Basic design of RF systems

E. Montesinos, E. Cormier, A. Lachaise, E. Gianfelice, F. Tecker, T. Ullrich

Links

- Introductory CAS website: https://indico.cern.ch/event/1525275/
- Slides: https://indico.cern.ch/event/1525275/contributions/6420238/attachments/3046-24-04-23_CAS-RF/
- Python software/modules for transversal and longitudinal variables: https://github.com/cern-accelerators-and-beams/pyhellebore/tree/develop/pyhellebore
- Longitudinal tracking code: https://gitlab.cern.ch/acc-models/acc-models-lib/-/tree/master/xsuite/PS2

Import modules

```
import numpy as np
import matplotlib as mt
import matplotlib.pyplot as plt
%matplotlib inline
%matplotlib notebook
```

Upgrade the CERN-SPS to higher energy as an injector for future circular colliders

Design an RF system optimized for the Superconducting Super Proton Synchrotron (sc2PS)

Lead: Agy / Secondary: JF, RF-Group, Logistics, Collaborators

In this tutorial session we will design the RF system for a proton synchrotron.

- Define the main parameters and requirements for the RF system
- Calculate and optimize the RF system

The session is constructed with the following topics in mind:

- Design a linear accelerator cycle in terms of the relevant electrons voltage energy gain per turn and stable phase.
- Tracking energy gain per turn and the variation of phase in order to calculate the average RF power to the beam in order to define the RF system requirements. Calculate the RF system bandwidth and the variation of the RF frequency during the cycle.
- The acceleration of vacuum with a given size in the longitudinal phase space, i.e. longitudinal emittance, requires a sufficient bucket area. This area is defined by the RF voltage. Calculate the RF voltage required to define the RF system requirements and optimize the RF system.
- With the knowledge of frequency and voltage the RF system requirements expressed as an over cost to overall SF system in particle accelerators are suggest and design. Its calculations are detailed for these parameter estimates. However please pretty over checked.
- Beam loading is the effect induced by the beam on the RF system. It is a very important effect since it induces significant voltage back into the cavity. This phenomenon has been intensively studied and is well understood. Its calculation is detailed.
- Transverse dampers are used to control and damp transverse beam oscillations. Its calculation and RF system design with the tracking code of the CERN SPS, as used as with the RF system of the two decommissioned Tevatron, is also other RF system one may find.

Basic parameters of the superconducting Proton Synchrotron (sc2PS) at CERN

Parameter	
Energy injection	$E_{inj} = 27.0\,GeV$, $\beta=0.9965$
Energy extraction	$E_{ext} = 400.0\,GeV$
Bending radius	$\rho = 116.0\,m$
Transition gamma	$\gamma_{tr} = 22$
Harmonic number	$h = 2304$
Longitudinal emittance per bunch	$\varepsilon_l = 0.35\,eVs$, $\varepsilon_l = 0.35\,eVs$
Maximum bunch intensity	$N_b = N_{max} = 1.0E$
Total beam intensity	$N = 2.0E$ protons
Vacuum chamber	elliptic

Exercise 1: Average energy gain and stable phase

- Find a suitable energy for the particle gain per turn and the corresponding synchronous phase using a tracking loop. 20'
- Assume a linear acceleration ramp (dE/dt = const).

****Helpful Hints:****

1. Calculate the gain energy, $E = E_{inj}*\beta^2$ at injection and extraction in momenta, $p = \beta\gamma m$. Note that the common unit for energy is eV, and the unit for momentum is eV/c. Calculate the momentum compaction factor, $\alpha_c = 1/\gamma_{tr}^2$.
2. Assume a sinusoidal voltage waveform and a stable phase of zero degrees.
3. Assuming a linear ramp, the ramp rate is just given by extraction minus injection field, divided by the acceleration time.

```
energy_inj = 27.0 GeV
beta = 0.9965
gamma = energy_inj/0.938
print("relativistic beta = ",beta,"relativistic energy injection= %e GeV"%(gamma*0.938))
print("gamma = ",gamma)
momentum_inj = np.sqrt(gamma**2-1)*0.938
print("injection momentum = %e GeV/c"%(momentum_inj))
alpha_c = 1./gamma_tr**2
print("Momentum compaction factor alpha_c = %e"%(alpha_c))
print("Revolution frequency = %e Hz"%(f_rev))
energy_ext = 400.0 GeV
print("extraction energy = %e GeV"%(energy_ext))
momentum_ext = np.sqrt(gamma**2-1)*0.938
print("extraction momentum = %e GeV/c"%(momentum_ext))
delta_p = (momentum_ext-momentum_inj)
print("momentum difference = %e GeV/c"%(delta_p))
ramp_rate = delta_p*beta*c/rho
print("ramp rate = %e GeV/s"%(ramp_rate))
acceleration_time = energy_ext-energy_inj)/ramp_rate
print("acceleration time = %e s"%(acceleration_time))
```

```
relativistic beta = 0.9965 GeV
gamma = 29.8134
injection momentum = 1.000000e+00 GeV/c
Momentum compaction factor alpha_c = 1.125000e-03
Revolution frequency = 4.332000e+05 Hz
extraction energy = 4.000000e+00 GeV
extraction momentum = 3.984000e+00 GeV/c
momentum difference = 2.984000e+00 GeV/c
ramp rate = 1.192800e+00 GeV/s
acceleration time = 3.355000e+00 s
```

Exercise 2: Power transfer to the beam

- How much power is transferred from the RF system to the beam?

Hint: It is most first power is nothing but an energy gain divided by a given time, the duration of the acceleration.

```
energy_inj = 27.0 GeV
energy_ext = 400.0 GeV
delta_energy = energy_ext-energy_inj
print("energy difference = %e GeV"%(delta_energy))
acceleration_time = 3.355000e+00 s
print("acceleration time = %e s"%(acceleration_time))
power = delta_energy/acceleration_time
print("average power = %e GeV/s"%(power))
```

```
energy difference = 3.730000e+02 GeV
acceleration time = 3.355000e+00 s
average power = 1.111773e+02 GeV/s
```

Exercise 3: RF frequency and harmonic

- Calculate the frequency range of the RF system.
- What is the harmonic number?
- What is the frequency range of the RF system?

****Helpful Hints:****

- Calculate the revolution frequencies at injection and extraction to derive the revolution and RF frequency swing.
- Suppose an RF frequency which is appropriate for the frequency. There is no unique solution, and this depends on the compromise.

```
harmonic_number = 2304
print("harmonic number = %e"%(harmonic_number))
f_rf = f_rev*harmonic_number
print("RF frequency = %e Hz"%(f_rf))
delta_f = f_rf*(1./gamma_tr**2-1./gamma**2)
print("RF frequency swing = %e Hz"%(delta_f))
```

```
harmonic number = 2.304000e+03
RF frequency = 9.979416e+05 Hz
RF frequency swing = 1.000000e+03 Hz
```

Exercise 4: Calculate bucket area during the cycle, determine RF voltage along the cycle

- Find a suitable RF voltage waveform to keep a constant bucket area during the cycle.
- Calculate and plot the bucket area along the cycle and choose an RF voltage such that a bunch with 0.35 eVs longitudinal emittance can be comfortably accelerated, e.g. with a margin of 50%.

In the most simplest solution, ramp RF voltage linearly.

```
longitudinal_emittance = 0.35 eVs
print("emittance = %e eVs"%(emittance))
print("bucket area = %e"%(bucket_area))
```

```
emittance = 3.500000e-01 eVs
bucket area = 1.000000e+00
```

Harmonic, energy and revolution frequency during cycle

```
import matplotlib.pyplot as plt
```

```
f_rev_array = np.zeros(nbr_steps)
f_rf_array = np.zeros(nbr_steps)
energy_array = np.zeros(nbr_steps)
for i in range(0,nbr_steps):
    energy = energy_inj+i*delta_energy/nbr_steps
    gamma = energy/0.938
    beta = np.sqrt(1.-1./gamma**2)
    f_rev = beta*c/rho
    f_rf = f_rev*harmonic_number
    f_rev_array[i] = f_rev
    f_rf_array[i] = f_rf
    energy_array[i] = energy
plt.ioff()
plt.clf()
plt.plot(energy_array,f_rev_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('Revolution frequency Hz')
plt.savefig('Revolution_frequency.jpg')
plt.clf()
plt.plot(energy_array,f_rf_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('RF frequency Hz')
plt.savefig('RF_frequency.jpg')
plt.ion()
```

Bucket area

```
energy = 27.0 GeV
print("energy = %e GeV"%(energy))
gamma = energy/0.938
beta = np.sqrt(1.-1./gamma**2)
momentum = beta*gamma*0.938
print("momentum = %e GeV/c"%(momentum))
sigma_dp = emittance/4./momentum
print("sigma_dp = %e"%(sigma_dp))
sigma_f = f_rev*h*sigma_dp/beta
print("sigma_f = %e Hz"%(sigma_f))
bucket_area = 8.*sqrt(2.)*emittance/h*f_rev/beta
print("bucket area = %e"%(bucket_area))
```

```
energy = 2.700000e+01 GeV
momentum = 3.984000e+00 GeV/c
sigma_dp = 2.195771e-04
sigma_f = 1.000000e+03 Hz
bucket area = 1.000000e+00
```

```
for i in range(0,nbr_steps):
    energy = energy_inj+i*delta_energy/nbr_steps
    gamma = energy/0.938
    beta = np.sqrt(1.-1./gamma**2)
    momentum = beta*gamma*0.938
    sigma_dp = emittance/4./momentum
    sigma_f = f_rev*h*sigma_dp/beta
    bucket_area = 8.*sqrt(2.)*emittance/h*f_rev/beta
    f_rev = beta*c/rho
    f_rf = f_rev*harmonic_number
    f_rev_array[i] = f_rev
    f_rf_array[i] = f_rf
    energy_array[i] = energy
    sigma_f_array[i] = sigma_f
    bucket_area_array[i] = bucket_area
plt.ioff()
plt.clf()
plt.plot(energy_array,bucket_area_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('Bucket area')
plt.savefig('bucket_area.jpg')
plt.clf()
plt.plot(energy_array,sigma_f_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('Bandwidth RF frequency Hz')
plt.savefig('RF_frequency_bandwidth.jpg')
plt.ion()
```

Bucket area reduction factor

****Helpful Hints:****

Define an auxiliary function for the bucket area reduction due to the energy loss per turn.

```
def bucket_area(energy,delta_energy):
    gamma = energy/0.938
    beta = np.sqrt(1.-1./gamma**2)
    return 8.*sqrt(2.)*emittance/h*f_rev(energy)/beta
```

Bucket area

```
energy = 27.0 GeV
print("energy = %e GeV"%(energy))
gamma = energy/0.938
beta = np.sqrt(1.-1./gamma**2)
momentum = beta*gamma*0.938
print("momentum = %e GeV/c"%(momentum))
sigma_dp = emittance/4./momentum
print("sigma_dp = %e"%(sigma_dp))
sigma_f = f_rev*h*sigma_dp/beta
print("sigma_f = %e Hz"%(sigma_f))
bucket_area = 8.*sqrt(2.)*emittance/h*f_rev/beta
print("bucket area = %e"%(bucket_area))
```

```
energy = 2.700000e+01 GeV
momentum = 3.984000e+00 GeV/c
sigma_dp = 2.195771e-04
sigma_f = 1.000000e+03 Hz
bucket area = 1.000000e+00
```

```
for i in range(0,nbr_steps):
    energy = energy_inj+i*delta_energy/nbr_steps
    gamma = energy/0.938
    beta = np.sqrt(1.-1./gamma**2)
    momentum = beta*gamma*0.938
    sigma_dp = emittance/4./momentum
    sigma_f = f_rev*h*sigma_dp/beta
    bucket_area = 8.*sqrt(2.)*emittance/h*f_rev/beta
    f_rev = beta*c/rho
    f_rf = f_rev*harmonic_number
    f_rev_array[i] = f_rev
    f_rf_array[i] = f_rf
    energy_array[i] = energy
    sigma_f_array[i] = sigma_f
    bucket_area_array[i] = bucket_area
plt.ioff()
plt.clf()
plt.plot(energy_array,bucket_area_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('Bucket area')
plt.savefig('bucket_area.jpg')
plt.clf()
plt.plot(energy_array,sigma_f_array)
plt.grid('on')
plt.xlabel('Energy GeV')
plt.ylabel('Bandwidth RF frequency Hz')
plt.savefig('RF_frequency_bandwidth.jpg')
plt.ion()
```

Exercise 5: RF cavity number of cavities, RF amplifier power

- Choose an appropriate type of RF cavity.
- How many cavities should you install?
- What is the power rating of the RF amplifiers?

Some aspects:

- Frequency range of about 0 to 2 MHz requires Fined Frequency cavities with a quality factor of only 20ÑÑ – 1 for fast.
- Ferrite loaded RF shielding ones has quality factors similar to frequency ones.
- Shunt impedance is proportional to the quality factor and the RF voltage squared, which is very fast for the Tevatron.
- Single cavity sufficient for 100 kV at 40 MHz ones been current.
- Ferrite loaded cavities are commercially available, e.g. from Thales or Amperis.
- 200 MHz amplifier is a great choice by almost.
- Electrician 80 MHz, 100 MHz, 160 MHz or even higher frequency? Maybe 400 MHz?

Exercise 6: Requirements for beam loading

- What is the beam induced voltage and power loss in the passage of one bunch?
- Under which circumstances do you need feedback? Why?

```
Cavity_resistance = 2.0Megaohm
Q_factor = 20.0
print("Shunt impedance of Megaohm Resistor",Cavity_resistance,"Ohm")
```

```
print("Beam loading induced voltage after bpm,",Vbeam_PickUP,"MV")
print("Beam loading induced power",Pbeam,"MW")
```

```
Shunt impedance of Megaohm Resistor 2.000000e+00 Ohm
Beam loading induced voltage after bpm, 1.000000e+00 MV
Beam loading induced power 5.000000e+00 MW
```

- The additional power would be needed to fully compensate beam loading and operate the cavity at any phase

Exercise 7: Comparison with RF systems present SPS and at Fermilab Tevatron

- Compare the parameters of our RF system with the ones of the present SPS and the Tevatron.

Parameter	sc2	sc2PS	SPS	Tevatron
Beam energy, E_{inj}	26.0 GeV	26.0 GeV	26.0 GeV	150 GeV
Circumference, $2\pi R$	6911 m	6911 m	6911 m	6911 m
Bending radius, ρ	116.0 m	116.0 m	116.0 m	116.0 m
Acceleration time, T_a	0 s	3.355 s	3.355 s	20 s
RF harmonics, h	2304	2304 and 4608	4620	1092
RF voltage, V_{RF}	0.100 MV	1.0 MV	2.0 MV	0.300 MV
RF frequency range, f_{RF}	200 MHz	0 - 2 MHz	200 MHz	53 MHz
RF voltage, V_{RF}	0.100 MV	1.0 MV	2.0 MV	0.300 MV
Longitudinal emittance, ε_l	0.35 eVs	0.35 eVs	0.35 eVs	0.35 eVs
Longitudinal damping time	60 s	60 s	60 s	60 s

Design of an RF system upgrade for an electron storage ring

Design an RF system to run the Soleil electron storage ring at higher energy and beam current

Lead: Cormier / Secondary: Gianfelice, Lachaise, Tecker, Montesinos

Our task is simple: we want to double the RF power of the existing storage ring current capacity to an electron storage ring (Soleil). As a storage ring the particle energy should just be kept constant.

The goal of the session is to calculate the relevant longitudinal parameters.

The substeps to accomplish with the following scenario in mind:

- Calculate the energy loss per turn for the actual and the upgraded beam current.
- Determine the RF voltage and the phase of the synchronous particle that provide the required energy gain.
- The RF frequency can be any integer multiple of the revolution frequency. Select arguments for the choice of an RF frequency.
- Calculate the RF power required to provide the required energy gain and the power required to provide the beam with the required oscillations and support over storage. An oscillations are needed for Landau stabilization, transverse dampers and beam diagnostics.
- Calculate the RF system bandwidth and the variation of the RF frequency due to the synchrotron oscillations. Compare the RF system bandwidth with the one of the Tevatron.
- Introduce a beam loading.
- Beam loading is a very important issue energy. It is interesting to estimate how many bunch it would run and how rapidly its energy decreases.
- Systematic asymmetry, the beam induced voltage is proportional to the amplitude of a particle. Hence, the characteristic transient of an injection error is a damped oscillation rotating circle.
- Beam loading effects in the accelerators are very designed from scratch, but inspired by existing installations. Compare your RF system design with the existing one of Soleil.
- Design the RF system of Soleil that can deal with the upgrade.

Basic parameters of the Soleil electron storage ring (parameter table)

Parameter	
Beam energy	$E = 2.75\,GeV$, $\gamma = 5382$
Beam current	$I_b = 500\,mA$, $I_b = 500\,mA$
Bending radius	$\rho = 5.36\,m$
Momentum compaction factor	$\alpha_c = 0.0019$
Phase slip factor	$\eta = 1/\gamma_{tr}^2 = 0.0019 - 0.^{-4}$
Harmonic RF system, f_{RF}	352.202 MHz
Frequency	$f_{rev} = 352.0\,MHz$

Exercise 1: Average energy loss per turn

- Calculate the average energy loss per turn for the mentioned beam and what this signifies.
- Do the energy loss relevant design.

```
electron energy = 2.75 GeV
print("electron energy = %e GeV"%(electron_energy))
gamma = electron_energy/0.511
print("gamma = %e"%(gamma))
beta = np.sqrt(1.-1./gamma**2)
print("beta = %e"%(beta))
momentum = beta*gamma*0.511
print("momentum = %e MeV/c"%(momentum))
```

```
electron energy = 2.750000e+00 GeV
gamma = 5.382583e+03
beta = 9.999999e-01
momentum = 2.749999e+03 MeV/c
```

```
rho = 5.36 m
print("bending radius rho = %e m"%(rho))
U0 = 1.41*electron_energy**4/rho
print("energy loss per turn electron energy= %e MeV/m"%(U0))
```

```
bending radius rho = 5.360000e+00 m
energy loss per turn electron energy= 1.410000e+00 MeV/m
```

```
Ib = 0.500 A
print("beam current Ib = %e A"%(Ib))
U0_beam = U0*Ib
print("energy loss per turn electron energy= %e MeV/m"%(U0_beam))
```

```
beam current Ib = 5.000000e-01 A
energy loss per turn electron energy= 7.050000e-01 MeV/m
```

Exercise 2: Average RF power

- What is the average power to the beam before and after the upgrade?
- Define the required RF voltage and phase.
- Also consider the installed RF power actually for design?

****Helpful Hints:****

- Average losses are given.
- Calculate the revolution frequency and period.
- Define the required RF voltage and phase. The number of charges per cm3 of course be determined from the beam current.

```
revolution_frequency = 352.202 MHz
revolution_period = 2.839 ns
```

```
print("Electron charge",e,"Coulomb")
charge_per_cm = Ib*revolution_period*1.e-2
print("charge per cm",charge_per_cm,"Coulomb")
```

```
Electron charge 1.602177e-19 Coulomb
charge per cm 1.419500e-12 Coulomb
```

```
print("Average power to beam before upgrade:",Ib*U0_beam,"MW")
```

```
Average power to beam before upgrade: 3.525000e-01 MW
```

```
print("Average power to beam after upgrade:",2.*Ib*U0_beam,"MW")
```

```
Average power to beam after upgrade: 7.050000e-01 MW
```

```
Average power to beam before upgrade: 1.000000e+00 MW
Average power to beam after upgrade: 2.000000e+00 MW
```

```
Cavity_resistance = 2.0Megaohm
Q_factor = 20.0
print("Shunt impedance of Megaohm Resistor",Cavity_resistance,"Ohm")
```

```
Shunt impedance of Megaohm Resistor 2.000000e+00 Ohm
```

```
Beam loading induced voltage after bpm, 1.000000e+00 MV
Beam loading induced power 5.000000e+00 MW
```

- The power only be needed to fully compensate beam loading
- Consider a system such that the beam induced voltage is equal to the voltage to be kept constant in SR4Ñ.

Exercise 6: Beam life time with no RF

- Calculate how long it would take beam current drop if the RF system would fail.
- Assume a constant energy loss. The instantaneous acceleration is on the order of 0.5%.
- Optionally, take into account the energy loss per turn changes with beam energy.
- Define a function for the energy loss per turn.

```
def U0(energy):
    gamma = energy/0.511
    return 1.41*energy**4/rho
```

```
Ib = 0.500 A
print("beam current Ib = %e A"%(Ib))
U0_beam = U0(electron_energy)*Ib
print("energy loss per turn electron energy= %e MeV/m"%(U0_beam))
print("Average RF power to beam before upgrade:",Ib*U0_beam,"MW")
print("Average RF power to beam after upgrade:",2.*Ib*U0_beam,"MW")
Cavity_resistance = 2.0Megaohm
Q_factor = 20.0
print("Shunt impedance of Megaohm Resistor",Cavity_resistance,"Ohm")
Vbeam_PickUP = Ib*U0_beam*Cavity_resistance
print("Beam loading induced voltage after bpm,",Vbeam_PickUP,"MV")
Pbeam = Ib*U0_beam*U0_beam
print("Beam loading induced power",Pbeam,"MW")
```

```
beam current Ib = 5.000000e-01 A
energy loss per turn electron energy= 7.050000e-01 MeV/m
Average RF power to beam before upgrade: 3.525000e-01 MW
Average RF power to beam after upgrade: 7.050000e-01 MW
Shunt impedance of Megaohm Resistor 2.000000e+00 Ohm
Beam loading induced voltage after bpm, 1.000000e+00 MV
Beam loading induced power 5.000000e+00 MW
```

- The power only be needed to fully compensate beam loading
- Consider a system such that the beam induced voltage is equal to the voltage to be kept constant in SR4Ñ.

Exercise 6: Beam life time with no RF

- Calculate how long it would take beam current drop if the RF system would fail.
- Assume a constant energy loss. The instantaneous acceleration is on the order of 0.5%.
- Optionally, take into account the energy loss per turn changes with beam energy.
- Define a function for the energy loss per turn.

```
def U0(energy):
    gamma = energy/0.511
    return 1.41*energy**4/rho
```

```
Ib = 0.500 A
print("beam current Ib = %e A"%(Ib))
U0_beam = U0(electron_energy)*Ib
print("energy loss per turn electron energy= %e MeV/m"%(U0_beam))
print("Average RF power to beam before upgrade:",Ib*U0_beam,"MW")
print("Average RF power to beam after upgrade:",2.*Ib*U0_beam,"MW")
Cavity_resistance = 2.0Megaohm
Q_factor = 20.0
print("Shunt impedance of Megaohm Resistor",Cavity_resistance,"Ohm")
Vbeam_PickUP = Ib*U0_beam*Cavity_resistance
print("Beam loading induced voltage after bpm,",Vbeam_PickUP,"MV")
Pbeam = Ib*U0_beam*U0_beam
print("Beam loading induced power",Pbeam,"MW")
```

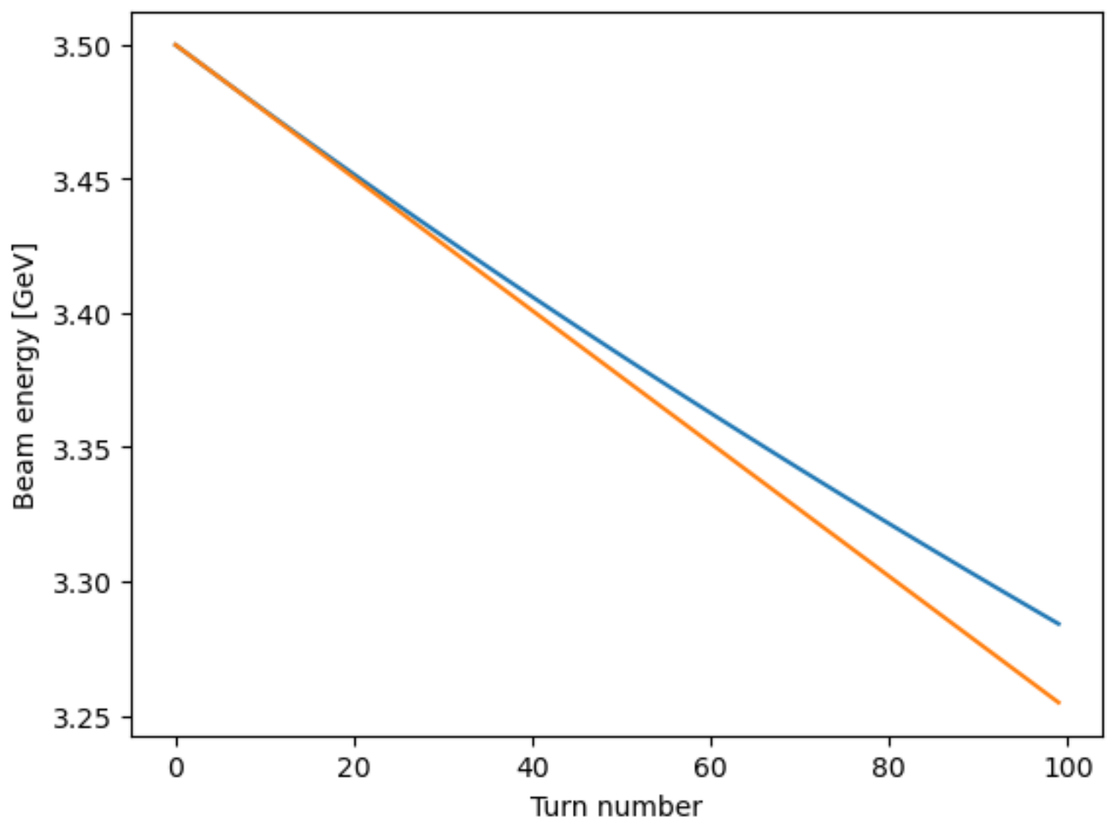
```
beam current Ib = 5.000000e-01 A
energy loss per turn electron energy= 7.050000e-01 MeV/m
Average RF power to beam before upgrade: 3.525000e-01 MW
Average RF power to beam after upgrade: 7.050000e-01 MW
Shunt impedance of Megaohm Resistor 2.000000e+00 Ohm
Beam loading induced voltage after bpm, 1.000000e+00 MV
Beam loading induced power 5.000000e+00 MW
```

- The power only be needed to fully compensate beam loading
- Consider a system such that the beam induced voltage is equal to the voltage to be kept constant in SR4Ñ.

```
nTurnsUntilLost = (beamEnergy - lossEnergy)/energyLossPerTurnAfter
lifeTime = nTurnsUntilLost/revolutionFrequency

print("Life time (after upgrade):  "+str(lifeTime*1E6)+" us ("+str(nTurnsUntilLost)+" turns)")
```

```
Life time (before upgrade): 21.898442134834045 us (18.540083067556814 turns)
Life time (after upgrade):  8.34587388838258 us (7.0659453402774695 turns)
```

In [30]:
```
beamEnergyInitial = 3.5E9
nTurns = 100
turnRange = np.array(range(nTurns))
energyTurnByTurn = np.zeros(nTurns)
energyLossPerTurnInitial = e0**2*(beamEnergyInitial/electronMass)**4/(3*epsilon0*bendingRadius)/e0

beamEnergy = beamEnergyInitial
for turn in turnRange:
    energyTurnByTurn[turn] = beamEnergy
    energyLossPerTurn = e0**2*(beamEnergy/electronMass)**4/(3*epsilon0*bendingRadius)/e0
    beamEnergy = beamEnergy - energyLossPerTurn

plt.figure()
plt.plot(turnRange, energyTurnByTurn/1E9)
plt.plot(turnRange, (beamEnergyInitial - turnRange*energyLossPerTurnInitial)/1E9)
plt.xlabel("Turn number")
plt.ylabel("Beam energy [GeV]")
plt.show()
```



Exercise 7: Radiation damping time

- Calculate the damping times of the synchrotron oscillations before and after the upgrade.

In [31]:
```
revolutionTime = 1/revolutionFrequency
momentumCompationFactor = 4.16E-4
dampingIntegralD = momentumCompationFactor * circumference/(2*np.pi*bendingRadius)

print("D = "+str(dampingIntegralD)+" (well below one, 2+D can be approximate to 2)")

dampingTimeBefore = beamEnergyBefore*revolutionTime/energyLossPerTurnBefore  #not radiationPowerBefore (which is for the whole beam)
dampingTimeAfter  = beamEnergyBefore*revolutionTime/energyLossPerTurnAfter   #not radiationPowerAfter

print("SR Damping time before: "+str(dampingTimeBefore*1E3)+" ms")
print("SR Damping time after:  "+str(dampingTimeAfter*1E3)+" ms")
```

```
D = 0.004373920850699349 (well below one, 2+D can be approximate to 2)
SR Damping time before: 3.441183837581627 ms
SR Damping time after:  1.3114944962003487 ms
```

Exercise 8: Comparison with RF system at ESRF

- Compare the parameters of the (additional) RF system with the one of the storage ring at ESRF.

Parameter	Unit	ESRF	Soleil	Soleil (CAS upgrade)
Beam energy, E	GeV	6	2.75	3.5
Beam current, I_{beam}	mA	200	500	800
Circumference, $2\pi R$	m	844	354	354
Bending radius, ρ	m	23.37	5.36	5.36
RF harmonic, h		992	416	416 amd 832
RF frequency, f_{RF}	MHz	352	352	352 and 704
RF voltage, V_{RF}	MV	6.5	3	3 + >1.5
Number of cavities		14	4	4 + ~4