

EIOPA RISK-FREE CURVE APRIL-23 RECALCULATION

The risk-free curve is one of the principal inputs into an economic scenario generator. This notebook recalculates the risk-free curve using the parameters that are claimed to be used. The European Insurance and Occupational Pensions Authority (EIOPA) publishes their own yield curve prediction. To do this they use the Smith & Wilson algorithm.

Summary

The goal of this test is to replicate the EIOPA yield curve. This test will use the methodology that EIOPA claims it is using and the calibration vector that they publish. If the test is passed, the user can be more confident, that EIOPA risk free rate (RFR) curve was generated using the described methodology/calibration and that the process was implemented correctly.

Table of Contents

1. [Note on Smith & Wilson algorithm](#)
2. [Data requirements](#)
3. [Success criteria](#)
4. [External dependencies](#)
5. [Calibration parameters and calibration vector provided by EIOPA](#)
6. [Smith & Wilson calculation functions](#)
7. [Generation of the risk-free curve](#)
8. [Test 1; Comparison test](#)
9. [Test 1; Success criteria](#)
10. [Test 1; Comparison test](#)
11. [Conclusion](#)

Note on Smith & Wilson algorithm

To replicate the calculations, this example uses a modified Smith&Wilson implementation (The original implementation is available on [GitHub](#):

- [Python](#)
- [Matlab](#)
- [JavaScript](#)

Limitations of the implementation

Current implementation only looks at a single currency and with/without Volatility Adjustment (VA). The day count convention assumes that each year has the same number of days.

Data requirements

This script contains the EIOPA risk-free rate publication for April 2023. The publication can be found on the [EIOPA RFR website](#).

The observed maturities `M_Obs` and the calibrated vector `Qb` can be found in the Excel sheet `EIOPA_RFR_20230430_Qb_SW.xlsx`.

The target maturities (`T_Obs`), the additional parameters (`UFR` and `alpha`), and the given curve can be found in the Excel `EIOPA_RFR_20230430_Term_Structures.xlsx`, sheet `RFR_spot_no_VA` if the test looks at the curve without the Volatility Adjustment and the sheet `RFR_spot_with_VA` if the test looks at the curve with the Volatility Adjustment.

[Back to the top](#)

Success criteria

The following success criteria is defined:

- Maximum difference between the calculated curve and the one provided by EIOPA is less than 0.1 bps
- Average difference between the calculated curve and the one provided by EIOPA is less than 0.05 bps

In [280...]

```
test_statistics_max_diff_in_bps = 0.1
test_statistics_average_diff_in_bps = 0.05
```

The success function is called at the end of the test to confirm if the success criteria have been met.

In [281...]

```
def SuccessTest(TestStatistics, threshold_max, threshold_mean):  
    out1 = False  
    out2 = False  
    if max(TestStatistics) < threshold_max:  
        print("Test passed")  
        out1 = True  
    else:  
        print("Test failed")  
  
    if np.mean(TestStatistics) < threshold_mean:  
        print("Test passed")  
        out2 = True  
    else:  
        print("Test failed")  
    return [out1, out2]
```

[Back to the top](#)

External dependencies

This implementation uses three well established Python packages widely used in the financial industry. Pandas (<https://pandas.pydata.org/docs/>), Numpy (<https://numpy.org/doc/>), and Matplotlib (<https://matplotlib.org/stable/index.html>)

In [282...]

```
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import matplotlib.ticker as mtick  
%matplotlib notebook  
pd.options.display.max_rows = 150
```

Importing data

In [283...]

```
selected_param_file = 'Param_VA.csv'  
selected_curves_file = 'Curves_VA.csv'  
  
#selected_param_file = 'Param_no_VA.csv'  
#selected_curves_file = 'Curves_no_VA.csv'
```

In [284...]

```
param_raw = pd.read_csv(selected_param_file, sep=',', index_col=0)
```

Parameter input

Parameters sheet

In [285...]

```
param_raw.head()
```

Out[285...]

| | Euro_Maturities | Euro_Values | Austria_Maturities | Austria_Values | Belgium_Maturities | Be |
|--------------------|-----------------|-------------|--------------------|----------------|--------------------|-----------|
| Country | | | | | | |
| Coupon_freq | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| LLP | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 20.000000 | 20.000000 |
| Convergence | 40.000000 | 40.000000 | 40.000000 | 40.000000 | 40.000000 | 40.000000 |
| UFR | 3.450000 | 3.450000 | 3.450000 | 3.450000 | 3.450000 | 3.450000 |
| alpha | 0.111906 | 0.111906 | 0.111906 | 0.111906 | 0.111906 | 0.111906 |

5 rows × 106 columns

The country selected is:

In [286...]

```
country = "Slovenia"
```

In [287...]

```
maturities_country_raw = param_raw.loc[:,country+"_Maturities"].iloc[6:]
param_country_raw = param_raw.loc[:,country + "_Values"].iloc[6:]
extra_param = param_raw.loc[:,country + "_Values"].iloc[:6]
```

Extra parameters

Smith-Wilson calibration parameters

In [288...]

```
extra_param
```

Out[288...]

| Country | |
|-------------|-----------|
| Coupon_freq | 1.000000 |
| LLP | 20.000000 |
| Convergence | 40.000000 |
| UFR | 3.450000 |
| alpha | 0.111906 |
| CRA | 10.000000 |

Name: Slovenia_Values, dtype: float64

In [289...]

```
relevant_positions = pd.notna(maturities_country_raw.values)
```

In [290...]

```
maturities_country = maturities_country_raw.iloc[relevant_positions]
```

Maturity vector

Vector of maturities used in the calibration

In [291...]

```
maturities_country.head(15)
```

Out[291...]

```
Country
1      1.0
2      2.0
3      3.0
4      4.0
5      5.0
6      6.0
7      7.0
8      8.0
9      9.0
10     10.0
11     11.0
12     12.0
13     13.0
14     14.0
15     15.0
Name: Slovenia_Maturities, dtype: float64
```

In [292...]

```
Qb = param_country_raw.iloc[relevant_positions]
```

Calibration vector

Vector **Qb** provided as input

In [293...]

```
Qb
```

Out[293...]

```
Country
1      -8.945909
2       0.541504
3      1.925282
4      1.058779
5      -0.475350
6      -0.387083
7       0.712179
8      -0.383027
9      -0.393004
10     2.715405
11     -3.918373
12     2.701016
13     -0.034363
14     -0.031608
15     -1.692104
16      0.023894
```

```
17    0.051802
18   -0.063782
19    0.361738
20    0.440014
Name: Slovenia Values, dtype: float64
```

```
In [294...]: curve_raw = pd.read_csv(selected_curves_file, sep=',', index_col=0)
```

```
In [295...]: curve_country = curve_raw.loc[:,country]
```

[Back to the top](#)

Calibration parameters and calibration vector provided by EIOPA

```
In [296...]: # Maturity of observations:
M_Obs = np.transpose(np.array(maturities_country.values))

# Ultimate forward rate ufr represents the rate to which the rate curve will converge
ufr = extra_param.iloc[3]/100

# Convergence speed parameter alpha controls the speed at which the curve converges
alpha = extra_param.iloc[4]

# For which maturities do we want the SW algorithm to calculate the rates. In this case, we want rates for all maturities
M_Target = np.transpose(np.arange(1,151))

# Qb calibration vector published by EIOPA for the curve calibration:
Qb = np.transpose(np.array(Qb.values))
```

[Back to the top](#)

Smith & Wilson calculation functions

In this step, the independent version of the Smith&Wilson algorithm is implemented. To do this, two functions are taken from the publicly available repository and modified to accept the product of Q^*b instead of the calibration vector b .

In [297...]

```

def SWExtrapolate(M_Target, M_Obs, Qb, ufr, alpha):
    # SWEXTRAPOLATE Interpolate or/and extrapolate rates for targeted maturities using a
    # out = SWExtrapolate(M_Target, M_Obs, Qb, ufr, alpha) calculates the rates for maturities
    #
    # Arguments:
    #     M_Target = k x 1 ndarray. Each element represents a bond maturity of interest. Ex. M_Target = [1; 2; 3]
    #     M_Obs = n x 1 ndarray. Observed bond maturities used for calibrating the curve.
    #     Qb = n x 1 ndarray. Calibration vector calculated on observed bonds.
    #     ufr = 1 x 1 floating number. Representing the ultimate forward rate.
    #         Ex. ufr = 0.042
    #     alpha = 1 x 1 floating number. Representing the convergence speed parameter alpha.
    #
    #
    # Returns:
    #     k x 1 ndarray. Represents the targeted rates for a zero-coupon bond. Each rate is
    #
    # For more information see https://www.eiopa.europa.eu/sites/default/files/risk_free_curve.ipynb

def SWHeart(u, v, alpha):
    # SWHEART Calculate the heart of the Wilson function.
    # H = SWHeart(u, v, alpha) calculates the matrix H (Heart of the Wilson
    # function) for maturities specified by vectors u and v. The formula is
    # taken from the EIOPA technical specifications paragraph 132.
    #
    # Arguments:
    #     u = n_1 x 1 vector of maturities. Ex. u = [1; 3]
    #     v = n_2 x 1 vector of maturities. Ex. v = [1; 2; 3; 5]
    #     alpha = 1 x 1 floating number representing the convergence speed parameter alpha.
    #
    # Returns:
    #     n_1 x n_2 matrix representing the Heart of the Wilson function for selected
    #
    # For more information see https://www.eiopa.europa.eu/sites/default/files/risk_free_curve.ipynb

    u_Mat = np.tile(u, [v.size, 1]).transpose()
    v_Mat = np.tile(v, [u.size, 1])
    return 0.5 * (alpha * (u_Mat + v_Mat) + np.exp(-alpha * (u_Mat + v_Mat)) - alpha)

H = SWHeart(M_Target, M_Obs, alpha) # Heart of the Wilson function from paragraph 132
p = np.exp(-np.log(1+ufr)* M_Target) + np.diag(np.exp(-np.log(1+ufr) * M_Target))
return p ** (-1/ M_Target) -1 # Convert obtained prices to rates and return price

```

[Back to the top](#)

Generation of the risk-free curve

The observed maturities, target maturities, and the model parameters provided by EIOPA are used to generate the target curve.

In [298...]

```

r_Target = SWExtrapolate(M_Target,M_Obs, Qb, ufr, alpha)
r_Target = pd.DataFrame(r_Target,columns=['Recalculated rates'])

```

Yield curve calculated

Yield curve calculated using the calibration vector **Qb**

In [299...]

```
r_Target.head(15)
```

Out[299...]

Recalculated rates

| | |
|----|----------|
| 0 | 0.038530 |
| 1 | 0.035419 |
| 2 | 0.033076 |
| 3 | 0.031777 |
| 4 | 0.031117 |
| 5 | 0.030734 |
| 6 | 0.030516 |
| 7 | 0.030446 |
| 8 | 0.030461 |
| 9 | 0.030552 |
| 10 | 0.030700 |
| 11 | 0.030764 |
| 12 | 0.030832 |
| 13 | 0.030851 |
| 14 | 0.030747 |

[Back to the top](#)

Test 1; Comparison test

Comparison of the calculated yield curve with the yield curve provided by EIOPA. The test is passed if the success criteria is reached.

The provided yield curve can be found in file *EIOPA_RFR_20230430_Term_Structures.xlsx*, sheet *RFR_spot_no_VA* if the test looks at the curve without the Volatility Adjustment and the sheet *RFR_spot_with_VA* if the test looks at the curve with the Volatility Adjustment.

In [300...]

```
target_curve = np.transpose(np.array(curve_country.values))
```

This implementation looks at two kinds of test statistics. The average deviation and the maximum deviation.

The average deviation is defined as:

$$S_{AVERAGE} = \frac{1}{T} \sum_{t=0}^T |r_{EIOPA}(t) - r_{EST}(t)|$$

The maximum deviation is defined as:

$$S_{MAX} = \max_t |r_{EIOPA}(t) - r_{EST}(t)|$$

Where T is the maximum maturity available.

The average difference test is successful if:

$$S_{AVERAGE} < 0.05bps$$

The maximum difference test is successful if:

$$S_{MAX} < 0.1bps$$

In [301...]

```
target_curve = pd.DataFrame(target_curve,columns=['Given rates'])
```

EIOPA curve provided

Yield curve provided by EIOPA

In [302...]

```
target_curve.head()
```

Out[302...]

| | Given rates |
|---|-------------|
| 0 | 0.03853 |
| 1 | 0.03542 |
| 2 | 0.03308 |
| 3 | 0.03178 |
| 4 | 0.03112 |

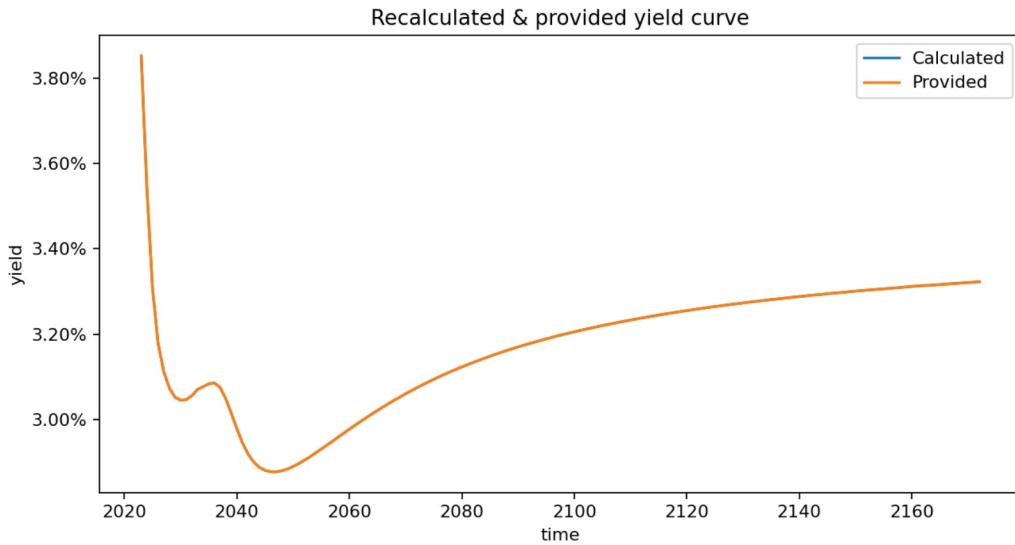
In [303...]

```
x_data_label = range(2023,2023+r_Target.shape[0],1)
```

In [304...]

```
fig, ax1 = plt.subplots(1,1)
ax1.plot(x_data_label, r_Target.values*100, color='tab:blue',label="Calculated")
ax1.plot(x_data_label, target_curve.values*100, color='tab:orange',label="Provided")

ax1.set_ylabel("yield")
ax1.set_title('Recalculated & provided yield curve')
ax1.set_xlabel("time")
ax1.legend()
ax1.yaxis.set_major_formatter(mtick.PercentFormatter())
fig.set_figwidth(6)
fig.set_figheight(3)
plt.show()
```



In [305...]

```
test_statistics_bdp = pd.DataFrame(abs(r_Target.values-target_curve.values)*10000, co
```

EIOPA curve comparison

Absolute difference in bps

In [306...]

```
test_statistics_bdp.head()
```

Out[306...]

| | Abs diff in bps |
|---|-----------------|
| 0 | 0.000165 |
| 1 | 0.013756 |
| 2 | 0.040949 |
| 3 | 0.029202 |

Abs diff in bps[Back to the top](#)

Test 1; Success criteria

The successful application of the success criteria marks the completion of the test.

In [307...]

```
result1 = SuccessTest(test_statistics_bdp.values, test_statistics_max_diff_in_bps, tolerance)
```

Test passed

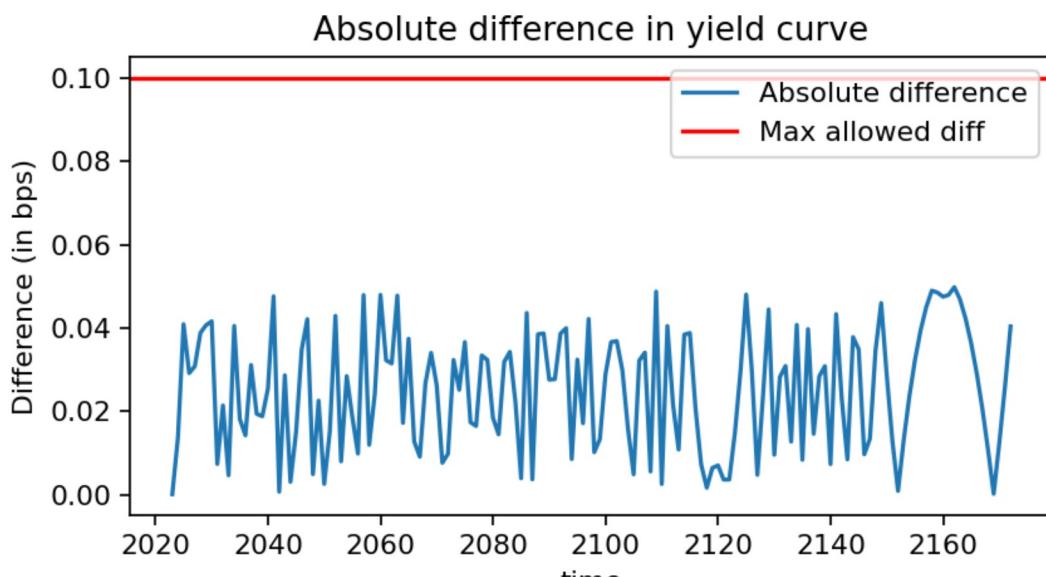
Test passed

In [308...]

```
x_data_label = range(2023,2023+r_Target.shape[0],1)
fig, ax1 = plt.subplots(1,1)
ax1.plot(x_data_label, test_statistics_bdp, label= "Absolute difference")
ax1.axhline(y = test_statistics_max_diff_in_bps, color = 'r', linestyle = '--',label='Max allowed diff')

ax1.set_xlabel("time")
ax1.set_ylabel("Difference (in bps)")
ax1.set_title('Absolute difference in yield curve')
ax1.legend()
fig.set_figwidth(6)
fig.set_figheight(3)

plt.show()
```



[Back to the top](#)

Conclusion

This test checks the success criteria on the EIOPA curve generated for April 2023. If the tests are passed, it is likely that the curve was generated using the Smith & Wilson algorithm with the calibration vector that was provided in the file *EIOPA_RFR_20230430_Qb_SW.xlsx* and the parameters displayed in the file *EIOPA_RFR_20230430_Term_Structures.xlsx*.

In [309...]

```
pd.DataFrame(data = [result1], columns = ["Mean test", "Max test"], \
              index= ["Provided vs calculated"])
```

Out[309...]

| | Mean test | Max test |
|--|-----------|----------|
|--|-----------|----------|

| | | |
|------------------------|------|------|
| Provided vs calculated | True | True |
|------------------------|------|------|

Final yield curve

Full yield curve provided by EIOPA in %

In [310...]

```
(curve_country*100).head(150)
```

Out[310...]

| Country | |
|---------|-------|
| 1 | 3.853 |
| 2 | 3.542 |
| 3 | 3.308 |
| 4 | 3.178 |
| 5 | 3.112 |
| 6 | 3.073 |
| 7 | 3.052 |
| 8 | 3.045 |
| 9 | 3.046 |
| 10 | 3.055 |
| 11 | 3.070 |
| 12 | 3.076 |
| 13 | 3.083 |
| 14 | 3.085 |
| 15 | 3.075 |
| 16 | 3.049 |
| 17 | 3.014 |
| 18 | 2.978 |
| 19 | 2.944 |
| 20 | 2.918 |
| 21 | 2.899 |
| 22 | 2.887 |
| 23 | 2.880 |
| 24 | 2.877 |
| 25 | 2.877 |
| 26 | 2.879 |
| 27 | 2.883 |
| 28 | 2.889 |

| | |
|----|-------|
| 29 | 2.896 |
| 30 | 2.904 |
| 31 | 2.912 |
| 32 | 2.921 |
| 33 | 2.930 |
| 34 | 2.939 |
| 35 | 2.948 |
| 36 | 2.958 |
| 37 | 2.967 |
| 38 | 2.977 |
| 39 | 2.986 |
| 40 | 2.995 |
| 41 | 3.004 |
| 42 | 3.012 |
| 43 | 3.021 |
| 44 | 3.029 |
| 45 | 3.037 |
| 46 | 3.045 |
| 47 | 3.052 |
| 48 | 3.060 |
| 49 | 3.067 |
| 50 | 3.074 |
| 51 | 3.081 |
| 52 | 3.087 |
| 53 | 3.094 |
| 54 | 3.100 |
| 55 | 3.106 |
| 56 | 3.112 |
| 57 | 3.117 |
| 58 | 3.123 |
| 59 | 3.128 |
| 60 | 3.133 |
| 61 | 3.138 |
| 62 | 3.143 |
| 63 | 3.148 |
| 64 | 3.153 |
| 65 | 3.157 |
| 66 | 3.161 |
| 67 | 3.166 |
| 68 | 3.170 |
| 69 | 3.174 |
| 70 | 3.178 |
| 71 | 3.181 |
| 72 | 3.185 |
| 73 | 3.189 |
| 74 | 3.192 |
| 75 | 3.196 |
| 76 | 3.199 |
| 77 | 3.202 |
| 78 | 3.205 |
| 79 | 3.208 |
| 80 | 3.211 |
| 81 | 3.214 |
| 82 | 3.217 |
| 83 | 3.220 |
| 84 | 3.223 |
| 85 | 3.225 |
| 86 | 3.228 |
| 87 | 3.230 |

| | |
|-----|-------|
| 88 | 3.233 |
| 89 | 3.235 |
| 90 | 3.238 |
| 91 | 3.240 |
| 92 | 3.242 |
| 93 | 3.245 |
| 94 | 3.247 |
| 95 | 3.249 |
| 96 | 3.251 |
| 97 | 3.253 |
| 98 | 3.255 |
| 99 | 3.257 |
| 100 | 3.259 |
| 101 | 3.261 |
| 102 | 3.263 |
| 103 | 3.265 |
| 104 | 3.266 |
| 105 | 3.268 |
| 106 | 3.270 |
| 107 | 3.271 |
| 108 | 3.273 |
| 109 | 3.275 |
| 110 | 3.276 |
| 111 | 3.278 |
| 112 | 3.279 |
| 113 | 3.281 |
| 114 | 3.282 |
| 115 | 3.284 |
| 116 | 3.285 |
| 117 | 3.287 |
| 118 | 3.288 |
| 119 | 3.289 |
| 120 | 3.291 |
| 121 | 3.292 |
| 122 | 3.293 |
| 123 | 3.295 |
| 124 | 3.296 |
| 125 | 3.297 |
| 126 | 3.298 |
| 127 | 3.300 |
| 128 | 3.301 |
| 129 | 3.302 |
| 130 | 3.303 |
| 131 | 3.304 |
| 132 | 3.305 |
| 133 | 3.306 |
| 134 | 3.307 |
| 135 | 3.308 |
| 136 | 3.309 |
| 137 | 3.311 |
| 138 | 3.312 |
| 139 | 3.313 |
| 140 | 3.314 |
| 141 | 3.314 |
| 142 | 3.315 |
| 143 | 3.316 |
| 144 | 3.317 |
| 145 | 3.318 |
| 146 | 3.319 |

147 3.320
148 3.321
149 3.322
150 3.323
151 3.324
152 3.325
153 3.326
154 3.327
155 3.328
156 3.329
157 3.330
158 3.331
159 3.332
160 3.333
161 3.334
162 3.335
163 3.336
164 3.337
165 3.338
166 3.339
167 3.340
168 3.341
169 3.342
170 3.343
171 3.344
172 3.345
173 3.346
174 3.347
175 3.348
176 3.349
177 3.350
178 3.351
179 3.352
180 3.353
181 3.354
182 3.355
183 3.356
184 3.357
185 3.358
186 3.359
187 3.360
188 3.361
189 3.362
190 3.363
191 3.364
192 3.365
193 3.366
194 3.367
195 3.368
196 3.369
197 3.370
198 3.371
199 3.372
200 3.373
201 3.374
202 3.375
203 3.376
204 3.377
205 3.378
206 3.379
207 3.380
208 3.381
209 3.382
210 3.383
211 3.384
212 3.385
213 3.386
214 3.387
215 3.388
216 3.389
217 3.390
218 3.391
219 3.392
220 3.393
221 3.394
222 3.395
223 3.396
224 3.397
225 3.398
226 3.399
227 3.400
228 3.401
229 3.402
230 3.403
231 3.404
232 3.405
233 3.406
234 3.407
235 3.408
236 3.409
237 3.410
238 3.411
239 3.412
240 3.413
241 3.414
242 3.415
243 3.416
244 3.417
245 3.418
246 3.419
247 3.420
248 3.421
249 3.422
250 3.423
251 3.424
252 3.425
253 3.426
254 3.427
255 3.428
256 3.429
257 3.430
258 3.431
259 3.432
260 3.433
261 3.434
262 3.435
263 3.436
264 3.437
265 3.438
266 3.439
267 3.440
268 3.441
269 3.442
270 3.443
271 3.444
272 3.445
273 3.446
274 3.447
275 3.448
276 3.449
277 3.450
278 3.451
279 3.452
280 3.453
281 3.454
282 3.455
283 3.456
284 3.457
285 3.458
286 3.459
287 3.460
288 3.461
289 3.462
290 3.463
291 3.464
292 3.465
293 3.466
294 3.467
295 3.468
296 3.469
297 3.470
298 3.471
299 3.472
300 3.473
301 3.474
302 3.475
303 3.476
304 3.477
305 3.478
306 3.479
307 3.480
308 3.481
309 3.482
310 3.483
311 3.484
312 3.485
313 3.486
314 3.487
315 3.488
316 3.489
317 3.490
318 3.491
319 3.492
320 3.493
321 3.494
322 3.495
323 3.496
324 3.497
325 3.498
326 3.499
327 3.500
328 3.501
329 3.502
330 3.503
331 3.504
332 3.505
333 3.506
334 3.507
335 3.508
336 3.509
337 3.510
338 3.511
339 3.512
340 3.513
341 3.514
342 3.515
343 3.516
344 3.517
345 3.518
346 3.519
347 3.520
348 3.521
349 3.522
350 3.523
351 3.524
352 3.525
353 3.526
354 3.527
355 3.528
356 3.529
357 3.530
358 3.531
359 3.532
360 3.533
361 3.534
362 3.535
363 3.536
364 3.537
365 3.538
366 3.539
367 3.540
368 3.541
369 3.542
370 3.543
371 3.544
372 3.545
373 3.546
374 3.547
375 3.548
376 3.549
377 3.550
378 3.551
379 3.552
380 3.553
381 3.554
382 3.555
383 3.556
384 3.557
385 3.558
386 3.559
387 3.560
388 3.561
389 3.562
390 3.563
391 3.564
392 3.565
393 3.566
394 3.567
395 3.568
396 3.569
397 3.570
398 3.571
399 3.572
400 3.573
401 3.574
402 3.575
403 3.576
404 3.577
405 3.578
406 3.579
407 3.580
408 3.581
409 3.582
410 3.583
411 3.584
412 3.585
413 3.586
414 3.587
415 3.588
416 3.589
417 3.590
418 3.591
419 3.592
420 3.593
421 3.594
422 3.595
423 3.596
424 3.597
425 3.598
426 3.599
427 3.600
428 3.601
429 3.602
430 3.603
431 3.604
432 3.605
433 3.606
434 3.607
435 3.608
436 3.609
437 3.610
438 3.611
439 3.612
440 3.613
441 3.614
442 3.615
443 3.616
444 3.617
445 3.618
446 3.619
447 3.620
448 3.621
449 3.622
450 3.623
451 3.624
452 3.625
453 3.626
454 3.627
455 3.628
456 3.629
457 3.630
458 3.631
459 3.632
460 3.633
461 3.634
462 3.635
463 3.636
464 3.637
465 3.638
466 3.639
467 3.640
468 3.641
469 3.642
470 3.643
471 3.644
472 3.645
473 3.646
474 3.647
475 3.648
476 3.649
477 3.650
478 3.651
479 3.652
480 3.653
481 3.654
482 3.655
483 3.656
484 3.657
485 3.658
486 3.659
487 3.660
488 3.661
489 3.662
490 3.663
491 3.664
492 3.665
493 3.666
494 3.667
495 3.668
496 3.669
497 3.670
498 3.671
499 3.672
500 3.673
501 3.674
502 3.675
503 3.676
504 3.677
505 3.678
506 3.679
507 3.680
508 3.681
509 3.682
510 3.683
511 3.684
512 3.685
513 3.686
514 3.687
515 3.688
516 3.689
517 3.690
518 3.691
519 3.692
520 3.693
521 3.694
522 3.695
523 3.696
524 3.697
525 3.698
526 3.699
527 3.700
528 3.701
529 3.702
530 3.703
531 3.704
532 3.705
533 3.706
534 3.707
535 3.708
536 3.709
537 3.710
538 3.711
539 3.712
540 3.713
541 3.714
542 3.715
543 3.716
544 3.717
545 3.718
546 3.719
547 3.720
548 3.721
549 3.722
550 3.723
551 3.724
552 3.725
553 3.726
554 3.727
555 3.728
556 3.729
557 3.730
558 3.731
559 3.732
560 3.733
561 3.734
562 3.735
563 3.736
564 3.737
565 3.738
566 3.739
567 3.740
568 3.741
569 3.742
570 3.743
571 3.744
572 3.745
573 3.746
574 3.747
575 3.748
576 3.749
577 3.750
578 3.751
579 3.752
580 3.753
581 3.754
582 3.755
583 3.756
584 3.757
585 3.758
586 3.759
587 3.760
588 3.761
589 3.762
590 3.763
591 3.764
592 3.765
593 3.766
594 3.767
595 3.768
596 3.769
597 3.770
598 3.771
599 3.772
600 3.773
601 3.774
602 3.775
603 3.776
604 3.777
605 3.778
606 3.779
607 3.780
608 3.781
609 3.782
610 3.783
611 3.784
612 3.785
613 3.786
614 3.787
615 3.788
616 3.789
617 3.790
618 3.791
619 3.792
620 3.793
621 3.794
622 3.795
623 3.796
624 3.797
625 3.798
626 3.799
627 3.800
628 3.801
629 3.802
630 3.803
631 3.804
632 3.805
633 3.806
634 3.807
635 3.808
636 3.809
637 3.810
638 3.811
639 3.812
640 3.813
641 3.814
642 3.815
643 3.816
644 3.817
645 3.818
646 3.819
647 3.820
648 3.821
649 3.822
650 3.823
651 3.824
652 3.825
653 3.826
654 3.827
655 3.828
656 3.829
657 3.830
658 3.831
659 3.832
660 3.833
661 3.834
662 3.835
663 3.836
664 3.837
665 3.838
666 3.839
667 3.840
668 3.841
669 3.842
670 3.843
671 3.844
672 3.845
673 3.846
674 3.847
675 3.848
676 3.849
677 3.850
678 3.851
679 3.852
680 3.853
681 3.854
682 3.855
683 3.856
684 3.857
685 3.858
686 3.859
687 3.860
688 3.861
689 3.862
690 3.863
691 3.864
692 3.865
693 3.866
694 3.867
695 3.868
696 3.869
697 3.870
698 3.871
699 3.872
700 3.873
701 3.874
702 3.875
703 3.876
704 3.877
705 3.878
706 3.879
707 3.880
708 3.881
709 3.882
710 3.883
711 3.884
712 3.885
713 3.886
714 3.887
715 3.888
716 3.889
717 3.890
718 3.891
719 3.892
720 3.893
721 3.894
722 3.895
723 3.896
724 3.897
725 3.898
726 3.899
727 3.900
728 3.901
729 3.902
730 3.903
731 3.904
732 3.905
733 3.906
734 3.907
735 3.908
736 3.909
737 3.910
738 3.911
739 3.912
740 3.913
741 3.914
742 3.915
743 3.916
744 3.917
745 3.918
746 3.919
747 3.920
748 3.921
749 3.922
750 3.923
751 3.924
752 3.925
753 3.926
754 3.927
755 3.928
756 3.929
757 3.930
758 3.931
759 3.932
760 3.933
761 3.934
762 3.935
763 3.936
764 3.937
765 3.938
766 3.939
767 3.940
768 3.941
769 3.942
770 3.943
771 3.944
772 3.945
773 3.946
774 3.947
775 3.948
776 3.949
777 3.950
778 3.951
779 3.952
780 3.953
781 3.954
782 3.955
783 3.956
784 3.957
785 3.958
786 3.959
787 3.960
788 3.961
789 3.962
790 3.963
791 3.964
792 3.965
793 3.966
794 3.967
795 3.968
796 3.969
797 3.970
798 3.971
799 3.972
800 3.973
801 3.974
802 3.975
803 3.976
804 3.977
805 3.978
806 3.979
807 3.980
808 3.981
809 3.982
810 3.983
811 3.984
812 3.985
813 3.986
814 3.987
815 3.988
816 3.989
817 3.990
818 3.991
819 3.992
820 3.993
821 3.994
822 3.995
823 3.996
824 3.997
825 3.998
826 3.999
827 4.000
828 4.001
829 4.002
830 4.003
831 4.004
832 4.005
833 4.006
834 4.007
835 4.008
836 4.009
837 4.010
838 4.011
839 4.012
840 4.013
841 4.014
842 4.015
843 4.016
844 4.017
845 4.018
846 4.019
847 4.020
848 4.021
849 4.022
850 4.023
851 4.024
852 4.025
853 4.026
854 4.027
855 4.028
856 4.029
857 4.030
858 4.031
859 4.032
860 4.033
861 4.034
862 4.035
863 4.036
864 4.037
865 4.038
866 4.039
867 4.040
868 4.041
869 4.042
870 4.043
871 4.044
872 4.045
873 4.046
874 4.047
875 4.048
876 4.049
877 4.050
878 4.051
879 4.052
880 4.053
881 4.054
882 4.055
883 4.056
884 4.057
885 4.058
886 4.059
887 4.060
888 4.061
889 4.062
890 4.063
891 4.064
892 4.065
893 4.066
894 4.067
895 4.068
896 4.069
897 4.070
898 4.071
899 4.072
900 4.073
901 4.074
902 4.075
903 4.076
904 4.077
905 4.078
906 4.079
907 4.080
908 4.081
909 4.082
910 4.083
911 4.084
912 4.085
913 4.086
914 4.087
915 4.088
916 4.089
917 4.090
918 4.091
919 4.092
920 4.093
921 4.094
922 4.095
923 4.096
924 4.097
925 4.098
926 4.099
927 4.100
928 4.101
929 4.102
930 4.103
931 4.104
932 4.105
933 4.106
934 4.107
935 4.108
936 4.109
937 4.110
938 4.111
939 4.112
940 4.113
941 4.114
942 4.115
943 4.116
944 4.117
945 4.118
946 4.119
947 4.120
948 4.121
949 4.122
950 4.123
951 4.124
952 4.125
953 4.126
954 4.127
955 4.128
956 4.129
957 4.130
958 4.131
959 4.132
960 4.133
961 4.134
962 4.135
963 4.136
964 4.137
965 4.138
966 4.139
967 4.140
968 4.141
969 4.142
970 4.143
971 4.144
972 4.145
973 4.146
974 4.147
975 4.148
976 4.149
977 4.150
978 4.151
979 4.152
980 4.153
981 4.154
982 4.155
983 4.156
984 4.157
985 4.158
986 4.159
987 4.160
988 4.161
989 4.162
990 4.163
991 4.164
992 4.165
993 4.166
994 4.167
995 4.168
996 4.169
997 4.170
998 4.171
999 4.172
1000 4.173
1001 4.174
1002 4.175
1003 4.176
1004 4.177
1005 4.178
1006 4.179
1007 4.180
1008 4.181
1009 4.182
1010 4.183
1011 4.184
1012 4.185
1013 4.186
1014 4.187
1015 4.188
1016 4.189
1017 4.190
1018 4.191
1019 4.192
1020 4.193
1021 4.194
1022 4.195
1023 4.196
1024 4.197
1025 4.198
1026 4.199
1027 4.200
1028 4.201
1029 4.202
1030 4.203
1031 4.204
1032 4.205
1033 4.206
1034 4.207
1035 4.208
1036 4.209
1037 4.210
1038 4.211
1039 4.212
1040 4.213
1041 4.214
1042 4.215
1043 4.216
1044 4.217
1045 4.218
1046 4.219
1047 4.220
1048 4.221
1049 4.222
1050 4.223
1051 4.224
1052 4.225
1053 4.226
1054 4.227
1055 4.228
1056 4.229
1057 4.230
1058 4.231
1059 4.232
1060 4.233
1061 4.234
1062 4.235
1063 4.236
1064 4.237
1065 4.238
1066 4.239
1067 4.240
1068 4.241
1069 4.242
1070 4.243
1071 4.244
1072 4.245
1073 4.246
1074 4.247
1075 4.248
1076 4.249
1077 4.250
1078 4.251
1079 4.252
1080 4.253
1081 4.254
1082 4.255
1083 4.256
1084 4.257
1085 4.258
1086 4.259
1087 4.260
1088 4.261
1089 4.262
1090 4.263
1091 4.264
1092 4.265
1093 4.266
1094 4.267
1095 4.268
1096 4.269
1097 4.270
1098 4.271
1099 4.272
1100 4.273
1101 4.274
1102 4.275
1103 4.276
1104 4.277
1105 4.278
1106 4.279
1107 4.280
1108 4.281
1109 4.282
1110 4.283
1111 4.284
1112 4.285
1113 4.286
1114 4.287
1115 4.288
1116 4.289
1117 4.290
1118 4.291
1119 4.292
1120 4.293
1121 4.294
1122 4.295
1123 4.296
1124 4.297
1125 4.298
1126 4.299
1127 4.300
1128 4.301
1129 4.302
1130 4.303
1131 4.304
1132 4.305
1133 4.306
1134 4.307
1135 4.308
1136 4.309
1137 4.310
1138 4.311
1139 4.312
1140 4.313
1141 4.314
1142 4.315
1143 4.316
1144 4.317
1145 4.318
1146 4.319
1147 4.320
1148 4.321
1149 4.322
1150 4