Understanding and Applying Yield Curve Modelling and Interpolation Techniques

Rafael Nicolas Fermin Cota

Purpose

This report will discuss the application of two relative value models for government bonds, namely a Yield To Maturity Benchmarking Model and a Discount Factor Model. The report will briefly highlight the merits and limitations of each model when used in a real-world trading environment.

Conceptual Model: Modelling the Term Structure of Interest Rates

Before discussing the two aforementioned relative value models, we need to understand why a practitioner needs to model the term structure of interest rates.

The term structure of interest rates defines the relationship between interest rates and time. This relationship shows at what rate an investment grows from now to time, t, priced at current market conditions. It defines how borrowers are currently compensating lenders for loans over different time horizons. Graphically, a yield curve or term structure of interest rates specifies the relationship between the yield of bonds of the same credit quality and maturity. If an appropriate yield curve is calculated and specified, it can be used to determine how bonds are trading relative to the rest of the securities in that market. Therefore, the model can be used to identify buying and selling opportunities in the market. The underlying assumption in using a yield curve model to identify trading opportunities is that the market is not completely efficient in pricing bonds.

Unfortunately, a continuous term structure of interest rates is not observable in reality because market bond yields are discrete since they only exist for certain maturities. Therefore, a method of interpolation must be developed. There are three components of the interpolation method that define the yield curve model: a pricing function; an approximation function; and an estimation method.

Pricing Function

A bond's price is equal to the present value of its future cash flows. An important decision in determining a bond's price is what rate should be used to discount each cash flow.

The Yield To Maturity Benchmarking Model uses the simplified function where every cash flow from the bond is discounted using the yield to maturity. Employing this method assumes that each coupon is reinvested at the yield to maturity, therefore the investor is exposed reinvestment risk. The accuracy of the model when using this method is highest when coupon rates are uniform and market yields and coupon rates are low. See Formula 1.

The Discount Factor Model splits a bond into each of its individual cash flows and applies a unique discount factor to each cash flow. The model treats a coupon bond as a simple bundle of zero coupon bonds. This follows the logic that each point of time has a unique interest rate associated with it. See Formula 3.

Approximation Function and Estimation Method

A method of approximating the uncertain component for the pricing function chosen for each of the models must be determined. The Yield To Maturity Benchmarking Model requires a function that approximates a yield to maturity curve (yield to maturity against their respective terms to maturity). The Discount Factor Model requires a function that approximates the discount function.

The Yield To Maturity Benchmarking Model approximates a yield to maturity function using a polynomial regression fit using the least squares method.

Formula 1: *Semi-annual compounding convention

$$\mathbf{MP_j = C_{j1}/(1+YTM_t/2)^1 + C_{j2}/(1+YTM_t/2)^2 + \ldots + C_{ji}/(1+YTM_t/2)^i + P_j/(1+YTM_t/2)^i + \ni_j}$$

MP_j = Cash price of bond j
C_{ji} = ith cash flow for bond j
YTM_i = Yield to maturity of a bond with t years until maturity
i = Denotes the total number of periods
j = 1, 2, …, j; j = the number bonds in the model
$\ni_j$ = Error term for bond j

Formula 2:

$$\mathbf{YTM_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \ldots + \beta_k t^k}$$

Where: YTM_t = Yield to maturity of a bond with t years until maturity
k = The degree of the polynomial
t = Number of years from settlement to maturity date
β_k = The coefficients of the model which describe how the time to cash flow determines the discount factor

The Discount Factor Model does not model the unique interest rate at each point in time since interest rate term structures come in all kinds of shapes that make the formulation of a suitable model very demanding. Instead, the Discount Factor Model approximates the discount function. Modelling the discount function is easier than modelling yield to maturity because the function has a clear boundary at time zero and is monotonously declining over time. The model specifies that the discount factors are solely a factor of time until reception of the cash flow (Formula 4). Since the model deals with a set of bonds with homogeneous credit risk , this specification is appropriate. The discount function is approximated by a polynomial regression fit using the method of least squares. See Formula 4, Formula 5, and Formula 6 for the specification of the regression. The degree of the polynomial used for the regression is left up to the user, however, common practice is cubic.

The Discount Factor Model also allows users to set boundary conditions. Two common boundary conditions include force the discount factor at time equal to zero to one and set the first derivative at time equal to zero to fit a short-term rate (for example, the overnight bank rate observed in the market).

Formula 3:

$$\mathbf{MP_j = C_{j1}D_1 + C_{j2}D_2 + \ldots + C_{ji}D_i + \ni_j}$$

MP_j = Cash price of bond j
C_{ji} = ith cash flow for bond j
D_i = Discount factor for the ith cash flow
i = Denotes when the cash flow would

$j = 1, 2, \ldots, j$; j = the number bonds in the model

$\backepsilon_j$ = Error term for bond j

Formula 4:

$$\mathbf{D_i = \beta_0 + \beta_1 T_i + \beta_2 T_i^2 + \ldots + \beta_k T_i^k}$$

Where: D_i = Discount factor at time = i

k = The degree of the polynomial

T_i = Time until the ith cash flow, in years

β_k = The coefficients of the model which describe how the time to cash flow determines the discount factor

Formula 5: Substituting Formula 4 into Formula 3

$$\mathbf{MP_j = C_{j1} [\beta_0 + \beta_1 T_1 + \beta_2 T_1^2 + \ldots + \beta_k T_1^k] + \ldots + C_{ji} [\beta_0 + \beta_1 T_i + \beta_2 T_i^2 + \ldots + \beta_k T_i^k] + \backepsilon_j}$$

Formula 6: Rearranging Formula 5

$$\mathbf{MP_j = [\beta_0]\ [C_{j1} + \ldots + C_{ji}] + [\beta_1]\ [C_{j1}T_1 + \ldots + C_{ji}T_i] + \ldots + [\beta_k]\ [C_{j1}T_1^k + \ldots + C_{ji}T_i^k]}$$

Formula 7: The matrix form based on Formula 6

$$\overset{\text{Dependent Variable Matrix}}{\begin{bmatrix} \text{Market Price Bond 1} \\ \text{Market Price Bond 2} \\ \text{Market Price Bond 3} \\ \text{Market Price Bond 4} \\ \text{Market Price Bond 5} \\ \vdots \\ \text{Market Price Bond j} \end{bmatrix}} = \overset{\text{Regression Coefficients}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}} \overset{\text{Independent Variable Matrix}}{\begin{bmatrix} \Sigma c_{1i}T_{1i}^0 & \Sigma c_{1i}T_{1i}^1 & \Sigma c_{1i}T_{1i}^2 & \cdots & \Sigma c_{1i}T_{1i}^k \\ \Sigma c_{2i}T_{2i}^0 & \Sigma c_{2i}T_{2i}^1 & \Sigma c_{2i}T_{2i}^2 & \cdots & \Sigma c_{2i}T_{2i}^k \\ \Sigma c_{3i}T_{3i}^0 & \Sigma c_{3i}T_{3i}^1 & \Sigma c_{3i}T_{3i}^2 & \cdots & \Sigma c_{3i}T_{3i}^k \\ \Sigma c_{4i}T_{4i}^0 & \Sigma c_{4i}T_{4i}^1 & \Sigma c_{4i}T_{4i}^2 & \cdots & \Sigma c_{4i}T_{4i}^k \\ \Sigma c_{5i}T_{5i}^0 & \Sigma c_{5i}T_{5i}^1 & \Sigma c_{5i}T_{5i}^2 & \cdots & \Sigma c_{5i}T_{5i}^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma c_{ji}T_{ji}^0 & \Sigma c_{ji}T_{ji}^1 & \Sigma c_{ji}T_{ji}^2 & \cdots & \Sigma c_{ji}T_{ji}^k \end{bmatrix}}$$

Errors in the Model

Error terms in the regressions of both the Yield To Maturity Benchmarking Model and the Discount Factor Model may reflect temporary or permanent effects. Temporary effects reflect short-term deviations from the fair price of the bond and are therefore trading opportunities. Permanent effects may reflect factors impacting the price of the bond outside of time to maturity, such as liquidity premiums/discounts, tax effects, or trading special in the repo market. Many of these factors are likely to persist and therefore need to be taken into consideration when assessing the relative value of a bond. Permanent effects should be separated from temporary effects to use the model to identify trading opportunities

Applied Model: Yield To Maturity Benchmarking Model

This segment of the report will discuss applying the Yield to Maturity Benchmarking Model. The user-defined function GOVERNMENT_BONDS_TRADING_DESK_ FUNC is used to replicate the conceptual Yield to Maturity Benchmarking Model. All supporting functions are discussed in depth in the appendix. The building of the GOVERNMENT_BONDS_TRADING_DESK_ FUNC will be discussed in the following sections:

I. Inputs & Model Parameters

II. Components of the Model
   a. YTM_VECTOR
   b. YTC_VECTOR
   c. YTW_VECTOR
   d. MATURITY_VECTOR
   e. INTER_VECTOR
   f. INTERP_SPREADS_VECTOR
   g. PRICE_VECTOR

III. GOVERNMENT_BONDS_TRADING_DESK_FUNC Output

Section I: Inputs & Model Parameters

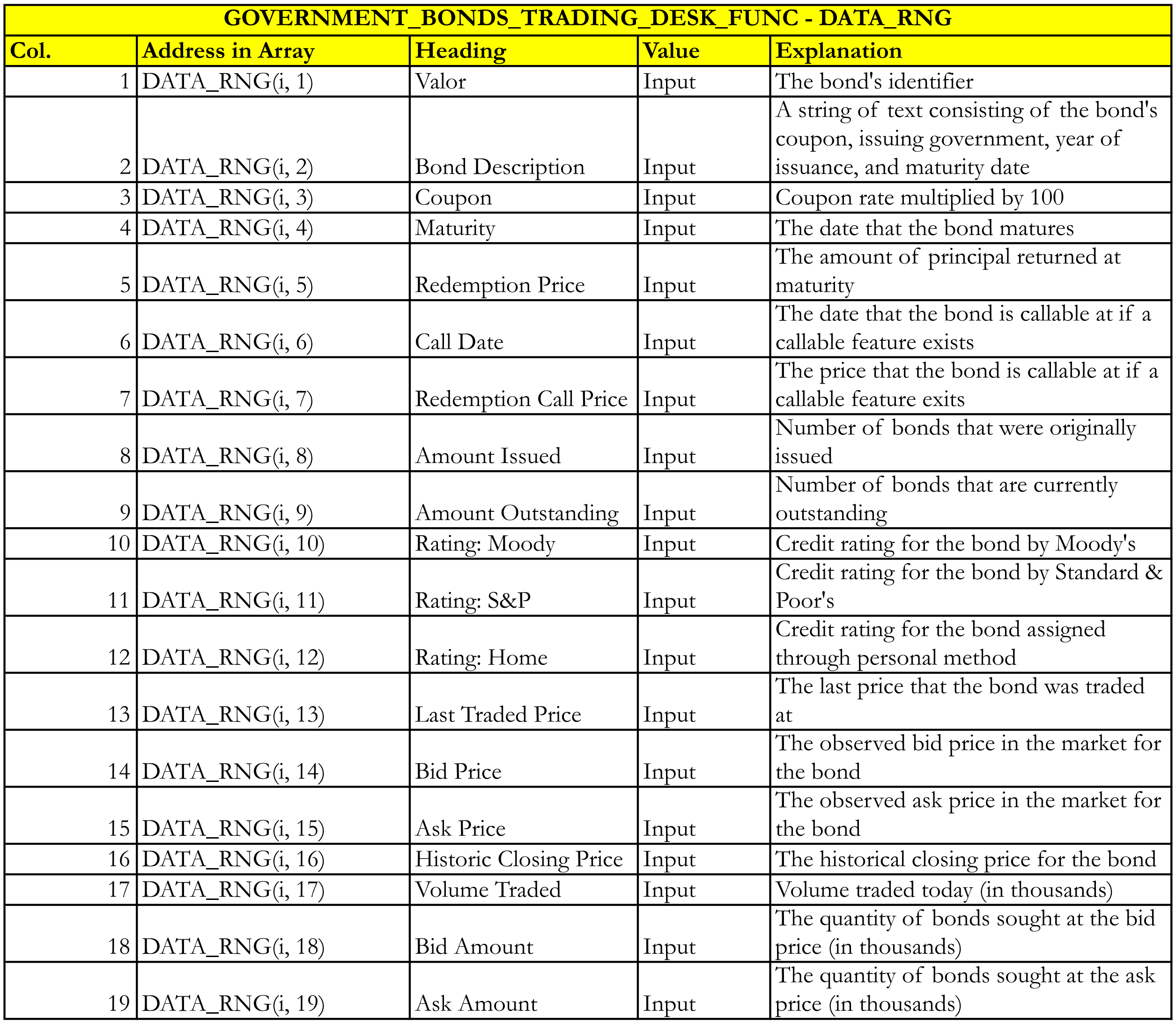
The Yield to Maturity Benchmarking Model requires data for a set of bonds and other parameters that will characterize the model. The model is a function of its inputs, therefore careful consideration must be paid to the choice of which bonds to include. Considerations include data integrity in addition to the impact of liquidity differences on prices. FIGURE 1 & FIGURE 2 detail the inputs required for the GOVERNMENT_BONDS_TRADING_DESK_FUNC. Going forward, DATA_MATRIX = DATA_RNG.

FIGURE 1: GOVERNMENT_BONDS_TRADING_DESK_FUNC Inputs

Function: GOVERNMENT_BONDS_TRADING_DESK_FUNC		
Inputs	**Format**	**Description**
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
DATA_RNG	Array	Described in detail in the following exhibit
SWAP_COEF_RNG	Array	An array of the coefficients that define the polynomial regression that fits the swap curve
ACTION_PRICE_SENSITIVITY	Number	Defines the cushion in pricing differences between the market and the model before the model will generate a buy or sell signal **Default = 0.4**
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention of bond **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**
GUESS_YIELD	Number	Starting value for the optimization engine when calculating the yield for each bond **Default: 30%**
OUTPUT	Integer {0, 1, 2, 3, 4, 5, 6, 7, 8}	The OUTPUT input determines the return value of the function: Case 0 - Generates the TEMP_MATRIX Case 1 - Generates the CHART_MATRIX Case 2 - Generates the YTM_VECTOR Case 3 - Generates the YTC_VECTOR Case 4 - Generates the YTW_VECTOR Case 5 - Generates the MATURITY_VECTOR Case 6 - Generates the INTER_VECTOR Case 7 - Generates the SPREAD_VECTOR Case 8 - Generates the PRICE_VECTOR **Default: 0**
REDEMPTION	Number	The redemption value of the bond at maturity **Default: 100**
FACTOR1_VAL	Number	Factor value that converts the coupon value into a percent **Default: 100**
FACTOR2_VAL	Number	Used to convert the bid amount and ask amount observed from thousands into millions **Default: 1000**
FACTOR3_VAL	Number	Used to convert spreads into basis points **Default: 10000**

**Italicized inputs are OPTIONAL*

FIGURE 2: Description of the DATA_RNG Input



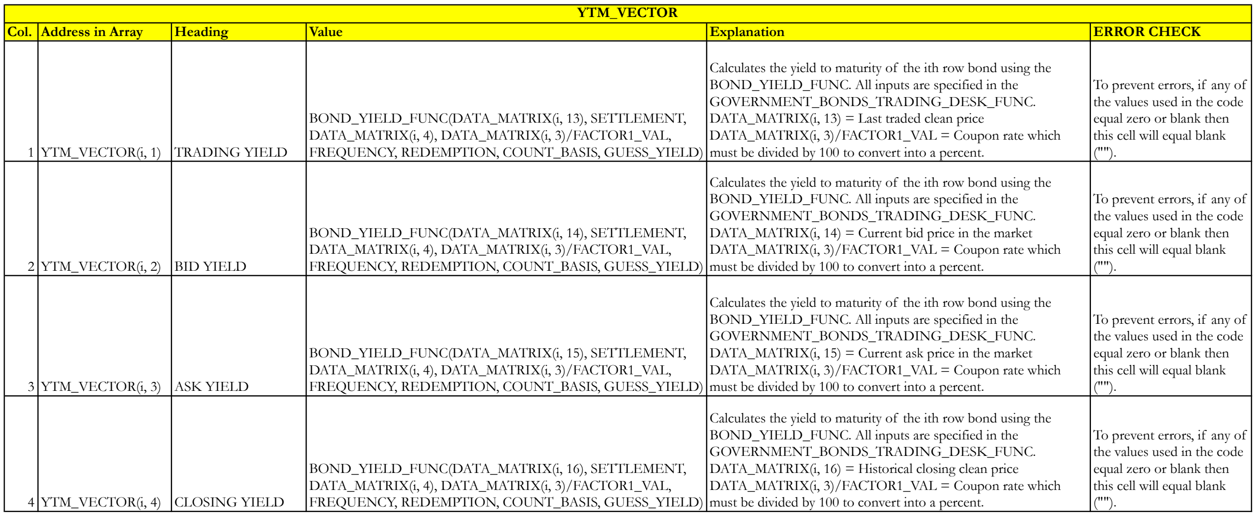
GOVERNMENT_BONDS_TRADING_DESK_FUNC - DATA_RNG				
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**
1	DATA_RNG(i, 1)	Valor	Input	The bond's identifier
2	DATA_RNG(i, 2)	Bond Description	Input	A string of text consisting of the bond's coupon, issuing government, year of issuance, and maturity date
3	DATA_RNG(i, 3)	Coupon	Input	Coupon rate multiplied by 100
4	DATA_RNG(i, 4)	Maturity	Input	The date that the bond matures
5	DATA_RNG(i, 5)	Redemption Price	Input	The amount of principal returned at maturity
6	DATA_RNG(i, 6)	Call Date	Input	The date that the bond is callable at if a callable feature exists
7	DATA_RNG(i, 7)	Redemption Call Price	Input	The price that the bond is callable at if a callable feature exits
8	DATA_RNG(i, 8)	Amount Issued	Input	Number of bonds that were originally issued
9	DATA_RNG(i, 9)	Amount Outstanding	Input	Number of bonds that are currently outstanding
10	DATA_RNG(i, 10)	Rating: Moody	Input	Credit rating for the bond by Moody's
11	DATA_RNG(i, 11)	Rating: S&P	Input	Credit rating for the bond by Standard & Poor's
12	DATA_RNG(i, 12)	Rating: Home	Input	Credit rating for the bond assigned through personal method
13	DATA_RNG(i, 13)	Last Traded Price	Input	The last price that the bond was traded at
14	DATA_RNG(i, 14)	Bid Price	Input	The observed bid price in the market for the bond
15	DATA_RNG(i, 15)	Ask Price	Input	The observed ask price in the market for the bond
16	DATA_RNG(i, 16)	Historic Closing Price	Input	The historical closing price for the bond
17	DATA_RNG(i, 17)	Volume Traded	Input	Volume traded today (in thousands)
18	DATA_RNG(i, 18)	Bid Amount	Input	The quantity of bonds sought at the bid price (in thousands)
19	DATA_RNG(i, 19)	Ask Amount	Input	The quantity of bonds sought at the ask price (in thousands)

Section II: Components of the Model

This segment of the report will discuss the various components necessary involved in building the model.

Part A – Building the YTM_VECTOR

The YTM_VECTOR stores the yield to maturity for all of the bonds in the data set based on the bond’s most recent trading price, bid price, ask price, and historic closing price.

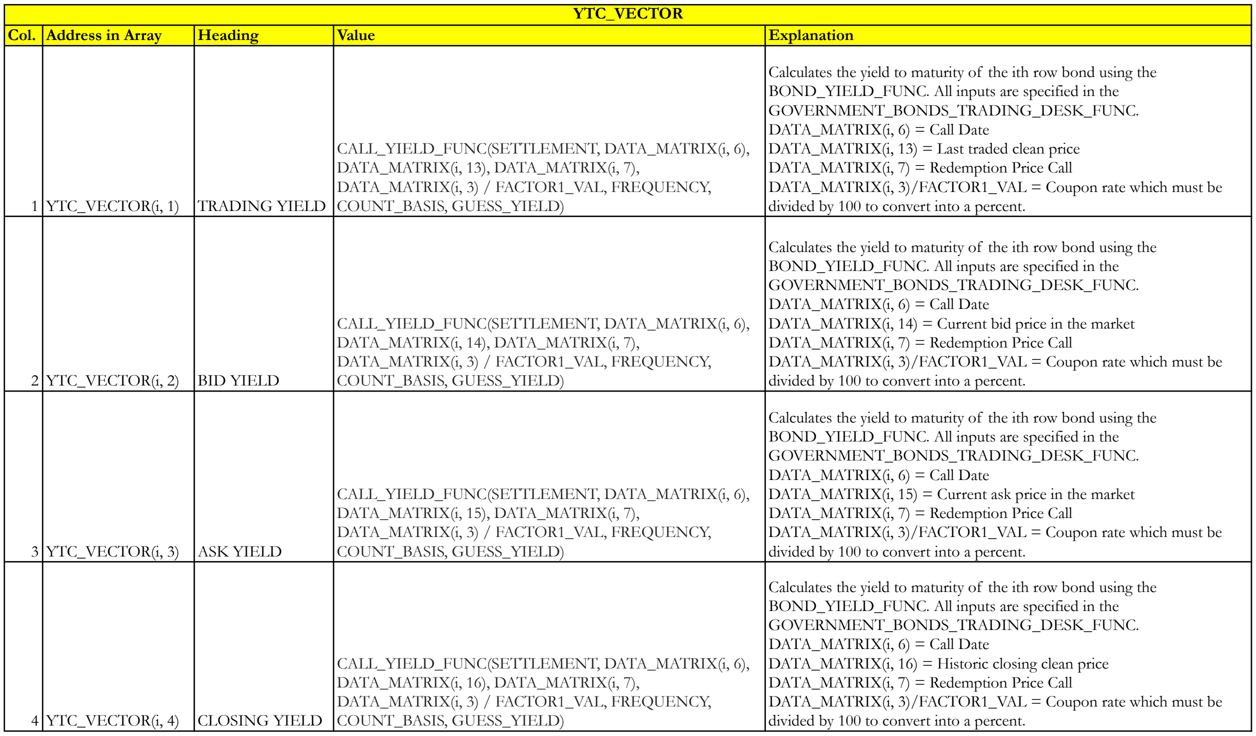


YTM_VECTOR

Col.	Address in Array	Heading	Value	Explanation	ERROR CHECK
1	YTM_VECTOR(i, 1)	TRADING YIELD	BOND_YIELD_FUNC(DATA_MATRIX(i, 13), SETTLEMENT, DATA_MATRIX(i, 4), DATA_MATRIX(i, 3)/FACTOR1_VAL, FREQUENCY, REDEMPTION, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 13) = Last traded clean price DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
2	YTM_VECTOR(i, 2)	BID YIELD	BOND_YIELD_FUNC(DATA_MATRIX(i, 14), SETTLEMENT, DATA_MATRIX(i, 4), DATA_MATRIX(i, 3)/FACTOR1_VAL, FREQUENCY, REDEMPTION, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 14) = Current bid price in the market DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
3	YTM_VECTOR(i, 3)	ASK YIELD	BOND_YIELD_FUNC(DATA_MATRIX(i, 15), SETTLEMENT, DATA_MATRIX(i, 4), DATA_MATRIX(i, 3)/FACTOR1_VAL, FREQUENCY, REDEMPTION, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 15) = Current ask price in the market DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
4	YTM_VECTOR(i, 4)	CLOSING YIELD	BOND_YIELD_FUNC(DATA_MATRIX(i, 16), SETTLEMENT, DATA_MATRIX(i, 4), DATA_MATRIX(i, 3)/FACTOR1_VAL, FREQUENCY, REDEMPTION, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 16) = Historical closing clean price DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").

Part B – Building the YTC_VECTOR

The YTC_VECTOR stores the yield to call for all of the bonds in the data set based on the bond's most recent trading price, bid price, ask price, and historic closing price. If a call date or call price does not exist, the array will have a blank ("") entry.



YTC_VECTOR

Col.	Address in Array	Heading	Value	Explanation
1	YTC_VECTOR(i, 1)	TRADING YIELD	CALL_YIELD_FUNC(SETTLEMENT, DATA_MATRIX(i, 6), DATA_MATRIX(i, 13), DATA_MATRIX(i, 7), DATA_MATRIX(i, 3) / FACTOR1_VAL, FREQUENCY, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 6) = Call Date DATA_MATRIX(i, 13) = Last traded clean price DATA_MATRIX(i, 7) = Redemption Price Call DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.
2	YTC_VECTOR(i, 2)	BID YIELD	CALL_YIELD_FUNC(SETTLEMENT, DATA_MATRIX(i, 6), DATA_MATRIX(i, 14), DATA_MATRIX(i, 7), DATA_MATRIX(i, 3) / FACTOR1_VAL, FREQUENCY, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 6) = Call Date DATA_MATRIX(i, 14) = Current bid price in the market DATA_MATRIX(i, 7) = Redemption Price Call DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.
3	YTC_VECTOR(i, 3)	ASK YIELD	CALL_YIELD_FUNC(SETTLEMENT, DATA_MATRIX(i, 6), DATA_MATRIX(i, 15), DATA_MATRIX(i, 7), DATA_MATRIX(i, 3) / FACTOR1_VAL, FREQUENCY, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 6) = Call Date DATA_MATRIX(i, 15) = Current ask price in the market DATA_MATRIX(i, 7) = Redemption Price Call DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.
4	YTC_VECTOR(i, 4)	CLOSING YIELD	CALL_YIELD_FUNC(SETTLEMENT, DATA_MATRIX(i, 6), DATA_MATRIX(i, 16), DATA_MATRIX(i, 7), DATA_MATRIX(i, 3) / FACTOR1_VAL, FREQUENCY, COUNT_BASIS, GUESS_YIELD)	Calculates the yield to maturity of the ith row bond using the BOND_YIELD_FUNC. All inputs are specified in the GOVERNMENT_BONDS_TRADING_DESK_FUNC. DATA_MATRIX(i, 6) = Call Date DATA_MATRIX(i, 16) = Historic closing clean price DATA_MATRIX(i, 7) = Redemption Price Call DATA_MATRIX(i, 3)/FACTOR1_VAL = Coupon rate which must be divided by 100 to convert into a percent.

Part C – Building the YTW_VECTOR

The YTW_VECTOR stores the yield to worst for all of the bonds in the data set based on the bond's most recent trading price, bid price, ask price, and historic closing price.

YTW_VECTOR

Col.	Address in Array	Heading	Value	Explanation

1	YTW_VECTOR(i, 1)	TRADING YIELD	If (YTC_VECTOR(i, 1) = 0) Or (YTC_VECTOR(i, 1) = "") Then
			      YTW_VECTOR(i, 1) = YTM_VECTOR(i, 1)
			   Else
			      If YTC_VECTOR(i, 1) < YTM_VECTOR(i, 1) Then
			         YTW_VECTOR(i, 1) = YTC_VECTOR(i, 1)
			      Else
			         YTW_VECTOR(i, 1) = YTM_VECTOR(i, 1)
			      End If
			   End If
				This cell equals the yield to worst for the ith row bond based on the last traded price. The yield to worst is calculated using the following logic:
				If yield to call based on the last traded price is zero or blank then the yield to worst based on the last traded price (YTW_VECTOR(i, 1)) equals the yield to maturity based on the last traded price . If the yield to call is less than the yield to maturity then the yield to worst equals the yield to call. Otherwise, the yield to worst equals the yield to maturity.

2	YTW_VECTOR(i, 2)	""	If (YTW_VECTOR(i, 1) = 0) Or (YTW_VECTOR(i, 1) = "") Then
			      YTW_VECTOR(i, 2) = ""
			   Else
			      If YTW_VECTOR(i, 1) = YTM_VECTOR(i, 1) Then
			         YTW_VECTOR(i, 2) = "M"
			      ElseIf YTW_VECTOR(i, 1) = YTC_VECTOR(i, 1) Then
			         YTW_VECTOR(i, 2) = "C"
			      End If
			   End If
				This cell is an identifier denoting whether the yield to maturity ("M") or yield to call ("C") was chosen as the yield to worst in YTW_VECTOR(i, 1). If the yield to worst based on the last traded price is zero or blank then the cell should equal blank or zero. Otherwise, the cell = "M" if yield to worst = yield to maturity or the cell = "C" if yield to worst = yield to call.

3	YTW_VECTOR(i, 3)	BID YIELD	If (YTC_VECTOR(i, 2) = 0) Or (YTC_VECTOR(i, 2) = "") Then
			      YTW_VECTOR(i, 3) = YTM_VECTOR(i, 2)
			   Else
			      If YTC_VECTOR(i, 2) < YTM_VECTOR(i, 2) Then
			         YTW_VECTOR(i, 3) = YTC_VECTOR(i, 2)
			      Else
			         YTW_VECTOR(i, 3) = YTM_VECTOR(i, 2)
			      End If
			   End If
				This cell equals the yield to worst for the ith row bond based on the bid price. The yield to worst is calculated using the following logic:
				If yield to call based on the bid price is zero or blank then the yield to worst based on the bid price (YTW_VECTOR(i, 1)) equals the yield to maturity based on the bid price . If the yield to call is less than the yield to maturity then the yield to worst equals the yield to call. Otherwise, the yield to worst equals the yield to maturity.

4	YTW_VECTOR(i, 4)	""	If (YTW_VECTOR(i, 3) = 0) Or (YTW_VECTOR(i, 3) = "") Then
			      YTW_VECTOR(i, 4) = ""
			   Else
			      If YTW_VECTOR(i, 3) = YTM_VECTOR(i, 2) Then
			         YTW_VECTOR(i, 4) = "M"
			      ElseIf YTW_VECTOR(i, 3) = YTC_VECTOR(i, 2) Then
			         YTW_VECTOR(i, 4) = "C"
			      End If
			   End If
				This cell is an identifier denoting whether the yield to maturity ("M") or yield to call ("C") was chosen as the yield to worst in YTW_VECTOR(i, 3). If the yield to worst based on the last traded price is zero or blank then the cell should equal blank or zero. Otherwise, the cell = "M" if yield to worst = yield to maturity or the cell = "C" if yield to worst = yield to call.

5	YTW_VECTOR(i, 5)	ASK YIELD	If (YTC_VECTOR(i, 3) = 0) Or (YTC_VECTOR(i, 3) = "") Then
			      YTW_VECTOR(i, 5) = YTM_VECTOR(i, 3)
			   Else
			      If YTC_VECTOR(i, 3) < YTM_VECTOR(i, 3) Then
			         YTW_VECTOR(i, 5) = YTC_VECTOR(i, 3)
			      Else
			         YTW_VECTOR(i, 5) = YTM_VECTOR(i, 3)
			      End If
			   End If
				This cell equals the yield to worst for the ith row bond based on the ask price. The yield to worst is calculated using the following logic:
				If yield to call based on the ask price is zero or blank then the yield to worst based on the ask price (YTW_VECTOR(i, 1)) equals the yield to maturity based on the ask price . If the yield to call is less than the yield to maturity then the yield to worst equals the yield to call. Otherwise, the yield to worst equals the yield to maturity.

6	YTW_VECTOR(i, 6)	""	If (YTW_VECTOR(i, 5) = 0) Or (YTW_VECTOR(i, 5) = "") Then
			         YTW_VECTOR(i, 6) = ""
			   Else
			      If YTW_VECTOR(i, 5) = YTM_VECTOR(i, 3) Then
			         YTW_VECTOR(i, 6) = "M"
			      ElseIf YTW_VECTOR(i, 5) = YTC_VECTOR(i, 3) Then
			         YTW_VECTOR(i, 6) = "C"
			      End If
			   End If
				This cell is an identifier denoting whether the yield to maturity ("M") or yield to call ("C") was chosen as the yield to worst in YTW_VECTOR(i, 5). If the yield to worst based on the last traded price is zero or blank then the cell should equal blank or zero. Otherwise, the cell = "M" if yield to worst = yield to maturity or the cell = "C" if yield to worst = yield to call.

7	YTW_VECTOR(i, 7)	CLOSING YIELD	If (YTC_VECTOR(i, 4) = 0) Or (YTC_VECTOR(i, 4) = "") Then
			      YTW_VECTOR(i, 7) = YTM_VECTOR(i, 4)
			   Else
			      If YTC_VECTOR(i, 4) < YTM_VECTOR(i, 4) Then
			         YTW_VECTOR(i, 7) = YTC_VECTOR(i, 4)
			      Else
			         YTW_VECTOR(i, 7) = YTM_VECTOR(i, 4)
			      End If
			   End If
				This cell equals the yield to worst for the ith row bond based on the ask price. The yield to worst is calculated using the following logic:
				If yield to call based on the historic closing price is zero or blank then the yield to worst based on the historic closing price (YTW_VECTOR(i, 1)) equals the yield to maturity based on the historic closing price . If the yield to call is less than the yield to maturity then the yield to worst equals the yield to call. Otherwise, the yield to worst equals the yield to maturity.

8	YTW_VECTOR(i, 8)	""	If (YTW_VECTOR(i, 7) = 0) Or (YTW_VECTOR(i, 7) = "") Then
			      YTW_VECTOR(i, 8) = ""
			   Else
			      If YTW_VECTOR(i, 7) = YTM_VECTOR(i, 4) Then
			         YTW_VECTOR(i, 8) = "M"
			      ElseIf YTW_VECTOR(i, 7) = YTC_VECTOR(i, 4) Then
			         YTW_VECTOR(i, 8) = "C"
			      End If
			   End If
				This cell is an identifier denoting whether the yield to maturity ("M") or yield to call ("C") was chosen as the yield to worst in YTW_VECTOR(i, 7). If the yield to worst based on the last traded price is zero or blank then the cell should equal blank or zero. Otherwise, the cell = "M" if yield to worst = yield to maturity or the cell = "C" if yield to worst = yield to call.

9	YTW_VECTOR(i, 9)	SELECTED YIELD	If ((YTW_VECTOR(i, 3) = 0) Or (YTW_VECTOR(i, 3) = "") Or (YTW_VECTOR(i, 5) = 0) Or (YTW_VECTOR(i, 5) = "")) Then
			      If (YTW_VECTOR(i, 1) = "") Or (YTW_VECTOR(i, 1) = 0) Then
			         YTW_VECTOR(i, 9) = YTW_VECTOR(i, 7)
			         YTW_VECTOR(i, 10) = YTW_VECTOR(i, 8)
			         YTW_VECTOR(i, 11) = "Hist"
			      Else
			         YTW_VECTOR(i, 9) = YTW_VECTOR(i, 1)
			         YTW_VECTOR(i, 10) = YTW_VECTOR(i, 2)
			         YTW_VECTOR(i, 11) = "Tr"
			      End If
			   Else
			      YTW_VECTOR(i, 9) = (YTW_VECTOR(i, 3) + YTW_VECTOR(i, 5)) / 2
			      If YTW_VECTOR(i, 4) = YTW_VECTOR(i, 6) Then
			         YTW_VECTOR(i, 10) = YTW_VECTOR(i, 4)
			      Else
			         YTW_VECTOR(i, 10) = YTW_VECTOR(i, 4) & "/" & YTW_VECTOR(i, 6)
			      End If
			      YTW_VECTOR(i, 11) = "Mid"
			   End If
				This cell equals the selected yield to worst for the model which is a proxy the mid yield to worst of a bond. If both a bid yield to worst and ask yield to worst are available for the bond, then this cell equals the average of the bid and ask yields. If either the bid yield or ask yield equal zero or blank then the model chooses the yield to worst based on the last traded price. If either the bid yield or ask yield as well as the last traded price yield equal zero then the cell equals the yield to worst based on the historic closing

10	YTW_VECTOR(i, 10)	""		This cell identifies if the yield to worst in YTW_VECTOR(i, 9) is a yield to call, a yield to maturtity, or a combination of both. If YTW_VECTOR(i, 9) was calculated using yield to call, this cell equals "C". If YTW_VECTOR(i, 9) was calculated using yield to maturity, this cell equals "M". If YTW_VECTOR(i, 9) was calculated using a combination of yield to call and yield to maturity as could be the case when using the average of bid and ask yields, this cell equals "C / M" or "M / C"
11	YTW_VECTOR(i, 11)	""		This cell identifies which yield to worst was chosen in YTW_VECTOR(i, 9). If the average of the bid and ask yields was chosen, this cell equals "Mid". If the yield to worst based on the last traded price was used, this cell equals "Tr". If the yield to worst was based on the historic closing price was used, this cell equals "Hist".

12	YTW_VECTOR(i, 12)	SELECTED BID	If (YTW_VECTOR(i, 3) = 0) Or (YTW_VECTOR(i, 3) = "") Then
			      YTW_VECTOR(i, 12) = YTW_VECTOR(i, 7)
			      YTW_VECTOR(i, 13) = YTW_VECTOR(i, 8)
			      YTW_VECTOR(i, 14) = "Hist"
			   Else
			      YTW_VECTOR(i, 12) = YTW_VECTOR(i, 3)
			      YTW_VECTOR(i, 13) = YTW_VECTOR(i, 4)
			      YTW_VECTOR(i, 14) = "Bid"
			   End If
				This cell equals the selected bid yield to worst for the model based. If the bid yield to worst, YTW_VECTOR(i , 5), is blank or zero, this cell equals the yield to worst based on the historic closing price. Otherwise, this cell equals the yield to worst based on the bid price.

13	YTW_VECTOR(i, 13)	""		This cell identifies whether the selected bid yield to worst used in YTW_VECTOR(i , 12) was based on yield to maturity ("M") or a yield to call ("C") calculation.
14	YTW_VECTOR(i, 14)	""		This cell identifies which yield to worst was chosen in YTW_VECTOR(i, 12). If the yield to worst based on the bid price was used, this cell equals "Bid". If the yield to worst was based on the historic closing price was used, this cell equals "Hist".

15	YTW_VECTOR(i, 15)	SELECTED ASK	If (YTW_VECTOR(i, 5) = 0) Or (YTW_VECTOR(i, 5) = "") Then
			      YTW_VECTOR(i, 15) = YTW_VECTOR(i, 7)
			      YTW_VECTOR(i, 16) = YTW_VECTOR(i, 8)
			      YTW_VECTOR(i, 17) = "Hist"
			   Else
			      YTW_VECTOR(i, 15) = YTW_VECTOR(i, 5)
			      YTW_VECTOR(i, 16) = YTW_VECTOR(i, 6)
			      YTW_VECTOR(i, 17) = "Ask"
			   End If
				This cell equals the selected ask yield to worst for the model based. If the ask yield to worst, YTW_VECTOR(i , 5), is blank or zero, this cell equals the yield to worst based on the historic closing price. Otherwise, this cell equals the yield to worst based on the ask price.

16	YTW_VECTOR(i, 16)	""		This cell identifies whether the selected ask yield to worst used in YTW_VECTOR(i , 15) was based on yield to maturity ("M") or a yield to call ("C") calculation.
17	YTW_VECTOR(i, 17)	""		This cell identifies which yield to worst was chosen in YTW_VECTOR(i, 15). If the yield to worst based on the bid price was used, this cell equals "Ask". If the yield to worst was based on the historic closing price was used, this cell equals "Hist".

Part D – Building the MATURITY_VECTOR

The MATURITY_VECTOR stores the tenor, time in years to maturity or to call date, for all bonds in the data set. The BENCHMARK vectors clean the MATURITY_VECTOR and YTW_VECTOR by removing any empty or incomplete data and stores the information for the regression. The BENCHMARK coefficient vectors generate and store the coefficients for the polynomial regression. The model builds three regressions using the SELECTED YIELD column, which is a proxy for the bond's mid yield, BID YIELD column, which is based on the market observed bid price, and the ASK YIELD column, which is based on the market observed ask price.

MATURITY_VECTOR				
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**
1	MATURITY_VECTOR(i, 1)	PAID	If YTW_VECTOR(i, 10) = "M" Then If (DATA_MATRIX(i, c(4)) = 0) Or (DATA_MATRIX(i, c(4)) = "") Then MATURITY_VECTOR(i, 1) = "" Else MATURITY_VECTOR(i, 1) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(4)), COUNT_BASIS) End If ElseIf YTW_VECTOR(i, 10) = "C" Then If (DATA_MATRIX(i, c(6)) = 0) Or (DATA_MATRIX(i, c(6)) = "") Then MATURITY_VECTOR(i, 1) = "" Else MATURITY_VECTOR(i, 1) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(6)), COUNT_BASIS) End If Else MATURITY_VECTOR(i, 1) = "" End If If IsError(MATURITY_VECTOR(i, 1)) Then: MATURITY_VECTOR(i, 1) = ""	This cell stores the amount of time until maturity or call (i.e. tenor) for the ith bond for the selected yield YTW_VECTOR(i, 9). If YTW_VECTOR(i, 9) is a yield to maturity then this cell equals the amount of time in years until the maturity date from the settlement date using the YEARFRAC_FUNC. If YTW_VECTOR(i, 9) is a yield to call then this cell equals the amount of time in years until the call date from the settlement date using the YEARFRAC_FUNC. If no maturity date or call date exists in the data, as signified by a blank or zero, or the function returns an error the cell will equal blank.
2	MATURITY_VECTOR(i, 2)	BID SELECT	If YTW_VECTOR(i, 13) = "M" Then If (DATA_MATRIX(i, c(4)) = 0) Or (DATA_MATRIX(i, c(4)) = "") Then MATURITY_VECTOR(i, 2) = "" Else MATURITY_VECTOR(i, 2) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(4)), COUNT_BASIS) End If ElseIf YTW_VECTOR(i, 13) = "C" Then If (DATA_MATRIX(i, c(6)) = 0) Or (DATA_MATRIX(i, c(6)) = "") Then MATURITY_VECTOR(i, 2) = "" Else MATURITY_VECTOR(i, 2) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(6)), COUNT_BASIS) End If Else MATURITY_VECTOR(i, 2) = "" End If If IsError(MATURITY_VECTOR(i, 2)) Then: MATURITY_VECTOR(i, 2) = ""	This cell stores the amount of time in years until maturity or call (i.e. tenor) for the ith bond for the selected bid yield YTW_VECTOR(i, 12). If YTW_VECTOR(i, 12) is a yield to maturity then this cell equals the amount of time in years until the maturity date from the settlement date using the YEARFRAC_FUNC. If YTW_VECTOR(i, 12) is a yield to call then this cell equals the amount of time in years until the call date from the settlement date using the YEARFRAC_FUNC. If no maturity date or call date exists in the data, as signified by a blank or zero, or the function returns an error the cell will equal blank.
3	MATURITY_VECTOR(i, 3)	ASK SELECT	If YTW_VECTOR(i, 16) = "M" Then If (DATA_MATRIX(i, c(4)) = 0) Or (DATA_MATRIX(i, c(4)) = "") Then MATURITY_VECTOR(i, 3) = "" Else MATURITY_VECTOR(i, 3) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(4)), COUNT_BASIS) End If ElseIf YTW_VECTOR(i, 16) = "C" Then If (DATA_MATRIX(i, c(6)) = 0) Or (DATA_MATRIX(i, c(6)) = "") Then MATURITY_VECTOR(i, 3) = "" Else MATURITY_VECTOR(i, 3) = YEARFRAC_FUNC(SETTLEMENT, DATA_MATRIX(i, c(6)), COUNT_BASIS) End If Else MATURITY_VECTOR(i, 3) = "" End If If IsError(MATURITY_VECTOR(i, 3)) Then: MATURITY_VECTOR(i, 3) = ""	This cell stores the amount of time in years until maturity or call (i.e. tenor) for the ith bond for the selected ask yield YTW_VECTOR(i, 15). If YTW_VECTOR(i, 15) is a yield to maturity then this cell equals the amount of time in years until the maturity date from the settlement date using the YEARFRAC_FUNC. If YTW_VECTOR(i, 15) is a yield to call then this cell equals the amount of time in years until the call date from the settlement date using the YEARFRAC_FUNC. If no maturity date or call date exists in the data, as signified by a blank or zero, or the function returns an error the cell will equal blank.

BENCHMARK VECTORS		
Vector Name	**Value**	**Explanation**
BENCH_PAID_TENOR_VECTOR	For i = 1 To NROWS If (MATURITY_VECTOR(i, 1) <> 0) And (MATURITY_VECTOR(i, 1) <> "") And (YTW_VECTOR(i, 9) <> 0) And (YTW_VECTOR(i, 9) <> "") Then BENCH_PAID_TENOR_VECTOR(i, 1) = MATURITY_VECTOR(i, 1) BENCH_PAID_RATE_VECTOR(i, 1) = YTW_VECTOR(i, 9) Else BENCH_PAID_TENOR_VECTOR(i, 1) = "" BENCH_PAID_RATE_VECTOR(i, 1) = "" End If Next i BENCH_PAID_TENOR_VECTOR = VECTOR_TRIM_FUNC(BENCH_PAID_TENOR_VECTOR, "") BENCH_PAID_RATE_VECTOR =	The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected yield to worst and associated tenor are not equal to zero or blank is the tenor value included in the BENCH_PAID_TENOR_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_PAID_TENOR_VECTOR.
BENCH_PAID_RATE_VECTOR		The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected yield to worst and the associated tenor are not equal to zero or blank is the yield included in the BENCH_PAID_RATE_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_PAID_RATE_VECTOR.
BENCH_BID_TENOR_VECTOR	For i to NROWS If (MATURITY_VECTOR(i, 2) <> 0) And (MATURITY_VECTOR(i, 2) <> "") And (YTW_VECTOR(i, 12) <> 0) And (YTW_VECTOR(i, 12) <> "") Then BENCH_BID_TENOR_VECTOR(i, 1) = MATURITY_VECTOR(i, 2) BENCH_BID_RATE_VECTOR(i, 1) = YTW_VECTOR(i, 12) Else BENCH_BID_TENOR_VECTOR(i, 1) = "" BENCH_BID_RATE_VECTOR(i, 1) = "" End If Next i BENCH_BID_TENOR_VECTOR = VECTOR_TRIM_FUNC(BENCH_BID_TENOR_VECTOR, "") BENCH_BID_RATE_VECTOR =	The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected bid yield to worst and associated tenor are not equal to zero or blank is the tenor value included in the BENCH_BID_TENOR_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_BID_TENOR_VECTOR.
BENCH_BID_RATE_VECTOR		The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected bid yield to worst and associated tenor are not equal to zero or blank is the yield included in the BENCH_BID_RATE_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_BID_RATE_VECTOR.
BENCH_ASK_TENOR_VECTOR	If (MATURITY_VECTOR(i, 3) <> 0) And (MATURITY_VECTOR(i, 3) <> "") And (YTW_VECTOR(i, 15) <> 0) And (YTW_VECTOR(i, 15) <> "") Then BENCH_ASK_TENOR_VECTOR(i, 1) = MATURITY_VECTOR(i, 3) BENCH_ASK_RATE_VECTOR(i, 1) = YTW_VECTOR(i, 15) Else BENCH_ASK_TENOR_VECTOR(i, 1) = "" BENCH_ASK_RATE_VECTOR(i, 1) = "" End If Next i BENCH_ASK_TENOR_VECTOR = VECTOR_TRIM_FUNC(BENCH_ASK_TENOR_VECTOR, "") BENCH_ASK_RATE_VECTOR = VECTOR_TRIM_FUNC(BENCH_ASK_RATE_VECTOR, "")	The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected ask yield to worst and associated tenor are not equal to zero or blank is the tenor value included in the BENCH_ASK_TENOR_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_ASK_TENOR_VECTOR.
BENCH_ASK_RATE_VECTOR		The benchmark vectors are used to eliminate any blanks, zeros, or incomplete data from the data set before running the regression. Only in situations where both the selected yield to worst and associated tenor are not equal to zero or blank is the yield included in the BENCH_ASK_RATE_VECTOR. Otherwise, a blank ("") is stored in its place in the vector. Once all of the empty or incomplete data has been idenitified as "" (blank), the VECTOR_TRIM_FUNC removes all the blanks from the BENCH_ASK_RATE_VECTOR.
BENCH_PAID_COEF_VECTOR	BENCH_PAID_COEF_VECTOR = POLYNOMIAL_REGRESSION_FUNC(BENCH_PAID_TENOR_VECTOR, BENCH_PAID_RATE_VECTOR, NDEG, 0)	A vector of the coefficients for the polynomial function that approximates the relationship between selected yield to maturity and time to maturity. The coefficients are calculated using the POLYNOMIAL_REGRESSION_FUNC. NDEG is the number of degrees of the polynomial and is specified as a input in the GOVERNMENT_BONDS_TRADING_DESK_FUNC.
BENCH_BID_COEF_VECTOR	BENCH_BID_COEF_VECTOR = POLYNOMIAL_REGRESSION_FUNC(BENCH_BID_TENOR_VECTOR, BENCH_BID_RATE_VECTOR, NDEG, 0)	A vector of the coefficients for the polynomial function that approximates the relationship between selected bid yield to maturity and time to maturity. The coefficients are calculated using the POLYNOMIAL_REGRESSION_FUNC. NDEG is the number of degrees of the polynomial and is specified as a input in the GOVERNMENT_BONDS_TRADING_DESK_FUNC.
BENCH_ASK_COEF_VECTOR	BENCH_ASK_COEF_VECTOR = POLYNOMIAL_REGRESSION_FUNC(BENCH_ASK_TENOR_VECTOR, BENCH_ASK_RATE_VECTOR, NDEG, 0)	A vector of the coefficients for the polynomial function that approximates the relationship between selected ask yield to maturity and time to maturity. The coefficients are calculated using the POLYNOMIAL_REGRESSION_FUNC. NDEG is the number of degrees of the polynomial and is specified as a input in the GOVERNMENT_BONDS_TRADING_DESK_FUNC.

Part E – Building the INTER_VECTOR

The INTER_VECTOR stores the interpolated bond yields generated from the model specified yield to maturity functions. Interpolated bond yields are calculated for the three unique functions as well as the swap function specified as an input.

INTER_VECTOR				
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**
1	INTER_VECTOR(i, 1)	PAID	If (MATURITY_VECTOR(i, 1) = 0) Or (MATURITY_VECTOR(i, 1) = "") Then INTER_VECTOR(i, 1) = "" Else INTER_VECTOR(i, 1) = DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC(BENCH_PAID_COEF_VECTOR, MATURITY_VECTOR(i, 1))(1, 1) If IsError(INTER_VECTOR(i, 1)) Then: INTER_VECTOR(i, 1) = "" End If	This cell equals the theoretical yield of the ith row bond based its tenor. If the MATURITY_VECTOR(i, 1) is empty or blank, this cell equals blank, since you cannot interpolate without a tenor. Otherwise, the cell equals the theoretical yield of the bond calculated using the DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC and the coefficients of the polynomial stored in the BENCH_PAID_COEF_VECTOR.
2	INTER_VECTOR(i, 2)	BID SELECT	If (MATURITY_VECTOR(i, 2) = 0) Or (MATURITY_VECTOR(i, 2) = "") Then INTER_VECTOR(i, 2) = "" Else INTER_VECTOR(i, 2) = DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC(BENCH_BID_COEF_VECTOR, MATURITY_VECTOR(i, 2))(1, 1) If IsError(INTER_VECTOR(i, 2)) Then: INTER_VECTOR(i, 2) = "" End If	This cell equals the theoretical bid yield of the ith row bond based its tenor. If the MATURITY_VECTOR(i, 2) is empty or blank, this cell equals blank, since you cannot interpolate without a tenor. Otherwise, the cell equals the theoretical yield of the bond calculated using the DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC and the coefficients of the polynomial stored in the BENCH_BID_COEF_VECTOR.
3	INTER_VECTOR(i, 3)	ASK SELECT	If (MATURITY_VECTOR(i, 3) = 0) Or (MATURITY_VECTOR(i, 3) = "") Then INTER_VECTOR(i, 3) = "" Else INTER_VECTOR(i, 3) = DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC(BENCH_ASK_COEF_VECTOR, MATURITY_VECTOR(i, 3))(1, 1) If IsError(INTER_VECTOR(i, 3)) Then: INTER_VECTOR(i, 3) = "" End If	This cell equals the theoretical ask yield of the ith row bond based its tenor. If the MATURITY_VECTOR(i, 3) is empty or blank, this cell equals blank, since you cannot interpolate without a tenor. Otherwise, the cell equals the theoretical yield of the bond calculated using the DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC and the coefficients of the polynomial stored in the BENCH_ASK_COEF_VECTOR.
4	INTER_VECTOR(i, 4)	SWAP APPROX	If (MATURITY_VECTOR(i, 1) = 0) Or (MATURITY_VECTOR(i, 1) = "") Then INTER_VECTOR(i, 4) = "" Else INTER_VECTOR(i, 4) = DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC(SWAP_COEF_VECTOR, MATURITY_VECTOR(i, 1))(1, 1) If IsError(INTER_VECTOR(i, 4)) Then: INTER_VECTOR(i, 4) = "" End If	This cell equals the theoretical yield of the fixed leg of a swap of equivalent tenor to the ith row bond's tenor found in MATURITY_VECTOR(i, 1). If the MATURITY_VECTOR(i, 1) is empty or blank, this cell equals blank, since you cannot interpolate without a tenor. Otherwise, the cell equals the theoretical yield of the swap calculated using the DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC and the coefficients of the polynomial stored in the SWAP_COEF_VECTOR.

Part F – Building the INTERP_SPREADS_VECTOR

The INTERP_SPREADS_VECTOR stores the spreads in basis points between the market observed yield to worst and the interpolated yields for the SELECTED, BID, ASK, and SWAP APPROX. The SWAP APPROX spread is calculated as the difference between the market observed SELECTED yield to worst and the interpolated swap yield.

INTERP_SPREADS_VECTOR				
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**
1	INTERP_SPREADS_VECTOR(i, 1)	SELECTED	If (YTW_VECTOR(i, 9) <> "") And (YTW_VECTOR(i, 9) <> 0) And (INTER_VECTOR(i, 1) <> "") And (INTER_VECTOR(i, 1) <> 0) Then INTERP_SPREADS_VECTOR(i, 1) = (-INTER_VECTOR(i, 1) + YTW_VECTOR(i, 9)) * FACTOR3_VAL Else INTERP_SPREADS_VECTOR(i, 1) = "" End If	This cell equals the difference, or spread, between the observed market selected yield to worst and the interpolated theoretical yield. The difference is multiplied by FACTOR3_VAL which equals 1000, converting the value into basis points. If no selected yield to worst data is available, when YTW_VECTOR(i, 9) is blank or zero, or there is no theoretical yield data, when INTER_VECTOR(i, 1) is blank or zero.
2	INTERP_SPREADS_VECTOR(i, 2)	BID SELECT	If (YTW_VECTOR(i, 12) <> "") And (YTW_VECTOR(i, 12) <> 0) And (INTER_VECTOR(i, 2) <> "") And (INTER_VECTOR(i, 2) <> 0) Then INTERP_SPREADS_VECTOR(i, 2) = (-INTER_VECTOR(i, 2) + YTW_VECTOR(i, 12)) * FACTOR3_VAL Else INTERP_SPREADS_VECTOR(i, 2) = "" End If	This cell equals the difference, or spread, between the observed market bid yield to worst and the interpolated theoretical bid yield. The difference is multiplied by FACTOR3_VAL which equals 1000, converting the value into basis points. If no selected yield to worst data is available, when YTW_VECTOR(i, 12) is blank or zero, or there is no theoretical yield data, when INTER_VECTOR(i, 2) is blank or zero.
3	INTERP_SPREADS_VECTOR(i, 3)	ASK SELECT	If (YTW_VECTOR(i, 15) <> "") And (YTW_VECTOR(i, 15) <> 0) And (INTER_VECTOR(i, 3) <> "") And (INTER_VECTOR(i, 3) <> 0) Then INTERP_SPREADS_VECTOR(i, 3) = (-INTER_VECTOR(i, 3) + YTW_VECTOR(i, 15)) * FACTOR3_VAL Else INTERP_SPREADS_VECTOR(i, 3) = "" End If	This cell equals the difference, or spread, between the observed market selected ask yield to worst and the interpolated theoretical ask yield. The difference is multiplied by FACTOR3_VAL which equals 1000, converting the value into basis points. If no selected yield to worst data is available, when YTW_VECTOR(i, 15) is blank or zero, or there is no theoretical yield data, when INTER_VECTOR(i, 3) is blank or zero.
4	INTERP_SPREADS_VECTOR(i, 4)	SWAP APPROX	If (YTW_VECTOR(i, 9) <> "") And (YTW_VECTOR(i, 9) <> 0) And (INTER_VECTOR(i, 4) <> "") And (INTER_VECTOR(i, 4) <> 0) Then INTERP_SPREADS_VECTOR(i, 4) = (-INTER_VECTOR(i, 4) + YTW_VECTOR(i, 9)) * FACTOR3_VAL Else INTERP_SPREADS_VECTOR(i, 4) = "" End If	This cell equals the difference, or spread, between the observed market selected yield to worst and the interpolated theoretical swap yield. The difference is multiplied by FACTOR3_VAL which equals 1000, converting the value into basis points. If no selected yield to worst data is available, when YTW_VECTOR(i, 9) is blank or zero, or there is no theoretical yield data, when INTER_VECTOR(i, 4) is blank or zero.

Part G – Building the PRICE_VECTOR

The PRICE_VECTOR stores the theoretical clean price of all the bonds in the data set based on the interpolated yields from each of the yield functions.

		PRICE_VECTOR		
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**
1	PRICE_VECTOR(i, 1)	SELECTED	If YTW_VECTOR(i, 10) = "M" Then TMATURITY_VAL = DATA_MATRIX(i, c(4)) TREDEMPTION_VAL = DATA_MATRIX(i, c(5)) Else TMATURITY_VAL = DATA_MATRIX(i, c(6)) TREDEMPTION_VAL = DATA_MATRIX(i, c(7)) End If If (INTER_VECTOR(i, 1) = 0) Or (INTER_VECTOR(i, 1) = "") Or TMATURITY_VAL = 0 Or TMATURITY_VAL = "" Or TREDEMPTION_VAL = 0 Or TREDEMPTION_VAL = "" Then PRICE_VECTOR(i, 1) = "" Else PRICE_VECTOR(i, 1) = BOND_CASH_PRICE_FUNC(SETTLEMENT, TMATURITY_VAL, DATA_MATRIX(i, c(3)) / FACTOR1_VAL, INTER_VECTOR(i, 1), FREQUENCY, TREDEMPTION_VAL, COUNT_BASIS, 1) If IsError(PRICE_VECTOR(i, 1)) Then: PRICE_VECTOR(i, 1) = "" End If	This cell equals the theoretical clean price of the bond based on the model calculated selected yield. The cash price is calculated using the BOND_CASH_PRICE_FUNC. If the market observed yield to worst was the yield to maturity of the bond, the maturity date is used in the pricing function. Otherwise, the call date is used. The coupon rate (DATA_MATRIX(i, 3)) must be divided by FACTOR1_VAL to convert into a percentage. If the interpolated yield, maturity, or redemption value is either blank or zero, the cell will equal blank (""). If an error occurs in the bond pricing calculation, the cell will equal blank ("").
2	PRICE_VECTOR(i, 2)	BID SELECT	If YTW_VECTOR(i, 13) = "M" Then TMATURITY_VAL = DATA_MATRIX(i, c(4)) TREDEMPTION_VAL = DATA_MATRIX(i, c(5)) Else TMATURITY_VAL = DATA_MATRIX(i, c(6)) TREDEMPTION_VAL = DATA_MATRIX(i, c(7)) End If If (INTER_VECTOR(i, 2) = 0) Or (INTER_VECTOR(i, 2) = "") Or TMATURITY_VAL = 0 Or TMATURITY_VAL = "" Or TREDEMPTION_VAL = 0 Or TREDEMPTION_VAL = "" Then PRICE_VECTOR(i, 2) = "" Else PRICE_VECTOR(i, 2) = BOND_CASH_PRICE_FUNC(SETTLEMENT, TMATURITY_VAL, DATA_MATRIX(i, c(3)) / FACTOR1_VAL, INTER_VECTOR(i, 2), FREQUENCY, TREDEMPTION_VAL, COUNT_BASIS, 1) If IsError(PRICE_VECTOR(i, 2)) Then: PRICE_VECTOR(i, 2) = "" End If	This cell equals the theoretical clean price of the bond based on the model calculated selected bid yield. The cash price is calculated using the BOND_CASH_PRICE_FUNC. If the market observed yield to worst was the yield to maturity of the bond, the maturity date is used in the pricing function. Otherwise, the call date is used. The coupon rate (DATA_MATRIX(i, 3)) must be divided by FACTOR1_VAL to convert into a percentage. If the interpolated yield, maturity, or redemption value is either blank or zero, the cell will equal blank (""). If an error occurs in the bond pricing calculation, the cell will equal blank ("").
3	PRICE_VECTOR(i, 3)	ASK SELECT	If YTW_VECTOR(i, 16) = "M" Then TMATURITY_VAL = DATA_MATRIX(i, c(4)) TREDEMPTION_VAL = DATA_MATRIX(i, c(5)) Else TMATURITY_VAL = DATA_MATRIX(i, c(6)) TREDEMPTION_VAL = DATA_MATRIX(i, c(7)) End If If (INTER_VECTOR(i, 3) = 0) Or (INTER_VECTOR(i, 3) = "") Or TMATURITY_VAL = 0 Or TMATURITY_VAL = "" Or TREDEMPTION_VAL = 0 Or TREDEMPTION_VAL = "" Then PRICE_VECTOR(i, 3) = "" Else PRICE_VECTOR(i, 3) = BOND_CASH_PRICE_FUNC(SETTLEMENT, TMATURITY_VAL, DATA_MATRIX(i, c(3)) / FACTOR1_VAL, INTER_VECTOR(i, 3), FREQUENCY, TREDEMPTION_VAL, COUNT_BASIS, 1) If IsError(PRICE_VECTOR(i, 3)) Then: PRICE_VECTOR(i, 3) = "" End If	This cell equals the theoretical clean price of the bond based on the model calculated selected ask yield. The cash price is calculated using the BOND_CASH_PRICE_FUNC. If the market observed yield to worst was the yield to maturity of the bond, the maturity date is used in the pricing function. Otherwise, the call date is used. The coupon rate (DATA_MATRIX(i, 3)) must be divided by FACTOR1_VAL to convert into a percentage. If the interpolated yield, maturity, or redemption value is either blank or zero, the cell will equal blank (""). If an error occurs in the bond pricing calculation, the cell will equal blank ("").

Section III: GOVERNMENT_BONDS_TRADING_DESK_FUNC Output

The GOVERNMENT_BONDS_TRADING_DESK_FUNC has nine output options that can be adjusted through toggling the OUTPUT input:

i. OUTPUT = 0 – Returns the TEMP_MATRIX (FIGURE 3)
ii. OUTPUT = 1 – Returns the CHART_MATRIX (FIGURE 4)
iii. OUTPUT = 2 – Returns the YTM_VECTOR
iv. OUTPUT = 3 – Returns the YTC_VECTOR
v. OUTPUT = 4 – Returns the YTW_VECTOR
vi. OUTPUT = 5 – Returns the MATURITY_VECTOR
vii. OUTPUT = 6 – Returns the INTER_VECTOR
viii. OUTPUT = 7 – Returns the INTERP_SPREADS_VECTOR
ix. OUTPUT = 8 – Returns the PRICE_VECTOR

FIGURE 3: Output = 0

GOVERNMENT_BONDS_TRADING_DESK_FUNC - OUTPUT = 0					
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**	**ERROR CHECK**
1	TEMP_MATRIX(i, 1)	BOND NO	TEMP_MATRIX(i, 1) = i	The bond's number in the set	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
2	TEMP_MATRIX(i, 2)	VALOR	TEMP_MATRIX(i, 2) = DATA_MATRIX(i, 1)	A string of text consisting of the bond's coupon, issuing government, year of issuance, and maturity date.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
3	TEMP_MATRIX(i, 3)	YEARS MATURITY	TEMP_MATRIX(i, 3) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	
4	TEMP_MATRIX(i, 4)	ISSUER	TEMP_MATRIX(i, 4) = DATA_MATRIX(i, 2)	Issuer of the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
5	TEMP_MATRIX(i, 5)	BID PRICE	TEMP_MATRIX(i, 5) = DATA_MATRIX(i, 14)	The observed bid price in the market for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
6	TEMP_MATRIX(i, 6)	BID SIZE (m)	TEMP_MATRIX(i, 6) = DATA_MATRIX(i, 18) / FACTOR2_VAL	The quantity of bonds that an investor could sell at the stated bid price. The value has been divided by FACTOR2_VAL (1000) to convert the units into millions.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
7	TEMP_MATRIX(i, 7)	ASK PRICE	TEMP_MATRIX(i, 7) = DATA_MATRIX(i, 15)	The observed ask price in the market for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
8	TEMP_MATRIX(i, 8)	ASK SIZE (m)	TEMP_MATRIX(i, 8) = DATA_MATRIX(i, 19) / FACTOR2_VAL	The quantity of bonds that an investor could buy at the stated ask price.The value has been divided by FACTOR2_VAL (1000) to convert the units into millions.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
9	TEMP_MATRIX(i, 9)	HIST PRICE	TEMP_MATRIX(i, 9) = DATA_MATRIX(i, 16)	Historic closing price of the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
10	TEMP_MATRIX(i, 10)	SELECTED (YIELD TO WORST)	TEMP_MATRIX(i, 10) = YTW_VECTOR(i, 9)	Selected yield to worst for the bond	
11	TEMP_MATRIX(i, 11)	MAT/CALL (YIELD TO WORST)	TEMP_MATRIX(i, 11) = YTW_VECTOR(i, 10)	Identifies whether the selected yield to worst was the bond's yield to maturity ("M") or yield to call ("C")	
12	TEMP_MATRIX(i, 12)	MID/HIST (YIELD TO WORST)	TEMP_MATRIX(i, 12) = YTW_VECTOR(i, 11)	Identifies whether the selected yield to worst was found using an average of bid yield to worst and ask yield to worst ("Mid"), the last trading price ("Tr"), or the historic closing price ("Hist").	
13	TEMP_MATRIX(i, 13)	BENCH_BID_SELECTED (YIELD TO WORST)	TEMP_MATRIX(i, 13) = YTW_VECTOR(i, 12)	Selected bid yield to worst for the bond	
14	TEMP_MATRIX(i, 14)	MAT/CALL (YIELD TO WORST)	TEMP_MATRIX(i, 14) = YTW_VECTOR(i, 13)	Identifies whether the selected bid yield to worst was the bond's yield to maturity ("M") or yield to call ("C")	
15	TEMP_MATRIX(i, 15)	MID/HIST (YIELD TO WORST)	TEMP_MATRIX(i, 15) = YTW_VECTOR(i, 14)	Identifies whether the selected bid yield to worst was found using the bid yield to worst ("Bid") or the historic closing price ("Hist").	
16	TEMP_MATRIX(i, 16)	BENCH_ASK_SELECTED (YIELD TO WORST)	TEMP_MATRIX(i, 16) = YTW_VECTOR(i, 15)	Selected ask yield to worst for the bond	
17	TEMP_MATRIX(i, 17)	MAT/CALL (YIELD TO WORST)	TEMP_MATRIX(i, 17) = YTW_VECTOR(i, 16)	Identifies whether the selected ask yield to worst was the bond's yield to maturity ("M") or yield to call ("C")	
18	TEMP_MATRIX(i, 18)	MID/HIST (YIELD TO WORST)	TEMP_MATRIX(i, 18) = YTW_VECTOR(i, 17)	Identifies whether the selected yield to worst was found using the ask yield to worst ("Ask") or the historic closing price ("Hist").	
19	TEMP_MATRIX(i, 19)	BENCH_CURVE (Spreads: Bps)	TEMP_MATRIX(i, 19) = INTERP_SPREADS_VECTOR(i, 1)	This cell equals the difference, or spread, in basis points between the observed market selected yield to worst and the interpolated theoretical yield	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
20	TEMP_MATRIX(i, 20)	SWAP_CURVE (Spreads: Bps)	TEMP_MATRIX(i, 20) = INTERP_SPREADS_VECTOR(i, 4)	This cell equals the difference, or spread, in basis points between the observed market selected yield to worst and the interpolated theoretical swap yield	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
21	TEMP_MATRIX(i, 21)	BID (MODEL PRICE)	TEMP_MATRIX(i, 21) = PRICE_VECTOR(i, 2)	This cell equals the bid price generated by the model	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
22	TEMP_MATRIX(i, 22)	ASK (MODEL PRICE)	TEMP_MATRIX(i, 22) = PRICE_VECTOR(i, 3)	This cell equals the ask price generated by the model	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
23	TEMP_MATRIX(i, 23)	ACTION (TRADING)	If (TEMP_MATRIX(i, 3) <> 0) And (TEMP_MATRIX(i, 3) <> "") And (TEMP_MATRIX(i, 5) <> "") And (TEMP_MATRIX(i, 5) <> 0) And (TEMP_MATRIX(i, 7) <> "") And (TEMP_MATRIX(i, 7) <> 0) Then If (TEMP_MATRIX(i, 21) <> "") And (TEMP_MATRIX(i, 21) <> 0) And (TEMP_MATRIX(i, 8) <> "") And (TEMP_MATRIX(i, 8) <> 0) Then If TEMP_MATRIX(i, 7) < (TEMP_MATRIX(i, 21) - ACTION_PRICE_SENSITIVITY) Then TEMP_MATRIX(i, 23) = "Buy " & Format(TEMP_MATRIX(i, 8), "0.00") & "m @ " & Format(TEMP_MATRIX(i, 7), "0.00") Else If (TEMP_MATRIX(i, 22) <> "") And (TEMP_MATRIX(i, 22) <> 0) And (TEMP_MATRIX(i, 6) <> "") And (TEMP_MATRIX(i, 6) <> 0) Then If TEMP_MATRIX(i, 22) < (TEMP_MATRIX(i, 5) - ACTION_PRICE_SENSITIVITY) Then TEMP_MATRIX(i, 23) = "Sell " & Format(TEMP_MATRIX(i, 6), "0.00") & "m @ " & Format(TEMP_MATRIX(i, 5), "0.00") Else TEMP_MATRIX(i, 23) = "" End If Else TEMP_MATRIX(i, 23) = "" End If End If Else TEMP_MATRIX(i, 23) = "" End If Else TEMP_MATRIX(i, 23) = "" End If	This cell will identify trading signals if market prices are significantly different than those generated by the model. A buy signal is generated if the model generated bid price is more than the ACTION_PRICE_SENSITIVITY value (default = 0.4) above the market observed ask price. A sell signal is generated if the model generated ask price is less than the market observed bid price by more than the ACTION_PRICE_SENSITIVITY. If neither of the signal conditions are met, the cell will equal blank (""). To prevent errors, if any of the input values equal zero or blank then this cell will equal blank (""). The ACTION_PRICE_SENSITIVITY should be adjusted using the user's discretion.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
24	TEMP_MATRIX(i, 24)	ISSUED/OUTST.AMT (TRADING)	TEMP_MATRIX(i, 24) = DATA_MATRIX(i, 8) & " / " & DATA_MATRIX(i, 9)	A string of text showing: "amount issued / the oustanding amount" of the ith row bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
25	TEMP_MATRIX(i, 25)	TRADING/VOLUME (TRADING)	TEMP_MATRIX(i, 25) = DATA_MATRIX(i, 17)	Trading volume for the day for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
26	TEMP_MATRIX(i, 26)	MOODY (RATING)	TEMP_MATRIX(i, 26) = DATA_MATRIX(i, 10)	Moody's rating for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
27	TEMP_MATRIX(i, 27)	S&P (RATING)	TEMP_MATRIX(i, 27) = DATA_MATRIX(i, 11)	S&P rating for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
28	TEMP_MATRIX(i, 28)	INTERNAL (RATING)	TEMP_MATRIX(i, 28) = DATA_MATRIX(i, c(12))	Internal rating for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").

FIGURE 4: Output = 1

GOVERNMENT_BONDS_TRADING_DESK_FUNC - OUTPUT = 1					
Col.	**Address in Array**	**Heading**	**Value**	**Explanation**	**ERROR CHECK**
1	CHART_MATRIX(i, 1)	SELECTED TENOR	CHART_MATRIX(i, 1) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
2	CHART_MATRIX(i, 2)	SELECTED YTW INTERP	CHART_MATRIX(i, 2) = INTER_VECTOR(i, 1)	Interpolated selected yield to worst based on the model	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
3	CHART_MATRIX(i, 3)	BID TENOR	CHART_MATRIX(i, 3) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
4	CHART_MATRIX(i, 4)	BID YTW INTERP	CHART_MATRIX(i, 4) = INTER_VECTOR(i, 2)	Interpolated bid yield to worst based on the model	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
5	CHART_MATRIX(i, 5)	ASK TENOR	CHART_MATRIX(i, 5) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
6	CHART_MATRIX(i, 6)	ASK YTW INTERP	CHART_MATRIX(i, 6) = INTER_VECTOR(i, 3)	Interpolated ask yield to worst based on the model	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
7	CHART_MATRIX(i, 7)	SWAP TENOR	CHART_MATRIX(i, 7) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
8	CHART_MATRIX(i, 8)	SWAP INTERP	CHART_MATRIX(i, 8) = INTER_VECTOR(i, 4)	Interpolated swap yield based on the inputted SWAP_COEF_VECTOR	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
9	CHART_MATRIX(i, 9)	MOODY RATING	CHART_MATRIX(i, 9) = DATA_MATRIX(i, 10)	Moody's credit rating for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
10	CHART_MATRIX(i, 10)	SP RATING	CHART_MATRIX(i, 10) = DATA_MATRIX(i, c(11))	Standard & Poor's credit rating for the bond	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
11	CHART_MATRIX(i, 11)	CALL ACTION	If ((YTW_VECTOR(i, 10) = "M") And (YTW_VECTOR(i, 10) = 0) And (YTW_VECTOR(i, 10) = "")) Then CHART_MATRIX(i, 11) = "" Else If ((DATA_MATRIX(i, c(6)) <> "") And (DATA_MATRIX(i, 5) <> 0) And (DATA_MATRIX(i, c(7)) <> "") And (DATA_MATRIX(i, 6) <> 0)) Then CHART_MATRIX(i, 11) = "(CALL per " & Format(DATA_MATRIX(i, c(6)), "m/yy") & " at " & Format(DATA_MATRIX(i, c(7)), "0.0") & ")" Else CHART_MATRIX(i, 11) = "" End If End If	If the yield to call has been used as the yield to worst in YTW_VECTOR(i, 9) then this cell equals a string of text that describes the call feature. The string of text is as follows "CALL per (call date in m/yy format) at (the call redemption price)".	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
12	CHART_MATRIX(i, 12)	FULL LABEL	CHART_MATRIX(i, 12) = Trim(DATA_MATRIX(i, c(2)) & "(" & CHART_MATRIX(i, 9) & "/" & CHART_MATRIX(i, 10) & ")" & CHART_MATRIX(i, 11))	A concatenation of multiple strings of texts including: the selected yield to worst, (Moody's rating / S&P rating), CALL ACTION.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
13	CHART_MATRIX(i, 13)	BOND ID	CHART_MATRIX(i, 13) = i & IIf(YTW_VECTOR(i, 10) = "c", "c", "")	Bond number in the set. If the bond's yield to worst is the bond's yield to call then the bond's number will have a "c" beside it.	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
14	CHART_MATRIX(i, 14)	BID SPREAD	CHART_MATRIX(i, 14) = YTW_VECTOR(i, 12) - YTW_VECTOR(i, 9)	The difference between the bond's bid yield to worst and the bond's selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
15	CHART_MATRIX(i, 15)	ASK SPREAD	CHART_MATRIX(i, 15) = -YTW_VECTOR(i, 15) + YTW_VECTOR(i, 9)	The difference between the bond's selected yield to worst and the bond's ask yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
16	CHART_MATRIX(i, 16)	YTW TENOR	CHART_MATRIX(i, 16) = MATURITY_VECTOR(i, 1)	Tenor of the bond until maturity or call date depending on the selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").
17	CHART_MATRIX(i, 17)	YTW	CHART_MATRIX(i, 17) = YTW_VECTOR(i, 9)	The bond's selected yield to worst	To prevent errors, if any of the values used in the code equal zero or blank then this cell will equal blank ("").

Applied Model: Discount Factor Model

This segment of the report will discuss applying the Discount Factor Model. The user-defined function CHEAP_RICH_GOVERNMENT_BONDS_FUNC is used to replicate the conceptual Discount Factor Model. All supporting functions are discussed in depth in the appendix. The building of the CHEAP_RICH_GOVERNMENT_BONDS_FUNC will be discussed in three sections:

IV. Inputs & Model Parameters

V. Approximating the Discount Function Using Polynomial Regression
   a. Building the YDATA_VECTOR
   b. Building the XDATA_MATRIX
   c. Running the Regression

VI. CHEAP_RICH_GOVERNMENT_BONDS_FUNC Output:
   a. OUTPUT 0 – The Regression Coefficients
   b. OUTPUT 1 – Fair Price and Cheap/Rich Analysis
   c. OUTPUT 2 – The Discount Curve and Zero Curve

Section I: Inputs & Model Parameters

The Discount Factor Model requires data for a set of bonds that will characterize the polynomial regression. The model is a function of its inputs, therefore careful consideration must be paid to the choice of which bonds to include. Considerations include data integrity in addition to the impact of liquidity differences on prices. FIGURE 5 & 6 details the inputs required for the CHEAP_RICH_GOVERNMENT_BONDS_FUNC.

The understanding of two inputs into the CHEAP_RICH_GOVERNMENT_BONDS_FUNC in particular should be emphasized: OUTPUT and SHORT_RATE. OUTPUT, which determines what the function will return, will be explained in depth in the final section of the CHEAP_RICH_GOVERNMENT_BONDS_FUNC description. The SHORT_RATE input provides three options to characterize the boundary condition of the discount function. First, if SHORT_RATE is set to equal zero, then, no constraints will exist for the regression. Second, if SHORT_RATE is set equal to one, then, the intercept for the regression will be constrained to equal one. Logically, this reflects the fact that cash flows received in the present should not be discounted and therefore the discount factor at time equal to zero should equal one. Second, if SHORT_RATE is set equal to any other value, this assigned value is assumed to be the observed short-term rate in a particular market. For example, if running the model for Government of Canada bonds, the assigned value would be the Overnight Rate set by the Bank of Canada. In addition, in this scenario the intercept for the regression is constrained to one.

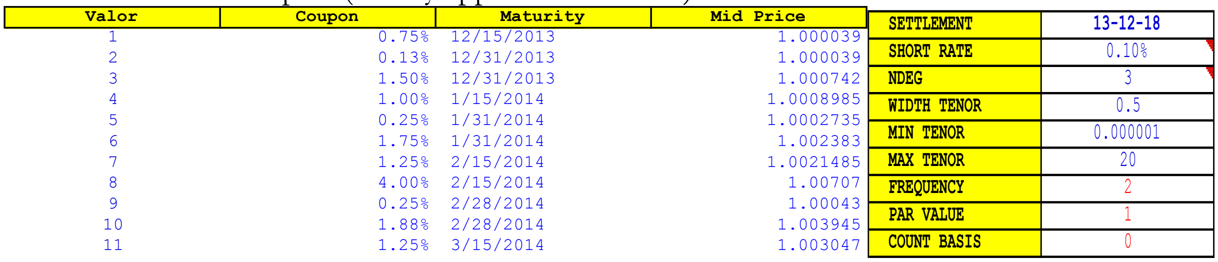
It is obvious that a user of the model should avoid using an unconstrained model, however, it is less clear whether a user should constrain the short-term rate or not. Continuing with the Government of Canada bonds example, the Overnight Rate set by the Bank of Canada is no doubt a close approximation of the rate that transactions will occur in the Canadian secured overnight lending market, however, including it as a constraint in the model will impact the coefficients of the discount function and therefore the zero rate curve. Although an in depth analysis of the consequences of each setting is beyond the scope of this report, it is important to note that the setting will produce materially different results in a trading environment.

FIGURE 5: Model Inputs for CHEAP_RICH_GOVERNMENT_BONDS_FUNC

Function: CHEAP_RICH_GOVERNMENT_BONDS_FUNC		
Inputs	**Format**	**Description**
OUTPUT	Integer {0, 1, 2}	The OUTPUT value will determine what the function ultimately produces. **0** - Produces a vector of the coefficients from the polynomial regression **1** - Calculates the theoretical prices of each of the bonds based on the discount function approximated by the polynomial regression. The function also calculates the difference between the market price price and the theoretical price to identify if it is trading rich or cheap. **2** - Calculates the discount factors and zero rates from MIN_TENOR to MAX_TENOR increasing the interval by WIDTH_TENOR
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
CLEAN_PRICE_RNG	Expressed as a percentage of par value	The range of the clean bond prices expressed as a percentage of par value. Clean price referes to a bond's price excluding accrued interest.
MATURITY_RNG	Date	The range of bond maturities
COUPON_RNG	Percentage (expressed as an annual rate)	The range of bond coupon rates
SHORT_RATE	Integer or Percentage {0, 1, Observed Short-Term Rate}	Defines the boundary conditions of the polynomial regression used to fit the discount function. **0** - No restrictions **1** - At time = 0, the discount function = 1. This is accomplished by setting the intercept term in the regression equal to 1. **Any other value** - The inputted value is assumed to be the observed short-term rate in the market (for example, the secured overnight lending rate). This rate should be expressed as an effective annual rate. The previous boundary condition is also set, at time = 0, the discount function = 1. **Default = 0.50%**
NDEG	Integer	The degree of the polynomial regression **Default = 3**
WIDTH_TENOR	Time expressed in years	Defines the width of each interval for the discount factors and zero rates calculated when OUTPUT = 2 **Default: 0.5 years**
MIN_TENOR	Time expressed in years	Defines the starting point for the discount factors and zero rates calculated when OUTPUT = 2 **Default: 0.000001 years**
MAX_TENOR	Time expressed in years	Defines the ending point for the discount factors and zero rates calculated when OUTPUT = 2 **Default: 10.5 years**
COEFFICIENT_RNG	Number	If predefined coefficients exist for the polynomial, those coefficients can be manually inputted using the COEFFICIENT_RNG. The model will use the COEFFICIENT_RNG as coefficients to the polynomial instead of calculating coefficients based on the sample set of bonds.
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
PAR_VALUE	Integer	The amount of principal repaid at maturity **Default: 100**
COUNT_BASIS	Integer {0, 1, 2, 3}	The day count convention of the sample set of bonds **Default: 0**
HOLIDAYS_RNG	Date	The dates of holidays

**Italicized inputs are OPTIONAL*

FIGURE 6*:* Model Inputs (as they appear in the Excel)



Valor	Coupon	Maturity	Mid Price
1	0.75%	12/15/2013	1.000039
2	0.13%	12/31/2013	1.000039
3	1.50%	12/31/2013	1.000742
4	1.00%	1/15/2014	1.0008985
5	0.25%	1/31/2014	1.0002735
6	1.75%	1/31/2014	1.002383
7	1.25%	2/15/2014	1.0021485
8	4.00%	2/15/2014	1.00707
9	0.25%	2/28/2014	1.00043
10	1.88%	2/28/2014	1.003945
11	1.25%	3/15/2014	1.003047

SETTLEMENT	13-12-18
SHORT RATE	0.10%
NDEG	3
WIDTH TENOR	0.5
MIN TENOR	0.000001
MAX TENOR	20
FREQUENCY	2
PAR VALUE	1
COUNT BASIS	0

Section II: Approximating the Discount Function Using Polynomial Regression

The core component of the Discount Factor Model is specifying the coefficients of a polynomial that approximates the unobserved market discount function. To run a polynomial regression, the CHEAP_RICH_GOVERNMENT_BONDS_FUNC generates two arrays: the YDATA_VECTOR and XDATA_MATRIX from the inputs and parameters. The model runs a regression between the YDATA_VECTOR and XDATA_MATRIX to determine the coefficients of the discount function. FIGURE 7 describes the key arrays involved in the CHEAP_RICH_GOVERNMENT_BONDS_FUNC.

$$\overset{\text{Dependent Variable Matrix}}{\begin{bmatrix} \text{Market Price Bond 1} \\ \text{Market Price Bond 2} \\ \text{Market Price Bond 3} \\ \text{Market Price Bond 4} \\ \text{Market Price Bond 5} \\ \vdots \\ \text{Market Price Bond j} \end{bmatrix}} = \overset{\text{Regression Coefficients}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}} \overset{\text{Independent Variable Matrix}}{\begin{bmatrix} \Sigma c_{1i} T_{1i}^{0} & \Sigma c_{1i} T_{1i}^{1} & \Sigma c_{1i} T_{1i}^{2} & \cdots & \Sigma c_{1i} T_{1i}^{k} \\ \Sigma c_{2i} T_{2i}^{0} & \Sigma c_{2i} T_{2i}^{1} & \Sigma c_{2i} T_{2i}^{2} & \cdots & \Sigma c_{2i} T_{2i}^{k} \\ \Sigma c_{3i} T_{3i}^{0} & \Sigma c_{3i} T_{3i}^{1} & \Sigma c_{3i} T_{3i}^{2} & \cdots & \Sigma c_{3i} T_{3i}^{k} \\ \Sigma c_{4i} T_{4i}^{0} & \Sigma c_{4i} T_{4i}^{1} & \Sigma c_{4i} T_{4i}^{2} & \cdots & \Sigma c_{4i} T_{4i}^{k} \\ \Sigma c_{5i} T_{5i}^{0} & \Sigma c_{5i} T_{5i}^{1} & \Sigma c_{5i} T_{5i}^{2} & \cdots & \Sigma c_{5i} T_{5i}^{k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma c_{ji} T_{ji}^{0} & \Sigma c_{ji} T_{ji}^{1} & \Sigma c_{ji} T_{ji}^{2} & \cdots & \Sigma c_{ji} T_{ji}^{k} \end{bmatrix}}$$

Part A: Building the YDATA_VECTOR

The YDATA_VECTOR is an array of the cash bond prices. The cash price of each bond is calculated by adding the accrued interest of each bond to its clean price. In order to calculate the accrued interest on the bond, the function ACCRINT_FUNC is used (See Appendix). The cash price reflects the amount of cash that investor would have to pay in order to acquire a particular bond. The cash price of the bond is used in the regression as opposed to the clean price because the XDATA_MATRIX includes the accrued interest in the first coupon payment.

$$\overset{\text{Dependent Variable Matrix}}{\begin{bmatrix} \text{Market Price Bond 1} \\ \text{Market Price Bond 2} \\ \text{Market Price Bond 3} \\ \text{Market Price Bond 4} \\ \text{Market Price Bond 5} \\ \vdots \\ \text{Market Price Bond j} \end{bmatrix}}$$

Part B: Building the XDATA_MATRIX

The XDATA_MATRIX is an array that reflects the independent variable data discussed in the conceptual model (Formula 7). The function CHEAP_RICH_POLYNOMIAL_COEF_FUNC is used to generate the array (see Appendix).

Independent Variable Matrix

$$\begin{bmatrix} \Sigma c_{1i}T_{1i}^{0} & \Sigma c_{1i}T_{1i}^{1} & \Sigma c_{1i}T_{1i}^{2} & \cdots & \Sigma c_{1i}T_{1i}^{k} \\ \Sigma c_{2i}T_{2i}^{0} & \Sigma c_{2i}T_{2i}^{1} & \Sigma c_{2i}T_{2i}^{2} & \cdots & \Sigma c_{2i}T_{2i}^{k} \\ \Sigma c_{3i}T_{3i}^{0} & \Sigma c_{3i}T_{3i}^{1} & \Sigma c_{3i}T_{3i}^{2} & \cdots & \Sigma c_{3i}T_{3i}^{k} \\ \Sigma c_{4i}T_{4i}^{0} & \Sigma c_{4i}T_{4i}^{1} & \Sigma c_{4i}T_{4i}^{2} & \cdots & \Sigma c_{4i}T_{4i}^{k} \\ \Sigma c_{5i}T_{5i}^{0} & \Sigma c_{5i}T_{5i}^{1} & \Sigma c_{5i}T_{5i}^{2} & \cdots & \Sigma c_{5i}T_{5i}^{k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma c_{ji}T_{ji}^{0} & \Sigma c_{ji}T_{ji}^{1} & \Sigma c_{ji}T_{ji}^{2} & \cdots & \Sigma c_{ji}T_{ji}^{k} \end{bmatrix}$$

Part C: Running the Regression

The CHEAP_RICH_GOVERNMENT_BONDS_FUNC runs a regression between the YDATA_VECTOR and XDATA_MATRIX using the function REGRESSION_MULT_COEF_FUNC. The REGRESSION_MULT_COEF_FUNC runs a least squares multiple linear regression using matrix algebra (see Appendix). The output from the regression is a vector of the coefficients which is stored in an array denoted COEFFICIENT_VECTOR.

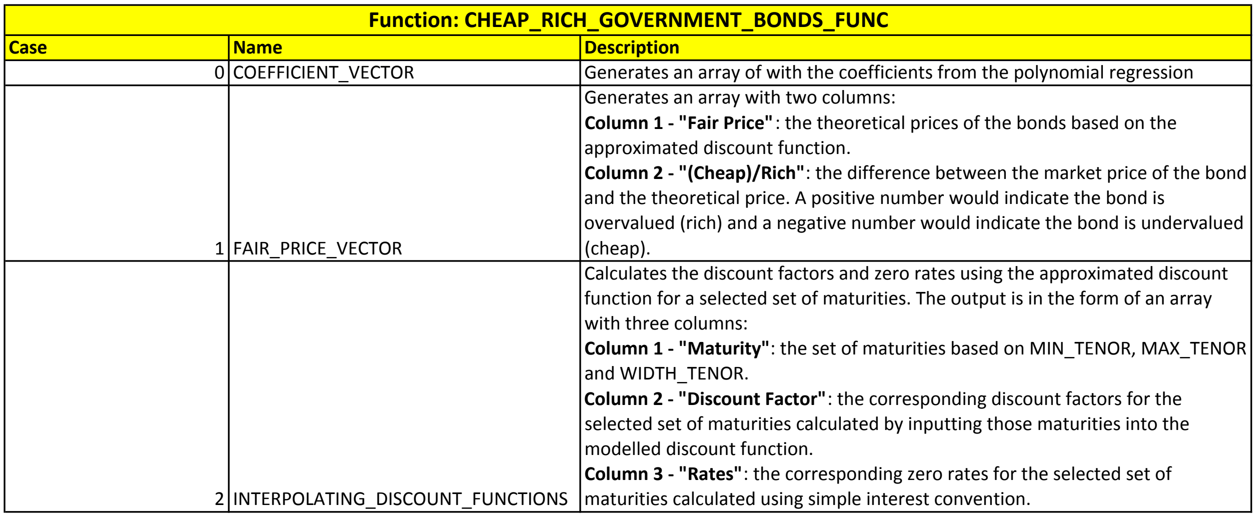
FIGURE 7*:* Key Components of the CHEAP_RICH_GOVERNMENT_BONDS_FUNC

Function: CHEAP_RICH_GOVERNMENT_BONDS_FUNC		
Number	**Components**	**Description**
1	NROWS	Number of bonds in the sample
2	PRICE_VECTOR(1 to NROWS, 1)	PRICE_VECTOR(i, 1) = CLEAN_PRICE_RNG(i, 1)
3	MATURITY_VECTOR(1 to NROWS, 1)	MATURITY_VECTOR(i, 1) = MATURITY_RNG(i, 1)
4	TENOR_VECTOR(1 to NROWS, 1)	
5	COUPON_VECTOR(1 to NROWS, 1)	COUPON_VECTOR(i, 1) = COUPON_RNG(i, 1)
6	INTEREST_VECTOR (1 to NROWS, 1)	An array containing accrued interest of each of the bonds in the sample divided by 100. Accrued interest is calculated using the function ACCRINT_FUNC.
7	CASH_VECTOR	An array of the cash price of the bonds calculated by adding accrued interest to the clean price: CASH_VECTOR = PRICE_VECTOR/PAR_VALUE + INTEREST_VECTOR
8	XDATA_MATRIX(1 to NROWS, 1 to NDEG)	An array containing the data for the independent variable in the regression. The XDATA_MATRIX is generated using the function CHEAP_RICH_POLYNOMIAL_COEF_FUNC: For i = 1 to NROWS For j = 1 To NDEG + 1 DATA_MATRIX(i, j) = CHEAP_RICH_POLYNOMIAL_COEF_FUNC(j - 1, SETTLEMENT, MATURITY_VECTOR(i, 1), COUPON_VECTOR(i, 1), FREQUENCY, COUNT_BASIS) Next j Next i
9	YDATA_MATRIX	The dependent variable in the regression: YDATA_MATRIX(i, 1) = CASH_VECTOR(i, 1)
10	COEFFICIENT_VECTOR	The coefficients of the polynomial regression performed on the XDATA_MATRIX and YDATA_MATRIX. The regression is run using the function REGRESSION_MULT_COEF_FUNC.

Section III: CHEAP_RICH_GOVERNMENT_BONDS_FUNC Output

The CHEAP_RICH_GOVERNMENT_BONDS_FUNC has three unique output options that are controlled from the OUTPUT input into the function. This section will discuss the three output options for the function. FIGURE 8 provides a brief overview of the output options.

FIGURE 8: Output options for CHEAP_RICH_GOVERNMENT_BONDS_FUNC



Function: CHEAP_RICH_GOVERNMENT_BONDS_FUNC

Case	Name	Description
0	COEFFICIENT_VECTOR	Generates an array of with the coefficients from the polynomial regression
1	FAIR_PRICE_VECTOR	Generates an array with two columns: **Column 1 - "Fair Price"**: the theoretical prices of the bonds based on the approximated discount function. **Column 2 - "(Cheap)/Rich"**: the difference between the market price of the bond and the theoretical price. A positive number would indicate the bond is overvalued (rich) and a negative number would indicate the bond is undervalued (cheap).
2	INTERPOLATING_DISCOUNT_FUNCTIONS	Calculates the discount factors and zero rates using the approximated discount function for a selected set of maturities. The output is in the form of an array with three columns: **Column 1 - "Maturity"**: the set of maturities based on MIN_TENOR, MAX_TENOR and WIDTH_TENOR. **Column 2 - "Discount Factor"**: the corresponding discount factors for the selected set of maturities calculated by inputting those maturities into the modelled discount function. **Column 3 - "Rates"**: the corresponding zero rates for the selected set of maturities calculated using simple interest convention.

Part A: OUTPUT 0 – The Regression Coefficients

When the input OUTPUT in the CHEAP_RICH_GOVERNMENT_BONDS_FUNC equals zero, the function will generate a vertical array of the coefficients for the discount function denoted the COEFFICIENT_VECTOR (FIGURE 9).

FIGURE 9: COEFFICIENT_VECTOR

COEFFICIENT_VECTOR

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

Part B: OUTPUT 1 – Fair Price and Cheap/Rich Analysis

When OUTPUT is equal to one, the CHEAP_RICH_GOVERNMENT_BONDS_FUNC will generate an array with the fair price of each bond from the input set and identifies by how much that bond is trading rich or cheap. The theoretical or fair price of the bond is determined by multiplying the regression coefficients with the independent variable matrix, XDATA_MATRIX (FIGURE 10). An alternative way of thinking about the theoretical price of the bond is illustrated logically in FIGURE 11. In order to get each bond's clean price, the accrued interest must be subtracted. A bond trading rich (cheap) refers to whether a bond is trading above (below) its fair price. Therefore to determine whether a bond is trading rich or cheap and by how much, the model subtracts the fair price from the observed market price. The return value of the CHEAP_RICH_GOVERNMENT_BONDS_FUNC is a two-column array with the fair price of the bond in the first column and cheap/rich analysis in the second (FIGURE 12).

FIGURE 10: Fair price of bond calculation

$$\overset{\text{Dependent Variable Matrix}}{\begin{bmatrix} \text{Dirty Fair Price Bond 1} \\ \text{Dirty Fair Price Bond 2} \\ \text{Dirty Fair Price Bond 3} \\ \text{Dirty Fair Price Bond 4} \\ \text{Dirty Fair Price Bond 5} \\ \vdots \\ \text{Dirty Fair Price Bond j} \end{bmatrix}} = \overset{\text{Regression Coefficients}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}} \overset{\text{Independent Variable Matrix}}{\begin{bmatrix} \Sigma c_{1i} T_{1i}^{0} & \Sigma c_{1i} T_{1i}^{1} & \Sigma c_{1i} T_{1i}^{2} & \cdots & \Sigma c_{1i} T_{1i}^{k} \\ \Sigma c_{2i} T_{2i}^{0} & \Sigma c_{2i} T_{2i}^{1} & \Sigma c_{2i} T_{2i}^{2} & \cdots & \Sigma c_{2i} T_{2i}^{k} \\ \Sigma c_{3i} T_{3i}^{0} & \Sigma c_{3i} T_{3i}^{1} & \Sigma c_{3i} T_{3i}^{2} & \cdots & \Sigma c_{3i} T_{3i}^{k} \\ \Sigma c_{4i} T_{4i}^{0} & \Sigma c_{4i} T_{4i}^{1} & \Sigma c_{4i} T_{4i}^{2} & \cdots & \Sigma c_{4i} T_{4i}^{k} \\ \Sigma c_{5i} T_{5i}^{0} & \Sigma c_{5i} T_{5i}^{1} & \Sigma c_{5i} T_{5i}^{2} & \cdots & \Sigma c_{5i} T_{5i}^{k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma c_{ji} T_{ji}^{0} & \Sigma c_{ji} T_{ji}^{1} & \Sigma c_{ji} T_{ji}^{2} & \cdots & \Sigma c_{ji} T_{ji}^{k} \end{bmatrix}}$$

FIGURE 11: Logic behind the fair price of a bond

Cash Flow Amount	Time Unti Cash Flow (years)	Discount Factor	PV of Cash Flow
0.5 * 5% * 100	0.5	$B_0 + 0.5\ B_1\ + 0.5^2\ B_2 + \ldots + 0.5^k\ B_k$	Cash Flow Amount * Discount Factor
0.5 * 5% * 100	1	$B_0 + 1\ B_1\ + 1^2\ B_2 + \ldots + 1^k\ B_k$	Cash Flow Amount * Discount Factor
⋮	⋮	⋮	⋮
100	5	$B_0 + 5\ B_1\ + 5^2\ B_2 + \ldots + 5^k\ B_k$	Cash Flow Amount * Discount Factor
		Fair Price of Bond	**Sum of PV of Cash Flows**

FIGURE 12: The Final Output Array

Fair Price	(Cheap) / Rich
1.0001	-0.0001
1.0000	0.0000
1.0008	-0.0001
1.0008	0.0001
1.0002	0.0001
1.0023	0.0001
1.0020	0.0002
1.0069	0.0001
1.0002	0.0002
1.0037	0.0002
1.0028	0.0002
1.0002	0.0003

Part C: OUTPUT 2 – The Discount Curve and Zero Rate Curve

In the final scenario, the CHEAP_RICH_GOVERNMENT_BONDS_FUNC generates an array that describes the discount curve and zero curve for a set of maturities. The MIN_TENOR, MAX_TENOR,

TENOR_WIDTH inputs into the function will determine the set of maturities, denoted as the TENOR_VECTOR (FIGURE 13). The set of maturities are converted from their original format, years from settlement, to the actual maturity date using the BOND_TENOR_DATES_FUNC (see Appendix). Discount factors for each maturity date in the set are calculated using the function DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC (FIGURE 14). The function DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC interpolates based on the coefficients of the polynomial discount function for a specified range (More details in the Appendix). From the discount factors, zero rates are computed using simple interest convention for each of the maturities in set. The output array is as appears in FIGURE 15.

FIGURE 13: The TENOR_VECTOR

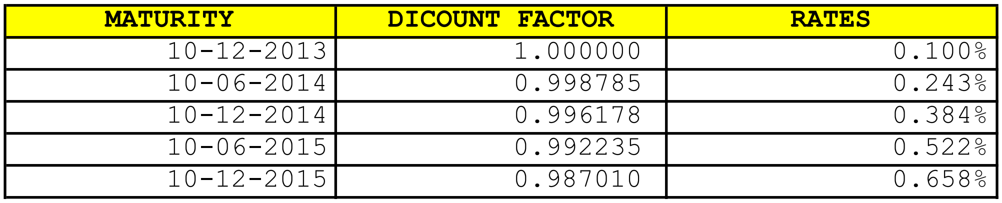
TENOR_VECTOR
MIN_TENOR
MIN_TENOR + WIDTH_TENOR
MIN_TENOR + 2*[WIDTH_TENOR]
⋮
MAX_TENOR

FIGURE 14: Calculating Zero Rates from Discount Factors Using Simple Interest

$\text{Zero Rate}_t = [1/D_t – 1]/t$

Where:
Zero Rate_t = The simple annual interest rate on a zero coupon bond maturing at t years from settlement
D_t = Discount factor for a cash flow received t years from settlement
t = Number of years until the cash flow is received

FIGURE 15: Output Array



MATURITY	DICOUNT FACTOR	RATES
10-12-2013	1.000000	0.100%
10-06-2014	0.998785	0.243%
10-12-2014	0.996178	0.384%
10-06-2015	0.992235	0.522%
10-12-2015	0.987010	0.658%

APPENDIX

Function: BOND_CASH_PRICE_FUNC

Purpose:

The BOND_CASH_PRICE_FUNC calculates the cash price of a bond by discounting the bond's cash flows using semi-annual compound convention.

Method:

1) The function calculates number of coupon payments remaining for the bond.
2) The function calculates the number of days from last coupon payment (PDAYS_VAL), the number of days until the next coupon payment (NDAYS_VAL), and the total days in the period (j = PDAYS_VAL + NDAYS_VAL) such that (PDAYS_VAL/j) is the fraction of a period that has elapsed.
3) The function calculates the present value of coupon and redemption payments by discounting each coupon and principal payment by the inputted yield using semi-annual compounding convention. The first coupon is discounted by (1 + YIELD/2)^(PDAYS_VAL/j), the second coupon is discounted by (1 + YIELD/2)^(1 + PDAYS_VAL/j), and so on. Discounting the ith coupon payment by (i -1 + PDAYS_VAL/j) reflects the fact that a partial period has elapsed. In calculating the present value of the cash flows, the function deducts the accrued interest. If OUTPUT = 1, the function will return this value, the clean price.
4) If OUTPUT = 0, the function will calculate the accrued interest of the bond using the function ACCRINT_FUNC and adding that value to the value calculated in step 3).

Inputs	Format	Description
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
CALL_DATE	Date	The call date of the bond
COUPON	Percentage	The bond's coupon rate (expressed as an annual rate)
YIELD	Percentage	The yield that will be used to discount the cash flows
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
REDEMPTION	Number	The amount of principal returned at maturity **Default: 100**
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention of bond **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**
OUTPUT	Integer {0, 1}	Case 0 - Function returns the cash price of the bond Case 1 - Function returns the clean price of the bond **Default: 0**

Function: BOND_YIELD_FUNC

Purpose:

The Bond_Yield_Func finds the yield to maturity of a bond, given its clean price, settlement date, maturity date, coupon rate, frequency of coupon payments, par value, day count convention, and a guess yield.

Method:

1) The function finds Y_VAL, yield to maturity, by inputting Call_Bond_Yield_Obj_Func into Parab_Zero_Func. The Parab_Zero_Func is an optimization function used to find a yield that will make the Call_Bond_Yield_Obj_Func equal zero or within the tolerance of plus or minus 10 ^ -15.
2) When the Call_Bond_Yield_Obj_Func equals zero the calculated bond price and the actual bond price are equal. At that point, the yield used to calculate the bond price is equal to the yield to maturity. If the Converg_Val is greater than or less than zero, which occurs when nLOOPS elapse and no solution is found in the Parab_Zero_Func, it will return the Guess_Yield inputted.
3) If an error occurs in any of the input functions, it will return Pub_Epsilon, which is set as equal to 2^52, resulting the Bond_Yield_Func equaling the inputted Guess_Yield.

Inputs	**Format**	**Description**
CLEAN_PRICE	Number	Clean price of the bond
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
MATURITY	Date	The bond's date of maturity
COUPON	Percentage	The bond's coupon rate (expressed as an annual rate)
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
REDEMPTION	Number	The amount of principal returned at maturity **Default: 100**
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention of bond **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**
GUESS_YIELD	Percent	Sets the interval for the optimization problem as [-GUESS_YIELD, GUESS_YIELD] **Default: 0.3**

Function: CALL_YIELD_FUNC

Purpose:

The CALL_YIELD_FUNC calculates the yield to maturity of a bond if the bond was redeemed at the call date.

Method:

1) The CALL_YIELD_FUNC uses the BOND_YIELD_FUNC to calculate the yield to maturity, but replaces the MATURITY date with the CALL_DATE.

Inputs	Format	Description
CLEAN_PRICE	Number	Clean price of the bond
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
CALL_DATE	Date	The call date of the bond
COUPON	Percentage	The bond's coupon rate (expressed as an annual rate)
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
REDEMPTION	Number	The amount of principal returned at maturity **Default: 100**
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention of bond **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**
GUESS_YIELD	Percent	Sets the interval for the optimization problem as [-GUESS_YIELD, GUESS_YIELD] **Default: 0.1**

Function: VECTOR_TRIM_FUNC

Purpose:

The function VECTOR_TRIM_FUNC is used to remove blanks (""), zeros, or any other specified value from an array. The function will resize the array for the newly "trimmed" vector.

Method:

1) The function checks if the array is a one dimensional horizatonal array. If this is the case, the function transposes the data into a vertical array using the MATRIX_TRANSPOSE_FUNC.

2) The function looks through each entry in the array to see if it is a blank (""), empty, or equal to the REF_VAL. If any of those conditions are true the counter, k, increased by 1 and the entry is not included in the vector that stores the newly trimmed data (TEMP_VECTOR). Otherwise, the entry is included at the address TEMP_VECTOR(i-k, 1). Subtracting k from i ensures that everytime a piece of data is removed, the vector shrinks by one preventing any empty addresses in the vector. The array is resized by subtracting k from the original height of the array.

Inputs	Format	Description
DATA_RNG	Array	The array that will be trimmed
REF_VAL	Any value	The value that will be removed in the process of trimming the vector **Default: 0**

Function: POLYNOMIAL_REGRESSION_FUNC

Purpose:
The function POLYNOMIAL_REGRESSION_FUNC uses LU decomposition to run a polynomial regression between the YDATA_RNG and XDATA_RNG returning a vector containing the coefficients of the regression.

Method:
1) The function sets XDATA_VECTOR = XDATA_RNG and YDATA_VECTOR = YDATA_RNG
2) The function checks if the XDATA_VECTOR or YDATA_VECTOR one dimensional horizatonal array. If this is the case, the function transposes the data into a vertical array using the MATRIX_TRANSPOSE_FUNC.
3) The function then checks if the XDATA_VECTOR and the YDATA_VECTOR are the same length. If they are not the same, the function cannot run a regression and therefore returns an error.
4) If NROWS is set to its default value of zero, the function determines NROWS by calculating the length of the XDATA_VECTOR. The maximum degrees of the polynomial is (NROWS - 1), therefore if NDEG > (NROWS - 1), the function resets NDEG to (NROWS - 1). Similarly, if the number of degrees is above 7 then the function resets NDEG to 7 in order to prevent too perfect a fit.
5) The function then uses an algorithm for LU decomposition to solve the set of linear equations, computing the coefficients of the polynomial regression.

Inputs	Format	Description
XDATA_RNG	Array	The data set for the independent variables in the regression
YDATA_RNG	Array	The data see for the dependent variable in the regression
NDEG	Integer	The number of degrees of the polynomial regression
NROWS	Number	The number of rows in the XDATA_RNG. If it is not entered it will be calculated in the function. **Default: 0**

Function: YEARFRAC_FUNC

Purpose:
The purpose of the function YEARFRAC_FUNC is to calculate the number of years (including any fractional component) between two points in time.

Method:
1) Check the inputs for errors. If START_DATE = 0, END_DATE = 0, or START_DATE > END_DATE (the start date is after the end date) then function will return an error.
2) If there are no errors, the function will count the number of days between the two dates and divide by the number of days in a year, returning a value that is the number of years between two dates (including any fractional component). Both calculations take into consideration the day count convention specified.

Inputs	Format	Description
START_DATE	Date	The start date of the time period you are measuring
END_DATE	Date	The end date of the time period you are measuring
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention to be used for the measurement **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**

Function: MATRIX_TRANSPOSE_FUNC

Purpose:

The purpose of the function MATRIX_TRANSPOSE_FUNC is to tranpose an array.

Method:

1) If the DATA_RNG is a two dimensional array then the function creates a temporary array (TEMP_MATRIX) and defines its size as having as many columns as DATA_RNG has rows and as many rows as DATA_RNG has columns. The function then inserts all of the DATA_RNG's columns into the TEMP_MATRIX as rows.
2) If the DATA_RNG is a one dimensional array then the function creates a temporary array (TEMP_MATRIX) and inserts the data in the DATA_RNG into the TEMP_MATRIX as a column.
3) If the DATA_RNG is not a one or two dimensional array, the function will return an error

Inputs	**Format**	**Description**
DATA_RNG	Array	The array that will be transposed

Function: BOND_TENOR_DATES_FUNC

Purpose:

The purpose of the function BOND_TENOR_DATES_FUNC is to convert a tenor (length of time in years) into a maturity date such that the time from the settlement date to the maturity dates equals the length of time prescribed by the tenors. The output may not lie on a weekend or any day included in the holiday range.

Inputs	**Format**	**Description**
TENOR_RNG	Array of periods of time (in years)	The array of tenors that will be converted to maturity dates. Tenors are lengths of time (i.e. 1 year, 2 years, 3 years, etc.)
SETTLEMENT	Date	The settlement date. In this case the settlement date acts as the start date when converting tenors into specific maturity dates.
HOLIDAYS_RNG	Array of Dates	Restricts any maturity date from occuring on any date included in this range

Function: ACCRINT_FUNC

Purpose:

The purpose of the function ACCRINT_FUNC is to calculate the accrued interest for a particular bond. All calculations take into consideration the day count convention specified by tbe COUNT_BASIS input.

Method:

1) If the MATURITY date is before the SETTLEMENT date, FREQUENCY of coupon payment is less than annual, or the coupon rate is less than or equal to zero then the function return value = 0
2) Otherwise, the function calculates the number for days since the last coupon payment and the number of days until the next coupon payment to find the total number of days in the period. The function then divides the number of days since the last coupon payment by the total days in the period to find the fraction of time that has occured in the period.
3) The fraction represents the fraction of the next coupon payment that is owed to the current bondholder in the event of a transaction. Therefore, by multiplying this fraction by the coupon payment you find the amount of accrued interest owing for a bond.

Inputs	**Format**	**Description**
SETTLEMENT	Integer	The settlement date if you were to buy or sell the bond
MATURITY	Date	The bond's date of maturity
COUPON	Percentage	The bond's coupon rate (expressed as an annual rate)
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
COUNT_BASIS	Integer {0, 1, 2, 3, 4}	The day count convention of bond **0** - US 30/360 **1** - Actual/Actual **2** - Actual/360 **3** - Actual/365 **4** - European 30/360 **Default: 0**

Function: REGRESSION_MULT_COEF_FUNC

Purpose:
The function REGRESSION_MULT_COEF_FUNC performs a multiple regression using the least squares method to calculate a straight line that best fits your data, and returns the array of coefficients.

Method:

1) The function checks to make sure that the YDATA_RNG and XDATA_RNG both are the same length, meaning that for all Y data their is corresponding X data and vice versa. If there is not the case, the function returns an error.
2) The function runs a multiple regression using the following matrix algebra:

$$[\text{Coefficients Vector}] = [X^T X]^{-1} [X^T] [Y]$$

X^T = Transposed X Data Matrix
Y = Y Data Matrix
$[]^{-1}$ = Inverse Matrix

Inputs	Format	Description
XDATA_RNG	Range of values	The data set for the independent variables in the regression
YDATA_RNG	Range of values	The data see for the dependent variable in the regression
INTERCEPT_FLAG	Boolean (TRUE or FALSE)	If the value = TRUE, then B_0, the intercept in the regression, is calculated normally. If the value = FALSE, then B_o is forced to equal zero. **Default: TRUE**
MATRIX_INVERSE_TYPE	Integer {0, 1, any other number}	Determines the method of calculating the inverse of a matrix: Case 0: Lu Decomposition Case 1: SVD Decomposition Case Else: Cholesky **Default: 0**

Function: CHEAP_RICH_POLYNOMIAL_COEF_FUNC

Purpose:
The function CHEAP_RICH_POLYNOMIAL_COEF_FUNC supports the CHEAP_RICH_GOVERNMENT_BONDS_FUNC in producing the XDATA_MATRIX for the polynomial regression.

Method:
1) For a single bond and a specified number of degreees, the CHEAP_RICH_POLYNOMIAL_COEF_FUNC will perform the following calculation:

$$[\text{COUPON/FREQUENCY}][P]^{NDEG} + [\text{Coupon/Frequency}][P + 1/\text{FREQUENCY}]^{NDEG} + \ldots + [\text{TENOR}]^{NDEG}$$

P = Fraction of a year until the next coupon payment
TENOR = The total number of years (including any fraction of a year) remaining until maturity

Inputs	Format	Description
NDEG	Integer	The degree of the polynomial regression **Default = 3**
SETTLEMENT	Date	The settlement date if you were to buy or sell the bond
MATURITY	Date	The bond's maturity
COUPON	Percentage	The bond's coupon rate (expressed as an annual rate)
FREQUENCY	Integer	The number of coupon payments paid per year **Default: 2 coupon payments per year**
COUNT_BASIS	Integer {0, 1, 2, 3}	The day count convention of the sample set of bonds **Default: 0**

Function: DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC

Purpose:

The function DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC will interpolate based on a polynomial coefficient vector and independent variable(s) data matrix. It will calculated the discount factors for a specific set of tenors prescribed in the XDATA_RNG given the polynomial regression coefficients definied by COEF_RNG.

Method:

1) If the COEF_RNG or XDATA_RNG are horizontal arrays, the function will transpose them to vertical arrays using the function MATRIX_TRANSPOSE_FUNC
2) The number of degrees is calculated by subtracting one from the number of rows in the COEF_VECTOR
3) For each tenor in the XDATA_MATRIX the function will perform the following equation:

$$\mathbf{D_i = \beta_0 + \beta_1 T_i + \beta_2 T_i^2 + \ldots + \beta_k T_i^k}$$

Where: D_i = Discount factor at time = i
k = The degree of the polynomial, determined in step 2
T_i = Time until the ith cash flow, in years defined by the XDATA_RNG
β_k = The coefficients from COEF_RNG

4) The output from the DISCOUNT_POLYNOMIAL_INTERPOLATION_FUNC is a vertical array of the discount factors calculated in the previous step for each tenor specified in the XDATA_MATRIX

Inputs	Format	Description
COEF_RNG	Array	The coefficients from a polynomial regression
XDATA_RNG	Array	Data matrix of tenors that discount factors will be derived for