EJIBCE 2018

FCT Fundação para a Ciência e a Tecnologia

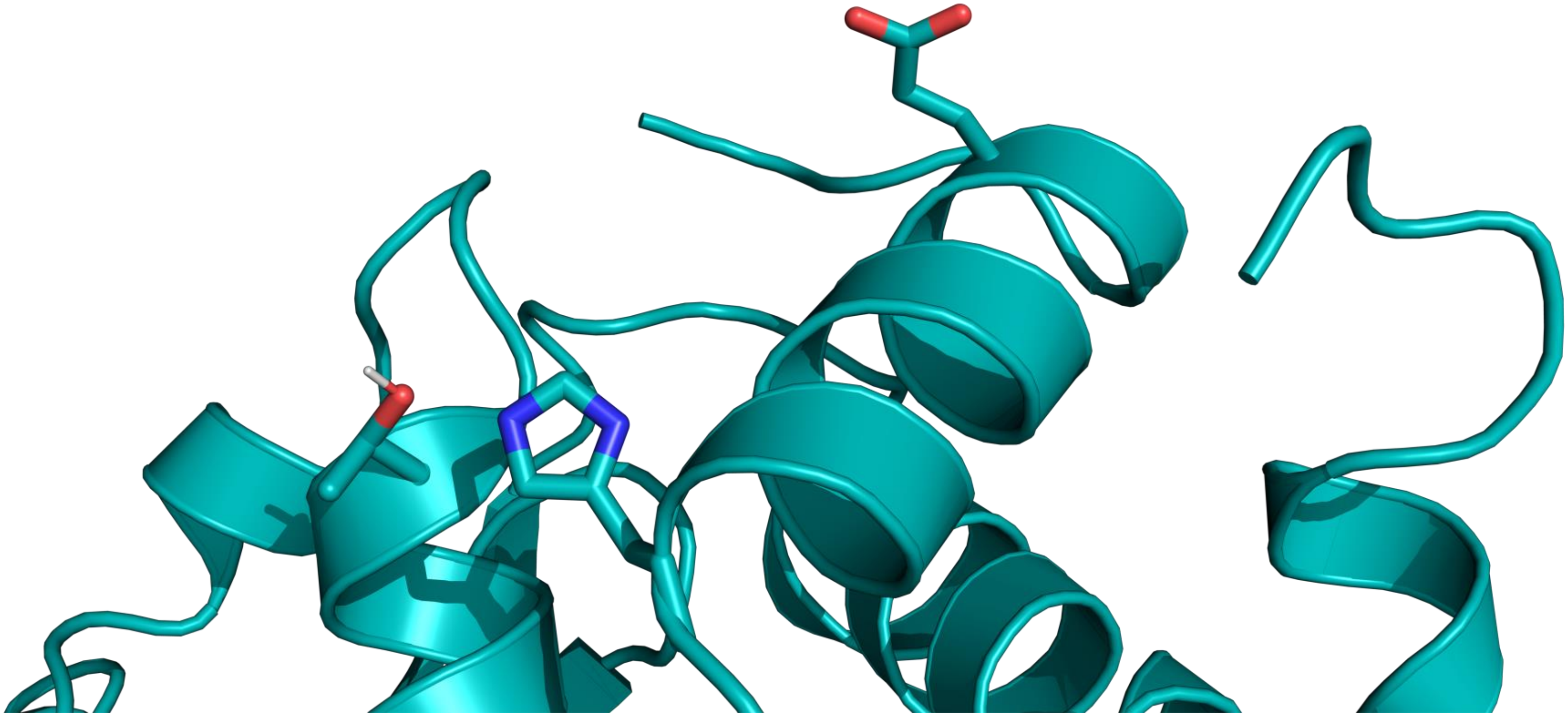
Ciências ULisboa

iit CONCEPT ISTITUTO ITALIANO DI TECNOLOGIA

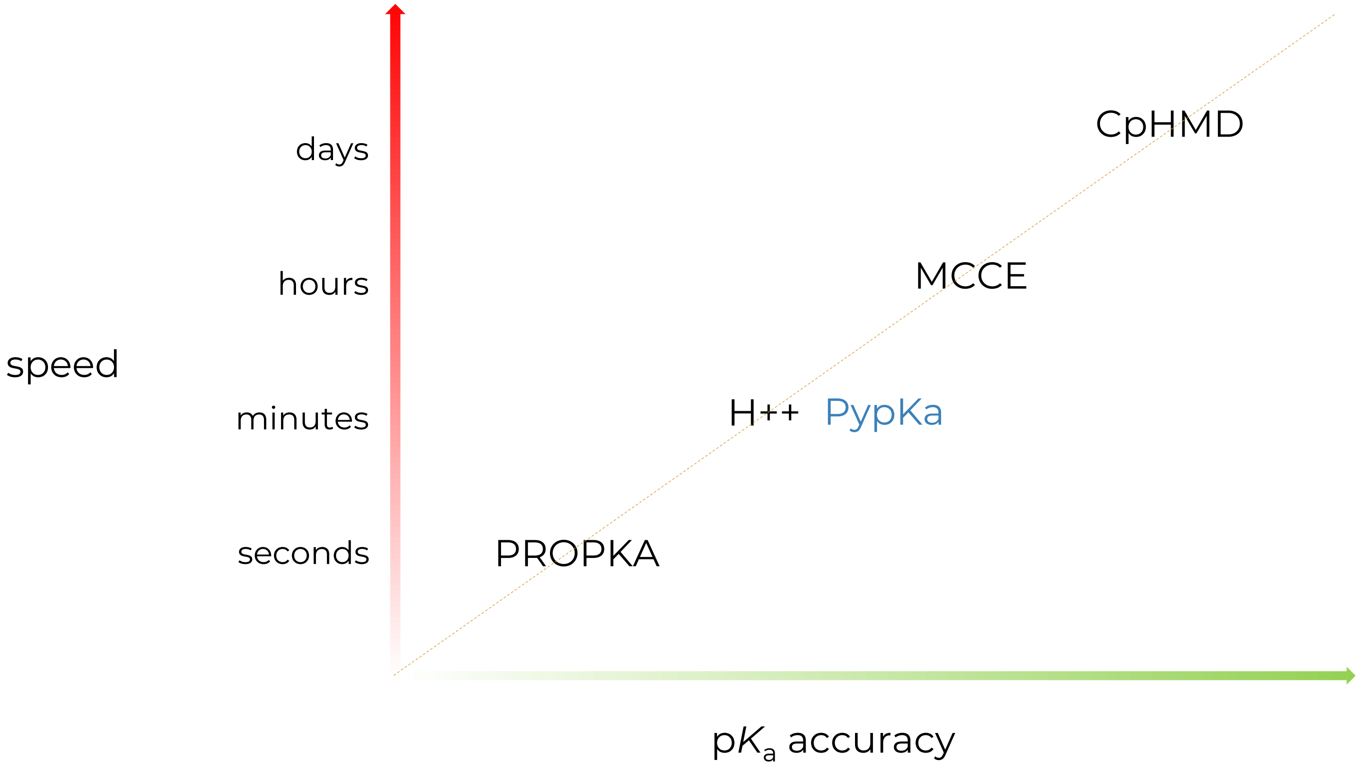
Pedro Reis

PypKa: a python module for flexible Poisson-Boltzmann based pK_a calculations with proton tautomerism

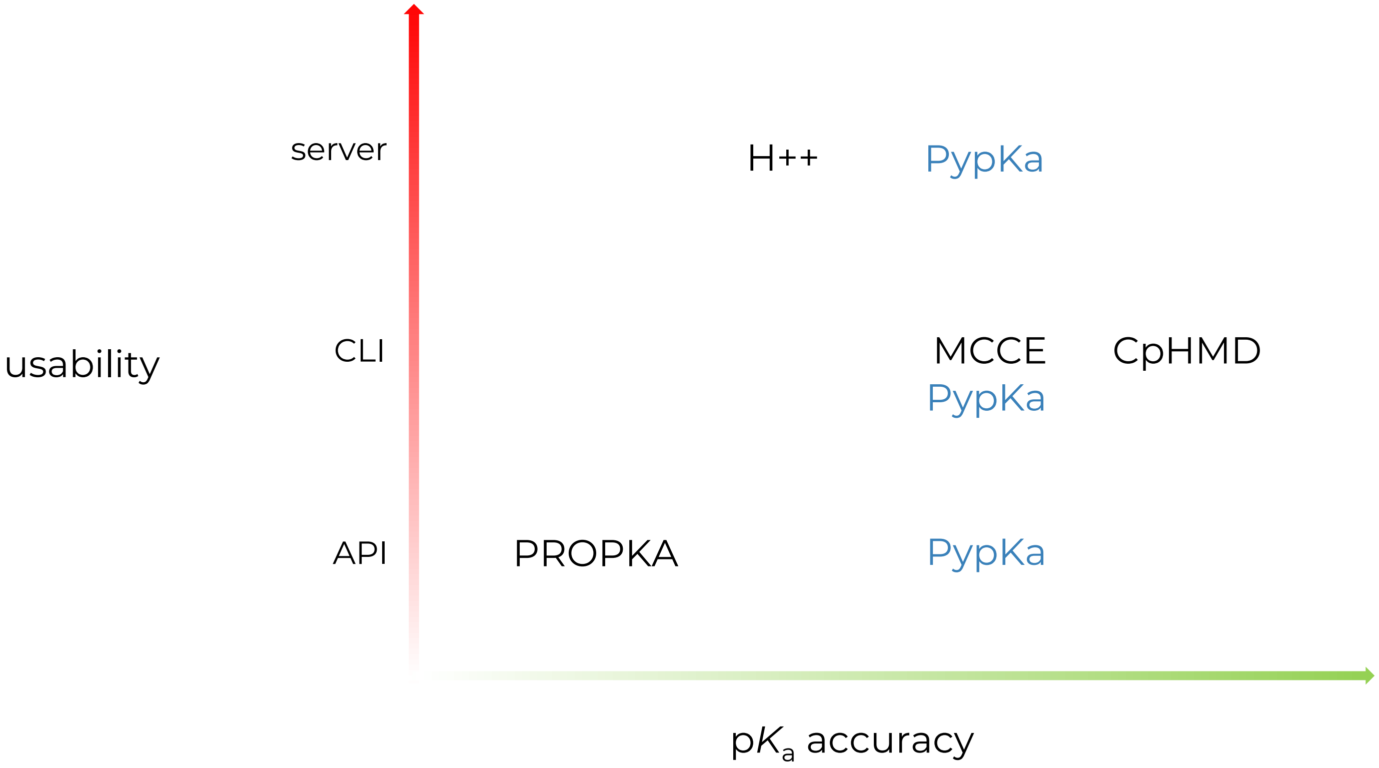
MD Protonation States



State of the art on pK_a calculations



State of the art on pK_a calculations



server

H++

PypKa

usability

CLI

MCCE

CpHMD

PypKa

API

PROPKA

PypKa

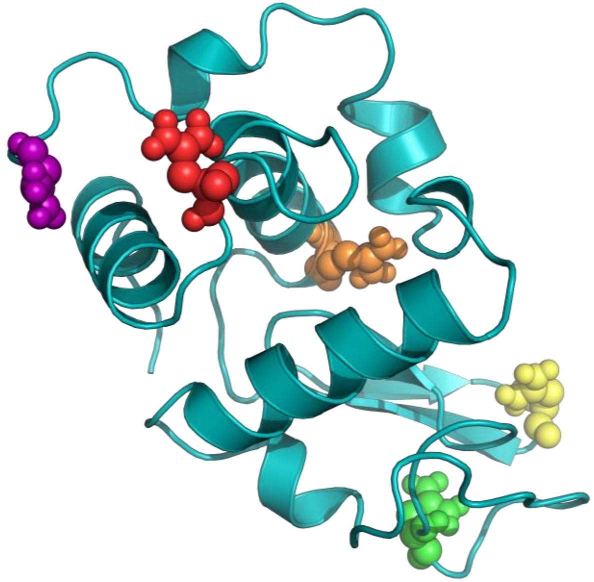
pK_a accuracy

Step 01: Input pdb file

RCSB PDB
PROTEIN DATA BANK

PDB2PQR

Dolinsky TJ et al. 2007



AMBER

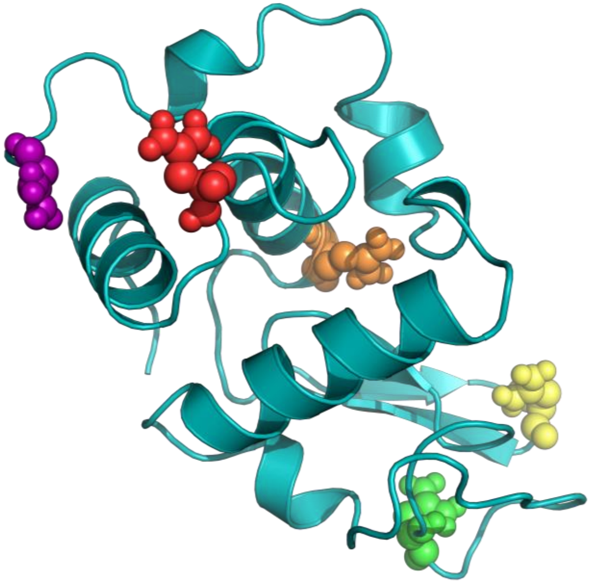
CHARMM

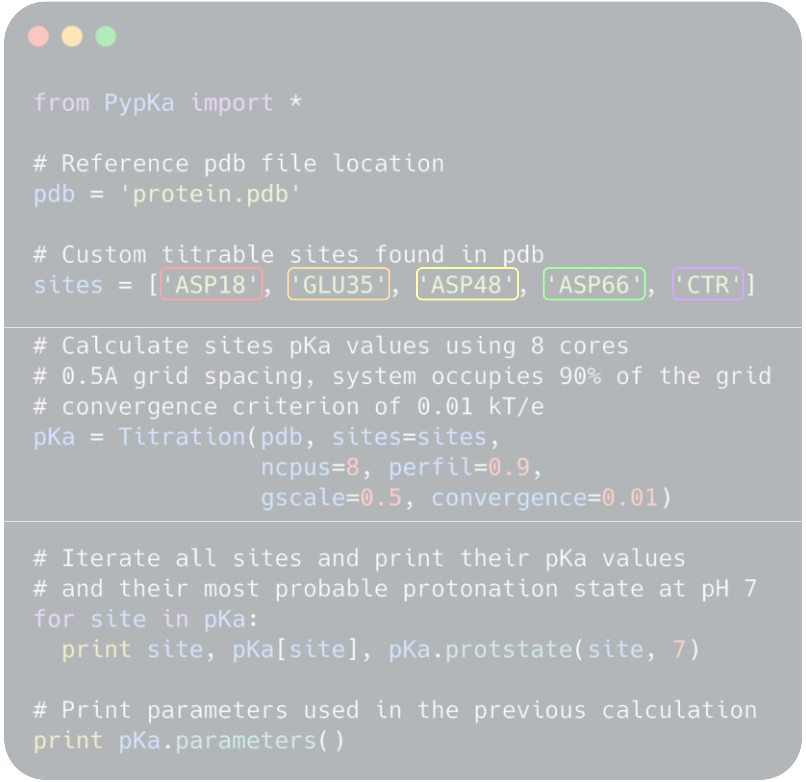
MD
Trajectories

GROMOS

HEWL PDB ID: 4LZT

Step 02: Titrable residues & Parameters





```
from PypKa import *

# Reference pdb file location
pdb = 'protein.pdb'

# Custom titrable sites found in pdb
sites = ['ASP18', 'GLU35', 'ASP48', 'ASP66', 'CTR']

# Calculate sites pKa values using 8 cores
# 0.5A grid spacing, system occupies 90% of the grid
# convergence criterion of 0.01 kT/e
pKa = Titration(pdb, sites=sites,
                ncpus=8, perfil=0.9,
                gscale=0.5, convergence=0.01)

# Iterate all sites and print their pKa values
# and their most probable protonation state at pH 7
for site in pKa:
  print site, pKa[site], pKa.protstate(site, 7)

# Print parameters used in the previous calculation
print pKa.parameters()
```

Single Conformation Unique Site in a Protein

$$\begin{tikzcd}[column sep=huge, row sep=huge]
A_{\boldsymbol{s}}H \arrow[r, "\mathbf{\Delta}G_{\boldsymbol{s}}(AH\rightarrow A)", "\mathrm{p}K_{\mathrm{mod}}"'] \arrow[d] & A_{\boldsymbol{p}}^{-}H_{\boldsymbol{s}}^{+} \arrow[d] \\
A_{\boldsymbol{p}}H \arrow[r, "\mathbf{\Delta}G_{\boldsymbol{p}}(AH\rightarrow A)", "\mathrm{p}K_{\mathrm{a}}"'] & A_{\boldsymbol{p}}^{-}H_{\boldsymbol{s}}^{+}
\end{tikzcd}$$

Single conformation Multiple Sites

$$\begin{tikzcd}
\sum_i \mathrm{M}^{(i)}(c_i, a_i) \arrow[r] \arrow[d] & \sum_i \mathrm{M}^{(i)}(c_i, 0) + z(\mathbf{a})\,\mathrm{H}^+ \arrow[d] \\
\mathrm{P}(c, \mathbf{a}) \arrow[r] & \mathrm{P}(c, \mathbf{0}) + z(\mathbf{a})\,\mathrm{H}^+
\end{tikzcd}$$

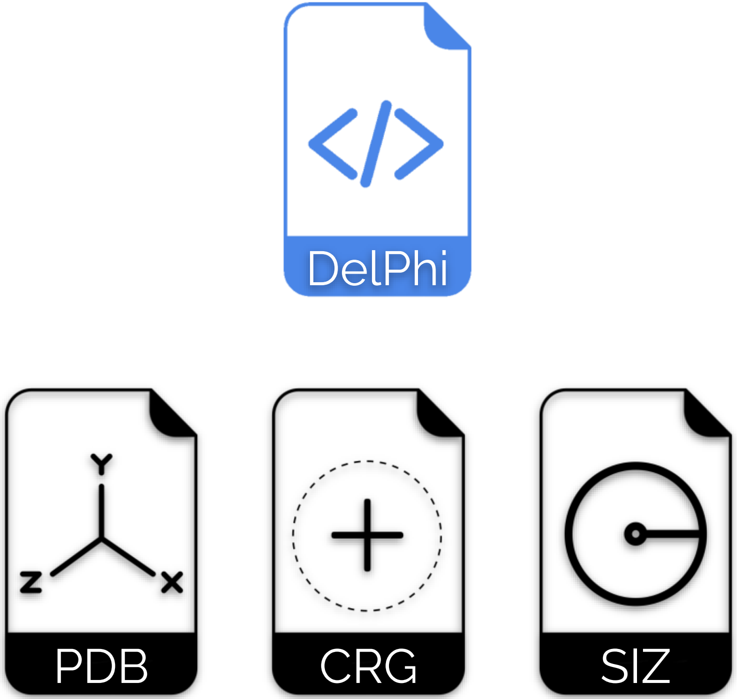
PB/MC

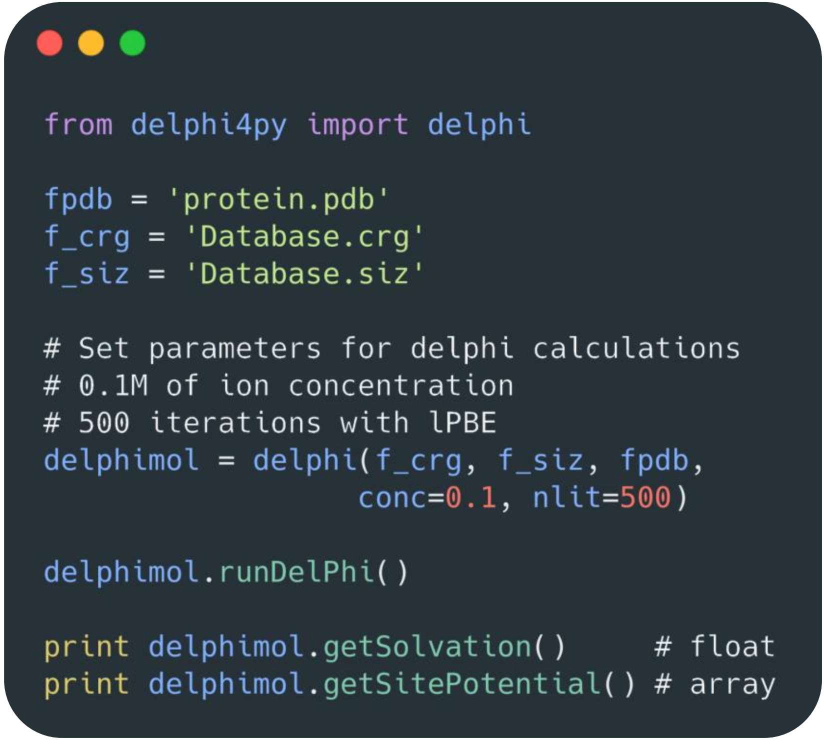
$$\Delta G^P(c,\mathbf{a}\to 0) = 2.3k_BT\sum_i a_i\gamma_i \mathrm{p}K_{intr}(c,a_i) + 2\sum_i\sum_{j<i} a_i a_j \Delta\mathcal{V}^P_{ij}(c)$$

$$\mathrm{p}K_{intr}(c,a_i) = \mathrm{p}K_{mod}(c,a_i) - \frac{\gamma_i}{2.3k_BT}\left[\Delta\Delta G^{M\to P}_{solv}(c,a_i^*) + \Delta\Delta G^{M\to P}_{back}(c,a_i^*)\right]$$

$$\mathrm{p}(\mathbf{a}|c) = \frac{e^{-\beta\Delta G^P(c,\mathbf{a}) - 2.3z(\mathbf{a})\mathrm{pH}}}{\sum e^{-\beta\Delta G^P(c,\mathbf{a}')-2.3z(\mathbf{a}')\mathrm{pH}}}$$

DelPhi4Py: a simple to use python wrapper for DelPhi

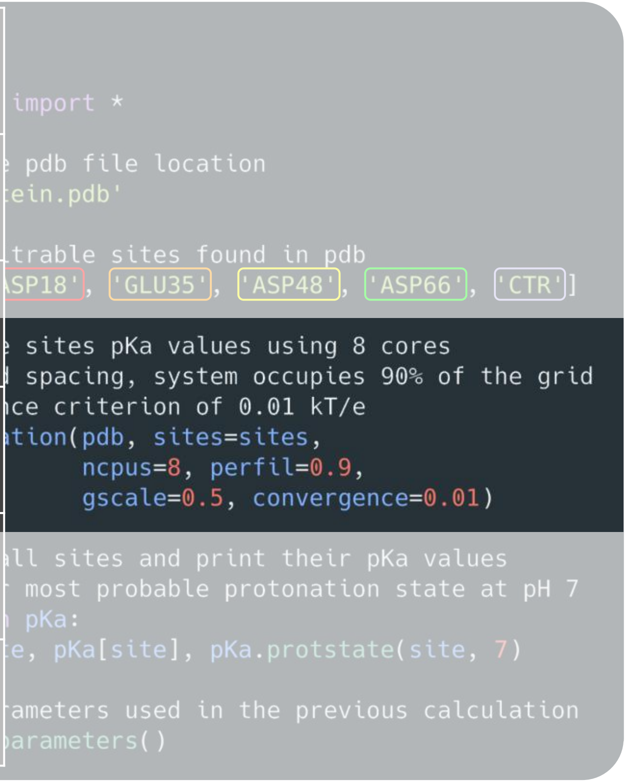




```
from delphi4py import delphi

fpdb = 'protein.pdb'
f_crg = 'Database.crg'
f_siz = 'Database.siz'

# Set parameters for delphi calculations
# 0.1M of ion concentration
# 500 iterations with lPBE
delphimol = delphi(f_crg, f_siz, fpdb,
                   conc=0.1, nlit=500)

delphimol.runDelPhi()

print delphimol.getSolvation()     # float
print delphimol.getSitePotential() # array
```

Step 02: Titrable residues & Parameters

SITE	pK_{mod}	pK_{comp}	pK_{exp}	Δ
ASP 18	3.8	2.9	2.7	0.2
GLU 35	4.2	4.6	6.2	-1.6
ASP 48	3.8	1.4	1.6	-0.2
ASP 66	3.8	1.3	0.9	0.4
CTR 129	3.0	2.9	2.8	0.1



```
 import *

e pdb file location
tein.pdb'

itrable sites found in pdb
ASP18', 'GLU35', 'ASP48', 'ASP66', 'CTR']

e sites pKa values using 8 cores
d spacing, system occupies 90% of the grid
nce criterion of 0.01 kT/e
ation(pdb, sites=sites,
       ncpus=8, perfil=0.9,
       gscale=0.5, convergence=0.01)

all sites and print their pKa values
r most probable protonation state at pH 7
n pKa:
te, pKa[site], pKa.protstate(site, 7)

rameters used in the previous calculation
parameters()
```

Main Features

- easy to use and incorporate in existing scripts
- anisotropic and isotropic systems
- GROMOS 54A7 validated parameters
- CPU & GPU parallel computing
- optional proton tautomerism
- open source project

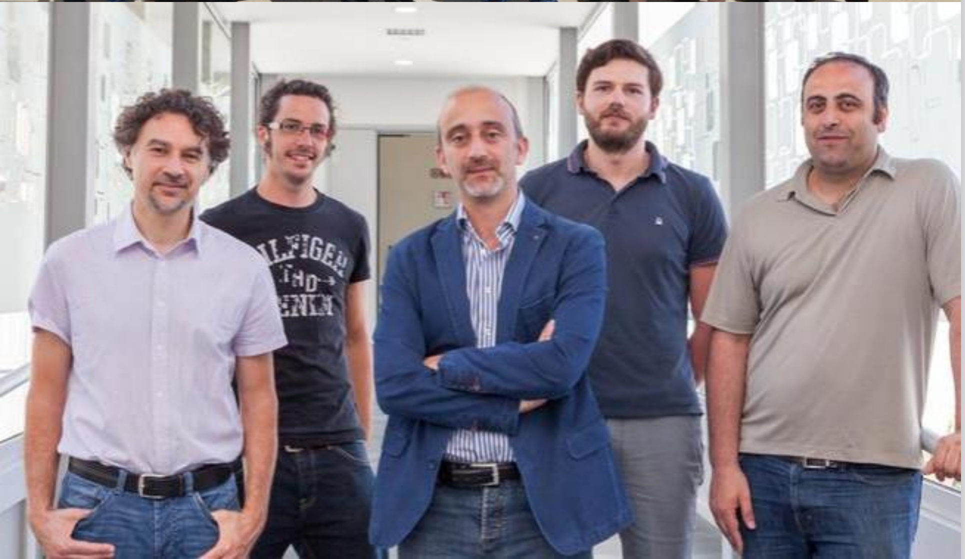
Future Development

1. Parameter calibration
   a. Dieletric constant
   b. Radii
   c. DelPhi Settings
2. Extensive benchmarking
3. Support more Force Fields
4. Online server
5. CpHMD integration

GitHub

https://github.com/mms-fcul/DelPhi4Py

https://github.com/mms-fcul/PypKa



Ciências ULisboa

Inorganic and Theoretical Chemistry Group

FCT Fundação para a Ciência e a Tecnologia

iit ISTITUTO ITALIANO DI TECNOLOGIA

CONCEPT Lab