

PyHEADTAIL examples

All our thanks to

D. Amorim, Kevin Li, Lotta Methner and Michael Schenk
CERN BE/ABP-HSC

and

Martial Fol and Saïd Errghioui (AZIMUTEC), Marie Gauthier and Coline Morin (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!

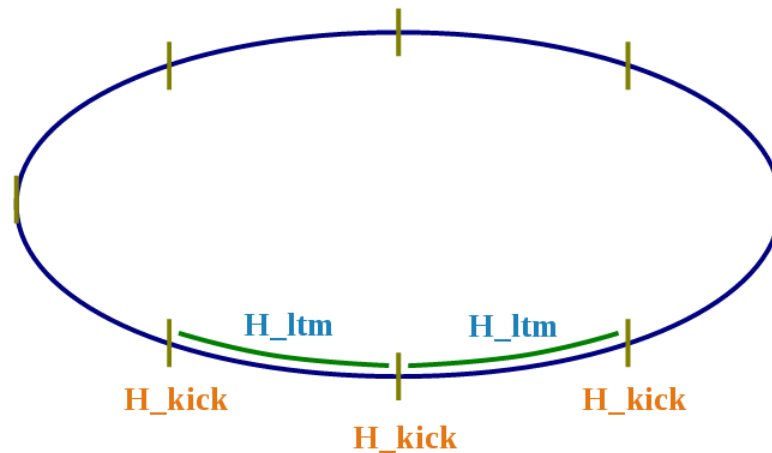
introduction to PyHEADTAIL

Courtesy Kevin Li
USPAS 2015

How does PyHEADTAIL work?



- PyHEADTAIL is a macroparticle tracking code designed specifically to simulate collective effects in circular accelerators



- H_ltm: linear transfer map
 - Chromaticity
 - Amplitude detuning
 - ...
- H_kick: collective interaction
 - Wakefields
 - Electron cloud
 - Feedback
 - Space-charge
 - ...



introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



- Load Python packages and modules

```
1 from __future__ import division
2 import cProfile, itertools, sys, time, timeit
3
4 from scipy.constants import c, e, m_p
5
6 from cobra_functions import stats, random
7 from beams.beams import *
8 from monitors.monitors import *
9 from spacecharge.spacecharge import *
10 from trackers.transverse_tracker import *
11 from trackers.longitudinal_tracker import *
12
```

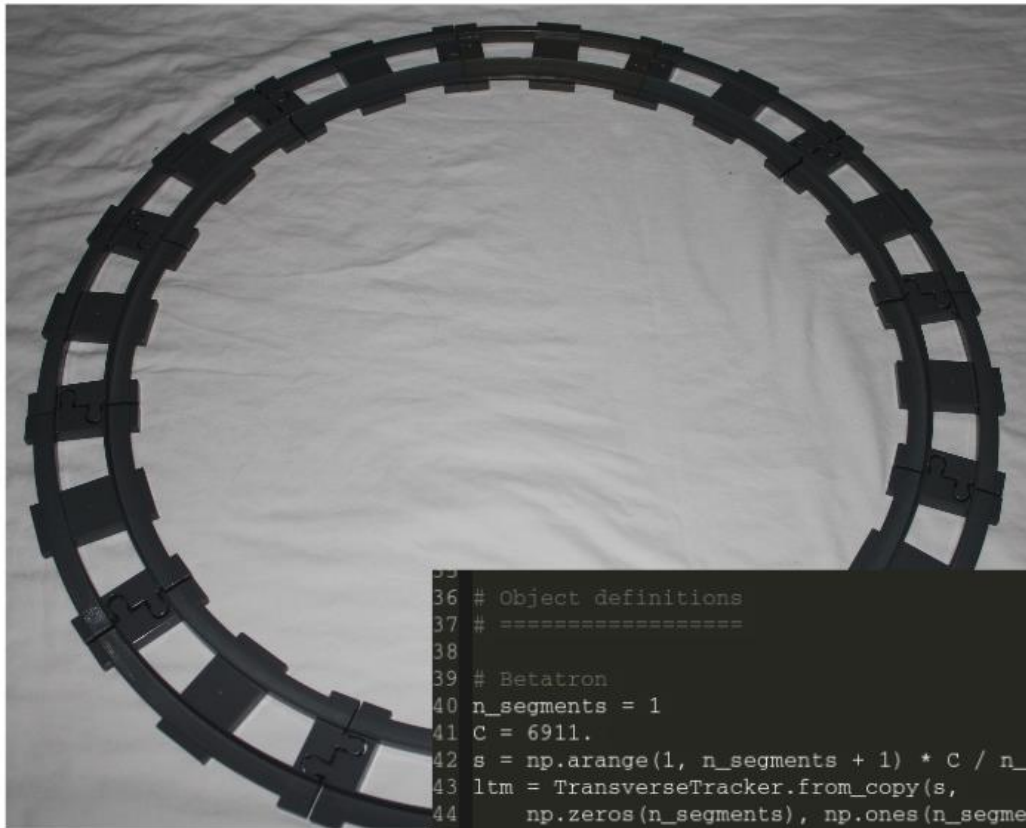


introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



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

```
35  
36 # Object definitions  
37 # =====  
38  
39 # Betatron  
40 n_segments = 1  
41 C = 6911.  
42 s = np.arange(1, n_segments + 1) * C / n_segments  
43 ltm = TransverseTracker.from_copy(s,  
44     np.zeros(n_segments), np.ones(n_segments) * beta_x, np.zeros(n_segments),  
45     np.zeros(n_segments), np.ones(n_segments) * beta_y, np.zeros(n_segments),  
46     Qx, 0, 0, Qy, 0, 0)  
47
```

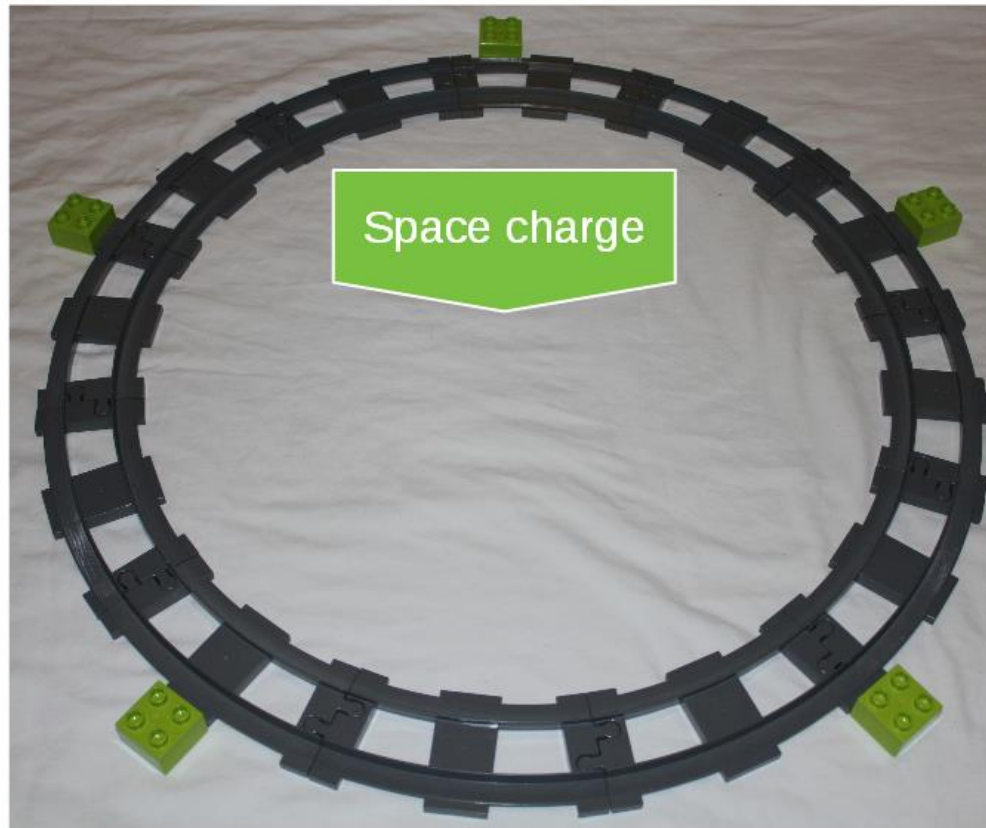


introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



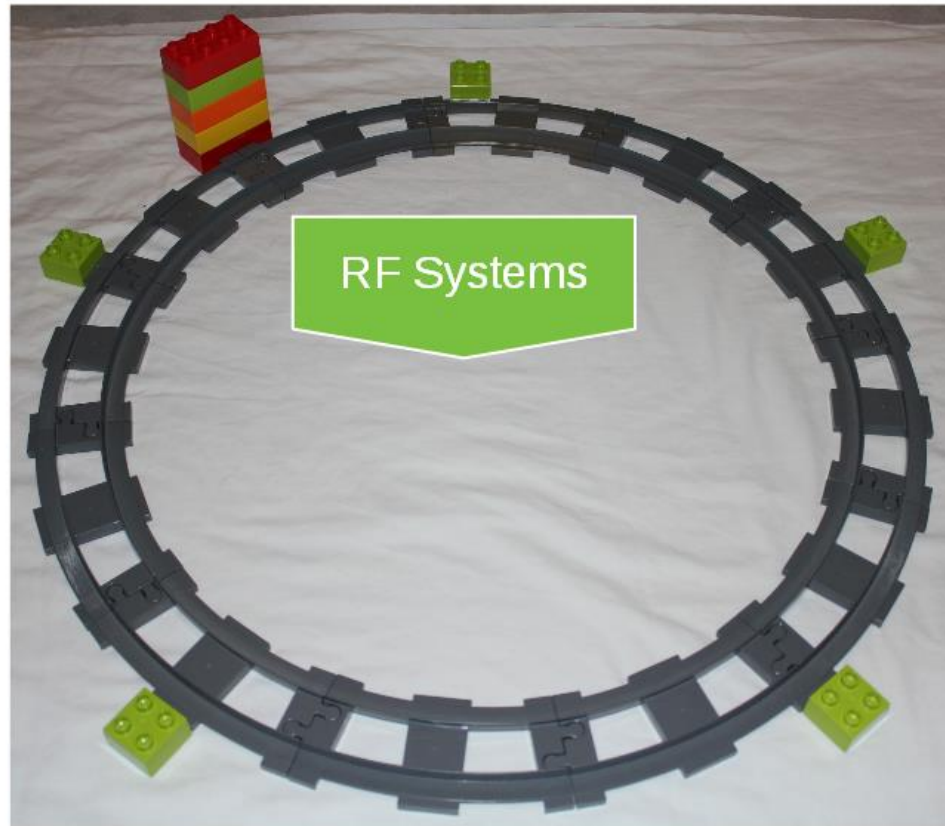
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



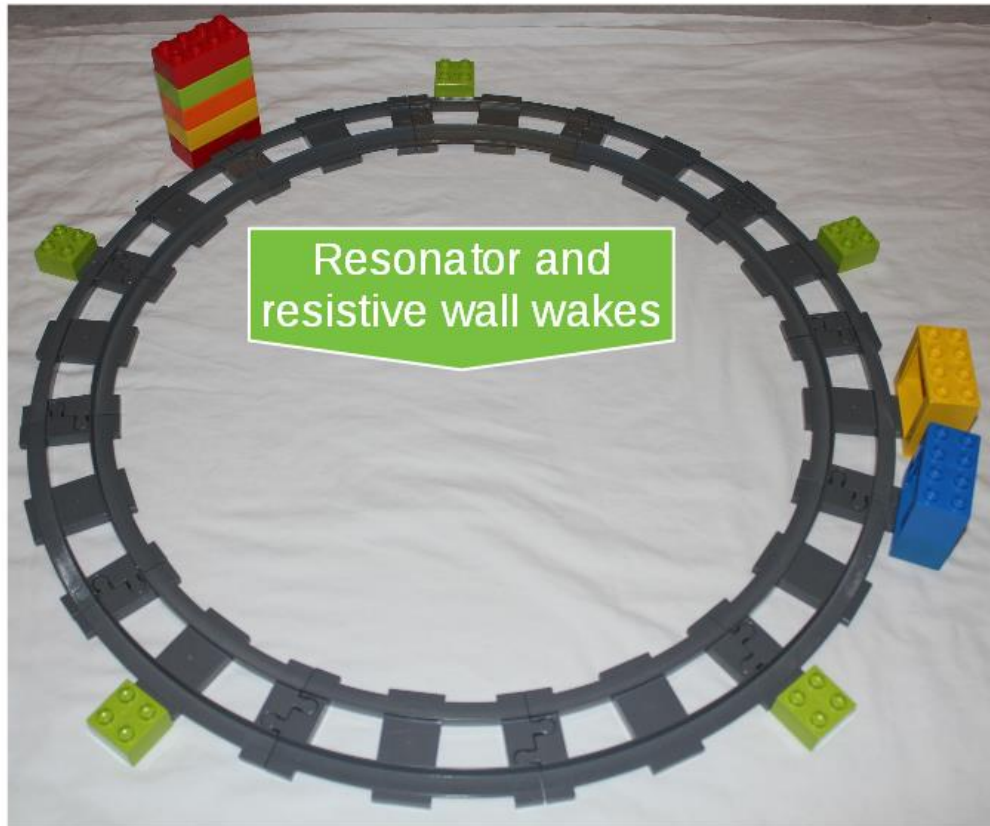
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



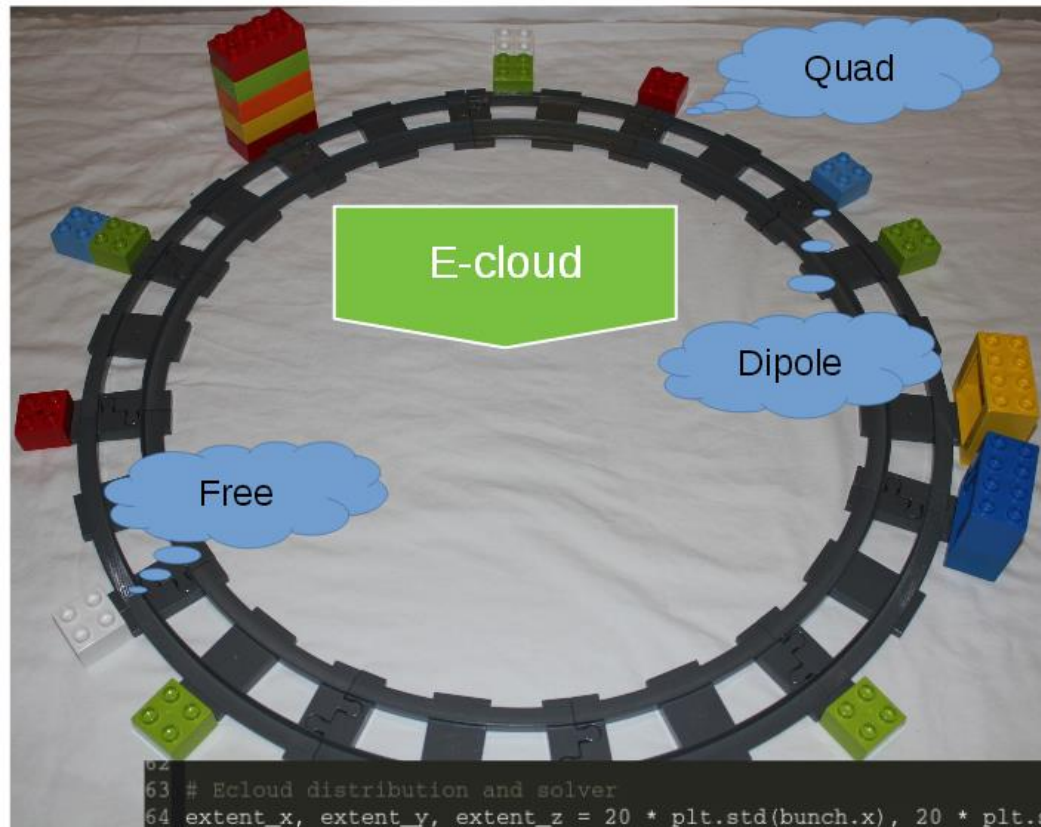
- Load Python packages and modules
- Build linear periodic transfer maps
- Add (collective) kick elements

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

```
62  
63 # Ecloud distribution and solver  
64 extent_x, extent_y, extent_z = 20 * plt.std(bunch.x), 20 * plt.std(bunch.y), C / n_segments  
65 cloud = Cloud(100000, lcell, extent_x, extent_y, extent_z)  
66 ecloud = SpaceCharge(cloud, 'cloud', extent_x, extent_y, 128, 128, slices)  
67
```

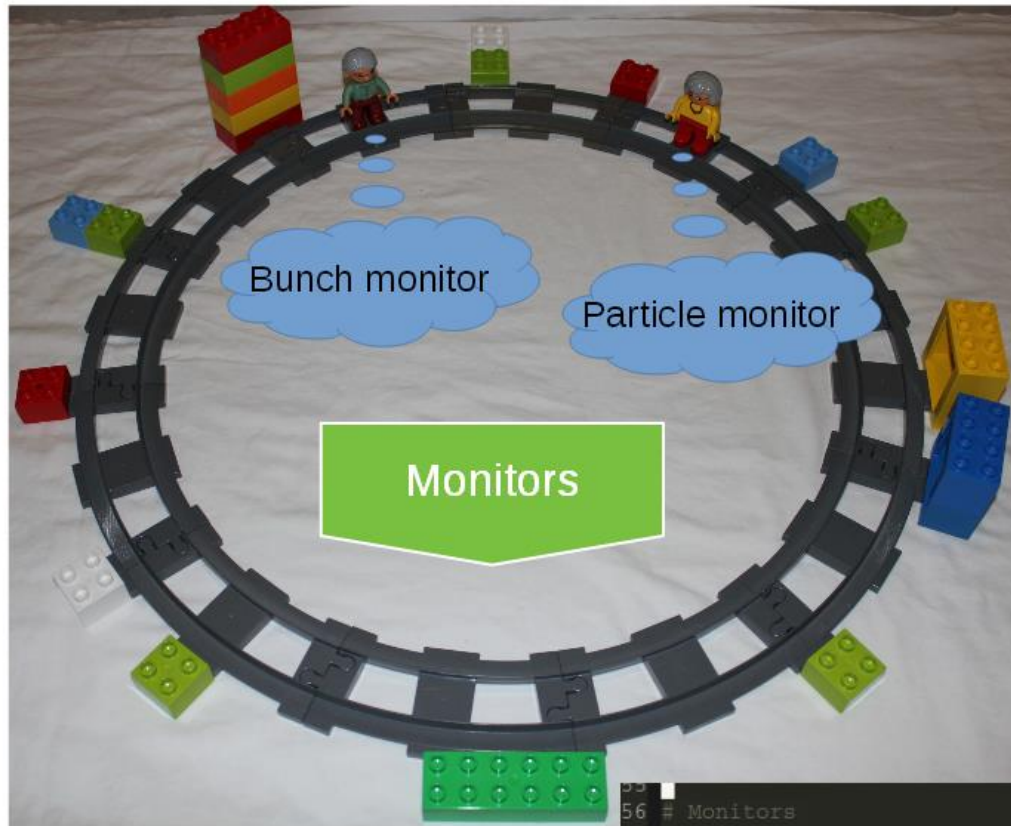


introduction to PyHEADTAIL



A real world example

Courtesy
Kevin Li
USPAS 2015



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

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

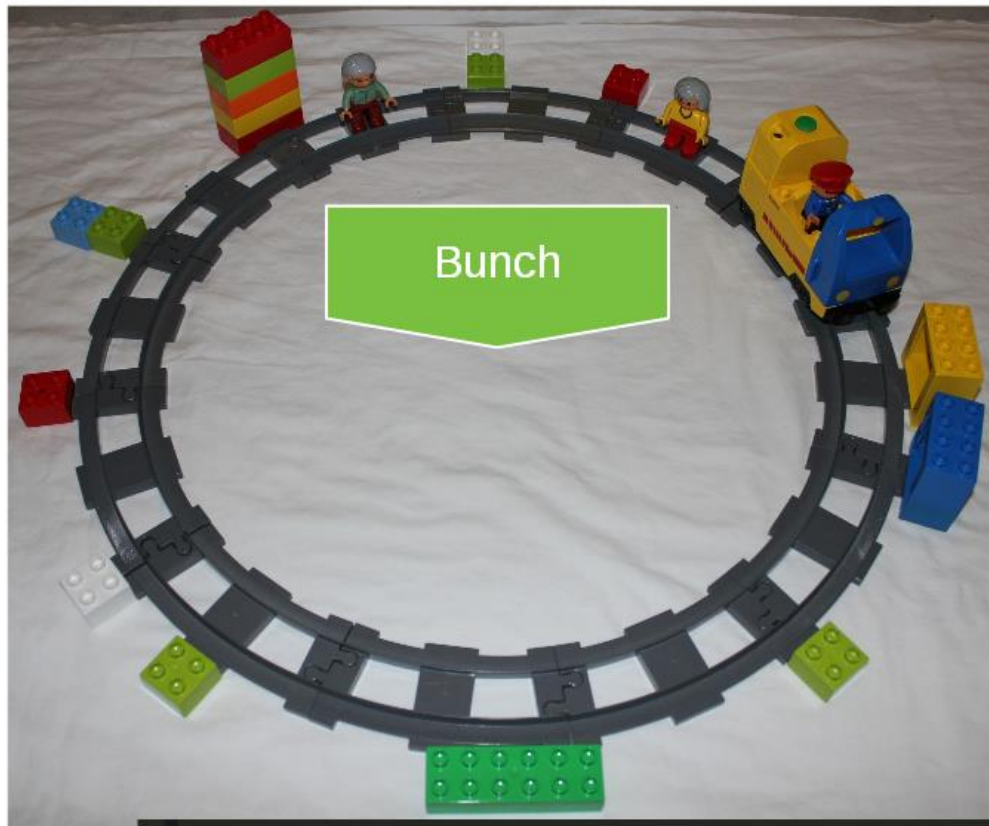


introduction to PyHEADTAIL



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)  
62
```

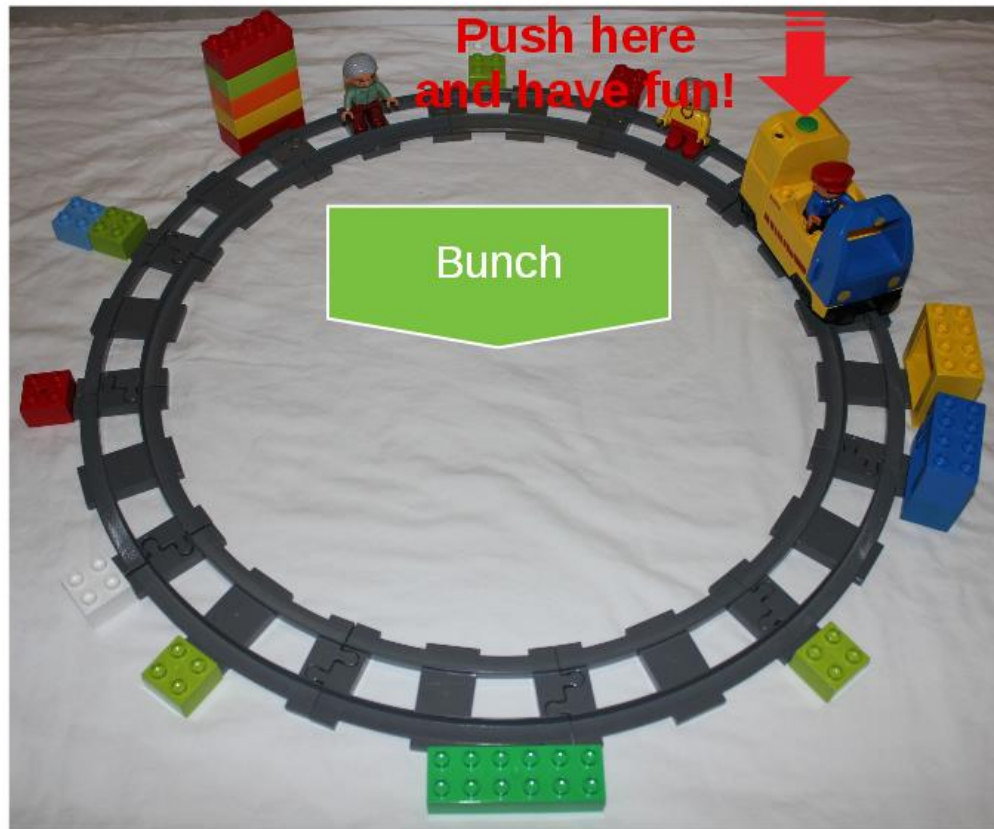


introduction to PyHEADTAIL



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

```
86 for i in range(n_turns):  
87     for m in map_:  
88         m.track(bunch)  
89
```



In our case: only RF systems

JUAS

Tracking

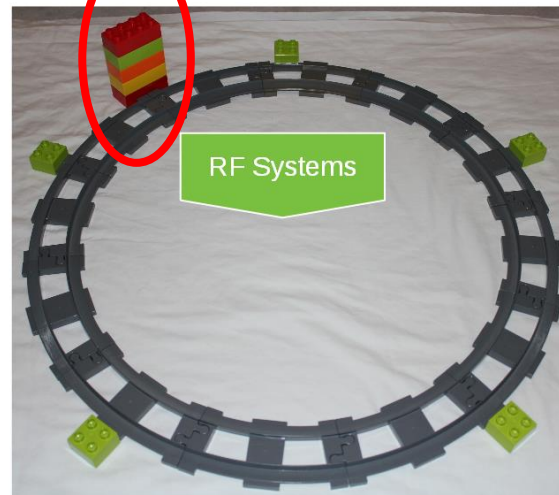
- ◆ The motion of the particles can be tracked turn by turn using the recurrence relation (between turn n and turn $n+1$)

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

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

JUAS - Jan 2016 - E.Métral

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

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

# machine parameters
circumference = 1000*np.pi
inj_alpha_x = 0e-1.2
inj_alpha_y = 0e15
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 = 5
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_alpha_x),
                                   distribution_y=generators.gaussian2D(epsn_y, alpha_y=inj_alpha_y),
                                   distribution_z=generators.RF_bucket_distribution(rfsystems)
                                   ).generate()
```

Define RF system

Generate a matched distribution corresponding to these parameters

Loop over number of turns

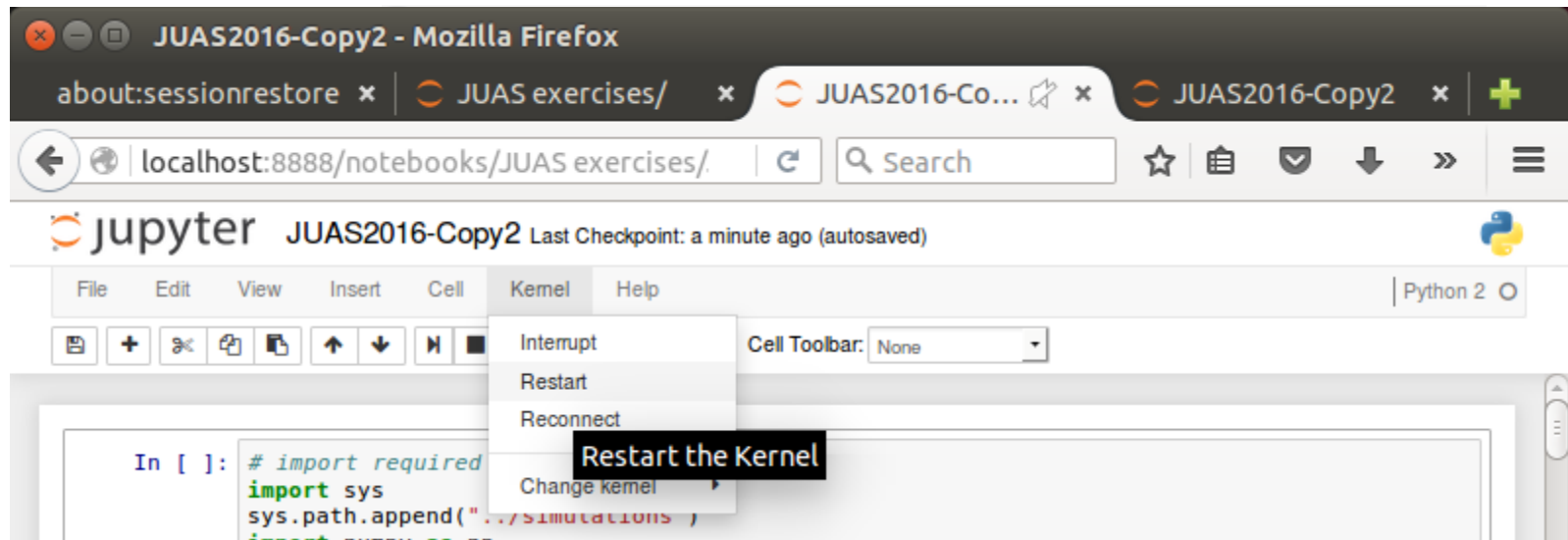
```
In [ ]: # plot phase space
plt.close()
plt.ion()
fig = plt.figure(1)
for i in np.arange(0, 50, 1):
    # track the particles
    rfsystems.track(bunch)
    if i%10 == 0:
        # monitor the particles
        bucket = rfsystems.get_bucket(gamma=bunch.gamma)
        # plot the RF bucket envelope
        z = np.linspace("bucket.interval", num=100)
        dp = np.linspace(-0.005, 0.005, num=100)
        Z2, DPP = np.meshgrid(z, dp)
        HH = bucket.hamiltonian(Z2, DPP)
        plt.contour(Z2, DPP, HH, levels=[0], colors='magenta')

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

track
monitor
plot

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: $\epsilon_{x,y}^*$ [μm]	5
Trans. average betatron function: $\beta_{x,y}$ [m]	16
Beam pipe [cm \times cm]	3.5 \times 7
Trans. tunes: $Q_{x,y}$	6.25

20 kV at injection

$\Rightarrow \gamma_t \approx 6.1$

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: $\epsilon_{x,y}^*$ [μm]	5
Trans. average betatron function: $\beta_{x,y}$ [m]	16
Beam pipe [cm \times cm]	3.5 \times 7
Trans. tunes: $Q_{x,y}$	6.25

$E_{kin} = 1.4e9 \# \text{ in [eV]}$

$\text{circumference} = 100 * 2 * \text{np.pi}$

$\text{bending_radius} = 70 \# \text{ in m}$

$\text{Bdot} = 0 \# \text{ in T/s}$

20 kV at injection

$\text{V_rf} = 20e3 \# \text{ in [V]}$

$\Rightarrow \gamma_t \approx 6.1$

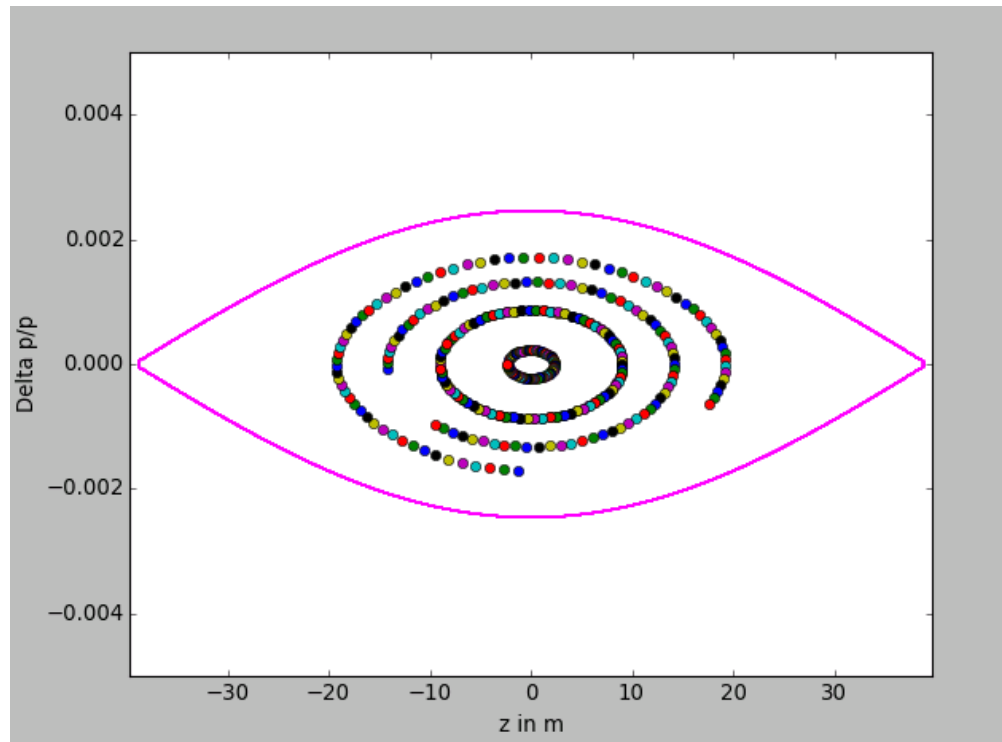
$\text{harmonic} = 8$

$\text{gamma_tr} = 6.1$

$\text{sigma_z_0} = 230e-9 / 4 * \text{beta} * c$

Let's run that example

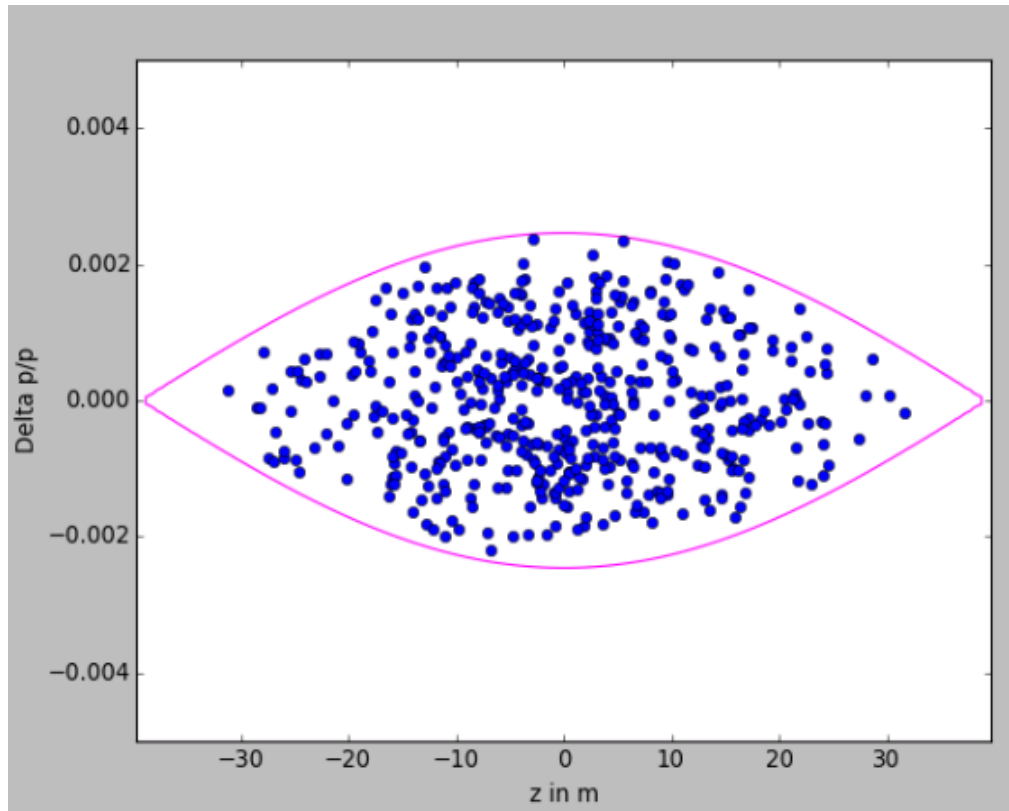
- Use a small number of particles ($n_{\text{particles}} \sim 10$) to observe the synchrotron motion at large and small amplitudes



→ Can you estimate the synchrotron tune Q_s ? 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

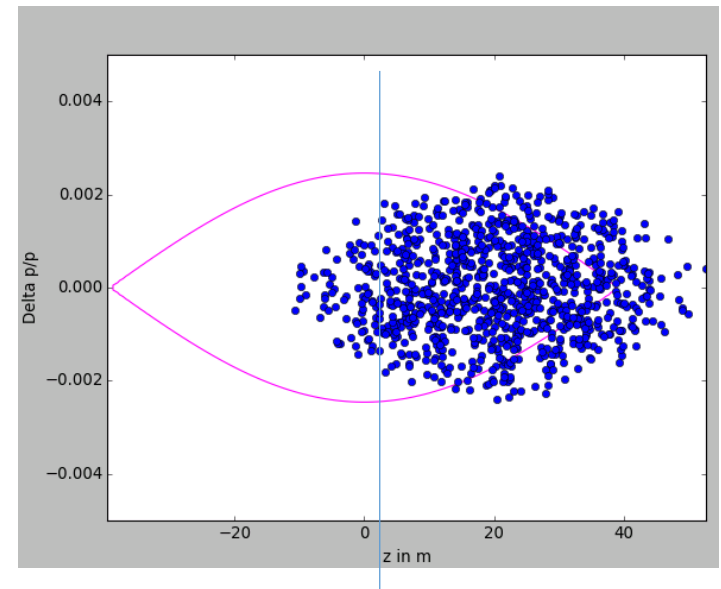
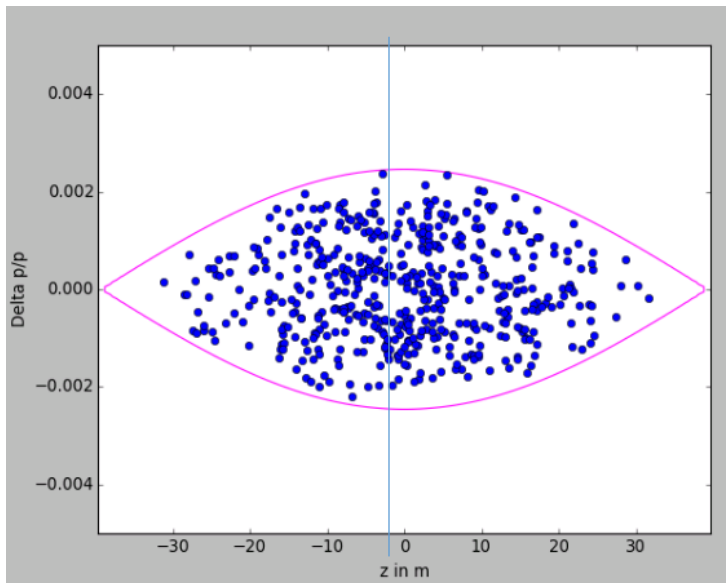
- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- Tracking examples
 - Impact of phase mismatch
 - Impact of voltage mismatch
 - Impact of acceleration
- Setting up the environment
 - Virtual box
 - Ubuntu
 - Anaconda
 - PyHEADTAIL

Impact of phase mismatch

- Generate ~ 1000 particles
- Add e.g. 5 or 20 m to all particles:
- Track particles for ~ 5000 turns
- Plot the distribution every 100 turns:

```
bunch.z = bunch.z+5
```

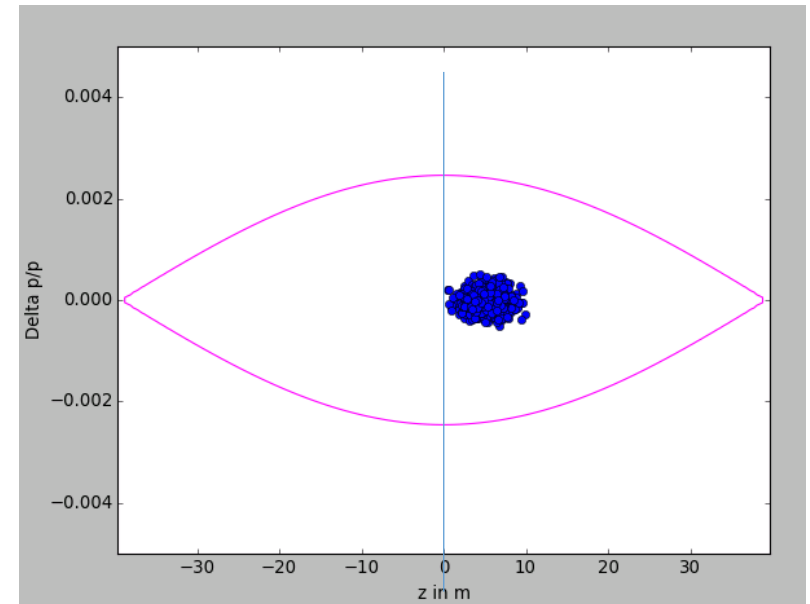
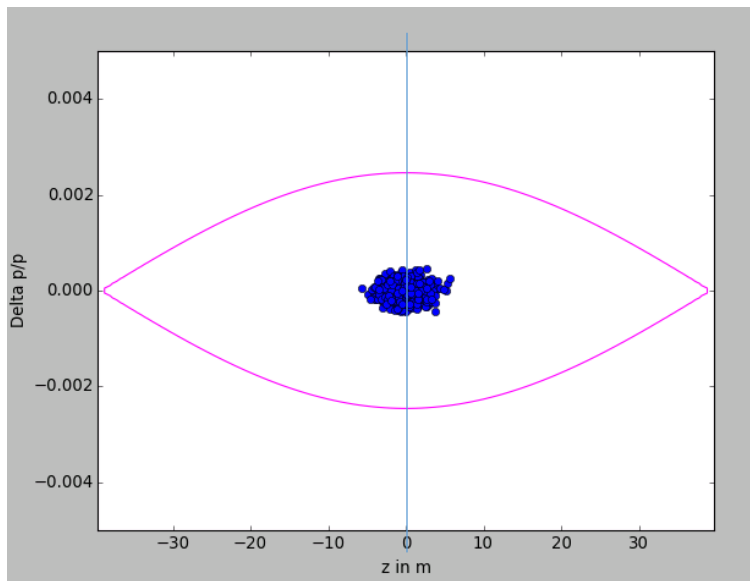
```
if i%100 == 0:
```



→ What will happen?

Impact of phase mismatch

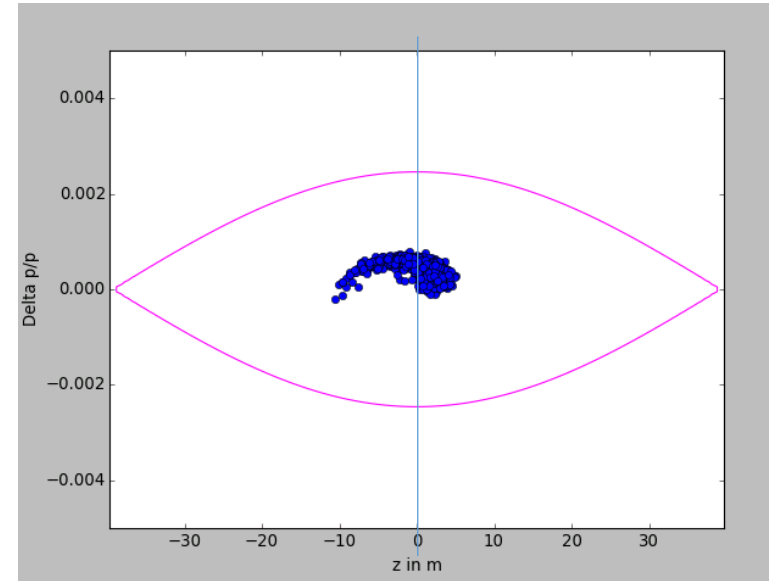
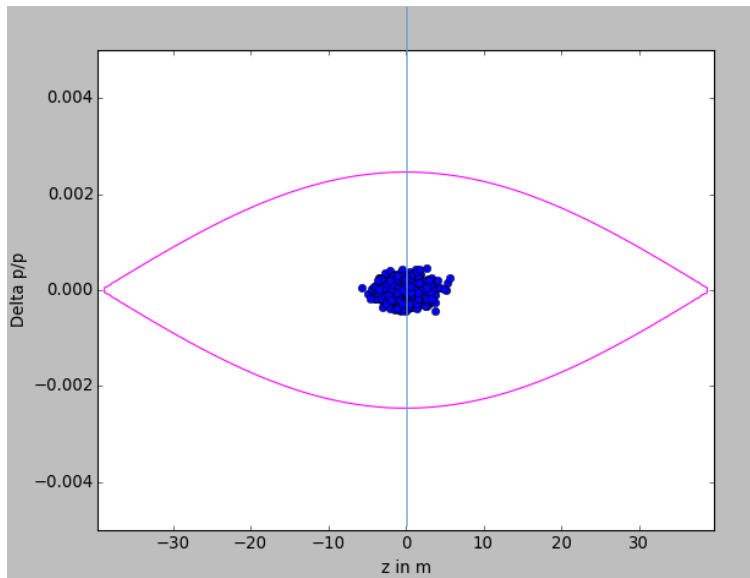
- Use a smaller bunch length (e.g. divide by 10)
- Generate ~ 1000 particles
- Add e.g. 5m to all particles: `bunch.z = bunch.z+5`
- Track particles for ~ 5000 turns
- Plot the distribution every 100 turns: `if i%100 == 0:`



→ What will happen?

Impact of phase mismatch

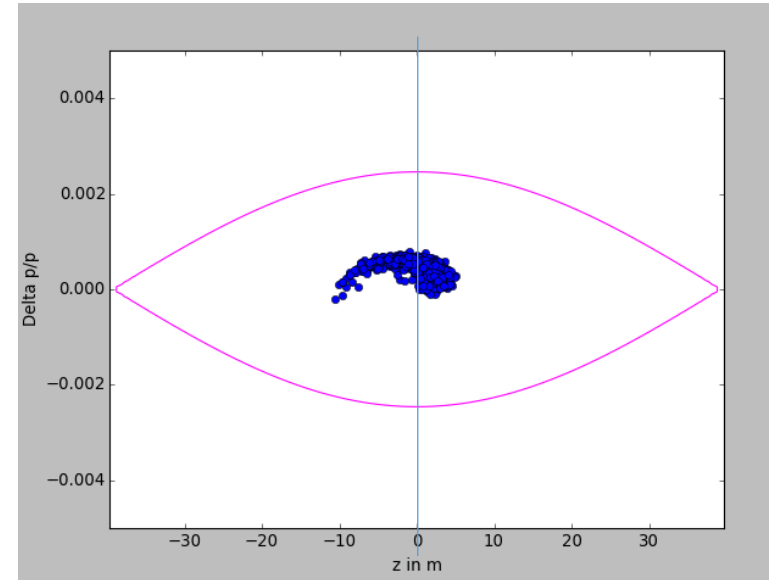
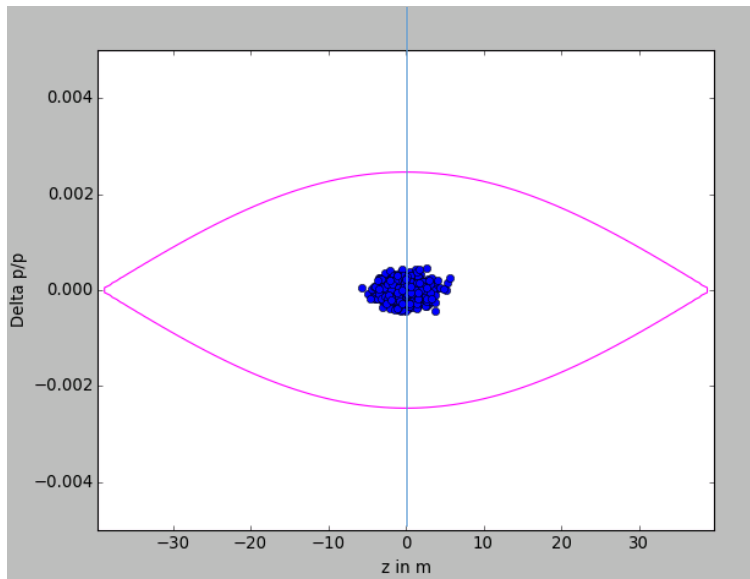
- Use a smaller bunch length (e.g. divide by 10)
- Use ~1000 particles
- Add e.g. 5m to all particles: `bunch.z = bunch.z+5`
- Track particles for ~5000 turns
- Plot the distribution every 100 turns: `if i%100 == 0:`



→ Filamentation and eventually?

Impact of phase mismatch

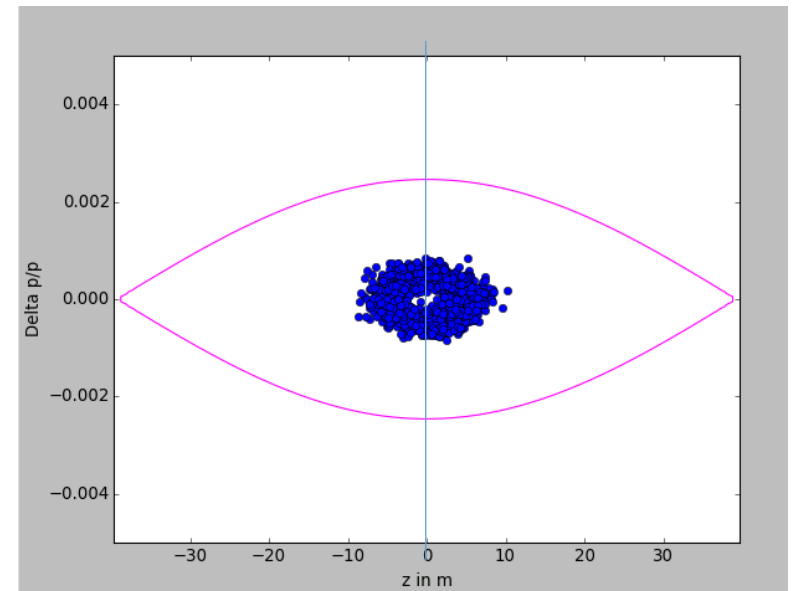
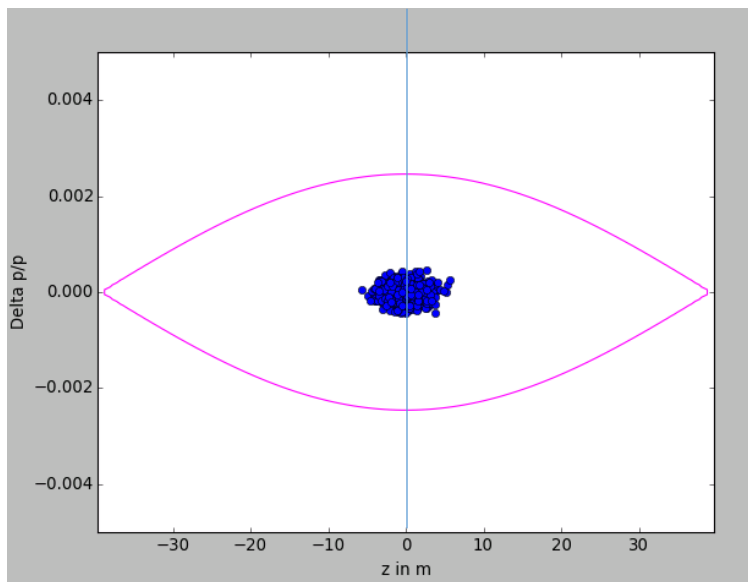
- Use a smaller bunch length (e.g. divide by 10)
- Use ~ 1000 particles
- Add e.g. 5m to all particles:
- Track particles for ~ 500000 turns
- Plot the distribution every 1000 turns



→ 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)

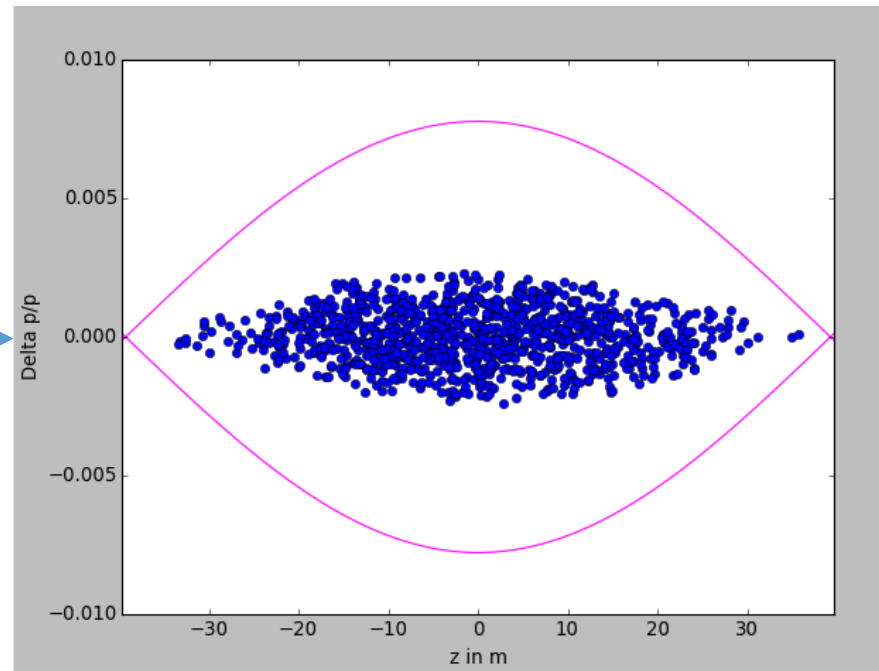
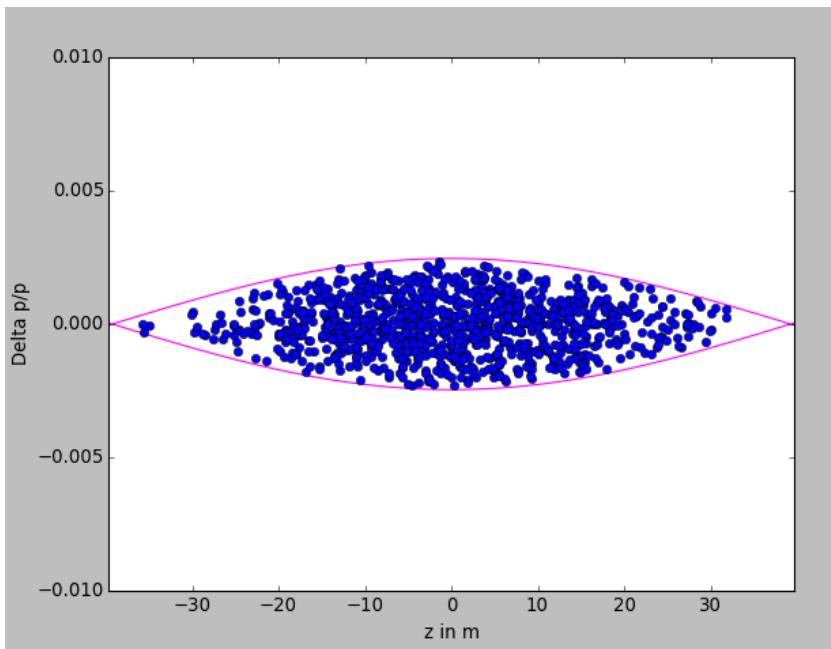
Agenda

- 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)
```

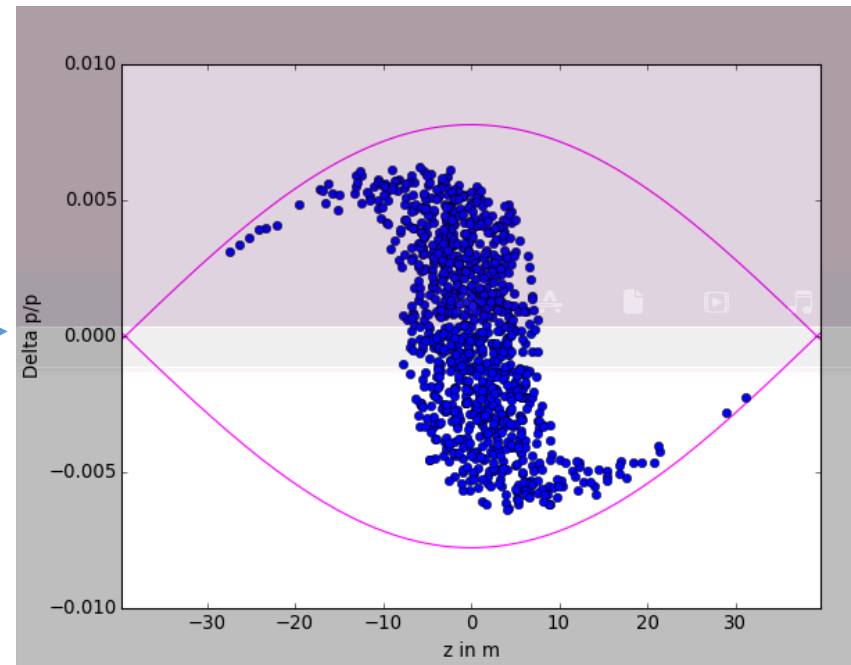
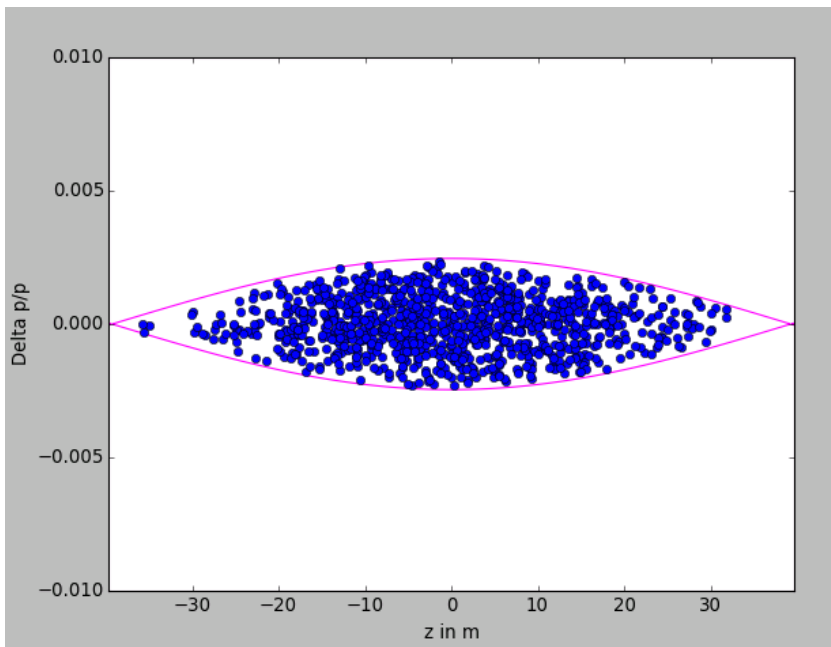


→ What will happen?

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)
```



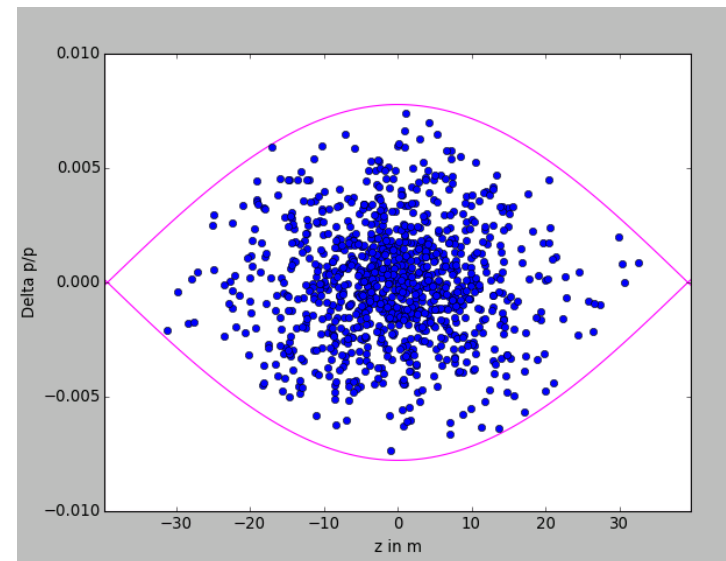
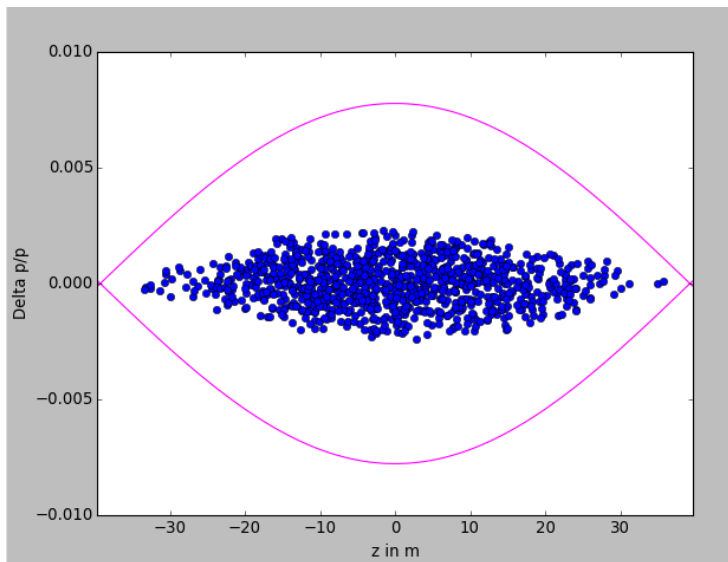
→ Bunch rotation

→ 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)
```

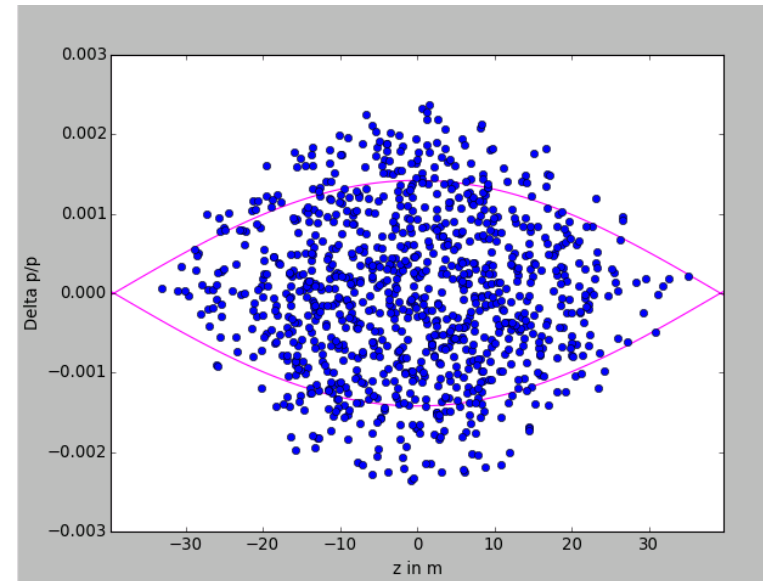
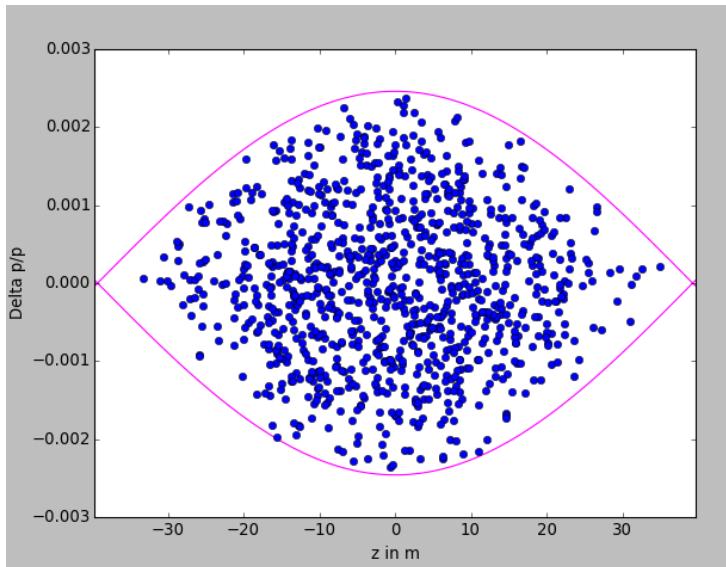


→ But should not wait too long!

→ Emittance growth!

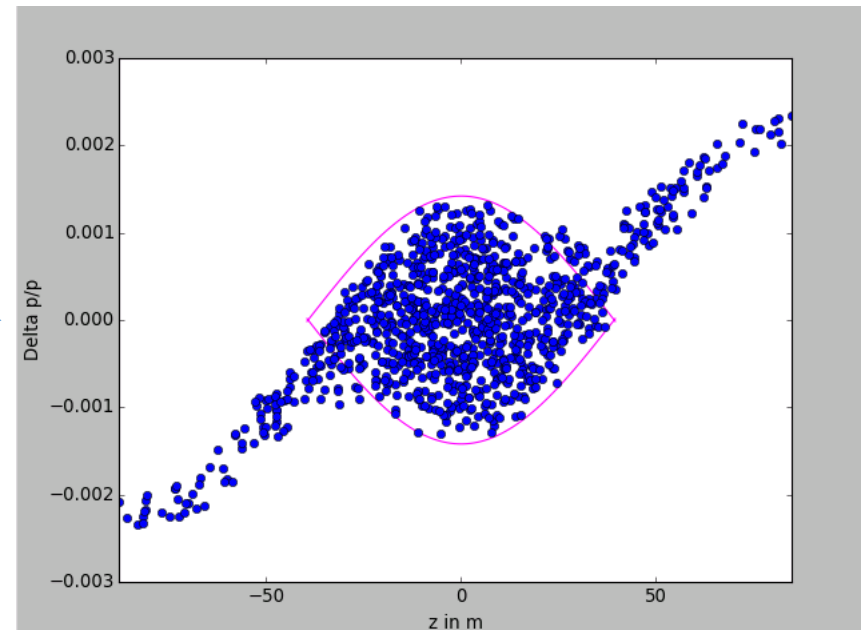
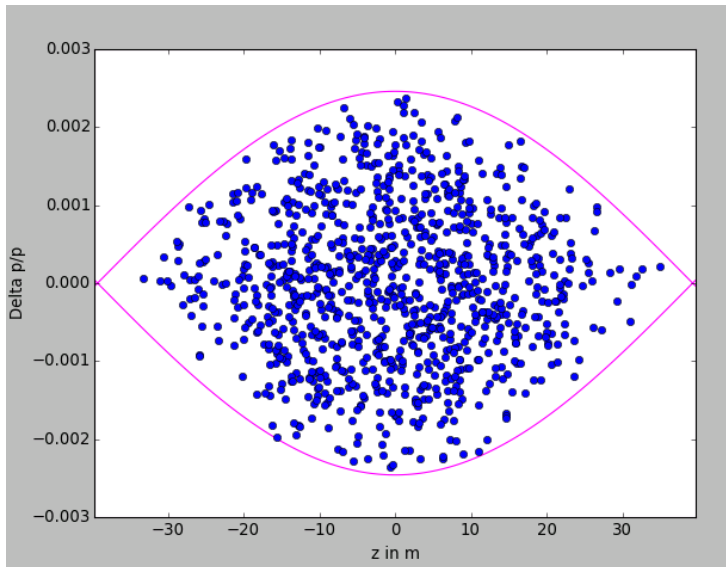
Impact of voltage mismatch (too low)

- 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

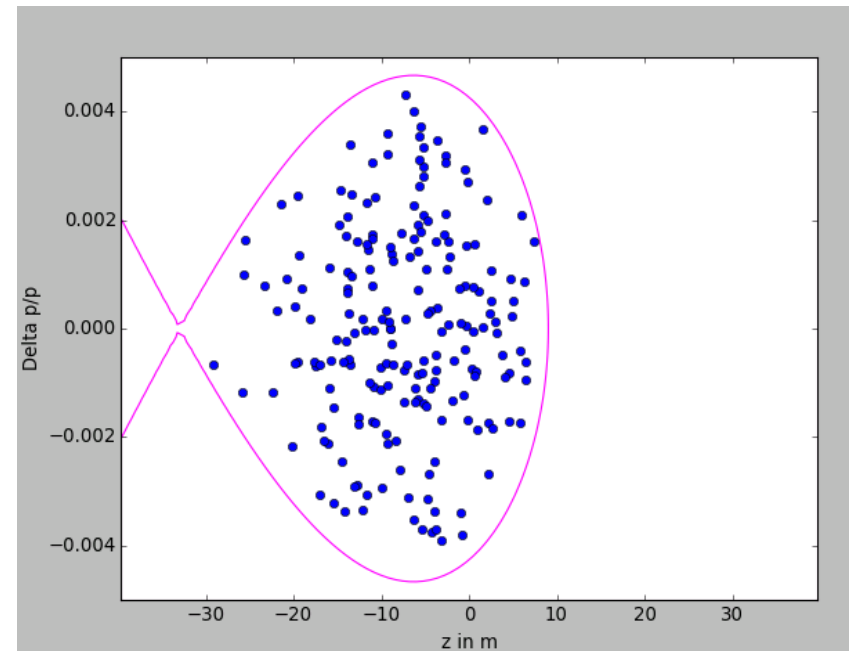
- Goals
- Plan
- Introduction to PyHEADTAIL
- Structure of ipython files
- **Tracking examples**
 - Impact of phase mismatch
 - Impact of voltage mismatch
 - **Impact of acceleration**
- Setting up the environment
 - Virtual box
 - Ubuntu
 - Anaconda
 - PyHEADTAIL

Impact of acceleration

- Use the nominal bunch length
- Use 200 particles
- Use $\dot{B}=2.2 \text{ T/s}$
- Does it work? Why?

Impact of acceleration

- Use the nominal bunch length
- Use 1000 particles
- Use $B\dot{=}2.2$ T/s
- Change VRF to 200 kV
- Compute the synchronous phase
- Plot the phase space 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>



VirtualBox

Download VirtualBox

Here you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

If you're looking for the latest VirtualBox 6.0 packages, see [VirtualBox 6.0 builds](#). Please also use version 6.0 if you need to run VMs with software virtualization, as this has been discontinued in 6.1. Version 6.0 will remain supported until July 2020.

If you're looking for the latest VirtualBox 5.2 packages, see [VirtualBox 5.2 builds](#). Please also use version 5.2 if you still need support for 32-bit hosts, as this has been discontinued in 6.0. Version 5.2 will remain supported until July 2020.

VirtualBox 6.1.2 platform packages

- [Windows hosts](#)
- [OS X hosts](#)
- [Linux distributions](#)
- [Solaris hosts](#)

The binaries are released under the terms of the GPL version 2.

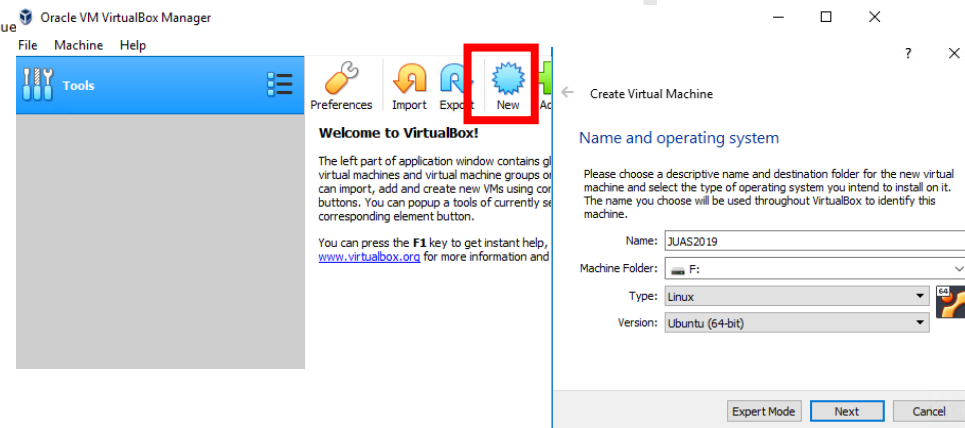
See the [changelog](#) for what has changed.

You might want to compare the checksums to verify the integrity of downloaded packages. *The SHA256 checksums should be favored as the MD5 algorithm must be treated as insecure!*

- [SHA256 checksums, MD5 checksums](#)

Note: After upgrading VirtualBox it is recommended to upgrade the guest

- Create a new virtual machine with Ubuntu 64-bit



Oracle VM VirtualBox Manager

File Machine Help

Tools

Preferences Import Export **New** Add

Welcome to VirtualBox!

The left part of application window contains grid of virtual machines and virtual machine groups or can import, add and create new VMs using corresponding buttons. You can popup a tools of currently selected element button.

You can press the **F1** key to get instant help, www.virtualbox.org for more information and


Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

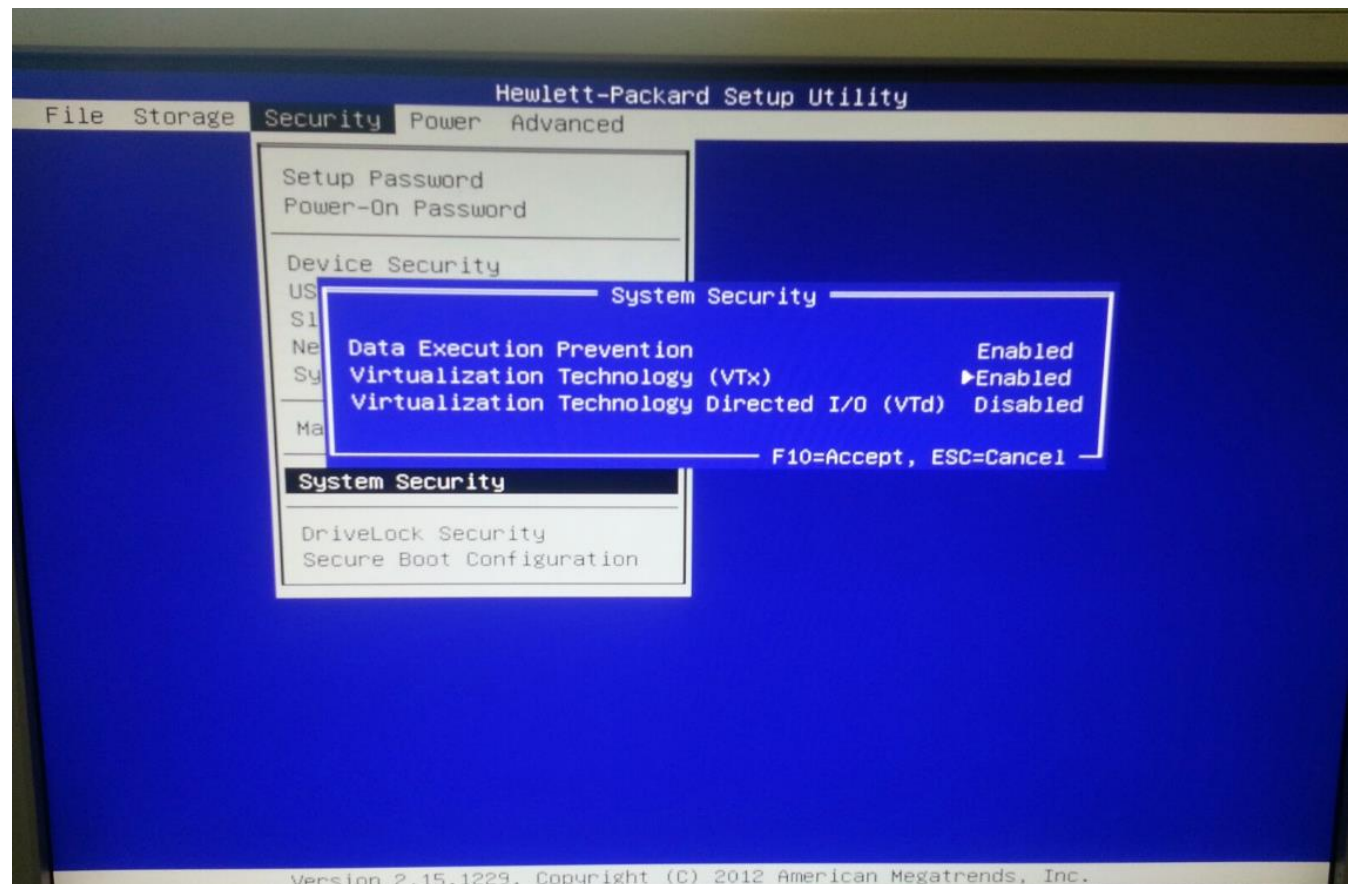
Machine Folder:

Type: 


Version:

Expert Mode

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




← Create Virtual Machine

Name and operating system

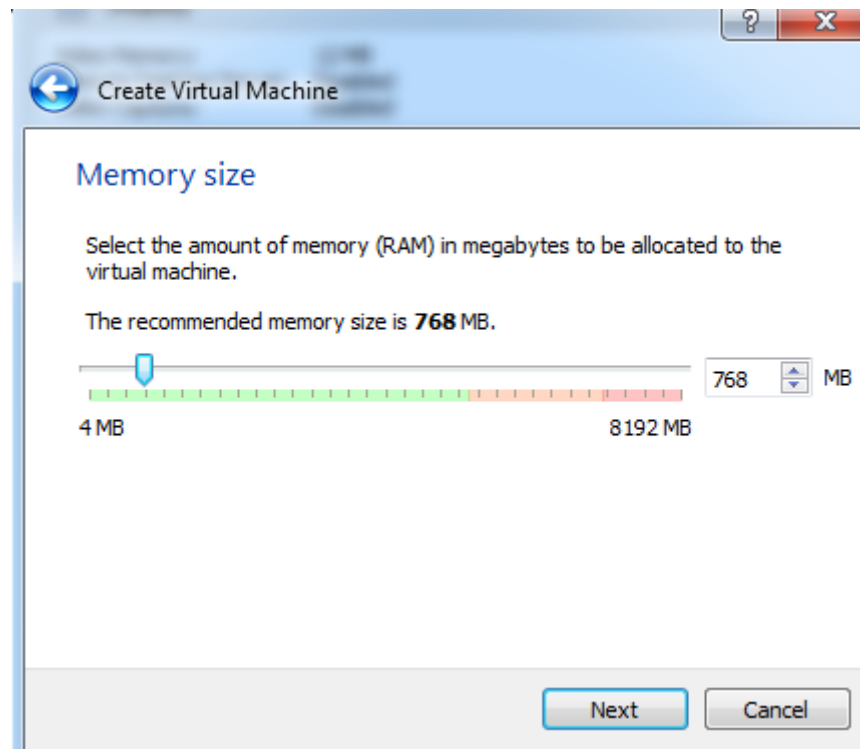
Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

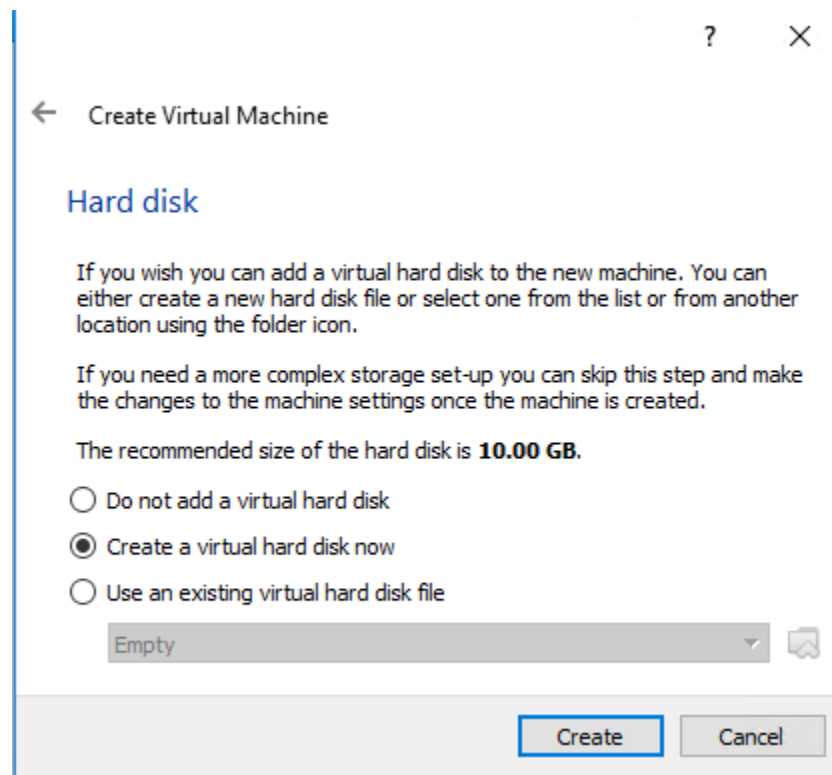
Machine Folder:

Type: 

Version:



→ 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).

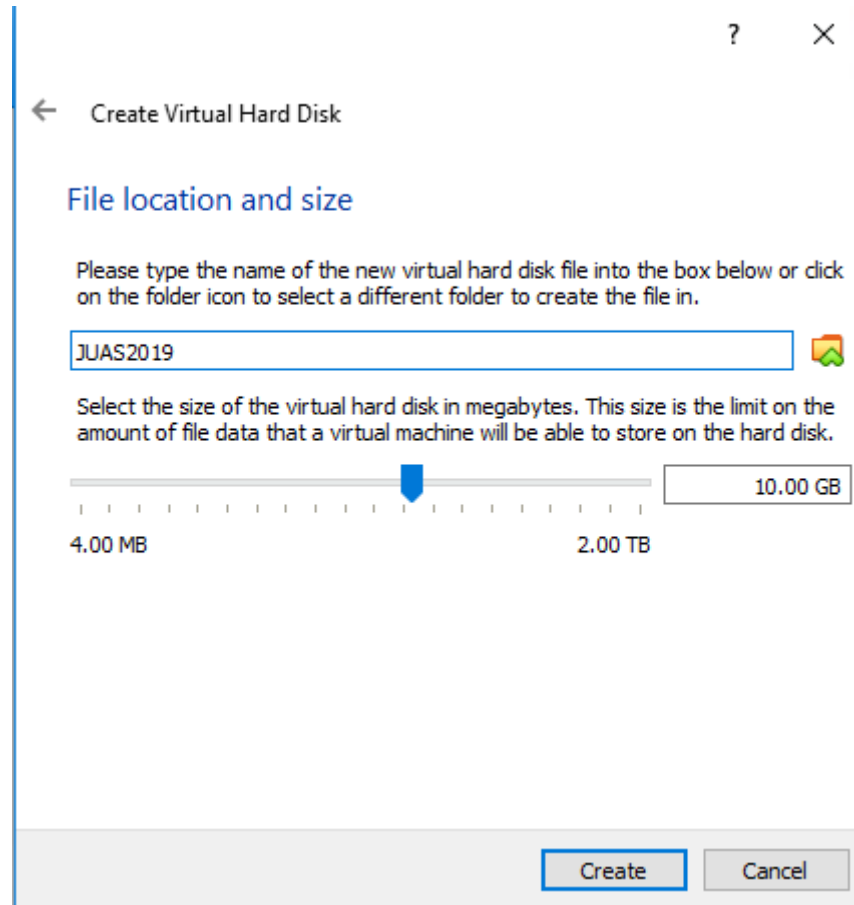
A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- Dynamically allocated
- Fixed size

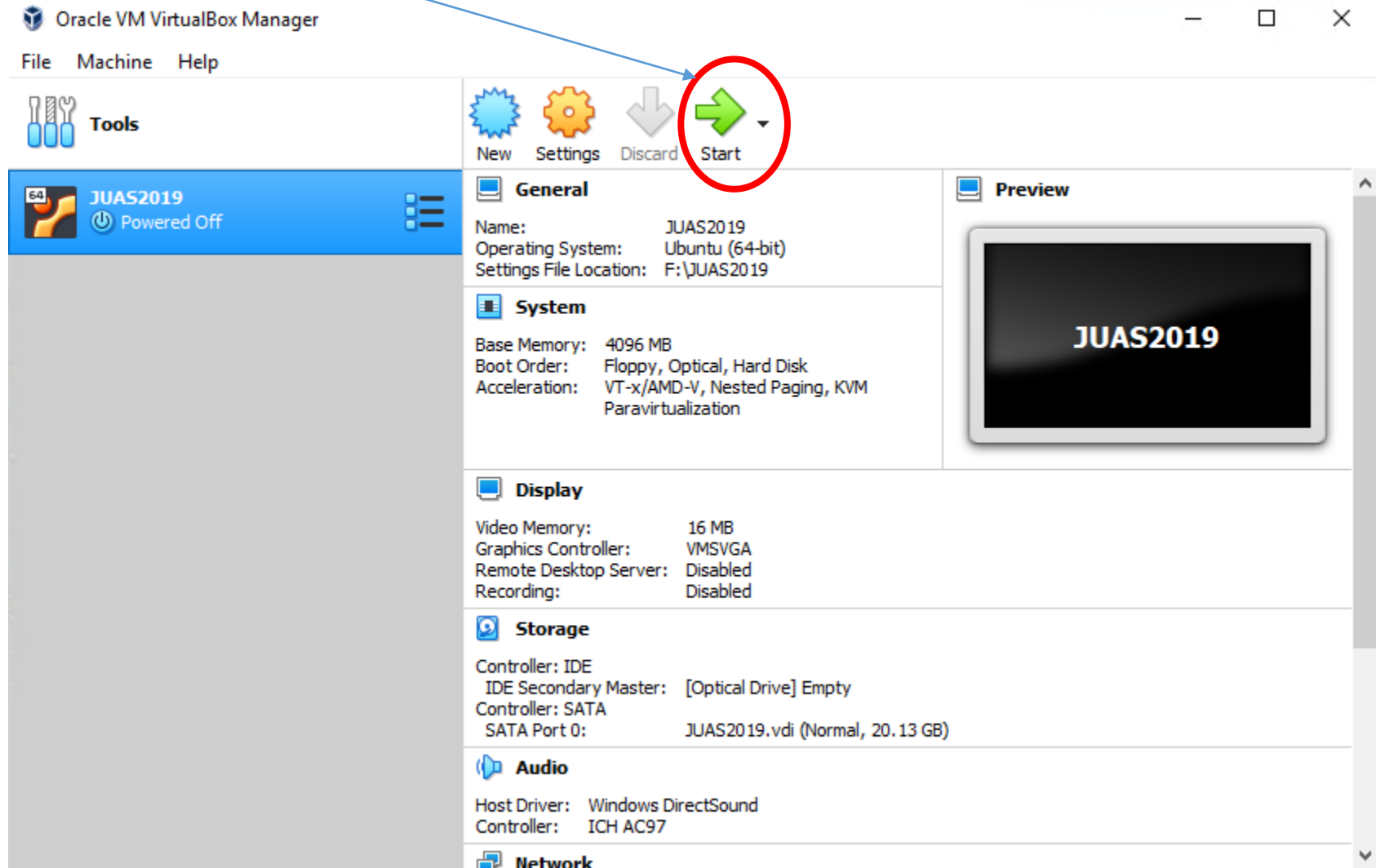
Next

Cancel



→ Assign 20 GB if possible.

Start the VM



The screenshot shows the Oracle VM VirtualBox Manager interface. The window title is "Oracle VM VirtualBox Manager". The menu bar includes "File", "Machine", and "Help". The "Tools" section contains icons for "New", "Settings", "Discard", and "Start". The "Start" button, represented by a green arrow, is circled in red. A blue arrow points from the text "Start the VM" to this button. The main area displays the configuration for a VM named "JUAS2019", which is currently "Powered Off". The configuration is organized into several sections: "General", "System", "Display", "Storage", "Audio", and "Network". The "Preview" window shows a black screen with the text "JUAS2019" in white.

Oracle VM VirtualBox Manager

File Machine Help

Tools

New Settings Discard Start

JUAS2019
Powered Off

General
Name: JUAS2019
Operating System: Ubuntu (64-bit)
Settings File Location: F:\JUAS2019

System
Base Memory: 4096 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

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: JUAS2019.vdi (Normal, 20.13 GB)

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

Network

Preview
JUAS2019

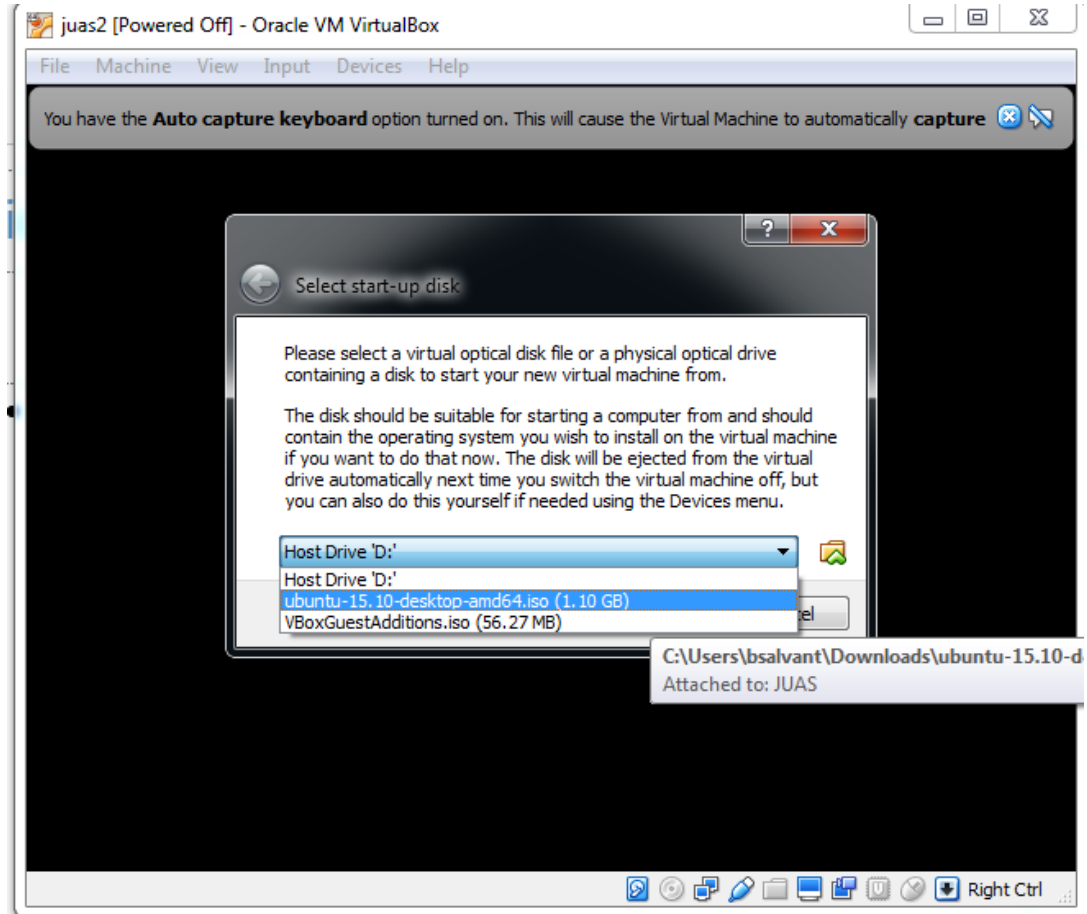
Agenda

- 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

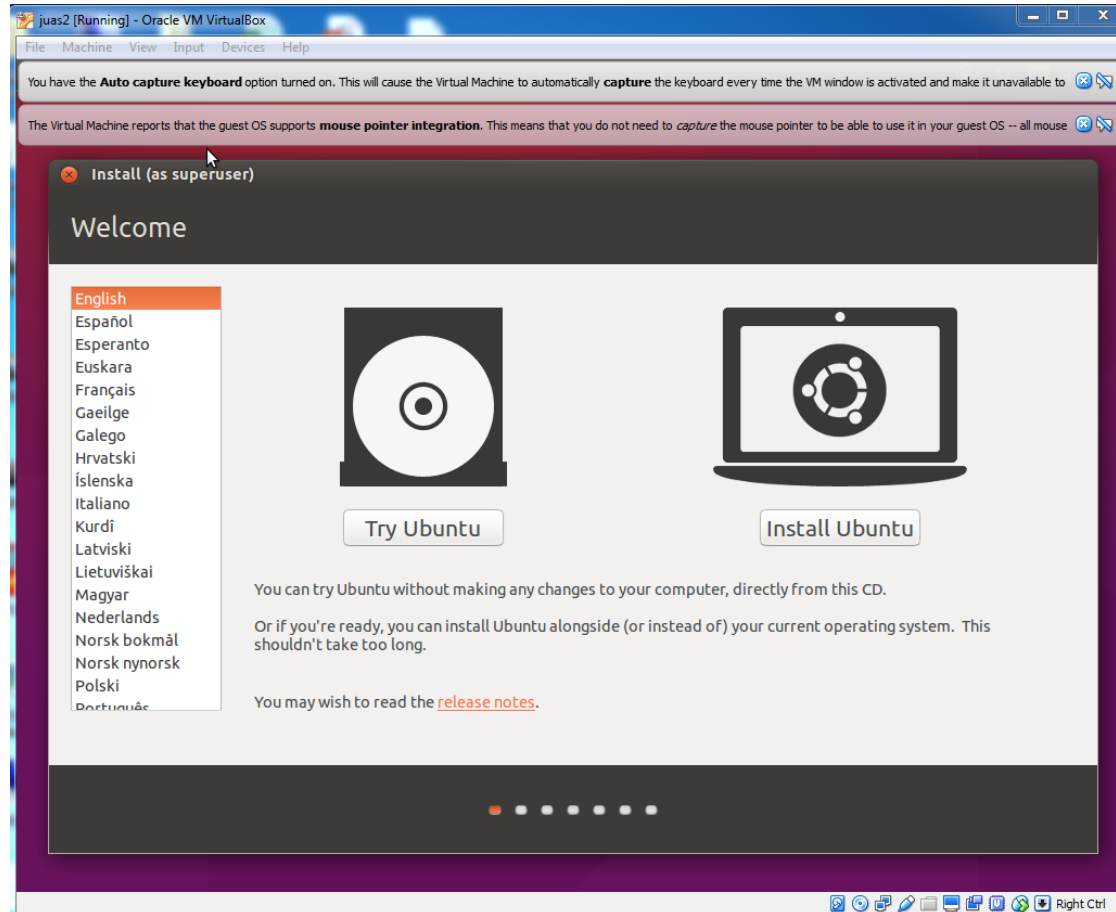
- <https://www.ubuntu.com/download/desktop>

Select Ubuntu start up disk



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

Install ubuntu



JUAS2020 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Fri 17:06

Install

Updates and other software

What apps would you like to install to start with?

Normal installation
Web browser, utilities, office software, games, and media players.

Minimal installation
Web browser and basic utilities.

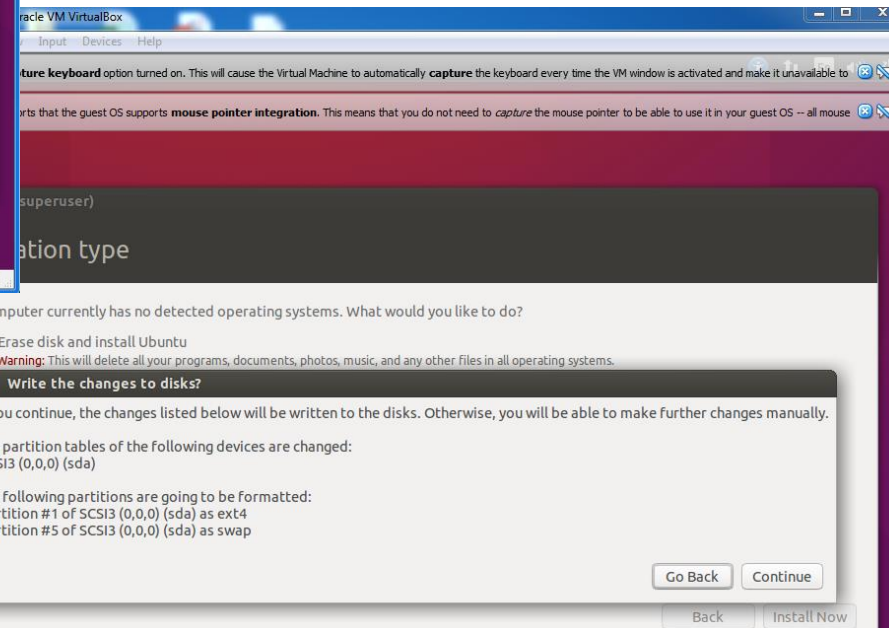
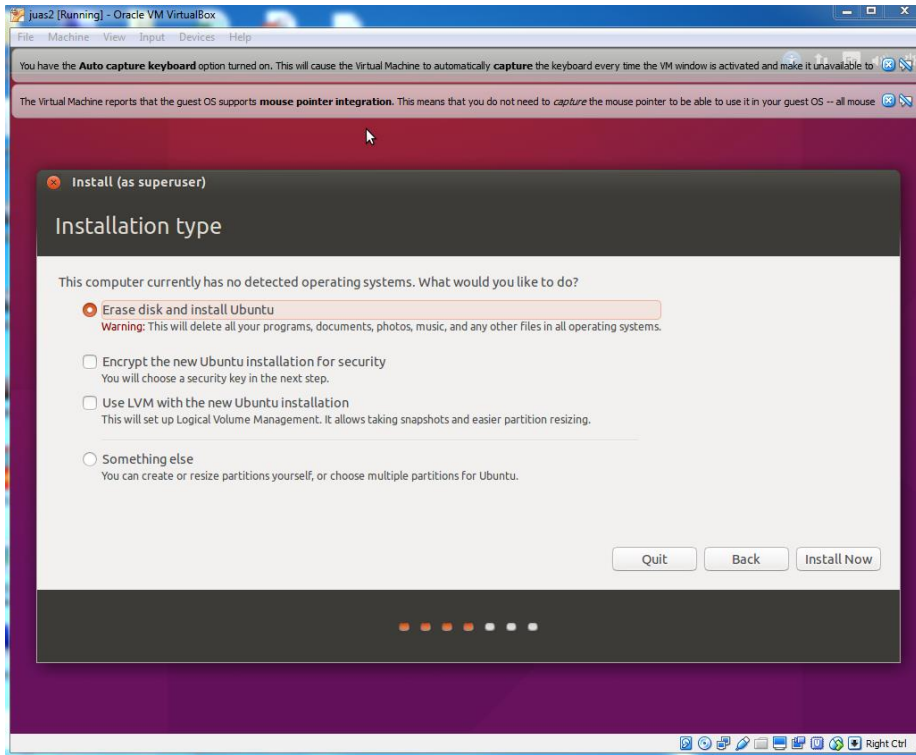
Other options

Download updates while installing Ubuntu
This saves time after installation.

Install third-party software for graphics and Wi-Fi hardware and additional media formats
This software is subject to license terms included with its documentation. Some is proprietary.

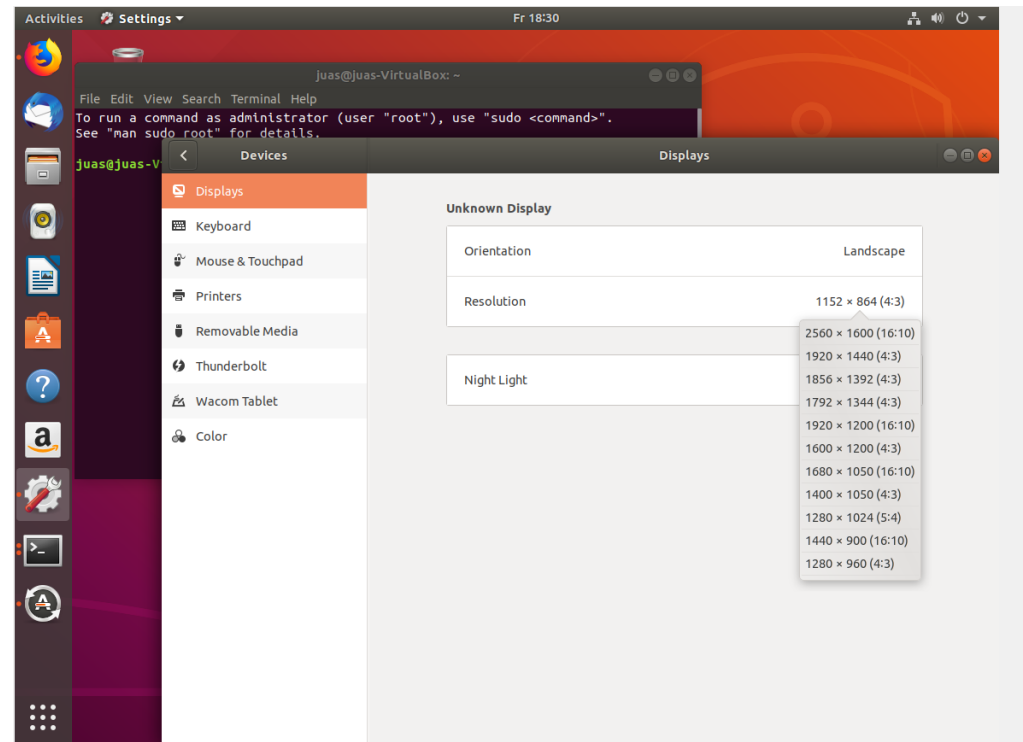
Quit Back Continue

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 display
settings inside Ubuntu



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

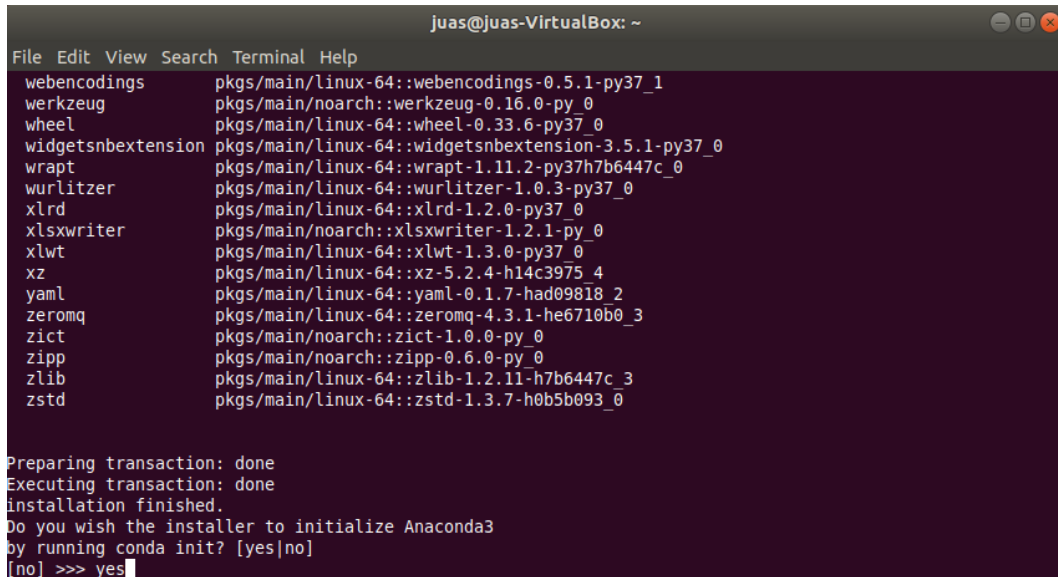
```
> wget https://repo.anaconda.com/archive/Anaconda3-2019.10-Linux-x86\_64.sh
```

→ Then install it by typing

```
> sh Anaconda3-2019.0-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)



```
Juas@Juas-VirtualBox: ~  
File Edit View Search Terminal Help  
webencodings pkgs/main/linux-64::webencodings-0.5.1-py37_1  
werkzeug pkgs/main/noarch::werkzeug-0.16.0-py_0  
wheel pkgs/main/linux-64::wheel-0.33.6-py37_0  
widgetsnbextension pkgs/main/linux-64::widgetsnbextension-3.5.1-py37_0  
wrapt pkgs/main/linux-64::wrapt-1.11.2-py37h7b6447c_0  
wurlitzer pkgs/main/linux-64::wurlitzer-1.0.3-py37_0  
xlrd pkgs/main/linux-64::xlrd-1.2.0-py37_0  
xlsxwriter pkgs/main/noarch::xlsxwriter-1.2.1-py_0  
xlwt pkgs/main/linux-64::xlwt-1.3.0-py37_0  
xz pkgs/main/linux-64::xz-5.2.4-h14c3975_4  
yaml pkgs/main/linux-64::yaml-0.1.7-had09818_2  
zeromq pkgs/main/linux-64::zeromq-4.3.1-he6710b0_3  
zict pkgs/main/noarch::zict-1.0.0-py_0  
zipp pkgs/main/noarch::zipp-0.6.0-py_0  
zlib pkgs/main/linux-64::zlib-1.2.11-h7b6447c_3  
zstd pkgs/main/linux-64::zstd-1.3.7-h0b5b093_0  
  
Preparing transaction: done  
Executing transaction: done  
Installation finished.  
Do you wish the installer to initialize Anaconda3  
by running conda init? [yes|no]  
[no] >>> yes
```

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

→ Check that “>>python 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 Indico site in the last tutorial of the course:

The screenshot shows the Indico website for the JUAS 2020 event. The browser address bar indicates the URL is <https://indico.cern.ch/event/850755/timetable/>. The page displays the timetable for Monday, 3 February. The events listed are:

- 16:00 → 17:00: Bus leaves at 17:00 from ESRF (1h)
- 09:00 → 10:00: Longitudinal dynamics (Speakers: Benoit Salvant (CERN), Elias Metral (CERN - BE/ABP)) (1h)
- 10:00 → 10:15: Coffee Break (15m)
- 10:15 → 12:15: Longitudinal dynamics (Speakers: Benoit Salvant (CERN), Elias Metral (CERN - BE/ABP)) (2h)
- 12:15 → 14:00: WORKING LUNCH (1h 45m)
- 14:00 → 16:00: Space charge (Speaker: Mauro Migliorati (University of Rome "LA SAPIENZA")) (2h)
- 16:00 → 16:15: Coffee Break (15m)
- 16:15 → 17:15: Space charge (Speaker: Mauro Migliorati (University of Rome "LA SAPIENZA")) (1h)

A red oval highlights a row of five file download links under the 10:15 → 12:15 Longitudinal dynamics session:

- 1_JUAS2017_inject...
- 2_JUAS2017_inject...
- 3_JUAS2017_inject...
- 4_JUAS2017_inject...
- 4_JUAS2017_inject...

→ Download the files and open them browsing with “ipython notebook”

Changing variables

JUAS

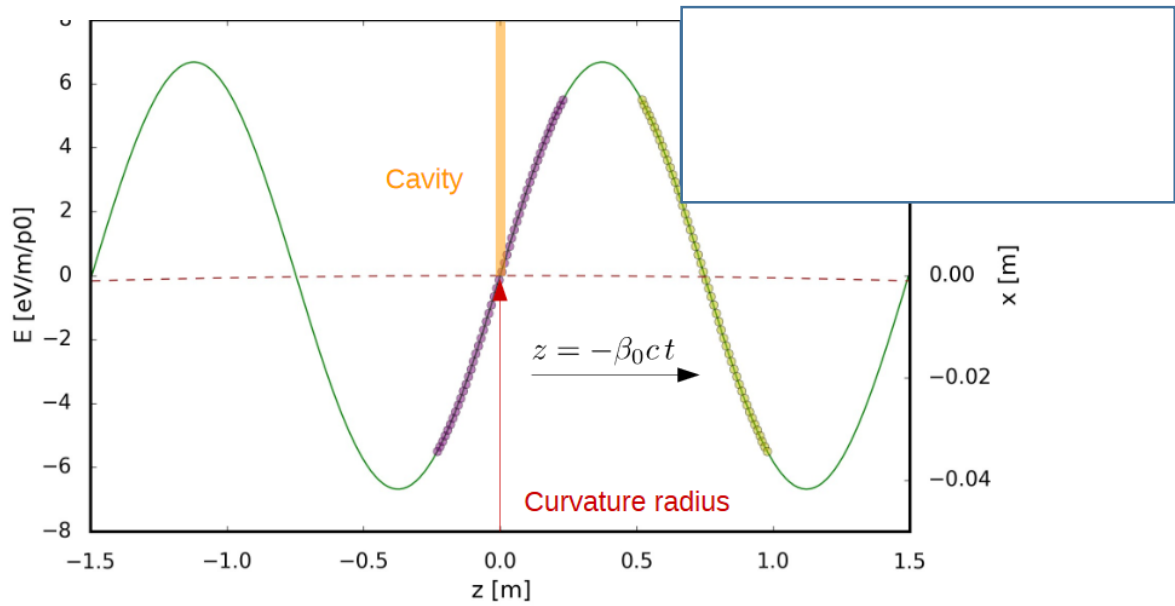
Johns Hopkins Accelerator School

- Change of variables if one wants to use $(\Phi, \Delta E)$ or $(\Delta t, \Delta E)$ instead of $(\Phi, d\Phi/dt)$

$$\begin{aligned}\Delta\phi &= \phi - \phi_s \\ &= \omega_{RF} \Delta t \\ &= h \omega_s \Delta t\end{aligned}$$

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

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



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

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

For protons

$$\Delta E[\text{GeV}] = \frac{\Delta p}{p} 0.938 \gamma \beta^2$$