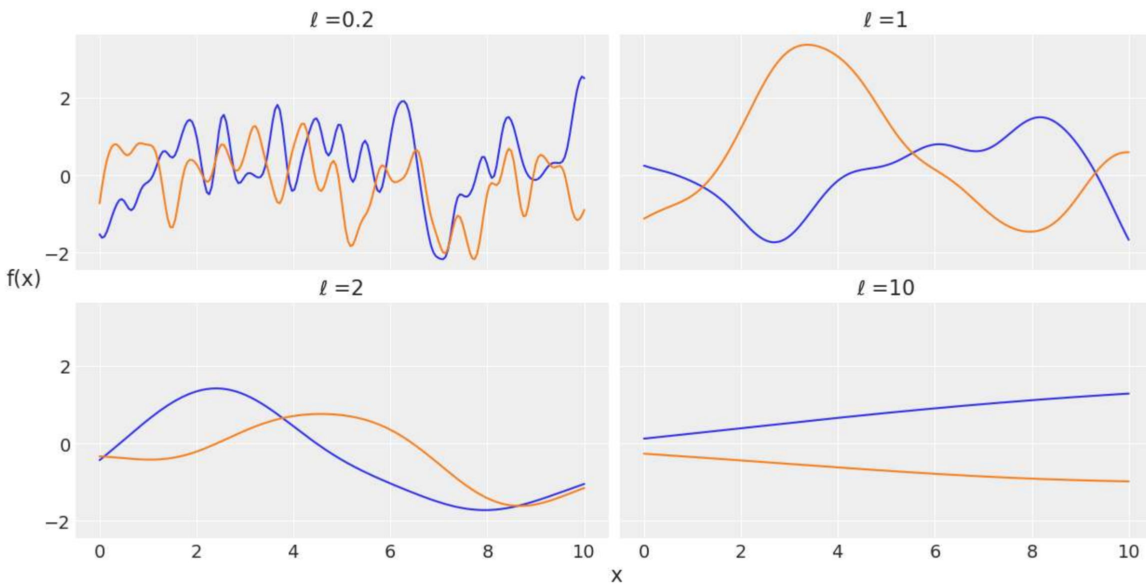
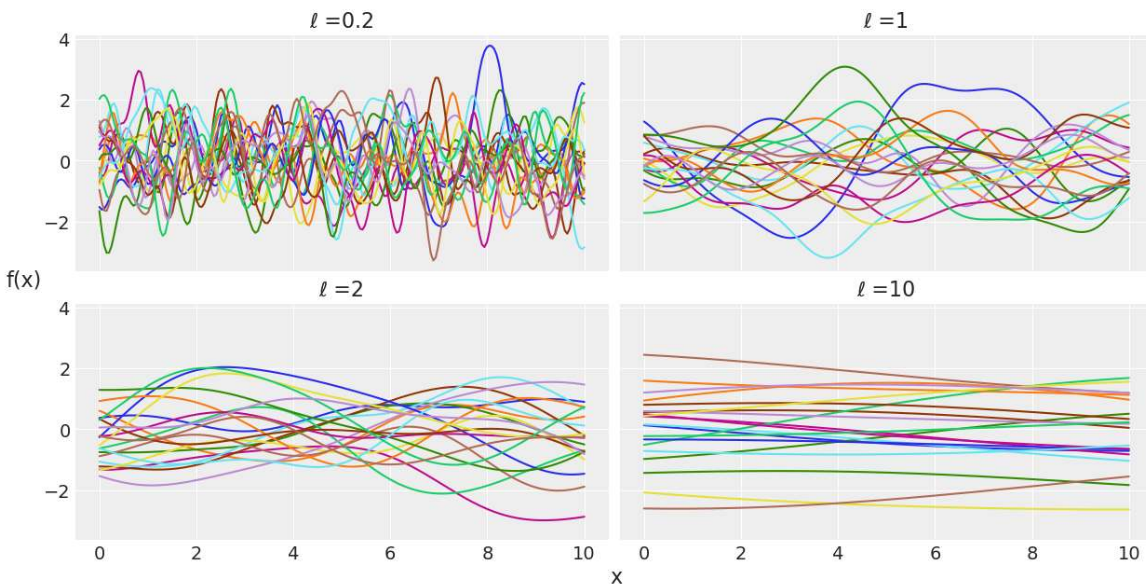
In [4]:

```
np.random.seed(24)
test_points = np.linspace(0, 10, 200)
fig, ax = plt.subplots(2, 2, figsize=(12, 6), sharex=True,
                       sharey=True, constrained_layout=True)
ax = np.ravel(ax)

for idx, ℓ in enumerate((0.2, 1, 2, 10)):
    cov = exp_quad_kernel(test_points, test_points, ℓ)
    ax[idx].plot(test_points, stats.multivariate_normal.rvs(cov=cov, size=2).T)
    ax[idx].set_title(f'ℓ ={ℓ}')
fig.text(0.51, -0.03, 'x', fontsize=16)
fig.text(-0.03, 0.5, 'f(x)', fontsize=16)
plt.savefig('B11197_07_03.png')
```



In [11]:
```
np.random.seed(24)
test_points = np.linspace(0, 10, 200)
fig, ax = plt.subplots(2, 2, figsize=(12, 6), sharex=True,
                       sharey=True, constrained_layout=True)
ax = np.ravel(ax)

for idx, ℓ in enumerate((0.2, 1, 2, 10)):
    cov = exp_quad_kernel(test_points, test_points, ℓ)
    ax[idx].plot(test_points, stats.multivariate_normal.rvs(cov=cov, size=20).T)
    ax[idx].set_title(f'ℓ ={ℓ}')
fig.text(0.51, -0.03, 'x', fontsize=16)
fig.text(-0.03, 0.5, 'f(x)', fontsize=16)
```

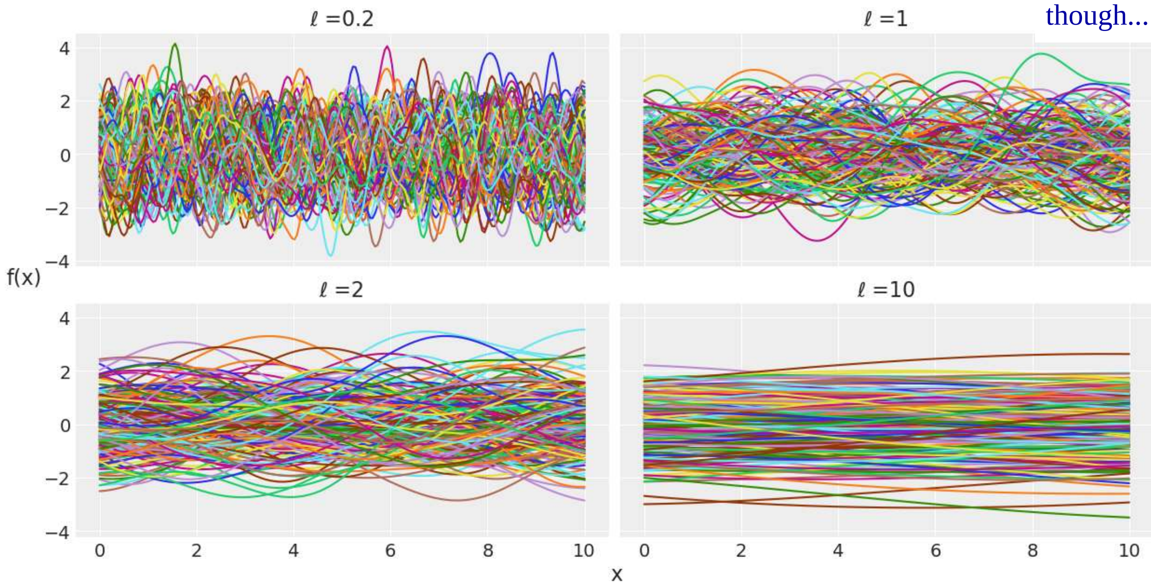
Out[11]:
```
Text(-0.03, 0.5, 'f(x)')
```



In [5]:
```python
np.random.seed(24)
test_points = np.linspace(0, 10, 200)
fig, ax = plt.subplots(2, 2, figsize=(12, 6), sharex=True,
                       sharey=True, constrained_layout=True)
ax = np.ravel(ax)

for idx, ℓ in enumerate((0.2, 1, 2, 10)):
    cov = exp_quad_kernel(test_points, test_points, ℓ)
    ax[idx].plot(test_points, stats.multivariate_normal.rvs(cov=cov, size=200).T)
    ax[idx].set_title(f'ℓ ={ℓ}')
fig.text(0.51, -0.03, 'x', fontsize=16)
fig.text(-0.03, 0.5, 'f(x)', fontsize=16)
```

I set size=200 so that I could get 200 multi-normal distributions. The plots are not very clear nor informative, though...

Out[5]: Text(-0.03, 0.5, 'f(x)')



In [7]:
```python
y_df = pd.DataFrame(stats.multivariate_normal.rvs(cov=cov, size=200))
```

In [8]:
```python
y_df.head()
```

Out[8]:

	0	1	2	3	4	5	6	7	8	
0	0.422100	0.409618	0.397062	0.384436	0.371738	0.358970	0.346135	0.333231	0.320261	0.307
1	-0.778149	-0.779094	-0.780079	-0.781104	-0.782169	-0.783274	-0.784421	-0.785607	-0.786835	-0.788
2	-1.156881	-1.157028	-1.157169	-1.157305	-1.157436	-1.157563	-1.157684	-1.157802	-1.157915	-1.158
3	0.001888	0.005178	-0.012204	-0.019190	-0.026132	-0.033033	-0.039892	-0.046708	-0.053480	-0.060
4	0.390947	0.388387	0.385801	0.383188	0.380547	0.377880	0.375186	0.372465	0.369717	0.366

"y" values at each "x"

5 rows × 200 columns

In [14]:
```python
len(y_df.columns)
```

Out[14]: 200

In [17]:
```
y_df_std = []
for i in range(len(y_df.columns)):
    y_df_std.append(round(y_df[i].std(),2))
```

Since each column represents "y" values at each "x". I computed the standard deviation of each vector in y_df.

In [28]:
```
np.linspace(0,199,200)
```

Out[28]:
```
array([  0.,   1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.,
        11.,  12.,  13.,  14.,  15.,  16.,  17.,  18.,  19.,  20.,  21.,
        22.,  23.,  24.,  25.,  26.,  27.,  28.,  29.,  30.,  31.,  32.,
        33.,  34.,  35.,  36.,  37.,  38.,  39.,  40.,  41.,  42.,  43.,
        44.,  45.,  46.,  47.,  48.,  49.,  50.,  51.,  52.,  53.,  54.,
        55.,  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.,  96.,  97.,  98.,
        99., 100., 101., 102., 103., 104., 105., 106., 107., 108., 109.,
       110., 111., 112., 113., 114., 115., 116., 117., 118., 119., 120.,
       121., 122., 123., 124., 125., 126., 127., 128., 129., 130., 131.,
       132., 133., 134., 135., 136., 137., 138., 139., 140., 141., 142.,
       143., 144., 145., 146., 147., 148., 149., 150., 151., 152., 153.,
       154., 155., 156., 157., 158., 159., 160., 161., 162., 163., 164.,
       165., 166., 167., 168., 169., 170., 171., 172., 173., 174., 175.,
       176., 177., 178., 179., 180., 181., 182., 183., 184., 185., 186.,
       187., 188., 189., 190., 191., 192., 193., 194., 195., 196., 197.,
       198., 199.])
```

In [29]:
```
x=np.linspace(0,199,200)
plt.scatter(x, y_df_std)
```

The plot of the std of each column. Stds are around "1". I think this result agree with the the diagonal line of the covariance matrix. But I am not sure if this is what was asked to compare in Question3, and I don't know how to observe the std directly from the "x-y" plots.

Out[29]: <matplotlib.collections.Path(