PyHEADTAIL examples

All our thanks to
D. Amorim, Kevin Li, Lotta Mether and Michael Schenk
CERN BE/ABP-HSC

and

Martial Fol and Saïd Errghioui (AZIMUTEC),
Marie Gauthier, Coline Morin, Stephanie Vandergooten et Lise Ribet (ESI)

Reference: http://kli.web.cern.ch/kli/

Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Goals

- Run macroparticle simulations with a state-of-the-art open source tracking code (PyHEADTAIL)
- Simulate several case-studies related to the course
- Play with beam parameters and observe the impact on the longitudinal beam dynamics

Plan

- 1h00 towards the end of the course in the computer room
- A virtual box with examples is prepared in the computer room
- Following popular demand, it is also possible to set up your own simulation environment on your own PC.
- The detailed procedure and examples are on the Indico site.
- Note: we should expect incompatibilities linked to open source codes!

Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

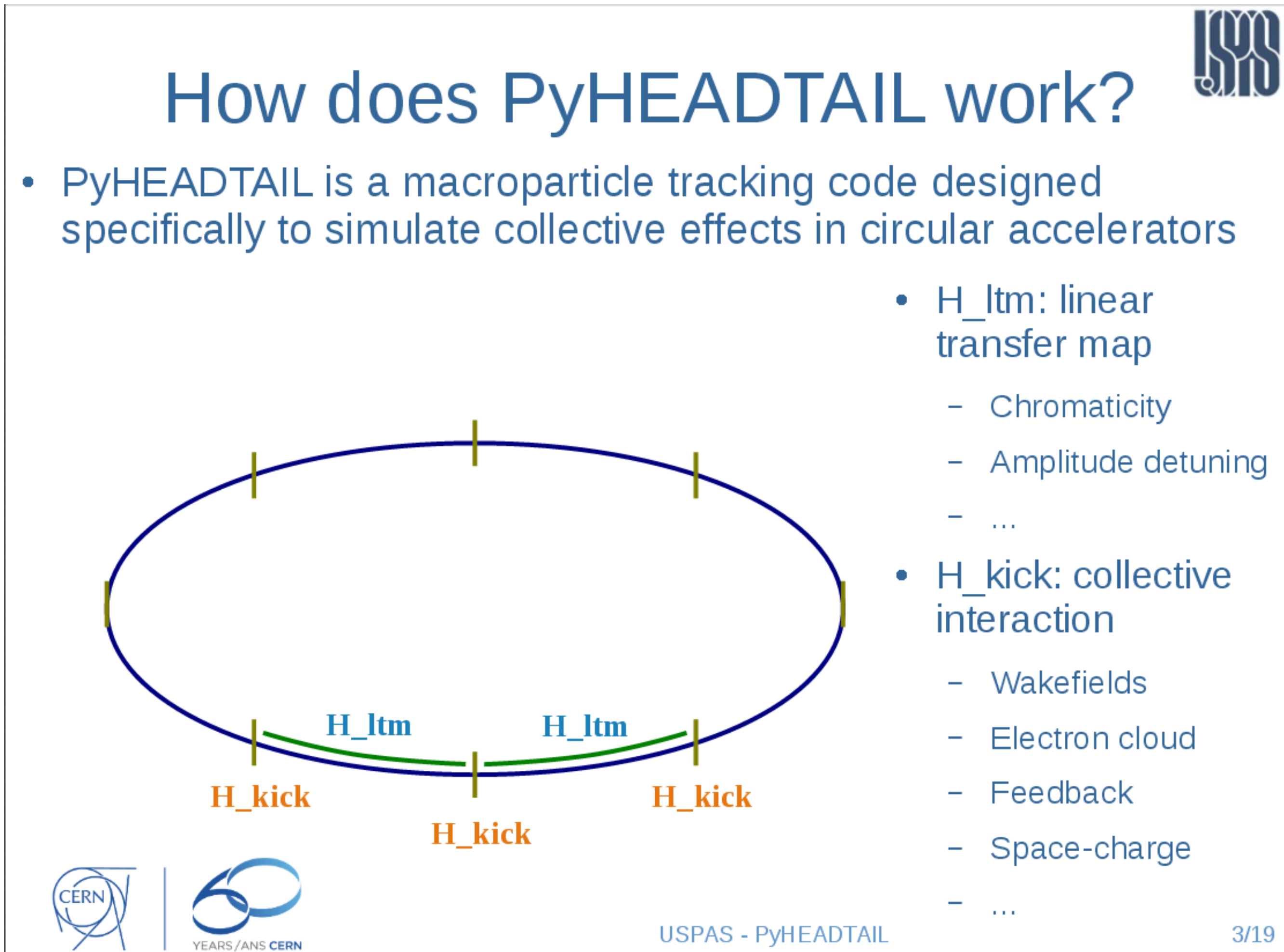
Introduction to PyHEADTAIL

- Open source macroparticle tracking code developed at CERN:
- Download link: https://github.com/PyCOMPLETE/PyHEADTAIL
- Primary use: tracking simulation of collective effects in synchrotron accelerators
  - Transverse and longitudinal beam dynamics (with feedback)
  - Electron/ion cloud
  - Impedances
  - Space charge
- Reference:
  Introduction to PyHEADTAIL: USPAS course by Kevin Li et al (2015)

- Not the only code of his kind!
  - BLOND: longitudinal dynamics simulation code https://blond.web.cern.ch/
  - HEADTAIL: the father of PyHEADTAIL! G. Rumolo et al (reference)
  - elegant: 6D tracking code developed at Argonne National Lab (link)
  - mbtrack and sbtrack: R. Nagaoka et al "Studies of Collective Effects in SOLEIL and DIAMOND Using the Multiparticle Tracking Codes sbtrack and mbtrack", PAC09, Vancouver, May 2009.
  - ORBIT (http://web.ornl.gov/~jzh/JHolmes/ORBIT.html), pyORBIT (http://sourceforge.net/projects/py-orbit/)
  - And so many others!

introduction to PyHEADTAIL

Courtesy Kevin Li
USPAS 2015



introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015

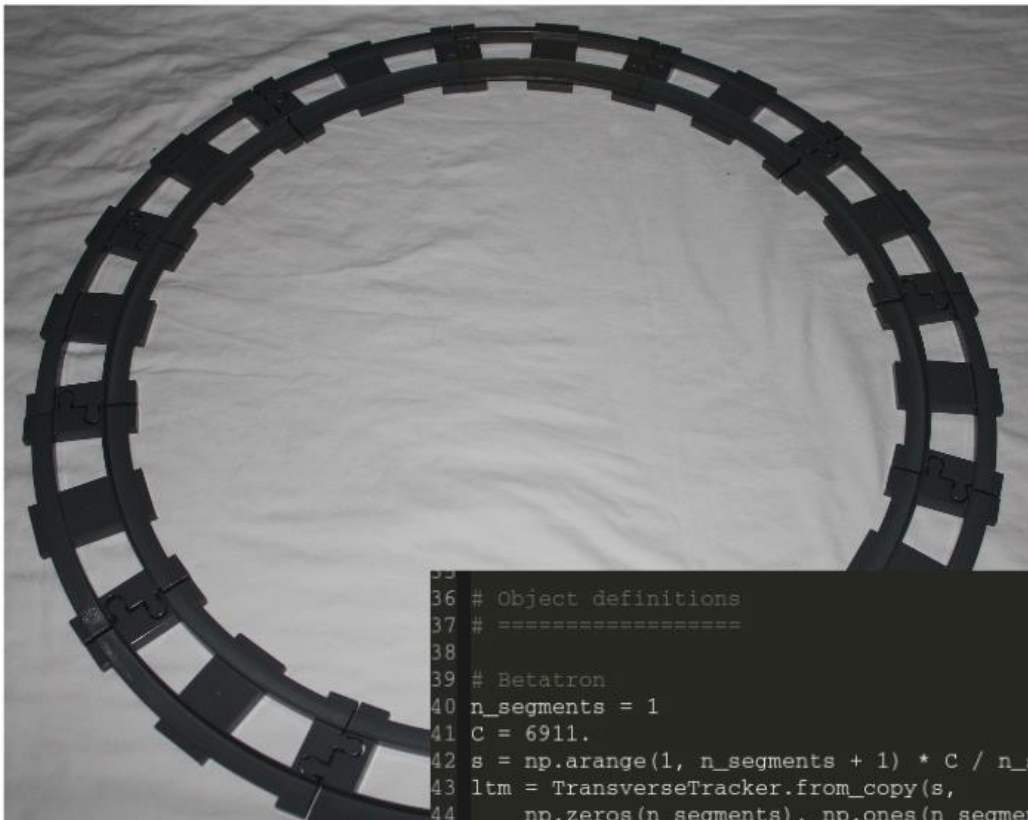
- Load Python packages and modules

```
from __future__ import division
import cProfile, itertools, sys, time, timeit

from scipy.constants import c, e, m_p

from cobra_functions import stats, random
from beams.beams import *
from monitors.monitors import *
from spacecharge.spacecharge import *
from trackers.transverse_tracker import *
from trackers.longitudinal_tracker import *
```

introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015



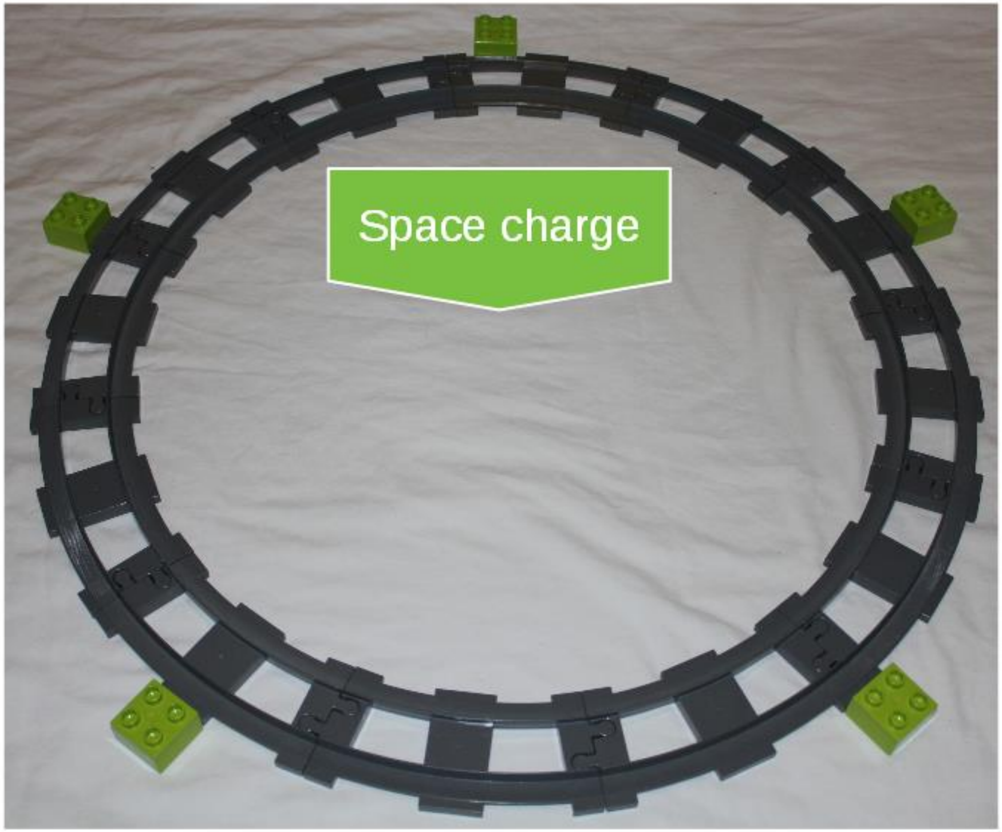
```
# Object definitions
# ==================

# Betatron
n_segments = 1
C = 6911.
s = np.arange(1, n_segments + 1) * C / n_segments
ltm = TransverseTracker.from_copy(s,
    np.zeros(n_segments), np.ones(n_segments) * beta_x, np.zeros(n_segments),
    np.zeros(n_segments), np.ones(n_segments) * beta_y, np.zeros(n_segments),
    Qx, 0, 0, Qy, 0, 0)
```

- Load Python packages and modules
- Build linear periodic transfer maps

introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015

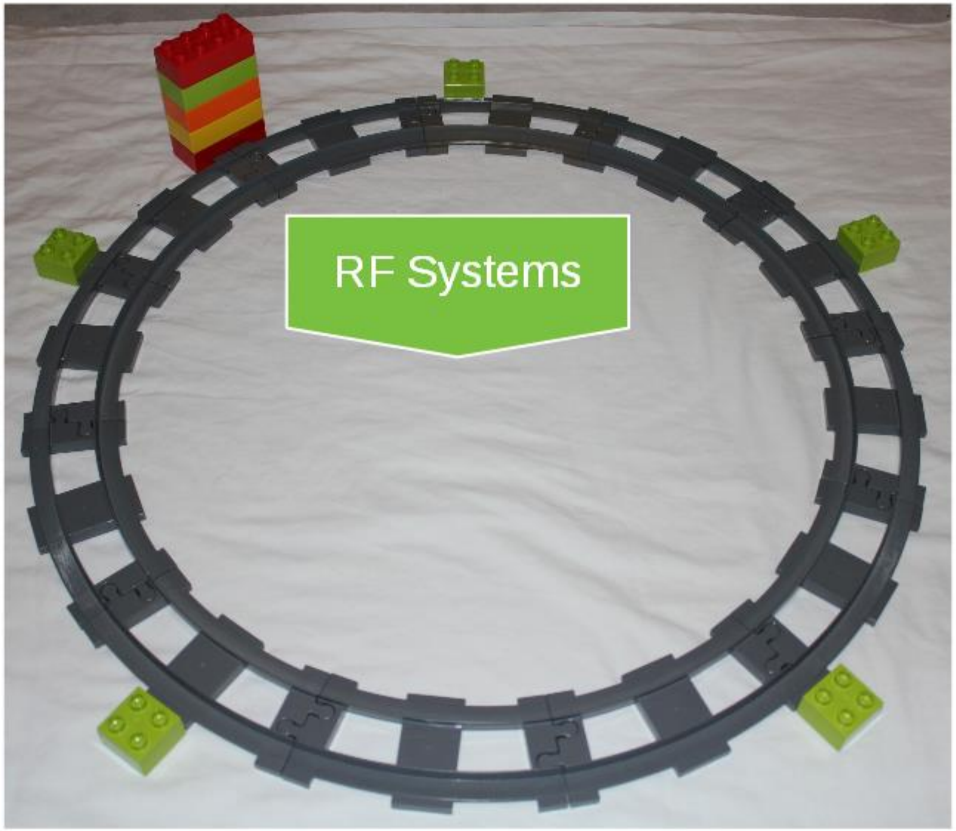


- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015

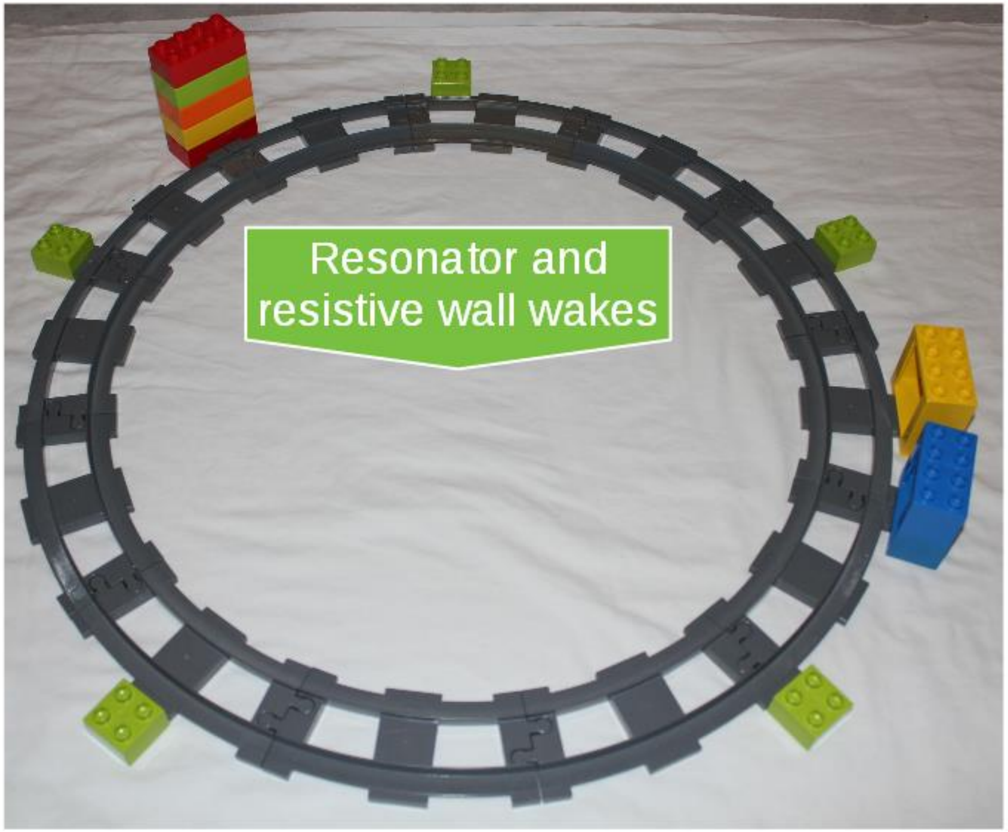


- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

introduction to PyHEADTAIL

Courtesy
Kevin Li
USPAS 2015

A real world example

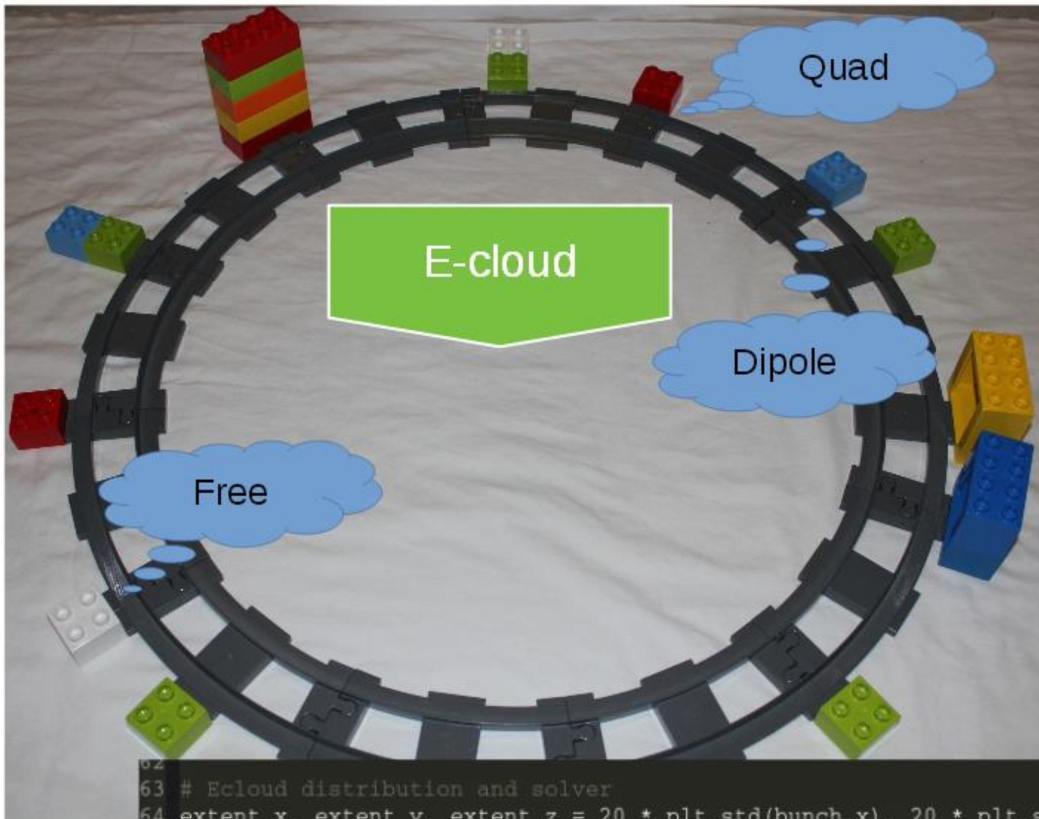


- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

introduction to PyHEADTAIL

Courtesy
Kevin Li
USPAS 2015

A real world example



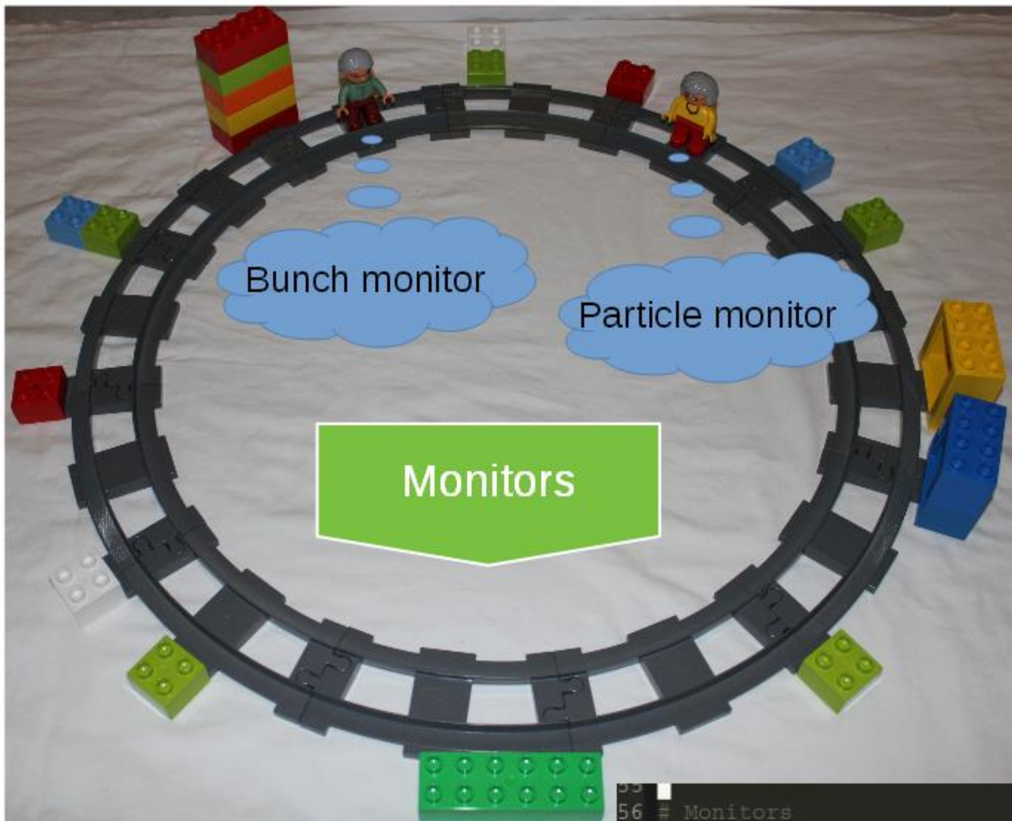
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

```
# Ecloud distribution and solver
extent_x, extent_y, extent_z = 20 * plt.std(bunch.x), 20 * plt.std(bunch.y), C / n_segments
cloud = Cloud(100000, 1e11, extent_x, extent_y, extent_z)
ecloud = SpaceCharge(cloud, 'cloud', extent_x, extent_y, 128, 128, slices)
```

introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015



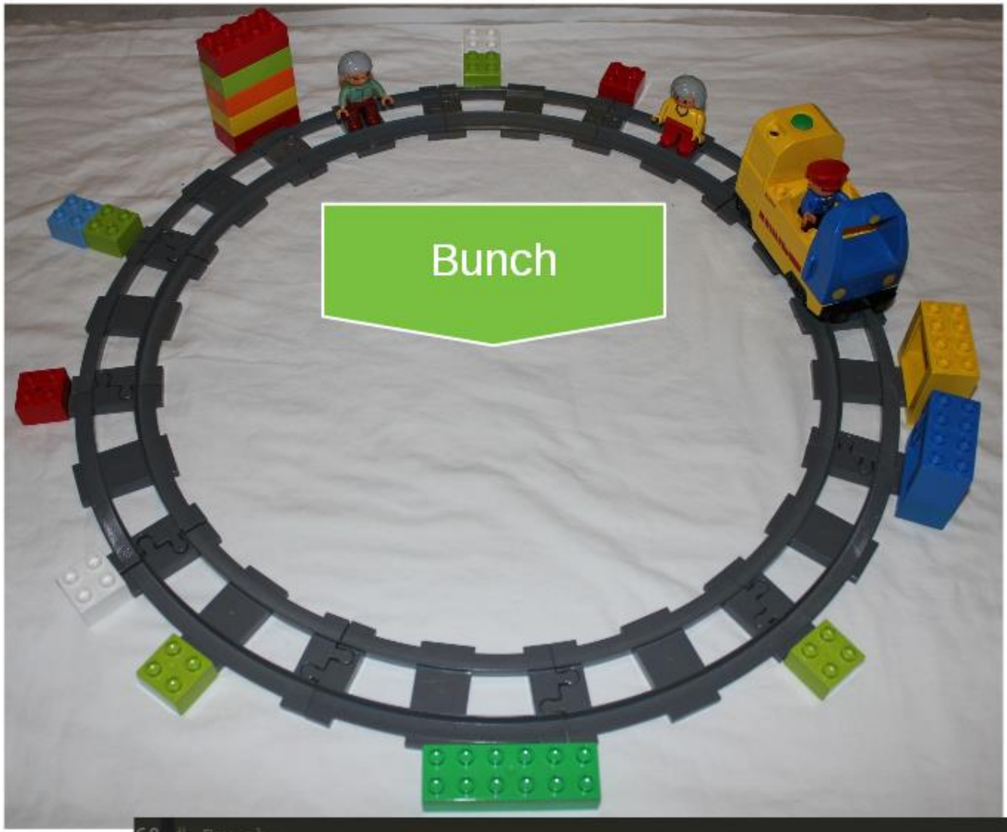
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

```
# Monitors
bunchmonitor = BunchMonitor('bunch', n_turns, slices)
particlemonitor = ParticleMonitor('particles', n_turns, slices)
```

introduction to PyHEADTAIL

A real world example

Courtesy
Kevin Li
USPAS 2015



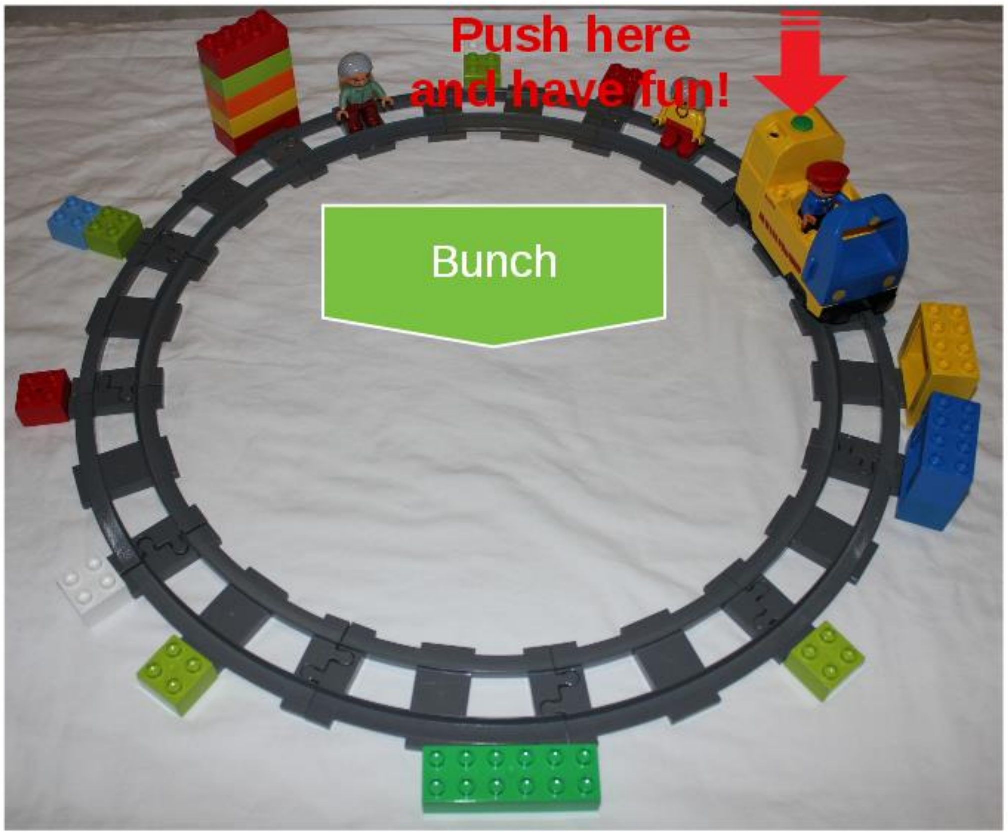
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements
- Place beam

```
60 # Bunch
61 bunch = Bunch(500000, e, gamma, 1.15e11, m_p, 0, beta_x, epsn_x, 0, beta_y, epsn_y, beta_z, sigma_z)
```

introduction to PyHEADTAIL

Courtesy
Kevin Li
USPAS 2015

A real world example



- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements
- Place beam

```
for i in range(n_turns):
    for m in map_:
        m.track(bunch)
```

In our case: only RF systems

JUAS

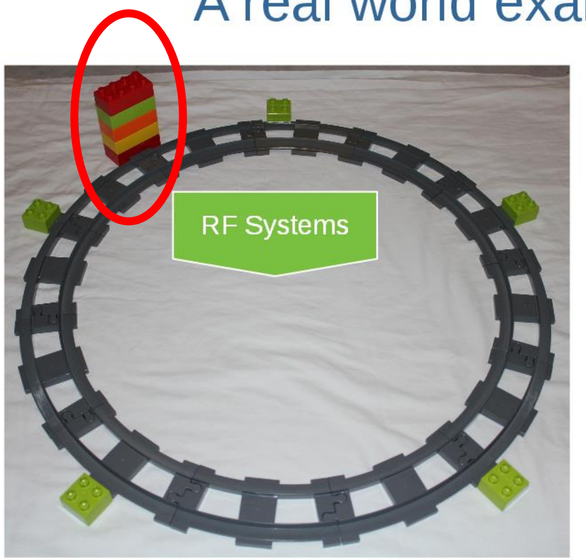
Tracking

- The motion of the particles can be tracked turn by turn using the recurrence relation (between turn n and turn $n+1$)

$$\Delta E_{n+1} = \Delta E_n + e\,\hat{V}_{RF}\left[\sin\phi_n - \sin\phi_s\right]$$

$$\phi_{n+1} = \phi_n - \frac{2\,\pi\,h\,\eta}{\beta_s^2\,E_s}\,\Delta E_{n+1}$$

A real world example



- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Structure of the ipython file

Import needed libraries

```
In [ ]: import sys
sys.path.append("../../../")
import numpy as np
from scipy.constants import m_p, c, e
from PyHEADTAIL.particles.particles import Particles
import PyHEADTAIL.particles.generators as generators
from PyHEADTAIL.trackers.transverse_tracking import TransverseMap
from PyHEADTAIL.trackers.simple_long_tracking import RFSystems, LinearMap
import PyHEADTAIL.cobra_functions.stats as st
import matplotlib.pyplot as plt
```

Set beam and machine parameters

Define RF system

Generate a matched distribution corresponding to these parameters

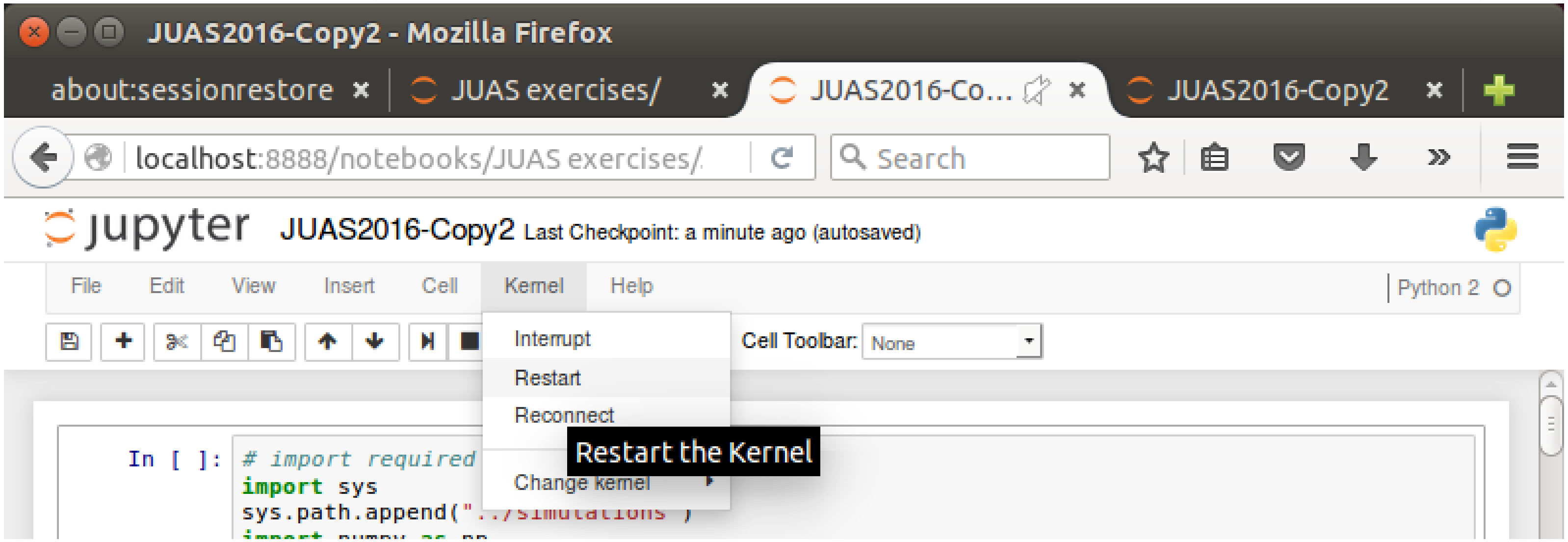
```
In [ ]: # general simulation parameters
n_particles = 1000
n_segments = 1

# machine parameters
circumference = 100*2*np.pi
inj_alpha_x = 0#-1.2
inj_alpha_y = 0#15
inj_beta_x = 16.#5.9 # in [m]
inj_beta_y = 16.#5.7 # in [m]
Qx = 6.25
Qy = 6.25
gamma_tr = 6.1
alpha_c_array = [gamma_tr**-2]
V_rf = 20e3 # in [V]
harmonic = 8
phi_offset = 0 # measured from aligned focussing phase (0 or pi)
#pipe_radius = 5e-2
Bdot=0 # in T/s
bending_radius=70 # in m

# beam parameters
Ekin = 1.4e9 # in [eV]
intensity = 8e12
epsn_x = 5e-6 # in [m*rad]
epsn_y = 5e-6 # in [m*rad]
#epsn_z = 1. # 4pi*sig_z*sig_dp (*p0/e) in [eVs]

# calculations
gamma = 1 + e * Ekin / (m_p * c**2)
beta = np.sqrt(1 - gamma**-2)
print('beta: ' + str(beta))
eta = alpha_c_array[0] - gamma**-2
print('eta: ' + str(eta))
if eta < 0:
    phi_offset = np.pi - phi_offset
Etot = gamma * m_p * c**2 / e
p0 = np.sqrt(gamma**2 - 1) * m_p * c
Qs = np.sqrt(np.abs(eta) * V_rf / (2 * np.pi * beta**2 * Etot))
print('Qs: ' + str(Qs))
beta_z = np.abs(eta) * circumference / (2 * np.pi * Qs)
print('beta_z: ' + str(beta_z))
turn_period = circumference / (beta * c)
p_increment_0= e*bending_radius*Bdot*turn_period
sigma_z_0= 230e-9/4*beta*c/10

# BETATRON
# Loop on number of segments and create the TransverseSegmentMap
# for each segment.
s = np.arange(0, n_segments + 1) * circumference / n_segments
alpha_x = inj_alpha_x * np.ones(n_segments)
beta_x  = inj_beta_x * np.ones(n_segments)
D_x     = np.zeros(n_segments)
alpha_y = inj_alpha_y * np.ones(n_segments)
beta_y  = inj_beta_y * np.ones(n_segments)
D_y     = np.zeros(n_segments)

# Define RF systems
rfsystems = RFSystems(circumference, [harmonic], [V_rf], [phi_offset],
                      alpha_c_array, gamma, p_increment=p_increment_0)
# Generate the particle distribution
bunch = generators.ParticleGenerator(macroparticlenumber=n_particles,
                                     intensity=intensity, charge=e, mass=m_p,
                                     circumference=circumference, gamma=gamma,
                                     distribution_x=generators.gaussian2D(epsn_x), alpha_x=inj
                                     distribution_y=generators.gaussian2D(epsn_y), alpha_y=inj
                                     distribution_z=generators.RF_bucket_distribution(rfsystem
                                     ).generate()
```

Loop over number of turns

```
In [ ]: # plot phase space
plt.close()
plt.ion()
fig = plt.figure(1)
for i in np.arange(0, 50, 1):
    # track the particles
    rfsystems.track(bunch)                                   <- track
    if i%10 == 0:
        # monitor the particles
        bucket = rfsystems.get_bucket(gamma=bunch.gamma)     <- monitor

        # plot the RF bucket envelope
        z = np.linspace(*bucket.interval, num=100)
        dp = np.linspace(-0.005, 0.005, num=100)
        ZZ, DPP = np.meshgrid(z, dp)
        HH = bucket.hamiltonian(ZZ, DPP)
        plt.contour(ZZ, DPP, HH, levels=[0], colors='magenta')

        # plot the particles in phase space
        plt.plot(bunch.z, bunch.dp, 'o')                     <- plot
        plt.xlabel('z in m')
        plt.ylabel('Delta p/p')
        plt.show()
        plt.pause(0.1)
        plt.cla()
plt.ioff()
```

ipython cheat sheet

- To restart the kernel at any point: “Kernel” → “Restart”



- To only interrupt the kernel: “Kernel” → “Interrupt”
- Shift-enter: execute current cell and move to next cell
- Ctrl-enter: execute current cell and stay on current cell

Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Tracking example: injection energy

- nTOF bunch in the CERN PS (near transition)

Average machine radius: R [m]	100
Bending dipole radius: ρ [m]	70
$\dot{B}$ [T/s]	0
$\hat{V}_{RF}$ [kV]	200
h	8
α_p	0.027
Longitudinal (total) emittance: ε_L [eVs]	2
Number of protons/bunch: N_b [1E10 p/b]	800
Norm. rms. transverse emittance: $\varepsilon^*_{x,y}$ [μm]	5
Trans. average betatron function: $\beta_{x,y}$ [m]	16
Beam pipe [cm × cm]	3.5 × 7
Trans. tunes: $Q_{x,y}$	6.25

20 kV at injection

=> $\gamma_t \approx 6.1$

Tracking example: injection energy

JUAS
Joint Universities Accelerator School

- nTOF bunch in the CERN PS (near transition)

Average machine radius: R [m]	100
Bending dipole radius: ρ [m]	70
$\dot{B}$ [T/s]	0
$\hat{V}_{RF}$ [kV]	200
h	8
α_p	0.027
Longitudinal (total) emittance: ε_L [eVs]	2
Number of protons/bunch: N_b [1E10 p/b]	800
Norm. rms. transverse emittance: $\varepsilon^*_{x,y}$ [µm]	5
Trans. average betatron function: $\beta_{x,y}$ [m]	16
Beam pipe [cm × cm]	3.5 × 7
Trans. tunes: $Q_{x,y}$	6.25

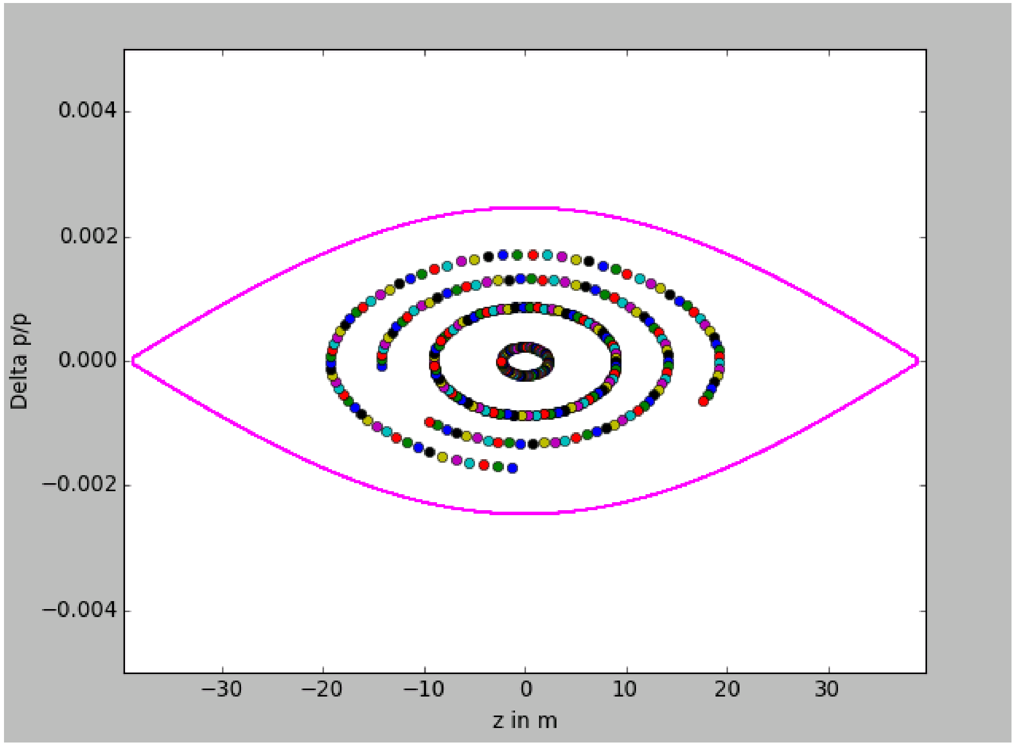
20 kV at injection

=> $\gamma_t \approx 6.1$

```
Ekin = 1.4e9 # in [eV]
circumference = 100*2*np.pi
bending_radius=70 # in m
Bdot=0 # in T/s
V_rf = 20e3 # in [V]
harmonic = 8
gamma_tr = 6.1
sigma_z_0= 230e-9/4*beta*c
```

Let's run that example

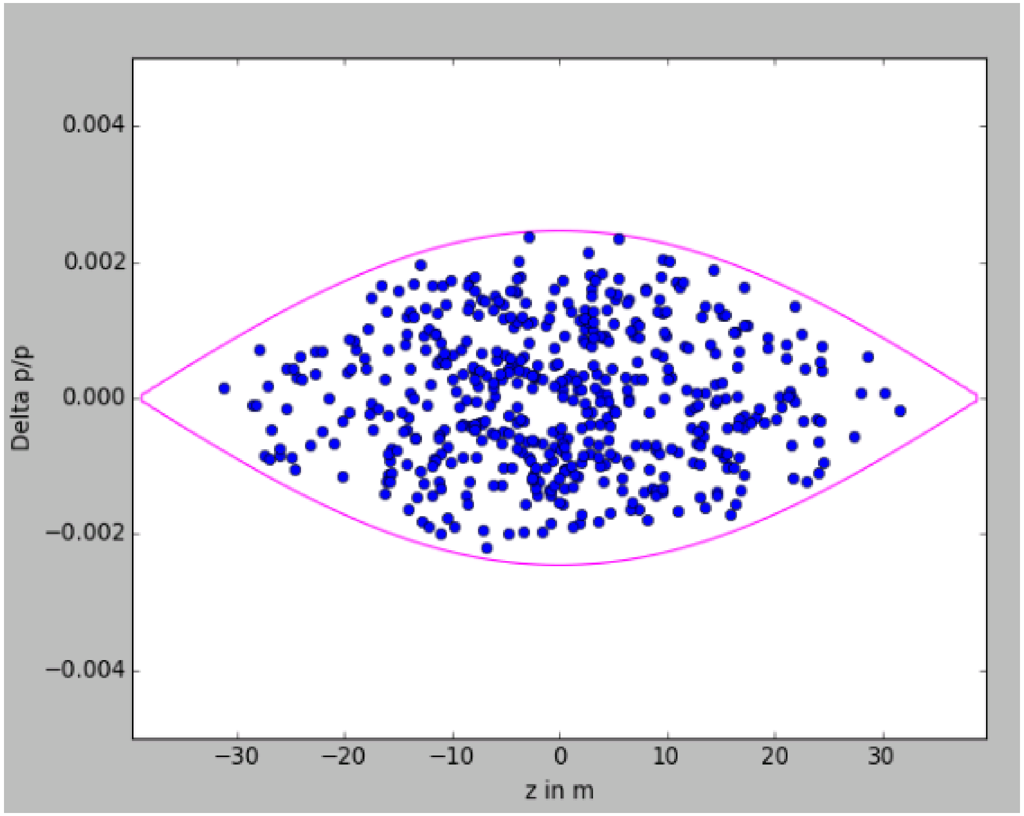
- Use a small number of particles (n_particles~ 10) to observe the synchrotron motion at large and small amplitudes



→ Can you estimate the synchrotron tune Qs? Is it consistent with the course?

Synchrotron motion

- Use a larger number of particles (~500) and reset the graph every time ("plt.cla()" uncommented)



→ Are the particles rotating in the correct direction?
→ Plot the phase space as in the course (in ϕ[degrees], ΔE [MeV])
→ Is the maximum energy consistent with the course?

Top energy

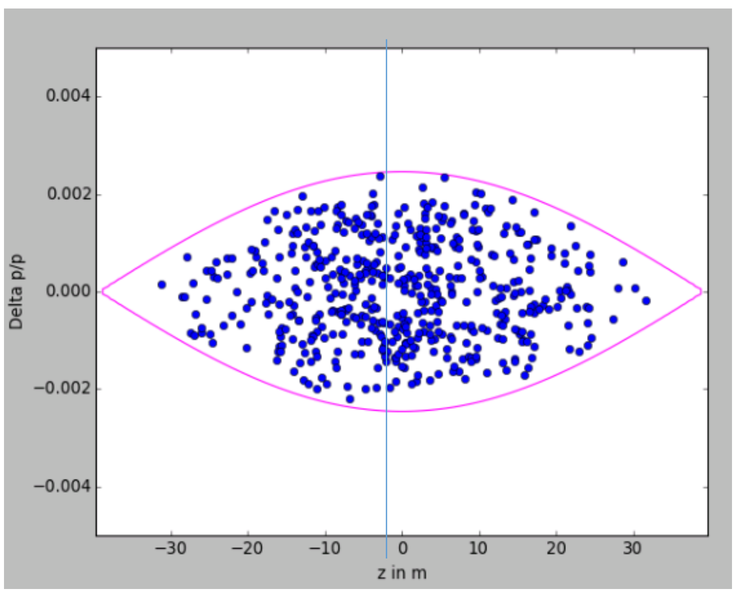
- What parameters should be changed for top energy (still no acceleration and h=8)?
- What changes in the plot compared to injection energy?

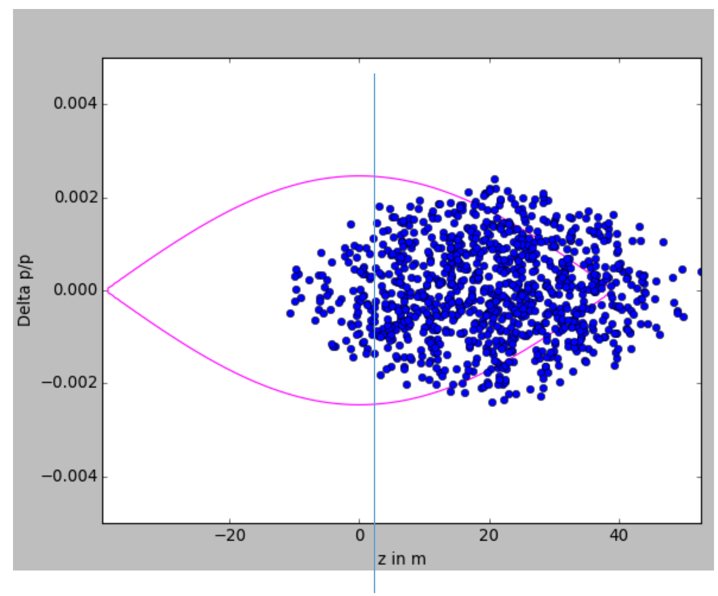
Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Impact of phase mismatch

- Remove the division by 10 in sigma_z_0
- Generate ~1000 particles
- Add e.g. 5 or 20 m to all particles: `bunch.z =bunch.z+5`
- Track particles for ~5000 turns
- Plot the distribution every 10 or 100 turns: `if i%100 == 0:`

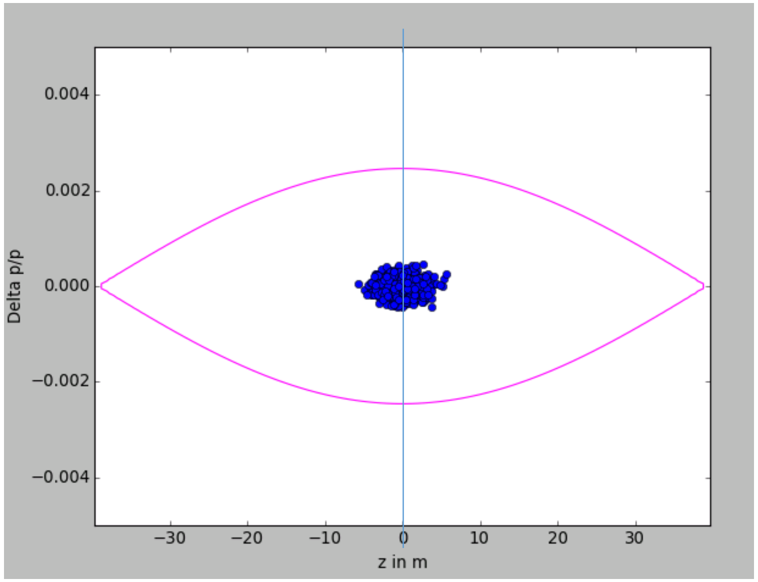


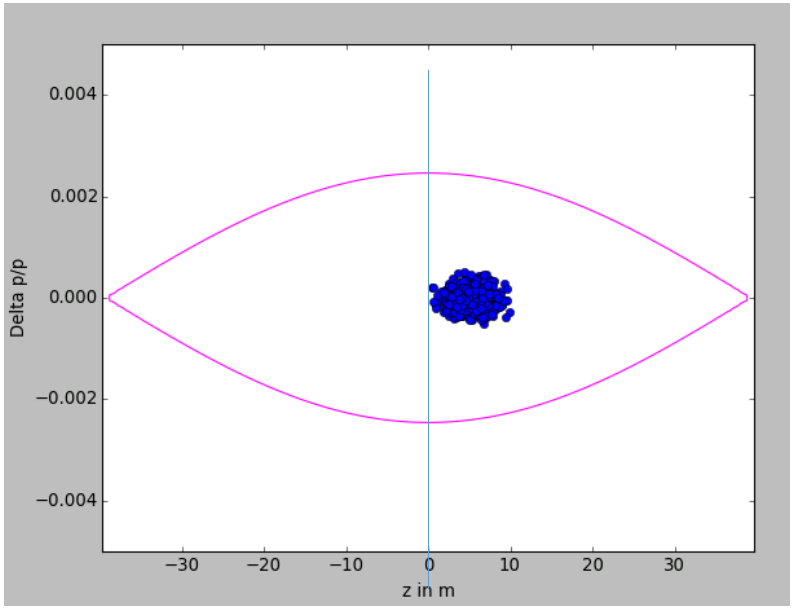


→ What will happen?

Impact of phase mismatch

- Use a smaller bunch length (e.g. divide by 10)
- Generate ~1000 particles
- Add e.g. 5m to all particles: `bunch.z =bunch.z+5`
- Track particles for ~5000 turns
- Plot the distribution every 100 turns: `if i%100 == 0:`

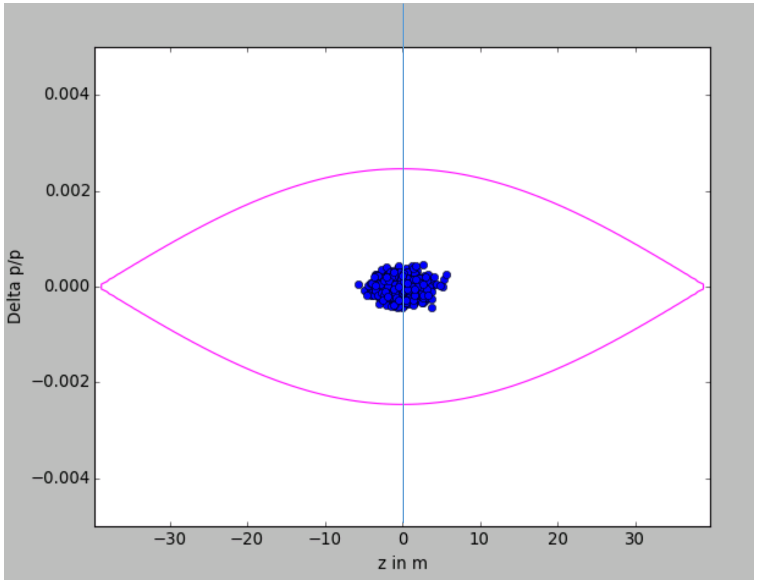


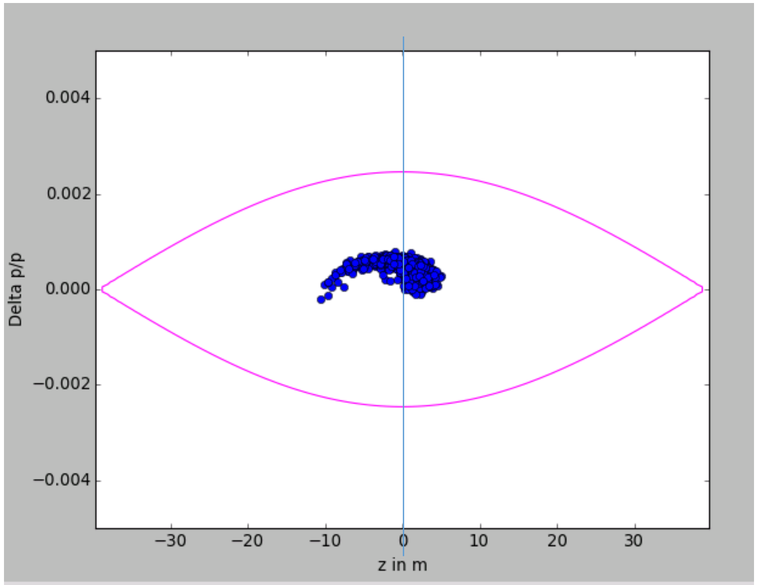


→ What will happen?

Impact of phase mismatch

- Use a smaller bunch length (e.g. divide by 10)
- Use ~1000 particles
- Add e.g. 5m to all particles: `bunch.z =bunch.z+5`
- Track particles for ~5000 turns
- Plot the distribution every 100 turns: `if i%100 == 0:`

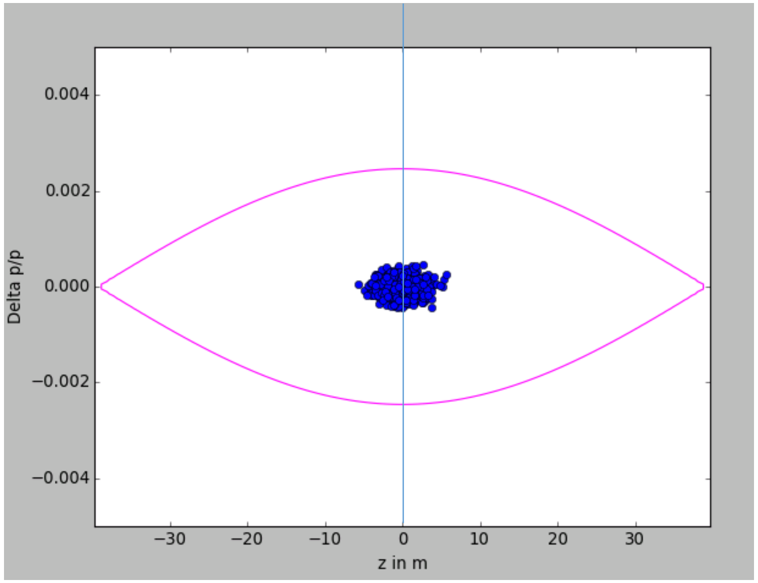


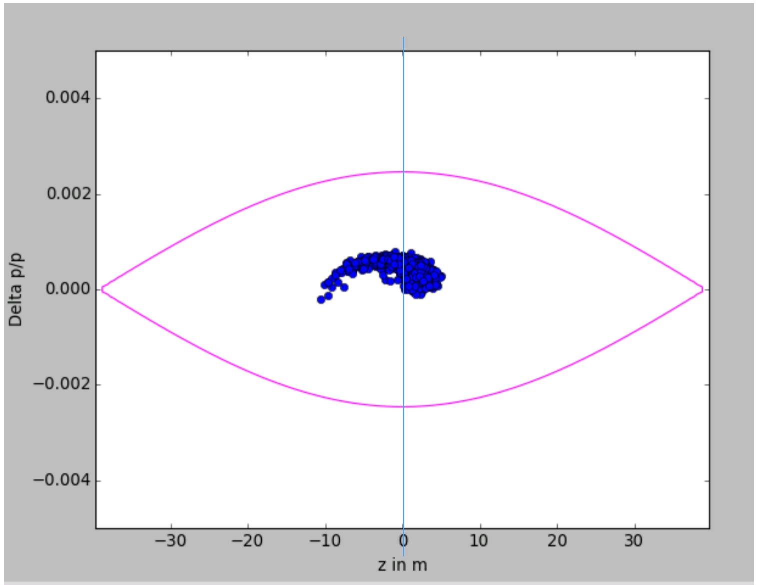


→ Filamentation and eventually?

Impact of phase mismatch

- Use a smaller bunch length (e.g. divide by 10)
- Use ~1000 particles
- Add e.g. 5m to all particles:
- Track particles for ~500000 turns
- Plot the distribution every 1000 turns

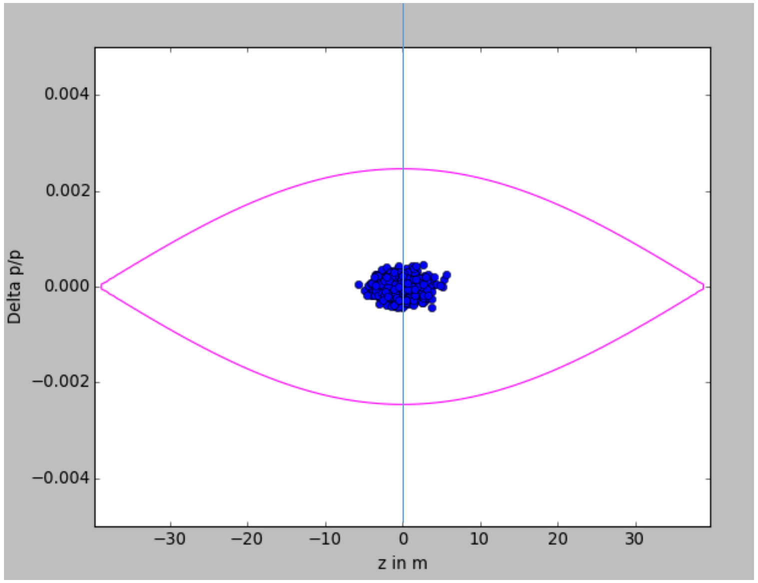


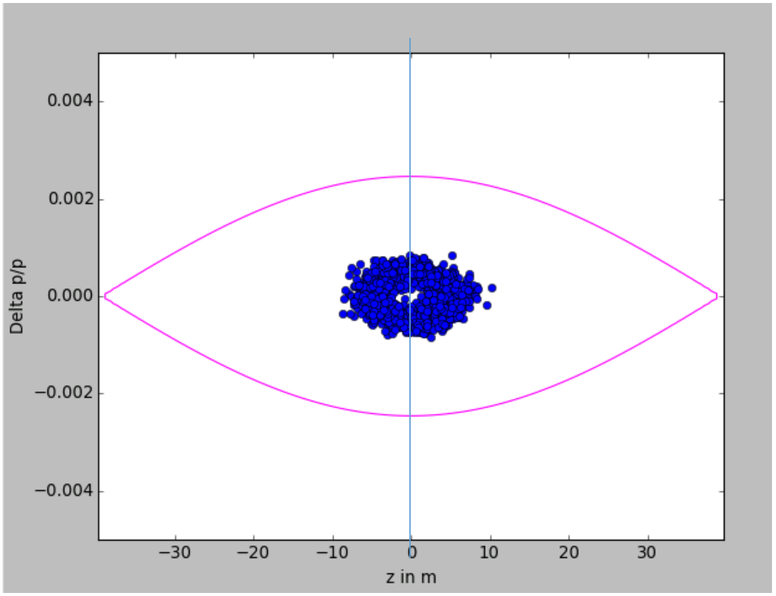


→ Filamentation and eventually?

Impact of phase mismatch

- Use a smaller bunch length (e.g. divide by 10)
- Use ~1000 particles
- Add e.g. 5m to all particles:
- Track particles for ~500000 turns
- Plot the distribution every 100 turns:





→ Eventually emittance growth
→ Try also with larger mismatch (20 m)

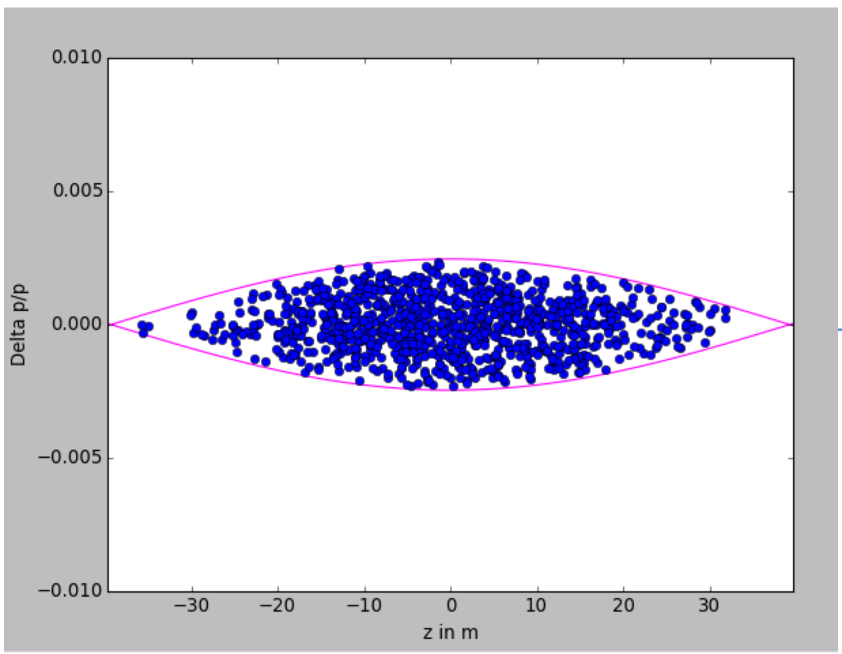
Agenda

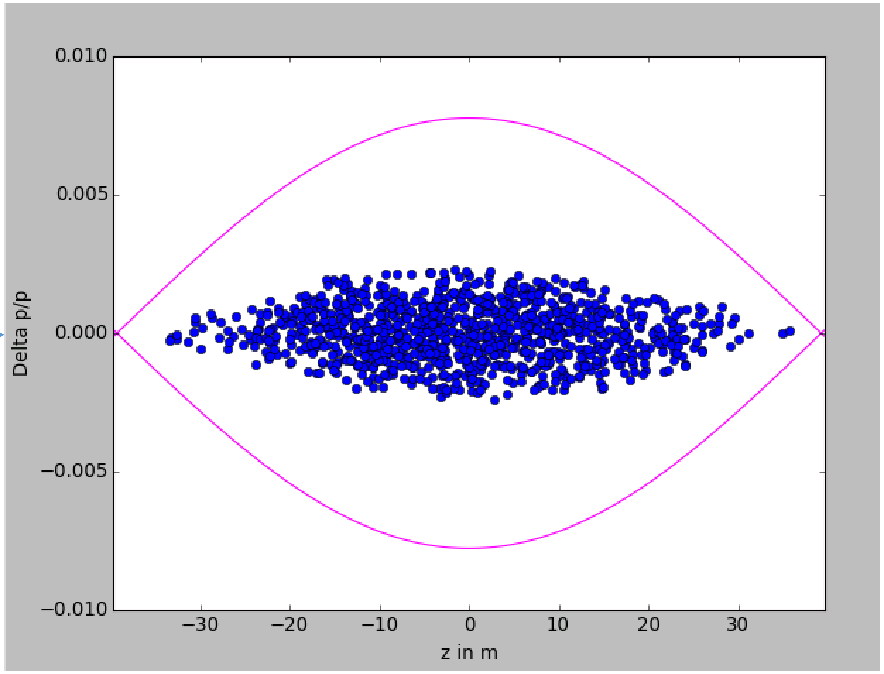
- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Impact of voltage mismatch (too high)

- Use the nominal bunch length
- track 1000 particles for 500 turns (plot every 10 turns)
- Change V_RF to V_RF*10 after the bunch is matched to V_RF

```
# Define RF systems
rfsystems = RFSystems(circumference, [harmonic], [V_rf*10], [phi_offset],
                      alpha_c_array, gamma, p_increment=p_increment_0)
```



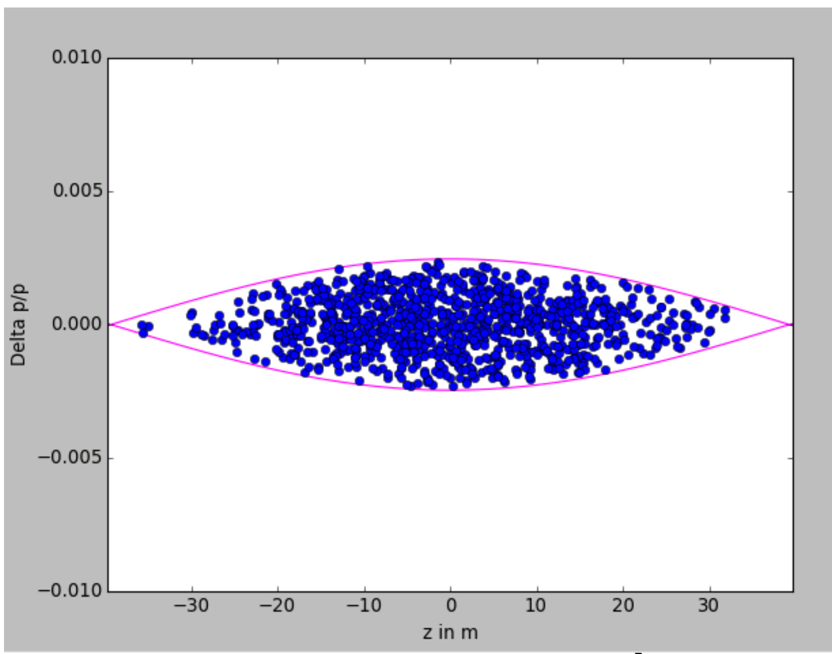


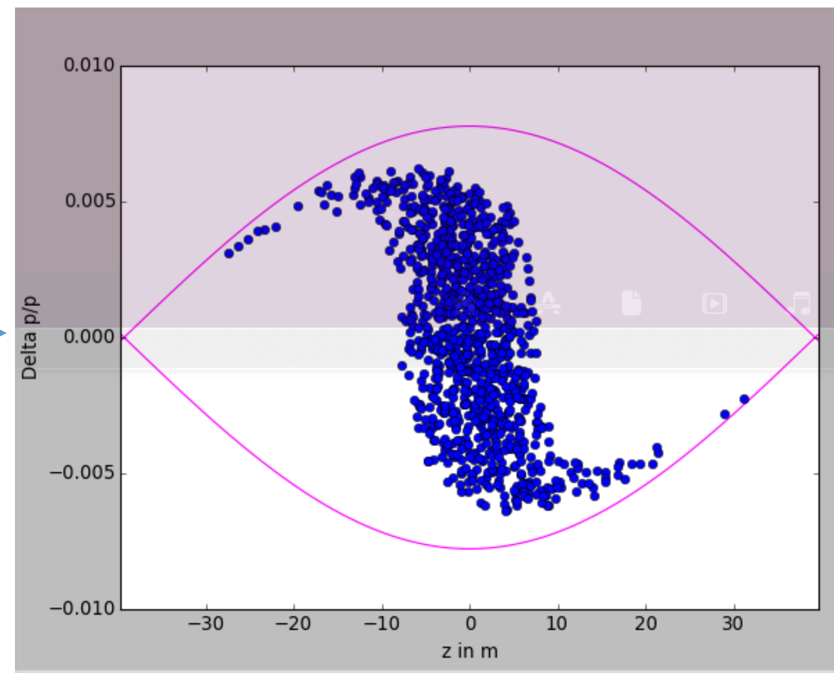
→ What will happen?

Impact of voltage mismatch (too high)

- Use the nominal bunch length
- track 1000 particles for 500 turns (plot every 10 turns)
- Change V_RF to V_RF*10 after the bunch is matched to V_RF

```
# Define RF systems
rfsystems = RFSystems(circumference, [harmonic], [V_rf*10], [phi_offset],
                      alpha_c_array, gamma, p_increment=p_increment_0)
```





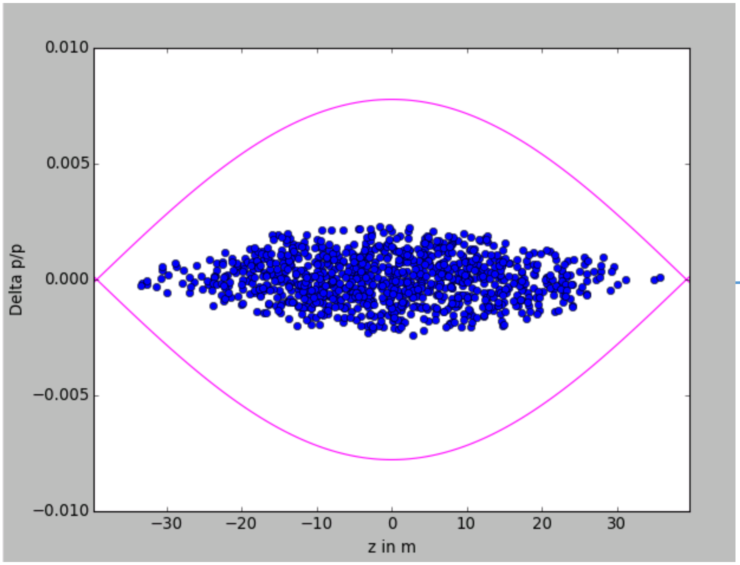
→ Bunch rotation

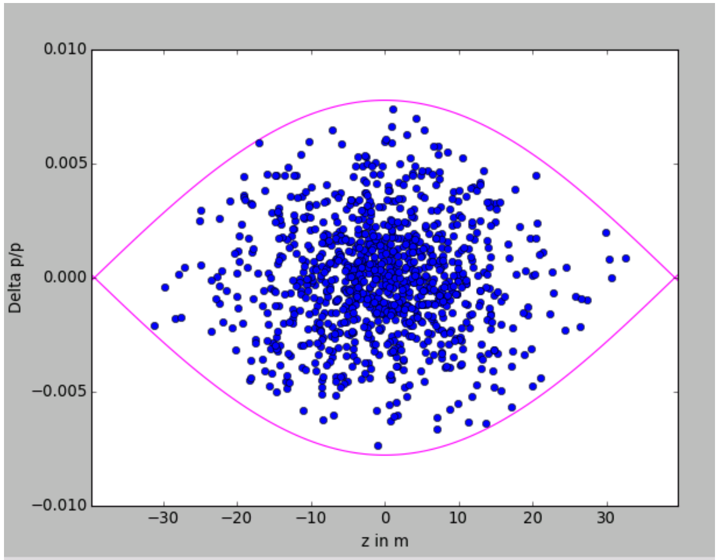
→ Bunch shortening, if phased correctly, but…

Impact of voltage mismatch (too high)

- Use the nominal bunch length
- track 1000 particles for 5000 turns (plot every 100 turns)
- Change V_RF to V_RF*10 after the bunch is matched to V_RF

```
# Define RF systems
rfsystems = RFSystems(circumference, [harmonic], [V_rf*10], [phi_offset],
                      alpha_c_array, gamma, p_increment=p_increment_0)
```



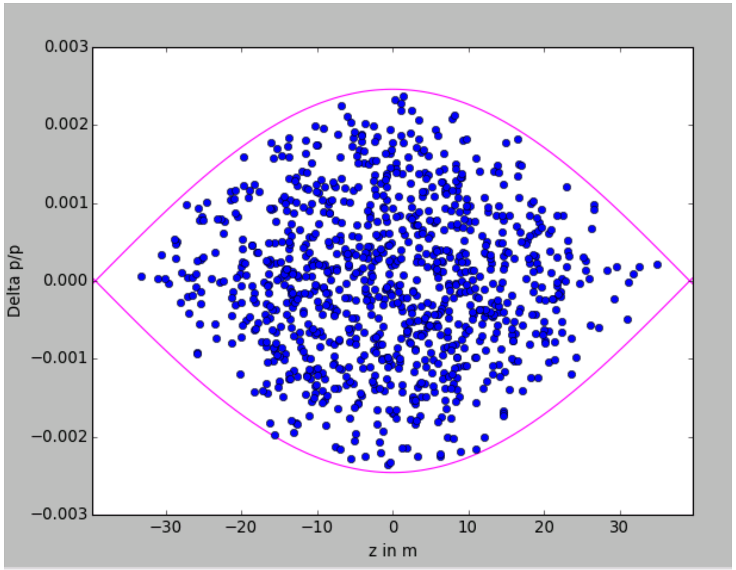


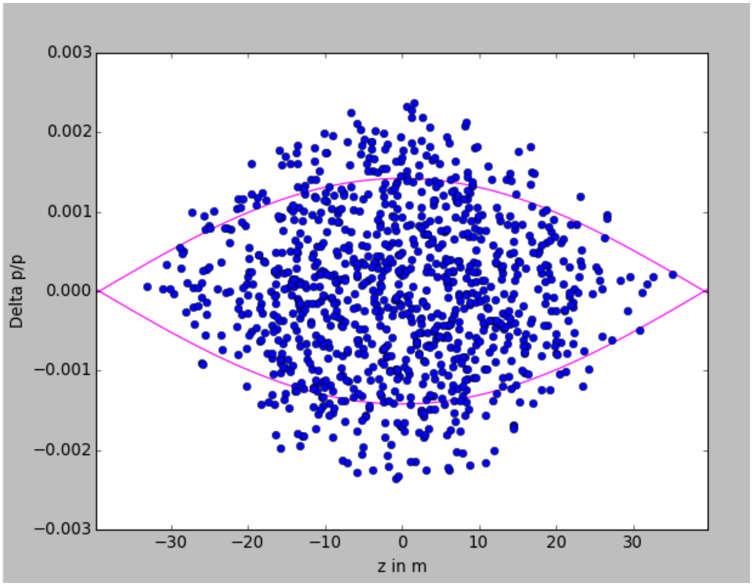
→ But should not wait too long!

→ Emittance growth!

Impact of voltage mismatch (too low)

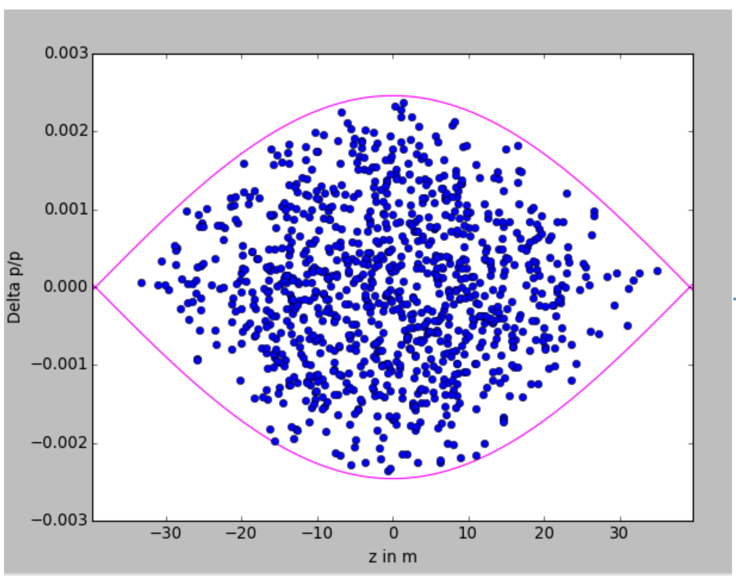
- Use the nominal bunch length
- track 1000 particles for 500 turns (plot every 10 turns)
- Change V_RF to V_RF/10 after the bunch is matched to V_RF
- Adapt the deltap plotting limits

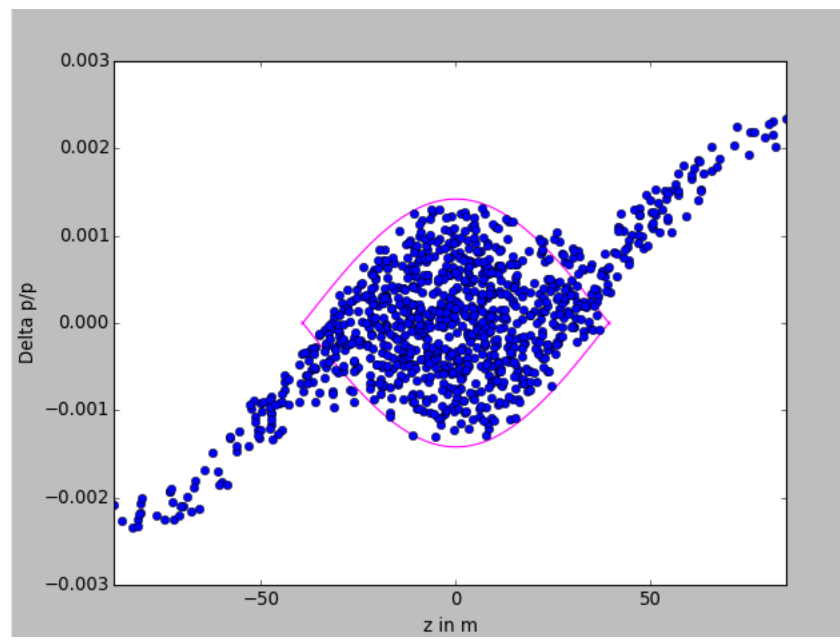




Impact of voltage mismatch (too low)

- Use the nominal bunch length
- track 1000 particles for 500 turns (plot every 10 turns)
- Change V_RF to V_RF/10 after the bunch is matched to V_RF
- Adapt the deltap plotting limits





Agenda

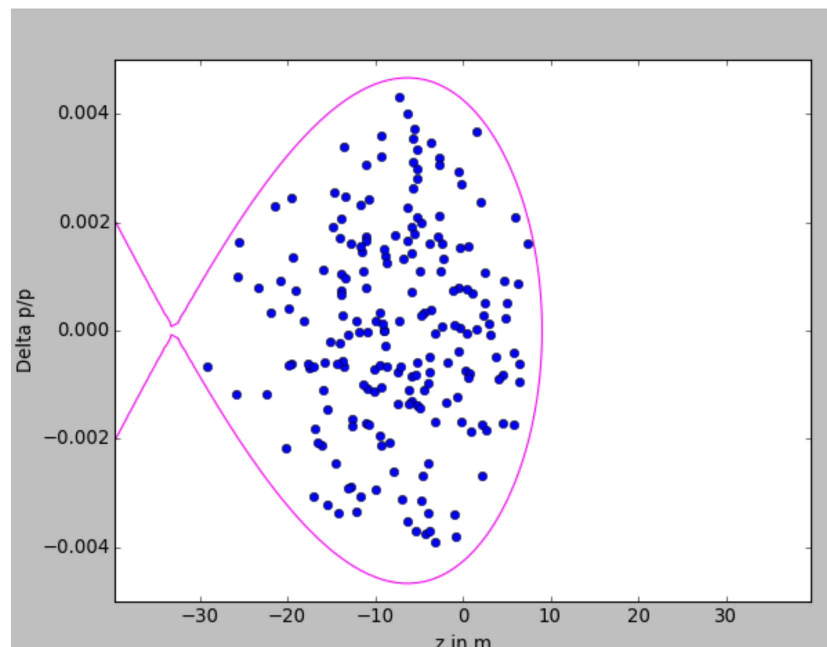
- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Impact of acceleration

- Use the nominal bunch length
- Use 200 particles
- Use V_RF=20 kV
- Use Bdot=2.2 T/s
- Does it work? Why?

Impact of acceleration

- Use the nominal bunch length
- Use 1000 particles
- Use Bdot=2.2 T/s
- Change VRF to 200 kV
- Compute the synchronous phase
- Plot the phase space in in ϕ[degrees], ΔE [MeV]



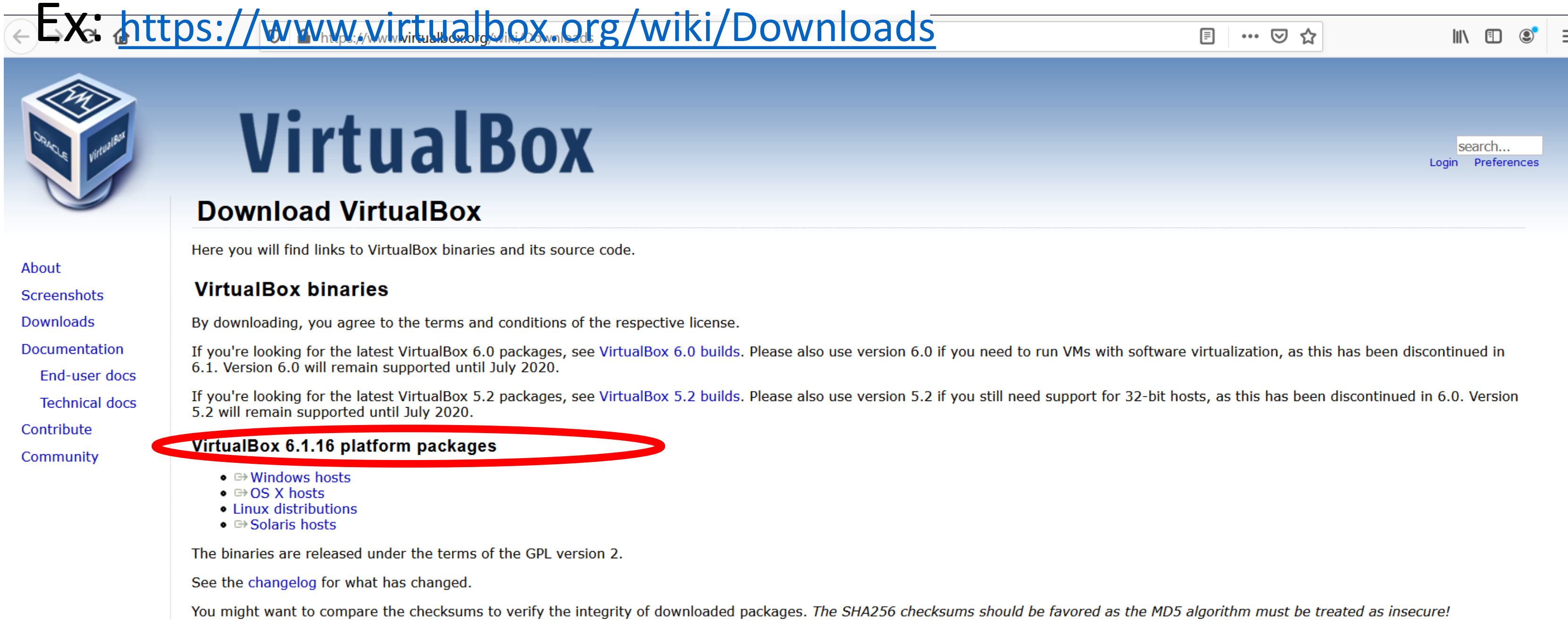
Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box (if you do not have Linux already!)
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

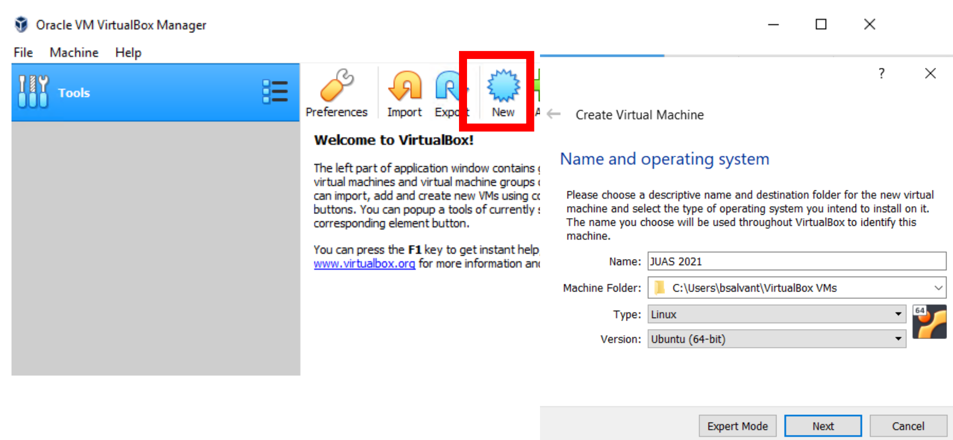
Setting up the operating system

- Install a virtual box to get an Ubuntu environment

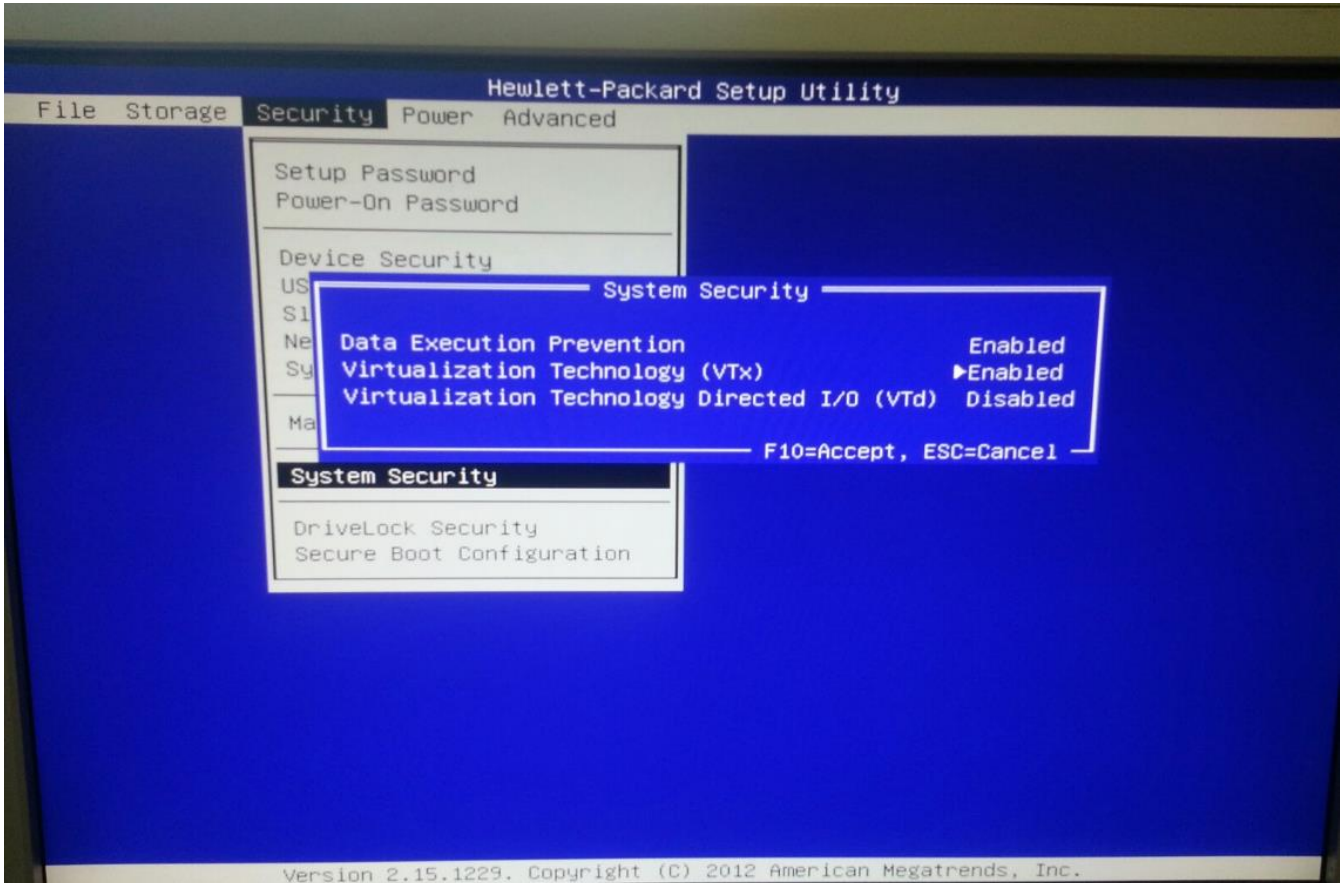
Ex: https://www.virtualbox.org/wiki/Downloads



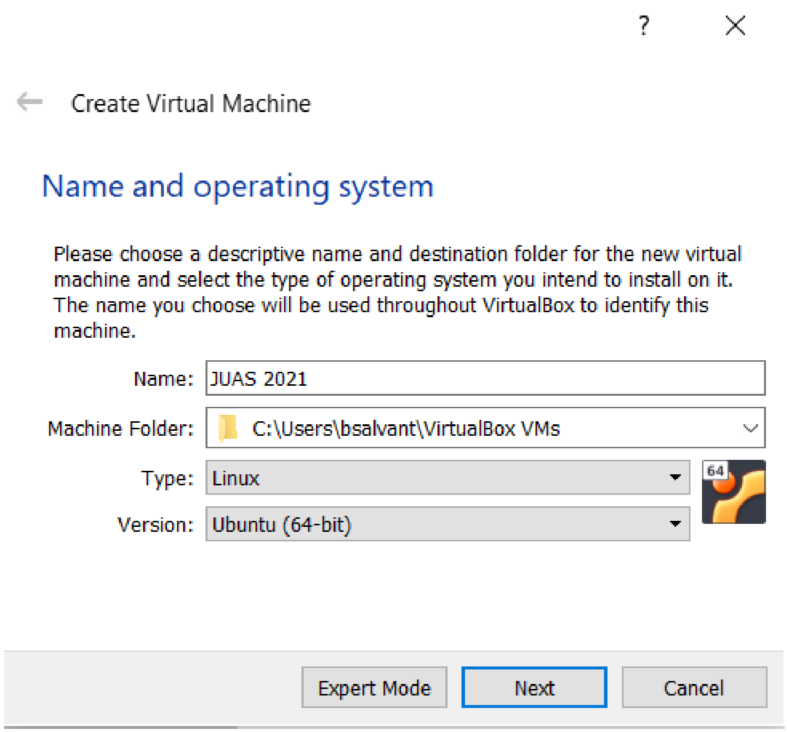
- Create a new virtual machine with Ubuntu 64-bit

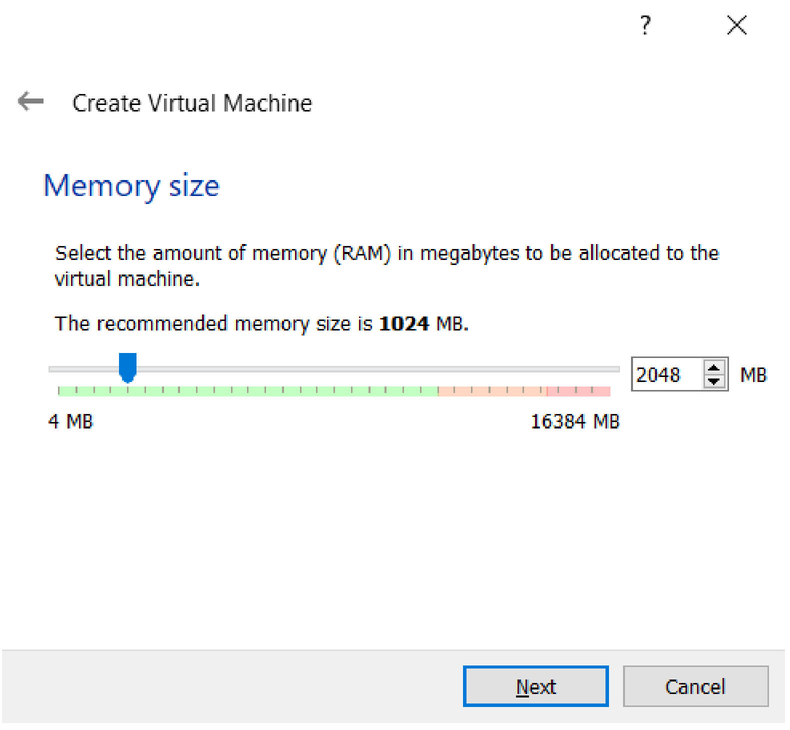


In case of a “VTx” error or an impossibility to select 64 bit operating system when launching the virtual machine, one needs to turn Virtualization Technology (VTx) on in the BIOS (see chapter 10.3 in https://www.virtualbox.org/manual/ch10.html#hwvirt)

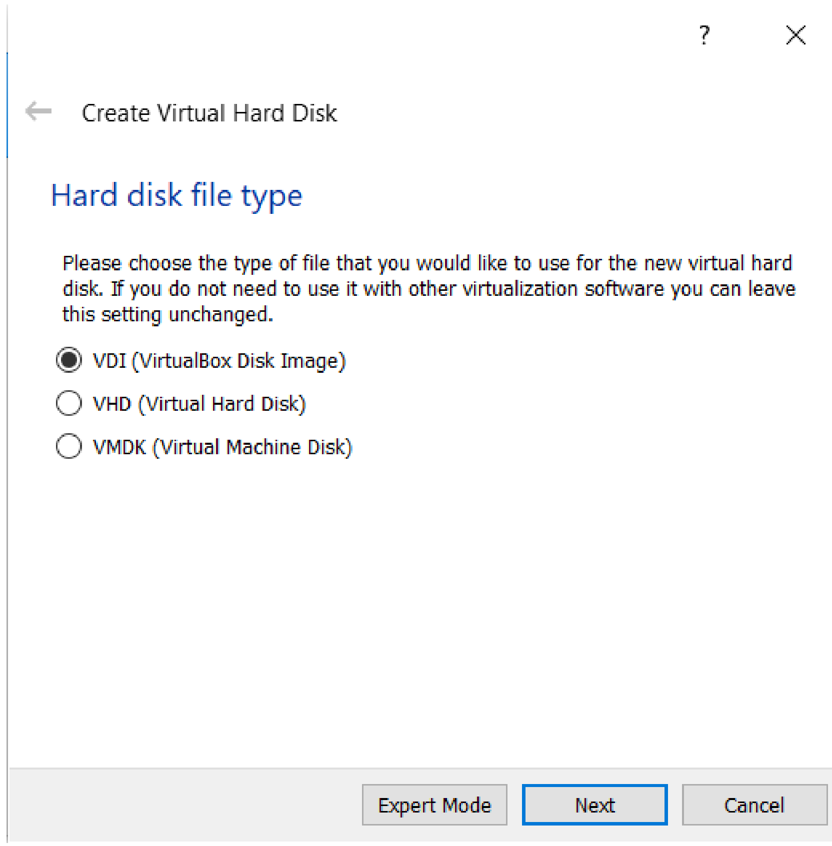


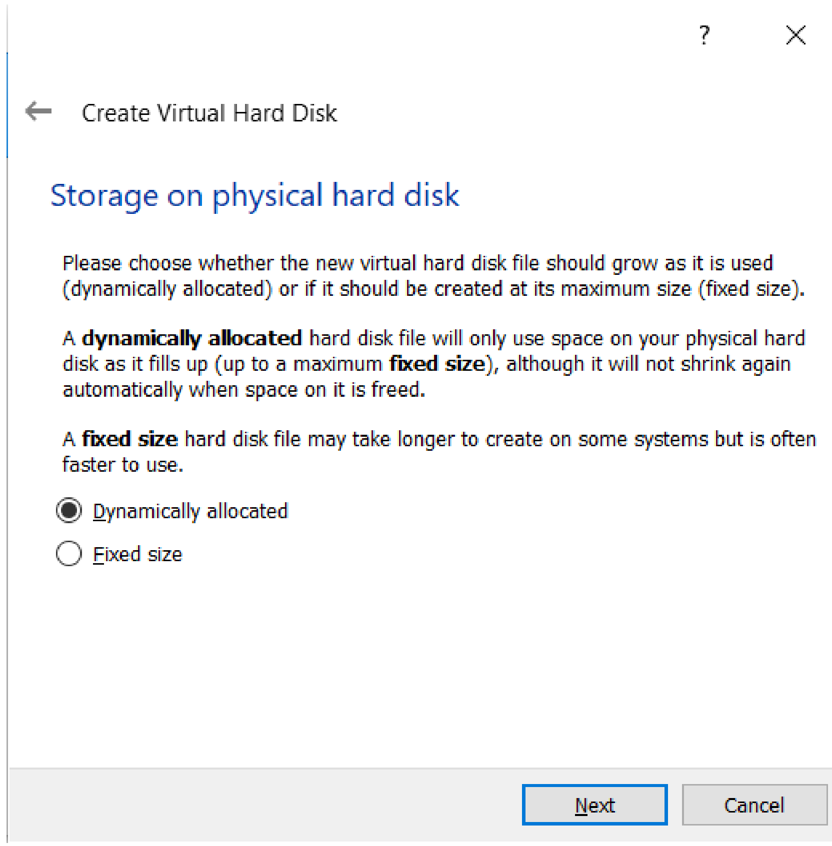
Create virtual machine

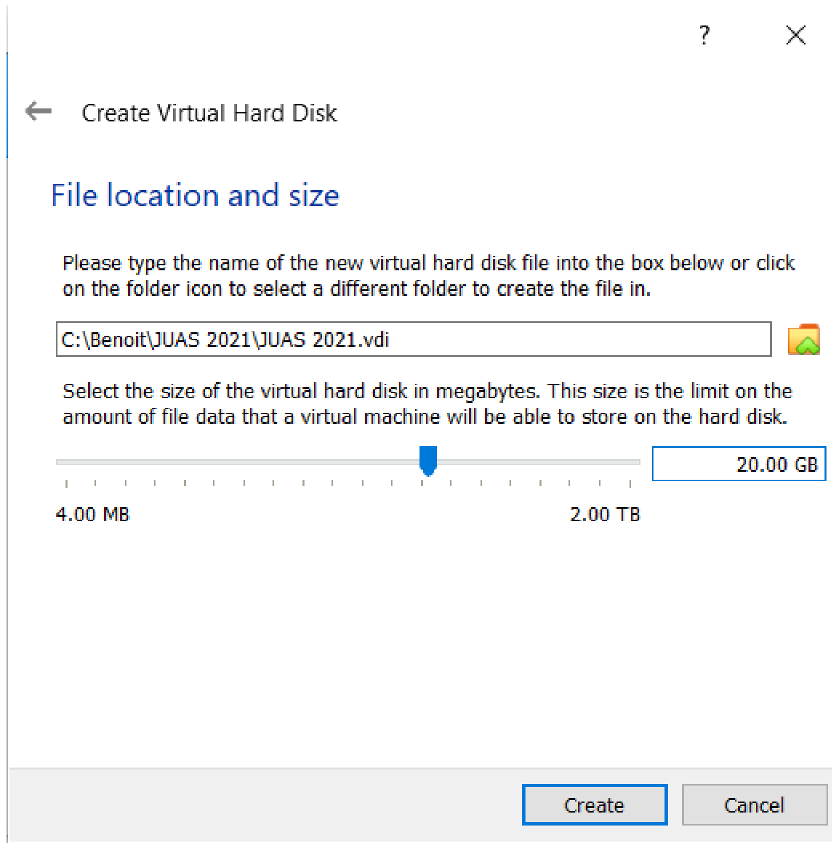




→ 2 GB is recommended by Ubuntu. This setting can be modified later.

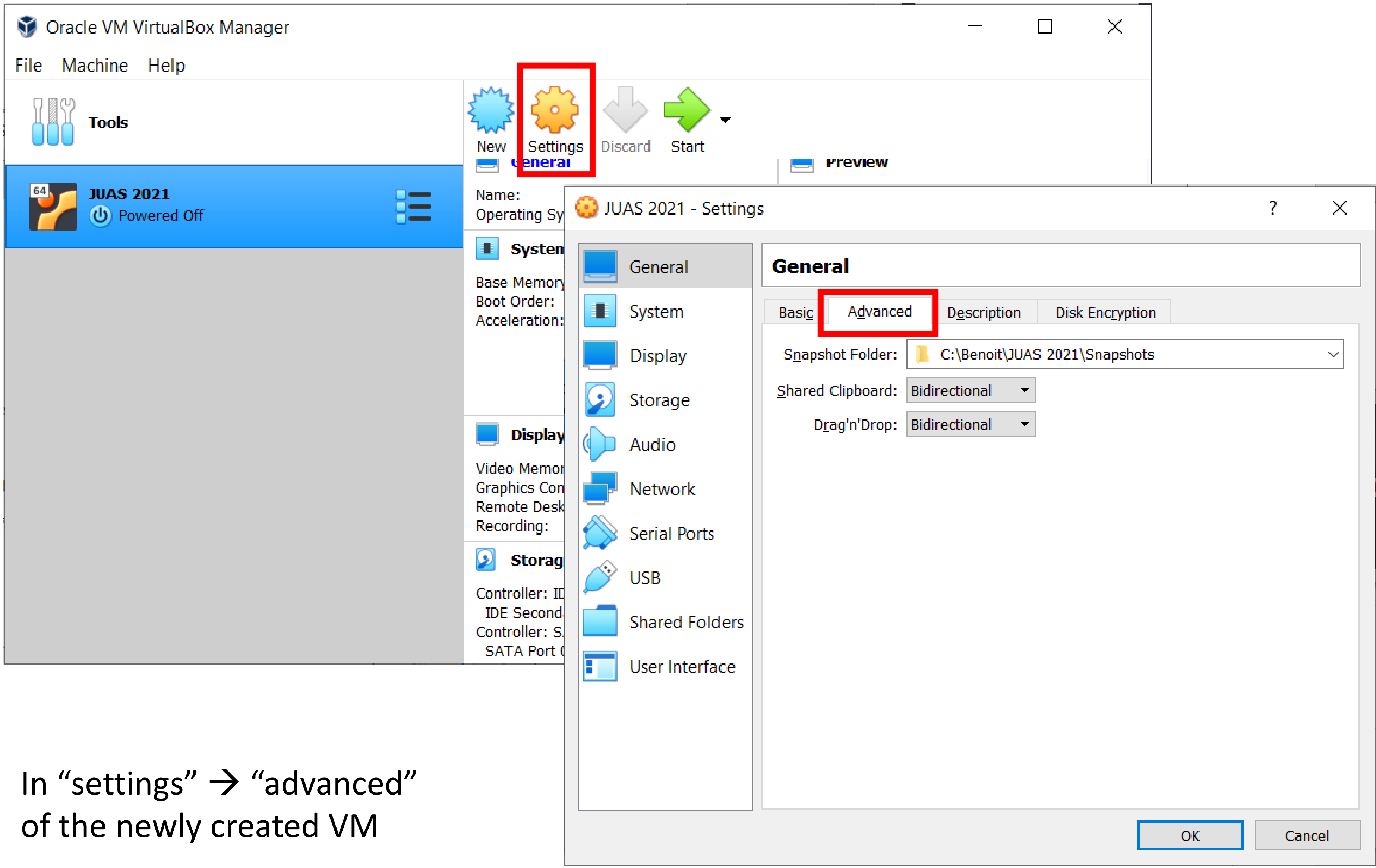

?
Create Virtual Hard Disk
Hard disk file type
Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.
VDI (VirtualBox Disk Image)
VHD (Virtual Hard Disk)
VMDK (Virtual Machine Disk)
Expert Mode
Next
Cancel


?
Create Virtual Hard Disk
Storage on physical hard disk
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).
A dynamically allocated hard disk file will only use space on your physical hard disk as it fills up (up to a maximum fixed size), although it will not shrink again automatically when space on it is freed.
A fixed size hard disk file may take longer to create on some systems but is often faster to use.
Dynamically allocated
Fixed size
Next
Cancel



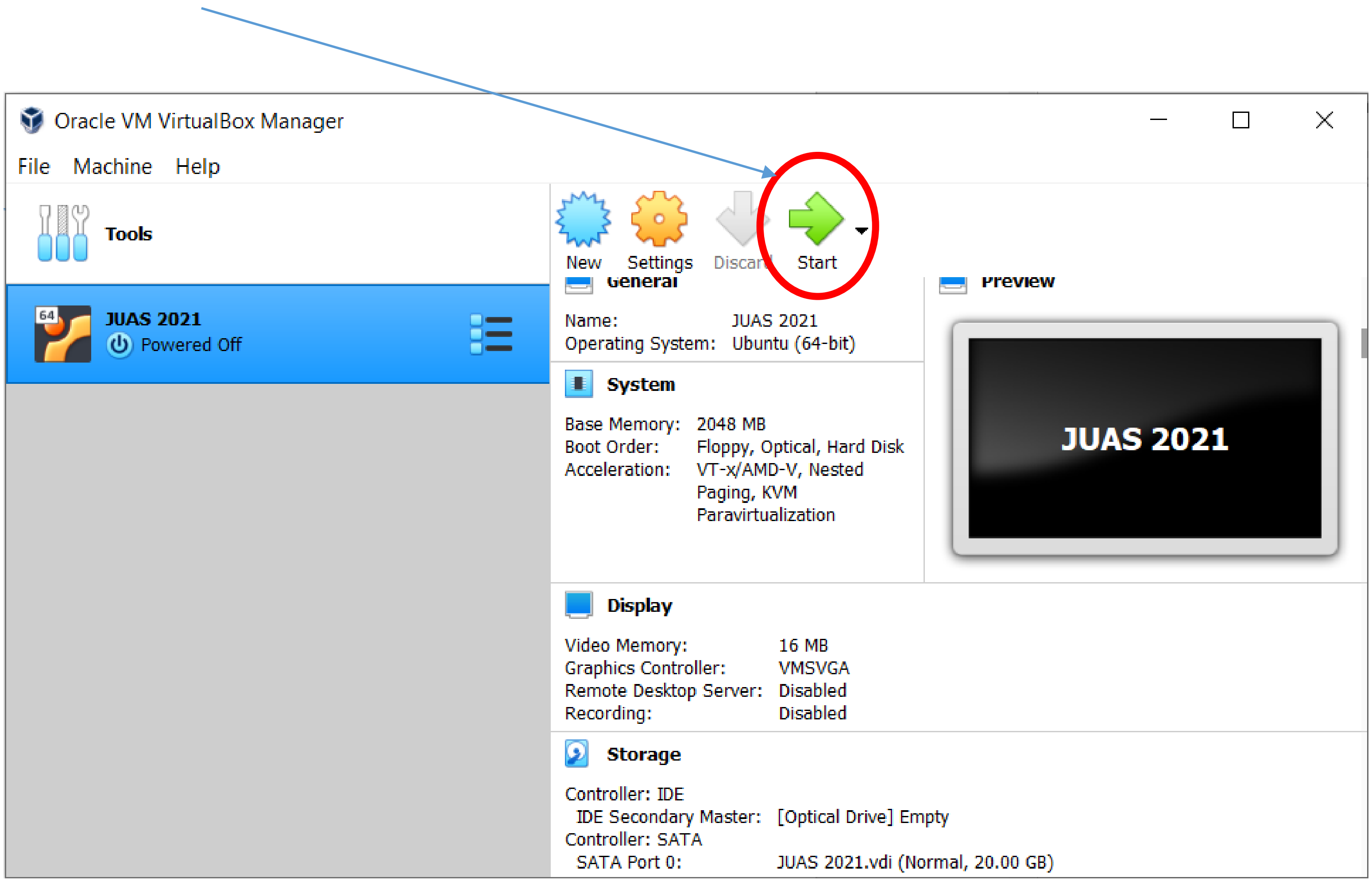
→ Assign 20 GB if possible.

Tip: allow shared clipboard if you wish



In “settings” → “advanced” of the newly created VM

Start the VM



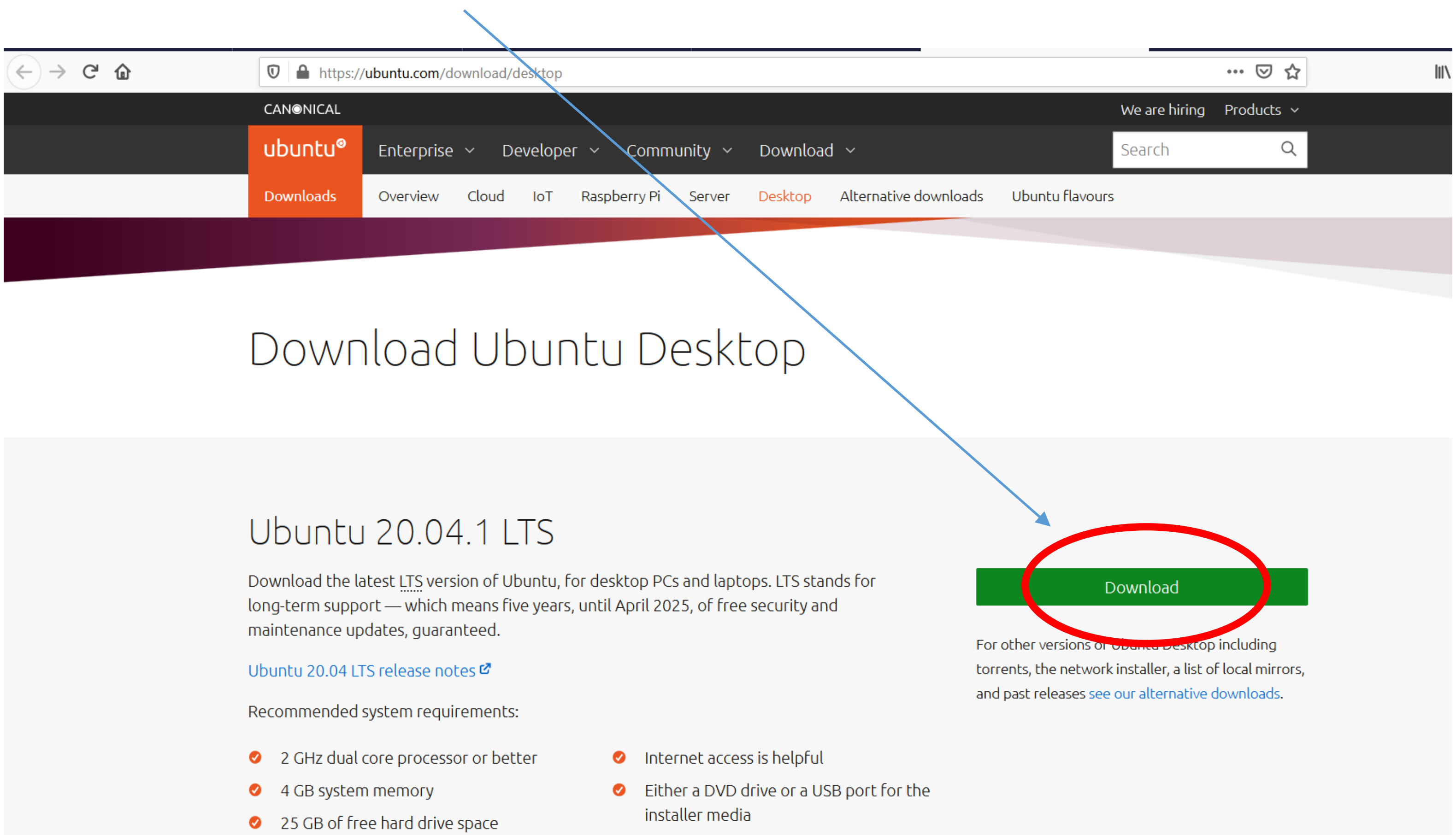
Oracle VM VirtualBox Manager
File
Machine
Help
Tools
New
Settings
Discard
Start
General
Preview
JUAS 2021
Powered Off
Name: JUAS 2021
Operating System: Ubuntu (64-bit)
System
Base Memory: 2048 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization
JUAS 2021
Display
Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled
Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: JUAS 2021.vdi (Normal, 20.00 GB)

Agenda

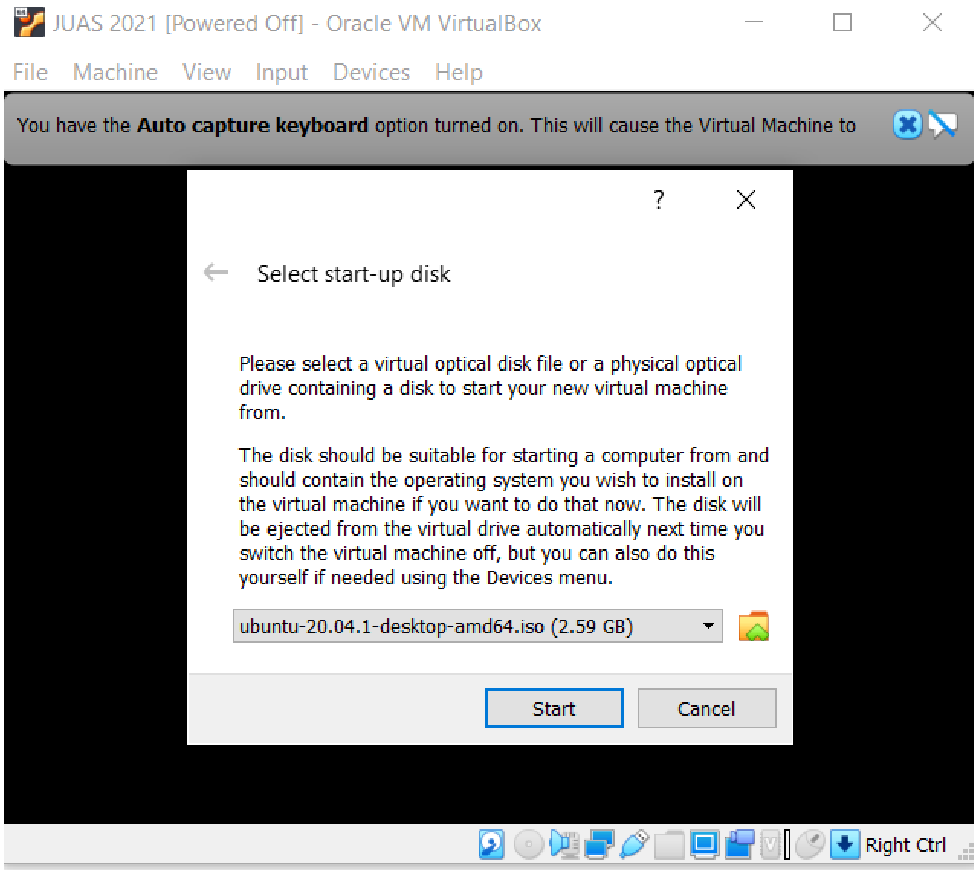
- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Download ubuntu

- https://www.ubuntu.com/download/desktop (beware quite large iso file – 2.6 Go)

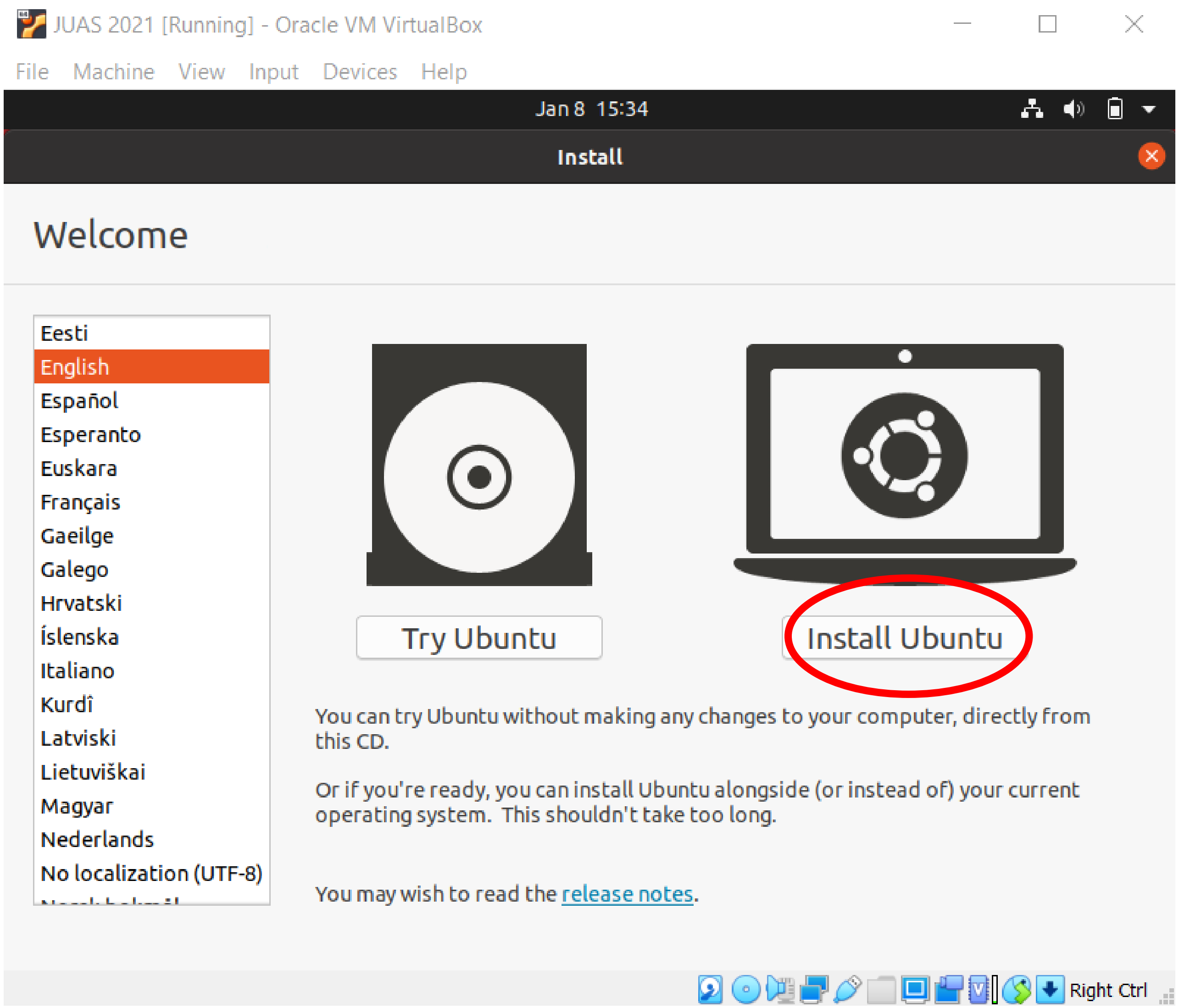


Select Ubuntu start up disk

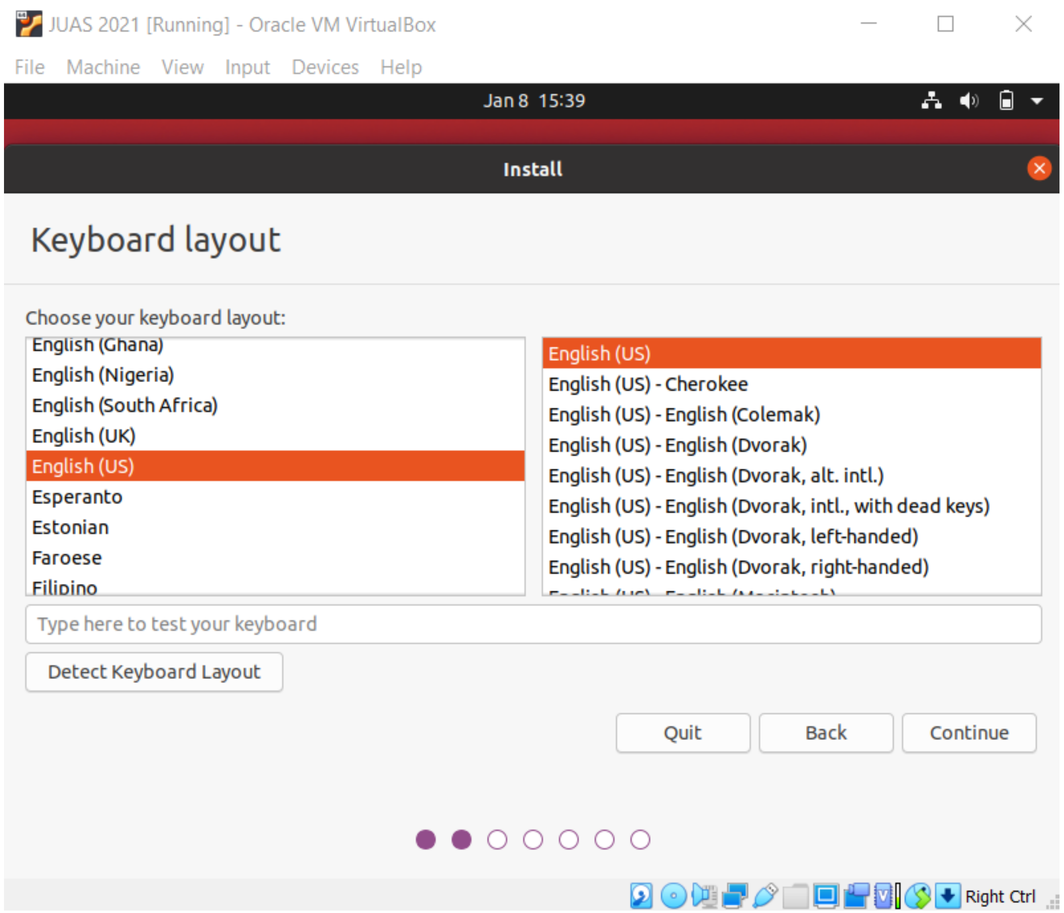


You may have to click on the small green arrow to browse for your Ubuntu installation image

Install ubuntu

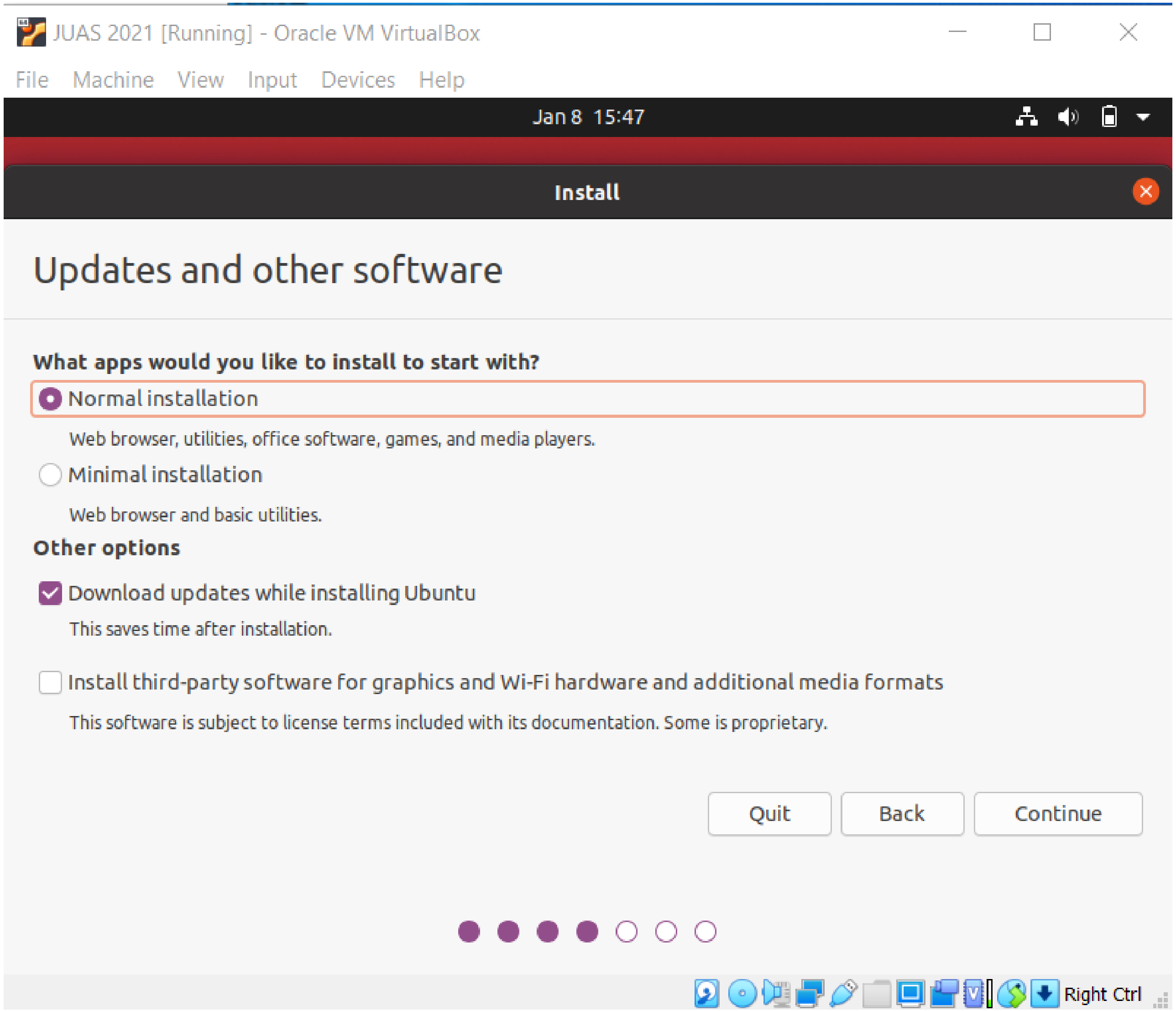


Choose the appropriate keyboard for your personal use (your physical keyboard configuration).

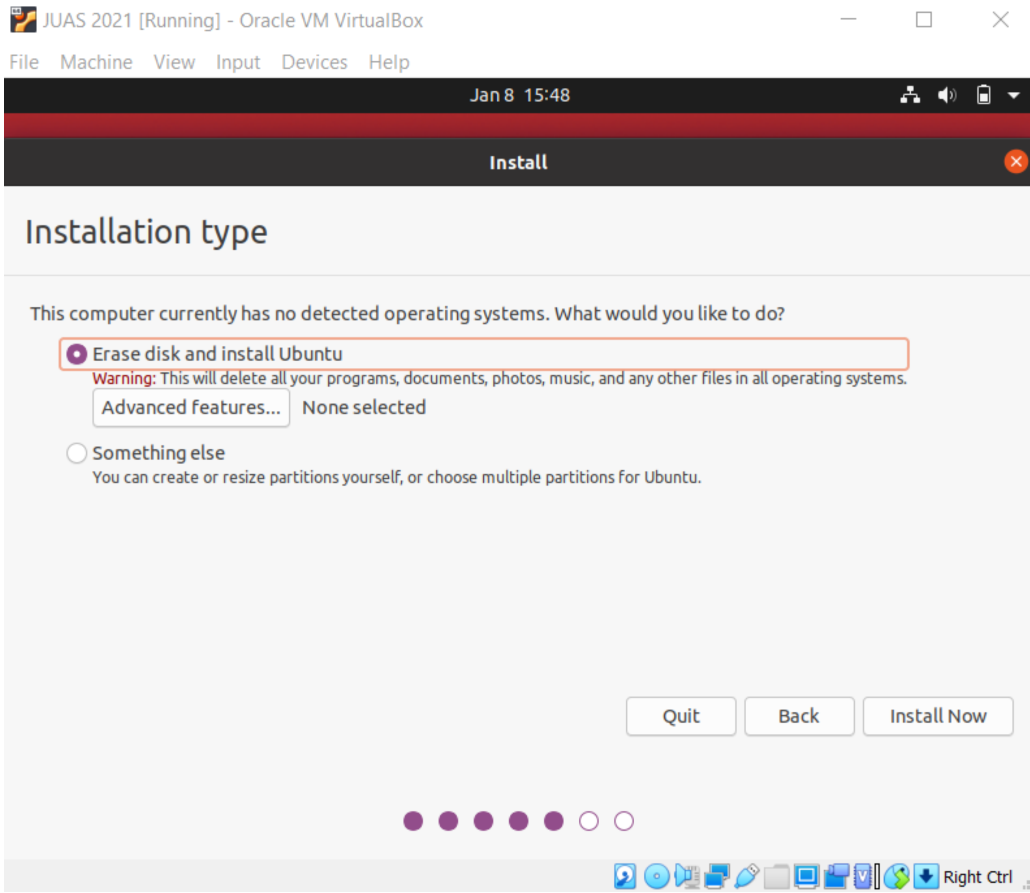


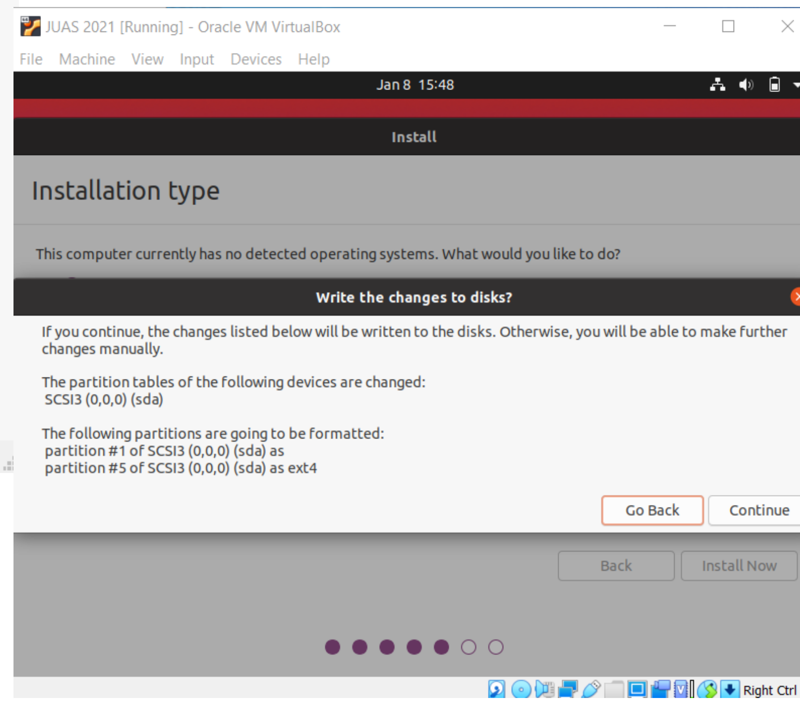
My keyboard is English (US)

Choose normal installation



Keep default installation and “install now” and confirm with “continue” on the pop up

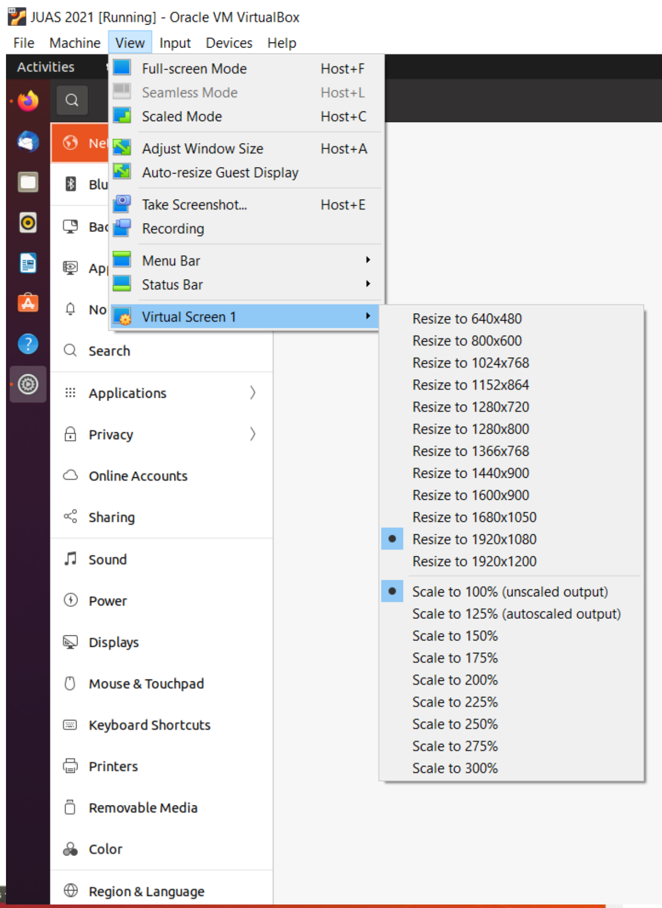


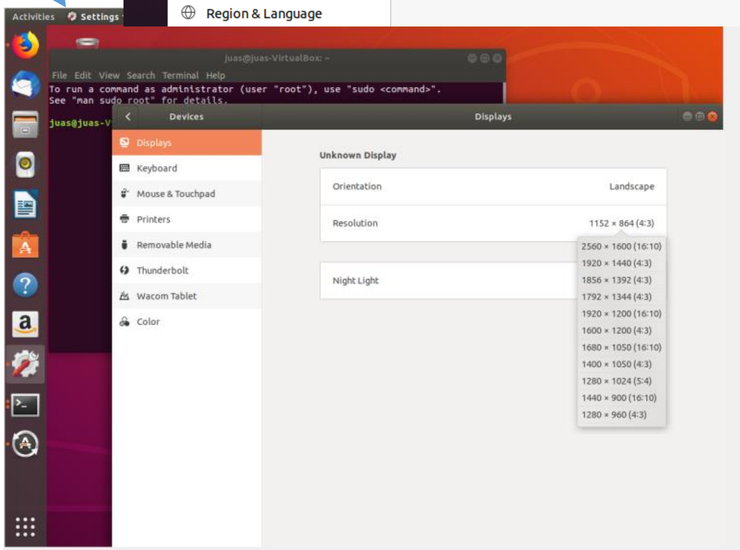


Then choose time zone and login details

- Then wait for file copy and Ubuntu installation restart the machine after it is complete

- If the screen is too small,
  - Change the scaling/seettings in the VM or
  - change the display settings inside Ubuntu or
  - shut down the VM
  - go to the VM Settings
  - » Display
  - » **Screen**
  - » Graphics Controller = VBoxVGA





Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Install anaconda

Download anaconda for Linux at
https://www.anaconda.com/distribution/#download-section

or type

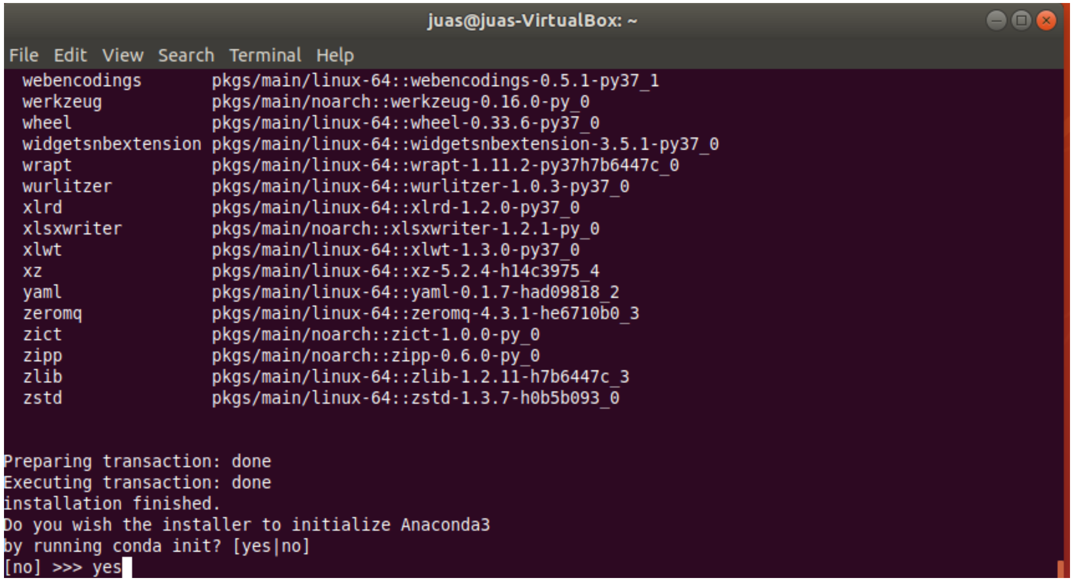
> wget https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh

→ Then install it by typing

> sh Anaconda3-2020.11-Linux86_64.sh

In the download directory

Do not forget to add Anaconda to path (answer yes or add the path yourself in the ".bashrc" configuration file)



→ Restart the shell (">>source .bashrc" from the home directory)

→ Check that ">>ipython notebook" works

→ Anaconda contains all the necessary libraries to run PyHEADTAIL.

Agenda

- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
  - Impact of phase mismatch
  - Impact of voltage mismatch
  - Impact of acceleration
- Setting up the environment
  - Virtual box
  - Ubuntu
  - Anaconda
  - PyHEADTAIL

Installing PyHEADTAIL

- website for installation and information: https://github.com/PyCOMPLETE/PyHEADTAIL

```
(base) juas@juas-VirtualBox:~/PyHEADTAIL$ sudo apt-get install gcc
(base) juas@juas-VirtualBox:~/PyHEADTAIL$ pip install PyHEADTAIL
```

Add path to the end of the .bashrc to be able to import PyHeadtail from everywhere:

```
juas@juas-VirtualBox:~$ gedit ~/.bashrc
```

```
# PyHEADTAIL
export PYTHONPATH=/home/juas/:$PYTHONPATH
```

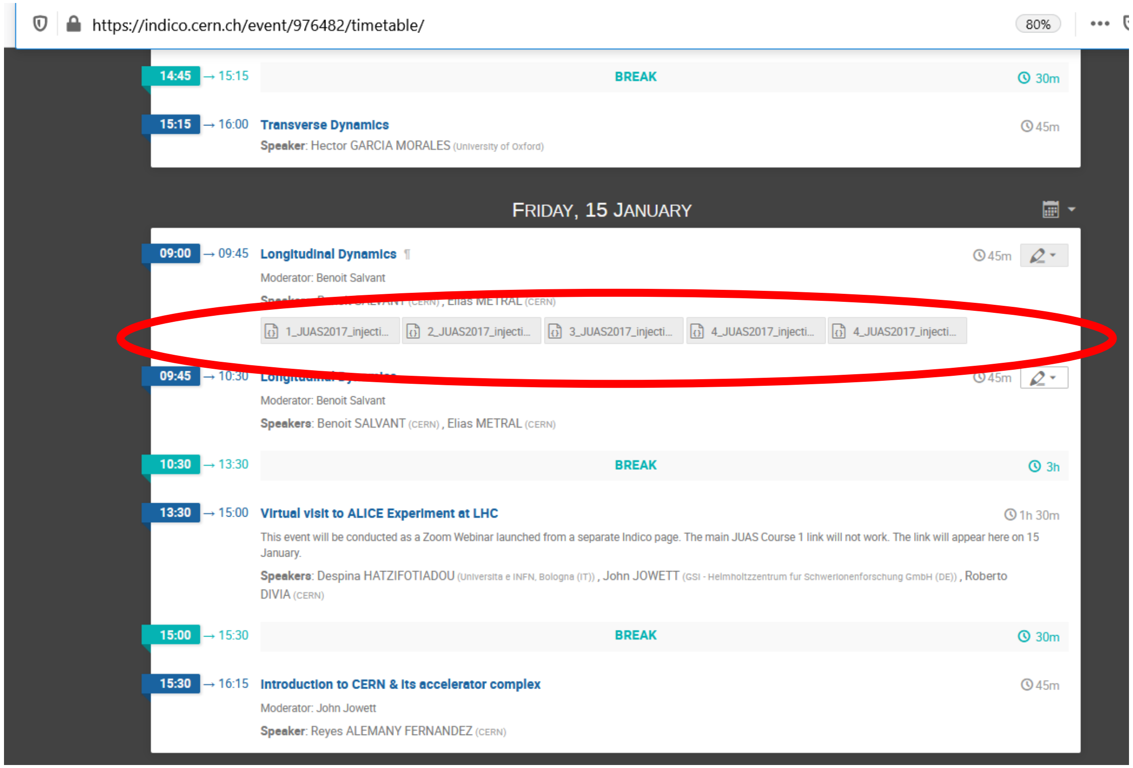
This only updates once a new terminal is opened (or do source ~/.bashrc)

In case of issues with PyHEADTAIL installation, most likely it is that fortran and C need to be installed:

- sudo apt-get install gfortran
- Sudo apt-get install gcc

Finding the examples

- From JUAS 2021 Indico site in the last tutorial of the course: https://indico.cern.ch/event/976482/timetable/



→ Download the *.ipynb files and open them browsing with "ipython notebook"

Changing variables

JUAS

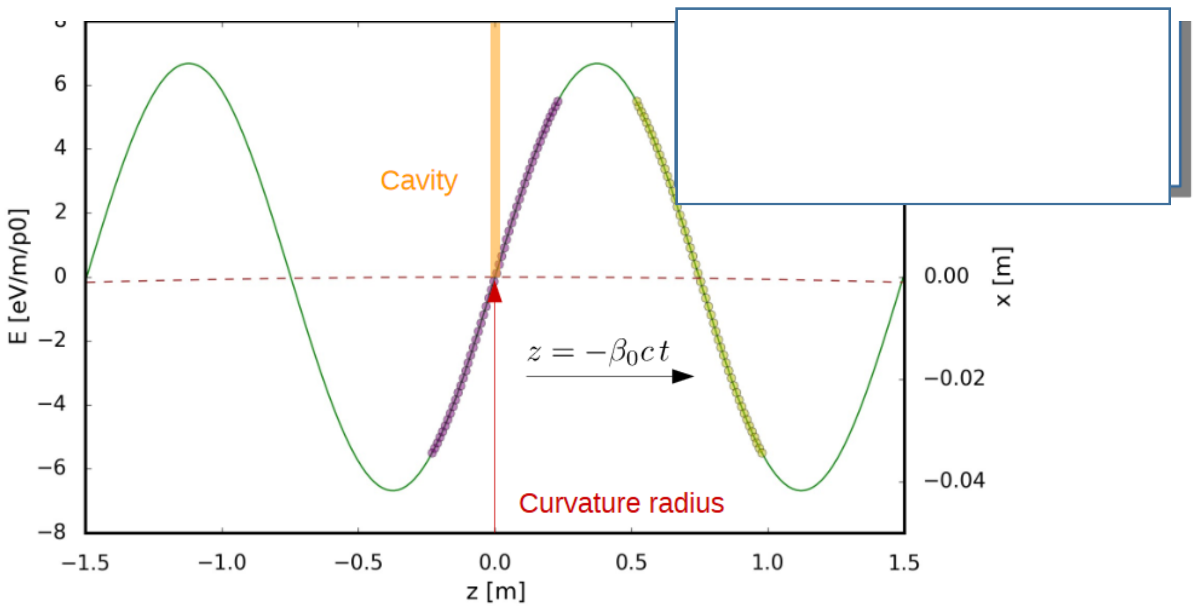
Joint Universities Accelerator School

- Change of variables if one wants to use (Φ, ΔE) or (Δt, ΔE) instead of (Φ, $d\Phi/dt$)

$$\Delta\phi = \phi - \phi_s$$
$$= \omega_{RF}\,\Delta t$$
$$= h\,\omega_s\,\Delta t$$

$$\Delta p = \frac{\Delta E}{\beta_s\,c}$$

$$\dot{\phi} = -\frac{\eta\,h\,c}{\beta_s\,E_s\,R_s}\,\Delta E$$



$$\Delta\phi[rad] = -\frac{h}{R}\,z[m]$$

$$\Delta \mathrm{E} = \frac{\Delta p}{p} m_0 \gamma \beta^2 c^2$$

For protons $$\Delta \mathrm{E}[GeV] = \frac{\Delta p}{p} 0.938\,\gamma\beta^2$$