Metody Obliczeniowe Fizyki i Techniki
Laboratorium 3
Równanie falowe dla struny

Krzysztof Tondera III rok

26.04.2021

Celem ćwiczenia było rozwiązanie równania:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial t^2} \tag{1}$$

W tym celu zastosowano prędkościowy schemat Verleta:

$$u(x, t + \Delta t) = u(x, t) + \Delta t v(x, t) + \frac{1}{2} a(x, t) \Delta t^2 \tag{2}$$

$$\frac{\partial u}{\partial t} = v(x, t + \Delta t) = v(x, t) + \frac{\Delta t}{2}(a(x, t + \Delta t) + a(x, t)) \tag{3}$$

$$\frac{\partial^2 u}{\partial t^2} = a(x, t + \Delta t) \frac{u(x + \Delta x, t) + u(x - \Delta x, t) - 2u(x, t)}{\Delta x^2} \tag{4}$$

1 Zad1 - Sztywne Warunki

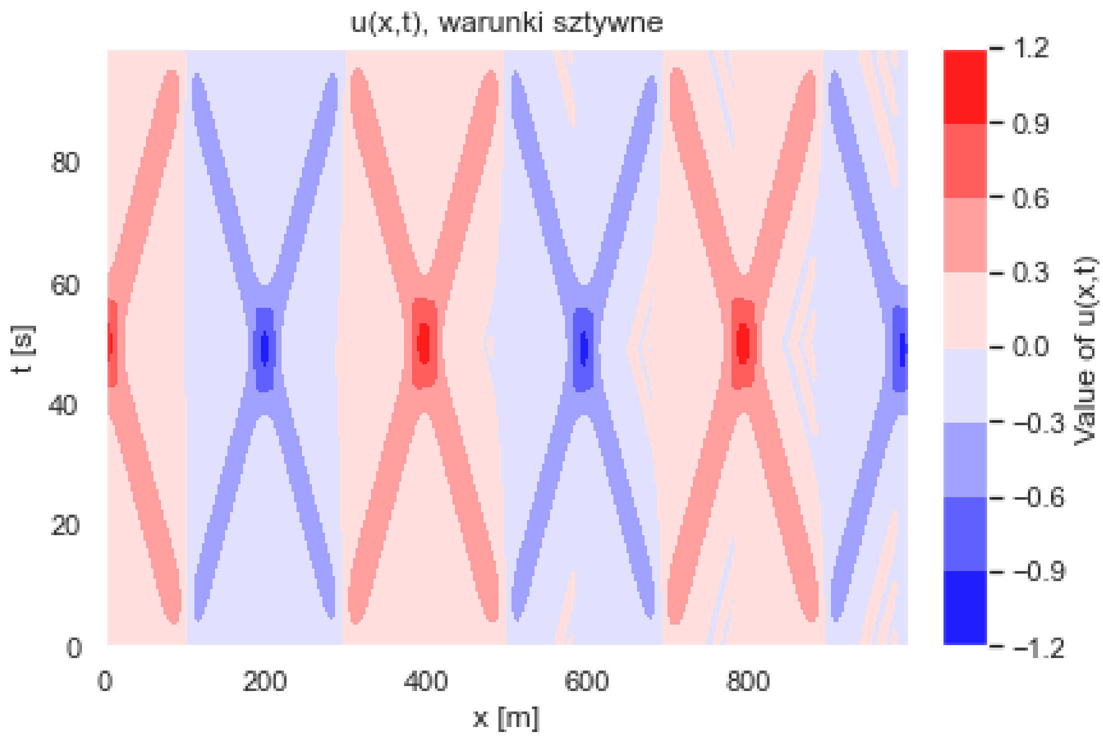
Rozwiązanie równania (1) dla warunków początkowych:

$$u_0(x) = exp(-100(x - 0.5)^2) \tag{5}$$

oraz,

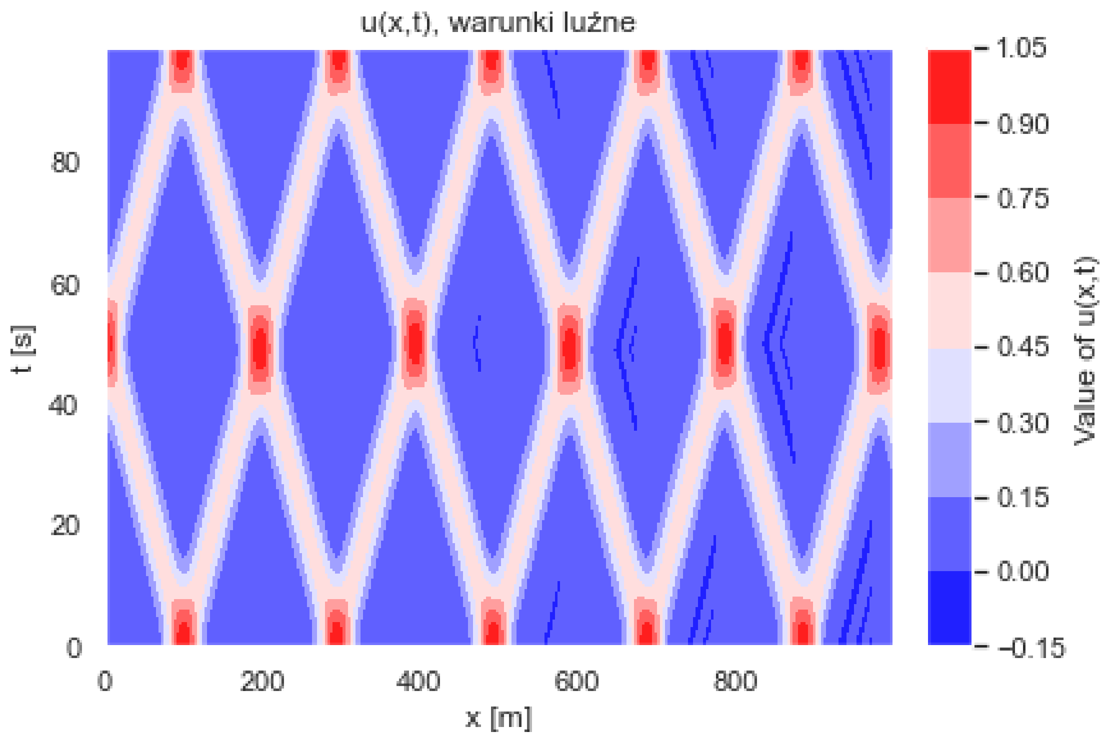
$$v_0(x) = 0 \tag{6}$$

[2]:
```
def fun_u0(x):
    return np.exp(-100*(x-0.5)**2)

def sztywne(x,t,dx,dt):
    v=np.zeros((len(x),len(t)))
    u=np.zeros((len(x),len(t)))
    a=np.zeros((len(x),len(t)))

    for i in range(len(x)):
        u[i,0]=fun_u0(i*dx)


    for j in range(1,len(t)):
        for i in range(1,len(x)-1):
            u[i,j]=u[i,j-1]+dt*v[i,j-1]+0.5*a[i,j-1]*dt**2
        for i in range(1,len(x)-1):
            a[i,j]=(u[i+1,j]+u[i-1,j]-2*u[i,j])/dx**2
        for i in range(1,len(x)-1):
            v[i,j]=v[i,j-1]+0.5*dt*(a[i,j]+a[i,j-1])

    return u,v
```



2 Zad2 - Luźne Warunki

[5]:
```
def luzne(x,t,dx,dt):
    v=np.zeros((len(x),len(t)))
    u=np.zeros((len(x),len(t)))
    a=np.zeros((len(x),len(t)))

    for i in range(len(x)):
        u[i,0]=fun_u0(i*dx)

    for j in range(1,len(t)):
        for i in range(1,len(x)-1):
            u[i,j]=u[i,j-1]+dt*v[i,j-1]+0.5*a[i,j-1]*dt**2
        u[0,j]=u[1,j]
        u[99,j]=u[98,j]
        for i in range(1,len(x)-1):
            a[i,j]=(u[i+1,j]+u[i-1,j]-2*u[i,j])/dx**2
        for i in range(1,len(x)-1):
            v[i,j]=v[i,j-1]+0.5*dt*(a[i,j]+a[i,j-1])

    return u,v
```



3 Zad3 - drgania tłumione

Do wzoru (1) wprowadzamy tłumienie drgań proporcjonalne do prędkości struny:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial t^2} - 2\beta \frac{\partial u}{\partial t} \tag{7}$$

gdzie β jest współczynnikiem tłumienia.
Przyspieszenie potrzebne do schematu Verleta dane jest przez:

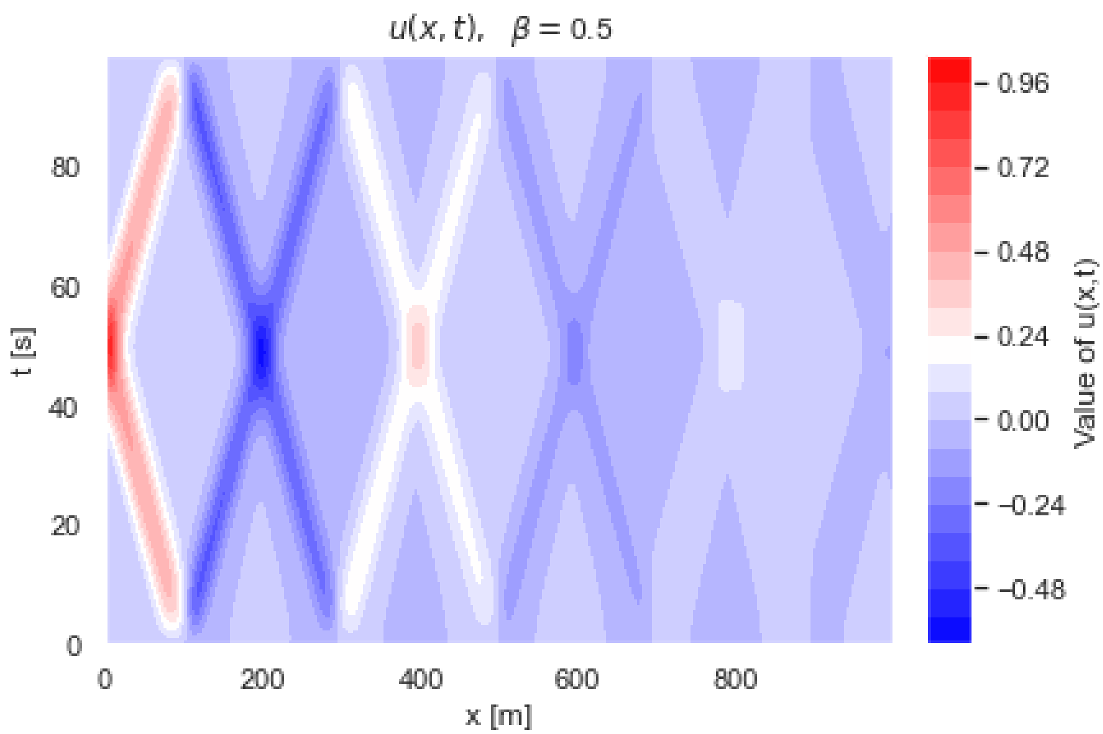
$$a_\beta = u''(x,t) - 2\beta v(x,t) \tag{8}$$

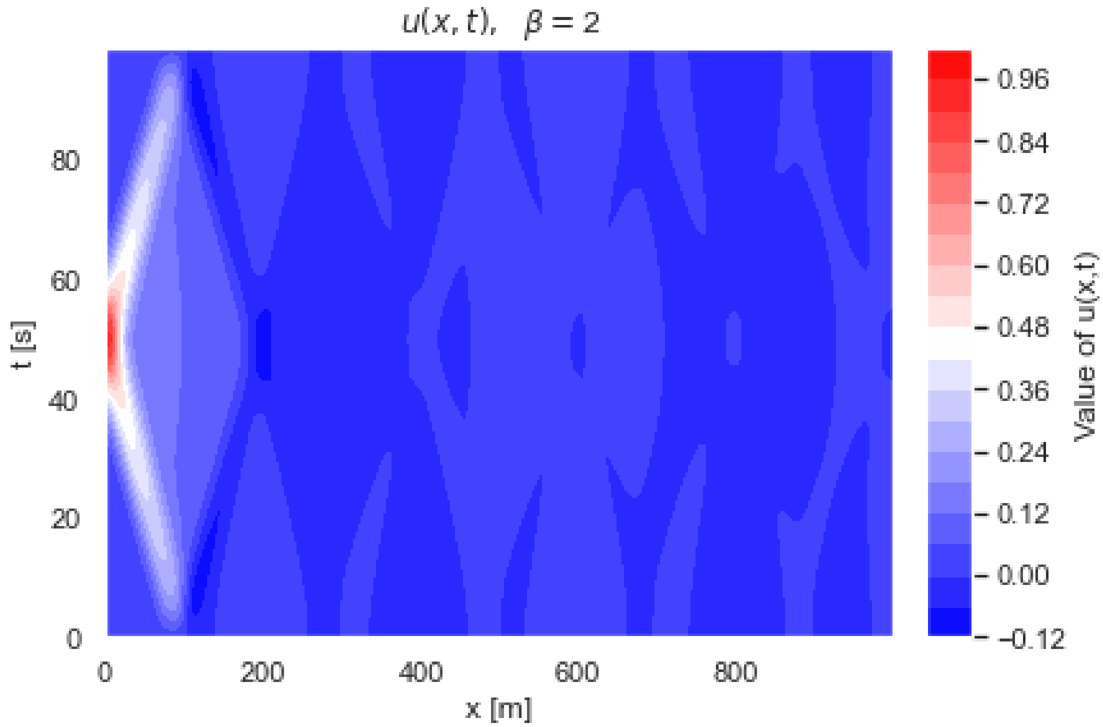
I wtedy przepis na zmianę prędkości przyjmuje postać:

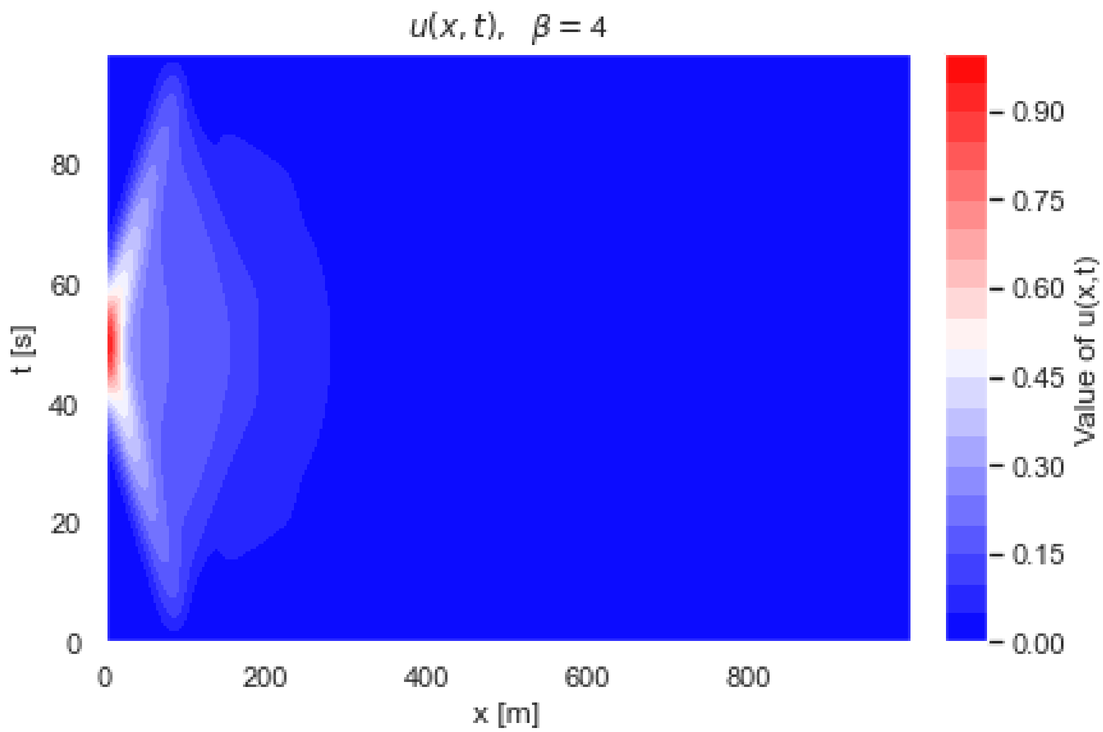
$$v(x,t+\Delta t) = [v(x,t) + \frac{\Delta t}{2}(u''(x,t+\Delta t) + a_\beta(x,t))]/(1+\beta\Delta t) \tag{9}$$

[7]:
```
def sztywne_damp(x,t,dx,dt,beta):
    v=np.zeros((len(x),len(t)))
    u=np.zeros((len(x),len(t)))
    a=np.zeros((len(x),len(t)))

    def a_b(u_tab,i,j,v):
        return a_fun(u_tab,i,j,v)-2*beta*v

    def a_fun(u_tab,i,j,v):
        if i>0 and i<len(x)-1:
            return (u_tab[i+1][j]+u_tab[i-1][j]-2*u_tab[i][j])/dx**2
```

```
        else:
            return 0

    for i in range(len(x)):
        u[i,0]=fun_u0(i*dx)


    for j in range(1,len(t)):
        for i in range(1,len(x)-1):
            u[i,j]=u[i,j-1]+dt*v[i,j-1]+0.5*a[i,j-1]*dt**2

        for i in range(1,len(x)-1):
            a[i,j]=(u[i+1,j]+u[i-1,j]-2*u[i,j])/dx**2

        for i in range(1,len(x)-1):
            v[i,j]=(v[i,j-1]+0.5*dt*(a[i,j]+a_b(u,i,j-1,v[i][j-1])))/(1+beta*dt)


    return u,v
```







4 Zad4 - Drgania wymuszone

Dodajemy siłę wymuszającą nadającą dodatkowe przyspieszenie strunie $a_F(x,t)$:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial t^2} - 2\beta \frac{\partial u}{\partial t} + a_F(x,t) \tag{10}$$

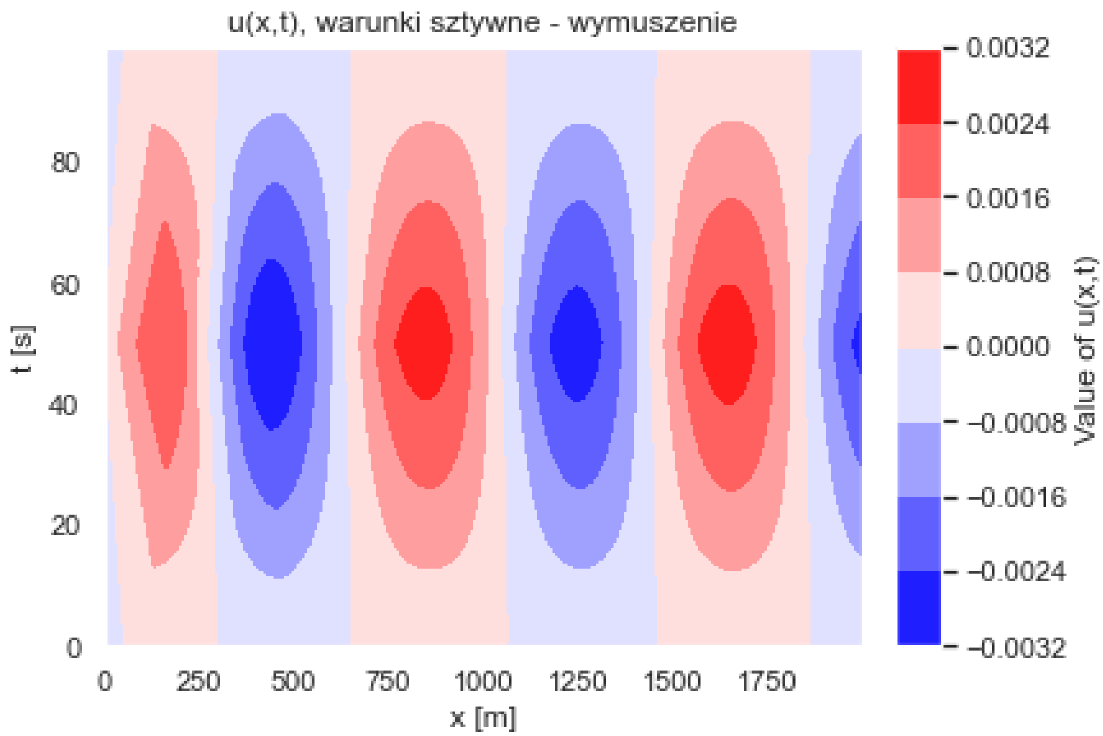
Siłę przykładamy punktowo:

$$a_F(x,t) = \begin{cases} cos(\omega t) \ dla \ x = x_0 \\ 0 \ dla \ x \neq x_0 \end{cases} \tag{11}$$

W obecności wymuszenia przepis na zmianę prędkości ma postać:

$$v(x, t+\Delta t) = [v(x,t) + \frac{\Delta t}{2}(u''(x, t+\Delta t) + a_\beta(x,t) + a_F(x, t+\Delta t) + a_F(x,t))]/(1+\beta\Delta t) \tag{12}$$

[9]:
```
def sztywne_forced(x,t,dx,dt,beta,x0,w):
    v=np.zeros((len(x),len(t)))
    u=np.zeros((len(x),len(t)))
    a=np.zeros((len(x),len(t)))

    def a_f(i,j):
        if(dx*i==x0):
            return np.cos(w*(j*dt))
        else:
            return 0

    def a_fun(u_tab,i,j,v):
        if i>0 and i<len(x)-1:
            return (u_tab[i+1][j]+u_tab[i-1][j]-2*u_tab[i][j])/dx**2
        else:
            return 0

    def a_b(u_tab,i,j,v):
        return a_fun(u_tab,i,j,v)-2*beta*v

    for j in range(1,len(t)-1):
        for i in range(1,len(x)-1):
            a[i][j]=a_fun(u,i,j-1,v[i][j-1])

        for i in range(1,len(x)-1):
            u[i][j]=u[i][j-1]+dt*v[i][j-1]+dt**2/2*a[i][j]

        for i in range(1,len(x)-1):
            v[i][j]=(v[i][j-1]+0.5*dt*(a[i][j]+a_b(u,i,j,v[i][j-1])+a_f(i,j)+a_f(i,j+1)))/
↪(1+beta*dt)

    return u,v
```


u(x,t), warunki sztywne - wymuszenie
t [s]
0
20
40
60
80
x [m]
0
250
500
750
1000
1250
1500
1750
Value of u(x,t)
0.0032
0.0024
0.0016
0.0008
0.0000
−0.0008
−0.0016
−0.0024
−0.0032

5 Zad 5 - rezonanse

Energia struny jest dana wyrażeniem:

$$E(t) = \frac{1}{2}\int_0^1 v^2(x,t)dx + \frac{1}{2}\int_0^1 (\frac{\partial u}{\partial t})^2 dx \tag{13}$$

Wyliczono średnią energię stanu ustalonego jako:

$$< E >= \frac{1}{t_2 - t_1}\int_{t_1}^{t_2} E(t)dt \tag{14}$$

dla t_1=16, t_2=20. Narysowano średnią energię w funkcji $\omega \in (0, 10\pi)$ i punkcie przyłożenia x_0=0.4.

[50]:
```
def rezonanse(x,t,dx,dt,beta,x0,w,t1,t2):
    v=np.zeros((len(x),len(t)))
    u=np.zeros((len(x),len(t)))
    a=np.zeros((len(x),len(t)))

    def a_f(i,j):
        if(i==40):
            return np.cos(w*(j*dt))
        else:
            return 0

    def a_fun(u_tab,i,j,v):
        if i>0 and i<len(x)-1:
            return (u_tab[i+1][j]+u_tab[i-1][j]-2*u_tab[i][j])/dx**2
        else:
            return 0

    def a_b(u_tab,i,j,v):
        return a_fun(u_tab,i,j,v)-2*beta*v


    for i in range(len(x)):
        u[0][i]=0

    E=0

    for j in range(len(t)):
    #for j in range(200):
        du=[]

        for i in range(1,len(x)-1):
            a[i][j]=a_fun(u,i,j-1,v[i][j-1])

        for i in range(1,len(x)-1):
            u[i][j]=u[i][j-1]+dt*v[i][j-1]+dt**2/2*a[i][j]

        for i in range(len(x)-1):
            if t2>j*dt>t1:
                E+=(u[i][j]-u[i-1][j])**2
```

```
        for i in range(1,len(x)-1):
            v[i][j]=(v[i][j-1]+0.5*dt*(a[i][j]+a_b(u,i,j,v[i][j-1])+a_f(i,j)+a_f(i,j+1)))/
↪(1+beta*dt)

            if t2>j*dt and j*dt>t1:
                E+=(v[i][j]**2)*dx

        #if t2>j*dt>t1:
            #E+=sum(du)

    return (E*0.5)/(t2-t1)
    #return u
```