

```
1  import numpy as np
2  import numpy.random as npr
3  from test_util import *
4  from funkyyak import grad
5  npr.seed(1)
6
7  ✓ def test_dot():
8      def fun(x, y): return to_scalar(np.dot(x, y))
9
10     mat1 = npr.randn(10, 11)
11     mat2 = npr.randn(10, 11)
12     vect1 = npr.randn(10)
13     vect2 = npr.randn(11)
14     vect3 = npr.randn(11)
15
16     check_grads(fun, mat1, vect2)
17     check_grads(fun, mat1, mat2.T)
18     check_grads(fun, vect1, mat1)
19     check_grads(fun, vect2, vect3)
20
21  ✓ def test_max():
22     def fun(x): return to_scalar(np.max(x))
23     d_fun = lambda x : to_scalar(grad(fun)(x))
24     mat = npr.randn(10, 11)
25     check_grads(fun, mat)
26     check_grads(d_fun, mat)
27
28  ✓ def test_sum_1():
29     def fun(x): return to_scalar(np.sum(x))
30     d_fun = lambda x : to_scalar(grad(fun)(x))
31     mat = npr.randn(10, 11)
32     check_grads(fun, mat)
33     check_grads(d_fun, mat)
```

```

34
35 ✓ def test_sum_2():
36     def fun(x): return to_scalar(np.sum(x, axis=0))
37     d_fun = lambda x : to_scalar(grad(fun)(x))
38     mat = npr.randn(10, 11)
39     check_grads(fun, mat)
40     check_grads(d_fun, mat)
41
42 ✓ def test_sum_3():
43     def fun(x): return to_scalar(np.sum(x, axis=0, keepdims=True))
44     d_fun = lambda x : to_scalar(grad(fun)(x))
45     mat = npr.randn(10, 11)
46     check_grads(fun, mat)
47     check_grads(d_fun, mat)
48
49 ✓ def test_mean_1():
50     def fun(x): return to_scalar(np.mean(x))
51     d_fun = lambda x : to_scalar(grad(fun)(x))
52     mat = npr.randn(10, 11)
53     check_grads(fun, mat)
54     check_grads(d_fun, mat)
55
56 ✓ def test_mean_2():
57     def fun(x): return to_scalar(np.mean(x, axis=0))
58     d_fun = lambda x : to_scalar(grad(fun)(x))
59     mat = npr.randn(10, 11)
60     check_grads(fun, mat)
61     check_grads(d_fun, mat)
62
63 ✓ def test_mean_3():
64     def fun(x): return to_scalar(np.mean(x, axis=0, keepdims=True))
65     d_fun = lambda x : to_scalar(grad(fun)(x))
66     mat = npr.randn(10, 11)
67     check_grads(fun, mat)
68     check_grads(d_fun, mat)
69
70 ✓ def test_index_ints():
71     A = npr.randn(5, 6, 4)
72     def fun(x): return to_scalar(x[3, 0, 1])
73     d_fun = lambda x : to_scalar(grad(fun)(x))
74     check_grads(fun, A)
75     check_grads(d_fun, A)
76
77 ✓ def test_index_slice():
78     A = npr.randn(5, 6, 4)
79     def fun(x): return to_scalar(x[:-1, 2:4, :])
80     d_fun = lambda x : to_scalar(grad(fun)(x))
81     check_grads(fun, A)
82     check_grads(d_fun, A)
83
84 ✓ def test_index_lists():
85     A = npr.randn(5, 6, 4)
86     def fun(x): return to_scalar(x[0, 1, 2, 3, 4])

```

```

86     def fun(x): return to_scalar(x[[0, 1, 2], :, :])
87     d_fun = lambda x : to_scalar(grad(fun)(x))
88     check_grads(fun, A)
89     check_grads(d_fun, A)
90
91 ✓ def test_index_mixed():
92     A = npr.randn(5, 6, 4)
93     def fun(x): return to_scalar(x[3, 2:, [1, 3]])
94     d_fun = lambda x : to_scalar(grad(fun)(x))
95     check_grads(fun, A)
96     check_grads(d_fun, A)
97
98 ✓ def test_vector_slice():
99     A = npr.randn(5)
100    def fun(x): return to_scalar(x[2:4])
101    d_fun = lambda x : to_scalar(grad(fun)(x))
102    check_grads(fun, A)
103    check_grads(d_fun, A)
104
105 ✓ def test_index_slice_fanout():
106     A = npr.randn(5, 6, 4)
107     def fun(x):
108         y = x[::-1, 2:4, :]
109         z = x[::-1, 3:5, :]
110         return to_scalar(y + z)
111     d_fun = lambda x : to_scalar(grad(fun)(x))
112     check_grads(fun, A)
113     check_grads(d_fun, A)
114
115 ✓ def test_index_multiple_slices():
116     A = npr.randn(7)
117     def fun(x):
118         y = x[2:6]
119         z = y[1:3]
120         return to_scalar(z)
121     d_fun = lambda x : to_scalar(grad(fun)(x))
122     check_grads(fun, A)
123     check_grads(d_fun, A)
124
125 ✓ def test_reshape_method():
126     A = npr.randn(5, 6, 4)
127     def fun(x): return to_scalar(x.reshape((5 * 4, 6)))
128     d_fun = lambda x : to_scalar(grad(fun)(x))
129     check_grads(fun, A)
130     check_grads(d_fun, A)
131
132 ✓ def test_reshape_call():
133     A = npr.randn(5, 6, 4)
134     def fun(x): return to_scalar(np.reshape(x, (5 * 4, 6)))
135     d_fun = lambda x : to_scalar(grad(fun)(x))
136     check_grads(fun, A)
137     check_grads(d_fun, A)
138

```

```

139 ✓ def test_ravel_method():
140     A = npr.randn(5, 6, 4)
141     def fun(x): return to_scalar(x.ravel())
142     d_fun = lambda x : to_scalar(grad(fun)(x))
143     check_grads(fun, A)
144     check_grads(d_fun, A)
145
146 ✓ def test_ravel_call():
147     A = npr.randn(5, 6, 4)
148     def fun(x): return to_scalar(np.ravel(x))
149     d_fun = lambda x : to_scalar(grad(fun)(x))
150     check_grads(fun, A)
151     check_grads(d_fun, A)
152
153 ✓ def test_concatenate_axis_0():
154     A = npr.randn(5, 6, 4)
155     B = npr.randn(5, 6, 4)
156     def fun(x): return to_scalar(np.concatenate((B, x, B)))
157     d_fun = lambda x : to_scalar(grad(fun)(x))
158     check_grads(fun, A)
159     check_grads(d_fun, A)
160
161 ✓ def test_concatenate_axis_1():
162     A = npr.randn(5, 6, 4)
163     B = npr.randn(5, 6, 4)
164     def fun(x): return to_scalar(np.concatenate((B, x, B), axis=1))
165     d_fun = lambda x : to_scalar(grad(fun)(x))
166     check_grads(fun, A)
167     check_grads(d_fun, A)
168
169 ✓ def test_concatenate_axis_1_unnamed():
170     """Tests whether you can specify the axis without saying "axis=1"."""
171     A = npr.randn(5, 6, 4)
172     B = npr.randn(5, 6, 4)
173     def fun(x): return to_scalar(np.concatenate((B, x, B), 1))
174     d_fun = lambda x : to_scalar(grad(fun)(x))
175     check_grads(fun, A)
176     check_grads(d_fun, A)
177
178 # TODO:
179 # squeeze, transpose, getitem

```