**Newton-Raphson-Method applied in Fixed-Income-Analysis**

Marc Grau, 13-614-722

December 20, 2018

# 1 Introduction

The goal is to set up a quick process to analyze fixed income instruments. There are two ways to approach this analysis. On the one hand a user can let the program do the approximate pricing of the bond or on the other hand he can use an own input. The main difference between these two methods is that in the first one the program calculates the price of the bond via the remaining payments and in the second one the priced is retrieved from the market. Hence, the second option allows to incorporate different sources of risk by using a real market price. In every case reinvestment of the coupons at the current market conditions is assumed.

# 2 Outset

## Inputs

The following user inputs are needed:

* Calculations based on a given price
  + Bond price
  + Face value
  + Coupon
  + Time to maturity
* Price calculation
  + Coupon
  + Time to maturity

To have a proxy for the general forward rates the US treasuries bond yield curve is chosen. These rates needed for the calculations are retrieved from Quandl1.

## Process

## Newton-Raphson-Method

The Newton-Raphson-Method is an approximation method to successively find better values for the roots of real-valued function and is defined as follows:

The process is iterated until a sufficiently good approximation is achieved. In this case for f(x) the following formula is used, where x leads to the approximation for the yield to maturity.

## Bond Price

To calculate the price the bond is split into its unique cash flows.

## Bond Duration

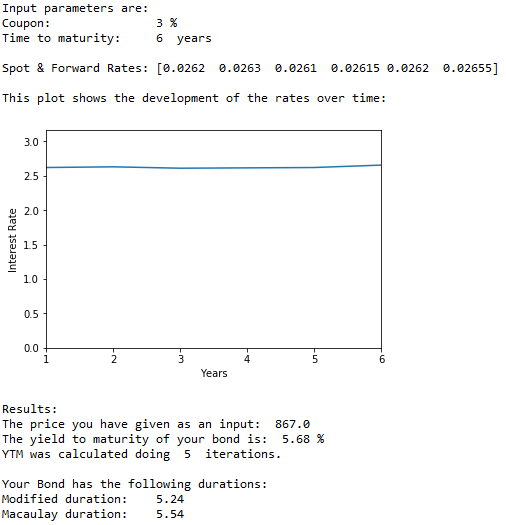
The Macaulay duration and the modified duration are part of the output and defined as follows:

## Output

As an output the user receives the following:

* Overview of the input parameters
* Forward rates
* Plot of forward rates
* Price
* Yield to maturity
* Modified duration
* Macaulay duration

An example output looks like this:



# 3 Code

1. # -\*- coding: utf-8 -\*-
2. """
4. @author: marcg
6. """
7. **import** sys
8. **import** pandas as pd
9. **import** quandl
10. **import** numpy as np
11. **import** matplotlib.pyplot as plt
13. # To get the data for the forward rates please enter your Quandl API Key
14. quandl.ApiConfig.api\_key = "ZzyxEEkVxNqQBKE9jXYd"
16. ##################
17. ### User Input ###
18. ##################
20. # strings defined as possible answers for the following questions
21. yes = {'YES', 'Yes', 'Y','yes','y', 'ye', ''}
22. no = {'NO', 'No', 'N', 'no','n'}
24. # In this stage the user is asked wether he wants to provide the price himself or let the program calculate it
25. # depending on the answer different mechanisms are triggered
26. # if he doesn't provide a suitable answer, he will be asked the same question again
27. **print**("What kind of analysis do you want to do? (y/n)")
28. **print**("yes - if you want to do calculations based on a given price")
29. **print**("no - if you want to calculate a fair price based on coupon and maturity (neglecting risks)")
30. choice = str(input().lower())
31. **if** choice **in** yes:
32. choice = True
33. **elif** choice **in** no:
34. choice = False
35. **else**:
36. sys.stdout.write("Please respond with 'yes' or 'no'")
38. # depending on the previous choice one flow to ask for inputs is triggered
39. # if the price is not known yet we only have to ask for the coupon and the time to maturity. Everything else will be calculated
40. # if the price and face value are given as input we store them and ask the same questions as in the other case
41. # if the answers aren't suitable to the program the user will be asked again or in case of an input of the wrong format he will be told to change
42. **if** choice == False:
43. coupon = -1
44. **while** 0 > coupon **or** 50 < coupon:
45. **try**:
46. coupon = float(input("Please enter the coupon of your bond (in percentage): "))
47. **except** ValueError:
48. **print**("Unfortunately, this program doesn't cover negative coupons or coupons above 50% as it appears unrealistic. Please enter another coupon.")
49. T = 0
50. **while** 1 >= T **or** 30 <= T:
51. **try**:
52. T = int(input("Please enter the remaining years until the bond matures: "))
53. **except** ValueError:
54. **print**("We're sorry. This program only uses yield curve estimates between 1 to 30 years. Please remain in this boundaries")
55. **else**:
56. input\_price = float(input("Please enter the price of your bond: "))
57. input\_fv = float(input("Please enter the face value of your bond: "))
58. coupon = -1
59. **while** 0 > coupon **or** 50 < coupon:
60. **try**:
61. coupon = float(input("Please enter the coupon of your bond (in percentage): "))
62. **except** ValueError:
63. **print**("Unfortunately, this program doesn't cover negative coupons or coupons above 50% as it appears unrealistic. Please enter another coupon.")
64. T = 0
65. **while** 1 >= T **or** 30 <= T:
66. **try**:
67. T = int(input("Please enter the remaining years until the bond matures: "))
68. **except** ValueError:
69. **print**("We're sorry. This program only uses yield curve estimates between 1 to 30 years. Please remain in this boundaries")
71. ###################
72. ### Needed Data ###
73. ###################

76. # in order to have an estimate for the forward rates we retrieve some data from Quandl
77. # This curve, which relates the yield on a security to its time to maturity is based on the closing
78. # market bid yields on actively traded Treasury securities in the over-the-counter market.
79. # These market yields are calculated from composites of quotations obtained by the Federal Reserve Bank of New York.
80. # The yield values are read from the yield curve at fixed maturities, currently 1, 2, 3, 5, 7, 10, 20, and 30 years.
81. # We always use the most recent estimations in this program
82. UStreasury = quandl.get("USTREASURY/YIELD", limit = 1)
84. UStreasury\_yields = UStreasury.iloc[:,4:12]
86. US\_1y = float(UStreasury\_yields.iloc[0,0])
87. US\_2y = float(UStreasury\_yields.iloc[0,1])
88. US\_3y = float(UStreasury\_yields.iloc[0,2])
89. US\_5y = float(UStreasury\_yields.iloc[0,3])
90. US\_7y = float(UStreasury\_yields.iloc[0,4])
91. US\_10y = float(UStreasury\_yields.iloc[0,5])
92. US\_20y = float(UStreasury\_yields.iloc[0,6])
93. US\_30y = float(UStreasury\_yields.iloc[0,7])
95. # Very simple extrapolation to have the forward rates of the missing years
96. # For more sophisticated analysis yield curve simulations (e.g. Vasicek-Model) could be implemented
97. # In this case we assume a linear relation between the known values of the yield curve
98. g\_7 = (US\_10y - US\_7y)/3
99. g\_10 = (US\_20y - US\_10y)/10
100. g\_20 = (US\_30y - US\_20y)/10
102. US\_4y = (US\_3y + US\_5y)/2
103. US\_6y = (US\_5y + US\_7y)/2
104. US\_8y = US\_7y + g\_7
105. US\_9y = US\_7y + 2\*g\_7
106. US\_11y = US\_10y + g\_10
107. US\_12y = US\_10y + 2\*g\_10
108. US\_13y = US\_10y + 3\*g\_10
109. US\_14y = US\_10y + 4\*g\_10
110. US\_15y = US\_10y + 5\*g\_10
111. US\_16y = US\_10y + 6\*g\_10
112. US\_17y = US\_10y + 7\*g\_10
113. US\_18y = US\_10y + 8\*g\_10
114. US\_19y = US\_10y + 9\*g\_10
115. US\_21y = US\_20y + g\_20
116. US\_22y = US\_20y + 2\*g\_20
117. US\_23y = US\_20y + 3\*g\_20
118. US\_24y = US\_20y + 4\*g\_20
119. US\_25y = US\_20y + 5\*g\_20
120. US\_26y = US\_20y + 6\*g\_20
121. US\_27y = US\_20y + 7\*g\_20
122. US\_28y = US\_20y + 8\*g\_20
123. US\_29y = US\_20y + 9\*g\_20
125. forward\_rates = np.array([US\_1y, US\_2y, US\_3y, US\_4y, US\_5y, US\_6y, US\_7y, US\_8y, US\_9y, US\_10y, US\_11y, US\_12y, US\_13y, US\_14y, US\_15y, US\_16y, US\_17y, US\_18y, US\_19y, US\_20y, US\_21y, US\_22y, US\_23y, US\_24y, US\_25y, US\_26y, US\_27y, US\_28y, US\_29y, US\_30y])
127. # depending on the years to maturity of the bond we're choosing the respective timeframe of the forward rates
128. fr = forward\_rates[0:T]
130. # to unify the numbers for the calculations we don't caluclate in percentage points
131. # the face value is standardized to 1
132. # a fixed value of 0.1 is set to start the calculations of the newton process
133. **if** choice == True:
134. fv = input\_fv
135. **else**:
136. fv = int(1)
138. coupon = coupon/100
139. fr = fr/100
140. m = np.arange(1,T+1)
141. cfs = [coupon\*fv]\*(T-1) + [fv+(coupon\*fv)]
142. initialGuess = 0.1
144. ###############################
145. ### Definition of functions ###
146. ###############################
148. ### All coupon payments are assumed to be reinvested at the possible returns at this time
150. # calculation of the bond price using the
151. **def** BondPrice(fr, cfs, m):
152. cfs\_disc = cfs/((1+fr)\*\*m)
153. bondprice = np.sum(cfs\_disc)
154. **return**(bondprice)
156. **if** choice == True:
157. bondprice = input\_price
158. **else**:
159. bondprice = BondPrice(fr,cfs,m)
161. # calculation of dollar duration, modified duration and macaulay duration
162. **def** BondDuration(bondprice, cfs, m, ytm):
163. dur\_cfs\_disc = (cfs\*m)/((1+ytm)\*\*(m+1))
164. dur = np.sum(dur\_cfs\_disc)
165. mod\_dur = dur/bondprice
166. mac\_dur = dur\*(1+ytm)/bondprice
167. durations = [dur, mod\_dur, mac\_dur]
168. **return**(durations)
170. # function to calculate ytm via newton process
171. **def** f(x):
172. y\_1 = -bondprice + (coupon\*fv)\*((1 - (1 + x)\*\*-T)/x) + fv\*(1 + x)\*\*-T
173. **return**(y\_1)
175. # first order derivative of f(x)
176. **def** f\_der(x):
177. y\_2 = (coupon\*fv)\*((x\*T\*((1 + x)\*\*(-T-1)) - (1 - ((1 + x)\*\*-T)))/(x\*\*2)) - fv\*T\*((1+x)\*\*(-T-1))
178. **return**(y\_2)
180. # the newton-raphson process is a method for finding successively better approximations to the roots (or zeroes) of a real-valued function
181. **def** Newton(x\_i):
182. x\_i = x\_i - f(x\_i)/f\_der(x\_i)
183. **return**(x\_i)
185. # using the defined formula from above we approximate the yield to maturity of the bond
186. **def** YieldToMaturity(initialGuess):
187. error = 0.000000001
188. x\_i = initialGuess
189. delta = 1
190. **global** count
191. count = 0
192. **while** delta > error:
193. x\_i\_next = Newton(x\_i)
194. count = count + 1
195. delta = abs(x\_i\_next - x\_i)
196. x\_i = x\_i\_next
197. **return**(x\_i)
199. # Specifiying the plot
200. xlimit = [1,T]
201. ylimit = [0, round((max(fr)\*100)+0.5, 2)]
203. plt.xlim(xlimit)
204. plt.ylim(ylimit)
205. plt.xlabel('Years')
206. plt.ylabel('Interest Rate')
207. plt.plot(m,fr\*100)
209. # applying the functions with the given input data we receive the following results
210. ytm = round(YieldToMaturity(initialGuess), 4)
211. durations = BondDuration(bondprice, cfs, m, ytm)
213. **print**('Input parameters are: ')
214. **print**('Coupon:              ', round(100\*coupon),'%')
215. **print**('Time to maturity:    ', T, ' years')
216. **print**('')
217. **print**('Spot & Forward Rates:', fr)
219. **print**('')
220. **print**('This plot shows the development of the rates over time:')
221. plt.show()
222. **print**('')
223. **print**('Results:')
224. **if** choice == True:
225. **print**('The price you have given as an input: ', bondprice)
226. **else**:
227. **print**('Given the parameters your bond should currently be priced the following: ', round(bondprice, 2))
228. **print**('The yield to maturity of your bond is: ', round(100\*ytm, 2), '%')
229. **print**('YTM was calculated doing ', count,' iterations.')
230. **print**('')
231. **print**('Your Bond has the following durations:')
232. **print**('Modified duration:   ', round(durations[1], 2))
233. **print**('Macaulay duration:   ', round(durations[2], 2))