ELTE Faculty of Science

Classical chaotic scattering

Alex Olar

2018

Abstract

The problem is the following. I projectile of mass `m` moves in a potential *V(x, y)* through space. The Newtonian equations describe the trajectory as:

$$\vec{F} = m\vec{a} \tag{1}$$

For a given potential:

$$-\frac{\partial V}{\partial x}\hat{i} - \frac{\partial V}{\partial y}\hat{j} = m\frac{d^2\vec{x}}{dt^2} \tag{2}$$

The given example for *V(x,y)* in the book [1]:

$$V(x, y) = \pm x^2 y^2 e^{-(x^2+y^2)} \tag{3}$$

However, I am planning on generalizing the algorithm to solve the ODEs for any given potential. On the other hand, using *well shaped* potentials must be taken into consideration since the idea behind the scattering [4] problem is modelling real world problems such as a pinball game, squash, nuclei and much more. So far I have taken into consideration the given example above and:

$$V(x, y) = \pm x^2 e^{-(x^2+y^2)} \tag{4}$$

Due to symmetry reasons the potential above is the same for a *x,y* change as it only means a different projectile direction in layman's term.

The second order ODEs after derivation are the following (for the given example):

$$m\frac{d^2x}{dt^2} = \mp 2y^2 x(1 - x^2)e^{-(x^2+y^2)} \tag{5}$$

$$m\frac{d^2y}{dt^2} = \mp 2x^2 y(1 - y^2)e^{-(x^2+y^2)} \tag{6}$$

Contents

I. Solution

The solution was computed with the *scipy* package and therefore the implementation was done in python. *Scipy* provides an interface for the LSODA solver from implemented in FORTRAN that is capable of identifying the type of the first order differential equation (stiff or non stiff) and solving it appropriately. It is way better than a basic Runge-Kutta method and thus gives more appropriate results and runs faster.

Listing 1: Derivative calculations

```
def scatter(X, t, m, sign):
x, y, u, v = X
# Hard coded potential
dVdx = (1./m)*sign*2*y**2*x*(1-x**2)*np.exp(-(x**2+y**2))
dVdy = (1./m)*sign*2*x**2*y*(1-y**2)*np.exp(-(x**2+y**2))
# Derivative
dXdt = [u, v, -dVdx, -dVdy]
return dXdt
```

It can be seen how simple it is to input a differential equation. This is the format in which the input function should have been constructed and the method from *scipy* simply solves it as:

Listing 2: Equation solver

```
solution = eqsolver(scatter, X0, t, args=(m, sign))
```

Where **X0** are the initial *x, y, u, v*. Since the second order differential equations were separated as:

$$
\begin{aligned}
I. \quad m\frac{d^2x}{dt^2} &= \mp 2y^2x(1-x^2)e^{-(x^2+y^2)} \\
\dot{x}(t) &= u(t) \\
m\dot{u}(t) &= \mp 2y^2x(1-x^2)e^{-(x^2+y^2)}
\end{aligned}
$$

$$
\begin{aligned}
II. \quad m\frac{d^2y}{dt^2} &= \mp 2x^2y(1-y^2)e^{-(x^2+y^2)} \\
\dot{y}(t) &= v(t) \\
m\dot{v}(t) &= \mp 2x^2y(1-y^2)e^{-(x^2+y^2)}
\end{aligned}
$$

Using the $(+)$ signed potential and initial conditions recommended in the book:

$$x_0 = [0., 1.] \quad [0., 5.] \quad y_0 = -10. \quad u_0 = 0. \quad v_0 = 0.5$$

I am only using positive impact parameters as symmetry reasons provide that every derivative value will be anti-symmetric to the y axis therefore I skip the unnecessary calculations.

I.1 Potentials

Book example (+)

Visualizing the potential the following results were acquired:

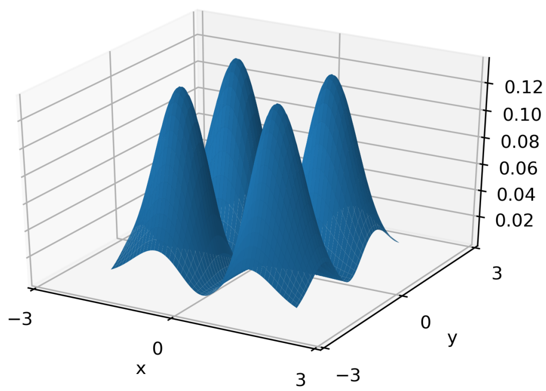

Figure 1: $V(x, y) = x^2y^2e^{-(x^2+y^2)}$

The impact parameter is the coordinate x_0. The initial angle of the velocity is $\frac{\pi}{2}$ as it points in the direction of the y-axis. The system is very dependent on the initial speed results in the elimination of the effect of the potential. I suppose this is due to the fact that the point does not stays enough in the zone where the potential is the most effective. For didactic purposes I made two plots showing the chaotic results. Setting $x_0 \in [0., 5.]$ one can see that the scattering is chaotic in the $[0., 1.]$ region. Using 1001 steps and setting the experiment duration to 1000 seconds and 4001 steps between *[0, 5]* and *[0, 1]*.

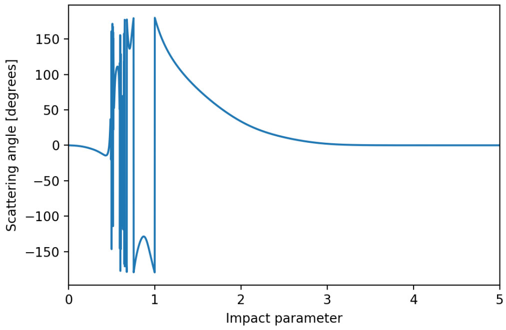

Figure 2: $x_0 \in [0., 5.]$

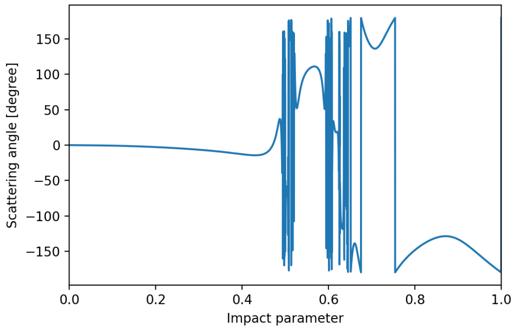

Figure 3: $x_0 \in [0., 1.]$

Book example (-)

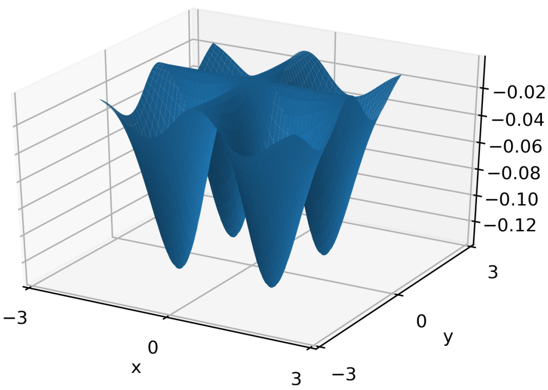

Figure 4: $V(x, y) = -x^2y^2e^{-(x^2+y^2)}$

Using the same parameters as before, only changed the sign of the potential.

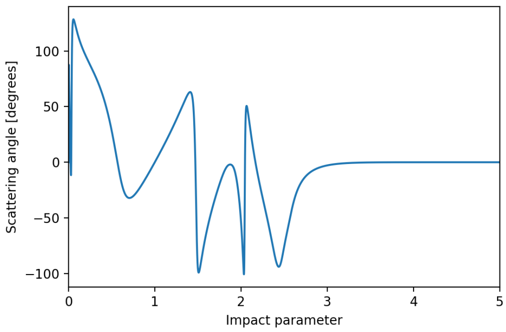

Figure 5: $x_0 \in [0., 5.]$

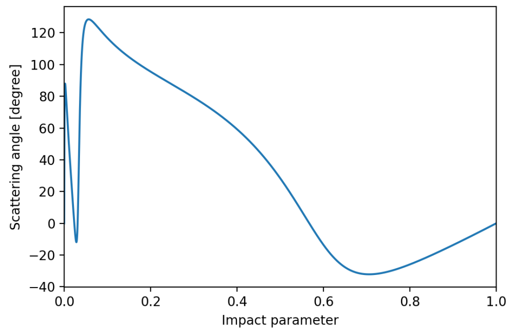

Figure 6: $x_0 \in [0., 1.]$

Custom potential (+)/(-)

I used the same constants and methods to do the experiment and the plots. The previously shown *scatter* function was obviously changed since I needed to update the potential and its derivatives:

$$I. \quad m\dot{u}(t) = \mp 2x(1-x^2)e^{-(x^2+y^2)}$$

$$II. \quad m\dot{v}(t) = \pm 2x^2 y e^{-(x^2+y^2)}$$

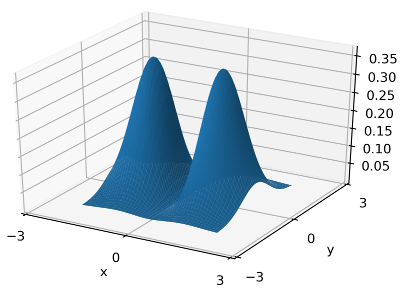

Figure 7: $V(x,y) = x^2 e^{-(x^2+y^2)}$

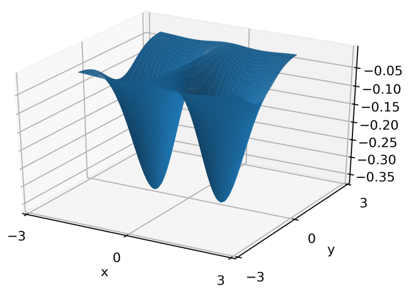

Figure 8: $V(x,y) = -x^2 e^{-(x^2+y^2)}$

Due to symmetry reasons changing **x-y** only rotates the potential and therefore it should not be updated, it can be tested with varying the initial conditions. The results are shown for both (+) and (-) signed potentials.

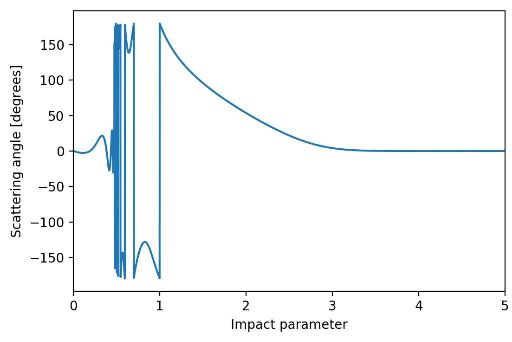

Figure 9: (+) $x_0 \in [0., 5.]$

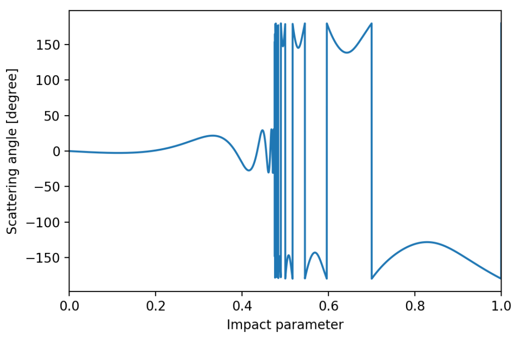

Figure 10: (+) $x_0 \in [0., 1.]$

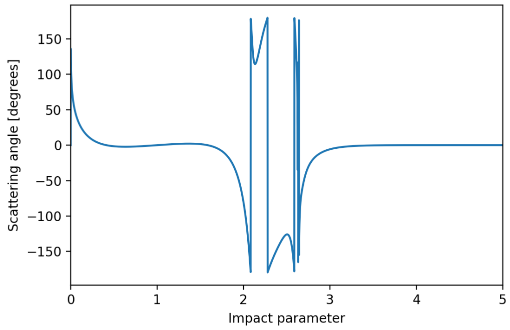

Figure 11: (-) $x_0 \in [0., 5.]$

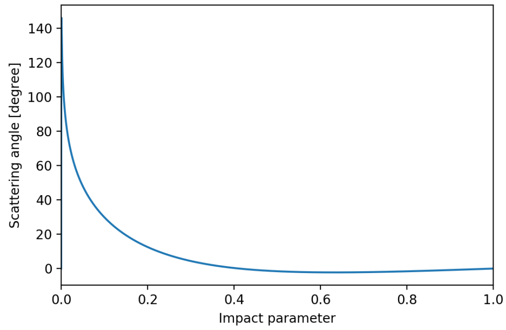

Figure 12: (-) $x_0 \in [0., 1.]$

II. The cross section

Using the acquired impact parameters and of course the acquired angles the cross section is the following function:

$$\sigma(\vartheta) = \left|\frac{d\vartheta}{db}\right| \frac{b}{sin(\vartheta)} \tag{7}$$

The derivative of theta could be easilly acquired by calculating:

$$\frac{d\vartheta}{db} \approx \frac{\Delta\vartheta}{\Delta b} = \frac{\vartheta_{i+1} - \vartheta_i}{\Delta b} \tag{8}$$

Acquiring the set from the outputted data b, ϑ I calculated the cross section for each potential for all signs.

The following cross sections were acquired:

II.1 Potentials

Book example (+)/(-)

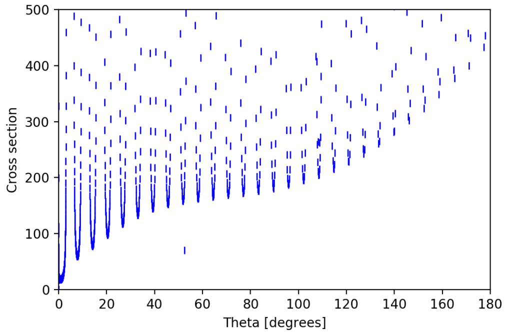

Figure 13: (+) $\vartheta \in [0., 180.]$

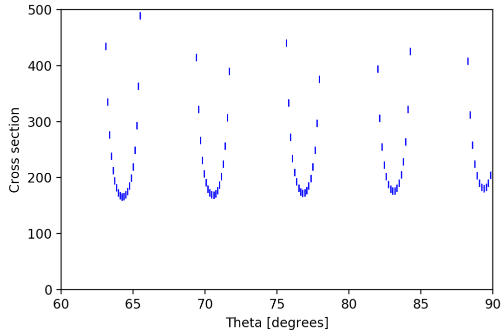

Figure 14: (+) $\vartheta \in [60., 90.]$

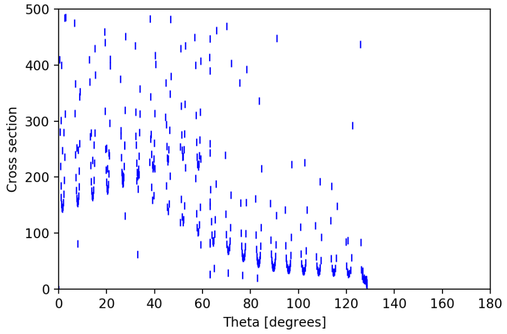

Figure 15: (-) $\vartheta \in [0., 180.]$

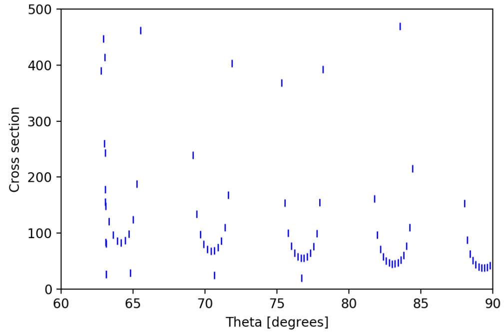

Figure 16: (-) $\vartheta \in [60., 90.]$

Custom (+)/(-)

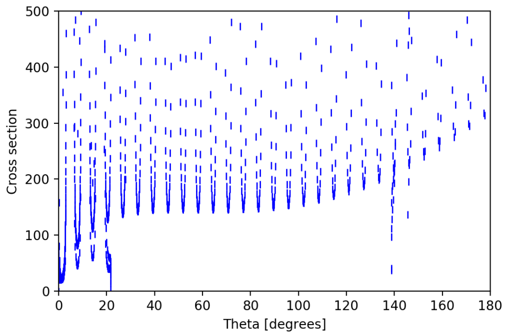

Figure 17: (+) $\vartheta \in [0., 180.]$

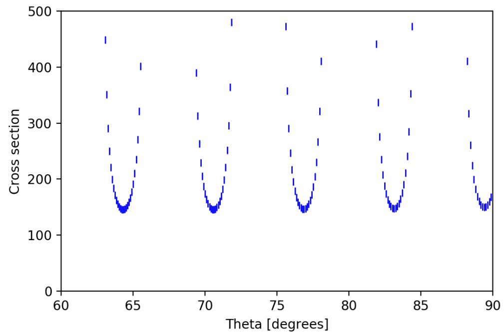

Figure 18: (+) $\vartheta \in [60., 90.]$

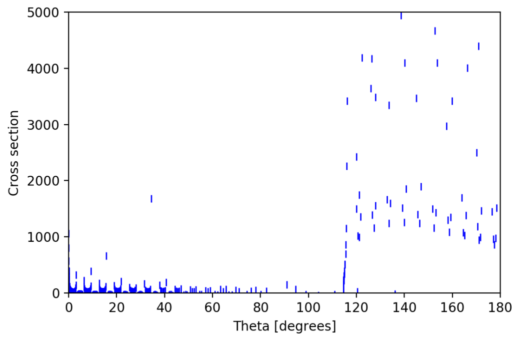

Figure 19: (-) $\vartheta \in [0., 180.]$

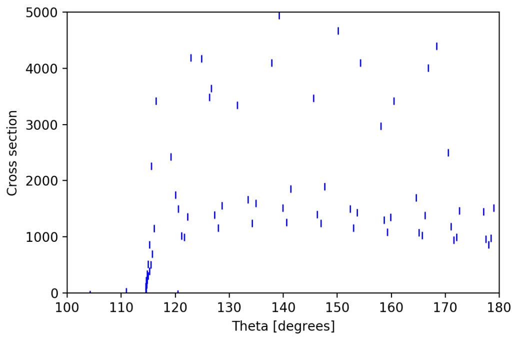

Figure 20: (-) $\vartheta \in [100., 180.]$

II.2 Interpretation

I wish I could give a satisfactory explanation to the cross sections seen above, however I cannot. I can check them qualitatively though. It can be seen clearly that for attractive potentials the cross section is getting smaller as the scattering angle gets bigger, this is due to the fact that I calculated the *atan2* of v_y, v_x and not the other way around, but the trend is trivially there. For repressive potentials the cross section is much higher in the angles under (inverse degree scale again) 90 degrees, this can be regarded as back scattering.

III. Phase space diagrams

For one scattering, only for visualization:

III.1 Potentials

Book example (+)/(-)

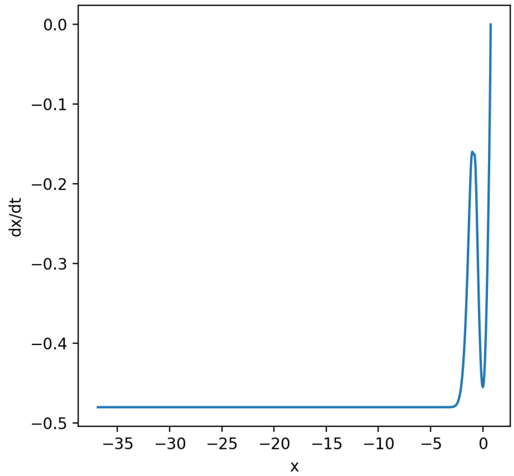

Figure 21: (+) $x(t), \dot{x}(t)$

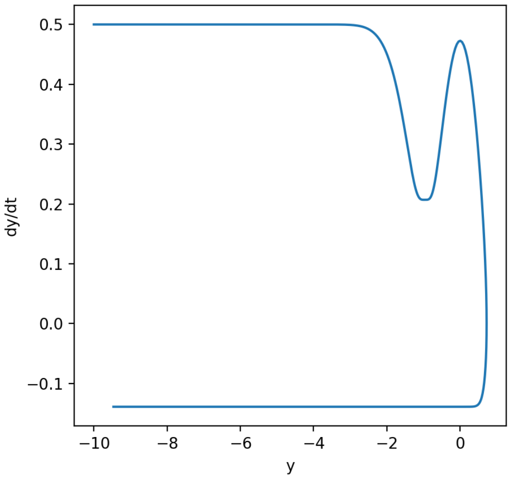

Figure 22: (+) $y(t), \dot{y}(t)$

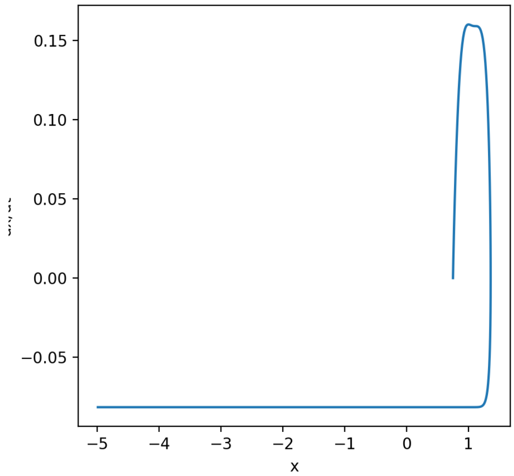

Figure 23: (-) $x(t), \dot{x}(t)$

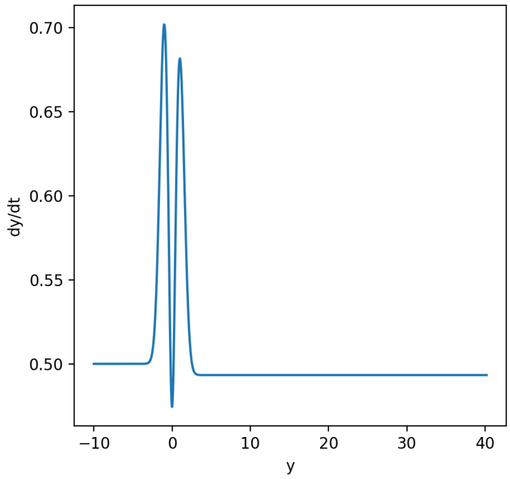

Figure 24: (-) $y(t), \dot{y}(t)$

Custom example (+)/(-)

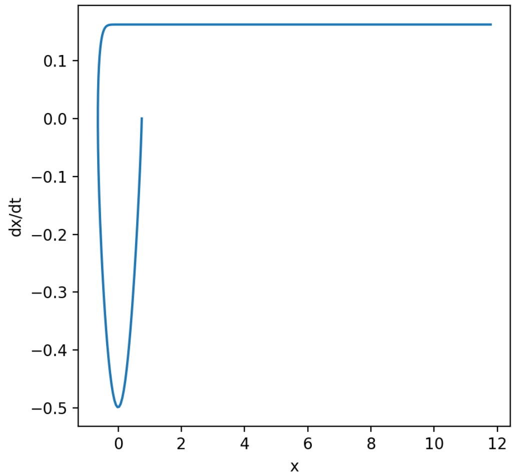

Figure 25: (+) $x(t), \dot{x}(t)$

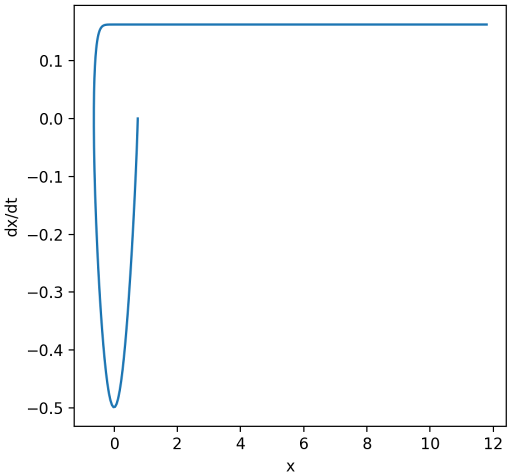

Figure 26: (+) $y(t), \dot{y}(t)$

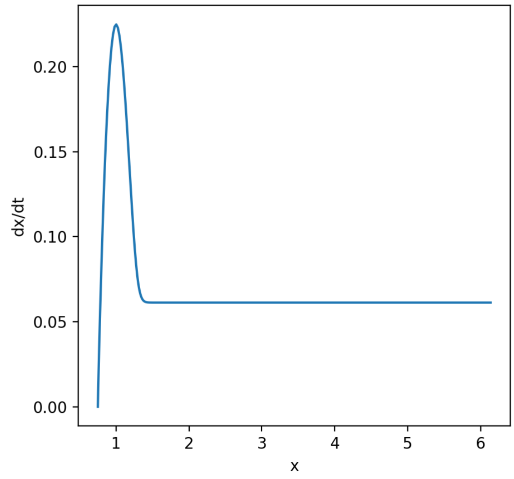

Figure 27: (-) $x(t), \dot{x}(t)$

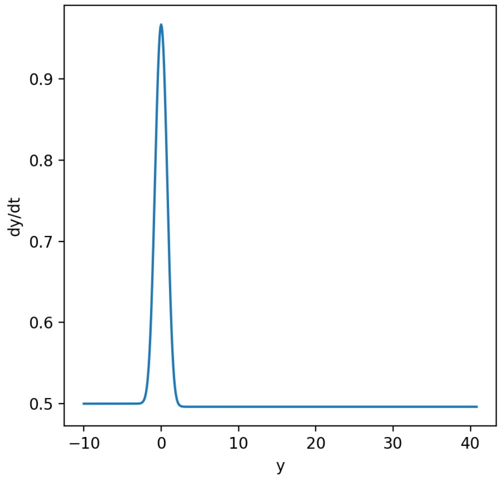

Figure 28: (-) $y(t), \dot{y}(t)$

IV. Overview

I solved the differential equation system efficiently using the odeint solver [3], although I have not tried different methods I am sure that real improvement can't be done for that.

However, I am certain that my cross section calculations are not the best, they could be improved by calculating much smaller regions for the impact parameter and concatenating the results. It can even be parallerized since the computation does not require to access the same memory regions.

There are computational errors evidently as I am dividing by sign of unknown parameters, therefore I am sometimes dividing by zero, luckily numpy took care of that efficiently.

For attractive potentials there must be some kind error that I have missed since my results seem wrong.

My code is attached at the end with a link pointing to my GitHub repository [2].

References

[1] Rubin H. Landau. A survey of computational physics. 2011.

[2] Alex Olar. Chaotic scattering - github repository. 2018.

[3] Scipy. Scipy docs for odeint. 2018.

[4] Wikipedia. Chaotic scattering. 2018.