iminuit 1.3 (release candidate)

Hans Dembinski, Christoph Deil
MPIK Heidelberg

iminuit

- Interactive wrapper around C++ MINUIT2
  - Uses latest version from ROOT-6.12
  - Features of MINUIT2 without ROOT dependency
- Easy to install: `pip install iminuit`
- Official successor of pyminuit & pyminuit2
- Enhanced for interactive use and Cython compatibility
  - Nice print out in Jupyter notebooks and console
  - Simple plots build-in
- Issue tracker on Github (PRs welcome)
  - https://github.com/iminuit/iminuit
- Citable paper coming soon

Reference and tutorials on readthedocs.io

iminuit

MINUIT from Python - Fitting like a boss

iminuit is a Python interface to the *MINUIT* C++ package.

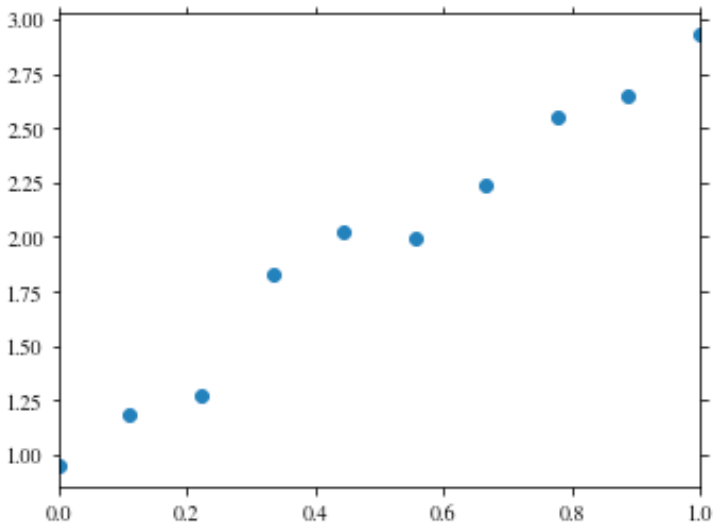
It can be used as a general robust function minimisation method, but is most commonly used for likelihood fits of models to data, and to get model parameter error estimates from likelihood profile analysis.

- Code: https://github.com/iminuit/iminuit
- Documentation: http://iminuit.readthedocs.org/
- Mailing list: https://groups.google.com/forum/#!forum/iminuit
- PyPI: https://pypi.org/project/iminuit/
- License: MINUIT is LGPL and iminuit is MIT
- Citation: https://github.com/iminuit/iminuit/blob/master/CITATION

In a nutshell

```
from iminuit import Minuit

def f(x, y, z):
    return (x - 2) ** 2 + (y - 3) ** 2 + (z - 4) ** 2

m = Minuit(f)

m.migrad()  # run optimiser
print(m.values)  # {'x': 2,'y': 3,'z': 4}

m.hesse()   # run covariance estimator
print(m.errors)  # {'x': 1,'y': 1,'z': 1}
```

Example: line fit




```python
def line(x, a, b):
    return a + x * b
```

```python
def least_squares(a, b):
    yvar = 0.01
    return sum((data_y - line(data_x, a, b)) ** 2 / yvar)
```

Parameter names are detected automatically by iminuit via introspection

```
from iminuit import Minuit
m = Minuit(least_squares, a=0, b=0, error_a=1, error_b=1, errordef=1)
```

In ipython notebook: `m.print_param()`

±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	a	0	1					No
1	b	0	1					No

Example: line fit

```
m.migrad() # do the actual minimization
```

FCN = 10.3870112514	TOTAL NCALL = 32	NCALLS = 32
EDM = 1.38398344664e-21	GOAL EDM = 1e-05	UP = 1.0

Valid	Valid Param	Accurate Covar	PosDef	Made PosDef
True	True	True	True	False
Hesse Fail	HasCov	Above EDM		Reach calllim
False	True	False		False

±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	a	0.990966	0.0587754					No
1	b	1.94494	0.0990867					No

```
m.values['a'] # access fit value
```

Hesse and Minos errors

```
m.hesse()
```

±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	a	0.990966	0.0587754					No
1	b	1.94494	0.0990867					No

±	a	b
a	1.00	-0.84
b	-0.84	1.00

```
m.errors
m.covariance
m.matrix()
m.matrix(correlation=True)
```

```
m.minos()
```

Minos status for a: VALID

Error	-0.0587753813645	0.0587753813645
Valid	True	True
At Limit	False	False
Max FCN	False	False
New Min	False	False

Minos status for b: VALID

Error	-0.0990867388614	0.0990867388614
Valid	True	True
At Limit	False	False
Max FCN	False	False
New Min	False	False

```
m.merrors
```

Parameter limits, fixing parameters

```
m = Minuit(least_squares, a=5, b=5,
           fix_a=True, limit_b=(0, 10),
           error_a=0.1, error_b=0.1,
           errordef=1)
m.migrad()
```

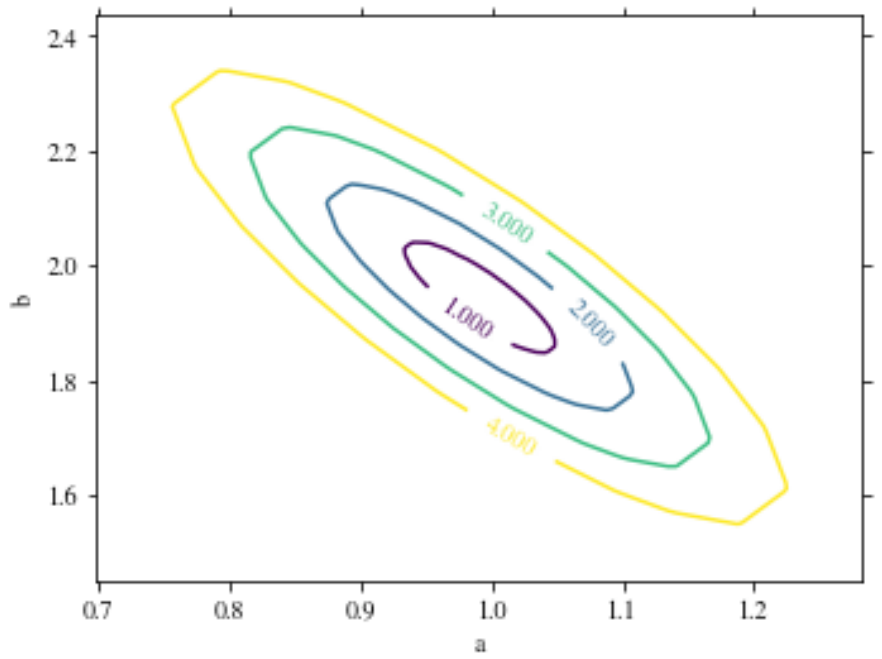
FCN = 9616.37716626	TOTAL NCALL = 13	NCALLS = 13
EDM = 3.49145226281e-13	GOAL EDM = 1e-05	UP = 1.0

Valid	Valid Param	Accurate Covar	PosDef	Made PosDef
True	True	True	True	False
Hesse Fail	HasCov	Above EDM		Reach calllim
False	True	False		False

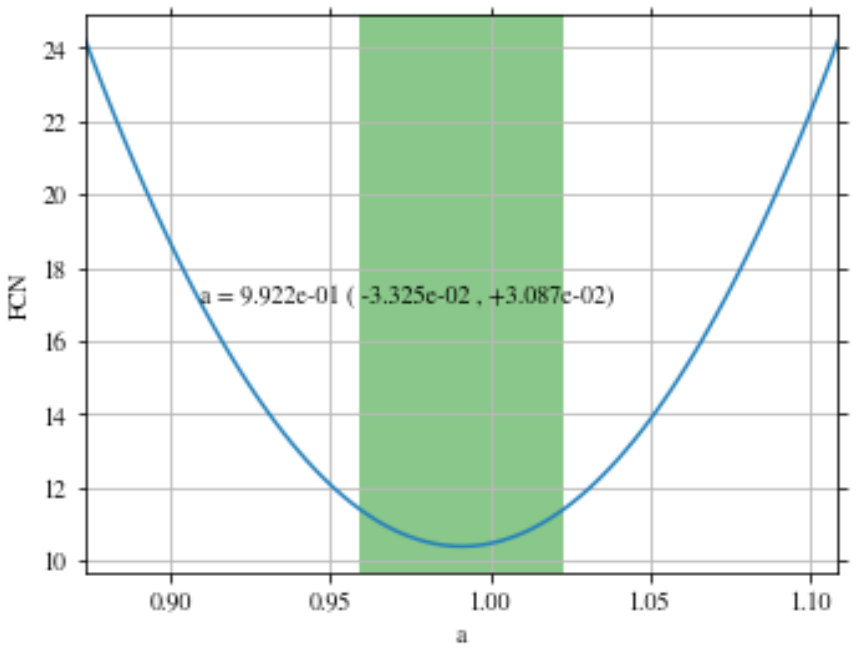
±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	a	5	0.1					Yes
1	b	0	0.00037873			0	10	No

Build-in Plotting

```
m.draw_mncontour('a','b', nsigma=4)
```




```
m.draw_profile('a')
```



Numpy support

```
import numpy as np

def least_squares_np(par):  # par is a numpy array here
    mu = np.polyval(par, data_x)  # for par = (a, b) this is a line
    yvar = 0.01
    return np.sum((data_y - mu) ** 2 / yvar)

m = Minuit.from_array_func(least_squares_np, (0, 0),
                           error=(1, 1), errordef=1)
m.migrad()
m.print_param()
```

±	Name	Value	Hesse Error	Minos Error-	Minos Error+	Limit-	Limit+	Fixed?
0	x0	1.94494	0.0990867					No
1	x1	0.990966	0.0587754					No

```
# accessors for numpy arrays
m.np_values()
m.np_errors()
m.np_covariance()
```

minimize interface

```
from iminuit import minimize  # same interface as scipy.optimize.minimize

result = minimize(least_squares_np, (0, 0))
```

```
      fun: 10.387011251394036
 hess_inv: array([[ 0.00981818, -0.00490909],
       [-0.00490909,  0.00345455]])
  message: 'Optimization terminated successfully.'
   minuit: <iminuit._libiminuit.Minuit object at 0x10ed3a668>
     nfev: 36
     njev: 0
  success: True
        x: array([ 1.9449447 ,  0.99096644])
```

Result is of type scipy.optimize.OptimizeResult