To compare with the p-values calculated by Python, we use the dataset 'GaussianData' generated by Python. Since the Matlab version cannot support too flexible arguments, we only test the basic setting: apply Gaussian kernel to all variables X,Y and Z and set the kernel width using empirical rules.

```
data = GaussianData;
```

- The value of the variable 'Approximate' can be manually set in the file 'UInd_KCItest.m' and the file 'algorithms/CInd_test_new_withGP'. When Approximate=True, the resulting p-values will be deterministic. Otherwise, p-values is not determinitic due to the simulation of null distribution. In this case, the program will be run 500 times and the mean of the p-values will be calculated.

- The value of the variable 'IF_GP' can be manually set in the file 'algorithms/CInd_test_new_withGP'.

1. Approximate = True, IF_GP = True

```
% Approximate = True, IF_GP = True
[p_val stat]=indtest_new(data(:,1),data(:,2),[],[]);
p_val % X and X_prime should be independent
```

```
p_val = 0.2662
```

```
[p_val stat]=indtest_new(data(:,1),data(:,4),[],[]);
p_val % X and Z should be dependent
```

```
p_val = 5.0105e-06
```

```
[p_val stat]=indtest_new(data(:,1),data(:,4),data(:,3),[]);
p_val % X and Z should be conditionally independent given Y
```

```
p_val = 0.3585
```

2. Approximate = True, IF_GP = False

```
% Approximate = True, IF_GP = False
[p_val stat]=indtest_new(data(:,1),data(:,2),[],[]);
p_val % X and X_prime should be independent
```

```
p_val = 0.2662
```

```
[p_val stat]=indtest_new(data(:,1),data(:,4),[],[]);
p_val % X and Z should be dependent
```

```
p_val = 5.0105e-06
```

```
[p_val stat]=indtest_new(data(:,1),data(:,4),data(:,3),[]);
p_val % X and Z should be conditionally independent given Y
```

```
p_val = 0.3977
```

When setting 'Approximate=False', please replace Line 60 in 'indtest_new.m':

'[stat, Cri, p_val_appr, Cri_appr, pval] = CInd_test_new_withGP(X, Y, Z, 0.8, pars.width);'

by

'[stat, Cri, pval, Cri_appr, p_val_appr] = CInd_test_new_withGP(X, Y, Z, 0.8, pars.width);'.

3. Approximate = False, IF_GP = True

```
% Approximate = False, IF_GP = True
rng(1,'philox');
j=500;
pXX = zeros(1,j);
pXZ = zeros(1,j);
pXZY = zeros(1,j);
for i=1:j
    [p_val stat]=indtest_new(data(:,1),data(:,2),[],[]);
    pXX(1,i) = p_val; %p_val % X and X_prime should be independent
    [p_val stat]=indtest_new(data(:,1),data(:,4),[],[]);
    pXZ(1,i) = p_val; %p_val % X and Z should be dependent
    [p_val stat]=indtest_new(data(:,1),data(:,4),data(:,3),[]);
    pXZY(1,i) = p_val; %p_val % X and Z should be conditionally independent given Y
end
fprintf('mean:');
```

```
mean:
```

```
mean(pXX)
```

```
ans = 0.2475
```

```
mean(pXZ)
```

```
ans = 1.0200e-04
```

```
mean(pXZY)
```

```
ans = 0.3377
```

```
fprintf('var:');
```

```
var:
```

```
var(pXX)
```

```
ans = 2.0764e-04
```

```
var(pXZ)
```

```
ans = 9.5788e-08
```

```
var(pXZY)
```

```
ans = 4.6791e-05
```

4. Approximate = False, IF_GP = False

```
% Approximate = False, IF_GP = False
rng(1,'philox');
j=500;
pXX = zeros(1,j);
pXZ = zeros(1,j);
pXZY = zeros(1,j);
for i=1:j
    [p_val stat]=indtest_new(data(:,1),data(:,2),[],[]);
    pXX(1,i) = p_val; %p_val % X and X_prime should be independent
    [p_val stat]=indtest_new(data(:,1),data(:,4),[],[]);
    pXZ(1,i) = p_val; %p_val % X and Z should be dependent
    [p_val stat]=indtest_new(data(:,1),data(:,4),data(:,3),[]);
    pXZY(1,i) = p_val; %p_val % X and Z should be conditionally independent given Y
end
fprintf('mean:');
```

```
mean:
```

```
mean(pXX)
```

```
ans = 0.2467
```

```
mean(pXZ)
```

```
ans = 6.8000e-05
```

```
mean(pXZY)
```

```
ans = 0.3740
```

```
fprintf('var:');
```

```
var:
```

```
var(pXX)
```

```
ans = 1.7177e-04
```

```
var(pXZ)
```

```
ans = 6.7511e-08
```

```
var(pXZY)
```

```
ans = 4.6800e-05
```