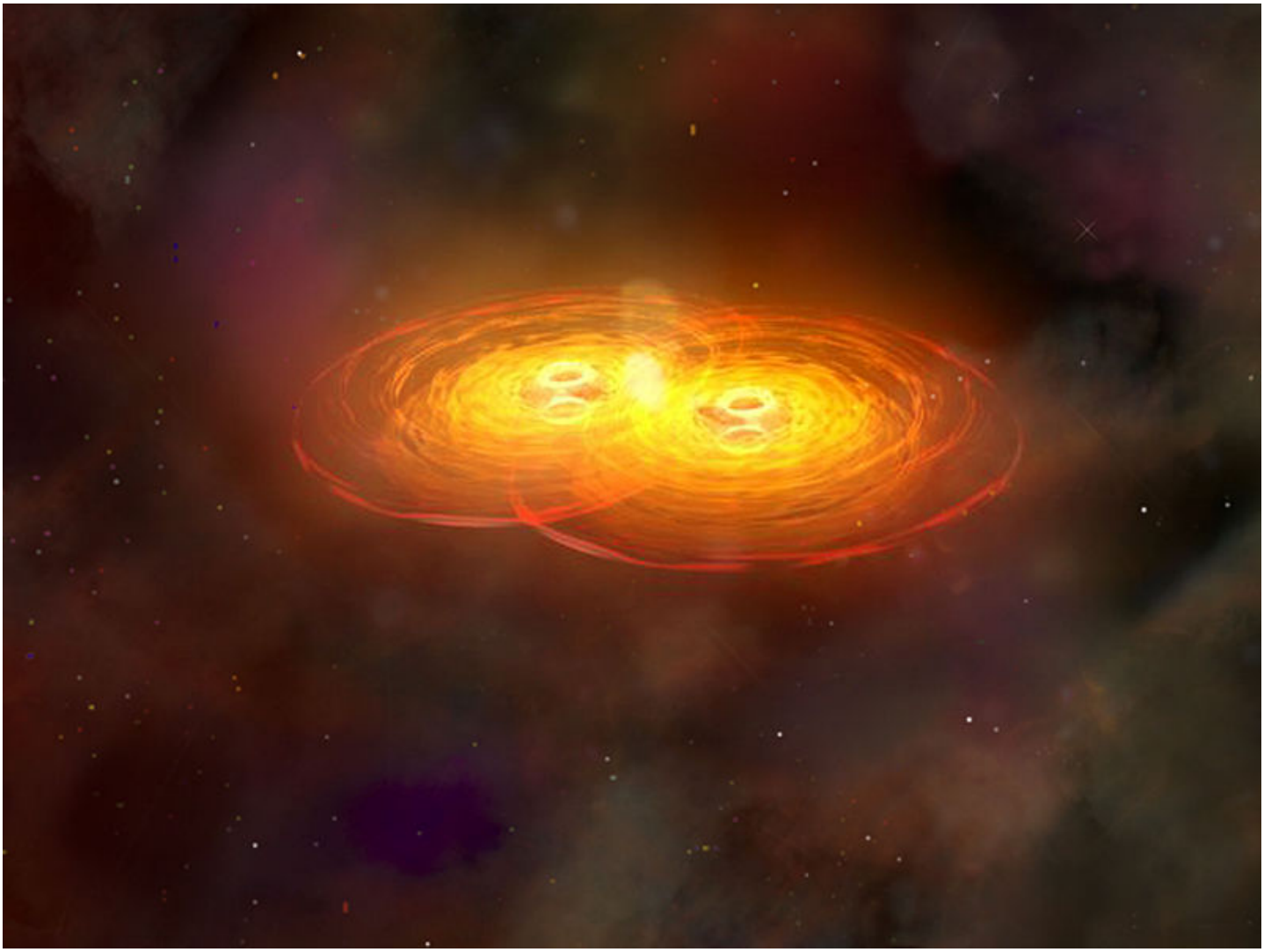
Black Holes collisions - LaCoNGA (June 29, 2021)



Credit: Wikipedia

This is the black holes collision part of the course (2021). by: J.M, Ramírez

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv
)
import os
```

In [2]:
```
import matplotlib.pyplot as plt
from matplotlib import transforms
from mpl_toolkits.axisartist.axislines import AxesZero
import subprocess
import sys
import re

import warnings
import arviz as az
import pymc3 as pm
warnings.simplefilter(action="ignore", category=FutureWarning)
###
%config InlineBackend.figure_format = 'retina'
az.style.use("arviz-darkgrid")
print(f"Running on ArviZ v{az.__version__}")

###
import io
from PIL import Image
import matplotlib as mpl
```

```
Running on ArviZ v0.11.2
```

In [3]:
```
mpl.__version__
```

Out[3]:
```
'3.4.1'
```

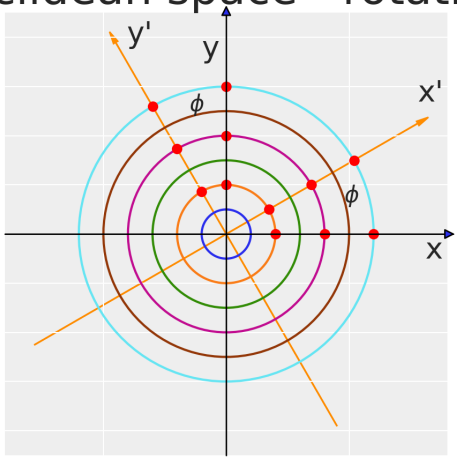
In [4]:

```
def doall(EqTitle,OneEq,locEq=0.2,FC="white"):
    # Colors used in Matplotlib online documentation.
    mpl_blue_rvb = (191. / 255., 209. / 256., 212. / 255.)
    mpl_orange_rvb = (202. / 255., 121. / 256., 0. / 255.)
    mpl_grey_rvb = (51. / 255., 51. / 255., 51. / 255.)

    # Creating figure and axis.
    plt.figure(figsize=(7, 0.8))
    plt.axes([0.01, 0.01, 0.98, 0.90], facecolor=FC, frameon=True)
    plt.gca().set_xlim(0., 1.)
    plt.gca().set_ylim(0., 1.)
    plt.gca().set_title(EqTitle,
                        color=mpl_grey_rvb, fontsize=24, weight='bol
d')
    plt.gca().set_xticklabels("", visible=False)
    plt.gca().set_yticklabels("", visible=False)

    # Plotting formula
    plt.annotate(OneEq,
                 xy=(locEq,0.3),
                 color=mpl_orange_rvb, ha='center', fontsize=35)
    plt.show()
```

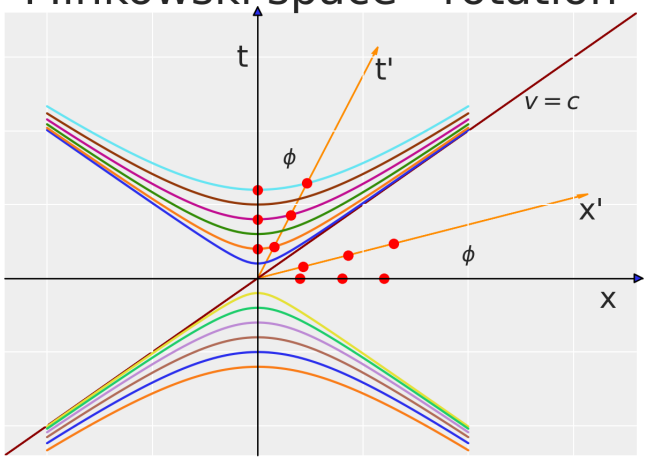
Relativity and Geometry

Show hidden code

Euclidean space - rotation



Minkowski space - rotation



Special relativity is encoded in the geometry of Minkowski space 📌

Relativity and Geometry

In [6]:

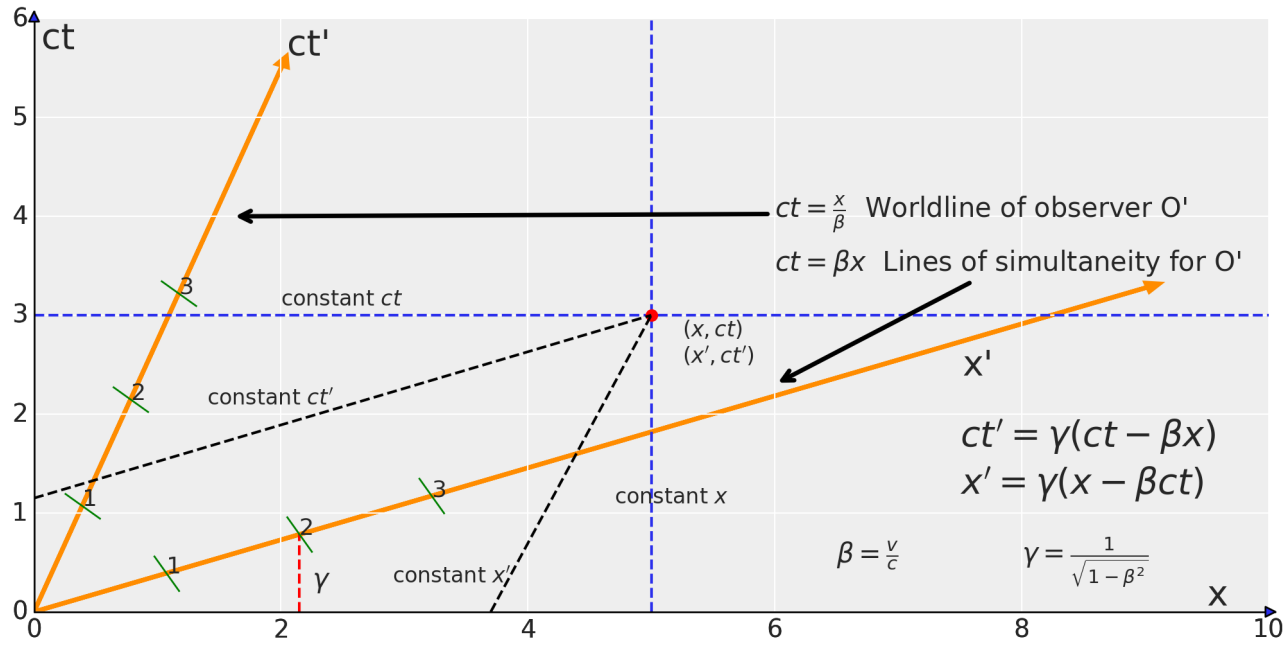
```
fig = plt.figure(figsize = (10, 13))
ax = fig.add_subplot(1,1,1, axes_class=AxesZero, aspect=0.8)


## Some Transformation
base = plt.gca().transData

for direction in ["xzero", "yzero"]:
    ax.axis[direction].set_axisline_style("-|>")
```

```
    ax.axis[direction].set_visible(True)

## Square of the plot
xl,xh = 0., 10.
yl,yh = xl,6.
plt.xlim(xl,xh)
plt.ylim(yl,yh)

#########################################################################
##################
## Rotated y axe
angle_t = -20
rot_t = transforms.Affine2D().rotate_deg(angle_t)
x_tail = 0.
y_tail = 0.
y_head = yh*0.98
x_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.09, lw=2.5,
         head_length=0.1, fc='darkorange', ec='darkorange', transfor
m= rot_t + base) #

#########################################################################
##################
## Some points (y axe)
xhAb=np.array([1, 2, 3])
## SR theory
t1=(1/np.sqrt(1-np.tan(angle_t*np.pi/(180))**2))*xhAb
x1=np.sqrt(-xhAb**2 + t1**2)
##
symbol=["1","2","3"]
for i,l in enumerate(symbol):
    #plt.plot(x1[i],t1[i],'go', marker=r"$ {} $".format(l), markersize
=12)
    tm = mpl.markers.MarkerStyle(marker='|')
    tm._transform = tm.get_transform().rotate_deg(55)
    plt.plot(x1[i],t1[i],'g', marker=tm, markersize=25)
```

```
    plt.annotate(str(xhAb[i]),(x1[i],t1[i]),  fontsize=13)
#########################################################################
##################

###
#########################################################################
##################
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.
x_head = yh*1.6
y_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.09, lw=2.5,
         head_length=0.1, fc='darkorange', ec='darkorange', transfor
m= rot_x + base) #

#########################################################################
##################
## Some points (x axe)
## SR theory
t1=(1/np.sqrt(1-np.tan(angle_t*np.pi/(180))**2))*xhAb
x1=np.sqrt(-xhAb**2 + t1**2)
##
for i,l in enumerate(symbol):
    #plt.plot(x1[i],t1[i],'go', marker=r"$ {} $".format(l), markersize
=12)
    tm = mpl.markers.MarkerStyle(marker='|')
    tm._transform = tm.get_transform().rotate_deg(35)
    plt.plot(t1[i],x1[i],'g', marker=tm, markersize=25)
    plt.annotate(str(xhAb[i]),(t1[i],x1[i]),  fontsize=13)
plt.vlines(t1[1],0.,x1[1], linestyle='--', color='r')
plt.annotate('$\gamma$',(t1[1]*1.05,0.25),  fontsize=15)
#########################################################################
##################
```

```
## Anotate axes
plt.annotate("x",(0.95*xh, 0.07),  fontsize=20)
plt.annotate("ct",(0.06, 0.95*yh),  fontsize=20)
# Rotated ct' & x'
plt.annotate('x\'',(0.8*xh*np.cos(angle_x*np.pi/(180)), 0.7*xh*np.si
n(angle_x*np.pi/(180))),  fontsize=20)
plt.annotate('ct\'',(yh*np.cos(np.pi/2-angle_x*np.pi/(180)), yh*np.s
in(np.pi/2-angle_x*np.pi/(180))),  fontsize=20)


## H&V Lines
x1=5
y1=3
plt.hlines(y1,0.,11., linestyle='--')
plt.annotate('constant $ct$',(2.0,3.1),  fontsize=12)
plt.vlines(x1,0.,11., linestyle='--')
plt.annotate('constant $x$',(4.7,1.1),  fontsize=12)
## point intersec
plt.plot([x1],[y1],'ro')
plt.annotate('($x, ct$)',(x1*1.05,y1*.93),  fontsize=12)
plt.annotate('($x\', ct\'$)',(x1*1.05,y1*.84),  fontsize=12)


## Ploting two lines of simultaneity --
plt.plot([0.,x1],[1.15,y1],'k',linestyle='--')
plt.annotate('constant $ct\'$',(1.4,2.1),  fontsize=12)
plt.plot([3.7,x1],[0.,y1],'k',linestyle='--')
plt.annotate('constant $x\'$',(2.9,0.3),  fontsize=12)

#########################################################################
##################
#########################################################################
##################
## Notes
ax.annotate(r'$ct = \frac{x}{\beta}$  Worldline of observer'+" O\'",
            xy=(1.6,4),#(0.3*yh*np.cos(np.pi/2-angle_x*np.pi/(180)),
0.8*yh*np.sin(np.pi/2-angle_x*np.pi/(180))),
```

```
            xytext=(6, 4),   fontsize=15,
            arrowprops=dict(arrowstyle="->",lw=2.5))
##
ax.annotate(r'$ct = \beta x$  Lines of simultaneity for'+" O\'",
            xy=(6,2.3),
            xytext=(6, 3.45),   fontsize=15,
            arrowprops=dict(arrowstyle="->",lw=2.5))
##
ax.annotate("$ct\'$"+r"$= \gamma(ct-\beta x)$",
            (7.5, 2.1*0.8),   fontsize=20,
            )
##
ax.annotate("$x\'$"+r"$ = \gamma(x-\beta ct)$",
            (7.5, 1.5*0.8),   fontsize=20,
            )
##
ax.annotate(r"$\beta = \frac{v}{c}$",
            (6.5, .5),   fontsize=15,
            )
##
ax.annotate(r"$\gamma = \frac{1}{\sqrt{1-\beta^2}}$",
            (8., .5),   fontsize=15,
            )
#########################################################################
##################
#########################################################################
##################


plt.show()
```



Invariant interval

In [7]:

```
## Here we write the Eq.
mpl_blue_rvb = (191. / 255., 209. / 256., 212. / 255.)
myEq1={1:
    r"$ds^2 = "
    r"\eta_{\mu \nu}dx^{\mu}dx^{\nu} = "
    r"-c^2t^2 + dx^2$",
    2:
    r"$ct'= \gamma(ct-\beta x)$",
    3:
    r"$x'= \gamma(x-\beta ct)$",
    4:
    r"$\eta_{\mu \nu} ~~\to~~ \binom{-1~~~0}{~~0~~~1}$",
    5:
    r"$ds'^2 = "
    r"\eta_{\mu \nu}dx'^{\mu}dx'^{\nu} = "
    r"-c^2t'^2 + dx'^2 ="
    r"ds^2$",
}
myTitle='Invariant interval'
## Here we plot it
doall(myTitle,myEq1[1])
## metric
doall('',myEq1[4],0.2)
doall('',myEq1[2])
doall('',myEq1[3])
doall('',myEq1[5])
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/pylabtools.py:13
2: UserWarning: There are no gridspecs with layoutgrids. Possibly di
d not call parent GridSpec with the "figure" keyword
  fig.canvas.print_figure(bytes_io, **kw)
```

Invariant interval

$$ds^2 = \eta_{\mu\nu} dx^\mu dx^\nu = -c^2t^2 + dx^2$$

$$\eta_{\mu\nu} \rightarrow \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$ct' = \gamma(ct - \beta x)$$

$$x' = \gamma(x - \beta ct)$$

$$ds'^2 = \eta_{\mu\nu} dx'^\mu dx'^\nu = -c^2t'^2 + dx'^2 = ds^2$$

Time Dilation

In [8]:

```
fig = plt.figure(figsize = (20, 17))
ax = fig.add_subplot(4,1,1, axes_class=AxesZero, aspect=0.8)
```

```
## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)


# Move left y-axis and bottim x-axis to centre, passing through (0,0)
ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)


#########################################################################
##################
#########################################################################
##################
## Notes
##
ax.annotate("Proper time elapsed        $\Delta t' = T_0$",
            (0.1,0.9),    fontsize=17,
            )
##
ax.annotate("Apparent time elapsed    $\Delta t = T$",
            (0.1,0.8),    fontsize=17,
            )
##
ax.annotate("$\Delta s^2_{AB} = - (c\Delta t')^2 = - (c\Delta t)^2
+ \Delta x^2$",
            (0.1,0.65),    fontsize=17,
            )
##
ax.annotate("But,   $\Delta x=v \Delta t$",
            (0.1,0.55),    fontsize=17,
            )
##
ax.annotate(r"$\Rightarrow \Delta t'^2= \Delta t^2 (1-\beta^2)=\frac
{\Delta t^2}{\gamma^2}$",
            (0.1,0.4),    fontsize=20,
            )
```

```
##
ax.annotate(r"$\Rightarrow T=\gamma T_0$",
            (0.1,0.3),   fontsize=20,
            )

######  End Notes ...
#########################################################################
##################
```

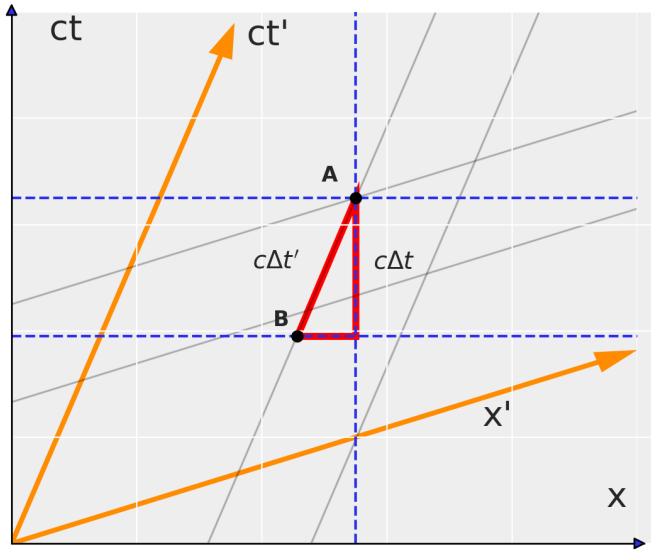
```
#########################################################################
#################################################
#########################################################################
################################################
### Right part for annnotations ( 2 )

ax = fig.add_subplot(4,2,1, axes_class=AxesZero, aspect=0.85)
## Some Transformation
base = plt.gca().transData

for direction in ["xzero", "yzero"]:
    ax.axis[direction].set_axisline_style("-|>")
    ax.axis[direction].set_visible(True)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)

ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)
#########################################################################
##################
## Rotated y axe
angle_t = -20
rot_t = transforms.Affine2D().rotate_deg(angle_t)
x_tail = 0.
```

```
y_tail = 0.
y_head = yh*0.98
x_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.03, lw=2.5,
         head_length=0.05, fc='darkorange', ec='darkorange', transfo
rm= rot_t + base) #
#########################################################################
##################
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.
x_head = yh*1.
y_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.03, lw=2.5,
         head_length=0.05, fc='darkorange', ec='darkorange', transfo
rm= rot_x + base) #
#########################################################################
##################
#########################################################################
##################

## Anotate axes
plt.annotate("x",(0.95*xh, 0.07),  fontsize=20)
plt.annotate("ct",(0.06, 0.95*yh),  fontsize=20)
# Rotated ct' & x'
plt.annotate('x\'',(0.8*xh*np.cos(angle_x*np.pi/(180)), 0.65*xh*np.s
in(angle_x*np.pi/(180))),  fontsize=20)
plt.annotate('ct\'',(1.1*yh*np.cos(np.pi/2-angle_x*np.pi/(180)), yh*
np.sin(np.pi/2-angle_x*np.pi/(180))),  fontsize=20)
```

```
## H&V Lines
x1=0.55
y1=0.65
plt.hlines(y1,0.,11., linestyle='--')
plt.hlines(y1*0.6,0.,11., linestyle='--')
plt.vlines(x1,0.,11., linestyle='--')
## point intersec
plt.plot([x1],[y1],'ko')
plt.annotate('A',(x1*0.9,y1*1.05),  fontsize=12,weight="bold")
plt.plot([x1*0.83],[(y1*0.6)],'ko')
plt.annotate('B',(x1*0.76,(y1*0.6)*1.05),  fontsize=12,weight="bold"
)
plt.annotate("$c\Delta t'$",(x1*0.7,(y1*0.76)*1.05),  fontsize=14,we
ight="bold")
plt.annotate("$c\Delta t$",(x1*1.05,(y1*0.76)*1.05),  fontsize=14,we
ight="bold")

# a Triangle
pivot_left = (x1*0.83, y1*0.6)
pivot_right = (x1,y1*0.6)
pivot_top = (x1, y1)
points = np.array([pivot_left, pivot_right, pivot_top])
pivot = plt.Polygon(points, closed=True, fill=False, edgecolor='r',
lw=4.)
ax.add_patch(pivot)

#########################################################################
########################
#########################################################################
########################
## Ploting two lines of simultaneity --
#########################################################################
########################
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.423
x_head = yh*1.5
```

```
y_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.25
x_head = yh*1.5
y_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
##########################################################################
########################
## Rotated t axe
angle_x = angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.295
y_tail = 0.
y_head = yh*1.5
x_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
## Rotated t axe
angle_x = angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.45
y_tail = 0.
y_head = yh*1.5
x_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
```

```
        head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
##########################################################################
########################
##########################################################################
########################

plt.show()
```



Proper time elapsed $\Delta t' = T_0$

Apparent time elapsed $\Delta t = T$

$$\Delta s_{AB}^2 = -(c\Delta t')^2 = -(c\Delta t)^2 + \Delta x^2$$

But, $\Delta x = v\Delta t$

$$\Rightarrow \Delta t'^2 = \Delta t^2(1-\beta^2) = \frac{\Delta t^2}{\gamma^2}$$

$$\Rightarrow T = \gamma T_0$$

Length Contraction

In [9]:

```
fig = plt.figure(figsize = (22, 22))
ax = fig.add_subplot(4,1,1, axes_class=AxesZero, aspect=0.6)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)
```

```
# Move left y-axis and bottim x-axis to centre, passing through (0,0)
ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)


#########################################################################
##################
#########################################################################
##################
## Notes
##
ax.annotate("Proper length        $\Delta x' = L_0$",
            (0.1,0.9),    fontsize=17,
            )
##
ax.annotate("Apparent length    $\Delta x = L$",
            (0.1,0.8),    fontsize=17,
            )
##
ax.annotate("$\Delta s^2_{AB} =  \Delta x^2 = - (\Delta x')^2  - (c\
Delta t')^2$",
            (0.1,0.65),    fontsize=17,
            )
##
ax.annotate("$\Delta s^2_{AC} =  \Delta x'^2 = (\Delta x+v\Delta t)^
2 - (c\Delta t)^2 $",
            (0.1,0.55),    fontsize=17,
            )
##
ax.annotate("and...   $\Delta t = \gamma \Delta t'$",
            (0.1,0.45),    fontsize=17,
            )
##
ax.annotate("After a bit of Baldor...",
            (0.1,0.35),    fontsize=17,
            )
##
ax.annotate(r"$\Rightarrow L = \frac{L_0}{\gamma}$",
```

```
            (0.2,0.16),    fontsize=30,
            )

######  End Notes ...
#########################################################################
##################
```

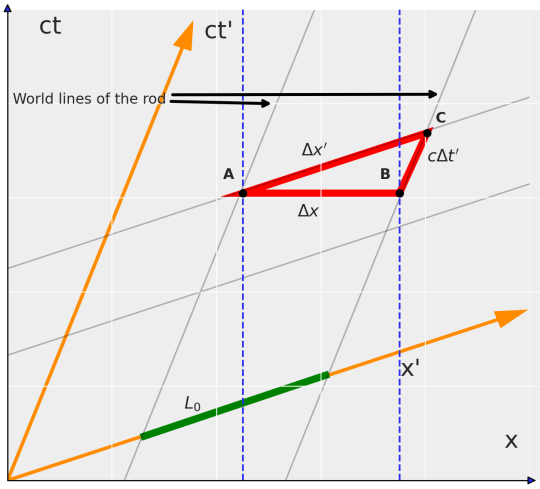
```
#########################################################################
#################################################
#########################################################################
#################################################
### Right part for annnotations ( 2 )

ax = fig.add_subplot(4,3,1, axes_class=AxesZero, aspect=0.9)
## Some Transformation
base = plt.gca().transData

for direction in ["xzero", "yzero"]:
    ax.axis[direction].set_axisline_style("-|>")
    ax.axis[direction].set_visible(True)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)

ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)
#########################################################################
##################
## Rotated y axe
angle_t = -20
rot_t = transforms.Affine2D().rotate_deg(angle_t)
x_tail = 0.
y_tail = 0.
y_head = yh*0.98
```

```
x_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.03, lw=2.5,
         head_length=0.05, fc='darkorange', ec='darkorange', transfo
rm= rot_t + base) #
########################################################################
##################
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.
x_head = yh*1.
y_head = 0.
dx = x_head - x_tail
dy = y_head - y_tail
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.03, lw=2.5,
         head_length=0.05, fc='darkorange', ec='darkorange', transfo
rm= rot_x + base) #
########################################################################
##################
########################################################################
##################

## Anotate axes
plt.annotate("x",(0.95*xh, 0.07),  fontsize=20)
plt.annotate("ct",(0.06, 0.95*yh),  fontsize=20)
# Rotated ct' & x'
plt.annotate('x\'',(0.8*xh*np.cos(angle_x*np.pi/(180)), 0.65*xh*np.s
in(angle_x*np.pi/(180))),  fontsize=20)
plt.annotate('ct\'',(1.1*yh*np.cos(np.pi/2-angle_x*np.pi/(180)), yh*
np.sin(np.pi/2-angle_x*np.pi/(180))),  fontsize=20)


## H&V Lines
x1=0.75
```

```
y1=0.61
plt.vlines(x1,0.,11., linestyle='--')
plt.vlines(x1*0.6,0.,11., linestyle='--')

## point intersec
plt.plot([x1*0.6],[y1],'ko')
plt.annotate('A',(x1*0.55,y1*1.05),  fontsize=12,weight="bold")
plt.plot([x1],[(y1)],'ko')
plt.annotate('B',(x1*0.95,y1*1.05),  fontsize=12,weight="bold")
plt.plot([x1*1.07],[y1*1.21],'ko')
plt.annotate('C',(x1*1.07*1.02,y1*1.21*1.03),  fontsize=12,weight="b
old")
plt.annotate("$\Delta x'$",(x1*0.75,(y1*1.13)),  fontsize=14,weight=
"bold")
plt.annotate("$c\Delta t'$",(x1*1.07,(y1*1.11)),  fontsize=14,weight
="bold")
plt.annotate("$\Delta x$",(x1*0.74,(y1*0.92)),  fontsize=14,weight="
bold")

# a Triangle
pivot_left = (x1*0.6,y1)
pivot_right = (x1,y1)
pivot_top = (x1*1.07,y1*1.21)
points = np.array([pivot_left, pivot_right, pivot_top])
pivot = plt.Polygon(points, closed=True, fill=False, edgecolor='r',
lw=6.)
ax.add_patch(pivot)
# a Segment
pivot_left = (x1*0.34,y1*0.15)
pivot_right = (x1*0.82,y1*0.37)
points = np.array([pivot_left, pivot_right])
pivot = plt.Polygon(points, closed=False, fill=False, edgecolor='g',
lw=6.)
ax.add_patch(pivot)
plt.annotate("$L_0$",(x1*0.45,y1*0.25),  fontsize=14,weight="bold")


## Notes
ax.annotate('World lines of the rod',
```

```
            xy=(0.51,0.8),
            xytext=(0.01, 0.8),   fontsize=12,
            arrowprops=dict(arrowstyle="->",lw=2.5))
## Notes
ax.annotate('                                    ',
            xy=(0.83,0.82),
            xytext=(0.01, 0.81),   fontsize=12,
            arrowprops=dict(arrowstyle="->",lw=2.5))


##########################################################################
########################
##########################################################################
########################
## Ploting two lines of simultaneity --
##########################################################################
########################
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.423
x_head = yh*1.5
y_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
## Rotated x axe
angle_x = -angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.
y_tail = 0.25
x_head = yh*1.5
y_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
```

```
#########################################################################
########################
## Rotated t axe
angle_x = angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.21
y_tail = 0.
y_head = yh*1.5
x_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
## Rotated t axe
angle_x = angle_t
rot_x = transforms.Affine2D().rotate_deg(angle_x)
x_tail = 0.5
y_tail = 0.
y_head = yh*1.5
x_head = 0.
# t' axis
ax.arrow(x_tail, y_tail, x_head, y_head, head_width=0.0, lw=0.9, alp
ha=0.2,
         head_length=0.0, fc='k', ec='k', transform= rot_x + base) #
#########################################################################
########################
#########################################################################
########################

plt.show()
```



Proper length $\Delta x' = L_0$

Apparent length $\Delta x = L$

$\Delta s^2_{AB} = \Delta x^2 = -(\Delta x')^2 - (c\Delta t')^2$

$\Delta s^2_{AC} = \Delta x'^2 = (\Delta x + v\Delta t)^2 - (c\Delta t)^2$

and... $\Delta t = \gamma \Delta t'$

After a bit of Baldor...

$$\Rightarrow L = \frac{L_0}{\gamma}$$

Equivalence Principle (Newtonian)

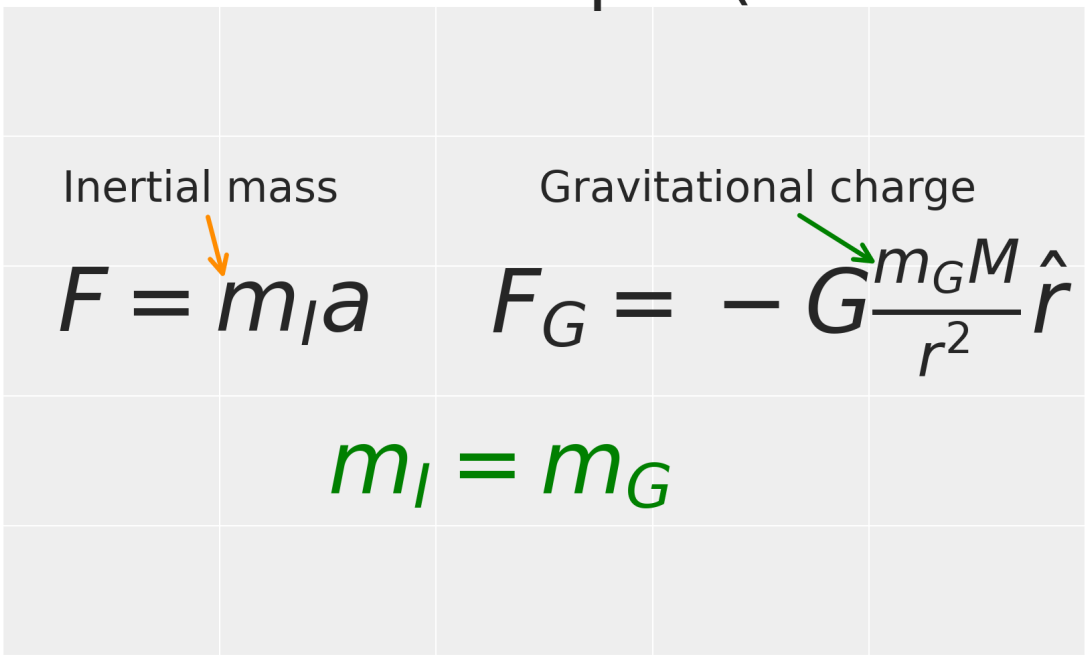
In [10]:

```
fig = plt.figure(figsize = (20, 20))
ax = fig.add_subplot(4,1,1, axes_class=AxesZero, aspect=0.6)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)


# Move left y-axis and bottim x-axis to centre, passing through (0,0)
ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)


##########################################################################
##################
##########################################################################
```

```
##################

## Notes
y1=0.5
x1=0.05
ax.annotate('Inertial mass',
            xy=(x1*4.1,y1*1.15),
            xytext=(x1*1.1, y1*1.4),   fontsize=23,
            arrowprops=dict(arrowstyle="->",lw=2.5, color='darkorang
e'))

ax.annotate("$F=m_I a$",
            (x1,y1),   fontsize=47,
            )
## Grav charge
y1=0.5
x1=0.45
ax.annotate('Gravitational charge',
            xy=(x1*1.8,y1*1.2),
            xytext=(x1*1.1, y1*1.4),   fontsize=23,
            arrowprops=dict(arrowstyle="->",lw=2.5, color='g'))

ax.annotate(r"$F_G=-G \frac{m_G M}{r^2} \hat{r}$",
            (x1,y1),   fontsize=47,
            )
##
y1=0.25
x1=0.3
ax.annotate(r"$m_I = m_G $",
            (x1,y1),   fontsize=47, color='g'
            )



######  End Notes ...
#########################################################################
##################
plt.title("Equivalence Principle (Newtonian)",  fontsize=40)
```

```
plt.show()
```

Equivalence Principle (Newtonian)



📌 "It was most unsatisfactory to me that, although the relation between inertia and energy is so beautifully derived [in Special Relativity], there is no relation between inertia and weight. I suspected that this relationship was inexplicable by means of Special Relativity" Albert Einstein, 1922

Pseudo forces in a non-inertial frames (Newtonian)

In [11]:

```
fig = plt.figure(figsize = (20, 20))
ax = fig.add_subplot(4,1,1, axes_class=AxesZero, aspect=0.5)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)


# Move left y-axis and bottim x-axis to centre, passing through (0,0)
ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)


######################################################################
##################
######################################################################
##################
## Notes
y1=0.49
x1=0.05
ax.annotate('Coriolis',
            xy=(x1+0.35,y1*1.2),
            xytext=(x1+0.35, y1*1.4),   fontsize=15,
            arrowprops=dict(arrowstyle="->",lw=2.5, color='darkorang
e'))
ax.annotate('Uniform acceleration',
            xy=(x1*4.4,y1*1.15),
            xytext=(x1*1.1, y1*1.4),   fontsize=15,
            arrowprops=dict(arrowstyle="->",lw=2.5, color='r'))
ax.annotate('Centripital',
            xy=(x1+0.7,y1*1.2),
            xytext=(x1+0.7, y1*1.4),   fontsize=15,
```

```
            arrowprops=dict(arrowstyle="->",lw=2.5, color='blue'))

ax.annotate(r"$\ddot{x}_{NI}=-a - [2 \omega \times \dot{x}] - [\ome
ga \times (\omega \times x)]$",
            (x1,y1),   fontsize=33,
            )
########
y1=0.84
x1=0.05
ax.annotate("The pseudo forces experienced  in a non-inertial frame
\n \
are automatically proportional to the inertial mass",
            (x1,y1),   fontsize=19, color='g'
            )

########
y1=0.1
x1=0.05
ax.annotate("We readily accept that Coriolis and Centripital forces
are due to \n \
us being in a non-internal frame tied to the surface of the Earth. \
n \
If we also accept that inertial observers should be in free-fall, \n
\
then gravity is also a pseudo force!",
            (x1,y1),   fontsize=19, color='k'
            )


######  End Notes ...
##########################################################################
##################
plt.title("Pseudo forces in a non-inertial frames (Newtonian)",  fon
tsize=28)



plt.show()
```

Pseudo forces in a non-inertial frames (Newtonian)

The pseudo forces experienced in a non-inertial frame are automatically proportional to the inertial mass

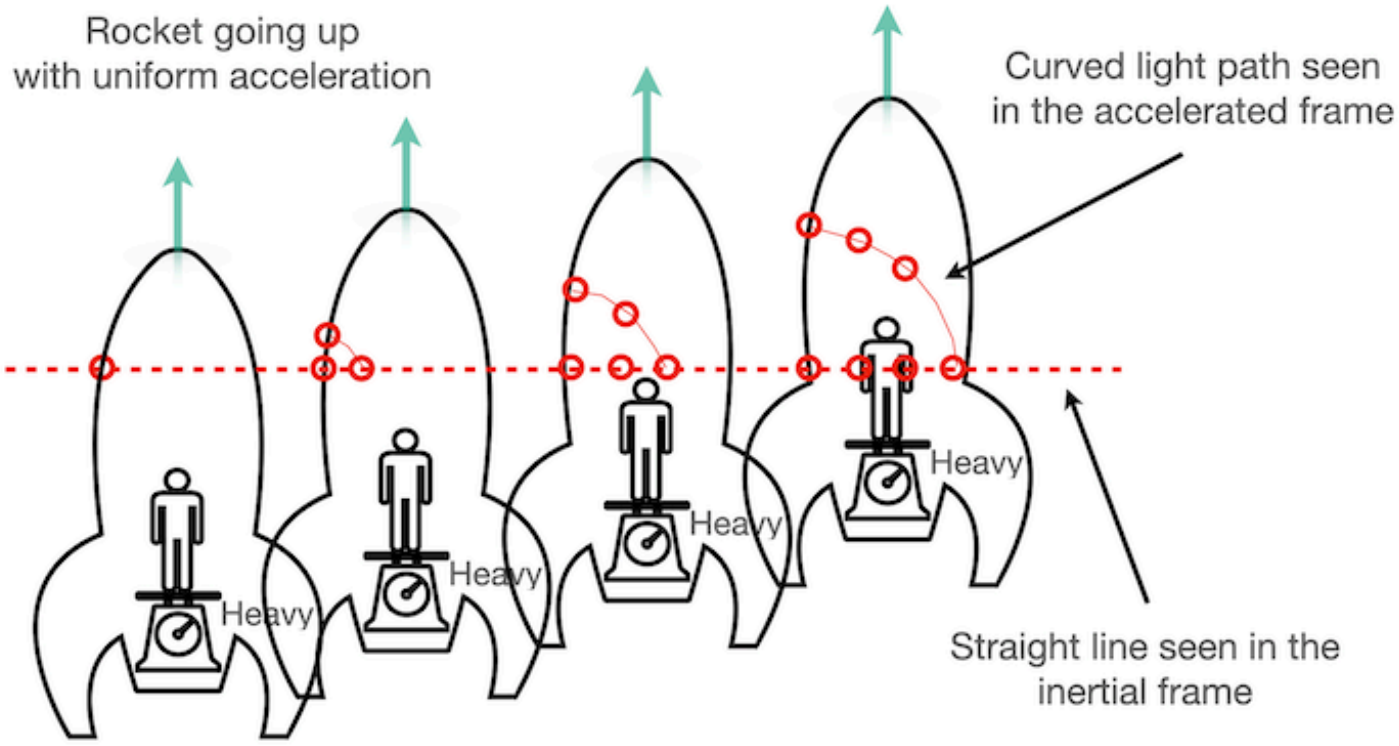
Uniform acceleration Coriolis Centripital

$$\ddot{x}_{NI} = -a - [2\omega \times \dot{x}] - [\omega \times (\omega \times x)]$$

We readily accept that Coriolis and Centripital forces are due to us being in a non-internal frame tied to the surface of the Earth. If we also accept that inertial observers should be in free-fall, then gravity is also a pseudo force!

📌 This is exactly what Einstein reasoned: "I was sitting in a chair in the patent office in Bern when all of a sudden a thought occurred to me: 'If a person falls freely he will not feel his own weight'. I was startled. This simple thought made a deep impression upon me. It impelled me towards a theory of gravitation"

Gravity as Geometry



📌 Lab session #1

In [12]:

```
%matplotlib inline

# General imports
from itertools import product
import matplotlib
import numba
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Math

# Basic imports and functions
from sympy import latex, symbols, sin, cos, pi, simplify, lambdify,
Matrix
from scipy.integrate import solve_ivp

from sympy.diffgeom import (
    Manifold,
    Patch,
    CoordSystem,
    metric_to_Christoffel_2nd,
    TensorProduct as TP
)

def lprint(v):
    display(Math(latex(v)))
```

In [13]:

```
# Create a manifold.
M = Manifold('M', 4)

# Create a patch.
patch = Patch('P', M)

# Basic symbols
c, acc = symbols('c acc')

# Coordinate system
cartes_coord = CoordSystem('cartesian', patch, ['t', 'x', 'y', 'z'])

# Get the coordinate functions
t, x, y, z = cartes_coord.coord_functions()

# Get the base one forms.
dt, dx, dy, dz = cartes_coord.base_oneforms()

# Auxiliar terms for the metric.
dt_2 = TP(dt, dt)
dx_2 = TP(dx, dx)
dy_2 = TP(dy, dy)
dz_2 = TP(dz, dz)
factor = (1 + 2*acc*z / c**2)

# Build the metric
metric = -factor * c ** 2 * dt_2 + dx_2 + dy_2 + dz_2

# Get the Christoffel symbols of the second kind.
christoffel = metric_to_Christoffel_2nd(metric)
# Let's print this in an elegant way ;)
for i, j, k in product(range(4), range(4), range(4)):
    if christoffel[i, j, k] != 0:
        display(Math(f'\Gamma^{i}_{{{j},{k}}} = ' + latex(christoff
el[i, j, k])))
```

```
/opt/conda/lib/python3.7/site-packages/sympy/diffgeom/diffgeom.py:27
4: SymPyDeprecationWarning:

Passing str as coordinate symbol's name has been deprecated since
SymPy 1.7. Use Symbol which contains the name and assumption for
coordinate symbol instead. See
https://github.com/sympy/sympy/issues/19321 for more info.

  deprecated_since_version="1.7"
```

$$\Gamma^{0}_{0,3} = \frac{acc}{2accz + c^2}$$

$$\Gamma^{0}_{3,0} = \frac{acc}{2accz + c^2}$$

$$\Gamma^{3}_{0,0} = acc$$

In [14]:

```
## Here we write the Eq.
myEq1={1: "\"For this bending of light:\"",
       2:
    r"$ds^2 = -c^2 \left(1+\frac{2a_z z}{c^2}  \right) dt^2 + dx^2 +
dy^2 +dz^2$",
       3:
    r"$\frac{d^2x^{\mu}}{ds^2} + \Gamma^{\mu}_{\alpha\beta}  \frac{d
x^{\alpha}}{ds} \frac{dx^{\beta}}{ds} = 0 $"
}
#
myTitle='Geodesic Equation'
## Here we plot it
doall(myTitle,myEq1[1])
doall("",myEq1[2])
doall("",myEq1[3])
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/pylabtools.py:13
2: UserWarning: There are no gridspecs with layoutgrids. Possibly di
d not call parent GridSpec with the "figure" keyword
  fig.canvas.print_figure(bytes_io, **kw)
```

Geodesic Equation

"For this bending of light:"

$$ds^2 = -c^2\left(1+\frac{2a_z z}{c^2}\right)dt^2 + dx^2 + dy^2 + dz^2$$

$$\frac{d^2x^\mu}{ds^2} + \Gamma^\mu_{\alpha\beta}\frac{dx^\alpha}{ds}\frac{dx^\beta}{ds} = 0$$

In [15]:

```
g_func = lambdify((c, acc, z), christoffel, modules='numpy')

## Specify c and r_s
def F(t, y):
    u = np.array(y[0:4])
    v = np.array(y[4:8])

    chris = g_func(1, 1, u[3])

    du = v
    dv = -np.dot(np.dot(chris, v), v)

    return np.concatenate((du, dv))
```

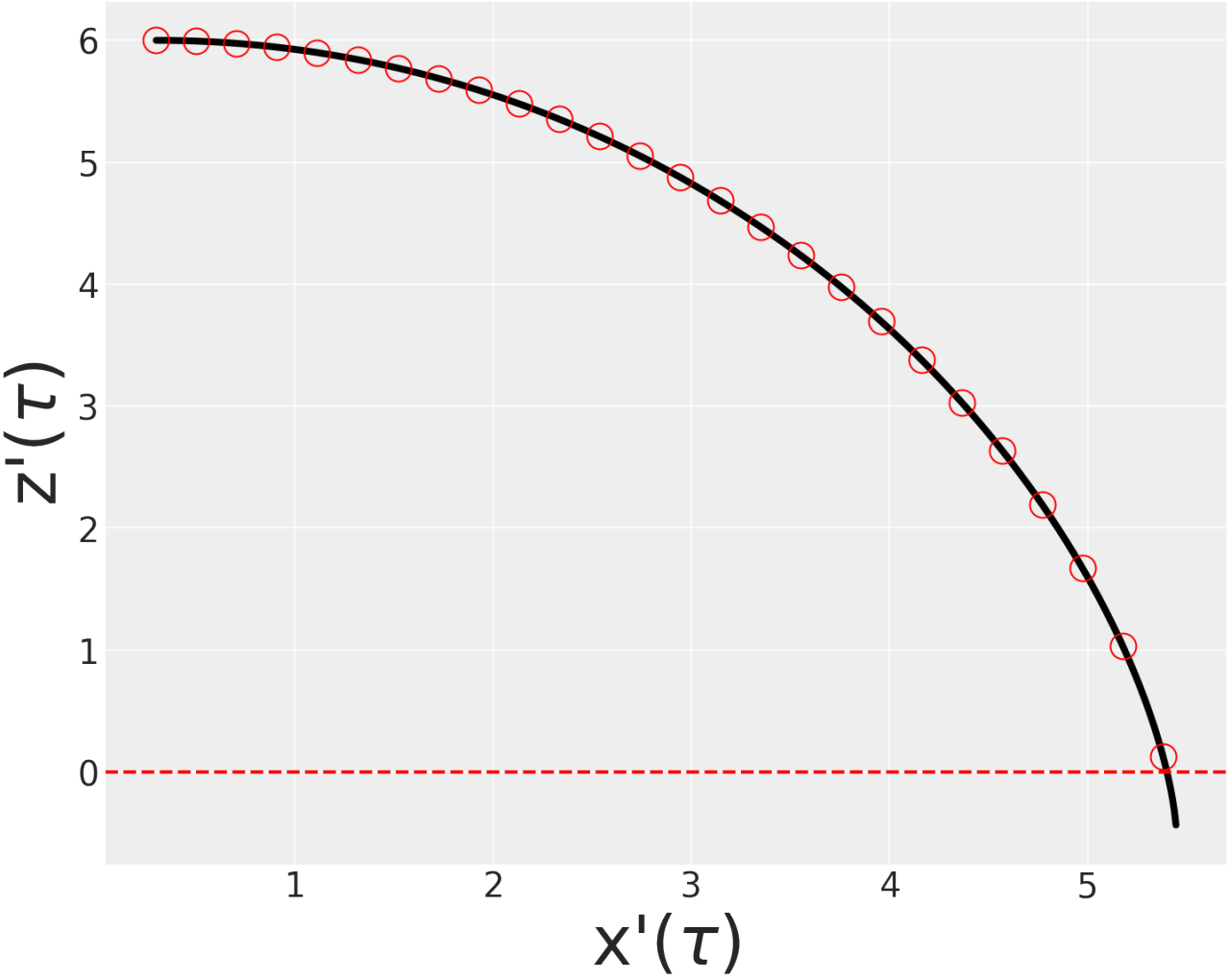
Running parameters

In [16]:

```
T = 60
pos_init=[0.001, 0.3, 0.1, 6.]
vel_init=[0.55, (1), 0., 0.]
## The solver --
sol_Bend_of_light = solve_ivp(F, [0, T], (pos_init+vel_init), t_eval
=np.linspace(0, T, int(T * 123 + 1)))
```

In [17]:

```
# Variables to plot
zplot=sol_Bend_of_light.y[3]
xplot=sol_Bend_of_light.y[1]

# New figure
fig=plt.figure(figsize=(10, 8),)
ax = fig.subplots(1, 1,  sharex=True, sharey=True)

# Extract solution
plt.plot(xplot, zplot, lw=4., c='k')
plt.plot(xplot[::25], zplot[::25], 'ro', fillstyle='none', markersiz
e=15)

ax = plt.gca()

ax.axhline(0, color="red", ls='--', lw=2)

#plt.grid()
plt.xlabel(r"x'($\tau$)",fontsize=40)
plt.xticks(fontsize=20)
plt.ylabel(r"z'($\tau$)",fontsize=40)
plt.yticks(fontsize=20)

## Borders
plt.rcParams["axes.edgecolor"] = "black"
plt.rcParams["axes.linewidth"] = 1

plt.show()
```



📌 This indeed confirms that photons do follow parabolic paths in the x-z spacetime plane.

Riemannian Geometry and Geodesics

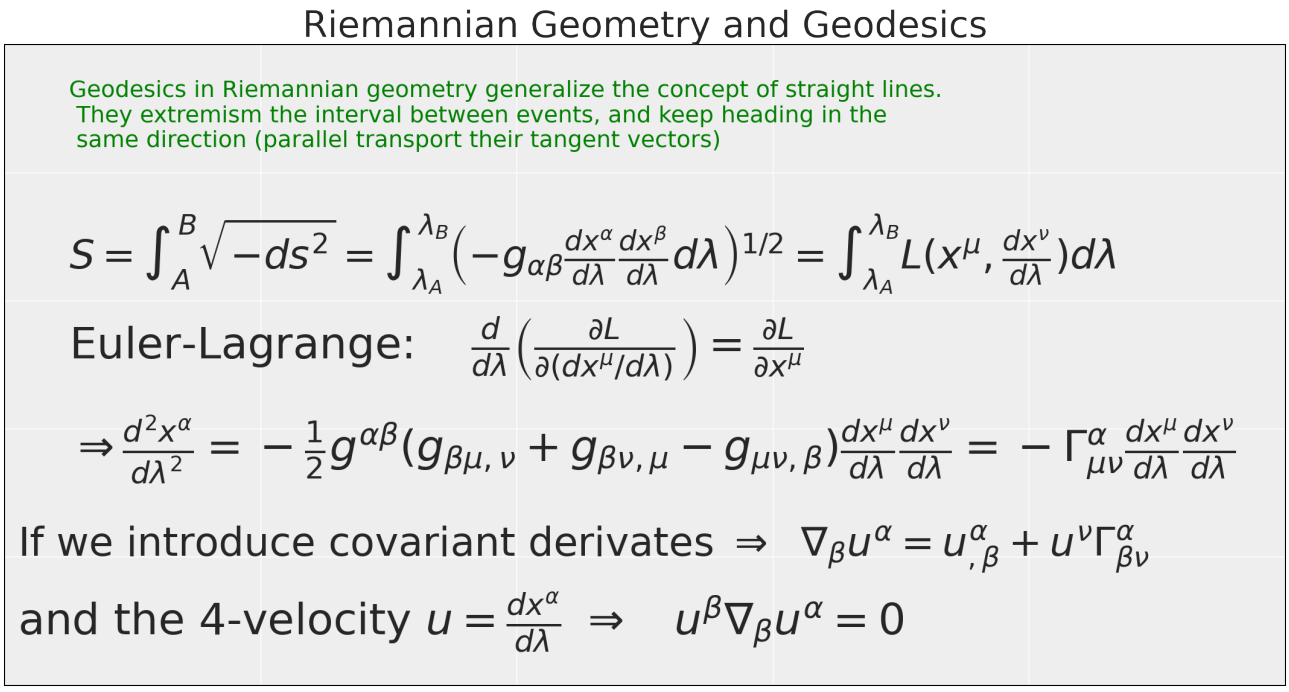
In [18]:

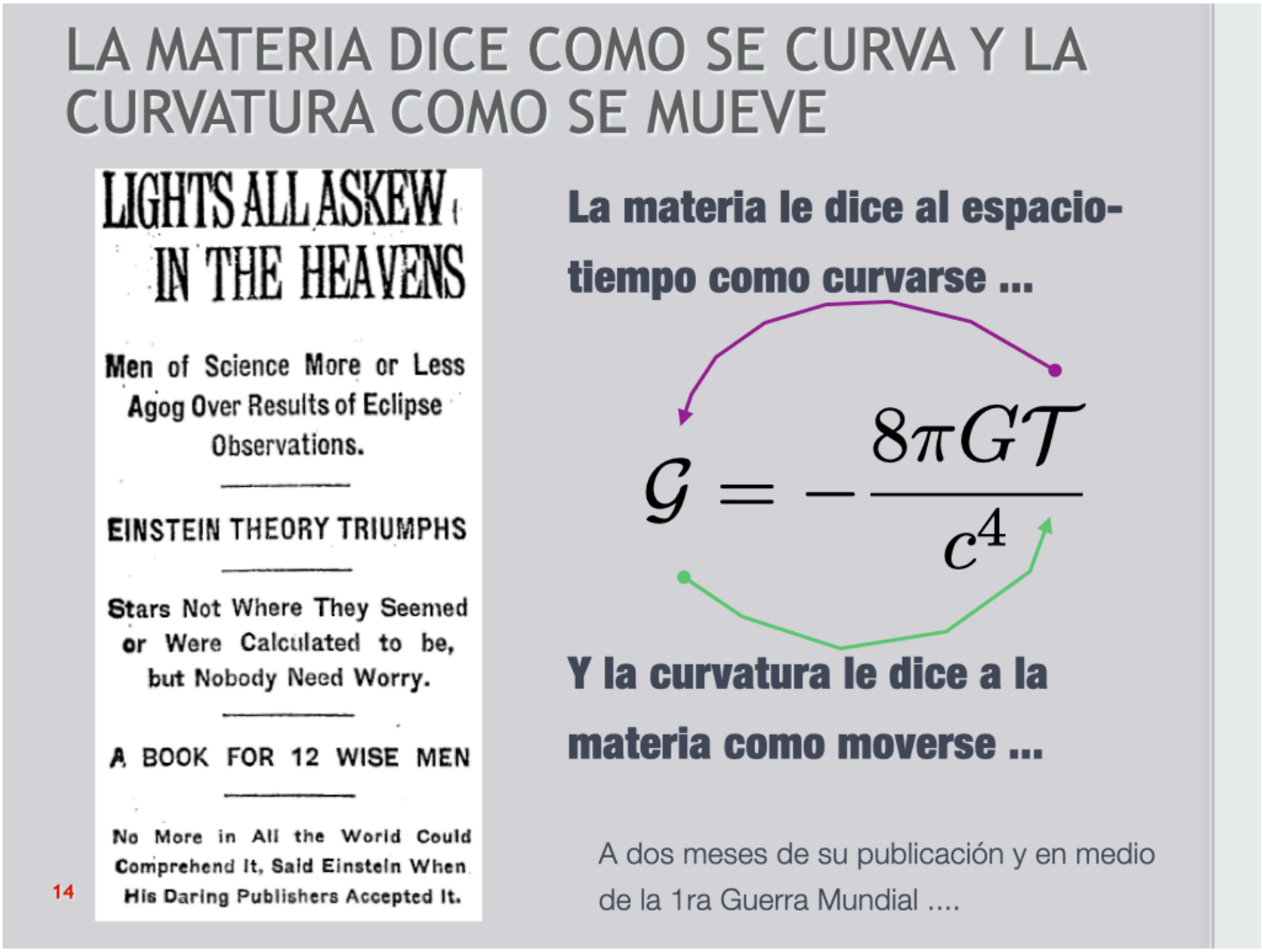
```
fig = plt.figure(figsize = (20, 30))
```

```
fig = plt.figure(figsize=(20, 30))
ax = fig.add_subplot(4,1,1, axes_class=AxesZero, aspect=0.5)

## Square of the plot
xl,xh = 0., 1.
yl,yh = xl,xh
plt.xlim(xl,xh)
plt.ylim(yl,yh)


# Move left y-axis and bottim x-axis to centre, passing through (0,0)
ax.set_xticklabels("", visible=False)
ax.set_yticklabels("", visible=False)


##########################################################################
##################
##########################################################################
##################
########
y1=0.84
x1=0.05
ax.annotate("Geodesics in Riemannian geometry generalize the concept
of straight lines. \n \
They extremism the interval between events, and keep heading in the
\n \
same direction (parallel transport their tangent vectors)",
            (x1,y1),   fontsize=19, color='g'
            )

## Notes
y1=0.65
x1=0.05
ax.annotate(r"$S = \int^B_A {\sqrt{-ds^2}} ="
r"\int^{\lambda_B}_{\lambda_A} \left( -g_{\alpha\beta} \frac{dx^{\al
pha}}{d\lambda} \frac{dx^{\beta}}{d\lambda} d\lambda \right)^{1/2}="
            r"\int^{\lambda_B}_{\lambda_A} L (x^{\mu}, \frac{dx^{\nu
}}{d\lambda})d\lambda$",
            (x1,y1),   fontsize=33,
            )
```

```
## Notes
y1=0.51
x1=0.05
ax.annotate(r"Euler-Lagrange:    "
            r"$ \frac{d}{d\lambda} \left(  \frac{\partial L}{\partia
l(dx^{\mu}/d\lambda) }   \right)="
            r"  \frac{\partial L}{\partial x^{\mu} }  $",
            (x1,y1),   fontsize=36,
            )


## Notes
y1=0.35
x1=0.05
ax.annotate(r"$\Rightarrow  \frac{d^2x^{\alpha}}{d\lambda^2} ="
            r"-\frac{1}{2}g^{\alpha\beta} ( g_{\beta\mu,\nu} + g_{\b
eta\nu,\mu}  - g_{\mu\nu,\beta} ) \frac{dx^{\mu}}{d\lambda}\frac{dx^
{\nu}}{d\lambda}="
            r"-\Gamma^{\alpha}_{\mu\nu}  \frac{dx^{\mu}}{d\lambda} \
frac{dx^{\nu}}{d\lambda}$",
            (x1,y1),   fontsize=36,
            )

## Notes
y1=0.2
x1=0.01
ax.annotate(r"If we introduce covariant derivates $\Rightarrow$  "
            r"$\nabla_{\beta} u^{\alpha} = u_{,\beta}^{\alpha} + u^{
\nu}\Gamma_{\beta\nu}^{\alpha}$",
            (x1,y1),   fontsize=33,
            )

## Notes
y1=0.08
x1=0.01
ax.annotate(
            r"and the 4-velocity $u=\frac{dx^{\alpha}}{d\lambda}$"
            r" $\Rightarrow$   $u^{\beta}  \nabla_{\beta} u^{\alpha}
```

```
=0 $",
            (x1,y1),    fontsize=36,
            )

######  End Notes ...
#########################################################################
##################
plt.title("Riemannian Geometry and Geodesics",  fontsize=30)



plt.show()
```





📌 Lab session #2

Geodesic with Schwarchild's metric:

In [19]:

```
# Create a manifold.
M = Manifold('M', 4)

# Create a patch.
patch = Patch('P', M)

# Basic symbols
c, r_s = symbols('c r_s')

# Coordinate system
schwarzchild_coord = CoordSystem('schwarzchild', patch, ['t', 'r', '
theta', 'phi'])

# Get the coordinate functions
t, r, theta, phi = schwarzchild_coord.coord_functions()

# Get the base one forms.
dt, dr, dtheta, dphi = schwarzchild_coord.base_oneforms()

# Auxiliar terms for the metric.
dt_2 = TP(dt, dt)
dr_2 = TP(dr, dr)
dtheta_2 = TP(dtheta, dtheta)
dphi_2 = TP(dphi, dphi)
factor = (1 - r_s / r)

# Build the metric
metric = factor * c ** 2 * dt_2 - 1 / factor * dr_2 - r ** 2 * (dthe
ta_2 + sin(theta)**2 * dphi_2)
metric = metric / c ** 2

# Get the Christoffel symbols of the second kind.
christoffel = metric_to_Christoffel_2nd(metric)
```

```
/opt/conda/lib/python3.7/site-packages/sympy/diffgeom/diffgeom.py:27
4: SymPyDeprecationWarning:

Passing str as coordinate symbol's name has been deprecated since
SymPy 1.7. Use Symbol which contains the name and assumption for
coordinate symbol instead. See
https://github.com/sympy/sympy/issues/19321 for more info.

  deprecated_since_version="1.7"
```

In [20]:

```
# Let's print this in an elegant way ;)
for i, j, k in product(range(4), range(4), range(4)):
    if christoffel[i, j, k] != 0:
        display(Math(f'\Gamma^{i}_{{{j},{k}}} = ' + latex(christoff
el[i, j, k])))
```

$$\Gamma^{0}_{0,1} = \frac{r_s}{2 \left(-\frac{r_s}{\mathbf{r}} + 1\right) \mathbf{r}^2}$$

$$\Gamma^{0}_{1,0} = \frac{r_s}{2 \left(-\frac{r_s}{\mathbf{r}} + 1\right) \mathbf{r}^2}$$

$$\Gamma^{1}_{0,0} = -\frac{r_s \left(\frac{c^2 r_s}{\mathbf{r}} - c^2\right)}{2\mathbf{r}^2}$$

$$\Gamma^{1}_{1,1} = \frac{c^2 r_s \left(\frac{c^2 r_s}{\mathbf{r}} - c^2\right)}{2\left(-\frac{c^2 r_s}{\mathbf{r}} + c^2\right)^2 \mathbf{r}^2}$$

$$\Gamma^{1}_{2,2} = \frac{\left(\frac{c^2 r_s}{\mathbf{r}} - c^2\right) \mathbf{r}}{c^2}$$

$$\Gamma^{1}_{3,3} = \frac{\left(\frac{c^2 r_s}{\mathbf{r}} - c^2\right)\sin^2(\theta)\mathbf{r}}{c^2}$$

$$\Gamma^{2}_{1,2} = \frac{1}{\mathbf{r}}$$

$$\Gamma^{2}_{2,1} = \frac{1}{\mathbf{r}}$$

$$\Gamma^{2}_{3,3} = -\sin(\theta)\cos(\theta)$$

$$\Gamma^{3}_{1,3} = \frac{1}{\mathbf{r}}$$

$$\Gamma^{3}_{2,3} = \frac{\cos(\theta)}{\sin(\theta)}$$

$$\Gamma^{3}_{3,1} = \frac{1}{\mathbf{r}}$$

$$\Gamma^{3}_{3,2} = \frac{\cos(\theta)}{\sin(\theta)}$$

EinsteinPy Project (https://einsteinpy.org)

In [21]:
```
!pip install einsteinpy
```

```
Collecting einsteinpy
  Downloading einsteinpy-0.4.0-py3-none-any.whl (88 kB)
     |████████████████████████████████| 88 kB 3.2 MB/s
```

```
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/sit
e-packages (from einsteinpy) (1.19.5)
Requirement already satisfied: sympy>=1.1 in /opt/conda/lib/python3.
7/site-packages (from einsteinpy) (1.8)
Requirement already satisfied: scipy>=1.0 in /opt/conda/lib/python3.
7/site-packages (from einsteinpy) (1.5.4)
Requirement already satisfied: numba!=0.49.0,>=0.46 in /opt/conda/li
b/python3.7/site-packages (from einsteinpy) (0.53.1)
Requirement already satisfied: plotly>=4.0 in /opt/conda/lib/python3
.7/site-packages (from einsteinpy) (4.14.3)
Collecting astropy
  Downloading astropy-4.2.1-cp37-cp37m-manylinux1_x86_64.whl (9.7 MB
)
     |████████████████████████████████| 9.7 MB 7.2 MB/s
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.
7/site-packages (from einsteinpy) (3.4.1)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.
7/site-packages (from numba!=0.49.0,>=0.46->einsteinpy) (49.6.0.post
20210108)
Requirement already satisfied: llvmlite<0.37,>=0.36.0rc1 in /opt/con
da/lib/python3.7/site-packages (from numba!=0.49.0,>=0.46->einsteinp
y) (0.36.0)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-
packages (from plotly>=4.0->einsteinpy) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in /opt/conda/lib/pyt
hon3.7/site-packages (from plotly>=4.0->einsteinpy) (1.3.3)
Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python
3.7/site-packages (from sympy>=1.1->einsteinpy) (1.2.1)
Collecting pyerfa
  Downloading pyerfa-2.0.0-cp37-cp37m-manylinux2010_x86_64.whl (746
kB)
     |████████████████████████████████| 746 kB 50.5 MB/s
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/pytho
n3.7/site-packages (from matplotlib->einsteinpy) (7.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/li
b/python3.7/site-packages (from matplotlib->einsteinpy) (2.8.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python
3.7/site-packages (from matplotlib->einsteinpy) (0.10.0)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/py
```

```
thon3.7/site-packages (from matplotlib->einsteinpy) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/p
ython3.7/site-packages (from matplotlib->einsteinpy) (1.3.1)
Installing collected packages: pyerfa, astropy, einsteinpy
Successfully installed astropy-4.2.1 einsteinpy-0.4.0 pyerfa-2.0.0
```

In [22]:

```
import numpy as np

from einsteinpy.geodesic import Timelike
from einsteinpy.plotting.geodesic import GeodesicPlotter
```

Initial conditions and parameters

In [23]:

```
position = [40., np.pi / 2, 0.]
momentum = [0., 0., 3.83405]
a = 0.
steps = 2000
delta = 1.
```

Computing the Geodesic

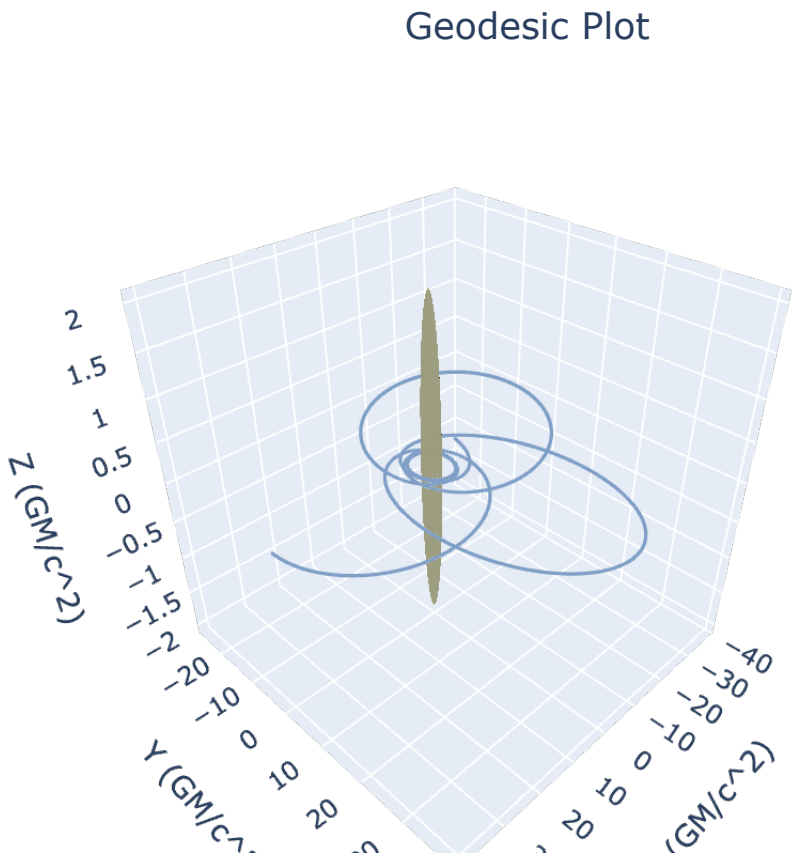
In [24]:

```
geod = Timelike(
    metric="Schwarzschild",
    metric_params=(a,),
    position=position,
    momentum=momentum,
    steps=steps,
    delta=delta,
    return_cartesian=True
)
```

Plotting

In [25]:

```
gpl = GeodesicPlotter()
gpl.plot(geod)
gpl.show()
```



Gravitational Waves (some equations)

In [26]:
```
az.style.use("arviz-darkgrid")
## Here we write the Eq.
myEq1={1:r"Do they look similar to Electrodynamics? ...",
       2:
    r"$G_{\mu \nu} = 8 \pi T_{\mu \nu}$"
}
#
myTitle='Fundamentals of Bayesian Statistics'
## Here we plot it
doall(myTitle,myEq1[1])
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/pylabtools.py:13
2: UserWarning:

There are no gridspecs with layoutgrids. Possibly did not call paren
t GridSpec with the "figure" keyword
```

GR	Maxwell
$G_{\mu\nu} = 8\pi T_{\mu\nu}$	
$\Box h^{\alpha\beta} = -16\pi\tau^{\alpha\beta}$	$\Box A^{\alpha} = -4\pi J^{\alpha}$
$\partial h^{\alpha\beta}/\partial x^{\beta} = 0$	$\partial A^{\alpha}/\partial x^{\alpha} = 0$

$$\Box \equiv -\partial^2/\partial t^2 + \nabla^2$$
$$h^{\alpha\beta} \equiv \eta^{\alpha\beta} - (-g)^{1/2} g^{\alpha\beta}$$ Tensor potencial gravitacional
$$\tau^{\alpha\beta} = (-g)T^{\alpha\beta} + (16\pi)^{-1}\Lambda^{\alpha\beta}$$ PseudoTensor efectivo stress-energy

Posterior distributions, Bayesian Learning and Model Evidence

In [ ]:

In [27]:

```
az.style.use("arviz-darkgrid")
## Here we write the Eq.
myEq1={1:r"From priors to posteriors ...",
       2:
    r"$\langle \theta ^j \rangle = \int \theta ^j p({\bf \theta} | {
\bf d}, \mathcal{H}) d {\bf \theta}$",
       3:
    r"$p( \theta ^j ) = \int  p({\bf \theta} | {\bf d}, \mathcal{H})
\prod _{k \neq j} d {\bf \theta }^k$",
       4:
       r"$p( \theta ^i, \theta ^j ) = \int  p({\bf \theta} | {\bf d}
, \mathcal{H}) \prod _{k \neq i,j} d {\bf \theta }^k$"
}
#
myTitle='Fundamentals of Bayesian Statistics'
## Here we plot it
doall(myTitle,myEq1[1])
doall("",myEq1[2])
doall("",myEq1[3])
doall("",myEq1[4])
```

Fundamentals of Bayesian Statistics

From priors to posteriors ...

$$\langle \theta^j \rangle = \int \theta^j p(\mathbf{\theta}|\mathbf{d}, \mathcal{H}) d\mathbf{\theta}$$

$$p(\theta^j) = \int p(\mathbf{\theta}|\mathbf{d}, \mathcal{H}) \prod_{k \neq j} d\mathbf{\theta}^k$$

$$p(\theta^i, \theta^j) = \int p(\mathbf{\theta}|\mathbf{d}, \mathcal{H}) \prod_{k \neq i,j} d\mathbf{\theta}^k$$

📌 Lab session #3

We will follow an old version of this scenario (https://git.ligo.org/daniel.wysocki/bayesian-parametric-population-models) (not available on the web anymore):

Let's work on a Gaussian population

We would like to extract the parameters (mean and variance) of a synthetic population (normally distributed). We apply an efficiency function, which causes positive-valued samples to only be detected 50% of the time, and negative-valued samples to always be detected.

Poisson posterior formula

The probability density function (pdf) is

$$\rho(\lambda \mid \Lambda) = \mathcal{R}\mathcal{N}(\lambda; \mu, \sigma^2) \tag{1}$$

where $\Lambda = (\mathcal{R}, \mu, \sigma^2)$.

The expected number of detections in an observing time T is

$$\mu(\Lambda) = T \int_{-\infty}^{+\infty} \mathrm{d}\lambda \, \eta(\lambda) \, \rho(\lambda \mid \Lambda) \tag{2}$$

where $\eta(\lambda)$ is the efficiency function

$$\eta(\lambda) = \begin{cases} 1/2, & \text{if } \lambda > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

We can evaluate $\mu(\Lambda)$ analytically to be

$$\mu(\Lambda) = \frac{T \cdot \mathcal{R}}{4} \left[3 + \mathrm{erf}\left(-\frac{\mu}{\sqrt{2\sigma^2}} \right) \right] \tag{4}$$

We assume a prior which is uniform in $\log_{10} \mathcal{R}$, μ, and $\log_{10}(\sigma^2)$

$$\pi(\Lambda) = \mathcal{U}(\log_{10} \mathcal{R}; \log_{10} \mathcal{R}_{\min}, \log_{10} \mathcal{R}_{\max}) \mathcal{U}(\log_{10} \mu; \log_{10} \mu_{\min}, \log_{10} \mu_{\max}) \mathcal{U}(\log_{10}(\sigma^2); \log_{10}($$

We also assume that our measurements of each detection have infinite precision, which makes the both the likelihood and posterior a Dirac delta function centered at the true value for that event, λ_n, so long as the prior has support there (which any reasonable prior would). We then only need one posterior sample from each event – the true value – in order to describe it accurately, and we set the prior samples to `None`, which instructs the code to perform no division by the prior, equivalent to a uniform prior (in the function posterior_samples).

Synthesizing our data set

We assume an event rate $\mathcal{R}$, and an observing time T, which correspond to an *intrinsic* mean number of events $\mathcal{R} \times T$ in that time. The *intrinsic* number of events which actually occur is determined by a Poisson random variable

$$N_{\rm int} \sim {\rm Poi}(\mathcal{R} \times T) \tag{6}$$

We draw one sample for $N_{\rm int}$. With this number in hand, we draw $N_{\rm int}$ samples from our intrinsic population distribution

$$\lambda_i \sim \mathcal{N}(\mu, \sigma^2) \tag{7}$$

for some assumed values for (μ, σ^2), and for $i \in \{1, \ldots, N_{\rm int}\}$. Due to our detection efficiency $\eta(\lambda)$ described above, we then must discard positive-valued samples with 50% probability each. Note: this does not necessarily mean we discard half of the samples, as each sample is considered independently, and its acception/rejection is random. In principle we could wind up discarding none of them, all of them, half of them, or anywhere inbetween – though the expected value is half. This leaves us with a (likely) smaller number of samples, λ_i for $i \in \{1, \ldots, N\}$, where N is the number of samples after the rejection procedure, which satisfies $0 \leq N \leq N_{\rm int}$.

Population inference

Putting together the pieces of the inhomogeneous Poisson posterior function, as well as our N synthesized detections, we perform a Markov chain Monte Carlo using several classes and methods found here (https://git.ligo.org/daniel.wysocki/bayesian-parametric-population-models). We initialize the MCMC by drawing samples from our prior $\pi(\Lambda)$. We evolve the chain using `n_walkers` in parallel, each obtaining `n_samples`. We then remove the first `half_n_samples` samples from each chain (to remove the effects of burnin). These are set to fixed numbers for simplicity, though in principle they should be determined empirically from the data. With all of the posterior samples $\Lambda_0, \Lambda_1, \ldots, \Lambda_S$ in hand, we create a `post_samples_cleaned.plot_corner` of the samples, with the true values overplotted. If the code is working properly, there is a high probability that the true values will lie in a region of high density.

In [28]:
```
!ls ../input/laconga-bhs-collisions-pics/bayesian-parametric-populat
ion-models-master/bayesian-parametric-population-models-master
```

```
LICENSE    docs      misc      requirements-docs.txt  setup.py
tests
README.md  examples  mypy.ini  setup.cfg              src
tox.ini
```

In [29]:

```
!pip install -U ../input/laconga-bhs-collisions-pics/bayesian-parametric-population-models-master/bayesian-parametric-population-models-master/
```

```
Processing /kaggle/input/laconga-bhs-collisions-pics/bayesian-parametric-population-models-master/bayesian-parametric-population-models-master
Requirement already satisfied: astropy<5.0.0,>=3.0.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (4.2.1)
Collecting corner<3.0.0,>=2.0.0
  Downloading corner-2.2.1-py3-none-any.whl (15 kB)
Collecting emcee<4.0.0,>=2.2.0
  Downloading emcee-3.0.2-py2.py3-none-any.whl (41 kB)
     |████████████████████████████████| 41 kB 317 kB/s
Requirement already satisfied: h5py<4.0.0,>=2.7.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (2.10.0)
Collecting mpmath<1.1.0,>=1.0.0
  Downloading mpmath-1.0.0.tar.gz (511 kB)
     |████████████████████████████████| 511 kB 6.9 MB/s
Requirement already satisfied: numpy<2.0.0,>=1.13.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (1.19.5)
Requirement already satisfied: matplotlib<4.0.0,>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (3.4.1)
Collecting matplotlib-label-lines<0.4.0,>=0.3.0
  Downloading matplotlib_label_lines-0.3.9-py3-none-any.whl (7.5 kB)
Requirement already satisfied: scipy<2.0.0,>=1.0.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (1.5.4)
Requirement already satisfied: six<2.0.0,>=1.10.0 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (1.15.0)
Requirement already satisfied: typing_extensions<4.0.0,>=3.6.2 in /opt/conda/lib/python3.7/site-packages (from pop-models==2.0.0a2) (3.7.4.3)
Requirement already satisfied: pyerfa in /opt/conda/lib/python3.7/site-packages (from astropy<5.0.0,>=3.0.0->pop-models==2.0.0a2) (2.0.0)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.7/site-packages (from matplotlib<4.0.0,>=2.0.0->pop-models==2.0.0a2) (2.8.1)
```

```
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/py
thon3.7/site-packages (from matplotlib<4.0.0,>=2.0.0->pop-models==2.
0.0a2) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/pytho
n3.7/site-packages (from matplotlib<4.0.0,>=2.0.0->pop-models==2.0.0
a2) (7.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python
3.7/site-packages (from matplotlib<4.0.0,>=2.0.0->pop-models==2.0.0a
2) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/p
ython3.7/site-packages (from matplotlib<4.0.0,>=2.0.0->pop-models==2
.0.0a2) (1.3.1)
Building wheels for collected packages: pop-models, mpmath
  Building wheel for pop-models (setup.py) ... - \ | done
  Created wheel for pop-models: filename=pop_models-2.0.0a2-py3-none
-any.whl size=399421 sha256=389cf19af8c5e2ed4eb28ae3322cd035ea390507
d7c75e398ee498932c6f13b3
  Stored in directory: /root/.cache/pip/wheels/b9/b7/31/e9017352b401
ae72dcb9d00835cf0da8e7299e343c59101a27
  Building wheel for mpmath (setup.py) ... - \ | done
  Created wheel for mpmath: filename=mpmath-1.0.0-py3-none-any.whl s
ize=531604 sha256=eef3deefe8bdd3ccf6a355829d74907b692038eb8ecf1261e3
25f375f8bcd8a5
  Stored in directory: /root/.cache/pip/wheels/a8/b3/48/21c44d991860
b1ffad1c0f01b5a02952fdd8a2442ba1413ffa
Successfully built pop-models mpmath
Installing collected packages: mpmath, matplotlib-label-lines, emcee
, corner, pop-models
  Attempting uninstall: mpmath
    Found existing installation: mpmath 1.2.1
    Uninstalling mpmath-1.2.1:
      Successfully uninstalled mpmath-1.2.1
Successfully installed corner-2.2.1 emcee-3.0.2 matplotlib-label-lin
es-0.3.9 mpmath-1.0.0 pop-models-2.0.0a2
```

In [30]:

```
%matplotlib inline
import numpy
#import matplotlib
```

```
matplotlib.use("Agg")
from matplotlib import gridspec
#import matplotlib.pyplot as plt
import scipy.stats
import scipy.special

import pprint
from pop_models.coordinate import Coordinate, CoordinateSystem
from pop_models.population import Population
from pop_models.poisson_mean import PoissonMean
from pop_models.detection import Detection
numpy.set_printoptions(threshold=10)
seed = 42
random = numpy.random.RandomState(seed)
sqrt2 = numpy.sqrt(2.0)




# Define a single coordinate called "x"
X = Coordinate("x")
# Define a 1-dimensional coordinate system containing only "x"
coord_system = CoordinateSystem(X)


param_names = ("rate", "mean", "var")
def accept(prob, random_state=random):
        """
        Returns ``True`` with probability ``prob``.
        """
        p = random_state.uniform()
        return p < prob



class SimplePopulation(Population):
    """
    Defines a simple population with a single observable ("x"), which
obey a
```

```
    Gaussian distribution with unknown mean "mean" and known unit vari
ance.
    Normalization of this distribution is given by an unknown paramete
r "rate".
    """
    def __init__(self):
        # No special information needed, just pass the CoordinateSyste
m to the
        # parent's constructor method.
        super().__init__(coord_system, param_names)

    def normalization(self, parameters, where=True, **kwargs):
        # The normalization of the intensity function is just the "rat
e"
        # parameter.
        return parameters["rate"]


    def pdf(self, observables, parameters, where=True, **kwargs):
        # First argument is a list of observables, but there's only on
e, "x", so
        # we pull that out of the list.
        #print(observables[0])
        x, = observables

        # The only parameter that affects the PDF is the "mean", so we
extract
        # that.
        mean = parameters["mean"]
        var = parameters["var"]

        # PDF is a normal distribution with unit variance.  This is th
e
        # functional form for such a distribution.  Note that by using
        # `numpy.subtract.outer(mean, x)` instead of `mean - x`, we co
mpute
        # all combinations of `mean[i_1,...,i_n] - x[j_1,...,j_m]`, an
d the
        # shape of the output array is `numpy.shape(mean) + numpy.shap
```

```
e(x)`.
        # Handling array shapes like this is a requirement of the PopM
odels API.
        delta = numpy.subtract.outer(mean, x)/var
        return (
            numpy.exp(-0.5*numpy.square(delta)) /
            (var*numpy.sqrt(2.0*numpy.pi))

        )



class SimpleExpectedDetections(PoissonMean):
    """
    We assume a perfect survey, so the expected number of detections i
s just the
    event rate multiplied by the observing time.
    """
    def __init__(self, obs_time, population):
        super().__init__(population)
        self.obs_time = obs_time

    def __call__(self, parameters, where=True, **kwargs):
        mu = parameters["mean"]
        sigma = parameters["var"]

        return (
            self.population.normalization(parameters, where=where) *
            self.obs_time * 0.25 * (3.0 + scipy.special.erf(-mu / (s
qrt2*sigma)))
        )



class SimpleDetection(Detection):
    """
    We assume detections with perfect measurements, so all posterior s
amples are
    of the correct value, and if the `likelihood` method were implemen
```

```
ted, it
    would be a delta function at the true value (which would not integ
rate well
    numerically, so we must use `posterior_samples` since that makes a
Monte
    Carlo integral perfectly adaptive).
    """
    def __init__(self, x_true):
        # Parent constructor needs to know the coordinate system.
        super().__init__(coord_system)
        # In addition, we need to know the true value.
        self.x_true = x_true

    def posterior_samples(self, n_samples, random_state=None):
        # Return the true value repeated once for each requested sampl
e.
        x_samples = numpy.broadcast_to(self.x_true, n_samples)
        # CoordinateSystem is 1-D so we return a 1-element list.
        post_samples = [x_samples]
        # Specify that the prior is uniform.
        prior_samples = None

        return post_samples, prior_samples
```

Simulated data parameters

In [31]:

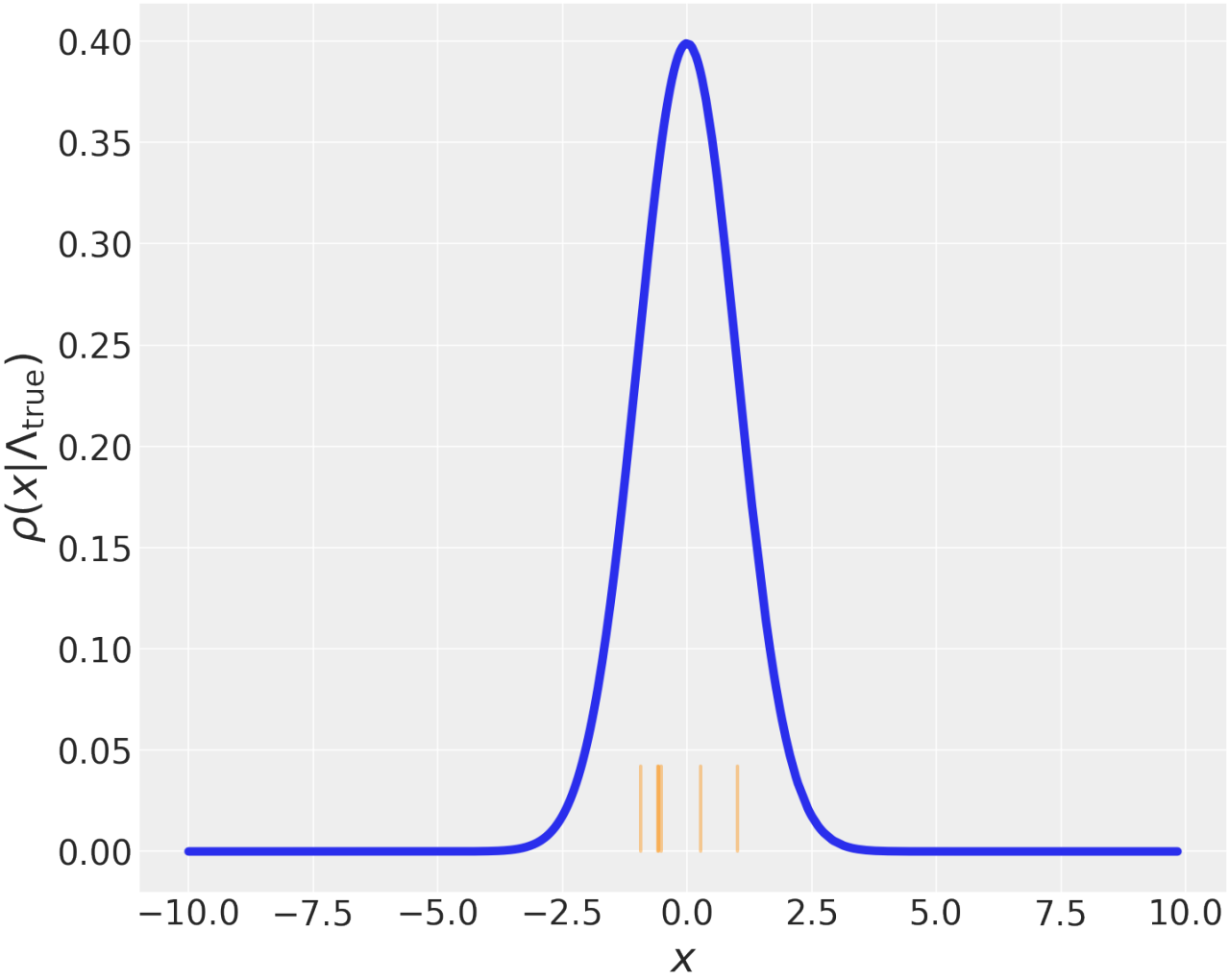
```
obs_time = 10.0
population = SimplePopulation()
expval = SimpleExpectedDetections(obs_time, population)


# Synthetic Parameters
rate_true = 1.0
mean_true = 0.0
var_true = 1.0
```

```
# Names of the parameters.
parameters_true = {"rate": rate_true, "mean": mean_true, "var": var_
true}


expval_true = expval(parameters_true)


## Here the simulation of the total detection ..
n_detections = scipy.stats.poisson(expval_true).rvs(1, random_state=
random)



## Drawing the synthetic values of "x" from a normal distribution with
variance=1.
x_truths = (
    scipy.stats.norm(mean_true, 1.0)
    .rvs(n_detections, random_state=random)
)



## Parameters for plotting ..
n_plotting = 500

x_min_plotting = -10.0
x_max_plotting = +10.0
x_plotting = numpy.linspace(x_min_plotting, x_max_plotting, n_plotti
ng)


## Apply selection effects, rejecting positive values with probability
50% ##
x_plotting = numpy.array([
            lamb
            for lamb in x_plotting
            if lamb <= 0.0 or accept(0.50)
            ])
```

Figure of the synthetic distribution of "x"

In [32]:

```
fig_pop, ax_pop = plt.subplots(figsize=(10,8))
# Plot the population's intensity function.
ax_pop.plot(
    x_plotting,
    population.intensity((x_plotting,), parameters=parameters_true),
    lw=5
)

## Maximum in y
_, ax_pop_ymax = ax_pop.get_ylim()

# Plot the detected events, and where they lie on the population.
for x_truth in x_truths:
    ax_pop.plot(
        [x_truth, x_truth], [0.0, 0.1*ax_pop_ymax],
        color="darkorange", linestyle="solid", alpha=0.4, lw=2
    )

ax_pop.set_xlabel(r"$x$",fontsize=25)
ax_pop.set_ylabel(r"$\rho(x | \Lambda_{\mathrm{true}})$",fontsize=25
)
ax_pop.tick_params(axis='both', which='major', labelsize=20)
fig_pop.show()
```



(MCMC) Running parameters (1)

In [33]:

```
n_walkers = 8
half_n_samples = 64
n_samples = 2*half_n_samples

output_directory = "posterior_samples"
# File to store cleaned posterior samples in.
output_samples_filename = "posterior_samples.hdf5"

chunk_size = 16
```

In [34]:

```
!rm {output_samples_filename}
!rm -r {output_directory}
!ls
```

```
rm: cannot remove 'posterior_samples.hdf5': No such file or director
y
rm: cannot remove 'posterior_samples': No such file or directory
__notebook__.ipynb
```

In [35]:

```
from pop_models.detection import ReweightedPosteriorsDetectionLikeli
hood
from pop_models.integrators import AdaptiveMCIntegrator

# Detections, based in the detection model class "SimpleDetection".
detections = [SimpleDetection(x_true) for x_true in x_truths]

mc_integrator = AdaptiveMCIntegrator(
    iter_start=1, iter_max=2,
)

detection_likelihoods = [
    ReweightedPosteriorsDetectionLikelihood(
        population, detection,
        mc_integrator=mc_integrator,
    )
    for detection in detections
]
```

(log-Priors) Running parameters (2)

In [36]:

```
rate_min_prior = 0.01
rate_max_prior = 3.50
mean_min_prior = -1.5
mean_max_prior = +1.0
var_min_prior = +0.5
var_max_prior = +3.0


## Defining the priors ########################################
def log_prior(parameters, *args, **kwargs):
    """
    Assume a prior uniform in rate and mean.
    """
    rate = parameters["rate"]
    mean = parameters["mean"]
    var = parameters["var"]

    has_support = (
        (rate >= rate_min_prior) &
        (rate <= rate_max_prior) &
        (mean >= mean_min_prior) &
        (mean <= mean_max_prior) &
        (var >= var_min_prior) &
        (var <= var_max_prior)
    )

    return numpy.where(has_support, 0.0, -numpy.inf)
##############################################################



rate_init = random.uniform(rate_min_prior, rate_max_prior, n_walkers
)
mean_init = random.uniform(mean_min_prior, mean_max_prior, n_walkers
)
var_init = random.uniform(var_min_prior, var_max_prior, n_walkers)
init_state = numpy.column_stack((rate_init, mean_init, var_init))
```

In [37]:

```
from pop_models.posterior import H5RawPosteriorSamples
from pop_models.posterior import PopulationInferenceEmceeSampler

# Create object for storing posterior samples, with an HDF5 backend.
post_samples = H5RawPosteriorSamples.create(
    output_directory,
    population.param_names,
    init_state,
    chunk_size=chunk_size,
)

pop_inference = PopulationInferenceEmceeSampler(
    post_samples,
    expval, detection_likelihoods, log_prior,
    random_state=random,
    verbose=True,
)

## Summary
post_samples.summary()
```

```
Directory: posterior_samples/
Samples: 1
Walkers: 8
Chunk size: 16
Variables: ['rate', 'mean', 'var']
Constants: {}
Duplicates: {}
Files:
├── master.hdf5 : size=49941B
└── subfile-000.hdf5 : size=6289B, samples=1
```

Running the MCMC

In [38]:

```
samples = pop_inference.posterior_samples(half_n_samples)
```

```
samples_equiv = post_samples.get_samples(slice(-half_n_samples, None
))

## testing if they are equal!
numpy.testing.assert_equal(
    samples,
    samples_equiv,
)
############################

init_state_equiv = post_samples.get_samples(0)
samples_all = post_samples.get_samples(...)


## testing if they are equal!
numpy.testing.assert_equal(
    init_state,
    samples_all[0],
)
numpy.testing.assert_equal(
    samples,
    samples_all[1:],
)
############################



# Close the old ``H5RawPosteriorSamples`` object.
post_samples.close()


# Create a new ``H5RawPosteriorSamples`` object, picking up where we l
eft off.
post_samples = H5RawPosteriorSamples(output_directory, "r+")
# Create another ``PopulationInference`` object, which doesn't care if
this is
# the beginning of a new run, or a resuming of an old run.  It just us
```

```
es the
# last sample in ``post_samples`` as a starting point.
pop_inference = PopulationInferenceEmceeSampler(
    post_samples,
    expval, detection_likelihoods, log_prior,
    random_state=random,
    verbose=True,
)


pop_inference.posterior_samples(half_n_samples)
log_post_samples = post_samples.get_posterior_log_prob(slice(-n_samp
les, None))
log_prior_samples = post_samples.get_prior_log_prob(slice(-n_samples
, None))
```

```
2021-06-23 20:51:38.699369; Progress: 0%; Samples: 0
2021-06-23 20:51:38.730208; Progress: 1%; Samples: 1
2021-06-23 20:51:38.740898; Progress: 3%; Samples: 2
2021-06-23 20:51:38.752596; Progress: 4%; Samples: 3
2021-06-23 20:51:38.763277; Progress: 6%; Samples: 4
2021-06-23 20:51:38.777011; Progress: 7%; Samples: 5
2021-06-23 20:51:38.788632; Progress: 9%; Samples: 6
2021-06-23 20:51:38.800848; Progress: 10%; Samples: 7
2021-06-23 20:51:38.812401; Progress: 12%; Samples: 8
2021-06-23 20:51:38.824829; Progress: 14%; Samples: 9
2021-06-23 20:51:38.835999; Progress: 15%; Samples: 10
2021-06-23 20:51:38.847730; Progress: 17%; Samples: 11
2021-06-23 20:51:38.860148; Progress: 18%; Samples: 12
2021-06-23 20:51:38.871433; Progress: 20%; Samples: 13
2021-06-23 20:51:38.883446; Progress: 21%; Samples: 14
2021-06-23 20:51:38.894326; Progress: 23%; Samples: 15
2021-06-23 20:51:38.908797; Progress: 25%; Samples: 16
2021-06-23 20:51:38.921200; Progress: 26%; Samples: 17
2021-06-23 20:51:38.931692; Progress: 28%; Samples: 18
2021-06-23 20:51:38.943314; Progress: 29%; Samples: 19
2021-06-23 20:51:38.954893; Progress: 31%; Samples: 20
2021-06-23 20:51:38.967839; Progress: 32%; Samples: 21
2021-06-23 20:51:38.979476; Progress: 34%; Samples: 22
```

```
2021-06-23 20:51:38.991269; Progress: 35%; Samples: 23
2021-06-23 20:51:39.002981; Progress: 37%; Samples: 24
2021-06-23 20:51:39.013795; Progress: 39%; Samples: 25
2021-06-23 20:51:39.026270; Progress: 40%; Samples: 26
2021-06-23 20:51:39.037160; Progress: 42%; Samples: 27
2021-06-23 20:51:39.050349; Progress: 43%; Samples: 28
2021-06-23 20:51:39.063447; Progress: 45%; Samples: 29
2021-06-23 20:51:39.075255; Progress: 46%; Samples: 30
2021-06-23 20:51:39.086204; Progress: 48%; Samples: 31
2021-06-23 20:51:39.099355; Progress: 50%; Samples: 32
2021-06-23 20:51:39.110603; Progress: 51%; Samples: 33
2021-06-23 20:51:39.121750; Progress: 53%; Samples: 34
2021-06-23 20:51:39.134710; Progress: 54%; Samples: 35
2021-06-23 20:51:39.144880; Progress: 56%; Samples: 36
2021-06-23 20:51:39.156879; Progress: 57%; Samples: 37
2021-06-23 20:51:39.167814; Progress: 59%; Samples: 38
2021-06-23 20:51:39.180310; Progress: 60%; Samples: 39
2021-06-23 20:51:39.191259; Progress: 62%; Samples: 40
2021-06-23 20:51:39.205204; Progress: 64%; Samples: 41
2021-06-23 20:51:39.217082; Progress: 65%; Samples: 42
2021-06-23 20:51:39.228590; Progress: 67%; Samples: 43
2021-06-23 20:51:39.240512; Progress: 68%; Samples: 44
2021-06-23 20:51:39.251371; Progress: 70%; Samples: 45
2021-06-23 20:51:39.263666; Progress: 71%; Samples: 46
2021-06-23 20:51:39.274915; Progress: 73%; Samples: 47
2021-06-23 20:51:39.291477; Progress: 75%; Samples: 48
2021-06-23 20:51:39.304489; Progress: 76%; Samples: 49
2021-06-23 20:51:39.316145; Progress: 78%; Samples: 50
2021-06-23 20:51:39.328314; Progress: 79%; Samples: 51
2021-06-23 20:51:39.338784; Progress: 81%; Samples: 52
2021-06-23 20:51:39.350144; Progress: 82%; Samples: 53
2021-06-23 20:51:39.361378; Progress: 84%; Samples: 54
2021-06-23 20:51:39.374173; Progress: 85%; Samples: 55
2021-06-23 20:51:39.386688; Progress: 87%; Samples: 56
2021-06-23 20:51:39.399379; Progress: 89%; Samples: 57
2021-06-23 20:51:39.413866; Progress: 90%; Samples: 58
2021-06-23 20:51:39.425441; Progress: 92%; Samples: 59
2021-06-23 20:51:39.435748; Progress: 93%; Samples: 60
2021-06-23 20:51:39.446839; Progress: 95%; Samples: 61
```

```
2021-06-23 20:51:39.457850; Progress: 96%; Samples: 62
2021-06-23 20:51:39.468017; Progress: 98%; Samples: 63
2021-06-23 20:51:39.474044; Progress: 100%; Samples: 64
2021-06-23 20:51:39.491114; Progress: 0%; Samples: 0
2021-06-23 20:51:39.516079; Progress: 1%; Samples: 1
2021-06-23 20:51:39.527202; Progress: 3%; Samples: 2
2021-06-23 20:51:39.538576; Progress: 4%; Samples: 3
2021-06-23 20:51:39.549670; Progress: 6%; Samples: 4
2021-06-23 20:51:39.561984; Progress: 7%; Samples: 5
2021-06-23 20:51:39.573437; Progress: 9%; Samples: 6
2021-06-23 20:51:39.584662; Progress: 10%; Samples: 7
2021-06-23 20:51:39.598833; Progress: 12%; Samples: 8
2021-06-23 20:51:39.609980; Progress: 14%; Samples: 9
2021-06-23 20:51:39.623336; Progress: 15%; Samples: 10
2021-06-23 20:51:39.635767; Progress: 17%; Samples: 11
2021-06-23 20:51:39.646482; Progress: 18%; Samples: 12
2021-06-23 20:51:39.657361; Progress: 20%; Samples: 13
2021-06-23 20:51:39.668850; Progress: 21%; Samples: 14
2021-06-23 20:51:39.679924; Progress: 23%; Samples: 15
2021-06-23 20:51:39.692450; Progress: 25%; Samples: 16
2021-06-23 20:51:39.704704; Progress: 26%; Samples: 17
2021-06-23 20:51:39.715475; Progress: 28%; Samples: 18
2021-06-23 20:51:39.729015; Progress: 29%; Samples: 19
2021-06-23 20:51:39.740485; Progress: 31%; Samples: 20
2021-06-23 20:51:39.751817; Progress: 32%; Samples: 21
2021-06-23 20:51:39.764885; Progress: 34%; Samples: 22
2021-06-23 20:51:39.776119; Progress: 35%; Samples: 23
2021-06-23 20:51:39.787095; Progress: 37%; Samples: 24
2021-06-23 20:51:39.798133; Progress: 39%; Samples: 25
2021-06-23 20:51:39.808729; Progress: 40%; Samples: 26
2021-06-23 20:51:39.821815; Progress: 42%; Samples: 27
2021-06-23 20:51:39.832829; Progress: 43%; Samples: 28
2021-06-23 20:51:39.843603; Progress: 45%; Samples: 29
2021-06-23 20:51:39.855200; Progress: 46%; Samples: 30
2021-06-23 20:51:39.865850; Progress: 48%; Samples: 31
2021-06-23 20:51:39.880507; Progress: 50%; Samples: 32
2021-06-23 20:51:39.890751; Progress: 51%; Samples: 33
2021-06-23 20:51:39.902833; Progress: 53%; Samples: 34
2021-06-23 20:51:39.914108; Progress: 54%; Samples: 35
```

```
2021-06-23 20:51:39.927300; Progress: 56%; Samples: 36
2021-06-23 20:51:39.937645; Progress: 57%; Samples: 37
2021-06-23 20:51:39.950373; Progress: 59%; Samples: 38
2021-06-23 20:51:39.961596; Progress: 60%; Samples: 39
2021-06-23 20:51:39.975649; Progress: 62%; Samples: 40
2021-06-23 20:51:39.990336; Progress: 64%; Samples: 41
2021-06-23 20:51:40.001744; Progress: 65%; Samples: 42
2021-06-23 20:51:40.014559; Progress: 67%; Samples: 43
2021-06-23 20:51:40.025469; Progress: 68%; Samples: 44
2021-06-23 20:51:40.036541; Progress: 70%; Samples: 45
2021-06-23 20:51:40.048149; Progress: 71%; Samples: 46
2021-06-23 20:51:40.058899; Progress: 73%; Samples: 47
2021-06-23 20:51:40.074365; Progress: 75%; Samples: 48
2021-06-23 20:51:40.085773; Progress: 76%; Samples: 49
2021-06-23 20:51:40.096461; Progress: 78%; Samples: 50
2021-06-23 20:51:40.109202; Progress: 79%; Samples: 51
2021-06-23 20:51:40.119709; Progress: 81%; Samples: 52
2021-06-23 20:51:40.132137; Progress: 82%; Samples: 53
2021-06-23 20:51:40.144919; Progress: 84%; Samples: 54
2021-06-23 20:51:40.158162; Progress: 85%; Samples: 55
2021-06-23 20:51:40.169036; Progress: 87%; Samples: 56
2021-06-23 20:51:40.181474; Progress: 89%; Samples: 57
2021-06-23 20:51:40.191856; Progress: 90%; Samples: 58
2021-06-23 20:51:40.202602; Progress: 92%; Samples: 59
2021-06-23 20:51:40.213036; Progress: 93%; Samples: 60
2021-06-23 20:51:40.223538; Progress: 95%; Samples: 61
2021-06-23 20:51:40.234345; Progress: 96%; Samples: 62
2021-06-23 20:51:40.248133; Progress: 98%; Samples: 63
2021-06-23 20:51:40.253042; Progress: 100%; Samples: 64
```

In [39]:
```
post_samples.summary()
```

```
Directory: posterior_samples/
Samples: 129
Walkers: 8
Chunk size: 16
Variables: ['rate', 'mean', 'var']
Constants: {}
Duplicates: {}
Files:
├── master.hdf5 : size=50066B
├── subfile-000.hdf5 : size=6289B, samples=16
├── subfile-001.hdf5 : size=6289B, samples=16
├── subfile-002.hdf5 : size=6289B, samples=16
├── subfile-003.hdf5 : size=6289B, samples=16
├── subfile-004.hdf5 : size=6289B, samples=16
├── subfile-005.hdf5 : size=6289B, samples=16
├── subfile-006.hdf5 : size=6289B, samples=16
├── subfile-007.hdf5 : size=6289B, samples=16
└── subfile-008.hdf5 : size=6289B, samples=1
```

Markov Chain summary

In [40]:
```
assert numpy.count_nonzero(log_prior_samples) == 0
parameter_samples = post_samples.get_params(slice(-10, None))

pprint.pprint(parameter_samples)
```

```
{'mean': array([[-0.90585548, -0.47634511, -0.08349078, ...,  0.0309
0422,
         0.16075682,  0.085209  ],
       [-0.90585548, -0.54659635, -0.09629298, ..., -0.00495507,
         0.04356078,  0.07100963],
       [-0.90585548, -0.54659635, -0.09629298, ..., -0.00495507,
```

```
         0.04356078,  0.35386423],
       ...,
       [-0.68737115, -0.51466589, -0.19189298, ..., -0.36858484,
         0.19775752,  0.10742515],
       [-0.68737115, -0.51466589, -0.282545  , ..., -0.36858484,
         0.17989723,  0.12858528],
       [-0.90402287, -0.50029903, -0.282545  , ..., -0.36858484,
         0.47150653,  0.32862788]]),
 'rate': array([[0.45202961, 0.76466029, 1.73553457, ..., 0.52365148
, 0.76755005,
        0.48953484],
       [0.45202961, 0.76203462, 2.28796798, ..., 0.5531211 , 0.71385
709,
        0.56601435],
       [0.45202961, 0.76203462, 2.28796798, ..., 0.5531211 , 0.71385
709,
        0.4762399 ],
       ...,
       [0.48302106, 0.74578696, 1.54940618, ..., 0.5239495 , 0.70503
722,
        0.53329652],
       [0.48302106, 0.74578696, 1.11005863, ..., 0.5239495 , 0.67108
11 ,
        0.57352635],
       [0.45520551, 0.71493337, 1.11005863, ..., 0.5239495 , 0.75016
724,
        0.59347421]]),
 'var': array([[1.80031009, 0.66221174, 0.92436039, ..., 1.37719113,
0.93834297,
        1.09256   ],
       [1.80031009, 0.64110624, 1.003853  , ..., 1.26119862, 0.92671
545,
        1.03601764],
       [1.80031009, 0.64110624, 1.003853  , ..., 1.26119862, 0.92671
545,
        1.21688135],
       ...,
       [1.35557547, 0.67138714, 0.85260541, ..., 1.58152753, 0.99724
377,
```

```
        1.21584593],
       [1.35557547, 0.67138714, 0.88642857, ..., 1.58152753, 1.04046
523,
        1.16463892],
       [1.20201524, 0.70587178, 0.88642857, ..., 1.58152753, 1.09423
299,
        0.99689859]])}
```

Chain runs

In [41]:

```
fig_chains, axes_chains = post_samples.plot_chains()

fig_chains.show()
```

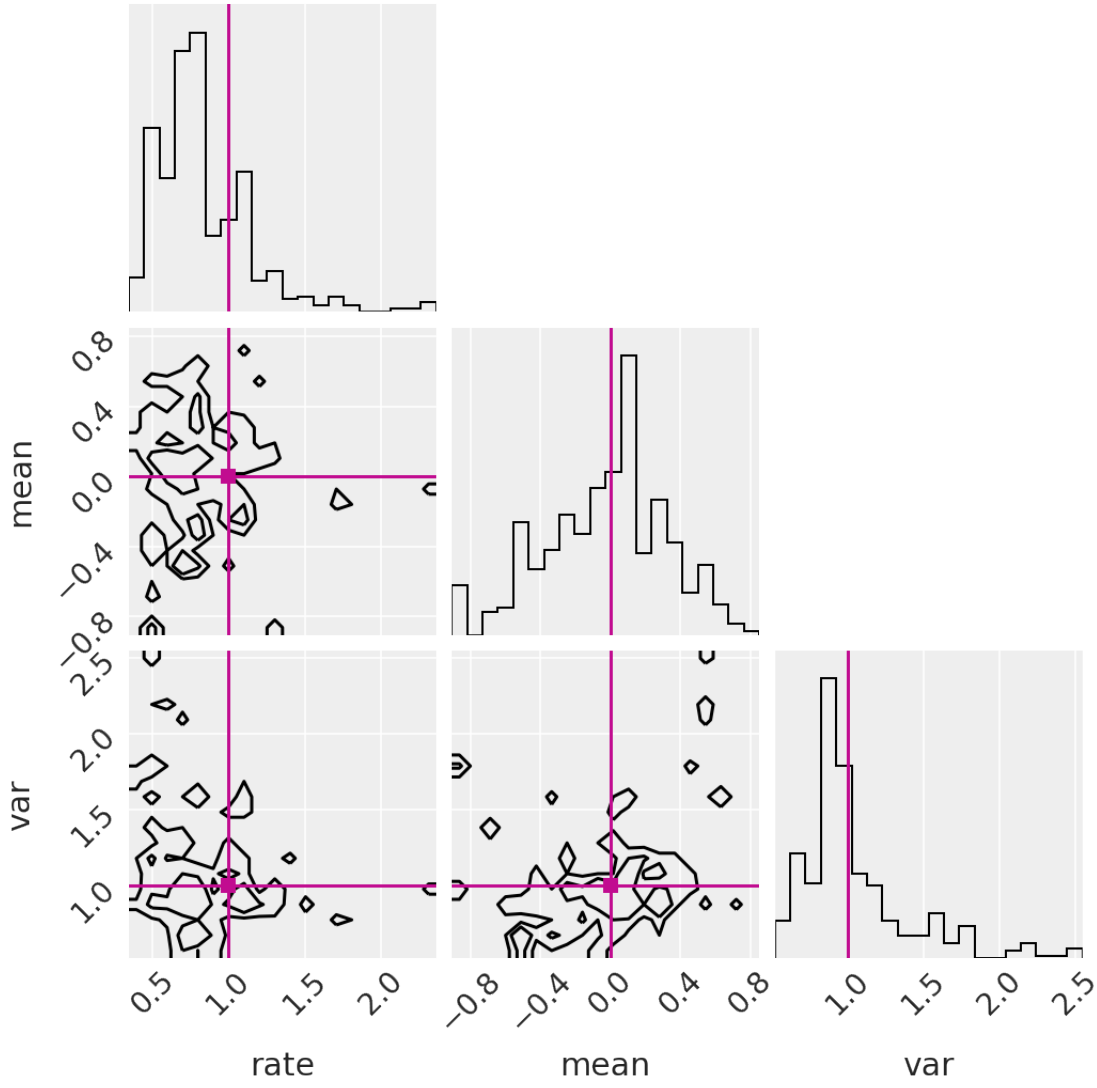

log(prob) + const
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
posterior
prior
rate
3.0
2.5
2.0
1.5
1.0
0.5
mean
1.0
0.5
0.0
−0.5
−1.0
var
3.0
2.5
2.0
1.5
1.0
0.5
0
50
100
sample #

"Corner" Plot of the joint and marginal posteriors

In [42]:

```
from pop_models.posterior import H5CleanedPosteriorSamples

post_samples_cleaned = H5CleanedPosteriorSamples.from_raw_manual(
    output_samples_filename,
    post_samples,
    burnin=int(0.6*n_samples), thinning=1,
)


fig_post = post_samples_cleaned.plot_corner(
    levels=[0.50, 0.90], truths=parameters_true,
)
fig_post.show()
```

```
/opt/conda/lib/python3.7/site-packages/corner/core.py:104: UserWarni
ng:

This figure was using constrained_layout, but that is incompatible w
ith subplots_adjust and/or tight_layout; disabling constrained_layou
t.
```



Gravitational waves - LIGO and PyCBC

Credit: Wikipedia

PyCBC - Gravitational Waves software

https://github.com/gwastro/PyCBC-Tutorials (https://github.com/gwastro/PyCBC-Tutorials)

Tutorial 1: Accesing Gravitational waves data:

https://colab.research.google.com/github/gwastro/pycbc-tutorials/blob/master/tutorial/1_CatalogData.ipynb (https://colab.research.google.com/github/gwastro/pycbc-tutorials/blob/master/tutorial/1_CatalogData.ipynb)

Tutorial 2: Data visualization and basic signal processing:

https://colab.research.google.com/github/gwastro/pycbc-tutorials/blob/master/tutorial/2_VisualizationSignalProcessing.ipynb (https://colab.research.google.com/github/gwastro/pycbc-tutorials/blob/master/tutorial/2_VisualizationSignalProcessing.ipynb)

📌 Lab session #4

Hands-on for this course (in Colab)

Corner plots of GW150914 (https://colab.research.google.com/github/gwastro/pycbc-tutorials/blob/master/tutorial/inference_5_results_io/IntroTo