Algorithms for obtaining Simulated and Theoretical Probabilities

February 15, 2021

1 Algorithms for obtaining Simulated and Theoretical Probabilities

1.1 Section 1: The Algorithms

Here we define the functions that help us obtain the simulated and theoretical probabilities of obtaining a real root when the coefficients are sampled from $\mathcal{U}(\alpha, \beta)$, where $\alpha, \beta \in \mathbb{R}$. The formula used for calculating theoretical probabilities is derived in the paper.

[1]:
```
import math
import random
```

[2]:
```
# The number of Monte Carlo simulations to perform
trials = 50_000
```

[3]:
```
# The volume when f = 0
v0 = math.log(2) / 6 + 5.0 / 36.0
```

[4]:
```
def get_simulated_probability_for(alpha, beta):
    """
    Returns the simulated probability of obtaining a real root of
    the equation Ax^2+Bx+c=0 when A, B, C are sampled from the distribution
    uniform(alpha, beta). It performs a set number of iterations
    (equal to "trials") and returns the ratio of the cases when
    the discriminant is non-negative to the total iteration count.
    """
    lo = min(alpha, beta)
    hi = max(alpha, beta)
    count = 0
    for _ in range(trials):
        a = random.uniform(lo, hi)
        b = random.uniform(lo, hi)
        c = random.uniform(lo, hi)

        if b * b >= 4 * a * c:
            count += 1

    return float(count) / trials
```

[5]:
```python
def signum(x):
    """
    Returns the signum value for x = abs(x) / x.
    """
    if x == 0:
        return 0
    return +1 if x > 0 else -1
```

[6]:
```python
def get_theoretical_probability_for(alpha, beta):
    """
    Returns the theoretical probability of obtaining a real root of
    the equation Ax^2+Bx+c=0 when A, B, C are sampled from the distribution
    uniform(alpha, beta). It scales the given distribution to uniform(f, 1),
    and then returns the probability based on the formula derived in the paper.
    """
    aa = float(abs(alpha))
    ab = float(abs(beta))
    theta = signum(alpha) * signum(beta) / max(aa, ab)
    f = min(aa, ab) * theta

    if f == 0:
        return v0
    if f > 0.5 or f < -1:
        return 0

    volume = (1 - f)**3
    r = abs(f)
    rr = r * r
    rrr = rr * r
    r32 = math.sqrt(rrr)
    lr6 = math.log(r) / 6

    if f > +0.25: # [0.25, 0.5]
        return (-v0 - lr6 + rr - 8.0 / 9 * rrr) / volume
    if f > +0:    # [0, 0.25]
        return (+v0 - 2 * r + 16.0 / 9 * r32 + rr - 8.0 / 9 * rrr) / volume
    if f > -0.5:  # [-0.5, 0]
        return (+v0 + 2 * r + 3 * rr + rrr * (2 * v0 - lr6 - 8.0 / 9)) / volume
    else:         # [-1, -0.5]
        return (2 * (v0 + r + rr) + lr6 + rrr * (2 * v0 - lr6)) / volume
```

1.2 Section 2: Tabulation and Visualization of Data for Validation

In this section, we show that our theoretical formula lines up well with the simulated results.

[7]:
```python
import matplotlib.pyplot as plt
import numpy as np
```

```python
import pandas as pd
```

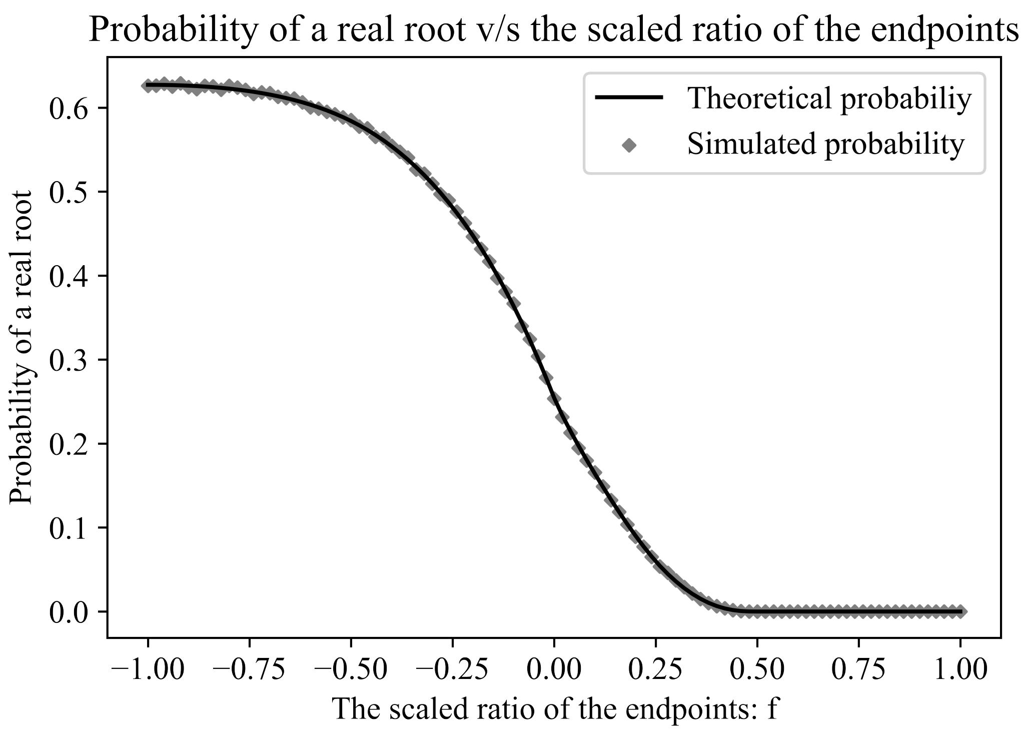
[8]:
```python
# The number of points to plot in the graph
N = 101
```

[9]:
```python
x = [f for f in np.linspace(-1, +1, num=N, endpoint=True)]
y1 = [get_simulated_probability_for(f, 1) for f in x]
y2 = [get_theoretical_probability_for(f, 1) for f in x]
```

[10]:
```python
# Configuration for the plots
%matplotlib inline
plt.rcParams["figure.dpi"] = 1000
plt.rcParams["scatter.marker"] = "D"
plt.rcParams['font.family'] = 'Times New Roman'
plt.rcParams['font.size'] = '12'

# Plot the data
plt.scatter(x, y1, color="grey", s=10.0, label="Simulated probability")
plt.plot(x, y2, color="black", label="Theoretical probabiliy")
plt.xlabel("The scaled ratio of the endpoints: f")
plt.ylabel("Probability of a real root")
plt.legend()
plt.title("Probability of a real root v/s the scaled ratio of the endpoints")
plt
```

[10]:
```
<module 'matplotlib.pyplot' from 'd:\\programs\\researchstuff\\uniform-general-
case\\venv\\lib\\site-packages\\matplotlib\\pyplot.py'>
```



[11]:
```python
pd.set_option("display.max_rows", N)
data = pd.DataFrame(data=zip(x, y1, y2), index=range(1, N + 1), columns=["f",␣
↪"Simulated", "Theoretical"])
data
```

[11]:

	f	Simulated	Theoretical
1	-1.00	0.62596	0.627207
2	-0.98	0.62618	0.627169
3	-0.96	0.62858	0.627053
4	-0.94	0.62512	0.626854
5	-0.92	0.62890	0.626567
6	-0.90	0.62442	0.626185
7	-0.88	0.62210	0.625703
8	-0.86	0.62612	0.625114
9	-0.84	0.62540	0.624412
10	-0.82	0.62156	0.623587
11	-0.80	0.62608	0.622633
12	-0.78	0.62396	0.621540
13	-0.76	0.62132	0.620299
14	-0.74	0.61632	0.618898
15	-0.72	0.61836	0.617326
16	-0.70	0.61728	0.615571
17	-0.68	0.61298	0.613618

18	-0.66	0.61132	0.611454
19	-0.64	0.61086	0.609060
20	-0.62	0.60642	0.606420
21	-0.60	0.60048	0.603512
22	-0.58	0.59886	0.600315
23	-0.56	0.59502	0.596805
24	-0.54	0.59216	0.592953
25	-0.52	0.58842	0.588731
26	-0.50	0.58532	0.584103
27	-0.48	0.57744	0.579035
28	-0.46	0.57530	0.573502
29	-0.44	0.56538	0.567483
30	-0.42	0.56362	0.560952
31	-0.40	0.55456	0.553886
32	-0.38	0.54744	0.546259
33	-0.36	0.54060	0.538044
34	-0.34	0.52646	0.529213
35	-0.32	0.52138	0.519739
36	-0.30	0.50950	0.509590
37	-0.28	0.49712	0.498737
38	-0.26	0.48982	0.487148
39	-0.24	0.47614	0.474791
40	-0.22	0.46244	0.461632
41	-0.20	0.44636	0.447638
42	-0.18	0.43180	0.432775
43	-0.16	0.41688	0.417009
44	-0.14	0.39708	0.400305
45	-0.12	0.38088	0.382630
46	-0.10	0.36680	0.363950
47	-0.08	0.33988	0.344233
48	-0.06	0.32428	0.323449
49	-0.04	0.30406	0.301568
50	-0.02	0.27852	0.278565
51	0.00	0.25350	0.254413
52	0.02	0.23152	0.233570
53	0.04	0.21290	0.214955
54	0.06	0.19464	0.197390
55	0.08	0.17988	0.180541
56	0.10	0.16558	0.164256
57	0.12	0.14886	0.148470
58	0.14	0.13268	0.133164
59	0.16	0.11864	0.118356
60	0.18	0.10346	0.104094
61	0.20	0.08908	0.090452
62	0.22	0.07724	0.077538
63	0.24	0.06506	0.065490
64	0.26	0.05350	0.054478

```
65   0.28   0.04606   0.044567
66   0.30   0.03700   0.035711
67   0.32   0.02918   0.027876
68   0.34   0.02152   0.021048
69   0.36   0.01484   0.015220
70   0.38   0.01000   0.010387
71   0.40   0.00632   0.006541
72   0.42   0.00396   0.003659
73   0.44   0.00154   0.001694
74   0.46   0.00038   0.000554
75   0.48   0.00008   0.000077
76   0.50   0.00000   0.000000
77   0.52   0.00000   0.000000
78   0.54   0.00000   0.000000
79   0.56   0.00000   0.000000
80   0.58   0.00000   0.000000
81   0.60   0.00000   0.000000
82   0.62   0.00000   0.000000
83   0.64   0.00000   0.000000
84   0.66   0.00000   0.000000
85   0.68   0.00000   0.000000
86   0.70   0.00000   0.000000
87   0.72   0.00000   0.000000
88   0.74   0.00000   0.000000
89   0.76   0.00000   0.000000
90   0.78   0.00000   0.000000
91   0.80   0.00000   0.000000
92   0.82   0.00000   0.000000
93   0.84   0.00000   0.000000
94   0.86   0.00000   0.000000
95   0.88   0.00000   0.000000
96   0.90   0.00000   0.000000
97   0.92   0.00000   0.000000
98   0.94   0.00000   0.000000
99   0.96   0.00000   0.000000
100  0.98   0.00000   0.000000
101  1.00   0.00000   0.000000
```

[12]:
```
def get_data_row_for(alpha, beta):
    aa = abs(alpha)
    ab = abs(beta)
    f =  signum(alpha) * signum(beta) * min(aa, ab) / max(aa, ab)
    return [
        alpha,
        beta,
        f,
        get_simulated_probability_for(alpha, beta),
```

```
        get_theoretical_probability_for(alpha, beta)
    ]
```

[13]:

```
ranges = [
    (0, 1.0),
    (-1, 1),
    (0, 10),
    (0.5, 1),
    (5, 10),
    (-20, 300),
    (200, 1000),
    (-300, -4000),
    (math.e, -math.pi),
    (35.3, 1e8)
]

rows = []
cols = ["alpha", "beta", "f", "Simulated", "Theoretical"]
for r in ranges:
    alpha, beta = r
    rows.append(get_data_row_for(alpha, beta))
```

[14]:

```
df = pd.DataFrame(data=rows, index=range(1, len(ranges) + 1), columns=cols)
df
```

[14]:

	alpha	beta	f	Simulated	Theoretical
1	0.000000	1.000000e+00	0.000000e+00	0.25308	0.254413
2	-1.000000	1.000000e+00	-1.000000e+00	0.62426	0.627207
3	0.000000	1.000000e+01	0.000000e+00	0.25422	0.254413
4	0.500000	1.000000e+00	5.000000e-01	0.00000	0.000000
5	5.000000	1.000000e+01	5.000000e-01	0.00000	0.000000
6	-20.000000	3.000000e+02	-6.666667e-02	0.33308	0.330497
7	200.000000	1.000000e+03	2.000000e-01	0.09018	0.090452
8	-300.000000	-4.000000e+03	7.500000e-02	0.18580	0.184696
9	2.718282	-3.141593e+00	-8.652560e-01	0.62892	0.625280
10	35.300000	1.000000e+08	3.530000e-07	0.25164	0.254413

[ ]: