Create Sim2L - Pendulum Motion Solution

nanoHUB jupyter Submit a ticket Terminate Session

```
clarksm@nanohub_1911864_32:~/Gateways21Tutorial$ cd gateways21tutorial
clarksm@nanohub_1911864_32:~/Gateways21Tutorial/gateways21tutorial$ git checkout solution
Branch solution set up to track remote branch solution from origin.
Switched to a new branch 'solution'
clarksm@nanohub_1911864_32:~/Gateways21Tutorial/gateways21tutorial$
```

nanoHUB jupyter Submit a ticket Terminate Session

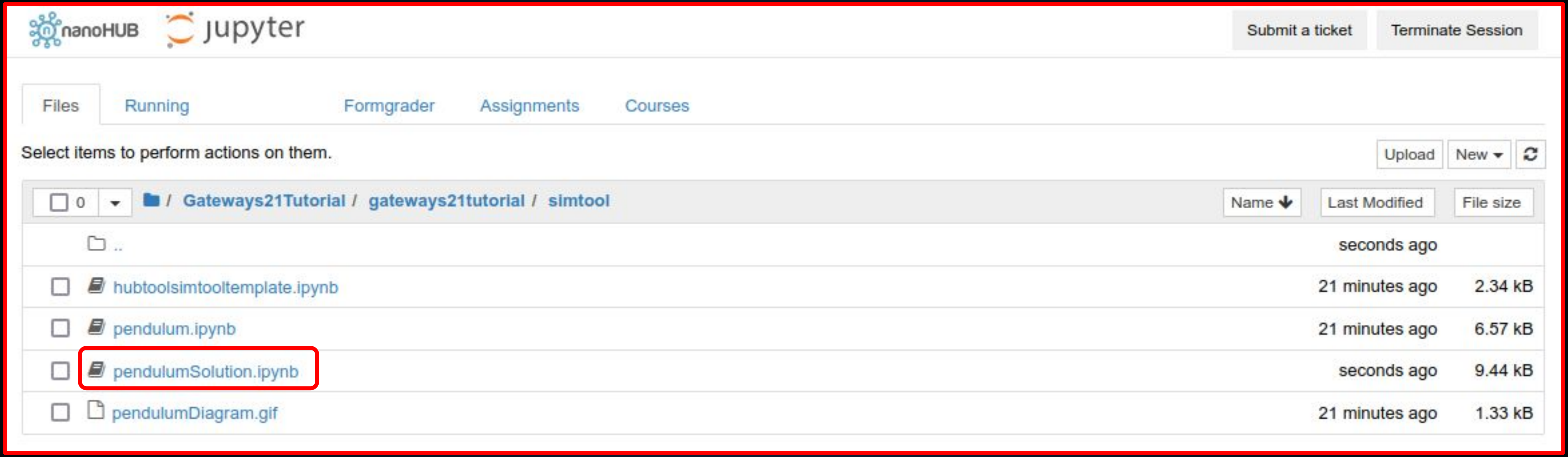
Files Running Formgrader Assignments Courses

Select items to perform actions on them. Upload New

0 / Gateways21Tutorial / gateways21tutorial / simtool

Name	Last Modified	File size
..	seconds ago	
hubtoolsimtooltemplate.ipynb	4 minutes ago	2.34 kB
pendulum.ipynb	4 minutes ago	6.57 kB
pendulumDiagram.gif	4 minutes ago	1.33 kB

Create Sim2L - Pendulum Motion



Create Sim2L - Sim2L notebook

Input Conditions and Parameters

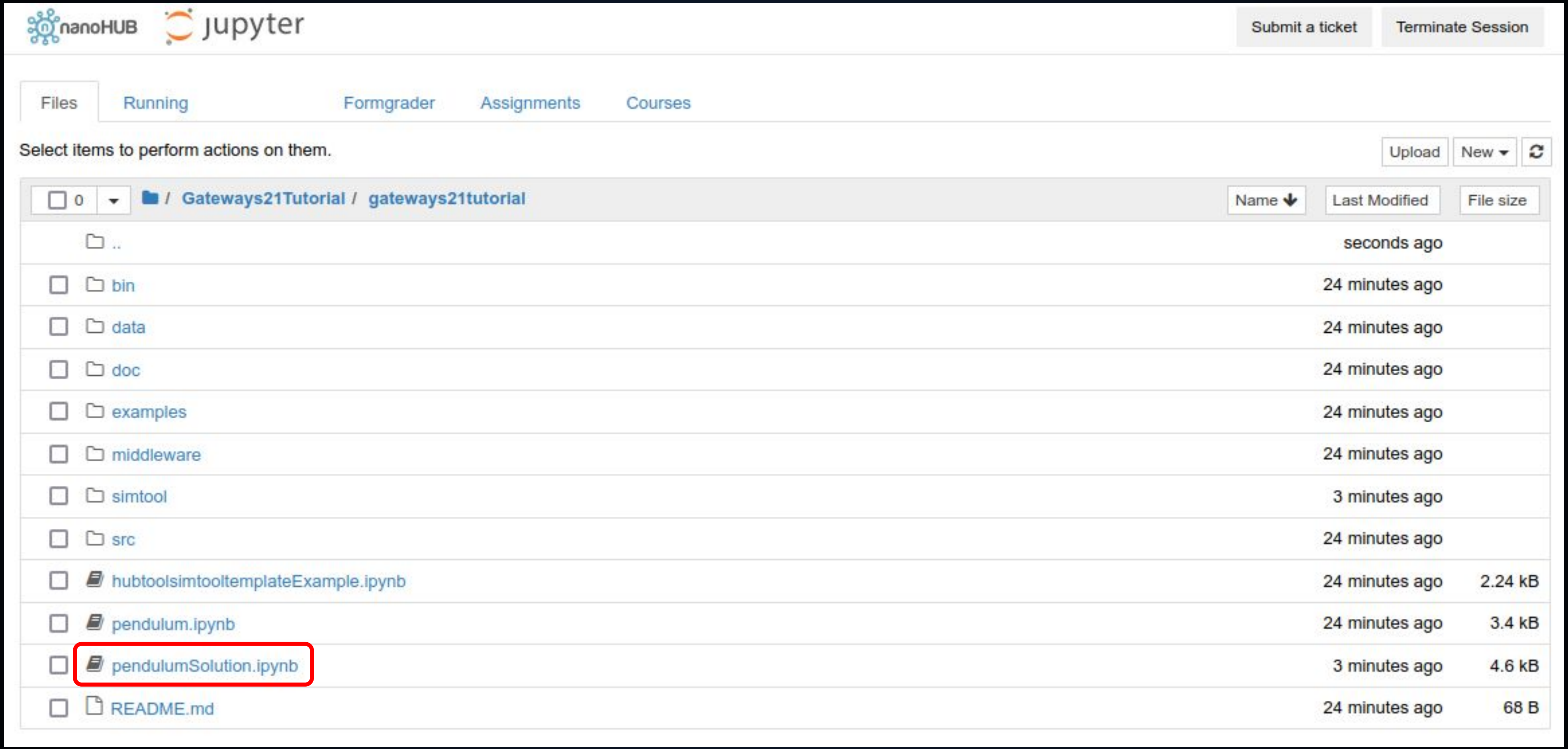
Based on the pedulum model described above and your choice of numerical method, choose a suitable set of conditions and parameters for the Sim2L.

In [ ]:

```
%%yaml INPUTS

length:
    type: Number
    description: Pendulum length
    value: 1
    units: m
    min: 0.1
    max: 100
mass:
    type: Number
    description: Pendulum mass
    value: 1
    units: kg
    min: 0.001
damp:
    type: Number
    description: Damping coefficient
    value: 0.1
    units: kg rad/s
    min: 0.0
initialPosition:
    type: Number
    description: Initial location
    value: 0.
    units: rad
initialVelocity:
    type: Number
    description: Initial velocity
    value: 1.
    units: rad/s
    min: -100
    max: +100
timeHorizon:
    type: Number
    description: Simulation duration
    units: s
    value: 100.
    min: 0
    max: 1000
nTimePoints:
    type: Integer
    description: Number of points to record simulation results
    value: 100
    min: 2
```

Create Sim2L - Sim2L notebook



Create Sim2L - workflow notebook

Assign input parameter values

All input parameters should have default values. If you want to run non-default values set them here.

In [ ]:
```
inputs.damp.value = 0.05
inputs.length.value = "100 cm"
inputs.mass.value = 0.1
inputs.initialPosition.value = "45 degree"
inputs.initialVelocity.value = "1"
inputs.timeHorizon.value = 30
inputs.nTimePoints.value = 150
```

Run Sim2L

Run the Sim2L and report the summary. The summary will indicate if any problems occurred during the run.

In [ ]:
```
run = Run(simToolLocation,inputs)
run.getResultSummary()
```

Extract results

Extract any results from the run needed to plot position and velocity versus time plots.

In [ ]:
```
success = run.read('success')
terminationMessage = run.read('terminationMessage')
time = run.read('time')
position = run.read('position')
velocity = run.read('velocity')

print(terminationMessage)
```

Plot results

Using the plot library of your choosing plot position and velocity versus time for the pendulum. If you are not familiar with any ploting libraries *matplotlib* is simple

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(time,position,"b--")
plt.xlabel("Time (" + str(outputs.time.units) + ")")
plt.ylabel("Position (" + str(outputs.position.units) + ")")
plt.show()

plt.plot(time,velocity,"r--")
plt.xlabel("Time (" + str(outputs.time.units) + ")")
plt.ylabel("Velocity (" + str(outputs.velocity.units) + ")")
plt.show()
```

Create Sim2L - workflow notebook

Extract results

Extract any results from the run needed to plot position and velocity versus time plots.

In [7]:
```
success = run.read('success')
terminationMessage = run.read('terminationMessage')
time = run.read('time')
position = run.read('position')
velocity = run.read('velocity')

print(terminationMessage)
```

```
The solver successfully reached the end of the integration interval.
```

Plot results

Using the plot library of your choosing plot position and velocity versus time for the pendulum. If you are not familiar with any ploting libraries *matplotlib* is simple

In [8]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(time,position,"b--")
plt.xlabel("Time (" + str(outputs.time.units) + ")")
plt.ylabel("Position (" + str(outputs.position.units) + ")")
plt.show()

plt.plot(time,velocity,"r--")
plt.xlabel("Time (" + str(outputs.time.units) + ")")
plt.ylabel("Velocity (" + str(outputs.velocity.units) + ")")
plt.show()
```