How to use the adaptively penalized Kolmogorov-Smirnov (apKS) method

Fatih Olmez
folmez@gmail.com

13th December 2016

Abstract

This manual aims to demonstrate how to use the apKS method written in MatLab. If you found this method useful, please remember to make a reference to our paper (CITE EPL).

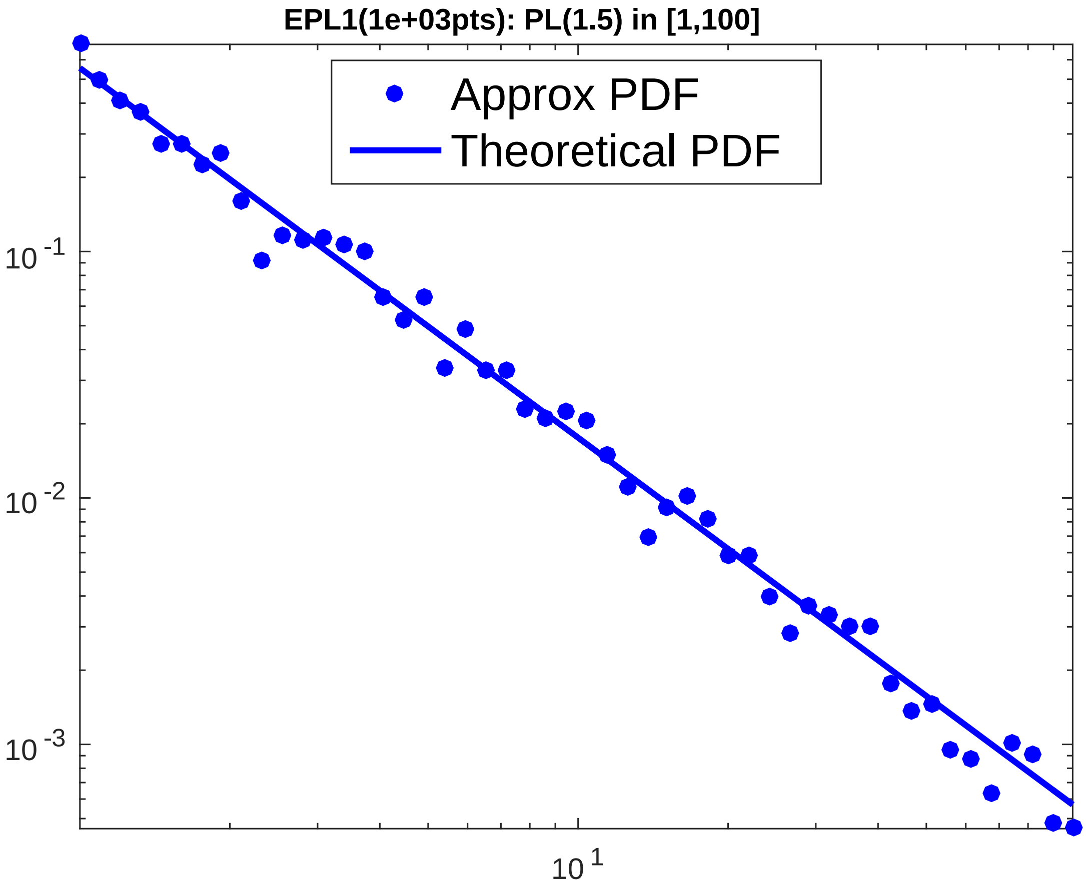
Contents

1 Generating synthetic data sets

There are three synthetic data set types discussed in (CITE EPL). These data sets can be generated as follows:

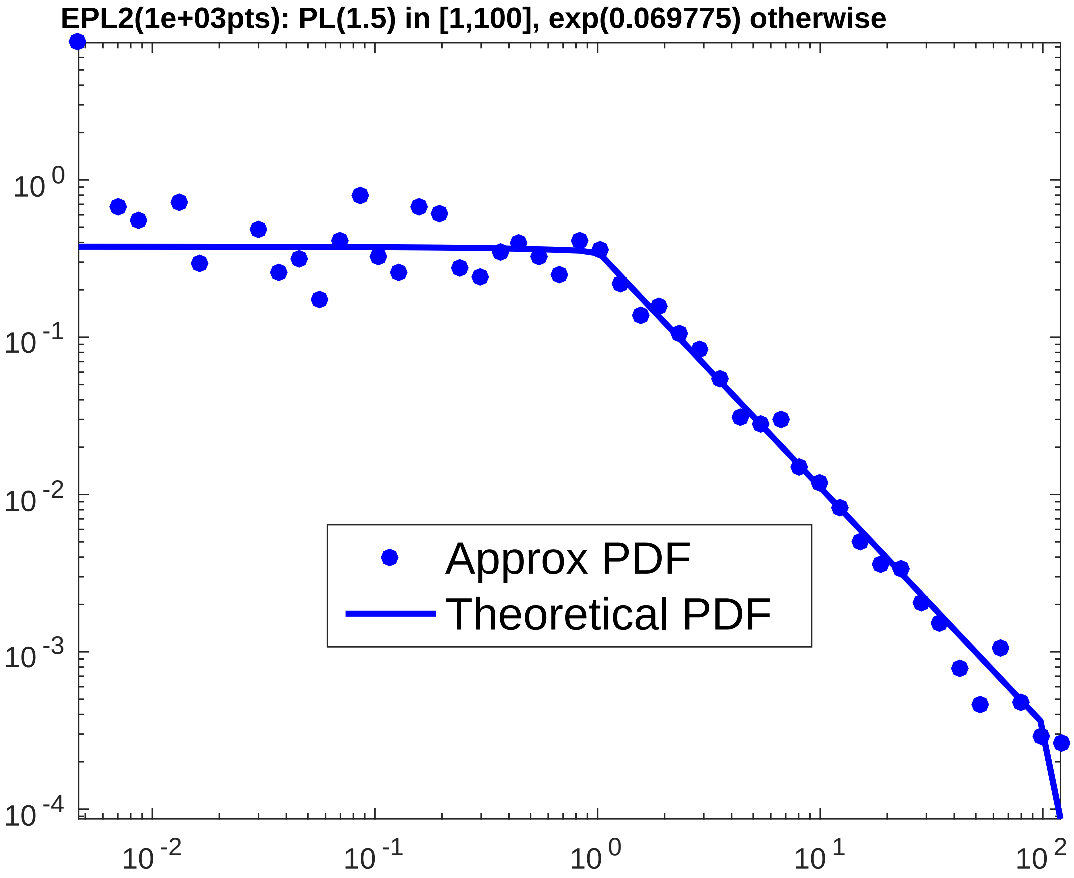
1.1 EPL1: Exact Power-Law 1

```
>> X = gsdf('EPL1', 1.5, [1 100], 1e3, 1);
Synthetic data generation completed in 0.00 minutes
EPL1(1e+03pts): PL(1.5) in [1,100]
>>
```



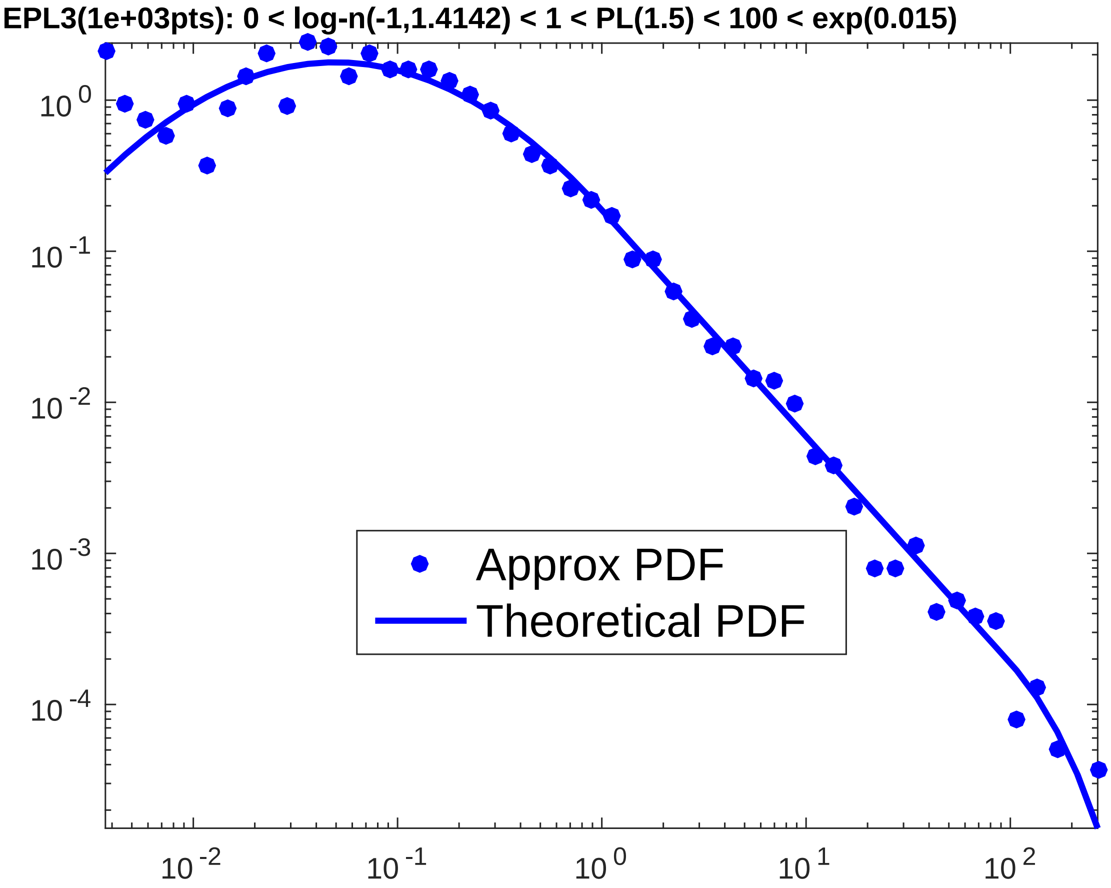
1.2 EPL2: Exact Power-Law 2

```
>> X = gsdf('EPL2', 1.5, [1 100],    1e3, 1);
Synthetic data generation completed in 0.00 minutes
EPL2(1e+03pts): PL(1.5) in [1,100], exp(0.069775) otherwise
>>
```



1.3 EPL3: Exact Power-Law 3

```
>> X = gsdf('EPL3', 1.5, [-1 1 100], 1e3, 1);
Synthetic data generation completed in 0.00 minutes
EPL3(1e+03pts): 0 < log-n(-1,1.4142) < 1 < PL(1.5) < 100 < exp(0.015)
>>
```



2 The apKS method

The basic apKS method

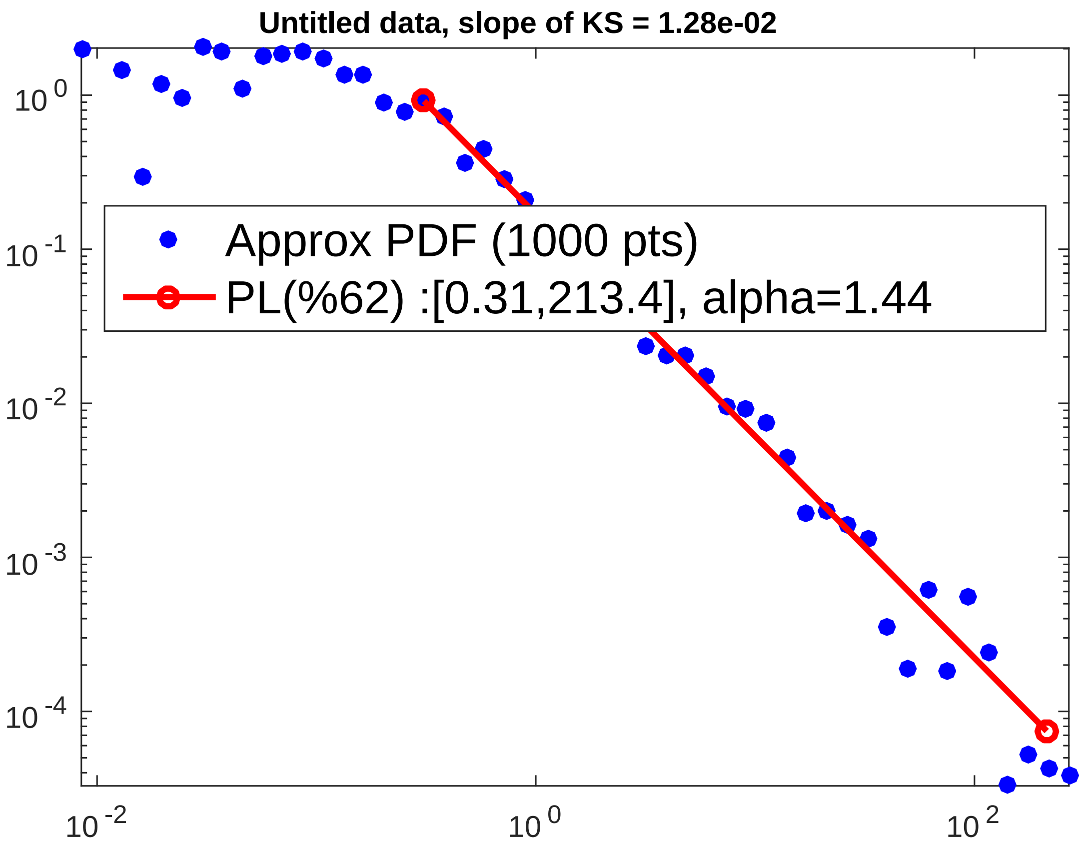
1. searches for a bounded power-law that is at least a decade long
2. estimates a $p-$value from 25 semiparametric samples
3. considers only 10 points in every decade as a candidate power-law bound

These three parameters can be set by the user. Let's first start with an example of the basic apKS method.

2.1 Basic apKS method

```
>> X = gsdf('EPL3', 1.5, [-1 1 100], 1e3, 1);
Synthetic data generation completed in 0.00 minutes
EPL3(1e+03pts): 0 < log-n(-1,1.4142) < 1 < PL(1.5) < 100 < exp(0.015)
>>
>> apKS(X)
Slope #0 = 0.00000
Power-law fit:  alpha=1.45 on (0.38,182.10)
[0]     p_KS    Time           Bounds(  0.38, 182.10)  K-S(0.0205)    alpha(1.45)
[1]     1.0000  [0.00m]                 0.58,  21.56   0.0230         1.42
[2]     0.5000  [0.01m]                 0.25,   4.78   0.0195         1.34
...
[25]    0.4000  [0.09m]                 0.25,  16.76   0.0225         1.38
Accepted with p-value 0.40 (computed from 25 reps)
... finished in 0.10 minutes

Slope #1 = 0.00001
Power-law fit:  alpha=1.45 on (0.38,182.10)
Accepted b/c same as the one for slope #0=0.00000
... finished in 0.00 minutes

Slope #2 = 1.00000
Power-law fit:  alpha=1.09 on (0.01,269.33)
[0]     p_KS    Time           Bounds(  0.01, 269.33)  K-S(0.1975)    alpha(1.09)
[1]     0.0000  [0.00m]                 0.01, 202.33   0.0152         1.07
[2]     0.0000  [0.01m]                 0.04,  20.49   0.0158         1.09
...
[25]    0.0000  [0.09m]                 0.01,  92.38   0.0124         1.11
Rejected with p-value 0.00 (computed from 25 reps)
... finished in 0.10 minutes

Slope #3 = 0.00316
Power-law fit:  alpha=1.45 on (0.38,213.45)
[0]     p_KS    Time           Bounds(  0.38, 213.45)  K-S(0.0209)    alpha(1.45)
[1]     1.0000  [0.00m]                 0.31, 269.33   0.0227         1.43
[2]     0.5000  [0.01m]                 0.58, 208.61   0.0189         1.44
...
[25]    0.3600  [0.09m]                 0.39, 202.41   0.0155         1.46
Accepted with p-value 0.36 (computed from 25 reps)
... finished in 0.09 minutes

Slope #4 = 0.05623
Power-law fit:  alpha=1.09 on (0.01,269.33)
Rejected b/c same as the one for slope #2=1.00000
... finished in 0.01 minutes

Slope #5 = 0.01334
Power-law fit:  alpha=1.45 on (0.31,269.33)
[0]     p_KS    Time           Bounds(  0.31, 269.33)  K-S(0.0259)    alpha(1.45)
[1]     0.0000  [0.00m]                 0.30,  19.67   0.0171         1.43
[2]     0.0000  [0.01m]                 0.30,  13.31   0.0204         1.55
...
[25]    0.0800  [0.09m]                 0.30, 248.99   0.0164         1.47
Rejected with p-value 0.08 (computed from 25 reps)
... finished in 0.09 minutes

Slope #6 = 0.00649
Power-law fit:  alpha=1.45 on (0.38,213.45)
Accepted b/c same as the one for slope #3=0.00316
... finished in 0.00 minutes

Slope #7 = 0.00931
Power-law fit:  alpha=1.44 on (0.31,213.45)
[0]     p_KS    Time           Bounds(  0.31, 213.45)  K-S(0.0229)    alpha(1.44)
[1]     0.0000  [0.00m]                 0.38, 150.33   0.0165         1.47
[2]     0.0000  [0.01m]                 0.27, 130.51   0.0191         1.43
...
[25]    0.2800  [0.09m]                 0.31,  60.11   0.0250         1.45
Accepted with p-value 0.28 (computed from 25 reps)
... finished in 0.09 minutes

Slope #8 = 0.01114
Power-law fit:  alpha=1.44 on (0.31,213.45)
Accepted b/c same as the one for slope #7=0.00931
... finished in 0.00 minutes

Slope #9 = 0.01219
Power-law fit:  alpha=1.44 on (0.31,213.45)
Accepted b/c same as the one for slope #7=0.00931
... finished in 0.00 minutes

Slope #10 = 0.01275
Power-law fit:  alpha=1.44 on (0.31,213.45)
Accepted b/c same as the one for slope #7=0.00931
... finished in 0.00 minutes
```

```
Slope #11 = 0.01304
Power-law fit:  alpha=1.45 on (0.31,269.33)
Rejected b/c same as the one for slope #5=0.01334
... finished in 0.00 minutes

Slope #12 = 0.01289
Power-law fit:  alpha=1.45 on (0.31,269.33)
Rejected b/c same as the one for slope #5=0.01334
... finished in 0.00 minutes

Slope #13 = 0.01282
Power-law fit:  alpha=1.44 on (0.31,213.45)
Accepted b/c same as the one for slope #7=0.00931
... finished in 0.00 minutes

        KS_slope         alpha    x_min    x_max    qof_val  p-value
        0.00000          1.45     0.38     1.8e+02  0.0205   0.40
        0.00001          1.45     0.38     1.8e+02  0.0205   0.40
        0.00316          1.45     0.38     2.1e+02  0.0209   0.36
        0.00649          1.45     0.38     2.1e+02  0.0209   0.36
        0.00931          1.44     0.31     2.1e+02  0.0229   0.28
        0.01114          1.44     0.31     2.1e+02  0.0229   0.28
        0.01219          1.44     0.31     2.1e+02  0.0229   0.28
        0.01275          1.44     0.31     2.1e+02  0.0229   0.28
        0.01282          1.44     0.31     2.1e+02  0.0229   0.28
        0.01289          1.45     0.31     2.7e+02  0.0259   0.08
        0.01304          1.45     0.31     2.7e+02  0.0259   0.08
        0.01334          1.45     0.31     2.7e+02  0.0259   0.08
        0.05623          1.09     0.01     2.7e+02  0.1975   0.00
        1.00000          1.09     0.01     2.7e+02  0.1975   0.00

        KS_slope         alpha    x_min    x_max    qof_val  p-value  Improvement(Percentage)
        0.00000          1.45     0.38     1.8e+02  0.0205   0.40      NaN
        0.01282          1.44     0.31     2.1e+02  0.0229   0.28       45

ans =

      nr_slopes: 14
         slopes: [14x1 double]
       xmin_hat: [14x1 double]
       xmax_hat: [14x1 double]
      alpha_hat: [14x1 double]
     qof_values: [14x1 double]
      p_val_hat: [14x1 double]
    improvement: 45
          lgpvi: 9
           xmin: 0.3091
           xmax: 213.4499
          alpha: 1.4400
          p_val: 0.2800

>>
```

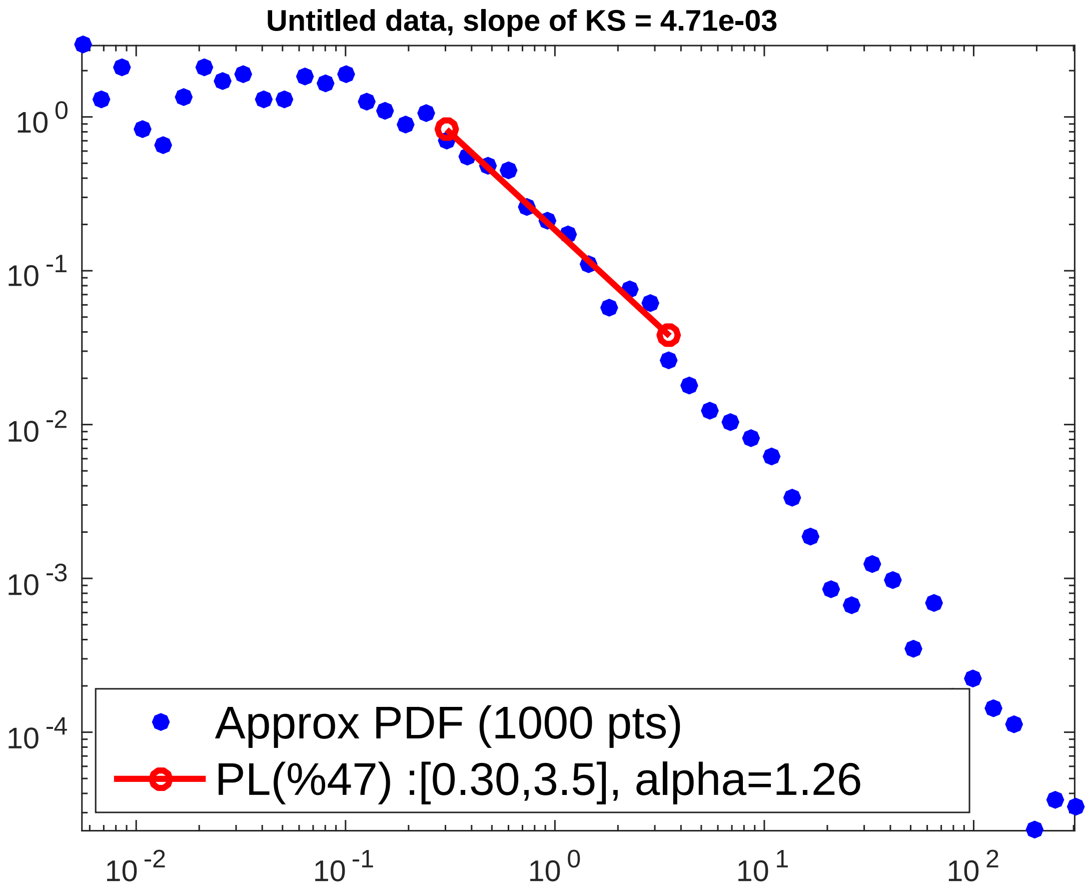


2.2 Suspiciously small power-law interval

Suppose you ran the basic apKS method but suspect the the power-law interval should be longer (e.g. Figure 2.2). Then you can try two things:

1. search for a bounded power-law that is longer than a decade long
2. consider more than the default 10 points in every decade as a candidate power-law bound (works rarely and slower)

```
>> apKS(X);
>>
```



2.2.1 Searching for a longer power-law interval

You can search for a longer power-law interval by using the following option:

```
>> apKS(X, 'interval_length_threshold', 20);
>>
```

Keep in mind that when *interval length threshhold* is increased, every aspect of the method uses the same threshold including the p-value estimation step.

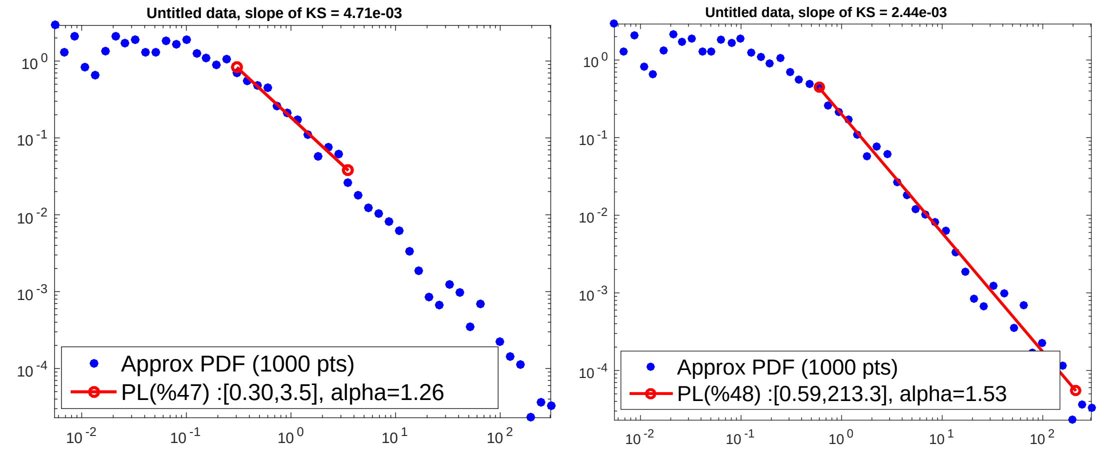


Figure 1: Data set contains a power-law interval in [1, 100] with exponent 1.5. Left half: apKS searching for a power-law interval $[x_{\min}, x_{\max}]$ with $x_{\max}/x_{\min} > 10$ (default). Right half: apKS searching for a power-law interval $[x_{\min}, x_{\max}]$ with $x_{\max}/x_{\min} > 20$

2.2.2 More candidate points

You can increase the number of points considered as a candidate power-law bound in every decade by using the following option:

```
>> apKS(X, 'min_nr_trial_pts_in_a_decade', 100);
>>
```

Keep in mind that increasing *the minimum number of trial point in a decade* by $k-$fold increases the runtime by k^2-fold. Furthermore, in practice, increasing the number of candidate points does not work most of the time and it is much slower compared to the default. This is why we adopted a default number of 10 which seems to be large enough so that increasing it does not improve results too much and small enough so that the simulations run fast.

3 Other options

3.1 Increasing number of semiparametric samples

```
>> apKS(X, 'nr_reps', 100);
>>
```

3.2 Just KS fit

If you just want the power-law fit obtained by the KS method (i.e. slope = 0) rather than the apKS method, then:

```
>> apKS(X, 'need_only_KS', 1);
>>
```

3.3 Changing p-value threshold

```
>> apKS(X, 'p_val_threshold', 0.05);
>>
```

3.4 Silencing p-value display output

This is useful when using many semiparametric samples:

```
>> apKS(X, 'nr_reps', 2500, 'display_p_val_stuff', 0);
>>
```

3.5 Silencing all display output

```
>> apKS(X, 'display_stuff', 0);
>>
```

3.6 No p-value estimation

If you just want the power-law fit obtained by the KS method and no validation, then:

```
>> apKS(X, 'need_p_val', 0);
>>
```

4 Summary of all files

apKS.m:	The apKS method
gsdf.m:	Generates synthetic data from CDF (EPL1, EPL2, EPL3, IAPL, EXP, NQPL)
papod.m:	Plots approximate PDF of data sets along with a power-law fit if provided
penKS.m:	Penalized KS for a given penalty coefficient
elspd.m:	Equally logarithmically spaced point detector
estpval.m:	Estimates p-value of a power-law fit
estexp.m:	Estimates exponent of power-law
estKS.m:	Estimates KS distance between empirical CDF and theoretical power-law CDF
gsbd.m:	Generates semi-parametric bootstrapped data