Improving and extending MapClass

David Martinez

Thursday 26th April, 2012

Overview

MapClass is a code written in *Python* conceived to optimise the non-linear aberrations of the Final Focus System of CLIC. It reads the map coefficients from a file generated by MADX-PTC.

Plan

- Currently
  - Use a version control system (Git[1])
  - Unify all the versions
  - Use the **pytpsa**[2] library in MapClass
- Future
  - Setup of unit tests
  - Profile and optimisation of new code
- Get rid of MADX-PTC and intermediate file

[1] http://git-scm.com/

[2] Library developed by Riccardo De Maria

Git

- Centralised and unified vision
- Everybody keeps their own copy
- Track all the changes
- Assign blame

- Different repositories for each item
  - The library **pytpsa** has its own repository

Setup

- git clone /afs/cern.ch/user/d/dmartine/Repos/MapClass2.git
  - git clone ssh://username@lxplus.cern.ch/afs/... from computers without **AFS**
- git submodule init
- git submodule update

- git pull
- git commit
- git push
- git blame/diff/log

ProGit - http://progit.org/book/ (FREE)

Unification

- Many files with slightly different implementations

```
mapclass25_6var.py  mapclass25_6var_tilt.py
mapclassGaussianDelta25.py  mapclass.GaussianDelta.py
mapclassGaussianDelta.py  mapclass.py  mapclass25.py
```

Parametrisation

- Now it's all in one file
  - Old and new files merged
  - Different functionality parametrised

```
def offset(self, xory, i, gaussianDelta=False)
def sigma(self, xory, i, gaussianDelta=False)
```

Generalisation

- Automatic detection of 5 or 6 dimensions
- Generalisation of methods
- Break up repeated code into multiple methods

now

```
ind=[sum(a) for a in zip(ind1, ind2)]
if all(n % 2 == 0 for n in ind):
  sigmaprod=self.__sigma(ind, i, gaussianDelta)
  if sigmaprod > 0:
   Gammasumln=self.__gamma(ind, gaussianDelta)
   factor=countfactor*self.__factor(ind, gaussianDelta)
   sx+=coeff1*coeff2*factor*exp(Gammasumln)*sigmaprod
```

before

```
jj=coeff1[1] + coeff2[1]
kk=coeff1[2] + coeff2[2]
ll=coeff1[3] + coeff2[3]
mm=coeff1[4] + coeff2[4]
nn=coeff1[5] + coeff2[5]
if ((jj/2==jj/2.) & (kk/2==kk/2.) & (ll/2==ll/2.) & (mm/2==mm/2.) & (nn
  sigmaprod=pow(i[0], jj)*pow(i[1], kk)*pow(i[2], ll)*pow(i[3], mm)*pow
  if (sigmaprod >0):
    Gammasumln=gammln(0.5+jj/2.)+gammln(0.5+kk/2.)+gammln(0.5+ll/2.)+ga
    factor = countfactor*pow(2, (jj+kk+ll+mm)/2.)/pow(pi, 2.)/(nn+1)
    sx += coeff1[0]*coeff2[0]*factor*exp(Gammasumln)*sigmaprod
```

Library pytpsa

Website definition

pytpsa is a python implementation of the truncated Taylor series algebra. It allows to approximate analytical functions or maps to any order.

pol	polynomial object
funset	functions for pol objects: sin, cos, exp, isqrt,...
polmap	polynomial map (it contains pol's)

- MapClass class Map2 extends polmap
- Supports:
  - substitution *map(x=1, y=...)*
  - composition *map1(**map2)*
  - fast composition *map1 * map2*
  - jacobian, extraction of linear maps, and so on...

Thanks for listening

Questions / Suggestions