Intelligent function evaluator:

- input parameters can be divided into slow evaluation and fast evaluation, if there is a file

- will consist of n STEPS

- the evaluator can pass the appropriate parameters to each step

- will have access to a database containing parameters for each step

- each step has several methods:

- load file (if one with appropriate parameters found)

- execute step

- save file and update database

- the execute step might need to load/save some intermediate file (MC for resolution)

- for fast evaluation, the save step can be allowed not to load/save a file, to conserve disk, and to increase speed (if I/O limited)

- the evaluator will always try to load a file first, starting from the one generated by the last step, and going up the chain

Possible input file:

#database

File MyMDEvaluation.dat

#Parameters

par\_1 value1

par\_2 value2

par\_3 value3

…

par\_j valuej

#Steps

NumberOfSteps 5

Step1

Name MD\_Init

Parameters par\_2, par\_7, par\_i

Step2

Name MD\_simulation

Parameters par\_2, par\_7, par\_i

Step3

Name ConvertToSQE

Parameters par\_1

Step4

Name ResolutionConvolution

Parameters

Step5

Name AdjustToExperiment

Parameters par\_3,par\_4, par\_5

Possible implementation of steps in python:

class MD\_Init:

“”” MD initialization step “””

def \_\_init\_\_(self, par1,par2,par3,database):

self.par1=par1

self.par2=par2

self.par3=par3

self.database=database

def isFile(self):

return FunctionToReturnFilenameFromParameters(self.database, self.par1, self.par2, self.par3, fileType)

def LoadMDFile(self,filename):

self.data=Load(filename)

def execute(self):

filename=self.isFile()

if (len(filename)>0):

try:

self.LoadMDFile(filename)

return sefl.data

except:

print “Could not load file”

#could not find or load file

#call an external library that saved the MD init file

outputfilename=”MDINIT”+str(self.par1)+”.mdi”

DOMDINIT(self.par1,self.par2,self.par3,outputfilename)

#update the database

UpdateDatabase(self.database, self.par1, self.par2, self.par3, fileType)

self.LoadMDFile(outputFilename)

return self.data