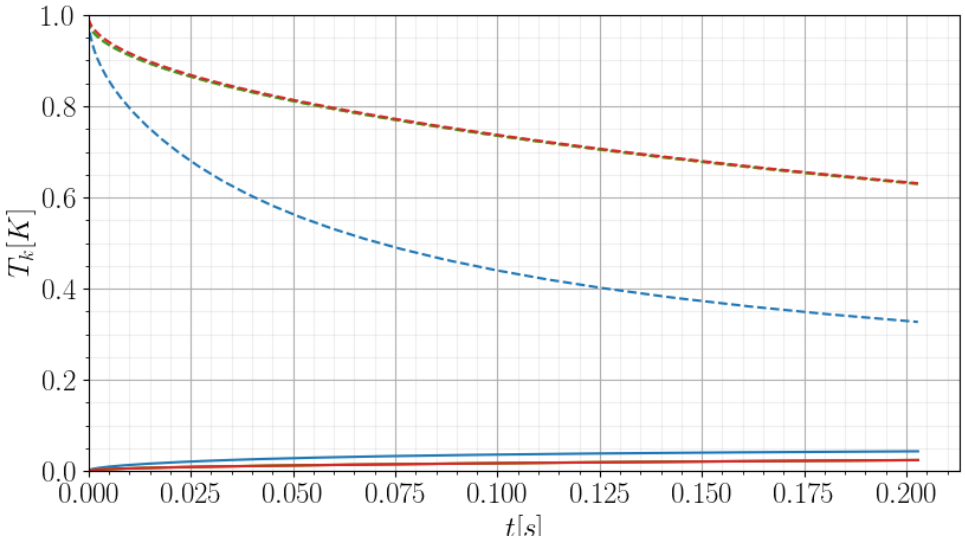
```
plot_en.plot_tpb(prob_clean_quick, label='QUICK')
plot_en.plot_tpb(prob_clean_upwind, label='Upwind')
plot_en.fig.savefig(os.path.join(os.getcwd(), '../figures/sans_saut/E_f_t.pdf'))
```

```
WENO
====
dE*/dt* = 1.23726e-19

QUICK
=====
dE*/dt* = -2.47453e-19

Upwind
======
dE*/dt* = 3.71179e-19
```



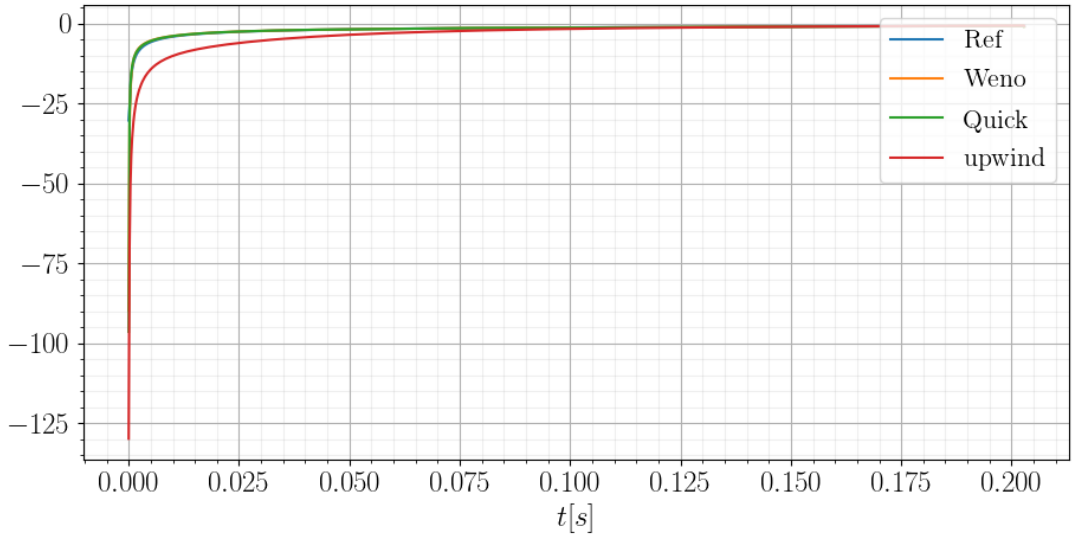
[9]:
```
plot_T = TemperaturePlot()
plot_T.plot_tpb(prob_clean_upwind, label=' Upwind')
plot_T.plot_tpb(prob_clean_quick, label=' QUICK')
plot_T.plot_tpb(prob_clean_weno, label=' WENO')
plot_T.plot_tpb(prob_clean_weno_ref, label=' Ref')
plot_T.ax.set_xlim(0, None)
plot_T.ax.set_ylim(0, 1)
# plot_T.fig.canvas.draw_idle()
# plot_T.fig.show()
# # plt.show()
# labels = plot_T.ax.get_yticklabels(minor=False)
```

```
# ticks = list(plot_T.ax.get_yticks(minor=False))
# ticks.append(plot_T.T_final)
# labels.append(r"$T_f$")
# plot_T.ax.set_yticks(ticks, minor=False)
# plot_T.ax.set_yticklabels(labels, minor=False)

plot_T.add_T_final()
plot_T.legend_Tl_Tv()
plot_T.fig.tight_layout()
plot_T.fig.savefig(os.path.join(os.getcwd(), '../figures/sans_saut/Tl_Tv.pdf'))
```



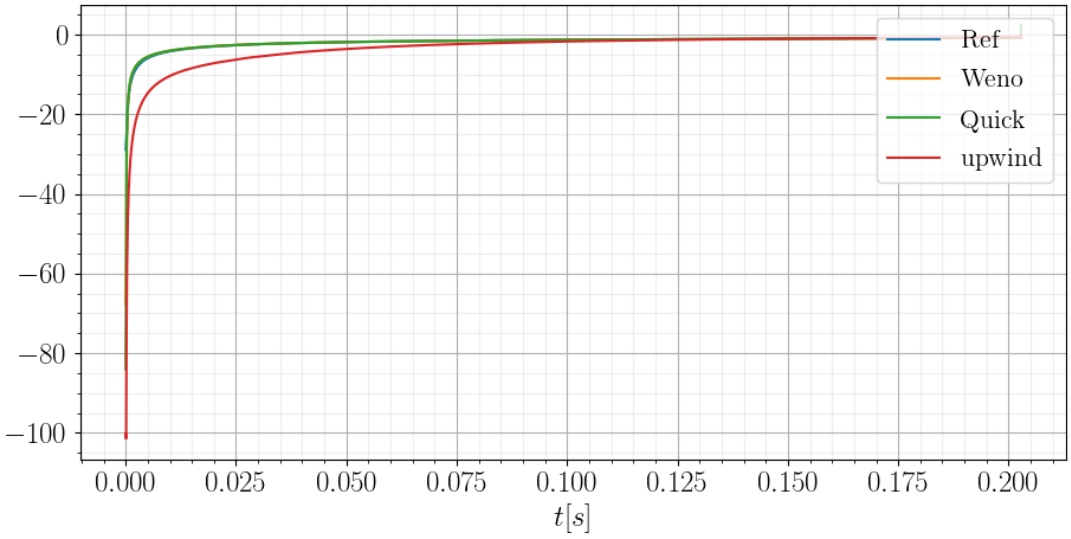
[10]:
```
def plot_dTdt(stat, plot, **args):
    dTdt = np.gradient(stat.Tv, stat.t)
    plot.ax.plot(stat.t, dTdt, **args)
```

[11]:
```
plot_dT = TimePlot()
plot_dTdt(prob_clean_weno_ref.stat, plot_dT, label="Ref")
plot_dTdt(prob_clean_weno.stat, plot_dT, label="Weno")
plot_dTdt(prob_clean_quick.stat, plot_dT, label="Quick")
plot_dTdt(prob_clean_upwind.stat, plot_dT, label="upwind")
le = plot_dT.ax.legend()
plot_dT.fig.tight_layout()
```



[12]:
```
def plot_dTdtp(stat, plot, **args):
    dTdt = np.gradient(stat.Tv_pure, stat.t)
    plot.ax.plot(stat.t, dTdt, **args)
```

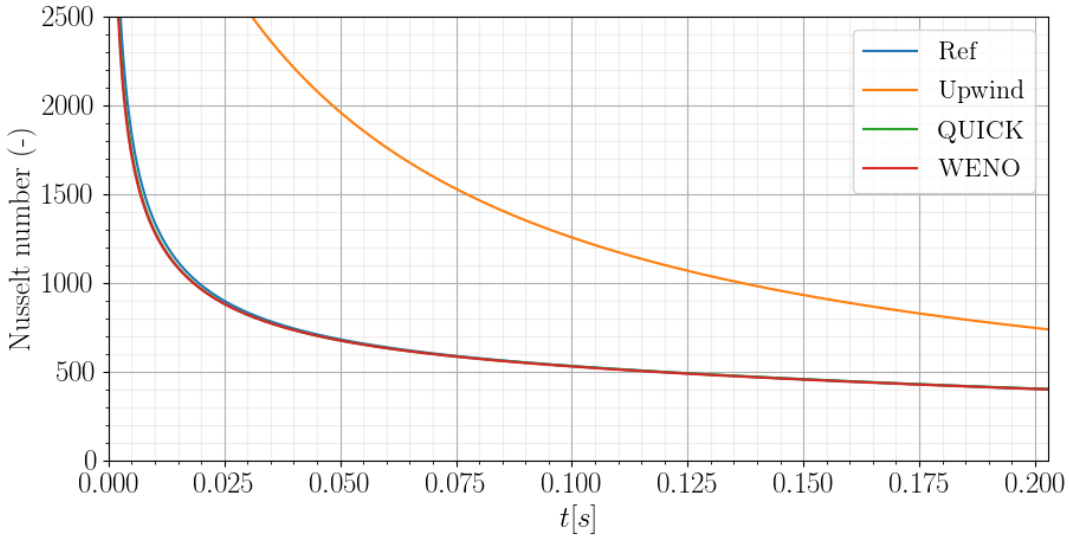
[13]:
```
plot_dTp = TimePlot()
plot_dTdtp(prob_clean_weno_ref.stat, plot_dTp, label="Ref")
plot_dTdtp(prob_clean_weno.stat, plot_dTp, label="Weno")
plot_dTdtp(prob_clean_quick.stat, plot_dTp, label="Quick")
plot_dTdtp(prob_clean_upwind.stat, plot_dTp, label="upwind")
le = plot_dTp.ax.legend()
plot_dTp.fig.tight_layout()
```



[14]:
```python
def compute_nu(stat, phy_prop):
    dTdt = np.gradient(stat.Tl, stat.t)
    DeltaT = stat.Tv - stat.Tl
    nu = (
        phy_prop.rho_cp1
        * dTdt
        * phy_prop.Delta
        * phy_prop.alpha
        * 3.
        / (phy_prop.lda1 * DeltaT)
    )
    return nu
```

[15]:
```python
plot_Nu = TimePlot()
plot_Nu.ax.plot(
    prob_clean_weno_ref.stat.t,
    compute_nu(prob_clean_weno_ref.stat, phy_prop_no_conv),
    label=r"Ref",
)
plot_Nu.ax.plot(
    prob_clean_upwind.stat.t[:-1],
    compute_nu(prob_clean_upwind.stat, phy_prop_conv)[:-1],
    label=r"Upwind",
)
plot_Nu.ax.plot(
    prob_clean_quick.stat.t[:-1],
    compute_nu(prob_clean_quick.stat, phy_prop_conv)[:-1],
    label=r"QUICK",
```

```
)
plot_Nu.ax.plot(
    prob_clean_weno.stat.t[:-1],
    compute_nu(prob_clean_weno.stat, phy_prop_conv)[:-1],
    label=r"WENO",
)
plot_Nu.ax.legend()
plot_Nu.ax.set_ylim(0.0, 2500)
plot_Nu.ax.set_xlim(0.0, prob_clean_quick.problem_state.time)
plot_Nu.ax.set_ylabel(r"Nusselt number (-)")
plot_Nu.fig.tight_layout()
plot_Nu.fig.savefig(os.path.join(os.getcwd(), '../figures/sans_saut/Nu.pdf'))
```



[ ]: