Ring Object

Usage in codes other than tracking

For data analysis

```python
if BLonDParameters['GeneralParameters'].n_turns == 1:

    self.indexTurn = np.arange(0, self.turnConstant*self.numberFrames, self.turnConstant)
    self.measurementTime = self.initialTime + self.indexTurn * BLonDParameters['GeneralParameters'].t_rev[0]

else:
    startIndexTurn = np.where(BLonDParameters['GeneralParameters'].cycle_time > self.initialTime)[0][0]

    self.indexTurn = np.arange(startIndexTurn, self.turnConstant*self.numberFrames, self.turnConstant)
    self.measurementTime = BLonDParameters['GeneralParameters'].cycle_time[self.indexTurn]
```

- Acquisition is based on triggers on f_{rev} clock. Usually need to recompute f_{rev} to get the distance between two frames.

For data analysis

```
# Detect the bunches and extend the analysis dataset
if multibunchOptions is not None:

    # Getting the size of a bucket in time and indexes
    if 'RFParameters' in self.BLonDParameters:
        if self.BLonDParameters['GeneralParameters'].n_turns == 1:
            sizeBucketTime = 2*np.pi/self.BLonDParameters['RFParameters'].omega_rf[0,0]
        else:
            # TO BE CORRECTED, SHOULD BE indexTurn and not actualFrame !!
            sizeBucketTime = 2*np.pi/self.BLonDParameters['RFParameters'].omega_rf[0,self.actualFrame]
        sizeBucketIndexes = np.round(sizeBucketTime/self.timeInterval)
    else:
        pass
```

- This can be used to get the bunch spacing for easier bunch detection
- **What is needed:** Mostly to get β, γ, t_{rev}. Programs are necessary if these parameters change a lot (e.g. measurements along the ramp)

For the impedance toolbox

```
class MachineParameters(object):
    '''
    The MachineParameters object is based on BLonD objects and gathers all
    information about the machine, beam and impedance
    '''

        self.generalParams = GeneralParameters(1, generalParameters['Circumference [m]'],
                                               1/generalParameters['Transition gamma']**2.,
                                               generalParameters['Momentum [1e9 eV/c]']*1e9,
                                               generalParameters['Particle type'])

        self.rfParams = RFSectionParameters(self.generalParams,
                                            len(rfParameters['Harmonic numbers']),
                                            rfParameters['Harmonic numbers'],
                                            rfParameters['RF voltages'],
                                            rfParameters['RF phases'])
```

- The Ring (and RFStation) object(s) are wrapped into a Machine object
- Transparent for the user, based on known structure for the developer

For the impedance toolbox

```python
# Input folders for impedance and machine parameters
machineParamsInput = './beams/LHC/LHC25ns_flat_top.yml'


# Generating the machine parameters and beam, in one object
machineParams = MachineParameters(machineParamsInput)

# machineParams.generateBeamCurrent(1, 2**19)
# machineParams.generateBeamCurrent(1, resolutionTime=1e-10)
machineParams.generateBeamCurrent(1, maxFreq=5e9)

machineParams.generateBeamSpectrum()
```

LHC25ns_flat_top.yml 499 Bytes

```yaml
Bunch parameters:
  Binomial exponent: 1.5
  Bunch length [ns]: 1.65
  Intensity per bunch [1e10 ppb]: 12
  Line density: binomial
Filling scheme:
  Buckets between batches: 45
  Buckets between bunches: 5
  Buckets between fills: 1
  Number of batches: 4
  Number of bunches: 72
  Number of fills: 1
General parameters:
  Circumference [m]: 6911.5038
  Momentum [1e9 eV/c]: 450
  Particle type: proton
  Transition gamma: 17.95142852
RF parameters:
  Harmonic numbers:
  - 4620
```

- Default machine configurations in YAML file to have a simpler interface for the Impedance team
- The info is used to compute the beam spectrum (numerically, for adjustable bunch spacing) and rf losses

For the impedance toolbox

```python
# Input folders for impedance and machine parameters
machineParamsInput = './beams/LHC/LHC25ns_flat_top.yml'


# Generating the machine parameters and beam, in one object
machineParams = MachineParameters(machineParamsInput)

# machineParams.generateBeamCurrent(1, 2**19)
# machineParams.generateBeamCurrent(1, resolutionTime=1e-10)
machineParams.generateBeamCurrent(1, maxFreq=5e9)

machineParams.generateBeamSpectrum()
```

LHC25ns_flat_top.yml 499 Bytes

```yaml
Bunch parameters:
  Binomial exponent: 1.5
  Bunch length [ns]: 1.65
  Intensity per bunch [1e10 ppb]: 12
  Line density: binomial
Filling scheme:
  Buckets between batches: 45
  Buckets between bunches: 5
  Buckets between fills: 1
  Number of batches: 4
  Number of bunches: 72
  Number of fills: 1
General parameters:
  Circumference [m]: 6911.5038
  Momentum [1e9 eV/c]: 450
  Particle type: proton
  Transition gamma: 17.95142852
RF parameters:
  Harmonic numbers:
  - 4620
```

- **What is needed:** Mostly to get β, γ, t_{rev}. No program needed, only single data points.