Doppler Effect Simulation

Yousef Sadeghi
Department of Physics, Faculty of Science, Ferdowsi University of Mashhad
Arash T. Jamshidi
Department of Physics, Faculty of Science, Ferdowsi University of Mashhad

I. INTRODUCTION

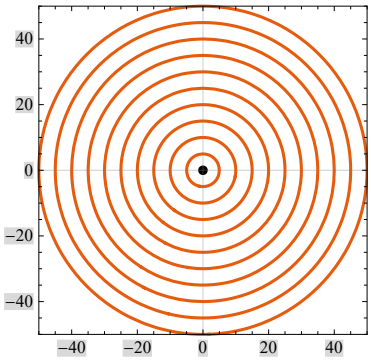
The Doppler effect is the change in frequency of a wave in relation to an observer who is moving relative to the wave source. It is named after the Austrian physicist Christian Doppler, who described the phenomenon in 1842. A common example of Doppler effect is the change of pitch heard when a vehicle sounding a horn approaches and recedes from an observer. Compared to the emitted frequency, the received frequency is higher during the approach, identical at the instant of passing by, and lower during the recession.[1]

The reason for the Doppler effect is that when the source of the waves is moving towards the observer, each successive wave crest is emitted from a position closer to the observer than the crest of the previous wave. Therefore, each wave takes slightly less time to reach the observer than the previous wave. Hence, the time between the arrivals of successive wave crests at the observer is reduced, causing an increase in the frequency.

II. Stationary Source

The program produces multiple circles with the same center and the radius of each circle is determined by variable ν so that radius difference between two consecutive circles is ν units, Circles visualize wave crests.

```
ν := 5
(*scale*)
f01[r_] := ContourPlot[x^2 + y^2 == (r)^2, {x, -10*ν, 10*ν}, {y, -10*ν, 10*ν},
  PlotTheme → "Scientific", PlotRange → {{-10*ν, 10*ν}, {-10*ν, 10*ν}}]
(*waves*)
g := Graphics[{AbsolutePointSize[5], Point[{0, 0}]}]
(*source*)
Manipulate[Show[
  f01[r],
  If[r ≥ ν, f01[r - ν], g],
  If[r ≥ 2*ν, f01[r - 2*ν], g],
  If[r ≥ 3*ν, f01[r - 3*ν], g],
  If[r ≥ 4*ν, f01[r - 4*ν], g],
  If[r ≥ 5*ν, f01[r - 5*ν], g],
  If[r ≥ 6*ν, f01[r - 6*ν], g],
  If[r ≥ 7*ν, f01[r - 7*ν], g],
  If[r ≥ 8*ν, f01[r - 8*ν], g],
  If[r ≥ 9*ν, f01[r - 9*ν], g],
  g, Background → White],
 {r, 0.1, 10*ν, 1}]
```



Stationary sound source produces sound waves at a constant frequency, and the wave-fronts propagate symmetrically away from the source at a constant speed. All observers will hear the same frequency, which will be equal to the actual frequency of the source.

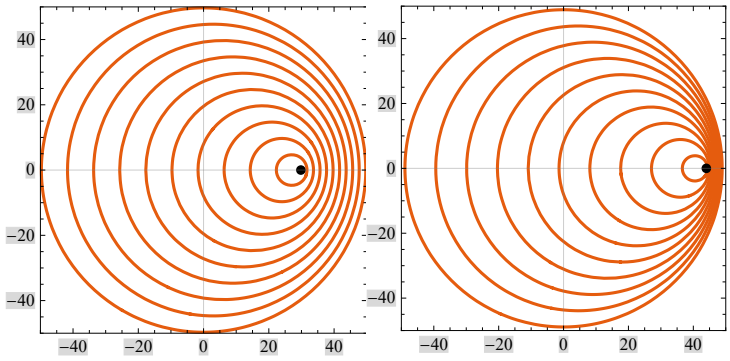
III. Moving Source

A. Two-Dimensional Space

The program produces multiple circles with different coordinates which shows us movement of the source, variables "vx" and "vy" determine velocity of the source respectively in x-direction and y-direction. Circles visualize wave crests which Expand at rate of "cs" that represents speed of sound.

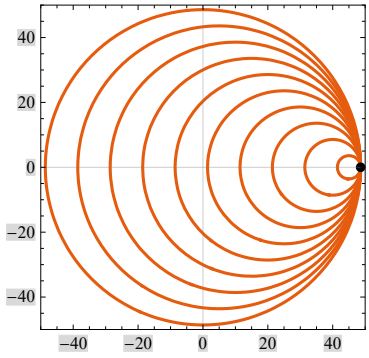
In[ ]:=

```
vx := 1.4
(*velocity in the x direction*)
vy := 0
(*velocity in the y direction*)
cs := 1
(*speed of wave / scale*)
f02[Xv_, Yv_, r_] :=
 ContourPlot[(x - Xv)^2 + (y - Yv)^2 == (r)^2, {x, -10*cs, 10*cs}, {y, -10*cs, 10*cs},
  PlotTheme → "Scientific", PlotRange → {{-10*cs, 10*cs}, {-10*cs, 10*cs}}]
(*waves*)
p01 := Plot[0, {x, 0, 0.0001}]
(*just a point (doesn't matter)*)
s2d[r_] := Graphics[{AbsolutePointSize[5], Black, Point[{vx*r/cs, vy*r/cs}]}]
(*source*)
Manipulate[Show[
  f02[0*vx, 0*vy, r],
  If[r ≥ cs, f02[1*vx, 1*vy, r - cs], p01],
  If[r ≥ 2*cs, f02[2*vx, 2*vy, r - 2*cs], p01],
  If[r ≥ 3*cs, f02[3*vx, 3*vy, r - 3*cs], p01],
  If[r ≥ 4*cs, f02[4*vx, 4*vy, r - 4*cs], p01],
  If[r ≥ 5*cs, f02[5*vx, 5*vy, r - 5*cs], p01],
  If[r ≥ 6*cs, f02[6*vx, 6*vy, r - 6*cs], p01],
  If[r ≥ 7*cs, f02[7*vx, 7*vy, r - 7*cs], p01],
  If[r ≥ 8*cs, f02[8*vx, 8*vy, r - 8*cs], p01],
  If[r ≥ 9*cs, f02[9*vx, 9*vy, r - 9*cs], p01],
  s2d[r], Background → White],
 {r, 0.1, 10*cs}]
```



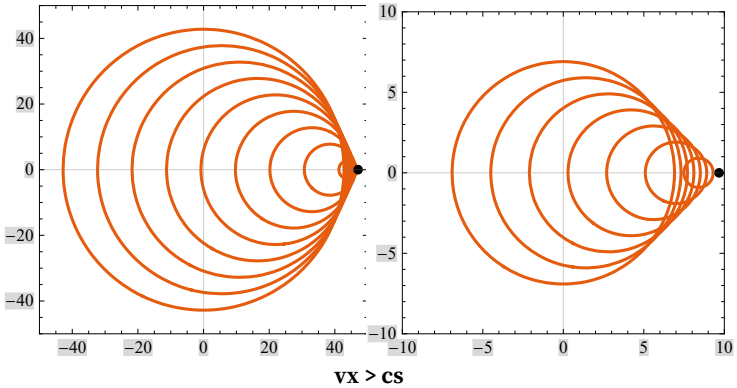
vx < cs

The same sound source is radiating sound waves at a constant frequency. Since the source is moving, the center of each new wave front is now slightly displaced to the right. As a result, the wave-fronts begin to bunch up on the right side (in front of) and spread further apart on the left side (behind) of the source. An observer in front of the source will hear a higher frequency and an observer behind the source will hear a lower frequency.



vx = cs

Now the source is moving at the speed of sound. The wave fronts in front of the source are now all bunched up at the same point. As a result, an observer in front of the source will detect nothing until the source arrives and an observer behind the source will hear a lower frequency.



vx > cs

The sound source has now surpassed the speed of sound. Since the source is moving faster than the sound waves it creates, it actually leads the advancing wave front(mach cone). The sound source will pass by a stationary observer before the observer hears the sound.

B. Three-Dimensional Space

The program produces multiple spheres with different coordinates which shows us movement of the source, variables "vx", "vy" and "vz" determine velocity of the source respectively in x-direction, y-direction and z-direction. Spheres visualize wave crests which Expand at rate of "cs" that represents speed of sound.

```
vx := 5
(*velocity in the x direction*)
vy := 5
(*velocity in the y direction*)
vz := 5
(*velocity in the z direction*)
cs := 5
(*speed of wave / scale*)
f03[Xv_, Yv_, Zv_, r_] :=
 ContourPlot3D[(x - Xv)^2 + (y - Yv)^2 + (z - Zv)^2 == r^2, {x, -10*cs, 10*cs}, {y, -10*cs, 10*cs},
  {z, -10*cs, 10*cs}, PlotRange → {{-10*cs, 10*cs}, {-10*cs, 10*cs}, {-10*cs, 10*cs}},
  Mesh → None, ContourStyle → Opacity[0.1]]
(*waves*)
p02 := Plot3D[0, {x, 0, 0.000001}, {y, 0, 0.000001}]
(*just a point (doesn't matter)*)
s3d[r_] := Graphics3D[{AbsolutePointSize[5], Point[{(vx*r)/cs, (vy*r)/cs, (vz*r)/cs}]}]
(*source*)
Manipulate[Show[
  f03[0*vx, 0*vy, 0*vz, r],
  If[r ≥ cs, f03[1*vx, 1*vy, 1*vz, r - cs], p02],
  If[r ≥ 2*cs, f03[2*vx, 2*vy, 2*vz, r - 2*cs], p02],
  If[r ≥ 3*cs, f03[3*vx, 3*vy, 3*vz, r - 3*cs], p02],
  If[r ≥ 4*cs, f03[4*vx, 4*vy, 4*vz, r - 4*cs], p02],
  If[r ≥ 5*cs, f03[5*vx, 5*vy, 5*vz, r - 5*cs], p02],
  If[r ≥ 6*cs, f03[6*vx, 6*vy, 6*vz, r - 6*cs], p02],
  If[r ≥ 7*cs, f03[7*vx, 7*vy, 7*vz, r - 7*cs], p02],
  If[r ≥ 8*cs, f03[8*vx, 8*vy, 8*vz, r - 8*cs], p02],
  If[r ≥ 9*cs, f03[9*vx, 9*vy, 9*vz, r - 9*cs], p02],
  s3d[r]],
 {r, 0.1, 10*cs}]
```

IV. Graphical Interchange Format

Mathematica can't show these animations correctly therefore in this section we try to pre render the animations and then export them into "GIF" formats, this allows us to have smoother animations.

A. Stationary Source

```
v := 5
(*scale*)
f01[r_] := ContourPlot[x^2 + y^2 == (r)^2, {x, -10*v, 10*v}, {y, -10*v, 10*v},
  PlotTheme → "Scientific", PlotRange → {{-10*v, 10*v}, {-10*v, 10*v}}]
(*waves*)
g := Graphics[{AbsolutePointSize[5], Point[{0, 0}]}]
(*source*)
time := Range[1, 10*v, (10*v)/100]
lst := {}
Do[
 AppendTo[lst, Show[
   f01[r],
   If[r ≥ v, f01[r - v], g],
   If[r ≥ 2*v, f01[r - 2*v], g],
   If[r ≥ 3*v, f01[r - 3*v], g],
   If[r ≥ 4*v, f01[r - 4*v], g],
   If[r ≥ 5*v, f01[r - 5*v], g],
   If[r ≥ 6*v, f01[r - 6*v], g],
   If[r ≥ 7*v, f01[r - 7*v], g],
   If[r ≥ 8*v, f01[r - 8*v], g],
   If[r ≥ 9*v, f01[r - 9*v], g],
   g]]
 , {r, time}]
file := "Doppler Efffect (stationary source).gif"
Export[file, lst];
```

B. Moving Source in 2D

```
vx := 3
(*velocity in the x direction*)
vy := 5
(*velocity in the y direction*)
cs := 5
(*speed of wave / scale*)
f02[Xv_, Yv_, r_] :=
 ContourPlot[(x - Xv)^2 + (y - Yv)^2 == (r)^2, {x, -10*cs, 10*cs}, {y, -10*cs, 10*cs},
  PlotTheme → "Scientific", PlotRange → {{-10*cs, 10*cs}, {-10*cs, 10*cs}}]
(*waves*)
p01 := Plot[0, {x, 0, 0.0001}]
(*just a point (doesn't matter)*)
s2d[r_] := Graphics[{AbsolutePointSize[5], Point[{vx*r/cs, vy*r/cs}]}]
(*source*)
time := Range[1, 10*cs, 10*cs/100]
lst := {}
Do[
 AppendTo[lst, Show[
   f02[0*vx, 0*vy, r],
   If[r ≥ cs, f02[1*vx, 1*vy, r - cs], p01],
   If[r ≥ 2*cs, f02[2*vx, 2*vy, r - 2*cs], p01],
   If[r ≥ 3*cs, f02[3*vx, 3*vy, r - 3*cs], p01],
   If[r ≥ 4*cs, f02[4*vx, 4*vy, r - 4*cs], p01],
   If[r ≥ 5*cs, f02[5*vx, 5*vy, r - 5*cs], p01],
   If[r ≥ 6*cs, f02[6*vx, 6*vy, r - 6*cs], p01],
   If[r ≥ 7*cs, f02[7*vx, 7*vy, r - 7*cs], p01],
   If[r ≥ 8*cs, f02[8*vx, 8*vy, r - 8*cs], p01],
   If[r ≥ 9*cs, f02[9*vx, 9*vy, r - 9*cs], p01],
   s2d[r]]]
 , {r, time}]
file := "Doppler Effect (moving source 2D).gif"
Export[file, lst];
```

C. Moving Source in 3D

```
vx := 5
(*velocity in the x direction*)
vy := 5
(*velocity in the y direction*)
vz := 5
(*velocity in the z direction*)
cs := 5
(*speed of wave / scale*)
f03[Xv_, Yv_, Zv_, r_] :=
 ContourPlot3D[(x - Xv)^2 + (y - Yv)^2 + (z - Zv)^2 == r^2, {x, -10*cs, 10*cs}, {y, -10*cs, 10*cs},
  {z, -10*cs, 10*cs}, PlotRange -> {{-10*cs, 10*cs}, {-10*cs, 10*cs}, {-10*cs, 10*cs}},
  Mesh -> None, ContourStyle -> Opacity[0.1]]
(*waves*)
p02 := Plot3D[0, {x, 0, 0.000001}, {y, 0, 0.000001}]
(*just a point (doesn't matter)*)
s3d[r_] := Graphics3D[{AbsolutePointSize[5], Point[{vx*r/cs, vy*r/cs, vz*r/cs}]}]
(*source*)
time := Range[1, 10*cs, 10*cs/100]
lst := {}
Do[
 AppendTo[lst, Show[f03[0*vx, 0*vy, 0*vz, r],
   If[r >= cs, f03[1*vx, 1*vy, 1*vz, r - cs], p02],
   If[r >= 2*cs, f03[2*vx, 2*vy, 2*vz, r - 2*cs], p02],
   If[r >= 3*cs, f03[3*vx, 3*vy, 3*vz, r - 3*cs], p02],
   If[r >= 4*cs, f03[4*vx, 4*vy, 4*vz, r - 4*cs], p02],
   If[r >= 5*cs, f03[5*vx, 5*vy, 5*vz, r - 5*cs], p02],
   If[r >= 6*cs, f03[6*vx, 6*vy, 6*vz, r - 6*cs], p02],
   If[r >= 7*cs, f03[7*vx, 7*vy, 7*vz, r - 7*cs], p02],
   If[r >= 8*cs, f03[8*vx, 8*vy, 8*vz, r - 8*cs], p02],
   If[r >= 9*cs, f03[9*vx, 9*vy, 9*vz, r - 9*cs], p02],
   s3d[r],
   ViewPoint -> Front (*point of view in 3D*)]]
 , {r, time}]
file := "Doppler Effect (moving source 3D).gif"
Export[file, lst];
```

V. References

1. Possel, Markus (2017). “Waves, motion and frequency: the Doppler effect”. Einstein Online, Vol. 5. Max Planck Institute for Gravitational Physics, Potsdam, Germany. Archived from the original on September 14, 2017. Retrieved September 4, 2017.

2. Doppler effect - Wikipedia
Viewed at: https://en.wikipedia.org/wiki/Doppler_effect

3. “Doppler and the Doppler effect”, E. N. da C. Andrade, Endeavour Vol. XVIII No. 69, January 1959 (published by ICI London).

4. Halliday, David; Robert; Walker, Jearl (2011). Fundamentals of physics (9th ed.). John Wiley & Sones. ISBN 978-470-56158-4

VI . AUTHOR CONTRIBUTIONS

A.T.J. realized the core idea for Doppler effect simulation and contributed expertise towards the theoretical part of the program. Y.S. expanded the
main idea and contributed expertise towards programming. Both authors participated in revising the final manuscript and analyzing the results.