5-2015

Perpetual Option Pricing Revision of the NPV Rule, Application in C++

Andy Ferguson
Utah State University

PERPETUAL OPTION PRICING

REVISION OF THE NPV RULE, APPLICATION IN C++

by

Andrew Ferguson

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Financial Economics

Approved:

Tyler Brough Major Professor	Jason Smith Committee Member

Alan Stephenson
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah

2015

ABSTRACT

Perpetual Option Pricing

Revision of The NPV Rule, Application in C++

by

Andy Ferguson, Master of Science

Utah State University, 2015

Major Professor: Tyler Brough

Department: Finance and Economics

The typical NPV rule lacks the option embedded value of taking on the project in question. The time in which we take on the project is this embedded option. I present the methodology for examples used in the Perpetual Option Pricing Program which are presented by Robert McDonald in his book, "Derivatives Markets". Refer to chapter 17.

CONTENTS

ABSTRACT ii

INTRODUCTION 1

NPV RULE AND INVESTMENT UNDER CERTAINTY 2

A. Static NPV 2

i. Static NPV Example 4

B. The Correct Use of NPV 5

C. The Project as an Option: Perpetual Option 5

i. The Project as an Option: Perpetual Option Example 7

VALUING PERPETUAL AMERICAN OPTIONS 8

A. Valuing Perpetual Options 8

B. Barrier Present Values 10

PERPETUAL OPTION PRICING PROGRAM 11

INVESTMENT UNDER UNCERTAINTY 14

A. A Simple DCF Problem 14

i. Simple DCF Example 16

B. Valuing Derivatives of Cash Flow 17

i. Valuing Derivatives on the Cash Flow Example 19

C. Evaluating a Project with a 2-Year Investment Horizon 20

D. Evaluating the Project with an Infinite Investment Horizon 23

i. Evaluating the Project with an Infinite Investment Horizon Example 24

COMMODITY EXTRACTION AS AN OPTION 25

A. SINGLE BARREL EXTRACTION UNDER CERTAINTY 25

B. Optimal extraction 25

C. Value and Appreciation of the Land 26

i. DCF Single Barrel Under Certainty Example 28

ii. Option Pricing Single Barrel Under Certainty Example 29

D. SINGLE BARREL EXTRACTION UNDER UNCERTAINTY 30

i. Single Barrel Extraction under Uncertainty Example 32

ii. Perpetual Option Example 33
E. VALUING AN INFINITE OIL RESERVE 35
i. Value of Producing Firm 35
ii. Value of the Option to Invest 36
iii. Value of the Producing Well 36
iii..1 Perpetual Option Example $\sigma = 0.000001$ 38
iii..2 Perpetual Option Example $\sigma = 0.15$ 39
COMMODITY EXTRACTION WITH SHUTDOWN AND RESTART OPTIONS 40
A. Initial Investment in the Well 40
i. Continue to produce 40
ii. Restart the Operating Well 40
iii. Permanent Shutdown 41
iii..1 Permanent Shutdown Example $k_s = \$0$ 43
iii..2 Permanent Shutdown Example $k_s = -\$25$ 45
B. The Value of the Producing Well 47
C. Investing when Shutdown is Possible 48
i. Value of Producing Well and Investing when Shutdown is Possible Example 50
D. Restarting Production 52
i. Restart Shutdown Well Example $k_r = \$0$ 53
CONCLUSIONS 55
REFERENCES 55
APPENDIX - PERPETUAL OPTION PRICING PROGRAM 57

INTRODUCTION

The typical net present value, NPV, is simple, easy to use, and works well for identifying the value of a project. But on the users side it can be difficult to find the project that *exceeds the NPV of all mutually exclusive alternative projects.* The mutual exclusivity consideration is usually thought of in terms of several projects of differing fundamental characteristics but not typically thought of in terms a multiple time varying project at each point in time. Projects with NPV that vary over time might see significant changes in NPV over time and would cause the decision maker to want to delay the project to receive this higher, more optimized NPV. This longer view can be more difficult to solve for since the calculation can get large and take much more time. Perpetual options do well at optimization and can tie out to the traditional NPV but reduces the problem to inputting the needed values into a simple function. To make the calculations simpler this function can be written into a computer program that executes the analytical solution; The Perpetual Option Pricing Program does such a thing. It is built using some unique aspects of computer programming and specifically C++ methods. The program delivers the needed answer with a few commands and some input values which depend on the type of real option problem. Since real options are not standard while the commands in the program are standard, the discretion of the user is required. Several example were built corresponding to the different uses of perpetual options based on Chapter 17 on real options in Robert McDonald's book, Derivative Markets.

NPV RULE AND INVESTMENT UNDER CERTAINTY

A. *Static NPV*

Starting with the NPV for a now-or-never project that is started today, the value of the project can be summarized in the following equation:[1]

$$\begin{aligned} & Revenue_{per\ unit} * \left(\frac{1}{(1+r)} + \frac{(1+\delta)}{(1+r)^2} + \frac{(1+\delta)^2}{(1+r)^3} + ... \right) \\ & -Cost_{per\ unit} * \left(\frac{1}{(1+r)} + \frac{1}{(1+r)^2} + \frac{1}{(1+r)^3} + ... \right) - I \\ & \frac{Revenue_{per\ unit}}{(1+\delta)} * \frac{1}{\frac{(1+r)}{(1+\delta)} - 1} - \frac{Cost_{per\ unit}}{r} - I \\ & \frac{Revenue_{per\ unit}}{(1+r)-(1+\delta)} - \frac{Cost_{per\ unit}}{r} - I \end{aligned}$$

This is the NPV if invested today in this project. If investment into the project can be made in the future then the following equation would be used:

$$NPV_{In\ n\ years} = \frac{1}{(1+r)^n} \left[(1+\delta)^n \frac{Revenue_{per\ unit}}{(1+r)-(1+\delta)} - \frac{Cost_{per\ unit}}{r} - I \right] \qquad (2.1)$$

By maximizing the equation 2.1 by using a simple benchmark that when achieved should trigger investment. Investment is ideal when revenues from production can cover the opportunity cost of the project plus the marginal cost of production. This simple benchmark is used; (the sum of cash flows lost from not taking on the project) or the opportunity cost of the project and the marginal cost of production. Using this combined benchmark of opportunity cost and marginal cost, the optimal NPV for investment with respect to time(n) can be found.

[1] McDonald, Derivative Markets Chapter 17, starting at page 510

$$(1+\delta)^n * Revenue_{per\ unit} = Marginal\ cost + Opportunity\ cost \tag{2.2}$$

solving for n

$$ln((Marginal\ cost + Opportunity\ cost)/Revenue_{per\ unit})/ln(1+\delta) = n\ years \tag{2.3}$$

This optimal time (n) gives us the break even for investment given our assumptions. Given we have the optimal time (n) now the NPV can be found.

The value of the project waiting (n) years is found by plugging these values into equation(2.1)

i. Static NPV Example

Given the following numbers, the static NPV for a now-or-never project is calculated

$$Revenue_{per\ unit} = \$0.55, Costs_{per\ unit} = \$0.90, r = 0.05, \delta = 0.04, Cost_{Initial} = \$10$$

$$\begin{aligned} &\$0.55 * \left(\frac{1}{1.05} + \frac{1.04}{1.05^2} + \frac{1.04^2}{1.05^3} + ...\right) \\ -&\$0.90 * \left(\frac{1}{1.05} + \frac{1}{1.05^2} + \frac{1}{1.05^3} + ...\right) - \$10 \\ &\frac{\$0.55}{1.04} * \left(\frac{1}{\frac{1.05}{1.04} - 1}\right) - \frac{\$0.90}{0.05} - \$10 \\ &\frac{\$0.55}{0.01} - \$28 = \mathbf{\$27} \end{aligned}$$

Using the price evolution and equating it to the sum of marginal cost of production and annual interest saved and then solving for n:

$$(1.04)^n * \$0.55 = \$0.90 + \$0.50 = \mathbf{\$1.40}$$

$$ln(\mathbf{\$1.40}/0.55)/ln(1.04) = n = \mathbf{23.82}\ years$$

The NPV of waiting n years is

$$NPV_{Wait\ n\ years} = \frac{1}{1.05^{\mathbf{23.82}}}\left[(1.04)^{\mathbf{23.82}}\frac{\$0.55}{0.01} - \frac{\$0.90}{0.05} - \$10\right] = \mathbf{\$35.03}$$

B. The Correct Use of NPV

The static NPV rule requires that a project is accepted if and only if its NPV is positive and *it exceeds the NPV of all mutually exclusive alternative projects.* The NPV of a project can change depending on the period of time in which it can be executed. If the project is under assumptions of executing on a now-or-forever basis then there may only be a few possible NPV where the decisions are decided today. When the decision to invest can be delayed or decisions to continue to invest in a project are on going then there is an NPV for each period in time and at each decision point. Finding the maximum NPV is then a matter of finding the optimal time NPV for each fixed scenario as demonstrated above or will be done here in using payoffs to model the option which becomes part of the expected cash flows that are modelled once and then optimized for time to exercise.

C. The Project as an Option: Perpetual Option

The decision to invest or delay is dependent on the value of delay. The decision to invest in the project involves a comparison of present values of cash in-flows and out-flows. Comparing the revenue and cost present values in equation (1), the flows are such:

$$Costs_{PV} = Cost_{Initial} + \frac{Cost_{per\ unit}}{r} \tag{2.4}$$

$$Revenue_{PV} = \frac{S_{+1}}{r - d} \tag{2.5}$$

These are analogous to the cash flows that are paid and the cash flows that are received for exercising an option, respectively. S_+1 is the per unit price the year after the investment is made. The present value of revenue is the twin security and is analogous to the stock price.

Early exercise of a call option is dependent on

1) Dividends foregone by not receiving the asset,

2) Interest saved delaying the payment of the strike, and the

3) Value of insurance lost by exercising the option. Given early exercise conditions for call options, the exercise or decision to invest in a project, can be assessed.

By delaying the investment decision, the cash flows from that project are not realized. This cash flow is analogous to receiving dividends from holding the stock. The first period cash flow is S_{+1} and the dividend yield is approximated by $(r - d)$, 0.01 as seen above.

By exercising the option the present value of costs will be paid. This includes the marginal cost of production/extraction plus the initial investment cost. Thus the annual value of delaying the investment is the interest saved on total investment, or $r * Costs_{PV} = InterestSavings$, $0.05 * \$28 = \$1.40$.

Value of insurance is dependent on whether there is uncertainty in terms of the price of the asset. When price certainty exists then there is no value of insurance, the price is known in the future and there is no need to price in the uncertainty into the option. When uncertainty exists then the option price will include this uncertainty regarding the future price of the asset.

To summarize, the spot price is the present value of the revenue from the project, the strike is the present value of costs for the project.

Using a perpetual call option, the optimal price for production can be found.

i. The Project as an Option: Perpetual Option Example

Using the general function for perpetual options:

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\} \tag{2.6}$$

but specifically to this problem,

$$CallPerpetual\left[\frac{S_{+1}}{r-\delta}, \frac{c}{r} + I, \sigma, r, \delta\right] = \{value, price\} \tag{2.7}$$

$$\begin{aligned}
S &= \frac{S_{+1}}{r-\delta} = \frac{\$0.55}{0.05 - 0.04} = \mathbf{\$55} \\
X &= I + \frac{Costs_{per_unit}}{r} = \$10 + \frac{\$0.90}{0.05} = \mathbf{\$28} \\
\sigma &= 0.00001 \\
r &= ln(1+r) = ln(1.05) = 0.04879 \\
\delta &= ln(1+r) - ln(1+\delta) \\
&= ln(1.04879) - ln(1.03922) = 0.009569
\end{aligned}$$

Perpetual option general function:

$$CallPerpetual[\mathbf{\$55}, \mathbf{\$28}, 0.00001, 0.04879, 0.09569] = \{price = \mathbf{\$35.028}\}$$

Perpetual option program:

PerpetualPayOffCall thePayOff [**$28**, 0.00001, 0.04879, 0.09569]

VanillaOption theOption(thePayOff)

PerpetualOptionPrice (theOption, **$55**) = *perpetualOptionPrice*

VALUING PERPETUAL AMERICAN OPTIONS

Perpetual options or "expirationless options" formulas presented here are based on Merton (1973b) and describe the price of options that do not expire which contrasts American options which have an finite time to expiration. This time to expiration is constant for each perpetual option which causes the optimal exercise price to be the constant through time. The optimal exercise strategy then is reduced to choosing the exercise price limit that maximizes the value of the option and then exercising the option when that limit is reached.

A. *Valuing Perpetual Options*

[2]

From the the Black-Scholes partial differential equation with dividends (δ) but without regard to time

$$V(S) = 0.5\sigma^2 S^2 V_{ss} + (r - \sigma)SV_s - rV = 0 \tag{3.1}$$

The solution being

$V(S) = AS^{h_1} + BS^{h_2}$

where A,B are constants

$$h_1 = \frac{1}{\sigma^2}\left[-\left(r - \delta - \frac{\sigma^2}{2}\right) + \left(\left(r - \delta - \frac{\sigma^2}{2}\right) + 2r\sigma^2\right)^{0.5}\right] \tag{3.2}$$

and

$$h_1 = \frac{1}{\sigma^2}\left[-\left(r - \delta - \frac{\sigma^2}{2}\right) - \left(\left(r - \delta - \frac{\sigma^2}{2}\right) + 2r\sigma^2\right)^{0.5}\right] \tag{3.3}$$

[2]From Paul Wilmott's book *Paul Wilmott on Quantitative Finance* and Nathan Whitehead's youtube video explain Paul Wilmott's text

which can be manipulated to give

$$h_1 = 0.5 - \frac{r-\delta}{\sigma^2} + \left(\left(\frac{r-\delta}{\sigma^2} - 0.5\right)^2 + \frac{2r}{\sigma^2}\right)^{0.5} \tag{3.4}$$

The other quadratic choice

$$h_2 = 0.5 - \frac{r-\delta}{\sigma^2} - \left(\left(\frac{r-\delta}{\sigma^2} - 0.5\right)^2 + \frac{2r}{\sigma^2}\right)^{0.5} \tag{3.5}$$

The value of a perpetual American call with strike price K that is exercised when $S \geq H_c$, H_c being the optimal price for a perpetual call, is

$$(H_c - K)\left(\frac{S}{H_c}\right)^{h1} \tag{3.6}$$

where H_c is calculated as

$$H_c = K\left(\frac{h_1}{h_1 - 1}\right) \tag{3.7}$$

Note that if $\delta = 0$ then $H_c = \infty$; exercising a call option on a non-dividend-paying- stock is never optimal.

The value of a perpetual American put with strike price K that is exercised when $SleqH_p$ is

$$(K - H_p)\left(\frac{S}{H_p}\right)^{h2} \tag{3.8}$$

where H_p is calculated as

$$H_p = K\left(\frac{h_2}{h_2 - 1}\right) \tag{3.9}$$

The value of each option depends on the optimal exercise price, H, that maximizes the value of the option.

B. Barrier Present Values

The value at time 0, of \$1 received when the stock price, S, reaches H from below is

$$\left(\frac{S_0}{H}\right)^{h1} \tag{3.10}$$

The value at time 0, of \$1 received when the stock price, S, reaches H from above is

$$\left(\frac{S_0}{H}\right)^{h2} \tag{3.11}$$

These are the barrier present values used get the present values of perpetual American options.

PERPETUAL OPTION PRICING PROGRAM

Overall Design

Uses the C++ computer language with its standard template library and features some components of object oriented programming and one instance of a programming design pattern.[3] *VanillaOption* class features the use of the rule of three a virtual copy constructor was used along with, virtual destructor, overloaded assignment operator. The virtual copy constructor clone pointed to *PayOff* object so *VanillaOption* has own copy of object but does not know details about the payoff. *VanillaPayOff* class features the use of inheritance with several classes inheriting from it. *VanillaPayOff* actually defines an interface. The class is also polymorphic in that it stores a pointer to the base *PayOff* class that points to inherited object. Therefore it can create a new payoff and use those new payoff methods in main function without re-writing code. These payoffs will be recognized and the appropriate payoff will be executed. *PayOffBridge* class is a bridge programming design pattern that takes care of memory management and stores pointer to option payoff. It helps to separate the implementation from *PayOff* interface and helps to facilitate several more payoffs with regard to changes in the interface.

***VanillaOption* class** has a constructor that takes in the the payoff of type *PayOffBridge* but will also accept the argument type of *PayOffCall*. Because there is a constructor for *PayOffBridge* which takes in an object of type *PayOff*. Compiler accepts the inherited class object as a substitute for the base class and then converts it into the *PayOffBridge* object which is then passed to the *VanillaOption* constructor. *VanillaOptions* does not know the type of the payoff object or about its inherited classes. But the object knows its own type so the object can make a copy of itself

[3]Code was used that accompanied Joshi's book *C++ Design and Derivatives Pricing, second edition*, Where the code was modified or used directly is indicated under each relevant piece of code in the APPENDIX

which *VanillOption* will store. Thus a virtual copy constructor is used; by defining a virtual method of the base class, where the object creates a copy of itself and returns a pointer to the copy. This is done by using the method clone() and attaching it to the *PayOff* pointer in a pure virtual function. In each inherited class it is defined as a call to the copy constructor of *PayOffCall* or *PayOff* Put with a return that is a pointer to the base class object, clone *PayOff**.

***VanillaPayOff* class** defines an interface. Contains pure virtual functions, function pointers, in order to allow for inheritance. Pure virtual function do not need to be defined in the base class and instead the interface (base class) must be defined in the inherited class. Contains the *PayOff*, *PayOffCall*, and *PayOffPut* classes. The latter two inheriting from the former all of its member methods and data members. Is polymorphic since it can make copies of *PayOff* objects of unknown types. *PayOff* is polymorphic in that it does not know the type of payoff but the payoff itself supplies the information needed to apply the correct payoff. The correct payoff therefore, is accepted where the base class is accepted. Payoffs of unknown type can be added and by writing:

*PayOff** *PayOffCall*::clone() const

{

return new *PayOffCall*(*this);

}

will allow each type to use the interface defined by its own classes constructor and overloaded () operator to the base class.

***PayOffBridge* class** stores a pointer to a no option pay-off and takes care of memory handling for *VanillaPayOff* class. Allows *VanillaOption* class to be coded as a an ordinary object requiring no special treatment in regards to needing to satisfy rule of three: handle assignment, construction, and destruction.

***PerpetualPayOff* class** contains two classes, *PerpetualPayOffClass* and *PerpetualPayOffCall* that inherit from the *PayOffClass* and are virtually copyable. Where the base class receives virtual copies of the objects that the objects themselves provide. The *PerpetualPayOffCall* receives the needed parameters through the constructor and saves them in the object data members and waits from the spot to be passed into the

***PerpetualOptionPricer* class** takes in a referenced *VanillaOption* object and a spot price and sends back the result of the spot price being sent to the *OptionPayOff* method of *VanillaOption* class. Then the *BridgePayOff* takes care the implementation while the spot is applied to the proper payoff, *PerpetualPayOffCall* or *PerpetualPayOffPut* and returns a pointer to the object back to the *OptionPayOff* which returns the calculated perpetual option price. *PerpetualOptionPricer* is minimal but can be extended to use an optimization function in order to solve for project value where perpetual put and call options are nested options within project value problem.

AnalystPackage file The AnalystPackage file contains the optimizing *HFinder()* function along with its input functions, *hFinderCall()* and *hFinderPut()* which are also included in the *PerpetualPayOffCall* and *PerpetualPayOffPut* classes but are held in AnalystPackage for further use in solving for Barrier option payoffs. A separate payoff can be created from these functions and used as another abstraction from the *PayOff* class.

INVESTMENT UNDER UNCERTAINTY

In cases where cash flows are certain it is optimal to take on a project immediately only when project has to be taken in a now-or-never situation or the project dividends are greater than the interest gained from deferring the project. If cash flows are certain and the project can be delayed or the project dividends are less than interest gained from deferring the project than it is optimal to wait until the project becomes optimal. In the case of where there is uncertainty in cash flows, the value of insurance which is the implicit call option influences the decision to delay the project.

A. A Simple DCF Problem

There is no market mechanism that provides sufficient information to directly estimate project returns, volatility, and covariances. Using economics fundamentals and expert estimates from comparable firms with similar projects an analyst can provide enough information for an estimate of project value.

Using the expected return on a project of comparable risk as the discount rate an expected cash flow can be generated. Using DCF the formula for project value is:

$$V = \frac{pX_u + (1-p)X_d}{(1+\alpha)^T} \tag{5.1}$$

Where the uncertain cash flows are predicted to be Xu and Xd. Also, there is an initial investment cost, I_0, and another investment cost, I_1, if the deciding manager chooses to go forward with the project, at that time, T. The real world probability, p, is the probability of the upper cash flow, X_u, even occurring. Assuming the investment is made now or never if

$$V \geq I_0 + \frac{I_T}{(1+r)^T} \qquad or \qquad V - I_0 - \frac{I_T}{(1+r)^T} \geq 0 \tag{5.2}$$

Note that the expected cash flows from equation (5.1) can be isolated

$$E(X) = pX_u + (1-p)X_d \tag{5.3}$$

Equation (5.1) can be rewritten to be simplified as

$$V = \frac{E(X)}{(1+\alpha)^T} \tag{5.4}$$

To finish solving for V, α is next calculated. Expert analysis of comparable firms and projects is used to form opinions around the calculation of α, selection of β, and the observation of r_f, r_M, in the market.

$$\alpha = r_f + \beta(r_M - r_f) \tag{5.5}$$

With the estimated risky expected return, α, and the expected cash flows, $E(X)$, the present value of the project cash flows can be calculated from equation (5.1), where the NPV needs to meet the condition in equation (5.2). Or where NPV is equal to $V - I_0 - I_T/(1+r)^T$

i. Simple DCF Example

Given values:

$$X_u = \$120, X_d = \$80, I_0 = \$10, I_1 = \$95,$$

$$\beta = 1.25, r_M = 0.10, r_f = 0.06, p = 0.60, T = 1$$

Use equation (5.3) to find the expected cashflows

$$E(X) = 0.6 * \$120 + 0.40 * \$80 = \mathbf{\$104}$$

Then using equation (5.5) to find the risky discount rate for the now-or-never project

$$\alpha = 0.06 + 1.25 * (0.10 - 0.06) = \mathbf{0.11}$$

V or the present value of the project is then found using equation (5.4) and where T = 1.

$$V = \frac{\mathbf{\$104}}{(1 + \mathbf{0.11})^1} = \mathbf{\$93.694}$$

Using equation (5.2) the present value of the project

$$\$93.694 - \$10 - \frac{\$95}{(1 + 0.06)^1} = \mathbf{-\$5.929}$$

B. Valuing Derivatives of Cash Flow

Using information from the above DCF problem, valuing the derivative for this project is simple using:

1) The future cash flows in their different states

2) The probabilities of those states

3) The comparability of the project to a traded asset

Example:

Initial investment, I_0, is made at time 0 while a subsequent payment of I_1 is made in time 1 if the project has enough value. Using a binomial option evaluation method is appropriate considering the possible higher and lower outcomes. Using risk neutral pricing is appropriate for valuation of the initial cash flow at year 0 but will be necessary to capture the value of the option to further invest at year 1. Beginning with the link between the value of the project and and the forward price through the discounted cash flow method.

$$F_{0,T} = V(1+r)^T \tag{5.6}$$

using the fact that the expected risk neutral price is the forward price

$$p_u^* X_u + p_d^* X_d = F_{0,T} \tag{5.7}$$

,where $p_d^* = (1 - p_u^*)$

Risk Neutral probabilities can be calculated

$$p_u^* = \frac{F_{0,T} - X_d}{X_u - X_d} \qquad and \qquad p_d^* = \frac{X_u - F_{0,T}}{X_u - X_d} \tag{5.8}$$

In calculating the risk neutral probabilities the project value can be constructed by equating $F_{0,T}$ in equation (5.6) and in equation (5.7) to get

$$\frac{p_u^* X_u + p_d^* X_d}{(1+r)^T} = V \tag{5.9}$$

where V will tie back to the same value as in equation (5.1). Showing that DCF and risk neutral pricing are two methods of deriving the same answer

Using the risk neutral probabilities from (5.8) and the payoff, the present value can be calculated. Besides the initial cost I_0 which is applied to the value of the whole discounted pay-off, the payoff itself uses the cost at time 1, I_1, in the decision to move forward with the project.

$$\frac{p_u^* * max[X_u - I_1, 0] + p_d^* * max[X_d - I_1, 0]}{(1+r)} - I_0 = PV \tag{5.10}$$

This gives the present value of the project but also includes the option in year 1 to continue given the risk neutral probabilities and uncertain cash flows. There isn't much difference between discounted cash flow valuation and real options valuation, both assign a dollar value today to an, at times uncertain, future cash flow, similar to valuing a bond, stock, or option. Typically, value of an option depends on the value of the underlying, but in this case, the valuation of the project. Normally, the market provides the needed valuation for stocks which is then used for stock based options, but the project valuation was not provided but instead estimated using traditional techniques in project option valuation. Risk neutral pricing and discounted cash flow are alternate methods of valuing a future cash flow but can at times yield different results depending on the assumptions used.

i. Valuing Derivatives on the Cash Flow Example

Given the assumptions:

$$X_u = \$120, X_d = \$80, I_0 = \$10, I_1 = \$95, r_f = 0.06$$

X_u, X_d, I_0, I_1, and r_f, were given as input previously in the last section on a simple DCF example but where V was calculated.

Calculating the forward price $F_{0,T}$ using equation (5.6)

$$F_{0,T} = \mathbf{\$93.694}(1 + 0.06)^1 = \mathbf{\$99.315}$$

Now equation (5.7) can be used to find the risk neutral probabilities

$$p_u^* = \frac{\mathbf{\$93.694} - \$80}{\$120 - \$80} = \mathbf{0.4829} \qquad \textit{and} \qquad p_d^* = \frac{\$120 - \mathbf{\$93.694}}{\$120 - \$80} = \mathbf{0.5171}$$

The payoff can now be calculated using equation (**??**). This is value of the project to take on the project initial in time 0 but given the option to continue with the project by applying the additional investment or not to invest further.

$$\frac{\mathbf{0.4829} * max[\$120 - \$95, 0] + \mathbf{0.5171} * max[\$80 - \$95, 0]}{(1 + 0.06)} - \$10 = \mathbf{\$1.389}$$

With the option the project has a positive NPV where previously it was **-$5.929** and would have been rejected without consideration for other mutually exclusive opportunities.

C. Evaluating a Project with a 2-Year Investment Horizon

The decision of when to invest in a risk project is like exercising an American option: The strike price (investment cost) is paid to receive the asset (present value of future cash flows).

By assuming an infinite cash flow stream, after the project has been initiated by paying an initial cost, the project can be treated as a perpetual growing annuity. And the present value is therefore

$$PV = \frac{E(CF_1)}{r_{project} - growth\ rate} \tag{5.11}$$

where $r_{project}$ is found using a pricing model like the CAPM [$r_{project} = r_f + \beta(r_M - r_f)$].

The static NPV is the investment present value less the initial cost. Under the assumption that there are two years in which investment can occur the NPV static rule is applicable at the end of two years but the option to wait needs to be calculated at time 0. The three key elements are needed for option valuation; dividend of the project which is the foregone initial or annual cash flow ($\$D$), interest savings ($r * C_0$) annually, and the implicit insurance for uncertain cash flows that is lost when investment in the project occurs.

Assuming, S, is equal to the static NPV after 2 years, X, to be the initial cost of investment, r to be the risk free rate used in calculating the project rate, volatility to be 0.50, and time to expiration of 2 years. Using the static NPV, or market value of future cashflows, S, and by using the initial cash flow, D, in one year, a constant dividend yield of D/S can be used in calculating the continuously compounding dividend $\delta = ln(1 + (D/S))$. The value of the investment decision can be modelled as an American call option. Where the risk neutral probabilities are

$$p_u^* = \frac{e^{(r-\delta)} - e^{-0.5}}{e^{0.5} - e^{-0.5}} \qquad and \qquad p_d^* = \frac{e^{0.5} - e^{(r-\delta)}}{e^{0.5} - e^{-0.5}} \tag{5.12}$$

A decision tree is typically used in project management and where this project has a decision component a binomial tree is a good fit since binomial trees use probabilities even though they are risk neutral probabilities and the discounting is done in the risk neutral weighting of aggregated cash flows at all nodes and not in the use of discounting cashflows using differing estimated "true" risk-weighted discount rates at each node. The risk neutral probabilities allow for weighting cashflows without needing to know true discount rates at each node but instead weights the cashflows according to the probability of up and down movements of cash flows using a proportion of the cash flow up and down movements. Thus, binomial pricing does not imply that any particular true expected return is constant; instead it tells us how to perform valuation so that the assumptions about the project and the assumptions about the tree are consistent with each other.

To evaluate the option or implicit insurance the uncertain cash flows need to be estimated and some assumptions concerning the cash flows need to be made. A simple assumption used in option pricing of stock and project cash flows is a lognormal distribution of cash flows provided by the Cox-Ross-Rubinstein approach in constructing a binomial tree, $e^{\pm\sigma}$, to be applied to cash flows. Here volatility, σ, is assumed to be 0.50.

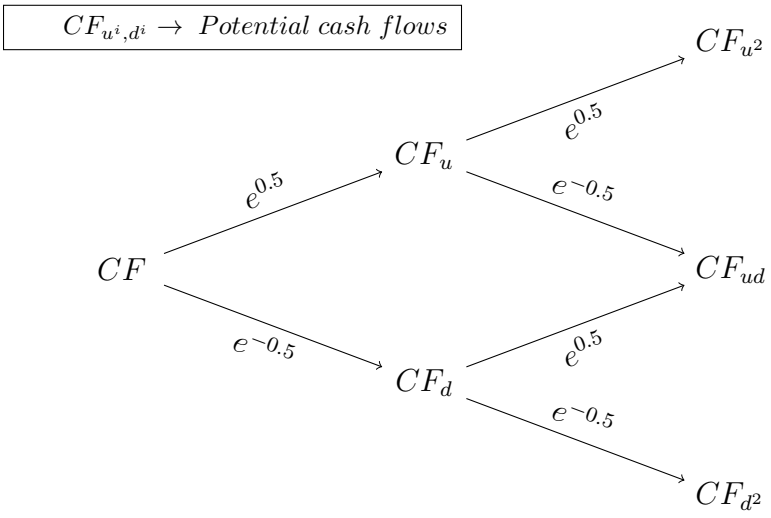

Figure 1: Project potential cash flows, CF_{u^i,d^i}

Next the evolution of the project's present values are mapped, using these potential cash flows, by applying a discount, $r_{project} - growth\ rate$, to each cash flow at each node.

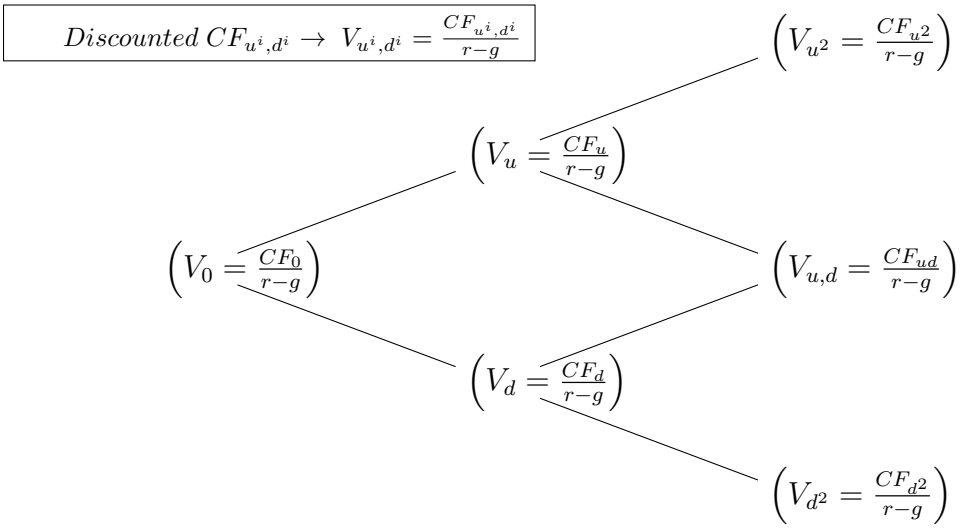

Figure 2: Discounted value of potential project cash flows

Finally a pay off is applied to each of the terminal nodes and then moving backward through the tree, risk neutral probabilities are applied until the present value of the origin node has been calculated; This is the NPV of the project with the

option to exercise early, the American option.

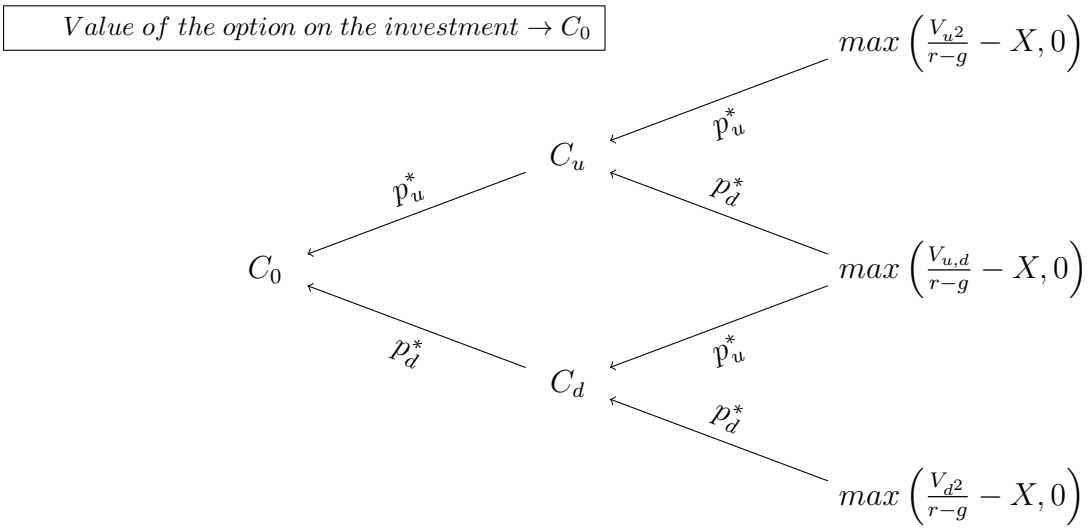

Figure 3: Value of option on project

By executing the option payoff at the year 2 node terminal nodes, $[max(0, V_{u^i,d^i} - X)]$, and in using p_u^* and p_d^* to discount each preceding node, the method ultimately leads to the year 0 node where the value at the year 0 node is the present value of the project. This number will be higher than the static NPV number calculated previously due to the ability to exercise at any time during the 2 years.

The use of the binomial tree here is to aid in creating *fair prices*, not arbitrage-free prices since the option pricing formulas used for the project are used where literal replication of the option is not possible and where a twin security is used instead.

D. Evaluating the Project with an Infinite Investment Horizon

Using the same input information for the previous project which had to be started within 2 years or not at all, the results of the perpetual call option are even higher than the preceding static NPV value and the binomial tree NPV.

i. Evaluating the Project with an Infinite Investment Horizon Example

Using the general function for perpetual options:

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual\left[\frac{E(CF_1)}{r_{project} - g}, C_0, \sigma, r, ln\left(\frac{E[CF_1]}{NPV_{static}}\right)\right] = \{value, price\}$$

where,

$$S = \frac{E(CF_1)}{r_{project} - g} = \frac{\$18}{0.15 - 0.03} = \mathbf{\$150}$$

$$X = C_0 = \mathbf{\$100}$$

$$\sigma = 0.50$$

$$r = ln(1 + r) = ln(1.07) = 0.0676$$

$$\delta = ln\left(\frac{E[CF_1]}{NPV_{static}}\right) = ln\left(\frac{\$18}{\$150}\right) = 0.1133$$

Perpetual option general function:

$$CallPerpetual[\mathbf{\$150}, \mathbf{\$100}, 0.50, 0.0676, 0.1133] = \{\mathbf{\$63.396}, \mathbf{\$245.71}\}$$

Perpetual option program:

PerpetualPayOffCall thePayOff [**$100**, 0.50, 0.0676, 0.1133]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\ (theOption, \mathbf{\$150}) = perpetualOptionPrice$

COMMODITY EXTRACTION AS AN OPTION

A. *SINGLE BARREL EXTRACTION UNDER CERTAINTY*

Assuming a plot of land has one barrel of oil and after the barrel has been extracted the land is worthless. Assuming the effective annual lease rate is interpolated from the oil forward curve to be constant over time and maturity, and that the risk free rate is also constant over time. Adding the known price of a barrel of oil today and the price of oil is known in the future; here using the forward price to tie out the price evolution process and create price certainty.

$$F_{0,T} = S_0 \frac{(1+r)}{(1+\delta)^T} \tag{6.1}$$

The spot price appreciates at a rate of $((1+r)/(1+\delta)) - 1$. The cost of the extraction is fixed at X and can be paid at any time. The value of the land could be as simple as, $S_0 - X$, which is the current bid. It is important to discover the value of the land but in order to do so, first the optimal time to reach the maximum payoff, or present value of net extraction revenue, must be uncovered.

$$PV = \frac{S_T - X}{(1+r)^T} \tag{6.2}$$

S_T is not yet know because of the ability to delay has not been estimated.

B. *Optimal extraction*

A simple rule to follow is that the project will be delayed as long as cost is greater than revenues. In this case the price of optimal extraction is fixed at S_t while the cost of X will increase by a rate of $(1+r)/(1+\delta)$ annually. The benefit of holding X each year is seen in the interest savings r but the cost of the lease d will eat into the benefit of holding the land and the cost of extraction which is the exercise strike

X. In a daily sense, the exercise of the investment occurs when

$$\frac{1}{1+r_{daily}}\left(S\frac{1+r_{daily}}{1+\delta_{daily}}-X\right) > S-X \tag{6.3}$$

or as long as tomorrow's discounted payoff is greater than today's payoff. Through manipulation this daily rate can be turned into a continuously compounded rate. This equation further reduces to:

$$S=\frac{r_{daily}}{\delta_{daily}}\frac{(1+r_{daily})}{(1+\delta_{daily})}X \tag{6.4}$$

Where the inequality is dropped because the point of interest is where costs equal benefits. Since daily rates are essentially continuously compounded rates and therefore the equation would be

$$S_T=\frac{ln(1+r_{annual})}{ln(1+d_{annual})}X \tag{6.5}$$

C. Value and Appreciation of the Land

Using S_T from equation (6.5) combined with given values for S_0, r, δ, t can be solved for in,

$$S_0\left(\frac{1+r}{1+\delta}\right)^t = S_T \tag{6.6}$$

where optimal time T,

$$T=\frac{ln\left(\frac{S_T}{S_0}\right)}{ln\left(\frac{1+r}{1+\delta}\right)} \tag{6.7}$$

Now the present value can be calculated now that the ending spot price has been calculated.

$$PV = \frac{S_T - X}{(1+r)^T} \tag{6.8}$$

The present value of the land is found. This is price of the land today which will either be greater than $S_0 - X$ or will be 0.

The oil in the land appreciates at a rate of $((1+r)/(1+d)) - 1$ whereas the land appreciates at $1 + r$ otherwise it would be better to invest in risk free bonds. This is the minimum return for the developed or undeveloped project to be of values to the investor.

i. DCF Single Barrel Under Certainty Example

Given values:

$$S = \$15, X = \$13.60, \sigma = 0.0001, r = 0.05, \delta = 0.04$$

Using equation (6.5) to find the optimal price given the constraint from equation (6.3) where the daily decision to invest is reduced to continuously compounded rates applied to cost.

$$S_T = \frac{ln(1+0.05)}{ln(1+0.04)} * \$13.60 = \mathbf{\$16.918}$$

Using this optimal price for extraction, S_T, the time to optimal extraction can be solved from equation (6.7)

$$\mathbf{\$16.918} = \$15 * \left(\frac{1+0.05}{1+0.04}\right)^{t} \rightarrow t = \mathbf{12.575}\, years$$

Having S_T, t, and they can now be used in the net extraction equation, equation (6.2), to find the NPV of the project.

$$\frac{\mathbf{\$16.918} - \$13.60}{(1.05)^{\mathbf{12.575}}} = \mathbf{\$1.796}$$

ii. Option Pricing Single Barrel Under Certainty Example

In the context of an option pricing formula the problem is analogous to deciding when to exercise a call option. An asset (oil) is received for the strike price (extraction cost). The trade off between interest saved and foregone dividends are considered for early exercise. When the oil is in possession is can be lease and the oil's lease rate is the dividend yield. General perpetual option function:

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual[S_0, X, \sigma, ln(1+r), ln(1+\delta)] = \{value, price\}$$

where,

$$S = \$15, X = \$13.60, \sigma = 0.0001, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$15, \$13.60, 0.0001, 0.04879, 0.03922] = \{\mathbf{\$1.796}, \mathbf{\$16.918}\}$$

Values in perpetual option program:

PerpetualPayOffCall thePayOff [\$13.60, 0.0001, 0.04879, 0.03922]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\ (theOption, \$15) = perpetualOptionPrice$

D. SINGLE BARREL EXTRACTION UNDER UNCERTAINTY

Given the same assumptions in terms of r, d, X, etc. the extraction price S_T and value of the undeveloped land will change. In the previous section the lease rate (dividend) and interest savings from holding delayed exercise of the option and paying out the extraction cost were considered but the value of insurance from holding the option was not considered. That is because there was no uncertainty about the cash flows given the assumption that the spot price given the discount rates will eventually hit the forward price and given that the exercise of the option is at the point where this price is equal to the continuously compounded evolution of the static extraction cost (X).

In this case when the foregone dividend is greater than the giving up the implicit insurance the option provides then it will be optimal to exercise the option. The uncertainty gives the insurance of the option it's value and the which then increases the value of delay. So as oil extraction is delayed more time is given to see where the oil price will go. Delay will continue to occur until the optimal investment price is reached. When price S reaches $\bar{S}$, the optimal price, the option will be exercised and $\bar{S} - X$ will be received. The equation to value the payoff of $\bar{S} - X$ when $\bar{S}$ is reached gives us the following value of the extraction option:

$$PV = (\bar{S} - X)\left(\frac{S}{\bar{S}}\right)^{h_1} \tag{6.9}$$

where

$$h_1 = 0.5 - \frac{r-\delta}{\sigma^2} + \sqrt{\left(\frac{r-\delta}{\sigma^2} - 0.5\right)^2 + \frac{2r}{\sigma^2}} \tag{6.10}$$

By varying $\bar{S}$, the investment trigger, and by observing the PV, the maximum PV can be found. With greater volatility, the extraction trigger price and the invest-

ment strategy present value both increase.

Given these same values in the perpetual option pricing program the same investment trigger and present value of the investment strategy.

i. Single Barrel Extraction under Uncertainty Example

$$S_0 = \$15, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Using the pay off of a perpetual American call, equation(6.9)

$$(\bar{S} - \$13.60)\left(\frac{\$15}{\bar{S}}\right)^{h_1}$$

where,

$$h_1 = 0.5 - \frac{0.05 - 0.04}{0.15^2} + \sqrt{\left(\frac{0.05 - 0.04}{0.15^2} - 0.5\right)^2 + 2 * \frac{0.05}{0.15^2}} = 2.05533$$

By maximizing equation(6.9) for $\bar{S}$, where $\bar{S}$ is equal to \$25.3388 which gives the value of the land calculated in equation(6.9)

$$(\mathbf{\$25.3388} - \$13.60)\left(\frac{\$15}{\mathbf{\$25.3388}}\right)^{2.0533} = \{\mathbf{\$3.7856}\}$$

In figure (1) the function in equation (**??**) is made with respect to a range of spot prices. Note the maximum price for the 0.15 volatility line and note the value or the height of that same line. The price of \$25.3388 and the value of \$3.7856 correspond to the example above. Also, the 0.30 volatility line is also with its price of \$1.796 and value of \$16.918 in the example previous to this.

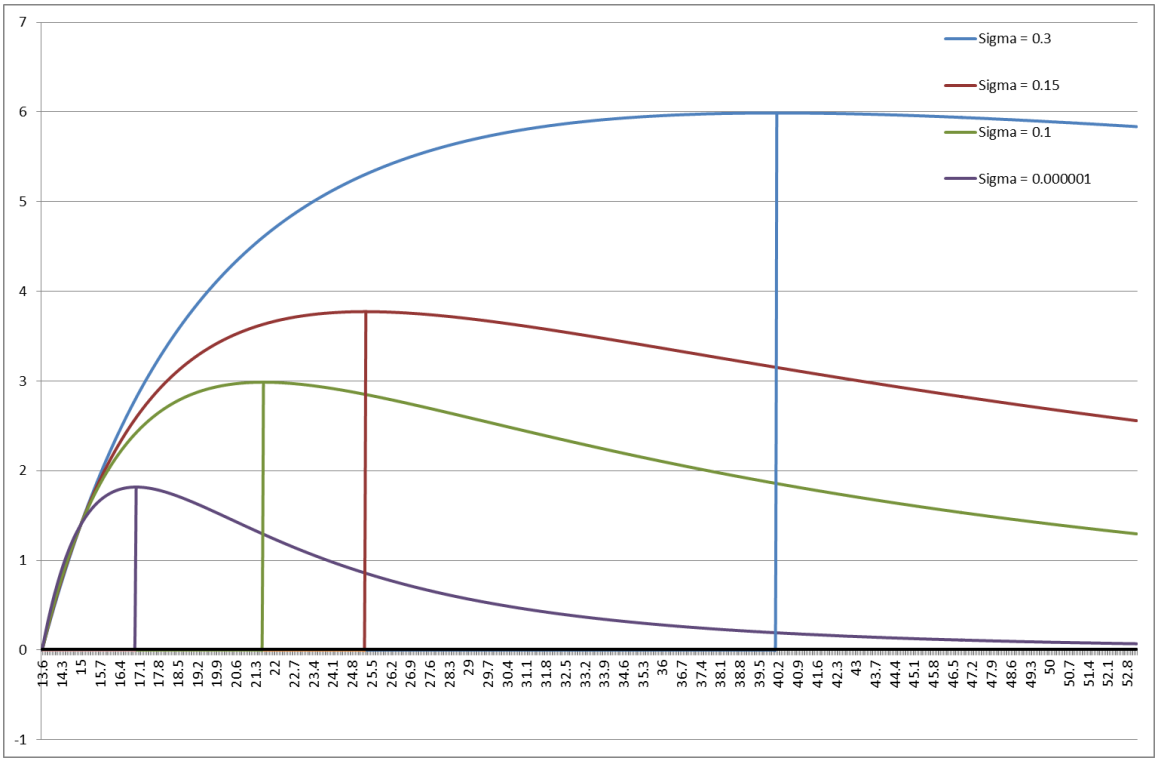

Figure 1: Maximizing equation (6.9) with respect to $\bar{S}$ given 4 different volatilities

ii. Perpetual Option Example

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual[S_0, X, \sigma, ln(1+r), ln(1+\delta)] = \{value, price\}$$

where,

$$S = \$15, X = \$13.60, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$15, \$13.60, 0.15, 0.04879, 0.03922] = \{\mathbf{\$3.7856}, \mathbf{\$25.3388}\}$$

Values in perpetual option program:

PerpetualPayOffCall thePayOff [\$13.60, 0.15, 0.04879, 0.03922]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\,(theOption, \$15) = perpetualOptionPrice$

E. VALUING AN INFINITE OIL RESERVE

Assuming that a barrel of oil can be extracted each year forever and that oil prices are certain and increasing the value of the producing firm can be assessed. Firm can at any time invest I in order to develop the undeveloped reserve. One year after the project is taken one barrel of oil will be received at a cost of c per barrel each year.

i. Value of Producing Firm

The lease rate is the discount rate that connects the future commodity price with the current commodity price. The time t value of a bond received at time T is

$$PV_t(F_{t,T}) = \frac{F_{t,T}}{(1+r)^{T-t}} = \frac{S_{t,T}}{(1+\delta)^{T-t}} \tag{6.11}$$

from

$$F_{0,T} = S_0 \frac{(1+r)^T}{(1+\delta)^T} \tag{6.12}$$

Value of the producing firm at time t,

$$\Sigma_{i=1}^{\infty} \frac{F_{0,i} - c}{(1+r)^i} = \Sigma_{i=1}^{\infty} \left(\frac{S_0}{(1+\delta)^i} - \frac{c}{(1+r)^i} \right) = \frac{S_0}{\delta} - \frac{c}{r} \tag{6.13}$$

Because a perpetual coupon bond paying \$c a year is worth $\frac{c}{r}$ the present value of a barrel of oil a year forever is $\frac{S_0}{\delta}$. Also, the lease rate on a commodity is analogous to the interest rate on a cash bond. Thus, the operating well is like a bond paying a unit of the commodity forever, so the lease rate, δ is the appropriate discount rate for a bond denominated in a commodity and $\frac{S_0}{\delta}$.

ii. Value of the Option to Invest

Investing at S_T, the value of the land is the value of the producing firm less the investment costs, I , or $\left(\frac{S_T}{\delta} - \frac{c}{r} - I\right)$. The value of the land, the value of the undeveloped reserve, is

$$\frac{1}{(1+r)^T}\left(\frac{S_T}{\delta} - \frac{c}{r} - I\right) \tag{6.14}$$

or in the single barrel case

$$\frac{1}{(1+r)^T}\frac{1}{\delta}\left(S_T - \delta\left(\frac{c}{r} + I\right)\right) \tag{6.15}$$

$$Single \quad \frac{S_T - X}{(1+r)^T} \quad \rightarrow \quad Infinite \quad \frac{\frac{S_T}{\delta} - [\frac{c}{r} - I]}{(1+r)^T} \tag{6.16}$$

Which is also the present value of oil extraction.

Now as in the case of the single barrel extraction the T must be chosen to maximize the value of the land today. Oil prices are assumed uncertainty but are assumed to grow indefinitely but these assumptions do not change the fundamentals of the problem therefore it is comparable to the single barrel problem.

iii. Value of the Producing Well

The value of the producing well is $\frac{S}{\delta} - \frac{c}{r}$. If the investment cost per-barrel extraction cost is, I, is such that when $\delta\left(\frac{c}{r} + I\right) = X$ then it is the same as having $\frac{1}{\delta}$ options to extract at a cost of X and the solution is the same as the single-barrel case. Thus δ becomes a scalar that links the single-barrel and infinite-barrel case.

The output from this function is the present value of the infinite barrel case, PV_∞, and the trigger price, S_{T_∞}. This relates to single barrel case where single barrel is a function of the infinite barrel case.

$$PV_{Single} = \delta * PV_{\infty} \tag{6.17}$$

$$S_{T,Single} = \delta * S_{T,\infty} \tag{6.18}$$

When σ is greater than some small number i.e., 0.000001, then the price of oil is said to be lognormally distributed.

iii..1 Perpetual Option Example $\sigma = 0.000001$

General perpetual option function

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual\left[\frac{S}{\delta}, \frac{c}{r} + I, \sigma, ln(1+r), ln(1+\delta)\right] = \{value, price\}$$

where,

$$S = \$15, c = \$8, I = \$180, \sigma = 0.000001, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$375, \$340, 0.000001, 0.04879, 0.03922] = \{\mathbf{\$44.914}, \mathbf{\$422.956}\}$$

The per-barrel value of the well at extraction

$$\delta * value = 0.04 * \mathbf{\$44.914} = \mathbf{\$1.796}$$

Per-barrel extraction occurs at

$$S = \delta * price = 0.04 * \mathbf{\$422.956} = \mathbf{\$16.918}$$

Values in perpetual option program:

PerpetualPayOffCall thePayOff [**$340**, 0.000001, 0.04879, 0.03922]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\ (theOption, \mathbf{\$375}) = perpetualOptionPrice$

iii..2 Perpetual Option Example $\sigma = 0.15$

General perpetual option function

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual\left[\frac{S}{\delta}, \frac{c}{r} + I, \sigma, ln(1+r), ln(1+\delta)\right] = \{value, price\}$$

where,

$$S = \$15, c = \$8, I = \$180, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$375, \$340, 0.15, 0.04879, 0.03922] = \{\mathbf{\$94.639}, \mathbf{\$633.469}\}$$

The per-barrel value of the well at extraction

$$\delta * value = 0.04 * \mathbf{\$94.639} = \mathbf{\$3.7856}$$

Per-barrel extraction occurs at

$$S = \delta * price = 0.04 * \mathbf{\$633.469} = \mathbf{\$25.3388}$$

Values in perpetual option program:

PerpetualPayOffCall thePayOff [**$340**, 0.15, 0.04879, 0.03922]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\ (theOption, \mathbf{\$375}) = perpetualOptionPrice$

COMMODITY EXTRACTION WITH SHUTDOWN AND RESTART OPTIONS

Given that there is production over time and that oil prices are uncertain then there are two choices that additionally come into play. The choice to continue the operation of the operating well or the restart of a shutdown well.

A. Initial Investment in the Well

There is undeveloped land with oil in it and the desire to drill for it. The question becomes when should the land be developed into an operating well.

i. Continue to produce

For a developed well the only two options is to continue producing or to shutdown the well and incur a cost of shutdown.

ii. Restart the Operating Well

Restarting the shutdown well and incur a restart cost.

Initial investment occurs when $\bar{S}$ is reached, shutdown when S_* is breached, and restarted again when S^* is hit. The value of each of these trigger prices needs to found in order to find the value of the land.

To find the value of the land, knowing how to determine S_* and S^*, determining the value of the producing well which is the present value of future cash flows at investment, and determining $\bar{S}$ which is the investment decision rule.

Determine (S_*, S^*) Shutdown once, permanently, land then has no value. Shutdown once, restart once, permanently Production can be shutdown and restarted infinitely many times

Each of these cases adds more layer of options with additional costs to do so.

iii. Permanent Shutdown

In the case of the one time permanent shutdown begins with the operating well $\frac{S}{\delta} - \frac{c}{r}$.

Included is that assumption that at any time, but most certainly when S reaches S_*, a shutdown cost of K_s can be paid, the well is abandoned permanently.

Value of shutting down when shutdown occurs $\frac{S}{\delta}$ which is the present value revenue streams, is given up, no more revenue is received $\frac{c}{r}$ which is the present value of costs, no more extraction costs are paid K_s is paid in order to shutdown

The value of shutting down at price S_* at a cost of K_S is similar to and even reduces to the payoff of a put option where strike price is X and asset price is S $(X - S)$.

$$-\frac{S_*}{\delta} + \frac{c}{r} - K_s = \left(\frac{c}{r} - K_s\right) - \left(\frac{S_*}{\delta}\right) \tag{7.1}$$

If during the operation of the well for any oil spot price S the value of this put can be used to determine the value of the option to shutdown as well as determining the trigger price S_* for shutting down.

From the perpetual put function below, a value for the perpetual put option is given along with a trigger price H. Note as the volatility decreases both the perpetual put option price and the shutdown trigger price.

Since shutdown is permanent, in this case, the zero NPV price $S = \delta * \frac{c}{r}$ is the point where the well is incurring operating losses but it isn't until much lower when the trigger price is hit that the well is shutdown. The shutdown trigger is much lower because the decision to shutdown is irreversible. Another natural benchmark for shutdown is when the price is equal to the marginal cost of production, c, in this

case. If $S > c$, then the well is making money and will not shutdown. If $c > S > \delta * \frac{c}{r}$, then the well is losing money at an operating lose but the NPV is still greater than 0. If $S < \delta * \frac{c}{r}$, it makes sense to shutdown the well even if the shutdown is permanent but shut down is irreversible and the future possible gains will be lost.

iii..1 Permanent Shutdown Example $k_s = \$0$

Given these assumptions:

$$S = \$10, c = \$8, k_s = \$0, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Using the value of shutting down, equation(7.1), at optimal the per-barrel shutdown trigger price, $\mathbf{S_*} = \mathbf{\$4.273}$.

$$-\frac{\mathbf{\$4.273}}{.04} + \frac{\$8}{0.05} - \$0 = \left(\frac{\$8}{0.05} - \$0\right) - \frac{\mathbf{\$4.273}}{0.04} = \$53.17$$

General perpetual option function

$$CallPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual\left[\frac{S}{\delta}, \frac{c}{r} - k_s, \sigma, ln(1+r), ln(1+\delta)\right] = \{value, price\}$$

where,

$$S = \$15, c = \$8, k_s = \$0, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$375, \$160, 0.15, 0.04879, 0.03922] = \{\mathbf{\$9.6333}, \mathbf{\$106.83}\}$$

The per-barrel value of the well at extraction

$$\delta * value = 0.04 * \mathbf{\$94.639} = \mathbf{\$3.7856}$$

where the optimal per-barrel extraction price is

$$Per\ barrel\ trigger\ price = \mathbf{\$106.83} * 0.04 = \mathbf{\$4.273}$$

Natural Benchmarks Zero NPV

1) $S = r * c/r = 0.04 * \$8/0.05 = \mathbf{\$6.40}$

2) Marginal cost, c, **$8**

Benchmark decision triggers

When $S > \$8$, making money

When $\$8 > S > \$6.40$, losing money but no shut down

When $S < \$6.40$, losing money, shutdown

Values in perpetual option program:

PerpetualPayOffPut thePayOff $[\$160, 0.15, 0.04879, 0.03922]$

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\,(theOption, \$375) = perpetualOptionPrice$

iii..2 Permanent Shutdown Example $k_s = -\$25$

Given these assumptions:

$$S = \$10, c = \$8, k_s = -\$25, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Using the value of shutting down, equation(7.1), at optimal the per-barrel shutdown trigger price, $\mathbf{S}_* = \$\mathbf{3.605}$.

$$-\frac{\mathbf{\$3.605}}{.04} + \frac{\$8}{0.05} - \$0 = \left(\frac{\$8}{0.05} - \$0\right) - \frac{\mathbf{\$3.605}}{0.04} = \$69.863$$

General perpetual option function

$$PutPerpetual[S, X, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$PutPerpetual\left[\frac{S}{\delta}, \frac{c}{r} - k_s, \sigma, ln(1+r), ln(1+\delta)\right] = \{value, price\}$$

where,

$$S = \$250, X = \$135, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$PutPerpetual[\$250, \$135, 0.15, 0.04879, 0.03922] = \{\mathbf{\$5.778}, \mathbf{\$90.137}\}$$

where the optimal per-barrel extraction price is

$$Per\ barrel\ trigger\ price = \mathbf{\$90.137} * 0.04 = \mathbf{\$3.605}$$

Natural Benchmarks Zero NPV

1) $S = r * c/r = 0.04 * \$8/0.05 = \mathbf{\$6.40}$

2) Marginal cost, c, **$8**

Benchmark decision triggers

When $S > \$8$, making money

When $\$8 > S > \$6.40$, losing money but no shut down

When $S < \$6.40$, losing money, shutdown

Values in perpetual option program:

PerpetualPayOffPut thePayOff $[\$135, 0.15, 0.04879, 0.03922]$

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\,(theOption, \$250) = perpetualOptionPrice$

B. The Value of the Producing Well

Given shutdown is possible the value of the producing well is the value of the perpetually producing well plus the value of the shutdown option

$$V_{operating}(S) = V_{no\ shutdown}(S) + V_{shutdown\ option}(S) \tag{7.2}$$

or

$$\frac{S}{\delta} - \frac{c}{r} + PutPerpetual\left[\frac{S}{\delta}, \frac{c}{r} - k_s, \sigma, ln(1+r), ln(1+\delta)\right] \tag{7.3}$$

Without the shutdown option the value of the well is like a stock and declines to $-\frac{c}{r}$ when $S = 0$. With the option, the well is worth zero once it it shutdown. The shutdown option impacts valuation of the operating well when the oil price is significantly above the shutdown price, the shutdown option is worth little and the value of the well changes by $\frac{1}{\delta}$ for each \$1 change in the oil price. The value of the well becomes less sensitive to the oil price the closer it comes to the shutdown price, because the shutdown option is increasing in value to absorb the effect of declines in oil price.

C. Investing when Shutdown is Possible

The ability to shutdown the well affects the initial investment decision. The investor should be willing to invest sooner because the shutdown option allows for a lower potential loss for the operating well.

It is appropriate then to work backward, accounting for the value of the shutdown option, the value of the time of investment then is found by

$$\frac{\bar{S}}{\delta} - \frac{c}{r} + PutPerpetual\left[\frac{\bar{S}}{\delta}, \frac{c}{r} - k_s, \sigma, ln(1+r), ln(1+\delta)\right] - I \tag{7.4}$$

By finding the present value of this equation the $\bar{S}$ needed to maximize the present value is found. Solving for $\bar{S}$ by finding the present value of the equation above, then choosing $\bar{S}$ to maximize this present value

Equation (7.4) indicates that for a given $\bar{S}$, the value of investing when $S = \bar{S}$. If the price of the oil today $S < \bar{S}$, the present value of equation (7.4) using equation (7.5) here below where the present value is

$$V_{Invest}(S;\bar{S}) = \left(\frac{S}{\bar{S}}\right)^{h1} * \left(\frac{\bar{S}}{\delta} - \frac{c}{r} + PutPerpetual\left[\frac{\bar{S}}{\delta}, \frac{c}{r} - K_s, \sigma, ln(1+r), ln(1+\delta)\right] - I\right) \tag{7.5}$$

This equation can be maximized with respect to $\bar{S}$ using numerical software or by visual inspection on a graph.

Three important facts can be seen when using these equations.

1) The ability to shutdown reduces the trigger price.

2) If a shutdown cost is required then the shutdown occurs at a higher price and less protection is observed. Causing the value of shutting down to decrease and raising the trigger price.

3) The investment trigger implied by maximizing the last equation is indepen-

dent of S, the current oil price; for any given $\bar{S}$ and any given S where $\bar{S} > S$, S will have to pass any number of prices to reach $\bar{S}$ but for each of these prices the resulting $\bar{S}$ will be the same.

i. Value of Producing Well and Investing when Shutdown is Possible Example

Given these assumptions:

$$c = \$8, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Where $\bar{S}$ is the value of equation (7.5) when it is maximized with respect to .

The value of the investment without the shutdown option and when S = 0

$$-c/r = -\$8/0.05 = -\$160$$

The change in the value of the investment for each \$1 change in oil price

$$1/\delta \ = \ 1/0.04 = \$25$$

The value of the investment at time of investment (equation 7.4) increases when shutdown is possible because maximum loss is reduced and investment will happen sooner.

The ability to shutdown reduces the investment trigger. Shutdown costs reduces the effect of the ability to shutdown, shutting down occurs at a higher price and provides less protection. The ability to shutdown provides incentive to invest when prices would have been otherwise less favorable thus initial investment occurs sooner. Shutdown costs reduces the benefit of the ability to shutdown and causes shutdown to occur sooner, at a higher price, because the cost reduces the protection provided by the option.

Shutdown	S	K_s	Sbar	Undeveloped Well
With	\$15	-	\$25.34	-
With	\$15	\$0	\$25.12	\$95.13
With	\$15	\$25	\$25.21	\$94.93
With	\$20	\$0	\$25.12	\$177.01
With	\$20	\$25	\$25.21	\$176.64

The ability to shutdown, noting previously observed without the shutdown option the investment trigger was \$25.3388, then

\$25.12 with shutdown option

\$25.21 with shutdown option and with a \$25 shutdown cost

Values in perpetual option program:

PerpetualPayOffPut thePayOff [$\$160 - k_s, 0.15, 0.04879, 0.03922$]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\left(theOption, \frac{\bar{S}}{\delta}\right) = perpetualOptionPrice$

D. Restarting Production

Now for the case where a one time permanent restart is possible. The ability to restart is a call option where the firm receives $\frac{S}{\delta}$ by paying $\frac{c}{r} + k_r$, future production costs plus the restart cost.

The value of the shutdown well is

$$CallPerpetual\left[\frac{S}{\delta}, \frac{c}{r} + k_r, \sigma, ln(1+r), ln(1+\delta)\right] \tag{7.6}$$

The ability to restart affects the decision to shutdown. Shutting down is a way to cut future losses but also there is an implicit option acquired, a call option, to restart. In equation (7.5), when investment occurred a put option was acquired and implicit to the investment decision, similarly when shutdown occurred an implicit call option was acquired. Since these options are fully embedded in the shutdown and restart of the well, the NPV solution then becomes an optimization of optimal trigger prices for the perpetual implicit embedded options of the optimal shutdown and restart decisions.

i. Restart Shutdown Well Example $k_r = \$0$

Given these assumptions:

$$S = \$10, k_r = \$0, c = \$8, \sigma = 0.15, r = 0.05, \delta = 0.04$$

General perpetual option function

$$CallPerpetual[S, K, \sigma, r, \delta] = \{value, price\}$$

but more specifically to this problem:

$$CallPerpetual\left[\frac{S}{\delta}, \frac{c}{r} + k_r, \sigma, ln(1+r), ln(1+\delta)\right] = \{value, price\}$$

where,

$$S = \$250, X = \$160, \sigma = 0.15, r = 0.05, \delta = 0.04$$

Values in perpetual option general function:

$$CallPerpetual[\$250, \$160, 0.15, 0.04879, 0.03922] = \{\mathbf{\$94.46}, \mathbf{\$11.92}\}$$

where the optimal per-barrel extraction price is

$$Per\ barrel\ trigger\ price = \mathbf{\$11.92} * 0.04 = \mathbf{\$0.4768}$$

where the value, per-barrel, value of the well is

$$Per\ barrel\ trigger\ price = \mathbf{\$94.46} * 0.04 = \mathbf{\$3.7784}$$

Values in perpetual option program:

PerpetualPayOffCall thePayOff [\$160, 0.15, 0.04879, 0.03922]

$VanillaOption\ theOption(thePayOff)$

$PerpetualOptionPrice\,(theOption, \$250) = perpetualOptionPrice$

CONCLUSIONS

Project valuation using American perpetual options is ideal for instances where a perpetual cash flow is used to assess the value of the project on an NPV basis. When uncertainty about the evolution of cash flows is present then the implicit insurance of the option of the project has value. As the value of additional insurance, or as uncertainty in the cash flow evolution increases, the value of the project increases. When additional costs are introduced into the project they reduce the value of the project and cause the option to trigger sooner and also reduce the effect of the option. A simple analytical program can be made similar to the general perpetual option function that is used in the beginning of each example in order to solve for American perpetual options in a setting where perpetuities are used to summarize growth of cash flows. The use of perpetuities to summarize cash flows in calculating NPV allows the analyst the opportunity to use American perpetual options to arrive at a value that also contains the implicit option for projects with potential delay and with uncertain cash flows. This method can provide greater flexibility and shorten the time that is taken for analysis.

REFERENCES

Joshi, M. S. (2008). *C++ Design Patterns and Derivatives Pricing, Second edition.* Cambridge, UK: Cambridge University Press.

McDonald, R. L. (2013). *Derivatives Markets.* Upper Saddle River, New Jersey: Prentice Hall.

Whitehead, N. (2011, February 6). Paul Wilmott on Quantitative Finance, Chapter 9, Perpetual American call. Retrieved from http://www.youtube.com/watch?v=xklbDx1gHGg

Wilmott, P. (2006). Paul Wilmott on Quantitative Finance, second edition. Chichester, West Essex, England: John Wiley Sons Ltd.

APPENDIX - PERPETUAL OPTION PRICING PROGRAM

```
////////////    Main.cpp
#include "PerpetualOptionExamples.hpp"
#include <iostream>
#include <cmath>


int main(){

  std::cout << "Welcome to the Perpetual Option Pricer" << std::endl;
  char yesNo = 'y';
  char selection = 'l';
  int selectionNumber = 0;

  do{

    if(yesNo == 'y'){
    //call menu return number
    std::cout << "\n\n\nWhat perpetual option problem would you like to
        explore: " << std::endl;
    std::cout << " 1: Static NPV/Perpetual Option NPV example, page 510-512
        " << std::endl;
    std::cout << " 2: Simple Discounted CashFlows, page 513-515" << std::
        endl;
    std::cout << " 3: Evaluating Project With a 2 Year Investment Horizon,
        page 515-516" << std::endl;
    std::cout << " 4: Single Barrel Certainty, page 525-527" << std::endl;
    std::cout << " 5: Single Barrel Certainty Perpetual Option, page
        527-528" << std::endl;
    std::cout << " 6: Single Barrel Uncertainty Perpetual Option, page,
        528-530" << std::endl;
    std::cout << " 7: Valuing an Infinite Oil Reserve" << std::endl;
```

```
std::cout << "     Example 17.3 - near zero volatility, page 531" <<
    std::endl;
std::cout << " 8: Valuing an Infinite Oil Reserve" << std::endl;
std::cout << "     Example 17.4 - 0.15 volatility, page 531" << std::
    endl;
std::cout << " 9: Valuing an Infinite Oil Reserve with Permanent
    Shutdown" << std::endl;
std::cout << "     Example 17.5A - zero shutdown cost, page 533" << std
    ::endl;
std::cout << " 10: Valuing an Infinite Oil Reserve with Permanent
    Shutdown" << std::endl;
std::cout << "     Example 17.5b - 15 shutdown cost, page 534" << std::
    endl;
std::cout << " 11: Valuing an Infinite Oil Reserve with Optimal
    Permanent Shutdown" << std::endl;
std::cout << "     Example 17.6, page 536" << std::endl;
std::cout << " 12: Valuing an Infinite Oil Reserve with Permanent
    Restart" << std::endl;
std::cout << "     Example 17.7, page 536" << std::endl;
std::cout << " 0: Quit" << std::endl;
std::cout << " \n:__\b\b";
std::cin >> selectionNumber;


//take number and run switch statment to find corresponding letter
switch(selectionNumber){
  case 1:
    selection = 'a';
    break;
  case 2:
    selection = 'b';
    break;
  case 3:
```

```
  selection = 'c';
  break;
case 4:
  selection = 'd';
  break;
case 5:
  selection = 'e';
  break;
case 6:
  selection = 'f';
  break;
case 7:
  selection = 'g';
  break;
case 8:
  selection = 'h';
  break;
case 9:
  selection = 'i';
  break;
case 10:
  selection = 'j';
  break;
case 11:
  selection = 'k';
  break;
case 12:
  selection = 'l';
  break;
case 0:
  selection = 'n';
  break;
}
```

```
//take letter and run switch statment to call corresponding function
switch(selection){
  case 'a':
    whiteSpace();
    exampleStaticNPVProjectAsOption();
    break;
  case 'b':
    whiteSpace();
    exampleSimpleDCF();
    break;
  case 'c':
    whiteSpace();
    exampleEvaluatingProject2YearInvestment();
    break;
  case 'd':
    whiteSpace();
    exampleSingleBarrelCertainty();
    break;
  case 'e':
    whiteSpace();
    exampleSingleBarrelCertaintyPO();
    break;
  case 'f':
    whiteSpace();
    exampleSingleBarrelUncertainty();
    break;
  case 'g':
    whiteSpace();
    exampleInfiniteWell173();
    break;
  case 'h':
```

```
    whiteSpace();
    exampleInfiniteWell174();
    break;
  case 'i':
    whiteSpace();
    exampleInfiniteWellPermShutdown175A();
    break;
  case 'j':
    whiteSpace();
    exampleInfiniteWellPermShutdown175B();
    break;
  case 'k':
    whiteSpace();
    exampleInfiniteWellPermShutdownOptimalPermShutdown176();
    break;
  case 'l':
    whiteSpace();
    exampleInfiniteWellPermRestart177();
    break;
  case '0':
    break;
}

if(selectionNumber == 0){
  return 0;
}

std::cout << "\n\nWould you like to do another example?: (y/n)\t_\b";
std::cin >> yesNo;
if(yesNo == 'n'){
  std::cout << "\n\nThank you for using the oil extraction pricer\n"
      << std::endl;
  break;
```

```
    }
   }
  } while(selection != '0');



return 0;
}



////////////    PerpetualOptionExamples.hpp

#include "AnalystPackage.hpp"
#include "PerpetualOptionPricer.hpp"
#include "PerpetualPayOff.hpp"
#include "VanillaOption.hpp"
#include <iostream>
#include <cmath>
#include <vector>
#include "minmax.hpp"


void exampleStaticNPVProjectAsOption();

void exampleSimpleDCF();

void exampleEvaluatingProject2YearInvestment();

void exampleSingleBarrelCertainty();

void exampleSingleBarrelCertaintyPO();
```

```
void exampleSingleBarrelUncertainty();

void exampleInfiniteWell173();

void exampleInfiniteWell174();

void exampleInfiniteWellPermShutdown175A();

void exampleInfiniteWellPermShutdown175B();

void exampleInfiniteWellPermShutdownOptimalPermShutdown176();

void exampleInfiniteWellPermRestart177();

void whiteSpace();

void FewStars();

void ManyStars();
```

```
////////////// PerpetualOptionMain.cpp

/*
requires PerpetualPayOff.cpp
        VanillaPayOff.cpp
        PerpetualOptionPricer.cpp
        VanillaOption.cpp
*/
```

```
#include "PerpetualOptionExamples.hpp"


void exampleStaticNPVProjectAsOption(){
  std::cout << "\n\n Static NPV example, page 510-511\n\n" << std::endl;


  double revenuePerUnit = 0;
  double costPerUnit = 0;
  double initialCost = 0;
  double originalLeaseRate = 0;
  double originalRiskFreeRate = 0;

  std::cout << "\nRevenue per unit:         : $_\b";
  std::cin >> revenuePerUnit;
  std::cout << "Cost per unit:            : $_\b";
  std::cin >> costPerUnit;
  std::cout << "Initial cost:             : $_\b";
  std::cin >> initialCost;
  std::cout << "Risk Free Rate(simple rate): : _\b";
  std::cin >> originalRiskFreeRate;
  std::cout << "Lease Rate(simple rate):  : _\b";
  std::cin >> originalLeaseRate;

  double PVrevenuePerUnit = revenuePerUnit/(originalRiskFreeRate-
     originalLeaseRate);
  double PVcostPerUnit = costPerUnit/originalRiskFreeRate;
  double logLeaseRate = log(1+originalLeaseRate);
  double opportunityCost = initialCost*originalRiskFreeRate;
  double benchmark = opportunityCost+costPerUnit;

  double NPVInvestToday = (revenuePerUnit/(1+originalLeaseRate))*(1/((1+
     originalRiskFreeRate)/(1+originalLeaseRate)-1))-PVcostPerUnit-
```

```
    initialCost;
std::cout << "\n\nStatic NPV today is           : $" << NPVInvestToday <<
     std::endl;



std::cout << "\n\n\n\n\nA benchmark is used for the investment decision"
    << benchmark << std::endl;
std::cout << "Interest Saved + Marginal Cost    : $" << benchmark << std
    ::endl;
double optTimeN = log((opportunityCost+costPerUnit)/revenuePerUnit)/
    logLeaseRate;
std::cout << "Number of years to reach this benchmark: " << optTimeN <<
    std::endl;

double optTimeNPV = (((pow((1+originalLeaseRate),optTimeN)*revenuePerUnit
    )/(originalRiskFreeRate-originalLeaseRate))-(costPerUnit/
    originalRiskFreeRate)-initialCost)*(1/pow((1+originalRiskFreeRate),
    optTimeN));
std::cout << "The NPV of waiting that long is   : $" << optTimeNPV << std
    ::endl;

double perUnitPrice = optTimeNPV*originalLeaseRate;
std::cout << "The per unit exercise price       : $" << perUnitPrice <<
    std::endl;


///////////////
FewStars();

std::cout << "\n\n Correct use of NPV example, page 510-511" << std::endl
    ;
```

```
std::cout << "\n\nPresent value of revenue per unit: $" <<
    PVrevenuePerUnit << std::endl;
std::cout << "Present value of cost per unit : $" << PVcostPerUnit +
    initialCost << std::endl;
double dividendYield = originalRiskFreeRate - originalLeaseRate;
std::cout << "Dividend yield              : " << dividendYield << std::
    endl;
double interestSavings = (initialCost + costPerUnit/originalRiskFreeRate)
    *originalRiskFreeRate;
std::cout << "Interest savings            : $" << interestSavings << std
    ::endl;
FewStars();


//////////////////
std::cout << "\n\n Project as an option, page 511-512" << std::endl;
std::cout << " Single unit production under certainty" << std::endl;

double Spot = 0;
double Strike = 0;
double Volatility = 0;
originalLeaseRate = 0;
originalRiskFreeRate = 0;

std::cout << "\nSpot:                  : _\b";
std::cin >> Spot;
std::cout << "Strike:               : _\b";
std::cin >> Strike;
std::cout << "Volatility:           : _\b";
std::cin >> Volatility;
std::cout << "Risk Free Rate(simple rate): _\b";
std::cin >> originalRiskFreeRate;
std::cout << "Lease Rate(simple rate) : _\b";
```

```
  std::cin >> originalLeaseRate;


  double LogLeaseRate = log(1+originalRiskFreeRate)-log(1+originalLeaseRate
      );
  double LogRiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffCall thePayOff(Strike, Volatility, LogRiskFreeRate,
      LogLeaseRate);
  VanillaOption theOption(thePayOff);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption, Spot);

  double optimalN = log(originalLeaseRate*perpetualOptionPrice/(Spot*(
      originalRiskFreeRate - originalLeaseRate)))/log(1+originalLeaseRate);

  std::cout << "\n\nPerpetual Option Price is : $_\b" <<
      perpetualOptionPrice << std::endl;
  std::cout << "Price per unit           : $_\b" << originalLeaseRate*
      perpetualOptionPrice << std::endl;
  std::cout << "Optimal time to wait to invest: _\b" << optimalN;
ManyStars();
}


void exampleSimpleDCF(){
  std::cout << "Simple Discounted CashFlows, page 513-515" << std::endl;
  std::cout << " The expected return on an asset that has the same risk
      profile as the project" << std::endl;

  double RiskFreeRate = .06;
  double MarketReturn = .10;
  double Beta = 1.25;
  double p = .60;
```

```
  double T = 1;
  double Xu = 120;
  double Xd = 80;
  double alpha = RiskFreeRate + Beta*(MarketReturn-RiskFreeRate);
  double ExpectedCF = p*Xu + (1-p)*Xd;
  double PresentValue = (ExpectedCF)/pow(1+alpha,T);
  std::cout << "\n The present value of the asset is: " << PresentValue <<
      std::endl;

  double InitialCost = 10;
  double AdditionalCost = 95;

  double forwardPrice = PresentValue*(pow((1+RiskFreeRate),T));
  std::cout << " Forward price is: " << forwardPrice << std::endl;
  double pStar = (forwardPrice-Xd)/(Xu-Xd);
  std::cout << " pStar is: " << pStar << std::endl;
  double projectValue = (pStar*max(0.0,Xu-AdditionalCost)+(1-pStar)*max
      (0.0,Xd-AdditionalCost))/(1+RiskFreeRate)-InitialCost;
  std::cout << " Project Value is: " << projectValue << std::endl;
  double NPV = PresentValue - InitialCost - AdditionalCost/(1+RiskFreeRate)
      ;

  std::cout << "\n\nIf the decision maker commits to paying $" <<
      InitialCost << " at time 0" << std::endl;
  std::cout << " then the NPV is: " << NPV << std::endl;
  ManyStars();
}


void exampleEvaluatingProject2YearInvestment(){
  std::cout << "Evaluating Project With a 2 Year Investment Horizon, page
      515-516\n" << std::endl;
  //calculating RNPs
```

```
double originalRiskFreeRate = .07;
double MarketReturn = .13;
double Beta = 1.33;
double InitialCost = 100;

double alpha = originalRiskFreeRate + Beta*(MarketReturn-
    originalRiskFreeRate);
double ExpectedAnnualCF = 18;
double AnnualCFGrowthRate = .03;
double PresentValue = (ExpectedAnnualCF)/(alpha - AnnualCFGrowthRate);
std::cout << "The present value of the project is: " << PresentValue <<
    std::endl;
std::cout << "The present value of the cost of the project is: " <<
    InitialCost << std::endl;
std::cout << "Therefore the Static NPV of the asset is: " << PresentValue
     - InitialCost << std::endl;
double InterestSavings = InitialCost*originalRiskFreeRate;
std::cout << "Interest Savings from foregoing initial cost is: " <<
    InterestSavings << std::endl;
std::cout << "Dividend of the project that is foregone is: " <<
    ExpectedAnnualCF << std::endl;

std::cout << "\nWithout including the value of insurance the project ";
if(ExpectedAnnualCF > InterestSavings){
std::cout << "should be: " << std::endl;
std::cout << "  started immediateiy since the dividend is greater " <<
    std::endl;
std::cout << "  than the interest savings from foregoing the intial cost"
     << std::endl;
} else if(ExpectedAnnualCF < InterestSavings){
std::cout << "should be delayed since: " << std::endl;
std::cout << "The dividend is less than the interest savings from
    foregoing the intial cost" << std::endl;
```

```
  } else{
  std::cout << "may be started now or later since: " << std::endl;
  std::cout << "The dividend is equal to the interest savings from
     foregoing the intial cost" << std::endl;
  }

  FewStars();
  std::cout << "\n\nTwo Year Horizon Project using Perpetual Option" << std
     ::endl;
  double Spot = PresentValue;
  double Strike = InitialCost;
  double Volatility = 0.5;
  double RiskFreeRate = log(1+originalRiskFreeRate);
  double LeaseRate = log(1+ExpectedAnnualCF/PresentValue);

  std::cout << "\nRiskFreeRate: " << RiskFreeRate << std::endl;
  std::cout << "LeaseRate: " << LeaseRate << std::endl;

  PerpetualPayOffCall thePayOff1a(Strike, Volatility, RiskFreeRate,
     LeaseRate);
  VanillaOption theOption1a(thePayOff1a);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption1a, Spot);
  double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);

  FewStars();
  std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
     std::endl;
  std::cout << "The value of the oil well at extraction: " << H << std::
     endl;
  ManyStars();
}
```

```
//////////

void exampleSingleBarrelCertainty(){
  std::cout << "Single Barrel Certainty, page 525-527\n" << std::endl;
//  std::cout << "Single-barrel extraction under certainty" << std::endl;
  std::cout << "\n\nThe value and appreciation of the land" << std::endl;
  double Spot = 0;
  double Strike = 0;
  double originalRiskFreeRate = 0;
  double originalLeaseRate = 0;

  std::cout << "\nSpot:                    : _\b";
  std::cin >> Spot;
  std::cout << "Strike:                 : _\b";
  std::cin >> Strike;
  std::cout << "Risk Free Rate(simple rate): _\b";
  std::cin >> originalRiskFreeRate;
  std::cout << "Lease Rate(simple rate) : _\b";
  std::cin >> originalLeaseRate;


  double LogRiskFreeRate = log(1+originalRiskFreeRate);
  double LogLeaseRate = log(1+originalLeaseRate);
  double LogRiskFree_LeaseRate_Ratio = LogRiskFreeRate/LogLeaseRate;
  double St = LogRiskFree_LeaseRate_Ratio * Strike;
  std::cout << "\n\nThe optimal price is    : $" << St << std::endl;
  double RiskFree_LeaseRate_Ratio = (1+originalRiskFreeRate)/(1+
     originalLeaseRate);
  std::cout << "The RiskFree_LeaseRate_Ratio: " << RiskFree_LeaseRate_Ratio
      << std::endl;
  double optTime = log(St/Spot)/log(RiskFree_LeaseRate_Ratio);
  std::cout << "The optimal time is     : " << optTime << std::endl;
  double optimalValue = St-Strike;
```

```
  std::cout << "At this point the value of extraction will be: " <<
     optimalValue << std::endl;
  double TodaysNPV = (St - Strike)/pow(1+originalRiskFreeRate,optTime);
  std::cout << "Today's NPV will be     : $" << TodaysNPV << std::endl;

  FewStars();
  std::cout << "\n\nChanging Extraction Costs" << std::endl;
  double ChangingCost = 0;
  std::cout << "Changing cost              : _\b";
  std::cin >> ChangingCost;
  double logG = log(1+ChangingCost);
  double ChangingRiskFree_LeaseRate_Ratio = ((LogRiskFreeRate - logG)/
     LogLeaseRate);
  std::cout << "The LogRiskFree_LeaseRate_Ratio: " <<
     ChangingRiskFree_LeaseRate_Ratio << std::endl;
  double StChange = ChangingRiskFree_LeaseRate_Ratio * Strike;
  std::cout << "The optimal spot price given increasing extraction costs: "
       << StChange << std::endl;
  double optTimeG = log(StChange/Spot)/log(RiskFree_LeaseRate_Ratio);
  std::cout << "The optimal time is        : " << optTimeG << " years" <<
     std::endl;
  double optimalValueG = StChange-Strike;
  std::cout << "Value of extraction will be : $" << optimalValueG << std::
     endl;
  double TodaysNPVG = (StChange - Strike)/pow(1+originalRiskFreeRate,
     optTimeG);
  std::cout << "Today's NPV will be        : $" << TodaysNPVG << std::endl;
  ManyStars();
}

////////////
void exampleSingleBarrelCertaintyPO(){
```

```
std::cout << "Single Barrel Certainty Perpetual Option, page 527-528\n"
   << std::endl;
double Spot = 0;
double Strike = 0;
double Volatility = 0;
double originalRiskFreeRate = 0;
double originalLeaseRate = 0;

std::cout << "\nSpot:                    : _\b";
std::cin >> Spot;
std::cout << "Strike:                 : _\b";
std::cin >> Strike;
std::cout << "Volatility:             : _\b";
std::cin >> Volatility;
std::cout << "Risk Free Rate(simple rate): _\b";
std::cin >> originalRiskFreeRate;
std::cout << "Lease Rate(simple rate) : _\b";
std::cin >> originalLeaseRate;


double LeaseRate = log(1+originalLeaseRate);
double RiskFreeRate = log(1+originalRiskFreeRate);

PerpetualPayOffCall thePayOff1(Strike, Volatility, RiskFreeRate,
   LeaseRate);
VanillaOption theOption1(thePayOff1);
double perpetualOptionPrice = PerpetualOptionPricer(theOption1, Spot);
double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
double H = HFinder(Strike, h);

FewStars();
std::cout << "\n\nSingle-barrel extraction under certainty" << std::endl;
```

```
  std::cout << "\nPerpetual Option Price is       : $" <<
      perpetualOptionPrice << std::endl;
  std::cout << "The value of the oil well at extraction: $" << H << std::
      endl;
  ManyStars();
}

//////////////////////////////

void exampleSingleBarrelUncertainty(){
  std::cout << "Single Barrel Uncertainty Perpetual Option, page, 528-530\n
      " << std::endl;
  double originalLeaseRate = .04;
  double originalRiskFreeRate = .05;

  double Spot = 15;
  double Strike = 13.60;
  double Volatility = .15;
  double LeaseRate = log(1+originalLeaseRate);
  double RiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffCall thePayOff2(Strike, Volatility, RiskFreeRate,
      LeaseRate);
  VanillaOption theOption2(thePayOff2);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption2, Spot);
  double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);

  FewStars();
  std::cout << "\n\nSingle-barrel extraction under uncertainty" << std::
      endl;
  std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
      std::endl;
```

```
  std::cout << "The value of the oil well at extraction: " << H << std::
      endl;
  ManyStars();
}

void exampleInfiniteWell173(){
  std::cout << "Valuing an Infinite Oil Reserve" << std::endl;
  std::cout << " Example 17.3 - near zero volatility, page 531\n" << std::
      endl;
  double Spot = 15;
  double ShutdownRestart = 180;
  double Volatility = .00001;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
  double c = 8;

  Spot = Spot/originalLeaseRate;
  double Strike = c/originalRiskFreeRate+ShutdownRestart;
  double LeaseRate = log(1+originalLeaseRate);
  double RiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffCall thePayOff3(Strike, Volatility, RiskFreeRate,
      LeaseRate);
  VanillaOption theOption3(thePayOff3);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption3, Spot);
  double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);

  FewStars();
  std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
      std::endl;
  std::cout << "The value of the oil well at extraction: " << H << std::
      endl;
```

```
  std::cout << "Extraction occurs when spot equals: " << originalLeaseRate*
      H << std::endl;

  ManyStars();
}


void exampleInfiniteWell174(){
  std::cout << "Valuing an Infinite Oil Reserve" << std::endl;
  std::cout << " Example 17.4 - 0.15 volatility, page 531\n" << std::endl;
  std::cout << "\n\nValuing an infinite well" << std::endl;
  double Spot = 15;
  double ShutdownRestart = 180;
  double Volatility = .15;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
  double c = 8;

  Spot = Spot/originalLeaseRate;
  double Strike = c/originalRiskFreeRate+ShutdownRestart;
  double LeaseRate = log(1+originalLeaseRate);
  double RiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffCall thePayOff4(Strike, Volatility, RiskFreeRate,
      LeaseRate);
  VanillaOption theOption4(thePayOff4);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption4, Spot);
  double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);

  std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
      std::endl;
```

```
  std::cout << "The value of the oil well at extraction: " << H << std::
      endl;
  std::cout << "Extraction occurs when spot equals: " << originalLeaseRate*
      H << std::endl;
  std::cout << "Per barrel the well is worth: " << originalLeaseRate*
      perpetualOptionPrice << std::endl;

  ManyStars();
}


void exampleInfiniteWellPermShutdown175A(){
  std::cout << "Valuing an Infinite Oil Reserve with Permanent Shutdown" <<
       std::endl;
  std::cout << " Example 17.5A - zero shutdown cost, page 533\n" << std::
      endl;
  //Example 17.5
  std::cout << "\n\nCommodity Extraction with Shutdown and Restart Options"
       << std::endl;
  std::cout << "When a permanent one-time shutdown of the well is possible"
       << std::endl;

  double Spot = 10;
  double ShutdownRestart = 0;
  double Volatility = .15;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
  double c = 8;

  Spot = Spot/originalLeaseRate;
  double Strike = c/originalRiskFreeRate+ShutdownRestart;
  double LeaseRate = log(1+originalLeaseRate);
  double RiskFreeRate = log(1+originalRiskFreeRate);
```

```
  PerpetualPayOffPut thePayOff5(Strike, Volatility, RiskFreeRate, LeaseRate
      );
  VanillaOption theOption5(thePayOff5);
  double h = hPutFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption5, Spot);

  std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
      std::endl;
  std::cout << "The value of the oil well at extraction: " << H << std::
      endl;
  std::cout << "Extraction/Shutdown occurs when spot equals: " <<
      originalLeaseRate*H << std::endl;
  std::cout << "Per barrel extraction cost: " << originalLeaseRate*Strike
      << std::endl;
  std::cout << "The value of shutting down at " << H*originalLeaseRate << "
      : " << Strike - H << std::endl;
  std::cout << "Loss avoided/Gain had for shutdown/restart: " << H*
      originalLeaseRate/originalLeaseRate - c/originalRiskFreeRate << std::
      endl;
  //std::cout << "The value of option is same amount saved" << std::endl;


  ManyStars();
}



void exampleInfiniteWellPermShutdown175B(){
  std::cout << "Valuing an Infinite Oil Reserve with Permanent Shutdown" <<
       std::endl;
```

```
std::cout << " Example 17.5B - 15 shutdown cost, page 534\n" << std::endl
    ;
double Spot = 10;
double ShutdownRestart = -25;
double Volatility = .15;
double originalRiskFreeRate = .05;
double originalLeaseRate = .04;
double c = 8;

Spot = Spot/originalLeaseRate;
double Strike = c/originalRiskFreeRate+ShutdownRestart;
double LeaseRate = log(1+originalLeaseRate);
double RiskFreeRate = log(1+originalRiskFreeRate);

PerpetualPayOffPut thePayOff6(Strike, Volatility, RiskFreeRate, LeaseRate
    );
VanillaOption theOption6(thePayOff6);
double h = hPutFinder(Volatility, RiskFreeRate, LeaseRate);
double H = HFinder(Strike, h);
double perpetualOptionPrice = PerpetualOptionPricer(theOption6, Spot);

FewStars();

std::cout << "\nPerpetual Option Price is: " << perpetualOptionPrice <<
    std::endl;
std::cout << "The value of the oil well at extraction: " << H << std::
    endl;
std::cout << "Extraction occurs when spot equals: " << originalLeaseRate*
    H << std::endl;
std::cout << "Per barrel extraction cost: " << originalLeaseRate*Strike
    << std::endl;
std::cout << "Loss avoided/Gain had for shutdown/restart: " << H*
    originalLeaseRate/originalLeaseRate - c/originalRiskFreeRate << std::
```

```
    endl;
  std::cout << "Shutdown occurs when spot equals: " << originalLeaseRate*H
    << std::endl;
  if(ShutdownRestart<0){
  std::cout << "We pay " << ShutdownRestart << " to avoid ";
  std::cout << "losses of " << H*originalLeaseRate/originalLeaseRate - c/
    originalRiskFreeRate << " by shutting down." << std::endl;
  } else if(ShutdownRestart>0){
  std::cout << "We pay " << ShutdownRestart << " to ";
  std::cout << "gain " << H*originalLeaseRate/originalLeaseRate - c/
    originalRiskFreeRate << " by continuiing operations." << std::endl;
  }

  ManyStars();
}

/*
void graph(){

  double Spot = 10;
  double ShutdownRestart = -25;
  double Volatility = .15;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
  double c = 8;

  Spot = Spot/originalLeaseRate;
  double Strike = c/originalRiskFreeRate+ShutdownRestart;
  double LeaseRate = log(1+originalLeaseRate);
  double RiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffPut thePayOff6(Strike, Volatility, RiskFreeRate, LeaseRate
    );
```

```
  VanillaOption theOption6(thePayOff6);
  double h = hPutFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption6, Spot);

  double functionProducingWell = Spot/LeaseRate - c/RiskFreeRate +
      perpetualOptionPrice;

}
*/


void exampleInfiniteWellPermShutdownOptimalPermShutdown176(){
  std::cout << "Valuing an Infinite Oil Reserve with Optimal Permanent
      Shutdown" << std::endl;
  std::cout << " Example 17.6, page 536\n" << std::endl;
  double Spot = 0;
  double MaxSpot = 0;
  double ShutdownRestart = 0;
  double preShutdownRestart = 0;
  std::cout << "\nCurrent Spot: __\b\b";
  std::cin >> Spot;
  std::cout << "Future Max Spot: __\b\b";
  std::cin >> MaxSpot;
  std::cout << "Shutdown Strike: __\b\b";
  std::cin >> preShutdownRestart;
  ShutdownRestart = preShutdownRestart*-1;
  //double Spot = 20;
  //double MaxSpot = 25.21;
  //double ShutdownRestart = 0;
  double Volatility = .15;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
```

```
double c = 8;
double I = 180;
double POMaxSpot = MaxSpot/originalLeaseRate;

double Strike = c/originalRiskFreeRate+ShutdownRestart;
double LeaseRate = log(1+originalLeaseRate);
double RiskFreeRate = log(1+originalRiskFreeRate);

PerpetualPayOffPut thePayOff7(Strike, Volatility, RiskFreeRate, LeaseRate
    );
VanillaOption theOption7(thePayOff7);
double hCall = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
double perpetualOptionPrice1 = PerpetualOptionPricer(theOption7, Spot);
double perpetualOptionPrice23 = PerpetualOptionPricer(theOption7,
    POMaxSpot);

FewStars();

std::cout << "Equation 1:" << std::endl;
std::cout << "Current value of the producing well given a shutdown option
     and current spot: " << std::endl;
std::cout << "Spot/LeaseRate - c/RiskFreeRate + perpetualOptionPrice = ";
std::cout << Spot/originalLeaseRate - c/originalRiskFreeRate +
    perpetualOptionPrice1 << std::endl;

FewStars();

std::cout << "Equation 2:" << std::endl;
std::cout << "Future value of the producing well given a shutdown option
    and max spot: " << std::endl;
std::cout << "MaxSpot/LeaseRate - c/RiskFreeRate + perpetualOptionPrice -
     I = ";
```

```
  std::cout << Spot/originalLeaseRate - c/originalRiskFreeRate +
      perpetualOptionPrice23 - I << std::endl;

  FewStars();

  std::cout << "Equation 3:" << std::endl;
  std::cout << "The present value of the producing well given a shutdown
      option and max spot: " << std::endl;
  std::cout << "pow(Spot/MaxSpot,hCall)*(MaxSpot/LeaseRate - c/RiskFreeRate
       + perpetualOptionPrice - I) = ";
  std::cout << pow(Spot/MaxSpot,hCall)*(MaxSpot/originalLeaseRate - c/
      originalRiskFreeRate + perpetualOptionPrice23 - I) << std::endl;

  ManyStars();
}



void exampleInfiniteWellPermRestart177(){
  std::cout << "Valuing an Infinite Oil Reserve with Permanent Restart" <<
      std::endl;
  std::cout << " Example 17.7, page 536\n" << std::endl;
  double Spot = 10;
  double ShutdownRestart = 0;
  double Volatility = .15;
  double originalRiskFreeRate = .05;
  double originalLeaseRate = .04;
  double c = 8;

  Spot = Spot/originalLeaseRate;
  double Strike = c/originalRiskFreeRate+ShutdownRestart;
  double LeaseRate = log(1+originalLeaseRate);
```

```
  double RiskFreeRate = log(1+originalRiskFreeRate);

  PerpetualPayOffCall thePayOff8(Strike, Volatility, RiskFreeRate,
      LeaseRate);
  VanillaOption theOption8(thePayOff8);
  double perpetualOptionPrice = PerpetualOptionPricer(theOption8, Spot);
  double h = hCallFinder(Volatility, RiskFreeRate, LeaseRate);
  double H = HFinder(Strike, h);

  std::cout << "\nValue of the well is: " << perpetualOptionPrice << std::
      endl;
  std::cout << "Extraction restart occurs when spot equals: " <<
      originalLeaseRate*H << std::endl;

  ManyStars();
}




void whiteSpace(){
  //int whiteSpaces = 4500;
  //const std::string whiteSpace = " ";
  const std::string createWhiteSpace(1000, ' ');
  std::cout << createWhiteSpace << std::endl;
  int backSpaces = 0;
  for (int i=0; i < backSpaces; i++){
    std::cout << "\b";
  }
}

void ManyStars(){
  //Creating a border
```

```
  std::cout << std::endl;
  char star = '*';
  for(int i=0; i < 80; i++){
    std::cout << star;
  }
  std::cout << std::endl;
}

void FewStars(){
  //Creating a border
  std::cout << std::endl;
  char star = '*';
  char spaces = ' ';
  for(int i=0; i < 4; i++){
    std::cout << spaces;
  }
  for(int j=0; j < 69; j++){
    std::cout << star;
  }
  for(int k=0; k < 4; k++){
    std::cout << spaces;
  }
  std::cout << std::endl;
}
```

```
/////////////////      PerpetualPayOff.hpp

#ifndef PERPETUALPAYOFF_HPP
#define PERPETUALPAYOFF_HPP
```

```
#include "VanillaPayOff.hpp"


class PerpetualPayOffCall : public PayOff
{
public:

    PerpetualPayOffCall(double Strike_, double v_, double r_, double d_);

    virtual double operator()(double Spot) const;
    virtual ~PerpetualPayOffCall(){}
    virtual PayOff* clone() const;

private:

    double Strike;
    double v;
    double r;
    double d;

};


class PerpetualPayOffPut : public PayOff
{
public:

    PerpetualPayOffPut(double Strike_, double v_, double r_, double d_);

    virtual double operator()(double Spot) const;
    virtual ~PerpetualPayOffPut(){}
    virtual PayOff* clone() const;
```

```
private:

    double Strike;
    double v;
    double r;
    double d;

};

#endif



//////////////////     PerpetualPayOff.cpp


#include "PerpetualPayOff.hpp"
#include <cmath>

PerpetualPayOffCall::PerpetualPayOffCall(double Strike_, double v_, double
    r_, double d_) : Strike(Strike_), v(v_), r(r_), d(d_) {}

double PerpetualPayOffCall::operator () (double Spot) const {
  double sigmaSqrd = v*v;
  double h = 0.5 - ((r - d) / sigmaSqrd) + sqrt(pow(((r - d) / sigmaSqrd) -
      0.5, 2) + (2 * r / sigmaSqrd));
  double H = Strike*h / (h - 1);
  double optval = (H - Strike)*pow((Spot / H), h);

if (Spot > H)
 optval = Spot - Strike;
```

```
return optval;
}

PayOff* PerpetualPayOffCall::clone() const
{
    return new PerpetualPayOffCall(*this);
}




PerpetualPayOffPut::PerpetualPayOffPut(double Strike_, double v_, double r_
    , double d_) : Strike(Strike_), v(v_), r(r_), d(d_) {}


double PerpetualPayOffPut::operator () (double Spot) const {
  double sigmaSqrd = v*v;
  double h = 0.5 - ((r - d) / sigmaSqrd) - sqrt(pow(((r - d) / sigmaSqrd) -
      0.5, 2) + (2 * r / sigmaSqrd));
  double H = Strike*h / (h - 1);
  double optval = (H - Strike)*pow((Spot / H), h);

if (H > Spot)
 optval = Strike - Spot;

return std::abs(optval);
}


PayOff* PerpetualPayOffPut::clone() const
{
    return new PerpetualPayOffPut(*this);
}
```

```
//////////   AnalystPackage.hpp

#ifndef HFINDER_HPP
#define HFINDER_HPP

double hCallFinder(double v, double r, double d);
double hPutFinder(double v, double r, double d);
double HFinder(double Strike, double h);

#endif
```

```
//////////    AnalystPackage.cpp

#include "AnalystPackage.hpp"
#include <cmath>

double hCallFinder(double v, double r, double d){
    double sigmaSqrd = v*v;
    double h = 0.5 - ((r - d) / sigmaSqrd) + sqrt(pow(((r - d) / sigmaSqrd)
        - 0.5, 2) + (2 * r / sigmaSqrd));

    return h;
}

double hPutFinder(double v, double r, double d){
    double sigmaSqrd = v*v;
```

```
    double h = 0.5 - ((r - d) / sigmaSqrd) - sqrt(pow(((r - d) / sigmaSqrd)
        - 0.5, 2) + (2 * r / sigmaSqrd));

    return h;
}

double HFinder(double Strike, double h){
  double H = Strike*h / (h - 1);

  return H;
}



///////////////          VanillaPayOff.hpp


#ifndef VANILLAPAYOFF_HPP
#define VANILLAPAYOFF_HPP

class PayOff
{
public:

    PayOff(){};

    virtual double operator()(double Spot) const=0;
    virtual ~PayOff(){}
    virtual PayOff* clone() const=0;

private:

};
```

```
class PayOffCall : public PayOff
{
public:

    PayOffCall(double Strike_);

    virtual double operator()(double Spot) const;
    virtual ~PayOffCall(){}
    virtual PayOff* clone() const;

private:

    double Strike;

};


class PayOffPut : public PayOff
{
public:

    PayOffPut(double Strike_);

    virtual double operator()(double Spot) const;
    virtual ~PayOffPut(){}
    virtual PayOff* clone() const;

private:

    double Strike;

};
```

```
#endif

//Copyright (c) 2002  Mark Joshi




///////////////        VanillaPayOff.cpp

#include "VanillaPayOff.hpp"
#include "minmax.hpp"

PayOffCall::PayOffCall(double Strike_) : Strike(Strike_)
{
}

double PayOffCall::operator () (double Spot) const
{
    return max(Spot-Strike,0.0);
}

PayOff* PayOffCall::clone() const
{
    return new PayOffCall(*this);
}



double PayOffPut::operator () (double Spot) const
{
    return max(Strike-Spot,0.0);
}
```

```
PayOffPut::PayOffPut(double Strike_) : Strike(Strike_)
{
}

PayOff* PayOffPut::clone() const
{
    return new PayOffPut(*this);
}

//Copyright (c) 2002  Mark Joshi




//////////////        PayOffBridge.hpp

#ifndef PAYOFFBRIDGE_HPP
#define PAYOFFBRIDGE_HPP

#include "VanillaPayOff.hpp"

class PayOffBridge
{
public:

    PayOffBridge(const PayOffBridge& original);
    PayOffBridge(const PayOff& innerPayOff);

    inline double operator()(double Spot) const;
    ~PayOffBridge();
    PayOffBridge& operator=(const PayOffBridge& original);

private:
```

```
    PayOff* ThePayOffPtr;

};

inline double PayOffBridge::operator()(double Spot) const
{
    return ThePayOffPtr->operator ()(Spot);
}

#endif

//Copyright (c) 2002  Mark Joshi




////////////      PayOffBridge.cpp

#include "PayOffBridge.hpp"

PayOffBridge::PayOffBridge(const PayOffBridge& original)
{
    ThePayOffPtr = original.ThePayOffPtr->clone();
}

PayOffBridge::PayOffBridge(const PayOff& innerPayOff)
{
    ThePayOffPtr = innerPayOff.clone();
}


PayOffBridge::~PayOffBridge()
```

```
{
    delete ThePayOffPtr;
}

PayOffBridge& PayOffBridge::operator=(const PayOffBridge& original)
{
    if (this != &original)
    {
        delete ThePayOffPtr;
        ThePayOffPtr = original.ThePayOffPtr->clone();
    }

    return *this;
}

//Copyright (c) 2002  Mark Joshi
```

```
//////////////            PerpetualOptionPricer.hpp

#ifndef PERPETUALOPTIONPRICER_HPP
#define PERPETUALOPTIONPRICER_HPP

#include "VanillaOption.hpp"

double PerpetualOptionPricer(const VanillaOption& TheOption, double Spot);

#endif
```

```
//Copyright (c) 2002  Mark Joshi



//////////        PerpetualOptionPricer.cpp


#include "PerpetualOptionPricer.hpp"
#include <cmath>

double PerpetualOptionPricer(const VanillaOption& TheOption, double Spot)
{

    double result = TheOption.OptionPayOff(Spot);

    return result;
}

//Copyright (c) 2002  Mark Joshi



/////////      VanillaOption.hpp

#ifndef VANILLAOPTION_HPP
#define VANILLAOPTION_HPP

#include "PayOffBridge.hpp"

class VanillaOption
{
public:
```

```
    VanillaOption(const PayOffBridge& ThePayOff_, double Expiry);

    VanillaOption(const PayOffBridge& ThePayOff_);

    double OptionPayOff(double Spot) const;
    double GetExpiry() const;

private:

    double Expiry;
    PayOffBridge ThePayOff;
};

#endif

//Copyright (c) 2002  Mark Joshi



////////////        VanillaOption.cpp

#include "VanillaOption.hpp"

VanillaOption::VanillaOption(const PayOffBridge& ThePayOff_, double Expiry_
    )
                            :  ThePayOff(ThePayOff_), Expiry(Expiry_)
{
}

VanillaOption::VanillaOption(const PayOffBridge& ThePayOff_)
                            :  ThePayOff(ThePayOff_)
{
```

```
}

double VanillaOption::GetExpiry() const
{
     return Expiry;
}

double VanillaOption::OptionPayOff(double Spot) const
{
    return ThePayOff(Spot);
}

//Copyright (c) 2002  Mark Joshi
```