

Jiaze He <jhe7@ncsu.edu>

Your comparison from Text Compare!

no-reply@text-compare.com <no-reply@text-compare.com> To: jhe7@ncsu.edu

Wed, Jul 4, 2018 at 7:37 PM

```
import sys
   import sys
   from os.path import basename, join
                                                                                                  from os.path import basename, join
   from glob import glob
                                                                                                  from glob import glob
   import numpy as np
                                                                                                  import numpy as np
   from seisflows.plugins.solver.specfem2d import smooth legacy
                                                                                                   from seisflows.plugins.solver.specfem2d import smooth legacy
   from seisflows.tools.seismic import getpar, setpar
                                                                                                  from seisflows.tools.seismic import getpar, setpar
                                                                                              10
10
11
   from seisflows.tools import msg
                                                                                              11
                                                                                                  from seisflows.tools import msg
12 from seisflows.tools import unix
                                                                                                  from seisflows.tools import unix
13 from seisflows.tools.seismic import call solver
                                                                                                  from seisflows.tools.seismic import call solver
14 from seisflows.tools.tools import exists
                                                                                                  from seisflows.tools.tools import exists
15 from seisflows.config import ParameterError, custom import
                                                                                                  from seisflows.config import ParameterError, custom import
                                                                                              16
16
| 17 | PAR = sys.modules['seisflows parameters']
                                                                                              17
                                                                                                  PAR = sys.modules['seisflows parameters']
18 PATH = sys.modules['seisflows paths']
                                                                                                  PATH = sys.modules['seisflows paths']
                                                                                               18
19
                                                                                              19
20
   system = sys.modules['seisflows system']
                                                                                              20
                                                                                                  system = sys.modules['seisflows system']
   preprocess = sys.modules['seisflows preprocess']
                                                                                                   preprocess = sys.modules['seisflows preprocess']
22
                                                                                              22
23
                                                                                              23
24
                                                                                              24
    class specfem2d new(custom import('solver', 'base')):
                                                                                                   class specfem2d new(custom import('solver', 'base')):
                                                                                              25
25
       """ Python interface for SPECFEM2D
                                                                                                      """ Python interface for SPECFEM2D
26
                                                                                              26
27
                                                                                              27
        See base class for method descriptions
                                                                                                       See base class for method descriptions
28
                                                                                              28
29
                                                                                              29
      if PAR.MATERIALS == 'LegacyAcoustic':
                                                                                                      if PAR.MATERIALS == 'LegacyAcoustic':
30
                                                                                              30
          parameters = []
                                                                                                          parameters = []
31
                                                                                              31
          parameters += ['vs']
                                                                                                          parameters += ['vs']
32
                                                                                              32
                                                                                              33
34
35
33
34
35
       def check(self):
                                                                                                      def check(self):
                                                                                                          """ Checks parameters and paths
           """ Checks parameters and paths
36
37
                                                                                              36
                                                                                              37
           super(specfem2d_new, self).check()
                                                                                                          super(specfem2d_new, self).check()
38
                                                                                              38
39
                                                                                              39
          # check time stepping parameters
                                                                                                          # check time stepping parameters
40
          if 'NT' not in PAR:
                                                                                              40
                                                                                                          if 'NT' not in PAR:
41
               raise Exception
                                                                                              41
                                                                                                              raise Exception
42
                                                                                              42
43
                                                                                              43
          if 'DT' not in PAR:
                                                                                                          if 'DT' not in PAR:
44
                                                                                               44
               raise Exception
                                                                                                              raise Exception
45
                                                                                              45
46
                                                                                               46
           if 'F0' not in PAR:
                                                                                                          if 'F0' not in PAR:
47
                                                                                               47
               raise Exception
                                                                                                              raise Exception
48
                                                                                               48
                                                                                                          # check data format
49
          # check data format
                                                                                              49
50
          if 'FORMAT' not in PAR:
                                                                                              50
                                                                                                          if 'FORMAT' not in PAR:
51
                                                                                              51
               raise Exception()
                                                                                                              raise Exception()
                                                                                              52
53
52
53
          if PAR.FORMAT != 'su':
                                                                                                          if PAR.FORMAT != 'su':
                                                                                              54
54
               raise Exception()
                                                                                                              raise Exception()
```

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

3 96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

115

116

117

118

119

114

```
55
 56
 57
        def check solver parameter files(self):
 58
            """ Checks solver parameters
 59
 60
           nt = getpar('NSTEP', cast=int)
 61
           dt = getpar('DT', cast=float)
 62
           f0 = getpar('f0', file='DATA/SOURCE', cast=float)
 63
 64
           if nt != PAR.NT:
 65
               if self.taskid == 0: print "WARNING: nt != PAR.NT"
 66
               setpar('nt', PAR.NT)
 67
 68
           if dt != PAR.DT:
 69
               if self.taskid == 0: print "WARNING: dt != PAR.DT"
 70
               setpar('deltat', PAR.DT)
 71
 72
           if f0 != PAR.F0:
 73
               if self.taskid == 0: print "WARNING: f0 != PAR.F0"
 74
               setpar('f0', PAR.F0, filename='DATA/SOURCE')
 75
 76
           if self.mesh properties.nproc != PAR.NPROC:
 77
                if self.taskid == 0:
 78
                    print 'Warning: mesh properties.nproc != PAR.NPROC'
 79
 80
           if 'MULTIPLES' in PAR:
 81
                if PAR.MULTIPLES:
 82
                    setpar('absorbtop', '.false.')
 83
 84
                    setpar('absorbtop', '.true.')
 85
 86
 87
        def generate data(self, **model kwargs):
 88
            """ Generates data
 89
 90
            self.generate mesh(**model kwargs)
 91
 92
           unix.cd(self.cwd)
 93
           setpar('SIMULATION TYPE', '1')
 94
           setpar('SAVE FORWARD', '.false.')
 95
96
            call solver(system.mpiexec(), 'bin/xmeshfem2D')
            call solver(system.mpiexec(), 'bin/xspecfem2D')
 97
 98
 99
           if PAR.FORMAT in ['SU', 'su']:
 100
                src = glob('OUTPUT FILES/*.su')
 101
                dst = 'traces/obs'
 102
               unix.mv(src, dst)
 103
 104
           if PAR.SAVETRACES:
 105
                self.export traces(PATH.OUTPUT+'/'+'traces/obs')
 106
 107
 108
       def initialize adjoint traces(self):
 109
            super(specfem2d new, self).initialize adjoint traces()
 110
 111
           # work around SPECFEM2D's use of different name conventions for
 112
           # regular traces and 'adjoint' traces
 113
           if PAR.FORMAT in ['SU', 'su']:
114
               files = glob('traces/adj/*.su')
               unix.rename('.su', '.su.adj', files)
 115
 116
           # work around SPECFEM2D's requirement that all components exist,
 117
           # even ones not in use
 118
           if PAR.FORMAT in ['SU', 'su']:
 119
```

```
def check solver parameter files(self):
    """ Checks solver parameters
   nt = getpar('NSTEP', cast=int)
    dt = getpar('DT', cast=float)
    f0 = getpar('f0', file='DATA/SOURCE', cast=float)
   if nt != PAR.NT:
       if self.taskid == 0: print "WARNING: nt != PAR.NT"
        setpar('nt', PAR.NT)
   if dt != PAR.DT:
       if self.taskid == 0: print "WARNING: dt != PAR.DT"
        setpar('deltat', PAR.DT)
   if f0 != PAR.F0:
        if self.taskid == 0: print "WARNING: f0 != PAR.F0"
        setpar('f0', PAR.F0, filename='DATA/SOURCE')
   if self.mesh properties.nproc != PAR.NPROC:
        if self.taskid == 0:
            print 'Warning: mesh properties.nproc != PAR.NPROC'
   if 'MULTIPLES' in PAR:
        if PAR.MULTIPLES:
            setpar('absorbtop', '.false.')
            setpar('absorbtop', '.true.')
def generate data(self, **model kwargs):
    """ Generates data
    self.generate mesh(**model kwargs)
    unix.cd(self.cwd)
    setpar('SIMULATION TYPE', '1')
    setpar('SAVE FORWARD', '.false.')
    call solver(system.mpiexec(), 'bin/xmeshfem2D',output='mesh.log')
    call solver(system.mpiexec(), 'bin/xspecfem2D')
   if PAR.FORMAT in ['SU', 'su']:
        src = glob('OUTPUT FILES/*.su')
        dst = 'traces/obs'
       unix.mv(src, dst)
   if PAR.SAVETRACES:
        self.export traces(PATH.OUTPUT+'/'+'traces/obs')
def initialize adjoint traces(self):
    super(specfem2d new, self).initialize adjoint traces()
    # work around SPECFEM2D's use of different name conventions for
    # regular traces and 'adjoint' traces
   if PAR.FORMAT in ['SU', 'su']:
       files = glob('traces/adj*/*.su'
       unix.rename('.su', '.su.adj', files)
    # work around SPECFEM2D's requirement that all components exist,
    # even ones not in use
   if PAR.FORMAT in ['SU', 'su']:
```

```
120
               unix.cd(self.cwd +'/'+ 'traces/adi')
                                                                                                              unix.cd(self.cwd +'/'+ 'traces/adi')
               for channel in ['x', 'y', 'z', 'p']:
                                                                                                              for channel in ['x', 'y', 'z', 'p']:
121
                                                                                              121
122
                   src = 'U%s file single.su.adj' % PAR.CHANNELS[0]
                                                                                               122
                                                                                                                  src = 'U%s file single.su.adj' % PAR.CHANNELS[0]
123
                   dst = 'U%s_file single.su.adj' % channel
                                                                                               123
                                                                                                                  dst = 'U%s file single.su.adj' % channel
124
                   if not exists(dst):
                                                                                               124
                                                                                                                 if not exists(dst):
125
                                                                                              125
                       unix.cp(src, dst)
                                                                                                                      unix.cp(src, dst)
126
                                                                                              126
127
                                                                                              127
128
       def generate mesh(self, model path=None, model name=None, model type='gll'):
                                                                                              128
                                                                                                     def generate mesh(self, model path=None, model name=None, model type='gll'):
                                                                                                          """ Performs meshing and database generation
129
           """ Performs meshing and database generation
                                                                                              129
130
                                                                                              130
                                                                                               131
131
           assert(model name)
                                                                                                          assert(model name)
132
           assert(model type)
                                                                                               132
                                                                                                          assert(model type)
133
                                                                                               133
134
                                                                                               134
           self.initialize solver directories()
                                                                                                          self.initialize solver directories()
135
                                                                                               135
           unix.cd(self.cwd)
                                                                                                          unix.cd(self.cwd)
136
                                                                                               136
137
                                                                                               137
                                                                                                          assert(exists(model_path))
           assert(exists(model_path))
138
                                                                                               138
           self.check mesh properties(model path)
                                                                                                          self.check mesh properties(model path)
139
                                                                                               139
140
                                                                                               140
           src = glob(join(model path, '*'))
                                                                                                          src = glob(join(model path, '*'))
141
                                                                                               141
           dst = join(self.cwd, 'DATA')
                                                                                                          dst = join(self.cwd, 'DATA')
142
           unix.cp(src, dst)
                                                                                               142
                                                                                                          unix.cp(src, dst)
143
                                                                                              143
           if self.taskid == 0:
                                                                                                         if self.taskid == 0:
144
                                                                                              144
145
               self.export model(PATH.OUTPUT +'/'+ model name)
                                                                                               145
                                                                                                              self.export model(PATH.OUTPUT +'/'+ model name)
146
                                                                                               146
147
                                                                                              147
148
      ### low-level solver interface
                                                                                              148
                                                                                                     ### low-level solver interface
149
                                                                                              149
150
       def forward(self, path='traces/syn'):
                                                                                               150
                                                                                                     def forward(self, path='traces/syn'):
151
           """ Calls SPECFEM2D forward solver
                                                                                              151
                                                                                                          """ Calls SPECFEM2D forward solver
152
                                                                                              152
153
           setpar('SIMULATION TYPE', '1')
                                                                                               153
                                                                                                          setpar('SIMULATION TYPE', '1')
154
           setpar('SAVE FORWARD', '.true.')
                                                                                               154
                                                                                                          setpar('SAVE FORWARD', '.true.')
155 #
                                                                                              155 #
            call_solver(system.mpiexec(), 'bin/xmeshfem2D')
                                                                                                            call_solver(system.mpiexec(), 'bin/xmeshfem2D')
156
                                                                                              156
157
                                                                                              157
           call solver(system.mpiexec(), 'bin/xspecfem2D')
                                                                                                          call solver(system.mpiexec(), 'bin/xspecfem2D')
                                                                                              158
158
159
          if PAR.FORMAT in ['SU', 'su']:
                                                                                              159
                                                                                                         if PAR.FORMAT in ['SU', 'su']:
160
              filenames = glob('OUTPUT FILES/*.su')
                                                                                              160
                                                                                                              filenames = glob('OUTPUT FILES/*.su')
161
               unix.mv(filenames, path)
                                                                                               161
                                                                                                              unix.mv(filenames, path)
162
                                                                                               162
163
                                                                                              163
164
                                                                                              164
      def adjoint(self):
                                                                                                     def adjoint(self):
165
           """ Calls SPECFEM2D adjoint solver
                                                                                              165
                                                                                                          """ Calls SPECFEM2D adjoint solver
166
                                                                                              166
167
           setpar('SIMULATION TYPE', '3')
                                                                                              167
                                                                                                          setpar('SIMULATION TYPE', '3')
           setpar('SAVE FORWARD', '.false.')
                                                                                                          setpar('SAVE FORWARD', '.false.')
168
                                                                                               168
169
           unix.rm('SEM')
                                                                                               169
                                                                                                          unix.rm('SEM')
170
           unix.ln('traces/adj', 'SEM')
                                                                                               170
                                                                                                          unix.ln('traces/adj', 'SEM')
171
                                                                                              171
172
                                                                                              172
           # hack to deal with different SPECFEM2D name conventions for
                                                                                                          # hack to deal with different SPECFEM2D name conventions for
173
                                                                                              173
           # regular traces and 'adjoint' traces
                                                                                                          # regular traces and 'adjoint' traces
          if PAR.FORMAT in ['SU', 'su']:
174
                                                                                              174
                                                                                                         if PAR.FORMAT in ['SU', 'su']:
                                                                                                              files = glob('traces/adj/*.su')
175
               files = glob('traces/adj/*.su')
                                                                                              175
               unix.rename('.su', '.su.adj', files)
176
                                                                                              176
                                                                                                              unix.rename('.su', '.su.adj', files)
177
                                                                                               177
178
                                                                                              178
           call_solver(system.mpiexec(), 'bin/xspecfem2D')
                                                                                                          call_solver(system.mpiexec(), 'bin/xspecfem2D')
179
                                                                                              179
                                                                                             180
                                                                                                     def adjoint att(self):
                                                                                                          """ Calls SPECFEM2D adjoint solver
                                                                                              181
                                                                                              182
                                                                                              183
                                                                                                          unix.rm('SEM')
                                                                                              184
                                                                                                         unix.ln('traces/adj att', 'SEM')
```

```
180
      def rename kernels(self):
           """ Works around conflicting kernel filename conventions
181
182
183
           files = []
184
           files += glob('*proc?????? alpha acoustic kernel.bin')
185
           unix.rename('alpha acoustic', 'vp', files)
186
           files = []
187
           files += glob('*proc??????_alpha[hv]_kernel.bin')
          files += glob('*proc??????_reg1_alpha_kernel.bin')
files += glob('*proc??????_reg1_alpha[hv]_kernel.bin')
188
189
190
           unix.rename('alpha', 'vp', files)
191
           files += glob('*proc??????? c acoustic kernel.bin')
192
           unix.rename('c acoustic', 'vp', files)
           unix.rename('c acoustic', 'Qkappa', files)
193
194
           files = []
195
           files += glob('*proc?????? rhop acoustic kernel.bin')
           unix.rename('rhop acoustic', 'rho', files)
196
197
198
           files = []
199
           files += glob('*proc?????? beta kernel.bin')
           files += glob('*proc?????? beta[hv] kernel.bin')
200
201
           files += glob('*proc?????? reg1 beta kernel.bin')
           files += glob('*proc?????? reg1 beta[hv] kernel.bin')
202
203
           unix.rename('beta', 'vs', files)
204
205
      def initialize solver directories(self):
206
           """ Creates directory structure expected by SPECFEM3D, copies
207
             executables, and prepares input files. Executables must be supplied
208
             by user as there is currently no mechanism for automatically
209
             compiling from source.
210
211
           unix.mkdir(self.cwd)
212
          unix.cd(self.cwd)
213
214
           # create directory structure
215
           unix.mkdir('bin')
216
           unix.mkdir('DATA')
217
           unix.mkdir('OUTPUT FILES')
218
219
           unix.mkdir('traces/obs')
220
           unix.mkdir('traces/syn')
```

```
186
           # hack to deal with different SPECFEM2D name conventions for
187
           # regular traces and 'adjoint' traces
188
           if PAR.FORMAT in ['SU', 'su']:
189
               files = glob('traces/adj att/*.su')
190
               unix.rename('.su', '.su.adj', files)
191
192
           call solver(system.mpiexec(), 'bin/xspecfem2D')
193
194
195
       def rename kernels(self):
196
           """ Works around conflicting kernel filename conventions
197
198
           files = []
199
           files += glob('*proc?????? alpha acoustic kernel.bin')
200
           unix.rename('alpha acoustic', 'vp', files)
201
           files = []
202
           files += glob('*proc?????? alpha[hv] kernel.bin')
           files += glob('*proc?????? reg1 alpha kernel.bin')
203
           files += glob('*proc?????? reg1 alpha[hv] kernel.bin')
204
205
           unix.rename('alpha', 'vp', files)
           files += glob('*proc??????? c acoustic kernel.bin')
206
207
           unix.rename('c acoustic', 'vp', files)
208
           unix.rename('c acoustic', 'Qkappa', files)
209
           files = []
210
           files += glob('*proc?????? rhop acoustic kernel.bin')
211
           unix.rename('rhop acoustic', 'rho', files)
212
213
           files = []
214
           files += glob('*proc?????? beta kernel.bin')
           files += glob('*proc?????? beta[hv] kernel.bin')
215
           files += glob('*proc?????? reg1_beta_kernel.bin')
216
           files += glob('*proc?????? reg1 beta[hv] kernel.bin')
217
218
           unix.rename('beta', 'vs', files)
219
220
       def export att kernel(self, path):
221
           unix.cd(self.kernel databases)
222
223
           # work around conflicting name conventions
224
           files = []
225
           files += glob('*proc?????? c acoustic kernel.bin')
226
           unix.rename('c acoustic', 'Qkappa', files)
227
228
           src = glob('*Qkappa kernel.bin')
229
           dst = join(path, 'kernels', self.source name)
230
           unix.mkdir(dst)
231
           unix.mv(src, dst)
232
233
234
       def initialize solver directories(self):
235
           """ Creates directory structure expected by SPECFEM3D, copies
236
             executables, and prepares input files. Executables must be supplied
237
             by user as there is currently no mechanism for automatically
238
            compiling from source.
239
240
           unix.mkdir(self.cwd)
241
           unix.cd(self.cwd)
242
243
           # create directory structure
244
           unix.mkdir('bin')
245
           unix.mkdir('DATA')
246
           unix.mkdir('OUTPUT FILES')
247
248
           unix.mkdir('traces/obs')
249
           unix.mkdir('traces/syn')
```

```
221
           unix.mkdir('traces/adj')
222
223
           unix.mkdir(self.model databases)
224
           unix.mkdir(self.kernel databases)
225
226
           # copy exectuables
227
           src = glob(PATH.SPECFEM BIN +'/'+ '*')
228
           dst = 'bin/'
229
           unix.cp(src, dst)
230
231
          # copy input files
232
           src = glob(PATH.SPECFEM DATA +'/'+ '*')
233
           dst = 'DATA/'
234
           unix.cp(src, dst)
235
236
           src = 'DATA/' + self.source_prefix +'_'+ self.source_name
237
           dst = 'DATA/' + self.source prefix
238
           unix.cp(src, dst)
239
240
           src = 'DATA/STATIONS'+'_'+ self.source_name
241
           dst = 'DATA/STATIONS'
242
           unix.cp(src, dst)
243
244
           self.check solver parameter files()
245
246
247
248
      ### file transfer utilities
249
250
      def import model(self, path):
251
           src = glob(path +'/'+ 'model/*')
252
          dst = join(self.cwd, 'DATA/')
253
254
           unix.cp(src, dst)
255
      def export model(self, path):
256
           unix.mkdir(path)
257
           src = glob(join(self.cwd, 'DATA/*.bin'))
258
           dst = path
259
          unix.cp(src, dst)
260
261
262
      @property
263
      def data filenames(self):
264
           if PAR.CHANNELS:
265
              if PAR.FORMAT in ['SU', 'su']:
266
                  filenames = []
267
                  for channel in PAR.CHANNELS:
268
                      filenames += ['U%s file single.su' % channel]
269
                  return filenames
270
271
           else:
272
               unix.cd(self.cwd)
273
              unix.cd('traces/obs')
274
275
              if PAR.FORMAT in ['SU', 'su']:
276
                   return glob('U? file single.su')
277
278
      @property
279
      def model databases(self):
280
           return join(self.cwd, 'DATA')
281
282
      @property
283
      def kernel databases(self):
```

```
unix.mkdir('traces/adi')
251
            if PAR.ATTENUATION == 'yes' :
 252
              unix.mkdir('traces/adj att')
 253
 254
            unix.mkdir(self.model databases)
 255
            unix.mkdir(self.kernel databases)
 256
 257
            # copy exectuables
 258
            src = glob(PATH.SPECFEM BIN +'/'+ '*')
 259
            dst = 'bin/'
 260
            unix.cp(src, dst)
 261
 262
            # copy input files
 263
            src = glob(PATH.SPECFEM DATA +'/'+ '*')
 264
            dst = 'DATA/'
 265
            unix.cp(src, dst)
 266
 267
            src = 'DATA/' + self.source_prefix +'_'+ self.source_name
            dst = 'DATA/' + self.source prefix
 268
 269
            unix.cp(src, dst)
 270
 271
            src = 'DATA/STATIONS'+' '+ self.source name
 272
            dst = 'DATA/STATIONS'
 273
            unix.cp(src, dst)
 274
 275
            self.check solver parameter files()
 276
 277
 278
 279
        ### file transfer utilities
 280
 281
        def import model(self, path):
 282
            src = glob(path +'/'+ 'model/*')
 283
            dst = join(self.cwd, 'DATA/')
 284
            unix.cp(src, dst)
 285
 286
        def export model(self, path):
 287
            unix.mkdir(path)
 288
            src = glob(join(self.cwd, 'DATA/*.bin'))
 289
            dst = path
 290
            unix.cp(src, dst)
 291
 292
 293
        @property
 294
        def data filenames(self):
 295
            if PAR.CHANNELS:
 296
                if PAR.FORMAT in ['SU', 'su']:
 297
                   filenames = []
 298
                   for channel in PAR.CHANNELS:
 299
                       filenames += ['U%s_file_single.su' % channel]
 300
                   return filenames
 301
 302
            else:
 303
                unix.cd(self.cwd)
 304
                unix.cd('traces/obs')
 305
 306
                if PAR.FORMAT in ['SU', 'su']:
 307
                     return glob('U?_file_single.su')
 308
 309
        @property
 310
        def model databases(self):
 311
            return join(self.cwd, 'DATA')
 312
 313
        @property
 314
        def kernel databases(self):
```

```
284
           return join(self.cwd, 'OUTPUT_FILES')
                                                                                                    315
                                                                                                                return join(self.cwd, 'OUTPUT FILES')
285
                                                                                                    316
286
       @property
                                                                                                    317
                                                                                                           @property
287
288
289
                                                                                                    318
319
320
                                                                                                           def source_prefix(self):
    return 'SOURCE'
       def source_prefix(self):
    return 'SOURCE'
290
       # workaround for older versions of SPECFEM2D,
                                                                                                    321
                                                                                                           # workaround for older versions of SPECFEM2D,
                                                                                                    322
323
291
292
                                                                                                           # which lacked a smoothing utility
       # which lacked a smoothing utility
       #if not exists(PATH.SPECFEM_BIN+'/'+'xsmooth_sem'):
                                                                                                           #if not exists(PATH.SPECFEM_BIN+'/'+'xsmooth_sem'):
293
       # smooth = staticmethod(smooth_legacy)
                                                                                                    324
                                                                                                                 smooth = staticmethod(smooth_legacy)
294
       #smooth = staticmethod(smooth_legacy)
                                                                                                    325
                                                                                                           #smooth = staticmethod(smooth_legacy)
295
                                                                                                    326
```