PYCVINTERFACE vs PYFUNCTION

Note! Only inn PYCVINTERFACE are you required to compile a shared object file. Make corrections to the examples below.

1. **Python code comparison for both.**

   PYFUNCTION

   i) Python Module:

```python
import plumedCommunications as PLMD

def plumedInit(action: PLMD.PythonFunction):
    return {"Value": PLMD.defaults.COMPONENT_NODEV}

def plumedCalculate(action: PLMD.PythonFunction):
    arg1 = action.argument(0)
    arg2 = action.argument(1)
    return arg1 * arg2  # Multiply the two distances
```

   Requires that the Value is from plumed for example distances and angles etc.

   ii) Dat file:

```
# Load the compiled shared object file
LOAD FILE=/path/to/PythonCVInterface.so

# Define distances as collective variables (CVs)
d1: DISTANCE ATOMS=1,2
d2: DISTANCE ATOMS=1,3

# Call the custom Python function through the loaded interface
fPY: PYFUNCTION IMPORT=pycvfunc CALCULATE=plumedCalculate ARG=d1,d2

# Print the output to a file
PRINT ARG=fPY FILE=colvar.out
```

PYCVINTERFACE

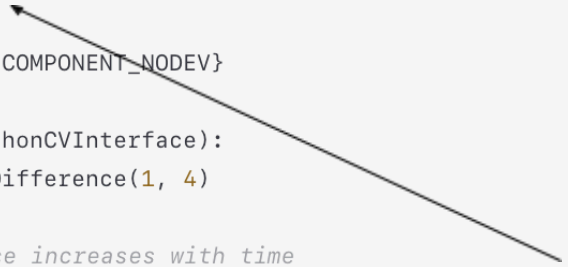
i) Python Module

My designed cv code or a simple one below

```
import plumedCommunications as PLMD

def plumedInit(action: PLMD.PythonCVInterface):
    action.data["force"] = 0.0
    return {"Value": PLMD.defaults.COMPONENT_NODEV}

def pysteeredForce(action: PLMD.PythonCVInterface):
    distance = action.getPositionsDifference(1, 4)
    time_step = action.getStep()
    force = 0.1 * time_step  # Force increases with time
    action.data["force"] = force
    return distance * force
```



ii) Dat file

```
# Load the Python Collective Variable (CV) interface
# This tells PLUMED to load the shared object file containing the
LOAD FILE=/path/to/PythonCVInterface.so

# Define the Collective Variable using PYCVINTERFACE
# PYCVINTERFACE is used to import and calculate custom collective
# ATOMS specifies the atoms involved in the CV calculation
# IMPORT specifies the Python module containing the CV function
# CALCULATE specifies the Python function that will be used for th
cvPY: PYCVINTERFACE ATOMS=1,4 IMPORT=pysteeredMD CALCULATE=pysteer

# Output the calculated CV to a file
# This line tells PLUMED to print the calculated value of cvPY to
PRINT ARG=cvPY FILE=colvar_output.dat
```

1. General comparison

Aspect	PYCVINTERFACE	PYFUNCTION
Definition	A more comprehensive Python interface to implement complex custom collective variables (CVs) and manage interactions between atoms.	A simpler Python function interface that primarily retrieves and processes arguments from the PLUMED input file for calculations.
Mandatory Keys	IMPORT, CALCULATE	IMPORT, CALCULATE, ARG
Data Handling	Uses the `plumedCommunications.PythonCVInterface` object to manage atom positions, time-step updates, and other data required for complex CVs.	Uses the `plumedCommunications.PythonFunction` object to retrieve the arguments (e.g., distances or angles) specified in the `plumed.dat` file.
Components	Can manage multiple components, derivatives, and periodicities, which must be specified in the `plumed.dat` or the Python code.	Focuses on simple, single-component calculations using direct arguments from PLUMED input.
Prepare/Update Support	Supports both the `PREPARE` and `UPDATE` keywords, allowing more flexibility to update or prepare variables at different stages of the simulation.	Doesn't directly support `PREPARE` or `UPDATE`, as it's designed for simpler use cases.
Initialization (INIT)	Allows customization through the `INIT` keyword to set up initial parameters.	Defaults to `plumedInit`, which is simpler and more rigid.
Typical Use Cases	- Complex CVs involving multiple atoms and time-step-dependent variables. - Handling dynamic systems with evolving parameters. - Implementing advanced biasing techniques such as metadynamics or steered MD.	- Simple calculations based on fixed arguments. - Distance-based biases (e.g., RMSD, COM distances). - Multiplying or combining basic collective variables.
Examples in `plumed.dat`	`cvPY: PYCVINTERFACE ATOMS=1,4 IMPORT=pydistancePBCs CALCULATE=pydist`	`fPY: PYFUNCTION IMPORT=pycvfunc CALCULATE=plumedCalculate ARG=d1,d2`