Bond Price Simulation

Jiahao Zhang

1 Background

Bonds usually serves as an financial contract issued by corporation, goverments and other financial companies to raise money from society. [1, 2] While the investors lend money to those issuers, they have an agreement on the fixed interest rate (usually called coupon rate) as rewards before the agreed payback date (usually called mature date). [3, 4]. The basic bond characteristics includes face values, coupon rate, coupon dates, maturity date and issue price. [5] However, bonds are usually trading at the market. They are not only determined by those basic characteristics, but also influenced by other properties such as interest rate and defaulting probability (surviving probability). [5, 6] For an example, when the market interest rate goes up, bond prices will fail in order to keep up with the interest rate.[7, 8] There are two ways to think about this. First, if the bond price doesn't fall, the bond holders will sell their bonds to chase for higher portfolio return. Thus, the bond price will drop because selling activity is predominant. Second, the issuer will need to make their bonds become attractive compare to interest rate so they will discount their bond value. The goal of this simulation is to explore the bond price change with respect to different factors.

2 Pricing Model Summary

The common approach to evaludate fixed coupon bonds is to discount the cashflows according to a risk free interest rate and probability of survival Q(t):

$$V = Pe^{-rt_N} \cdot Q(t_N) + \sum_{i=1}^{N} \frac{C}{f} e^{-rt_i} \cdot Q(t_i) + R \cdot P \sum_{i=1}^{N} e^{-rt_i} [Q(t_{i-1}) - Q(t_i)] \tag{1}$$

where V is the value of the bond, P is the principle amount, C is the annual coupon amount, f is the number of coupon cashflows per year, and t_N denotes the mature date. The first term is the expectation value of buying price when bond matures. Considering a bond with face value P takes t_N years to become mature. The money consider interest rate you should consider to pay satisfies $x \cdot (1+r)^{t_N} = P$. So, the cost is $\frac{P}{(1+r)^{t_N}}$. A simple way to calculate this is by substituting $(1+r)^{-t_N}$ with e^{-rt_N} when r is very small.[2] The factor $Q(t_N)$ is capturing the risk that bonds got fault and all the money get lost. The second term is capturing all the future bond coupon rewards discounting to the presenting time. The third term $Q(t_{i-1}) - Q(t_i)$ denotes the situtation when bonds got defaulted between time t_{i-1} and t_i. We multiply the time discount and recycling rate. The three term captures all the cost of the bonds.

3 Modeling Survival Rate

Let's assume at the very short inteval Δt the probability of bond got default is $\lambda\Delta t$. So the probability that at time $t = N\Delta t$ bond is still sfe:

$$\begin{aligned} P(t > N\Delta t) &= (1 - \lambda\Delta t)^N \\ &= (1 - \lambda\Delta t)^{t/\Delta t} \end{aligned}$$

It's obvious that when $\Delta t \approx 0$, we have $P(S > t) = \exp(-\lambda t)$. So, it's nature to assume that surviving probability is following exponential distribution with unknown λ. We use least square linear regression fitting method with relation $\ln(y) = \lambda x$. The fitted result is showed in FIG. 1. The equation is $y = exp(-0.106t)$. The R^2 of the fitting is 0.997, which indicates the robusty of our theoretical analysis , and validate our assumption.

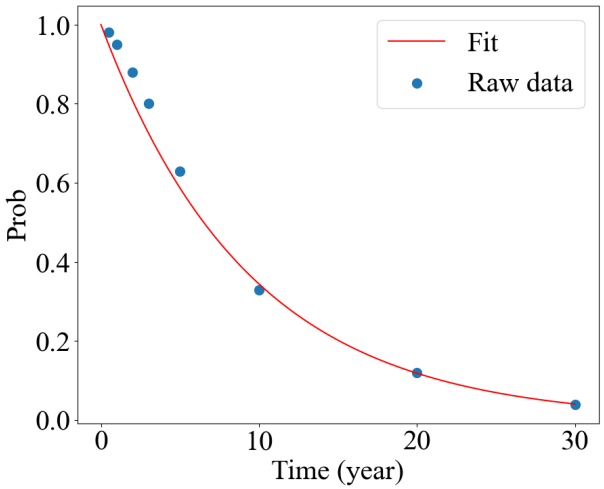


Figure 1: Survival probability as a function of time fitted model and raw data.

4 Q1 Pricing

4.1 a

1- year bond with 3% coupon paid semi-annually($f = 2$). For principle part $P = 100$, the annual paid coupon is $C = 3$. The calculation formula is :

$$V = 100 * exp(-0.03 * 1) * Q(1) + \sum_{i=1}^{2} \frac{3}{2} exp(-0.03 * i/2) Q(i/2)$$

$$+ 0.4 * 100 * \sum_{i=1}^{2} exp(-0.03 * i/2)[Q((i-1)/2) - Q(i/2)]$$

The result is 93.8994. (See code attachement)

4.2 b

5 - year bond with 5% coupon paid semi-annually f = 4. For principle part P = 100, the annual paid coupon is C = 5. The calculation formula is:

$$V = 100 * exp(-0.03 * 5) * Q(5) + \sum_{i=1}^{20} \frac{5}{4} exp(-0.03 * i/4) Q(i/4)$$
$$+ 0.4 * 100 * \sum_{i=1}^{20} exp(-0.03 * i/4)[Q((i-1)/4) - Q(i/4)]$$

The result is 83.7165 (See code attachment)

4.3 c

x - year bond with y% coupon paid frequency. For principle part P = 100, the annual paid coupon is C = y. The calculation formula is:

$$V = 100 * exp(-0.03 * x) * Q(x) + \sum_{i=1}^{xf} \frac{y}{f} exp(-0.03 * i/f) Q(i/f)$$
$$+ 0.4 * 100 * \sum_{i=1}^{xf} exp(-0.03 * i/f)[Q((i-1)/f) - Q(i/f)]$$

(See code attachement) We plot the bond price with respect to coupon rate and bond mature years when pay frequency is at different numbers in Fig. 2. The bond price increase little and insignificant when pay frequency increase.

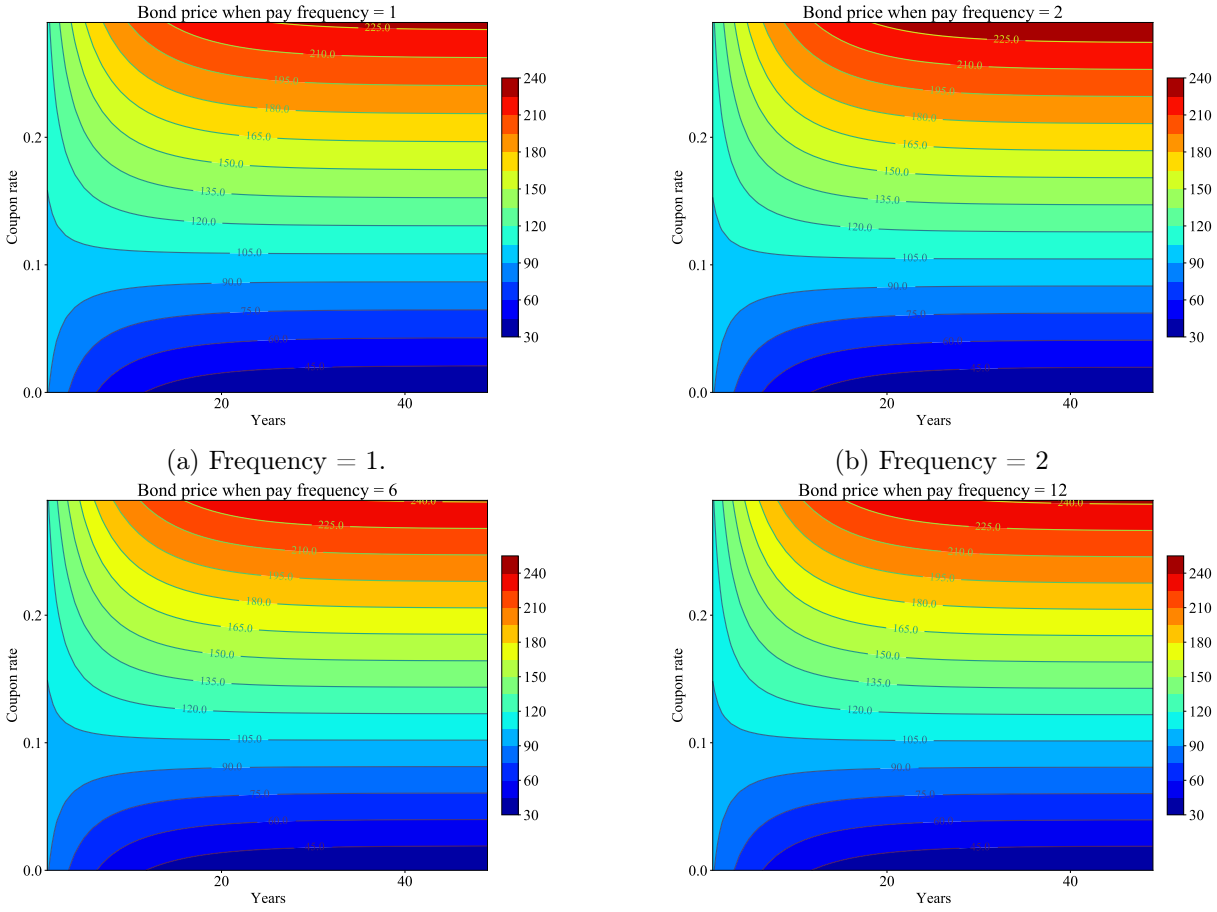
5 Spread Calibration

5.1 a

1 - year bond with a 3% coupon paid semi-annually with market quota = 102 We first plot interest rate vs bond price for this specific bond. The problem now is "Given market quote is 102, how do we inverse solve the interest rate". Here, we use the bisection algorithm.[9] Let's suppose we find the interest rate should lie in the inteval [x1, x2] with $(\mathrm{V(x1)} - \mathrm{M}) * (\mathrm{V(x2)} - \mathrm{M}) < 0$. Our goal is finding the root x with precision ϵ. The recursion relation is that as long as $(\mathrm{V(x1)} - \mathrm{M}) * (\mathrm{V(x2)} - \mathrm{M}) < 0$, there must be a position where $\mathrm{V(x)} - \mathrm{M} = 0$ simply because the continuity of the function. We recursively check if the middle point and edge point satisfying the relation until we find the inteval length that's within the precision. The algorithm is attached as below in Algorithm 1:
The time complexity can be understand from the relation that after each recursion, the inteval length become half of the previous one. To reach the accuracy region, there must be $\mathrm{O}(\log((\mathrm{x2} - \mathrm{x1})/\epsilon))$ recursions. The result we got for "1 - year bond with 3 % coupon paid semi-annually with market quote = 102" is -0.054262. So, the calibrate spread s is -0.084262. Checking with the plot we did first shown in FIG. 3a, the results agree well.

5.2 b

Apply the same logic as the last subsection, "5 - year bond with 5 % coupon paid quarterly with market quote = 98" is -0.00943. So, the calibrate spread s is -0.0394, this result agrees with FIG. 3b



(a) Frequency = 1.

(b) Frequency = 2

(c) Frequency = 6

(d) Frequency = 12

Figure 2: Bond Price as a function of coupon rate and mature years at different pay frequency. When Pay frequency increase, the bond price increase little and insignificant.

6 Sensitivities

There are two ways that we compute the numerical derivatives. First, we can compute this by finite difference method.[10, 11, 12]. We choose this by five point stencil.

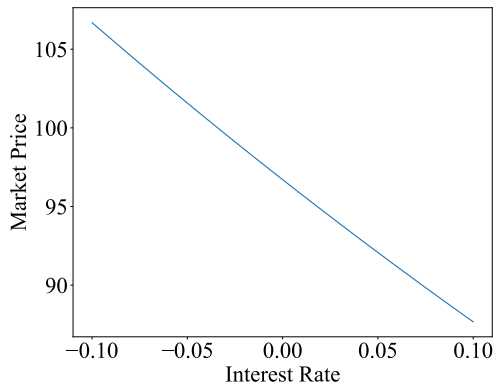
$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} \tag{2}$$

It's easy to prove this truncation error is $O(h^4)$. One may think to obtain high accuracy derivatives, we can simply reduce h. However, this is not the case. Becuase, numerically, we will need to substract two very close number and divide a very small number. The most popular way of doing is by auto-differentiation machine, which is not only numerical stable but also cheap. It has been widely implemented in Machine learning algorithms.[13, 14, 15] The basic idea is that track the operators in the mathematic functions when program is calling the operator. This is the exact numerical derivatives without analytical derivation. We will use both in this section.

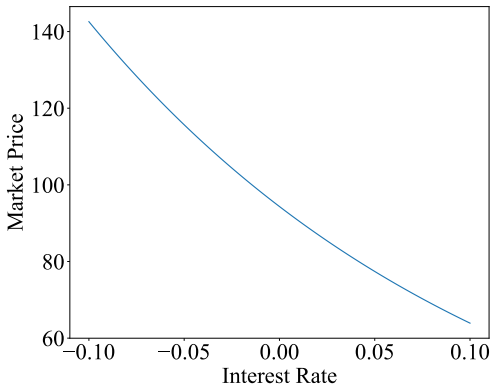
6.1 1a

1- year bond with 3% coupon paid semi-annually(f = 2). For principle part P = 100, the annual paid coupon is C = 3. The sensitivity is listed as following

	numerical	autodiff
bond sensitivity	-92.17677861850385	-92.17677861840514



(a) 1 - year bond with a 3% coupon paid semi-annually changes as an function of interest rate.



(b) 5 - year bond with a 5% coupon paid semi-annually changes as an function of interest rate.

Figure 3: Interest rate influence on different bonds

Algorithm 1 Bisection(x1, x2, Target, ϵ)

Require: $(V(x1) - Target) * (V(x2) - Target) < 0$
  if $abs(x1 - x2) < \epsilon$ **then**
    $Return(x1 + x2)/2$
  else
    middle = (x1 + x2) / 2;
    $dv = V(middle) - Target$;
    $dv1 = V(x1) - Target$
    if $dv * dv1 < 0$ **then**
      return Bisection(x1, middle, Target ,ϵ)
    else
      return Bisection(middle, x2, Target ,ϵ)
    end if
  end if

6.2 1b

5 - year bond with 5% coupon paid semi-annually f = 4. For principle part P = 100, the annual paid coupon is C = 5. The sensitivity is listed as following

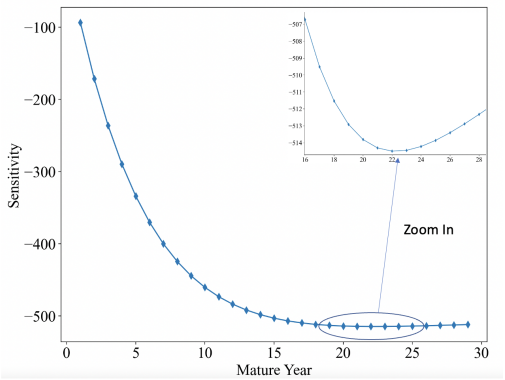
	numerical	autodiff
bond sensitivity	-330.398722946145	-330.39872294609074

6.3 1c

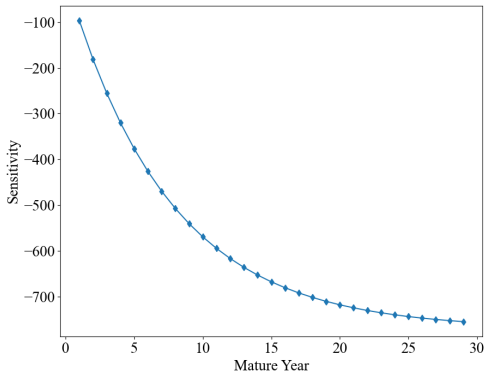
We plot the bond sensitivity with respect to coupon rate and bond mature years when pay frequency is at different numbers in Fig. 5. From the figure 5, generally, the long term bond is more sensitive to interest rate. But, we can see that when coupon rate is low. We can see that long term bond has an maximum sensitivity at finite mature year. When coupon rate is high, we didn't see any instability (or maximum sensitivity)

6.4 b

We use the same logic and plot the bond- sensitivity with regarding to mature time. From the figure 4, we found that long term bonds are more sensitive to interest rate change. But for lower



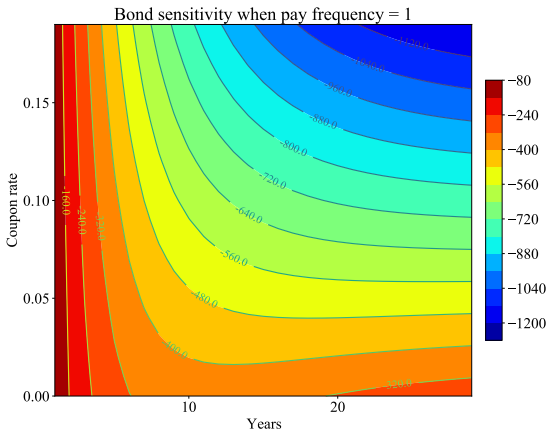
(a) Coupon = 0.05.



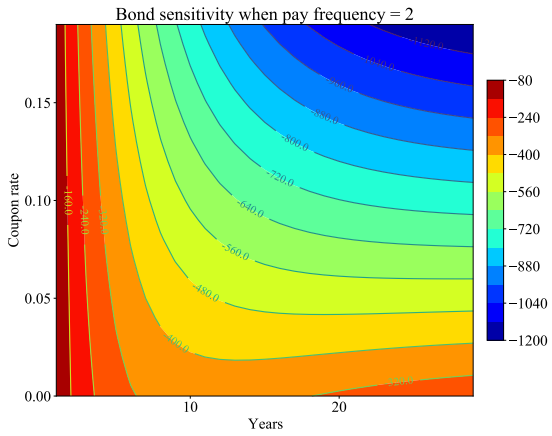
(b) Coupon = 0.10

Figure 4: long term bonds are more sensitive to interest rate change. but under small coupon rate, there will be a maximum sensitivity versus marture time

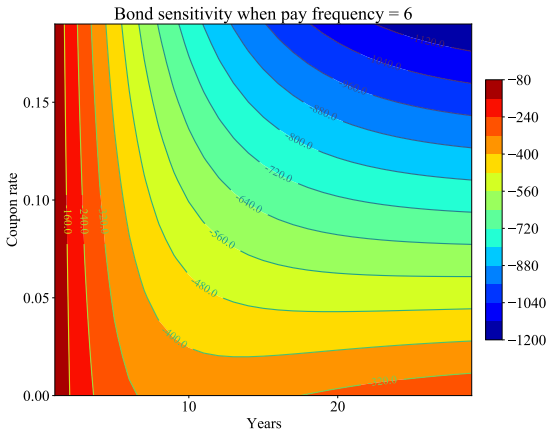
coupon rate, it could have unstable behavior (maximum sensitivity at finite mature time). For higher coupon rate, it simply increase as a function of mature time[16]



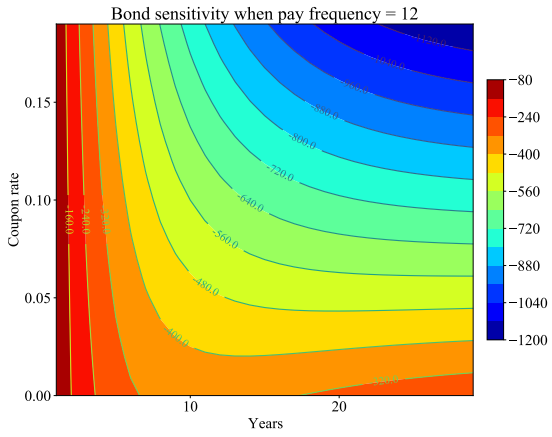
(a) Frequency = 1.



(b) Frequency = 2



(c) Frequency = 6



(d) Frequency = 12

Figure 5: Bond Price Sensitivity as a function of coupon rate and mature years at different pay frequency. From the figure, we can see that when coupon rate is low. We see that long term bond has a maximum sensitivity at finite mature year. When coupon rate is high, we didn't see any instability (or maximum sensitivity)

7 Appendix

7.1 Basic Class of Bond

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Nov 11 16:05:09 2022

@author: jiahaozhang
"""

import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from sklearn.linear_model import LinearRegression
font={'size':30, 'family':'Times New Roman'};
plt.rcParams['figure.figsize']=(10, 8);
matplotlib.rc('font', **font);
```

```
class Bond:
    def __init__(self, year, fn, principle, InterestR, Recovery):
        self.year = year; # How many years are there for the payment exist;
        self.fn = fn; # How many times each year pays the Coupon.
        self.principle = principle; # Principle part of the Bond
        self.InterestR = InterestR; # interest Rate in the capital market.
        self.Recovery = Recovery; # Recovery Rate when bond got defaulted.
        self.Paytimes = int( self.year * self.fn ); # How many times it should pa
        self.paygrid = np.arange(1, self.Paytimes + 1, 1) / self.fn; # the time g

    def update_InterestR(self, r):
        self.InterestR = r;

    def paycoupon(self, AnnualC):
        self.AnnualC = AnnualC;

    def calculateV(self, IR = None):
        if IR is None:
            IR = self.InterestR;
        Qgrid = self.predictQ(self.paygrid);
        term1 = self.principle * np.exp( -1 * IR * self.year) * Qgrid[-1];
        term2 = self.AnnualC / self.fn * np.exp( -1 * IR * self.paygrid ) * Qgrid
        term2 = np.sum(term2);
        Qi_1 = np.array( [1] + list(Qgrid[0 : (self.Paytimes - 1)] ) );
        term3 = np.exp(-1 * IR * self.paygrid) * (Qi_1 - Qgrid);
        term3 = np.sum(term3) * self.Recovery * self.principle;
        return term1 + term2 + term3;

    def loadQ(self, filename):
        self.Q = np.loadtxt(filename);
        xgrid = self.Q[:, 0];
        ygrid = self.Q[:, 1];
        lnxgrid = xgrid;
        lnygrid = np.log(ygrid);
        length = len(ygrid);
        regx = np.reshape( lnxgrid, (length, 1) );
        regy = np.reshape( lnygrid, (length, 1) );
        self.regQ = LinearRegression(fit_intercept = False).fit(regx, regy);
        self.r_squared = self.regQ.score(regx, regy)

    def predictQ(self, year):
        yearinput = np.array(year).flatten();
        yearinput = np.array(year).reshape((len(yearinput), 1));
        ygrid = self.regQ.predict( yearinput );
        return np.exp( ygrid.flatten() );

    def plotQ(self):
        plt.scatter(self.Q[:, 0], self.Q[:, 1], marker='o', s = 100, label = "Raw
        xgrid_p = np.linspace(0, np.max(self.Q[:, 0]), 100);
        ygrid_p = self.predictQ(xgrid_p);
        plt.plot(xgrid_p, ygrid_p, '-r', label = 'Fit')
```

```
        plt.legend()
        plt.xlabel("Time (year)")
        plt.ylabel("Prob")
if __name__=='__main__':
    bond1 = Bond(1, 2, 100, 0.03, 0.4);
    bond1.loadQ("./Qdat.dat")
    #bond1.plotQ();
    bond1.paycoupon( bond1.principle * 0.03 );
    re1 = bond1.calculateV();
    print(re1)
    bond5 = Bond(5, 4, 100, 0.03, 0.4);
    bond5.loadQ("./Qdat.dat")
    bond5.paycoupon( bond5.principle * 0.05 );
    re2 = bond5.calculateV();
    print(re2)
    bond5 = Bond(5, 12, 100, 0.03, 0.4);
    bond5.loadQ("./Qdat.dat")
    bond5.paycoupon( bond5.principle * 0.05 );
    re2 = bond5.calculateV();
    print(re2)
```

7.2 Calibration s

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Nov 11 21:57:10 2022

@author: jiahaozhang
"""
from bond import Bond
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from sklearn.linear_model import LinearRegression
font={'size':30, 'family':'Times New Roman'};
plt.rcParams['figure.figsize']=(10, 8);
matplotlib.rc('font', **font);
plt.subplots_adjust(left=0.18, right=0.95, top=0.95, bottom=0.18);
class calibration(Bond):
    def plotr(self):
        xgrid = np.linspace(-0.1, 0.1, 100);
        ygrid = np.zeros(len(xgrid));
        for i in range(len(xgrid)):
            self.update_InterestR(xgrid[i]);
            ygrid[i] = self.calculateV();
        plt.plot(xgrid, ygrid)
        plt.xlabel("Interest Rate")
        plt.ylabel("Market Price")

    def bisect(self, xleft, xright, target, precision):
```

```
        self.update_InterestR(xleft);
        dvl = self.calculateV() - target;
        self.update_InterestR(xright);
        dvr = self.calculateV() - target;
        if dvl * dvr > 0:
            print("Bad Initial Guess");
            return np.inf;
        def bi(x1, x2):
            nonlocal target;
            middle = (x1 + x2) / 2;
            if np.abs(x1 - x2) < precision:
                return (x1 + x2) / 2;
            self.update_InterestR(middle);
            dvm = self.calculateV() - target;
            self.update_InterestR(x1);
            dvl = self.calculateV() - target;
            if dvm * dvl < 0:
                return bi(x1, middle);
            else:
                return bi(middle, x2);
        return bi(xleft, xright);
if __name__ == "__main__":
#     cal = calibration(1, 2, 100, 0.03, 0.4);
#     cal.loadQ("./Qdat.dat");
#     cal.paycoupon(3)
#     cal.plotr();
#     print(cal.bisect(-1, 1, 102, 1e-6))
#     plt.show()
    cal = calibration(5, 4, 100, 0.03, 0.4);
    cal.loadQ("./Qdat.dat");
    cal.paycoupon(5)
    cal.plotr();
    print(cal.bisect(-1, 1, 98, 1e-6))
    plt.show()
```

7.3 2D plot

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 12 10:03:56 2022

@author: jiahaozhang
"""
from bond import Bond
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import cm
from sklearn.linear_model import LinearRegression
font={'size':20, 'family':'Times New Roman'};
```

```
plt.rcParams['figure.figsize']=(10, 8);
matplotlib.rc('font', **font);
fn = 1;
xgrid = np.arange(1, 50);
ygrid = np.arange(0, 0.3, 0.01);
X, Y = np.meshgrid(xgrid, ygrid);
shape = np.shape(X);
Vprice = np.zeros(shape);
for x in range(shape[0]):
    for y in range(shape[1]):
        year = X[x, y];
        coupon = Y[x, y];
        bd = Bond(year, fn, 100, 0.03, 0.4);
        bd.paycoupon( bd.principle * coupon );
        bd.loadQ("./Qdat.dat");
        Vprice[x, y] = bd.calculateV();
fig, ax = plt.subplots(constrained_layout=True);
cf = ax.contourf(X, Y, Vprice, cmap = cm.jet, levels = 15)
cs = ax.contour(X, Y, Vprice, levels = 15)
ax.clabel(cs, inline = 1, fontsize = 15, fmt = "%5.1f")
ax.set_xlabel("Years")
ax.set_ylabel("Coupon rate")
fig.colorbar(cf, ax=ax, shrink=0.7)
ax.set_title("Bond price when pay frequency = {0:d}".format(fn))
ax.locator_params(nbins=4)
```

7.4 Bond Sensitivity

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 12 11:22:50 2022

@author: jiahaozhang
"""
import matplotlib.pyplot as plt
import matplotlib
from bond import Bond
from sklearn.linear_model import LinearRegression
import autograd.numpy as np
from autograd import grad
from autograd import elementwise_grad as egrad
font={'size':30, 'family':'Times New Roman'};
plt.rcParams['figure.figsize']=(10, 8);
matplotlib.rc('font', **font);
class bondsensive(Bond):
    def fivestencil(self, x, h= 0.0001):
        fx_2h = self.calculateV(x - 2*h);
        fx_h = self.calculateV(x - h);
        fxph = self.calculateV(x + h);
        fxp2h = self.calculateV(x + 2*h);
```

```
        return (-1*fxp2h + 8 * fxph - 8 * fx_h + fx_2h) / 12 / h;

    def calculateV(self, IR):
        Qgrid = self.predictQ(self.paygrid);
        term1 = self.principle * np.exp( -1 * IR * self.year) * Qgrid[-1];
        term2 = self.AnnualC / self.fn * np.exp( -1 * IR * self.paygrid ) * Qgrid
        term2 = np.sum(term2);
        Qi_1 = np.array( [1] + list(Qgrid[0 : (self.Paytimes - 1)] ) );
        term3 = np.exp(-1 * IR * self.paygrid) * (Qi_1 - Qgrid);
        term3 = np.sum(term3) * self.Recovery * self.principle;
        return term1 + term2 + term3;

    def automaticgrad(self, x):
        gradV = egrad(self.calculateV);
        return gradV(x);

if __name__ == "__main__":
    bond1 = bondsensive(1, 2, 100, 0.03, 0.4);
    bond1.loadQ("./Qdat.dat");
    bond1.paycoupon(bond1.principle * 0.03)
    print(bond1.fivestencil(0.03))
    print(bond1.automaticgrad(0.03))
    bond5 = bondsensive(5, 4, 100, 0.03, 0.4);
    bond5.loadQ("./Qdat.dat")
    bond5.paycoupon( bond5.principle * 0.05 );
    print(bond5.fivestencil(0.03));
    print(bond5.automaticgrad(0.03));
```

7.5 2D sensitivity

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 12 12:39:26 2022

@author: jiahaozhang
"""

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 12 10:03:56 2022

@author: jiahaozhang
"""
from bondsensitive import bondsensive
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import cm
from sklearn.linear_model import LinearRegression
```

```
font={'size':20, 'family':'Times New Roman'};
plt.rcParams['figure.figsize']=(10, 8);
matplotlib.rc('font', **font);
fn = 12;
xgrid = np.arange(1, 20);
ygrid = np.arange(0, 0.2, 0.01);
X, Y = np.meshgrid(xgrid, ygrid);
shape = np.shape(X);
Vprice = np.zeros(shape);
for x in range(shape[0]):
    for y in range(shape[1]):
        year = X[x, y];
        coupon = Y[x, y];
        bd = bondsensive(year, fn, 100, 0.03, 0.4);
        bd.paycoupon( bd.principle * coupon );
        bd.loadQ("./Qdat.dat");
        Vprice[x, y] = bd.automaticgrad( 0.03 );
fig, ax = plt.subplots(constrained_layout=True);
cf = ax.contourf(X, Y, Vprice, cmap = cm.jet, levels = 15)
cs = ax.contour(X, Y, Vprice, levels = 15)
ax.clabel(cs, inline = 1, fontsize = 15, fmt = "%5.1f")
ax.set_xlabel("Years")
ax.set_ylabel("Coupon rate")
fig.colorbar(cf, ax=ax, shrink=0.7)
ax.set_title("Bond sensitivity when pay frequency = {0:d}".format(fn))
ax.locator_params(nbins=4)
plt.show();
```

7.6 bond sensitivity 1D Q(3.2)

```
"""
Created on Sat Nov 12 12:39:26 2022

@author: jiahaozhang
"""


#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 12 10:03:56 2022

@author: jiahaozhang
"""
from bondsensitive import bondsensive
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
from matplotlib import cm
from sklearn.linear_model import LinearRegression
font={'size':20, 'family':'Times New Roman'};
plt.rcParams['figure.figsize']=(10, 8);
```

```
matplotlib.rc('font', **font);
fn = 4;
coupon = 0.05;
xgrid = np.arange(1, 50);
sensprice = np.zeros(len(xgrid));
for i in range(len(xgrid)):
    bd = bondsensive(xgrid[i], fn, 100, 0.03, 0.4);
    bd.paycoupon( bd.principle * coupon );
    bd.loadQ("./Qdat.dat");
    sensprice[i] = bd.automaticgrad( 0.03 );
plt.plot(xgrid, sensprice,'-d')
plt.xlabel("Mature Year")
plt.ylabel("Sensitivity")
plt.show();
```

References

[1] John D Finnerty. Financial engineering in corporate finance: An overview. *Financial management*, pages 14–33, 1988.

[2] Jieyi Chen. An overview of bond pricing models and duration of bonds. In *2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*, pages 2316–2320. Atlantis Press, 2022.

[3] Young Ho Eom, Jean Helwege, and Jing-zhi Huang. Structural models of corporate bond pricing: An empirical analysis. *The Review of Financial Studies*, 17(2):499–544, 2004.

[4] Jonathan A Batten, Karren Lee-Hwei Khaw, and Martin R Young. Convertible bond pricing models. *Journal of Economic Surveys*, 28(5):775–803, 2014.

[5] JASON FERNANDO. Bond: Financial meaning with examples and how they are priced. *Investopedia*, 2022.

[6] Guohe Deng. Pricing american put option on zero-coupon bond in a jump-extended cir model. *Communications in Nonlinear Science and Numerical Simulation*, 22(1-3):186–196, 2015.

[7] Burton G Malkiel. Expectations, bond prices, and the term structure of interest rates. *The Quarterly Journal of Economics*, pages 197–218, 1962.

[8] Terry A Marsh and Eric R Rosenfeld. Stochastic processes for interest rates and equilibrium bond prices. *The Journal of Finance*, 38(2):635–646, 1983.

[9] JC Ehiwario and SO Aghamie. Comparative study of bisection, newton-raphson and secant methods of root-finding problems. *IOSR Journal of Engineering*, 4(4):01–07, 2014.

[10] DAVIDW Zingg and D Zingg. A review of high-order and optimized finite-difference methods for simulating linear wave phenomena. In *13th Computational Fluid Dynamics Conference*, page 2088, 1997.

[11] Khaled Rouabah, Salim Atia, Mustapha Flissi, Smail Medjdoub, and Djamel Chikouche. Gnss multipath mitigation using finitedifference derivatives with five-point stencil. In *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, pages 690–695. IEEE, 2012.

[12] Wikipedia contributors. Five-point stencil — Wikipedia, the free encyclopedia, 2022. [Online; accessed 12-November-2022].

[13] Dougal Maclaurin. *Modeling, inference and optimization with composable differentiable procedures*. PhD thesis, 2016.

[14] Jerome Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. *Advances in Neural Information Processing Systems*, 33:10809–10819, 2020.

[15] Atilim Gunes Baydin and Barak A Pearlmutter. Automatic differentiation of algorithms for machine learning. *arXiv preprint arXiv:1404.7456*, 2014.

[16] César Villazón. Bond duration, yield to maturity and bifurcation analysis. In *Second AFIR Colloquium. At www. actuaries. org/AFIR/Colloquia/Brighton/Villazon. pdf*, 1991.