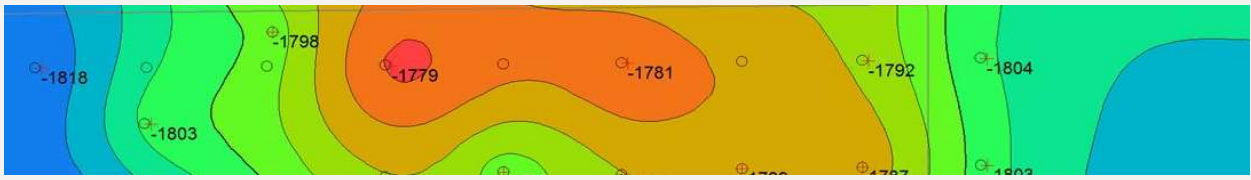
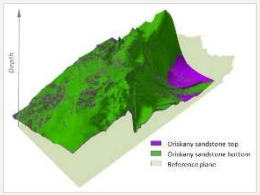
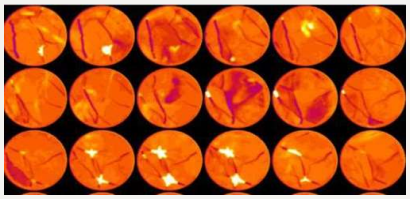
NATIONAL ENERGY TECHNOLOGY LABORATORY




User's Manual for the FE/NETL Onshore CO_2 EOR Cost Model, Version 1

July 31, 2020
DOE/NETL-2020/2614

Disclaimer

User's Manual for the FE/NETL Onshore CO_2 EOR Cost Model, Version 1

David Morgan[1], Donald Remson[1], Vello Kuuskraa[2], and Matthew Wallace[2]

[1]National Energy Technology Laboratory (NETL)

[2]Advanced Resources International (ARI)

The reviewers and editors for this report were:

Hannah Hoffman, KeyLogic Systems, LLC

The contacts for this report are:

Donald Remson
NETL SubCLIN COR
412.386.5379
donald.remson@netl.doe.gov

Derek Vikara
KeyLogic Systems, LLC
412.386.7409
derek.vikara@netl.doe.gov

Allison Guinan
Leidos
412.386.6855
allison.guinan@netl.doe.gov

This report was prepared by MESA for the U.S. DOE NETL. This work was completed under DOE NETL Contract Number DE-FE0025912. This work was performed under MESA Activity 205.003.

DOE Contract Number DE-FE0025912

This page intentionally left blank.

TABLE OF CONTENTS

ACRONYMS AND ABBREVIATIONS

3-D	Three dimensional
ac	acres
API	American Petroleum Institute oil gravity
ARI	Advanced Resources International
atm	atmosphere
bbl	Barrel
BY$	Base year dollars
CO_2	Carbon dioxide
cp	centipoise
csv	comma separated values
DOE	Department of Energy
EOR	Enhanced oil recovery
FE	Fossil Energy
ft	feet
ft^2	square feet
ft^3	cubic feet
G&A	General and administrative
HCPV	Hydrocarbon pore volume
in	inches
IRR	Internal rate of return
kg	kilogram
ktonne	kilotonne
L	Liter
lbs	pounds
M	Thousands in English units
mD	milliDarcy
MESA	Mission Execution and Strategic Analysis
mi	mile
mi^2	square miles
MM	Million
mol	Mole
MPa	Megapascal
MRV	Monitoring, Reporting and Verification
NETL	National Energy Technology Laboratory
O&M	Operation and maintenance
OOIP	Original oil in place
patt	Pattern
psia	pressure per square inch absolute
res	reservoir
scf	standard cubic feet
STB	Stock tank barrel
tonne	Metric ton
U.S.	United States
WACC	Weighted average cost of capital
y, yr	year

1 INTRODUCTION

This document is a user's manual for the FE/NETL Onshore CO_2 EOR Cost Model. This document has been prepared by the National Energy Technology Laboratory (NETL) which is part of the Office of Fossil Energy (FE) in the United States (U.S.) Department of Energy (DOE). The focus of this document is on estimating the cost of implementing enhanced oil recovery (EOR) using supercritical carbon dioxide (CO_2).

CO_2 EOR is a tertiary oil recovery technique, where primary recovery is the initial phase of pumping oil from an oil reservoir and secondary recovery involves injecting water to re-pressurize the oil reservoir and produce additional oil. CO_2 EOR is typically implemented in oil fields where the reservoir temperatures and pressures create supercritical conditions for the CO_2 allowing miscibility between the oil and CO_2. The term "miscibility" means oil and supercritical CO_2 will dissolve in each other in all proportions. In practice, no oil is completely miscible in supercritical CO_2, but some oils are almost completely miscible in CO_2. These are the oils that are the target of most CO_2 EOR operations.

The FE/NETL Onshore CO_2 EOR Cost Model performs a cash flow analysis for an oil field where CO_2 EOR is implemented—either at an existing oil field operation following primary and secondary recovery (brownfield) or at an oil field with no prior recovery operation (greenfield). The model includes capital costs, operation and maintenance (O&M) costs and costs associated with financing. The model performs cash flow analysis both before and after federal income tax. Given a user-supplied oil price, the model will calculate the return on investment before and after federal income taxes. Alternatively, given a minimum desired return on investment after federal income taxes, the model will calculate the break-even oil price or break-even CO_2 price. Given a CO_2 price, the break-even oil price is the lowest oil price that gives this desired minimum return on investment. Similarly, given an oil price, the break-even CO_2 price is the CO_2 price that gives this desired minimum return on investment.

The FE/NETL Onshore CO_2 EOR Cost Model is a Fortran 90/95/2003 computer program. The result of compiling a Fortran program is an executable file and the expression "CO2EORCMOn" is used as an abbreviation for the FE/NETL CO_2 EOR Cost Model in the name of the executable file. This abbreviation is occasionally used in this document to refer to the FE/NETL Onshore CO_2 EOR Cost Model.

To perform a cash flow analysis of oil field operations, the performance of CO_2 EOR at an oil field must be simulated. The FE/NETL Onshore CO_2 EOR Cost Model does not directly simulate the performance of CO_2 EOR at an oil field but uses the output from a program called StrmtbFlow. This program is part of the FE/NETL CO_2 Prophet Model, which is an upgraded version of the CO_2 Prophet program originally developed in the 1990s by the Texaco Exploration and Production Technology Department for DOE. The FE/NETL CO_2 Prophet Model is a reservoir simulation model that simulates CO_2 EOR at the pattern level. StrmtbFlow simulates the pattern-level output fluid flows (oil, hydrocarbon gas, water, and CO_2) given user-specified input fluid flows (water and CO_2). The FE/NETL Onshore CO_2 EOR Cost Model scales these pattern-level input and output fluid flows to the entire oil field level, which can consist of several to many patterns. [1] [2]

This user's manual describes the operation of the FE/NETL Onshore CO_2 EOR Cost Model along with the file structure, inputs, outputs, and associated costs used in the model. The document assumes the required output file from StrmtbFlow is in the appropriate directory so that the FE/NETL Onshore CO_2 EOR Cost Model is ready to run.

The different sections included in this user manual are described below:

- **Section 2 – Basic Operation of the FE/NETL Onshore CO2 EOR Cost Model**: This section provides instructions for installing the FE/NETL Onshore CO_2 EOR Cost Model software, using command prompts to call the cost model executable program, and running the cost model.
- **Section 3 – File Structure:** This section outlines the input and output files associated with the FE/NETL Onshore CO_2 EOR Cost Model program.
- **Section 4 – Mathematical Basis:** This section presents the mathematical foundation for the FE/NETL Onshore CO_2 EOR Cost Model, including all capital, operating, and financial cost calculations and associated variables.
- **Section 5 – Inputs to the FE/NETL Onshore CO2 EOR Cost Model:** This section includes a list of the model input files and briefly describes their contents.
- **Section 6 – Outputs from the FE/NETL Onshore CO2 EOR Cost Model:** This section identifies the output file generated by the FE/NETL Onshore CO_2 EOR Cost Model and briefly describes the contents of this file.
- **Section 7 – References:** This section lists the articles and publications cited in this user's manual.

2 BASIC OPERATION OF THE FE/NETL ONSHORE CO_2 EOR COST MODEL

The FE/NETL Onshore CO_2 EOR Cost Model is a Fortran program developed principally for the Microsoft Windows operating system and has only been tested on the Microsoft Windows operating system. The program is run from a "command prompt" window within Microsoft Windows. The "command prompt" window is essentially a version of the old DOS operating system.

Before attempting to execute the FE/NETL Onshore CO_2 EOR Cost Model, the user should establish a file folder where the FE/NETL Onshore CO_2 EOR Cost Model will be stored. The Fortran source code, executable file "CO2EORCMOn_gf.exe", and input files ("Cost_in.dat", "Gen_in.dat", and "ystrmtbflow_outyr.csv") should all be placed in this folder.

The FE/NETL Onshore CO_2 EOR Cost Model has been compiled using the Gnu Fortran (gfortran). The phrase "gf" in the name of the included executable file "CO2EORCMOn_gf.exe" indicates the FE/NETL Onshore CO_2 EOR Cost Model was compiled and linked using gfortran. While the user should be able to compile and link the FE/NETL Onshore CO_2 EOR Cost Model with other Fortran compilers, the NETL developers have not tried to compile and link the FE/NETL Onshore CO_2 EOR Cost Model with other Fortran compilers.

After the FE/NETL Onshore CO_2 EOR Cost Model has been installed in the desired file folder and compiled into an executable file (if the user does not use the included executable file "CO2EORCMOn_gf.exe"), the user can run the program as follows.

- The user should open a "command prompt" window.
- The user should then change directories so that the command prompt is in the file folder where the FE/NETL Onshore CO_2 EOR Cost Model is located. For example, if the files for the FE/NETL Onshore CO_2 EOR Cost Model are in the directory "C:\CO2EOR_CostMod", then the user should change directories by typing the following commands at the command prompt. If the command prompt is not at the C: drive (for example, if the command prompt displays "D:\>"), then type the following command at the command prompt:

  C:

  The command prompt should now display:

  C:\>

  The user should now type the following command at the command prompt (it is not necessary to capitalize commands since lower and uppercase letters are treated as the same in Windows):

  cd CO2EOR_CostMod

  The command prompt should now display:

  C:\CO2EOR_CostMod>

- The user should modify inputs in the text input file "Gen_in.dat" using a text editor (not a word processor) so the inputs reflect the characteristics of their oil field. This file must reside in the same directory as the FE/NETL Onshore CO_2 EOR Cost Model executable file. This input file is described in more detail in Section 5.
- The user should also modify the text file "Cost_in.dat" if the user has updated cost information. This file has general unit cost data and the user should only update this file if they have more appropriate cost information relevant to their oil field. This file must reside in the same directory as the FE/NETL Onshore CO_2 EOR Cost Model executable file. This input file is described in more detail in Section 5.
- Prior to running the FE/NETL Onshore CO_2 EOR Cost Model, the user needs to run the stream tube reservoir simulation program StrmtbFlow. StrmtbFlow is part of the FE/NETL CO_2 Prophet Model. StrmtbFlow is a pattern-level reservoir simulator that will simulate the application of CO_2 EOR for a pattern at an oil field. One of the output files generated by StrmtbFlow is "ystrmtbflow_outyr.csv". This output file works directly with the FE/NETL Onshore CO_2 EOR Cost Model and needs to be copied and placed in the same directory as the FE/NETL Onshore CO_2 EOR Cost Model executable file.
- The user should then type the name of the executable file at the command prompt. The executable file will have an .exe extension. This extension does not need to be included. The name of the executable file is "CO2EORCMOn_gf.exe", so the user should type the following at the command prompt:

  CO2EORCMOn_gf

  At the conclusion of its execution, the FE/NETL Onshore CO_2 EOR Cost Model returns a command prompt (e.g., C:\C CO2EOR_CostMod>).

  Note: Windows ignores the case of the letters in the file name, so the expressions "CO2EORCMOn_gf" and "co2eorcmon_gf" are equivalent.
- The FE/NETL Onshore CO_2 EOR Cost Model generates a single output file named "CO2EORCMOn_det_out.csv". This output file is a text file in comma separated values (csv) format. The file can be examined using a text editor or opened using Excel since Excel can directly open files with a csv extension. The Excel program places all lines of output in a row and places all results between commas in a cell.

3 FILE STRUCTURE

The FE/NETL Onshore CO_2 EOR Cost Model uses several text files for input and output.

The text file "Gen_in.dat" is an input file. This file includes many variables that should be tailored to the specific oil field and needs of the user. This file is described in Section 5.

The text file "Cost_in.dat" is another input file. This file provides generic unit costs and default values for variables used in equations to generate unit costs. If the user has updated values they wish to use in their analysis, then the user should modify the values in this file. This file is described in Section 5.

The file "ystrmtbflow_outyr.csv" is an output file generated by the program StrmtbFlow. The user must run the program StrmtbFlow, which is part of the FE/NETL CO_2 Prophet Model, before running the FE/NETL Onshore CO_2 EOR Cost Model. This file serves as an input file to the FE/NETL Onshore CO_2 EOR Cost Model. The contents of this file are described in Section 4.4.1. The user needs to copy the file "ystrmtbflow_outyr.csv" and place it in the directory where the FE/NETL Onshore CO_2 EOR Cost Model executable file resides.

The FE/NETL Onshore CO_2 EOR Cost Model generates one output file, "CO2EORCM_det_out.csv". This file is described in Section 6.

4 MATHEMATICAL BASIS

This section provides the mathematical foundation for the FE/NETL Onshore CO_2 EOR Cost Model. Because this model is a Fortran computer program, the variables and equations presented in this section are the same, whenever possible, as those in the computer code. Should a user wish to modify the code for their own purposes, this should help them understand the Fortran code.

4.1 NOTES ON FORTRAN

The discussion of the mathematical basis for the FE/NETL Onshore CO_2 EOR Cost Model often uses blocks of Fortran code or pseudo code to illustrate how a particular item is calculated in the model. For individuals who are not familiar with Fortran, a very brief introduction to some of the key features of Fortran is provided.

4.1.1 Source Code, Object Files, and Executable File

Fortran source code is stored in one or more text files. These files must be edited with a text editor, such as Notepad, which comes with Windows, or Notepad++, which is an open-source text editor that is more powerful than Notepad. These source code files must be compiled by a Fortran compiler to object files, which are machine code files. The various object files are then combined with operating system files in a link step by the Fortran compiler to create an executable file. The executable file is used to run the program. The open-source gfortran compiler was used to compile the source files that accompany this user's manual.

Fortran, like all modern programming languages, consists of a main program and subprograms. The main program is the code that is run when the user runs the executable file. The main program may be completely self-contained, but it most cases, the main program will run one or more subprograms. In Fortran, the subprograms are either functions or subroutines. A function is a subprogram that is designed to generate a single output. A function almost always has one or more inputs but should generate a single output. A subroutine is a subprogram that can generate multiple outputs. Like a function, a subroutine almost always has one or more inputs.

4.1.2 Variables

The FE/NETL Onshore CO_2 EOR Cost Model uses three types of variables provided in Fortran: integers, real numbers, and character variables. Character variables store text. Most of the variables in the FE/NETL Onshore CO_2 EOR Cost Model are integer or real number variables. In Fortran, variables can contain a single value or they can be arrays that store multiple values. In the FE/NETL Onshore CO_2 EOR Cost Model, there are many array variables and most of these arrays are one dimensional, but a few are two-dimensional variables. For example, the variable *x(i)* is a one-dimensional array with the index *i* indicating the location in the array. All arrays in the FE/NETL Onshore CO_2 EOR Cost Model start with an index of 1. The variable *y(i, j)* is a two-dimensional array, similar to a matrix in linear algebra. The index *i* can be thought of as pointing to row *i* and the index *j* as pointing to column *j*.

4.1.3 Assignment Statement and Arithmetic Operations

In Fortran, the equal sign (=) is used to assign whatever is on the right-hand side of the equal sign to the variable on the left-hand side. For example, the statement

```
x = 2
```

assigns the value 2 to the variable x. In any subsequent code that uses this variable, the value for x is 2. For example, in the following statement

```
y = x * 2
```

the variable y is assigned the value of 4 (i.e., 2 * 2).

In Fortran, the symbols +, -, * and / are used for addition, subtraction, multiplication, and division. In the following statement

```
y = x**2
```

the symbol ** indicates the variable x is raised to the second power (i.e., x is squared) and the result is assigned to the variable y.

In Fortran, all variables are really locations in the computer's memory that store data. As such, the same variable can be used on both sides of the equal sign in an assignment statement. For example, in the following code

```
x = 0
x = x + 1
```

The variable x is assigned the value 0 in the first statement. In the next statement, 1 is added to the value in x (i.e., 0) and the result (i.e., 1) is assigned to the variable x. Thus, the value for x is now 1.

4.1.4 If Then Else Blocks

Fortran has a key feature that allows different blocks of code to be executed based on the value of certain conditions. A generic "If Then Else" block looks like this:

```
If( condition1 ) Then
        Executable code 1
Else If( condition2 ) Then
        Executable code 2
Else If( condition3 ) Then
        Executable code 3
……
Else
        Executable code n
End If
```

When this code block is executed, the first conditional statement (condition1) is tested and if it is true, then the code labeled "Executable code 1" is executed and the rest of the If Then Else

block is ignored. If condition1 is false, then the next conditional statement (condition2) is tested and if this is true then the code labeled "Executable code 2" is executed and the rest of the If Then Else block is ignored. This process continues until a conditional statement tests true. If all of the If and Else If conditional statements are false then the code after the Else statement (the code labeled "Executable code n") is executed.

The following code example is used to illustrate how an If Then Else block works.

```
If( x > 1 ) Then
        y = 1
Else If( x <=  1 and x >  0 ) Then
        y = 0
Else
        y = -1
End If
```

In this code, the value assigned to y depends on the value of x when this If Then Else block of code is executed. The first condition is tested and if x is greater than 1, then y is assigned the value of 1. If x is not greater than 1, then the next condition is tested and if x is greater than 0 and less than or equal to 1, then y is assigned the value of 0. If the first two conditions are false (i.e., if x is less than or equal to 0), then y is assigned the value of -1.

In evaluations of conditional statements (such as x > 1), the following logical operators can be used to test if the conditional statements are true or false.

x == y translates to: x equals y

x /= y translates to: x is not equal to y

x < y translates to: x is less than y

x <= y translates to: x is less than or equal to y

x > y translates to: x is greater than y

x >= y translates to: x is greater than or equal to y

Other logical operators include "and" and "or". These operators connect two conditional statements.

4.1.5 Do Loops

A programmer will often need to have the same process executed over and over. Fortran provides "Do Loops" to allow a block of code to be executed repeatedly. The most common form of a Do Loop is as follows.

```
Do i = ibeg, iend
        Statement1
        Statement2
        …
        Statementn
```

```
End Do
```

In this type of a Do Loop, the variable *i* is an integer and is called the index. When the Do Loop is first executed, the index *i* is set to the value stored in the variable *ibeg* and then the statements within the Do Loop (Statement1 to Statementn) are executed. The program then returns to the top of the Do Loop and increments the index *i* by 1 and the statements within the Do Loop are executed again. This process is repeated until the index *i* equals *iend*. After *i* has been set to *iend* and the statements in the Do Loop have been executed, the program exits the Do Loop.

To illustrate how this type of a Do Loop functions, the following example Do Loop is used to calculate the square root of the sum of the squared values within the array variable *x(i)*.

```
sumsq = 0.0
Do i = 1, xend
        sumsq = sumsq + x(i)**2
End Do
srss = Sqrt(sumsq)
```

In this code, the variable *sumsq* is used to store the cumulative sum of the squared values within the array variable *x(i)*. The length of this array is stored in the variable *xend*. After this Do Loop has been executed, the variable *sumsq* stores the sum of each value in the array variable *x(i)* squared. The last statement takes the square root of *sumsq* (the function *Sqrt()* is a built-in Fortran function that performs this operation) and stores the result in the variable *srss* (an acronym for square root of the sum of the squares).

A less common form of a Do Loop is the following:

```
Do While(condition1)
        Statement1
        Statement2
        …
        Statement n
        If(condition2) Then
                ! code that makes "condition1"  False
        End If
End Do
```

In this type of a Do Loop, the statements within the Do Loop are executed as long as the conditional statement "condition1" is True. At some point within the Do Loop, a second condition is tested ("condition2") and based on this test, condition1 is made to be False. At this point, when the program returns to the top of the Do Loop, condition1 is False and the program exits the Do Loop. If condition1 is never made false, the program will execute the Do Loop forever (the proverbial infinite Do Loop). Thus, it is important to make sure that there is a way for the program to exit the Do Loop.

In any Do Loop, a condition can be tested within the Do Loop and based on the results of the test the Do Loop can be exited with the Exit statement.

4.1.6 Comments

In Fortran, any text entered after an exclamation point (!) is ignored by the Fortran compiler. Such text is referred to as comments. Comments are added to code to help explain what the code is attempting to do. The exclamation point forces the Fortran compiler to ignore text after the exclamation point on the line where the exclamation point is place. If the exclamation point is placed at the beginning of a line, then the entire line of text is ignored by the Fortran compiler. If the exclamation point is placed on a line after Fortran code, then the Fortran compiler processes the Fortran code and ignores all the text that occurs after the exclamation point.

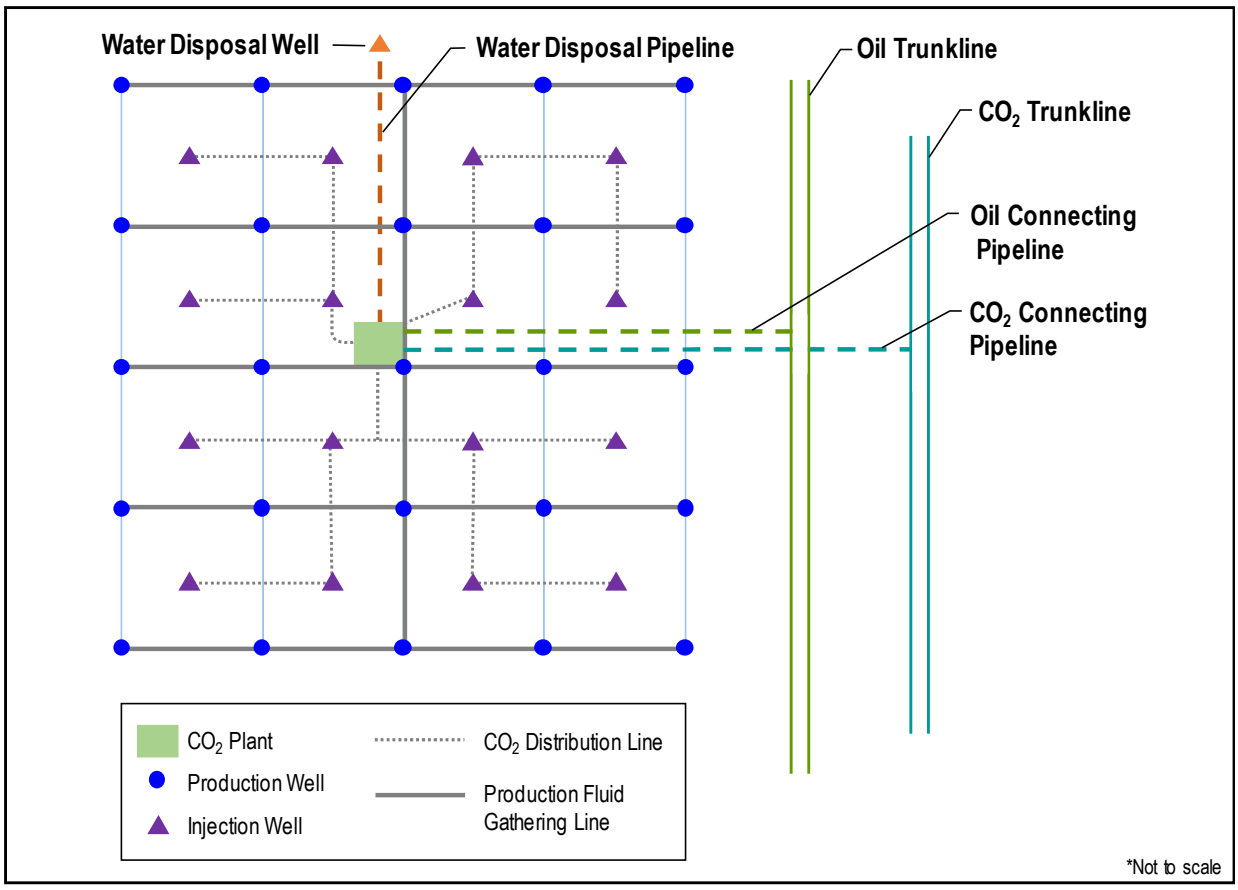
```
! This line is a comment line.
x = 1  ! assignment statement. This is a comment at the end of a line of code.
```

4.2 CONCEPTUALIZATION OF ONSHORE CO_2 EOR OPERATIONS

Exhibit 4-1 is a conceptual diagram of a CO_2 EOR operation showing injection wells, production wells, pipelines, and the CO_2 processing plant. This specific example shows 16 40-acre patterns arranged within 1 section of an oil field (640 acres total). The diagram shows an inverted 5-spot pattern, with 16 CO_2 and water injection wells central to each pattern and 25 production wells at the corner of each pattern.

Exhibit 4-1 shows the CO_2 processing plant in a central location in the oil field, CO_2 and water distribution lines connecting the injection wells, and fluid gathering lines extending from the production wells. There are also two connecting pipelines, one to an external CO_2 delivery trunk line, which brings CO_2 to the field, and an external oil trunk line, which takes produced oil to a refinery. Exhibit 4-1 also shows water disposal pipelines and water injection wells for projects that require water disposal.

Exhibit 4-1. CO_2 EOR field infrastructure conceptual diagram



Several cost elements are included in the FE/NETL Onshore CO_2 EOR Cost Model. At a minimum, the FE/NETL Onshore CO_2 EOR Cost Model provides the incremental capital costs incurred when implementing CO_2 EOR at an existing oil field operation, which are referred to as "brownfield" operations in this document. The FE/NETL Onshore CO_2 EOR Cost Model also provides the capital costs incurred when implementing CO_2 EOR at a new site without any existing oil extraction facilities. Such sites are referred to as "greenfield" operations in this document. Greenfield sites will incur nearly all the capital costs associated with brownfield CO_2 EOR sites, (except well recompletion) and will incur additional costs associated with site development. In the list of cost elements in Exhibit 4-2, the costs are identified as brownfield or greenfield. Most of the cost elements have both capital costs incurred when the element is installed and O&M costs that are incurred during the years of operation. The details of capital and O&M costs are discussed in later portions of this section.

Exhibit 4-2. Cost elements in the FE/NETL Onshore CO_2 EOR Cost Model

Cost Element	Description of Cost Element	Greenfield Cost	Brownfield Cost
Lease Acquisition Cost	Costs for acquiring the oil field lease, mineral rights, and any necessary permits	X	
Seismic Characterization	Costs for performing 3-D seismic characterization of the oil reservoir and processing the results	X	
Site Buildings and Access Roads	Costs for constructing access roads and new administrative buildings	X	
Test Characterization Wells	Costs for drilling and completing wells to obtain geologic data to characterize the oil reservoir	X	
Drilling and Completing New Wells	Cost for drilling and completing new injection, production, or water disposal wells	X	X
Recompleting a Well	Costs for recompleting existing injection or production wells		X
Plugging a Well	Costs for plugging old wells that are no longer useful		X
Surface Equipment for New Wells	Costs for surface equipment needed for new injection and production wells	X	X
CO_2 Recycling Plant	Costs for constructing a new CO_2 recycling plant	X	X
CO_2 Connecting Pipeline	Costs for constructing a new pipeline that brings CO_2 to the facility from a CO_2 trunk pipeline	X	X
Oil Connecting Pipeline	Costs for constructing a pipeline that delivers oil to an oil trunk pipeline to send the oil to a refinery for processing	X	
CO_2 Distribution system	Costs for constructing a pipeline network that carries CO_2 from the CO_2 processing plant to injection wells	X	X
Water Distribution System	Costs for constructing a pipeline network that carries water from the CO_2 processing plant to injection wells	X	
Produced Fluid Gathering System	Costs for constructing a pipeline network that carries produced fluids from the water from the CO_2 processing plant to injection wells	X	
Water Disposal Wells	Costs of drilling and completing injection wells to dispose of excess produced water	X	
Water Transport Pipelines	Costs of installing new pipelines to carry excess produced water to water disposal wells	X	
Monitoring Technologies for MRV Plans	Costs for installing and operating monitoring technologies needed to implement a Monitoring, Reporting and Verification (MRV) Plan for demonstrating storage of CO_2	X	X

4.3 TIME FRAMES AND ARRAY INDICES

The FE/NETL Onshore CO_2 EOR Cost Model has different time frames for different components of the model. The model implements patterns over time across an oil field and uses this information to estimate the number of injection wells, production wells, water disposal wells, and other types of equipment that are needed in a given year. The model uses input and output fluid flows for a pattern calculated by StrmtbFlow along with the schedule of patterns implemented over time to develop input and output fluid flows over time for the entire oil field.

Clearly, time is an important aspect of the model and there are many variables that store data at different times. In the Fortran code, these variables are arrays and each element of the array represents a result at a point in time. More specifically, the model operates on an annual (as opposed to monthly or quarterly) basis, so all the data in an array is an annual value of some kind.

4.3.1 Pattern-Level Time Frame

The results generated by StrmtbFlow are annual values for a pattern, but the first value produced by StrmtbFlow is the value at the beginning of CO_2 injection. For example, the variable *patcvco2in(jt)* stores the cumulative volume of CO_2 injected into the pattern in MMscf, the variable *pattim(jt)* stores the time in years and the variable *jt* is an index variable. When *jt* = 1, this is the beginning of CO_2 injection, so the variable *pattim(1)* equals 0 and the variable *patcvco2in(1)* also equals 0 since no CO_2 has been injected. The results at the end of the first year are associated with the index *jt* = 2. The variable *pattim(2)* equals 1 since this is the end of the first year and the variable *patcvco2in(2)* stores whatever value was generated by StrmtbFlow for the cumulative volume of CO_2 injected. Similarly, for values of *jt* greater than 2, the variable *pattim(jt)* = *jt* – 1 and the variable *patcvco2in(jt)* stores the cumulative volume of CO_2 injected that was calculated by StrmtbFlow for the end of year *jt* – 1.

While StrmtbFlow produces results over time with the first result corresponding to the start of CO_2 injection (i.e., year 0), the FE/NETL Onshore CO_2 EOR Cost Model calculates fluid flows for several variables that are annual values. For example, the variable *patyrvco2in(jy)* stores the volume of CO_2 injected into the pattern in a year with the index *jy* indicating the year of CO_2 injection. That is, for the first year of CO_2 injection, the index *jy* equals 1.

For variables storing results generated by StrmtbFlow (such as cumulative volumes of fluid injected or produced), the number of data points in each array is given by the variable *npattim*. The number of years the pattern operates is given by the variable *npatyr*, where

$$npatyr = npattim - 1$$

Thus, the index *jt* extends from 1 to *npattim* and the index *jy* ranges from 1 to *npatyr*.

4.3.2 CO_2 EOR Oil Field Operation Time Frame

The FE/NETL Onshore CO_2 EOR Cost Model implements patterns over time and calculates the fluid flows for the oil field over time. The variable *FldDevyr* is the number of years it takes to implement all the patterns. The FE/NETL Onshore CO_2 EOR Cost Model calculates field level

values and stores them in arrays where the first element in the array is the value for that variable at the start of CO_2 injection. For example, the variable *fldcvco2in(kt)* stores the cumulative volume of CO_2 injected into the oil field in MMscf, the variable *fldtim(kt)* stores the time in years and the variable *kt* is an index variable. When *kt* = 1, this is the beginning of CO_2 injection, so the variable *fldtim(1)* equals 0 and the variable *fldcvco2in(1)* also equals 0 since no CO_2 has been injected. The results at the end of the first year are associated with the index *kt* = 2. The variable *fldtim(2)* equals 1 since this is the end of the first year and the variable *fldcvco2in(2)* stores whatever value was generated by the FE/NETL Onshore CO_2 EOR Cost Model for the cumulative volume of CO_2 injected. Similarly, for values of *kt* greater than 2, the variable *fldtim(kt)* = *kt* – 1 and the variable *fldcvco2in(kt)* stores the cumulative volume of CO_2 injected that was calculated by the FE/NETL Onshore CO_2 EOR Cost Model for the end of year *kt* – 1.

For arrays storing results where the first element in the array is associated with the start of CO_2 injection into the oil field, the number of data points in each array is given by the variable *nfldtim*, where this variable is calculated as follows:

$$nfldtim = npattim + FldDevyr - 1$$

The index *kt* extends from 1 to *nfldtim*.

The FE/NETL Onshore CO_2 EOR Cost Model calculates fluid flows for several variables that are annual values. For example, the variable *fldyrvco2in(ky)* stores the volume of CO_2 injected into the oil field in a year with the index *ky* indicating the year of CO_2 injection. That is, for the first year of CO_2 injection, the index *ky* equals 1. The number of years the oil field operates is given by the variable *nfldyr*, where

$$nfldyr = npatyr + FldDevyr - 1$$

The index *ky* extends from 1 to *nfldyr*.

4.3.3 CO_2 EOR Project Time Frame

Key calculations in the FE/NETL Onshore CO_2 EOR Cost Model are cash flows. Cash flows for different variables are stored in arrays where each element in the array is a cash value in a specific year. For example, the variable *fldoilgrevcn(iy)* stores the gross revenue earned in year *iy* in constant dollars from producing and selling oil at the relevant price for oil. For this variable, the value stored in the arrays when *iy* equals 1 is the revenue generated in the first year of the oil field project. Before an oil field can begin actual CO_2 EOR operations (i.e., injecting CO_2), the oil field operator must install equipment, such as drilling new wells, recompleting other wells and installing a CO_2 processing plant. In the FE/NETL Onshore CO_2 EOR Cost Model, it is assumed that all equipment is installed in the year before it begins operation. For example, any new wells needed for a pattern must be drilled and completed the year before the pattern begins operation. Thus, the number of years the oil field project lasts, given by the variable *nfldeconyr*, is one year longer than the years that the oil field undergoes CO_2 EOR operations (given by the variable *nfldyr*).

$$nfldeconyr = nfldyr + 1$$

The cash flow index *iy* extends from 1 to *nfldeconyr*.

4.3.4 Summary of Time Frames for Different Array Types

To summarize, the FE/NETL Onshore CO_2 EOR Cost Model uses arrays to store data over time, where the time increment between data point is always a year.

- For a pattern, there are two types of arrays that store data over time relating to fluid flows and reservoir conditions (e.g., average oil saturation).
    - The first type of array stores data with the first element in each array corresponding to the start of CO_2 injection into the pattern. These arrays store *npattim* values and the array *pattim(jt)* stores the time associated with each element in the arrays. The value for *pattim(1)* is 0 since this is the beginning of CO_2 injection.
    - The second type of array stores data on an annual basis. These arrays store *npatyr* values and the first element of these arrays stores a value corresponding to the first year (such as the volume of CO_2 injected into the pattern in the first year).
    - The time frame associated with these arrays is referred to as the **pattern time frame** in this document, where the start time is the beginning of CO_2 injection into a pattern.
- For the entire oil field, there are two types of arrays that store data over time relating to fluid flows and reservoir conditions (e.g., average oil saturation).
    - The first type of array stores data with the first element in each array corresponding to the start of CO_2 injection into the oil field. These arrays store *nfldtim* values and the array *fldtim(kt)* stores the time associated with each element in the arrays. The value for *fldtim(1)* is 0 since this is the beginning of CO_2 injection into the oil field.
    - The second type of array stores data on an annual basis. These arrays store *nfldyr* values and the first element of these arrays stores a value corresponding to the first year of CO_2 injection (such as the volume of CO_2 injected into the oil field in the first year).
    - The time frame associated with these arrays is referred to as the **CO_2 EOR operation time frame** in this document, where the start time is the beginning of CO_2 injection into the oil field.
- For cash flows, there is only one type of array that stores monetary data over time.
    - These arrays store data on an annual basis. These arrays store *nfldeconyr* values and the first element of these arrays stores a value corresponding to the first year of CO_2 EOR project (such as the cost of drilling and completing new wells in the first year of the CO_2 EOR project). The FE/NETL Onshore CO_2 EOR Cost Model assumes that all equipment is installed a year before the equipment

begins operation. Thus, the first year associated with these arrays is the year before CO_2 EOR operations begin.

- The time frame associated with these arrays is referred to as the **CO_2 EOR project time frame** in this document, where the start time is the beginning of the CO_2 EOR project.

4.4 Implementing Patterns at an Oil Field and Scaling Pattern Level Fluid Flows to Field Level Fluid Flows

The FE/NETL Onshore CO_2 EOR Cost Model is not a reservoir simulation program. Instead the FE/NETL Onshore CO_2 EOR Cost Model uses results from the stream tube reservoir simulation program StrmtbFlow. StrmtbFlow creates an output file, "ystrmtbflow_outyr.csv", that provides input and output fluid flows for a pattern. This output file also provides the oil field area, pattern area, thickness of the reservoir, depth to the reservoir and fluid properties for water/brine, CO_2, oil, and hydrocarbon gas produced along with the oil. This information along with other user inputs is used to implement patterns over time at the oil field and translate annual pattern-level fluid flows to annual field level fluid flows. This section discusses how pattern level fluid flows generated by StrmtbFlow are scaled up to field level fluid flows by the FE/NETL Onshore CO_2 EOR Cost Model.

4.4.1 Data Extracted from the "ystrmtbflow_outyr.csv" File

The FE/NETL Onshore CO_2 EOR Cost Model extracts several sets of information from the "ystrmtbflow_outyr.csv" file.

The first set is basic descriptive information about the oil field:

DBName – in many cases, the StrmtbFlow and FE/NETL Onshore CO_2 EOR Cost Model are executed on oil fields that are part of a data set of oil fields. *DBName* is a character variable storing the name of the file storing the data set.

DBNum – *DBNum* is the number of the oil field in an oil field data set (i.e., in the file associated with the variable *DBName*).

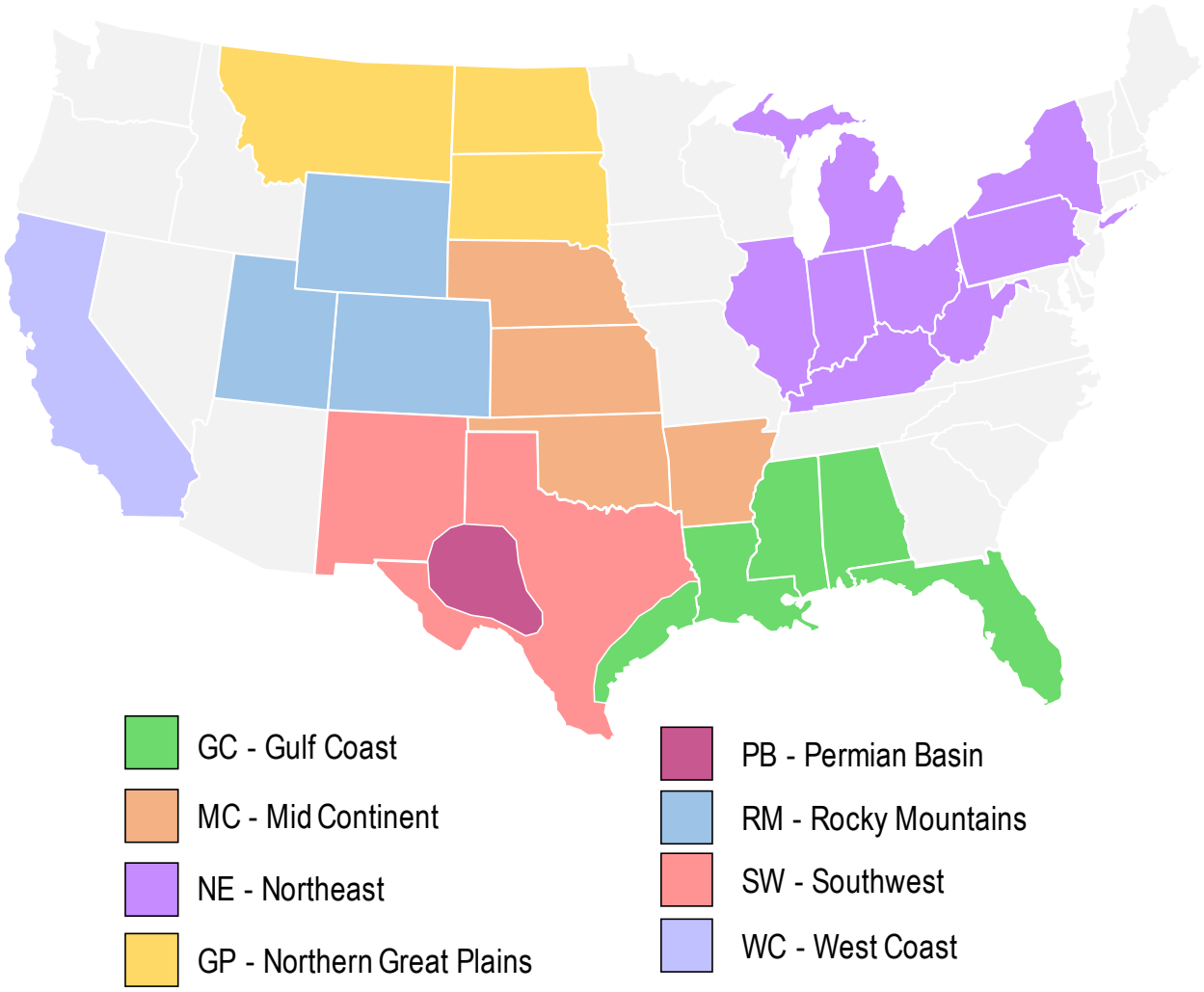
FldName – a character variable storing the name of the oil field.

ResName – a character variable storing the name of the reservoir at this oil field where CO_2 EOR is being implemented.

State_id – a character variable storing the two-letter abbreviation of the state where this oil field is located.

Rgn_id – a character variable storing the two-letter abbreviation for the economic region where the oil field is located. The FE/NETL Onshore CO_2 EOR Cost Model currently provides default cost information in the "Cost_in.dat" file for eight regions in the United States. These regions are illustrated in Exhibit 4-3 with their two-letter abbreviations.

Exhibit 4-3. Eight regions in the U.S. where costs are provided in the FE/NETL Onshore CO_2 EOR Cost Model



FldDescr1, FldDescr2, FldDescr3 – character variables storing information about the oil field that the user considers important.

FldVal1, FldVal2, FldVal3 – integer variables storing numerical information about the oil field that the user considers important.

latit1, longit1 – latitude and longitude for the oil field (degrees). A value greater than 999.0 or less than -999.0 indicates that the latitude or longitude are not known.

The second set is basic geologic information about the oil field. Some background on StrmtbFlow is useful for understanding the definitions of some of these variables. In StrmtbFlow, the reservoir is the subsurface volume where oil is located. The reservoir is conceptualized as consisting of several regions, with some regions encompassing other regions. The thickness of the reservoir is given by the variable *formthick* and is a region bounded on the top and bottom by relatively impermeable rock so that fluid flow is primarily within this region. Within the reservoir is a region containing oil that is readily extractable, the gross pay. The variable *grosspay* is the thickness of the gross pay. The gross pay is typically made up of layers

with relatively high oil saturations and permeabilities, the net pay, and it is the net pay that contains oil that is readily extractable. The remainder of the gross pay consists of layers where the oil saturations or permeability or both are too low for oil to be readily extracted. The thickness of the net pay is given by the variable *thick*. The variables *poros* (porosity), *permav* (average permeability), and *So_ooip* (oil saturation before any oil production began) are defined for the net pay.

In StrmtbFlow, the stream tubes model the net pay. The CO_2 EOR operator injects CO_2 into the reservoir and intends for the CO_2 to flow through the net pay, but much anecdotal evidence from CO_2 EOR operators indicates that CO_2, which is very mobile, can flow outside the net pay. Thus, StrmtbFlow allows some of the injected CO_2 to be displaced outside the net pay (i.e., the stream tubes) into the rest of the reservoir over the duration of CO_2 EOR operations. This is called the "other" portion of the reservoir in StrmtbFlow and this portion of the reservoir is the thickness of the reservoir minus the thickness of the net pay (i.e., *formthick* – *thick*). In StrmtbFlow, all CO_2 that remains in the reservoir at the end of CO_2 EOR operations will be present as pure phase CO_2 in the stream tubes, CO_2 dissolved in water in the stream tubes or CO_2 in the "other" portion of the reservoir. In StrmtbFlow, the "other" portion of the reservoir is conceptualized as a single volume and this volume is treated as a sink for CO_2 (i.e., CO_2 can enter the "other" portion of the reservoir, but CO_2 cannot leave the "other" portion of the reservoir). For simplicity, the CO_2 in the "other" portion of the reservoir is assumed to be present as pure phase CO_2. The documentation for StrmtbFlow should be referenced for more information on the definition of the variables *thick*, *grosspay,* and *formthick*. In general:

thick < *grosspay* < *formthick*

The second set of information is the following variables:

fldarea – area of the oil field (acres)

patarea – area of the pattern (acres)

depth – depth to the reservoir (ft)

thick – net pay of the reservoir (ft)

grosspay – gross pay of the reservoir (ft)

formthick – thickness of the reservoir (ft)

poros – porosity of the net pay. For simplicity, it is assumed that the porosity of the "other" portion of the reservoir is the same as the porosity of the net pay

permav – average permeability of the net pay in the horizontal or lateral direction (mD)

DPcoef – Dykstra-Parsons coefficient, which is a measure of the reservoir heterogeneity with respect to horizontal permeability. This heterogeneity is expressed in StrmtbFlow with layers having different horizontal or lateral permeabilities. Typically, this value is constrained to be between 0.5 and 0.93 for effective use with StrmtbFlow. A value of 0 indicates the reservoir is perfectly homogenous so that all layers in the model have the same horizontal or lateral permeability. In this instance, fluid will flow through all layers with the same propensity. A value close to 1

suggests the reservoir is extremely heterogeneous with a layer that is highly permeable and the remaining layers almost impermeable. In this instance, virtually all the fluid will flow through the layer with high permeability and essentially no fluid will flow through the other layers

The third set provides oil, water, and CO_2 saturations in the reservoir:

Soinit – average residual oil saturation of the reservoir at the start of the CO_2 flood

Swinit – average residual water saturation of the reservoir at the start of the CO_2 flood

Swinit = 1 – Soinit

Sginit – average residual gas saturation of the reservoir at the start of the CO_2 flood. Typically, this value is zero

So_ooip – saturation of oil in the net pay before any oil production began. This variable is used to calculate the original oil in place (OOIP)

The fourth set provides temperature and pressure data:

tsurff, tsurfc, tsurfK – temperature at the surface or at standard conditions (deg F, deg C, and K, respectively)

psurfpsia, psurfatm, psurfmpa – pressure at the surface or at standard conditions (psia, atm, MPa, respectively)

tres, tresc, tresk – temperature in the reservoir (deg F, deg C, and K, respectively)

prespsia, presatm, presmpa – pressure in the reservoir (psia, atm, MPa, respectively)

The fifth set provides fluid properties. For oil, the properties are

denosurfL, denosurfbbl – density of oil at surface or standard temperature and pressure (kg/L and kg/STB, respectively)

denoresL, denoresft3 – density of oil at reservoir temperature and pressure (kg/L and kg/ft^3, respectively)

boin – initial formation volume factor for oil, i.e., formation volume factor before any oil production began (res-bbl/STB)

boc, bocft3stb – current formation volume factor for oil (res-bbl/STB, res-ft^3/STB, respectively)

visop – viscosity of oil at reservoir temperature and pressure (cp)

api – API gravity of the oil (degrees API)

For hydrocarbon gas, the properties are

denhcgsurfL, denhcgsurfft3 – density of hydrocarbon gas at surface or standard temperature and pressure (kg/L and kg/scf, respectively)

For brine, the properties are

denwsurfL, denwsurfft3 – density of brine at surface or standard temperature and pressure (kg/L and kg/ft^3, respectively)

denwresL, denwresft3 – density of brine at reservoir temperature and pressure (kg/L and kg/ft^3, respectively)

bwft3stb – formation volume factor for brine (res-ft^3/STB)

viswp – viscosity of brine at reservoir temperature and pressure (cp)

For pure water, the properties are

denwpresL, denwpresft3 – density of pure water at reservoir temperature and pressure (kg/L and kg/ft^3, respectively)

For CO_2, the properties are

dengsurfL, dengsurfft3 – density of CO_2 at surface or standard temperature and pressure (kg/L and kg/scf, respectively)

dengresL, dengresft3 – density of CO_2 at reservoir temperature and pressure (kg/L and kg/ft^3, respectively)

bgft3mmcf – formation volume factor for CO_2 (res-ft^3/MMscf)

visgp – viscosity of CO_2 at reservoir temperature and pressure (cp)

For CO_2 dissolved in brine at reservoir conditions, the properties are

Cwgsolm, Cwgsol, Cwgsolft3 – solubility limit of CO_2 dissolved in brine at reservoir temperature and pressure (mol/kg-wat, kg/L-solution, kg-ft^3-solution, respectively)

For CO_2 dissolved in brine at surface conditions, the properties are

Cwgsolsurfm, Cwgsolsurf, Cwgsolsurfft3 – solubility limit of CO_2 dissolved in brine at surface or standard temperature and pressure (mol/kg-wat, kg/L-solution, kg-ft^3-solution, respectively)

The sixth set provides basic reservoir characteristics:

patpv – pattern pore volume (1000 ft^3)

pathcpv – pattern hydrocarbon pore volume (1000 ft^3)

patOOIP – pattern original oil in place (MSTB)

The seventh set provides fluid flow and reservoir data over time for the pattern (the first element in each array stores the value of the variable at the start of CO_2 injection into the reservoir (i.e., time 0)):

pattim(jt) – time value in period *jt* (yr)

When *jt* = 1, the time is 0 since it is the start of CO_2 injection

For *jt* > 1, the time is year *jt* - 1

patcvwatin(jt) – cumulative volume of water injected at the end of the time associated with *pattim(jt)* (MSTB)

patcvco2in(jt) – cumulative volume of CO_2 injected at the end of the time associated with *pattim(jt)* (MMscf)

patcvoilpr(jt) – cumulative volume of oil produced at the end of the time associated with *pattim(jt)* (MSTB)

patcvhcgpr (jt) – cumulative volume of hydrocarbon gas produced at the end of the time associated with *pattim(jt)* (MMscf)

patcvwatpr(jt) – cumulative volume of water produced at the end of the time associated with *pattim(jt)* (MSTB)

patcvco2pr(jt) – cumulative volume of CO_2 produced at the end of the time associated with *pattim(jt)* (MMscf)

patsoil(jt) – average saturation of oil in stream tubes in pattern at the time associated with *pattim(jt)*

patswat(jt) – average saturation of water in stream tubes in pattern at the time associated with *pattim(jt)*

patsco2(jt) – average saturation of CO_2 in stream tubes in pattern at the time associated with *pattim(jt)*

patcwco2(jt) – average concentration of CO_2 dissolved in water in stream tubes in pattern at the time associated with *pattim(jt)*

patVstbco2pp(jt) – volume of pure phase CO_2 in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

patVstbco2dw(jt) – volume of CO_2 dissolved in water in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

patVstbco2tot(jt) – volume of both pure phase CO_2 and CO_2 dissolved in water in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

patVothco2pp(jt) – volume of pure phase CO_2 in "other" portion of the reservoir in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

The eighth set provides additional variables related to the pattern that are not time dependent:

patVothavpv – volume in the "other" portion of the reservoir that is available for storing CO_2. Pattern value in reservoir volume (MM res-ft^3)

So_co2injbeg – average oil saturation in pattern at start of CO_2 injections

4.4.2 Implementing Patterns over Time

The number of patterns implemented at the oil field depends on the area of the oil field and the area of the pattern. The FE/NETL Onshore CO_2 EOR Cost Model allows the user to specify a

fraction of the oil field area that will be developed for CO_2 EOR. The area of the oil field that is developed for CO_2 EOR is given by the following equation.

$$flddevarea1 = fldarea \cdot frflddev$$

Where:

flddevarea1 – area of oil field that will potentially be developed for CO_2 EOR (acres)

fldarea – area of oil field (acres)

frflddev – fraction of the oil field developed for CO_2 EOR

The variable *frflddev* is a user input. The variable *fldarea* is a property of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

The total number of patterns is calculated with the following equation:

$$Npattot = floor(flddevarea1/patarea)$$

Where:

Npattot – total number of patterns implemented at the oil field

flddevarea1 – area of oil field that will potentially be developed for CO_2 EOR (acres)

patarea – area of the pattern (acres)

In this equation, the function *floor* indicates the quantity in the parentheses is rounded down to the nearest integer. The variable *patarea* is a property of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

The area of the oil field developed for CO_2 EOR is given by the following expression:

$$flddevarea = Npattot \cdot patarea$$

Where:

flddevarea – area of oil field that is developed for CO_2 EOR (acres)

Npattot – total number of patterns implemented at the oil field

patarea – area of the pattern (acres)

The number of patterns developed each year, on average, is calculated as follows:

$$Npateayrav1 = ceiling(Npattot/FldDevyr1)$$

Where:

Npateayrav1 – number of patterns potentially developed each year on average

Npattot – total number of patterns implemented at the oil field

FldDevyr1 – number of years potentially needed to implement all the patterns; this is a user input

In this equation, the function *ceiling* indicates the quantity in the parentheses is rounded up to the nearest integer. The variable *FldDevyr1* is a user input.

The number of patterns implemented each year is constrained by a minimum value (*Npattyr_min*) and a maximum value (*Npattyr_max*), with both values input by the user. The algorithm used to specify the average number of patterns implemented each year (*Npateayrav*) is:

```
If( Npateayrav1 < Npattyr_min )  Then:
        Npateayrav = Npattyr_min
If( Npateayrav1 > Npattyr_max )  Then:
        Npateayrav = Npattyr_max
Else:
        Npateayrav = Npateayrav1
End If
```

Where:

Npateayrav1 – number of patterns potentially developed each year on average

Npateayrav – number of patterns developed each year on average

Npattyr_min – minimum number of patterns implemented each year

Npattyr_max – maximum number of patterns implemented each year

The number of years needed to implement all the patterns is updated:

$$FldDevyr = ceiling(Npattot/Npateayrav)$$

Where:

FldDevyr – number of years needed to implement all the patterns

Npattot – total number of patterns implemented at the oil field

Npateayrav – number of patterns developed each year on average

An array storing the number of patterns implemented each year (*Npateayr(iy)*) is created with the following algorithm:

```
Npattot1 = 0
Do iy = 1, FldDevyr
        Npateayr(iy) = Npateayrav
        Npattot1 = Npattot1 + Npateayr(iy)
End Do
!
! Check for too many patterns being implemented
! If so, reduce number of patterns implemented in last year of development
If( Npattot < Npattot1 ) Then
        iy = FldDevyr
        Npateayr(iy) = Npateayr(iy) – (Npattot1 – Npattot)
End If
```

Where:

Npattot1 – variable used to total number of patterns implemented at end of each CO_2 EOR project year *iy*

FldDevyr – number of years needed to implement all the patterns

Npateayr(iy) – array variable used to store the number of patterns implemented in CO_2 EOR project year *iy*

Npateayrav – number of patterns developed each year on average

Npattot – total number of patterns implemented at the oil field

In this algorithm, each element of the array *Npateayr(iy)* is initially set equal to *Npateayrav*. The total number of patterns that would be implemented over *FldDevyr* years is stored in the variable *Npattot1*. If *Npattot1* exceeds *Npattot*, then the number of patterns implemented in the last year (*iy* = *FldDevyr*) is reduced so that the total number of patterns implemented is *Npattot*.

The FE/NETL Onshore CO_2 EOR Cost Model needs to calculate the number of patterns active or operational in any given year. By operational, this means that CO_2 is being injected into the pattern and CO_2 EOR operations are occurring at the pattern. Any equipment that needs to be installed to make the pattern operational (such as drilling and completing new wells), is done the year before operations begin. The variable *Npatacyr(ky)* stores the number of active patterns in year *ky*, where *ky* = 1 is the first year of CO_2 EOR operations and *ky* = *nfldyr* is the last year of CO_2 EOR operations.

The algorithm for calculating the elements of *Npatacyr(ky)* comprises two parts. First, the number of patterns that come on line each year is calculated:

```
Do ky = 1, nfldyr
    If( ky = 1 ) Then
        iy = ky
        Npatacyr(ky) = Npateayr(iy)
    Else If( (ky > 1) and (ky <= FldDevyr) ) Then
        iy = ky
        Npatacyr(ky) = Npatacyr(ky-1) + Npateayr(iy)
    Else
        Npatacyr(ky) = Npattot
    End If
End Do
```

Where:

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

Npatacyr(ky) – array variable storing the number of patterns active in CO_2 EOR operational year *ky*

Npateayr(iy) – array variable used to store the number of patterns implemented in CO_2 EOR project year *iy*

FldDevyr – number of years needed to implement all the patterns

Npattot – total number of patterns implemented at the oil field

Next, the patterns that have been shut down are subtracted from each relevant year. In the algorithm below, the variable *sum1* is the total number of patterns that have been shut down in year *ky*:

```
Do ky = 1, nfldyr
    If( ky > npatyr ) Then
        kend = ky - npatyr
        sum1 = 0.0
        Do k = 1, kend
            sum1 = sum1 + Npateayr(k)
        End Do
        Npatacyr(ky) = Npatacyr(ky) - sum1
    End If
End Do
```

Where:

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

npatyr – number of years CO_2 EOR is operated at a pattern

kend – in CO_2 EOR operational year *ky*, this variable stores the number of years over which patterns have been shut down

sum1 – variable used to sum the number of patterns that have been shut down in CO_2 EOR operational year *ky*

Npatacyr(ky) – array variable storing the number of patterns active in CO_2 EOR operational year *ky*

Npateayr(k) – array variable used to store the number of patterns implemented in CO_2 EOR project year *k*

4.4.3 Implementing Wells over Time

This section presents the equations used to calculate the number of injection and production wells that are implemented each year and the number of injection and production wells that are active in each year. This section also presents the equations used to calculate the number of test characterization wells installed each year and the number of wells that are plugged and abandoned each year.

4.4.3.1 Injection and Production Wells

The number of injection and production wells that are needed depends on the number of patterns. Currently, the StrmtbFlow program can simulate the injection and production of fluid (e.g., oil) for two patterns, a five-spot pattern and a line-drive pattern.

A five-spot pattern has four production wells arranged at the corners of a square and one injection well in the center of the square. If a second five-spot pattern is added next to the original pattern, then a new injection well is needed but only two new production wells are necessary because two of the original production wells will serve as production wells for both the original pattern and the second pattern. If a third pattern is added above the original pattern, then once again, one additional injection well is needed and two additional production wells are needed since two of the original production wells will serve as production wells for both the original pattern and the third pattern. If a fourth pattern is added above the second pattern, then there are four patterns arranged as a square (two rows and two columns of patterns). An additional injection well is needed for the fourth pattern but only one additional production well is needed since production wells associated with the other three patterns will also serve as production wells for the fourth pattern.

Based on this heuristic analysis, the number of injection wells needed equals the number of patterns implemented. The number of production wells needed depends on the number of patterns implemented and the arrangement of the patterns. An actual oil field will have patterns implemented based on the areal distribution of oil in the oil field and other considerations such as subsurface features that might affect fluid flow. The FE/NETL Onshore CO_2 EOR Cost Model has a function named *ProdWellNum(NumPatt)* that calculates the number of production wells needed for one or more five-spot patterns. The variable *NumPatt* is the number of five-spot patterns. The algorithm in the function *ProdWellNum()* uses the assumption that the patterns are implemented in as close to a square configuration as possible. Thus, 4 patterns would be arranged as a square with 2 rows and 2 columns; 9 patterns would be arranged as a square with 3 rows and 3 columns; 16 patterns would be arranged as a square with 4 rows and 4 columns; and so on. This is the most efficient arrangement of patterns resulting in the fewest number of production wells that would be needed for the given number of patterns. An actual oil field that implements five-spot patterns cannot have fewer production wells than the number calculated with the function *ProdWellNum()*.

A line-drive pattern has two production wells and an injection well located halfway between the two production wells. A single line-drive pattern has one injection well and two production wells. If a second line-drive pattern is added next to the original pattern, then a new injection well is needed along with two new production wells. If a third pattern is added above the original pattern, then once again, one additional injection well is needed but only one production well is needed since one of the original production wells will serve as production wells for both the original pattern and the third pattern. If a fourth pattern is added above the second pattern, then there are four patterns arranged as a square (two rows and two columns of patterns). An additional injection well is needed for the fourth pattern but only one additional production well is needed since a production well associated with the second pattern will also serve as a production well for the fourth pattern.

The number of injection wells needed for a collection of line-drive patterns equals the number of patterns. The number of production wells depends on the number of patterns and how these patterns are arranged. The number of production wells needed will be slightly less than the number of production wells needed for the same number of five-spot patterns arranged in a similar manner. However, in the current version of the FE/NETL Onshore CO_2 EOR Cost Model, the function *ProdWellNum(NumPatt)* only calculates the number of production wells needed for a collection of five-spot patterns and this number is also used for a collection of line-drive patterns.

The number of injection wells implemented in each CO_2 EOR project year *iy*, the number of these wells that are new and the number of these wells that are recompleted is specified by the following algorithm. The number of wells that are new versus recompleted depends on the variable *frnewwells*, which is a user input. However, if this is a greenfield site (i.e., the variable *grnfldcon* equals 1), then all wells are new, so the variable *frnewwells* is set to 1. In the algorithm given below, the function *Nint(x)* is a built-in Fortran function that rounds the value stored in the variable *x* to the nearest integer value (so *x* can be rounded up or down):

```
!
If( grnfldcon == 1 ) Then
        frnewwells = 1.0
End If
!
Ninjwtot = 0
Ninjwnewtot = 0
Ninjwrectot = 0
Do iy = 1, FldDevyr
        !
        ! Determine number of injection wells implemented in each
        ! year as the oil field is developed, including the number
        ! of new and recompleted injection wells
        Ninjwtotyr(iy) = NPateayr(iy)
        Ninjwtot = Ninjwtot + Ninjwtotyr(iy)
        Ninjwnewtotp = Ninjwnewtot
        Ninjwnewtot = Nint( frnewwells * Ninjwtot )
        Ninjwnewyr(iy) = Ninjwnewtot - Ninjwnewtotp
        Ninjwrecyr(iy) = Ninjwtotyr(iy) - Ninjwnewyr(iy)
        Ninjwrectot = Ninjwrectot + Ninjwrecyr(iy)
End Do
```

Where:

grnfldcon – if *grnfldcon* equals 1, then this is a greenfield site; otherwise this is a brownfield site

frnewwells – fraction of injection and production wells that are new; remaining wells are assumed to be recompleted

Ninjwtot – variable storing cumulative number of injection wells implemented through CO_2 EOR project year *iy*

Ninjwnewtot – variable storing cumulative number of new injection wells that have been implemented through CO_2 EOR project year *iy*

Ninjwrectot – variable storing cumulative number of recompleted injection wells that have been implemented through CO_2 EOR project year *iy*

FldDevyr – number of years needed to implement all the patterns

Ninjwtotyr(iy) – array variable used to store the number of injection wells implemented in CO_2 EOR project year *iy*

Npateayr(iy) – array variable used to store the number of patterns implemented in CO_2 EOR project year *iy*

Ninjwnewtotp – variable storing cumulative number of new injection wells that have been implemented through CO_2 EOR project year *iy-1*

Ninjwnewyr(iy) – array variable storing the number of new injection wells implemented in CO_2 EOR project year *iy*

Ninjwrecyr(iy) – array variable storing the number of recompleted injection wells implemented in CO_2 EOR project year *iy*

The variables *grnfldcon* and *frnewwells* are user inputs.

The number of production wells implemented in each CO_2 EOR project year *iy*, the number of these wells that are new and the number of these wells that are recompleted is specified by the following algorithm. The value for the variable *frnewwells* is the same for injection and production wells and depends on the value of the variable *grnfldcon* as discussed previously. The function *ProdWellNum(NumPatt)* was discussed previously.

```
Npat1 = 0
Nprwtot = 0
Nprwnewtot = 0
Nprwrectot = 0
Do iy = 1, FldDevyr
    !
    ! Determine number of production wells implemented in each
    ! year as the oil field is developed, including the number
    ! of new and recompleted production wells
    Npat1 = Npat1 + NPateayr(iy)
    Nprwtotp = Nprwtot
    Nprwtot = ProdWellNum( Npat1 )
    Nprwtotyr(iy) = Nprwtot - Nprwtotp
    Nprwnewtotp = Nprwnewtot
    Nprwnewtot = Nint( frnewwells * Nprwtot )
    Nprwnewyr(iy) = Nprwnewtot - Nprwnewtotp
```

Nprwrecyr(iy) = Nprwtotyr(iy) - Nprwnewyr(iy)
Nprwrectot = Nprwrectot + Nprwrecyr(iy)
End Do

Where:

Npat1 – variable storing cumulative number of patterns implemented through CO_2 EOR project year *iy*

Nprwtot – variable storing cumulative number of production wells implemented through CO_2 EOR project year *iy*

Nprwnewtot – variable storing cumulative number of new production wells that have been implemented through CO_2 EOR project year *iy*

Nprwrectot – variable storing cumulative number of recompleted production wells that have been implemented through CO_2 EOR project year *iy*

FldDevyr – number of years needed to implement all the patterns

Npateayr(iy) – array variable used to store the number of patterns implemented in CO_2 EOR project year *iy*

Nprwtotp – variable storing cumulative number of production wells that have been implemented through CO_2 EOR project year *iy-1*

Nprwtotyr(iy) – array variable used to store the number of production wells implemented in CO_2 EOR project year *iy*

Nprwnewtotp – variable storing cumulative number of new production wells that have been implemented through CO_2 EOR project year *iy-1*

frnewwells – fraction of injection and production wells that are new; remaining wells are assumed to be recompleted

Nprwnewyr(iy) – array variable storing the number of new production wells implemented in CO_2 EOR project year *iy*

Nprwrecyr(iy) – array variable storing the number of recompleted production wells implemented in CO_2 EOR project year *iy*

The variable *frnewwells* is a user input.

After the number of new and recompleted injection and production wells have been calculated, the total number of these wells is calculated with the following equations:

Nwtot = Ninjwtot + Nprwtot
Nwnewtot = Ninjwnewtot + Nprwnewtot
Nwrectot = Ninjwrectot + Nprwrectot

Where:

Nwtot – variable storing the total number of injection and production wells implemented

Ninjwtot – variable storing the total number of injection wells implemented

Nprwtot – variable storing the total number of production wells implemented

Nwnewtot – variable storing the total number of new injection and production wells implemented

Ninjwnewtot – variable storing the total number of new injection wells implemented

Nprwnewtot – variable storing the total number of new production wells implemented

Nwrectot – variable storing total number of recompleted injection and production wells implemented

Ninjwrectot – variable storing total number of recompleted injection wells implemented

Nprwrectot – variable storing total number of recompleted production wells implemented

The FE/NETL Onshore CO_2 EOR Cost Model needs to calculate the number of injection and production wells that are active or operational in any given year. By operational, this means that CO_2 EOR operations are occurring at the pattern. Any equipment that needs to be installed to make the pattern operational (such as drilling and completing new wells) is installed the year before operations begin. The variable *Ninjwacyr(ky)* stores the number of active injection wells in year *ky* and the variable *Nprwacyr(ky)* stores the number of active production wells in year *ky*, where *ky* = 1 is the first year of CO_2 EOR operations and *ky* = *nfldyr* is the last year of CO_2 EOR operations. This is the CO_2 EOR operations time frame.

The number of injection wells and production wells that are active in any given year depends on the number of patterns that are active. The algorithm for calculating the number of active injection and production wells in any given year is as follows:

Do ky = 1, nfldyr
Ninjwacyr(ky) = Npatacyr(ky)
Nprwacyr(ky) = ProdWellNum(Npatacyr(ky))
Nwacyr(ky) = Ninjwacyr(ky) + Nprwacyr(ky)
End Do

Where:

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

Ninjwacyr(ky) – array variable storing the number of injection wells active in CO_2 EOR operational year *ky*

Npatacyr(ky) – array variable storing the number of patterns active in CO_2 EOR operational year *ky*

Nprwacyr(ky) – array variable storing the number of production wells active in CO_2 EOR operational year *ky*

Nwacyr(ky) – array variable storing the number of injection and production wells active in CO_2 EOR operational year *ky*

4.4.3.2 Test Characterization Wells

If the CO_2 EOR project is a greenfield site, then the user can specify one or more wells that can be drilled to gather detailed characterization data about the target oil reservoir. These can be test characterization wells that are drilled and completed, are used for testing and then are plugged (specified by the variable *Ntestwellchar*). Alternatively, they can be test characterization wells that are drilled and completed, are used for testing and are retained for use as injection or production wells (*Ntestwellused*). All test characterization wells are drilled in the first year of the CO_2 EOR project. The test characterization wells that are subsequently used as injection or production wells are drilled as early as possible in the project depending on the number of these wells specified by the user and the number of new injection and production wells implemented each year. The algorithm used to specify the number of these test characterization wells implemented each year is as follows:

```
!
If( grnfldcon == 1 ) Then
        Ntestwusedneeded1 = Ntestwellused
Else
        Ntestwusedneeded1 = 0
End If
!
Do iy = 1, FldDevyr
        If( grnfldcon == 1 ) Then
                !
                ! This is a greenfield site.
                ! First determine test wells installed for characterization
                ! only. These wells are installed in the first year and
                ! then are plugged after installation.
                If( iy == 1 ) Then
                        Ntestwcharyr(iy) = Ntestwellchar
                End If
                !
                ! Next determine test characterization wells installed that are
                ! eventually completed as new injection or production wells.
                If( Ntestwusedneeded1 >  0 ) Then
                        Nnewwellyr1 = Ninjwnewyr(iy) + Nprwnewyr(iy)
                        If( Ntestwusedneeded1 <= Nnewwellyr1 ) Then
                                !
                                ! The number of test wells still needed is less than
                                ! or equal to the number of new wells drilled this
```

```
                    ! year. All remaining test wells are installed this year.
                    Ntestwusedyr(iy) = Ntestwusedneeded1
                    Ntestwusedneeded1 = 0
                Else
                    !
                    ! The number of test wells still needed is greater than
                    ! the number of new wells drilled this year.
                    ! The number of test wells drilled this year equals
                    ! the number of new injection and production wells
                    ! drilled this year. The number of test wells
                    ! needed in future years is reduced by the number
                    ! of test wells drilled in this year.
                    Ntestwusedyr(iy) = Nnewwellyr1
                    Ntestwusedneeded1 = Ntestwusedneeded1 - Nnewwellyr1
                End If
            End If
        End If
End Do
```

Where:

grnfldcon – if *grnfldcon* equals 1, then this is a greenfield site; otherwise this is a brownfield site

Ntestwusedneeded1 – number of test characterization wells that are to be used as injection or production wells that still need to be installed at the beginning of CO_2 EOR project year *iy*

Ntestwellused – number of test characterization wells to be installed that are subsequently used as injection or production wells. This is a user input

FldDevyr – number of years needed to implement all the patterns

Ntestwcharyr(iy) – array variable used to store the number of test characterization wells that are subsequently plugged that are implemented in CO_2 EOR project year *iy*

Ntestwellchar – number of test characterization wells to be installed that are subsequently plugged. This is a user input

Nnewwellyr1 – total number of new injection wells and new production wells implemented in CO_2 EOR project year *iy*

Ninjwnewyr(iy) – array variable storing the number of new injection wells implemented in CO_2 EOR project year *iy*

Nprwnewyr(iy) – array variable storing the number of new production wells implemented in CO_2 EOR project year *iy*

Ntestwusedyr(iy) – array variable used to store the number of test characterization wells that are subsequently used as injection or production wells that are implemented in CO_2 EOR project year *iy*

The variables *grnfldcon*, *Ntestwellchar*, and *Ntestwellused* are user inputs.

4.4.3.3 Plugged and Abandoned Wells

The last thing calculated is the number of wells that are plugged and abandoned. For a brownfield site, the FE/NETL Onshore CO_2 EOR Cost Model assumes that all new wells replace existing wells and that these existing wells are plugged. For a brownfield sites, no test characterization wells are installed so no test characterization wells are plugged. For a greenfield site, there are no existing wells to be plugged. The only wells that need to be plugged are test characterization wells that are installed and subsequently plugged. The algorithm used to calculate the number of plugged wells is as follows.

```
Nplugwtot = 0
Do iy = 1, FldDevyr
        !
        ! Determine number of wells plugged in each year as the oil
        ! field is developed.
        ! For a greenfield site, no wells are plugged except test wells
        !   only installed for characterization
        ! For a brownfield site, the number of plugged wells are assumed to
        !   equal the number of new wells plus all test wells only
        !   installed for characterization
        If( grnfldcon == 1 ) Then
                If( iy == 1 ) Then
                        Nplugwyr(iy) = Ntestwcharyr(iy)
                End If
        Else
                Nplugwyr(iy) = Ninjwnewyr(iy) + Nprwnewyr(iy)
        End If
        Nplugwtot = Nplugwtot + Nplugwyr(iy)
End Do
```

Where:

Nplugwtot – total number of wells that are plugged

FldDevyr – number of years needed to implement all the patterns

grnfldcon – if *grnfldcon* equals 1, then this is a greenfield site; otherwise this is a brownfield site

Nplugwyr(iy) – number of wells plugged in CO_2 EOR project year *iy*

Ninjwnewyr(iy) – array variable storing the number of new injection wells implemented in CO_2 EOR project year *iy*

Nprwnewyr(iy) – array variable storing the number of new production wells implemented in CO_2 EOR project year *iy*

The variable *grnfldcon* is a user input.

4.4.3.4 Water Disposal Wells

If the CO_2 EOR project is a greenfield site, then the FE/NETL Onshore CO_2 EOR Cost Model calculates the number of water disposal wells that need to be installed each year. The model uses the rule of thumb that 1 water disposal well is installed for every 10 patterns implemented. The algorithm used is as follows:

```
temp = 0.0
Nwatinjtot = 0
Do iy = 1, FldDevyr
    !
    ! Determine total number of patterns implemented in year iy
    ! and store in temp
    temp = temp + Npateayr(iy)
    !
    ! Round temp up to nearest integer and store in Nwatinjwtot1
    Nwatinjwtot1 = ceiling( 0.1 * temp )
    !
    ! The number of water disposal wells installed in year iy is the total number
    ! of water disposal wells by the end of year iy (Nwatinjwtot1) minus
    ! the total number of water disposal wells installed by the end of
    ! the previous year iy – 1 (Nwatinjwtot)
    Nwatinjwnewyr(iy) = Nwatinjwtot1 – Nwatinjwtot
    !
    ! Update Nwatinjwtot to store total number of water disposal wells installed
    ! by end of year iy
    Nwatinjwtot = Nwatinjwtot1
End Do
```

Where:

temp – stores the cumulative number of water disposal wells installed in CO_2 EOR project year *iy*

Nwatinjtot – stores the cumulative number of water disposal wells installed at the end of CO_2 EOR project year *iy - 1* (at beginning of algorithm) or CO_2 EOR project year *iy* (at end of algorithm)

FldDevyr – number of years needed to implement all the patterns

Npateayr(iy) – array variable used to store the number of patterns implemented in CO_2 EOR project year *iy*

Nwatinjtot1 – stores the cumulative number of water disposal wells installed at the end of CO_2 EOR project year *iy*

Nwatinjwnewyr(iy) – array variable used to store the number of water disposal wells installed in CO_2 EOR project year *iy*

In this algorithm, the function *ceiling()* indicates the quantity in the parentheses is rounded up to the nearest integer.

The FE/NETL Onshore CO_2 EOR Cost Model also determines the number of water disposal wells that are active or operational in each CO_2 EOR operations year *ky*:

```
Do ky = 1, nfldyr
    If( ky == 1) Then
        iy = ky
        Nwatinjwtotyr(ky) = Nwatinjwnewyr(iy)
    Else If( ky <= FldDevyr ) Then
        iy = ky
        Nwatinjwtotyr(ky) = Nwatinjwtotyr(ky - 1) + Nwatinjwnewyr(iy)
    Else
        Nwatinjwtotyr(ky) = Nwatinjwtot
    End If
End Do
```

Where:

ky – index for CO_2 EOR operations year *ky*

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

iy – index for CO_2 EOR project year *iy*

Nwatinjwtotyr(ky) – array variable used to store the number of water disposal wells active in CO_2 EOR operations year *ky*

Nwatinjwnewyr(iy) – array variable used to store the number of new water disposal wells installed in CO_2 EOR project year *iy*

FldDevyr – number of years needed to implement all the patterns

Nwatinjtot – stores the total number of water disposal wells installed during CO_2 EOR project

4.4.4 Scaling Up Pattern-level Fluid Flows to Field-level Fluid Flows

This section describes how pattern-level fluid flows generated by StrmtbFlow are scaled up to field-level fluid flows. This is done is two steps. First, the fluid flows output by StrmtbFlow, which are cumulative fluid flows over time for a pattern, are converted to annual flows for the pattern. Second, these pattern-level fluid flows are scaled up to field-level fluid flows over time

by accounting for the operation of patterns over time. Both cumulative and annual field-level fluid flows are calculated.

4.4.4.1 Annual Pattern-level Fluid Flows

The program StrmtbFlow outputs cumulative pattern-level fluid flows for a number of fluids: water and CO_2 injected and oil, hydrocarbon gas, water, and CO_2 produced. The FE/NETL Onshore CO_2 EOR Cost Model needs annual pattern-level flows of these fluids to calculate field-level annual fluid flows. The following algorithm is used to convert cumulative flows to annual flows. The variables storing cumulative flows (such as *patcvwatin*, the cumulative flow of water injected into the pattern) are array variables where the first element in the array (e.g., *patcvwatin(1)*) is the cumulative fluid flow when the pattern begins CO_2 EOR operations. The cumulative fluid flow is always 0 at the beginning since the CO_2 EOR operations have just begun. The subsequent elements in these arrays store cumulative fluid flows at the end of a year, so *patcvwatin(2)* is the cumulative flow of water injected into the patten at the end of the first year, *patcvwatin(3)* is the cumulative flow of water injected into the patten at the end of the second year and so on.

```
Do jy = 1, npatyr
    If( jy == 1 ) Then
        patyrvwatin(jy) = patcvwatin(jy+1)
        patyrvco2in(jy) = patcvco2in(jy+1)
        patyrvoilpr(jy) = patcvoilpr(jy+1)
        patyrvhcgpr(jy) = patcvhcgpr(jy+1)
        patyrvwatpr(jy) = patcvwatpr(jy+1)
        patyrvco2pr(jy) = patcvco2pr(jy+1)
    Else
        patyrvwatin(jy) = patcvwatin(jy+1) - patcvwatin(jy)
        patyrvco2in(jy) = patcvco2in(jy+1) - patcvco2in(jy)
        patyrvoilpr(jy) = patcvoilpr(jy+1) - patcvoilpr(jy)
        patyrvhcgpr(jy) = patcvhcgpr(jy+1) - patcvhcgpr(jy)
        patyrvwatpr(jy) = patcvwatpr(jy+1) - patcvwatpr(jy)
        patyrvco2pr(jy) = patcvco2pr(jy+1) - patcvco2pr(jy)
    End If
End Do
```

Where:

npatyr – number of years CO_2 EOR is operated at a pattern

patyrvwatin(jy) – volume of water injected during year *jy* (MSTB)

patcvwatin(jy + 1) – cumulative volume of water injected at the end of year *jy* (MSTB)

patyrvco2in(jy) – volume of CO_2 injected during year *jy* (MMscf)

patcvco2in(jy + 1) – cumulative volume of CO_2 injected at the end of year *jy* (MMscf)

patyrvoilpr(jy) – volume of oil produced during year *jy* (MSTB)

patcvoilpr(jy + 1) – cumulative volume of oil produced at the end of year *jy* (MSTB)

patyrvhcgpr(jy) – volume of hydrocarbon gas produced during year *jy* (MMscf)

patcvhcgpr(jy + 1) – cumulative volume of hydrocarbon gas produced at the end of year *jy* (MMscf)

patyrvwatpr(jy) – volume of water produced during year *jy* (MSTB)

patcvwatpr(jy + 1) – cumulative volume of water produced at the end of year *jy* (MSTB)

patyrvco2pr(jy) – volume of CO_2 produced during year *jy* (MMscf)
patcvco2pr(jy + 1) – cumulative volume of CO_2 produced at the end of year *jy* (MMscf)

4.4.4.2 Annual and Cumulative Field-level Fluid Flows

The FE/NETL Onshore CO_2 EOR Cost Model needs annual field-level fluid flows for the cash flow calculations. The annual pattern-level fluid flows are scaled to the field level in a two-step algorithm. First, the algorithm cycles through the years where patterns are implemented. Second, for each pattern that is implemented, the algorithm cycles through the years when this pattern operates and the fluid flows for each pattern operations year are added to the fluid flows for the entire field in the applicable CO_2 EOR operations year.

```
! Cycle through CO2 EOR operations years when patterns
! begin CO2 EOR operations
Do kybeg = 1, FldDevyr
    ! For each pattern, cycle through years that pattern operates CO2 EOR
    ! Add the fluid flows for each pattern to the fluid flows for the entire oil field
    ! The index ky refers to a field-level CO2 EOR operations year
    ! The index ky + 1 – kybeg refers to a pattern-level operations year
    kyend = kybeg + npatyr - 1
    Do ky = kybeg, kyend
        fldyrvwatin(ky) = fldyrvwatin(ky) + NPateayr(kybeg) * patyrvwatin( ky +1 -
            kybeg )
        fldyrvco2in(ky) = fldyrvco2in(ky) + NPateayr(kybeg) * patyrvco2in( ky + 1 -
            kybeg)
        fldyrvoilpr(ky) = fldyrvoilpr(ky) + NPateayr(kybeg) * patyrvoilpr( ky + 1 -
            kybeg )
        fldyrvhcgpr(ky) = fldyrvhcgpr(ky) + NPateayr(kybeg) * patyrvhcgpr( ky + 1
            - kybeg )
        fldyrvwatpr(ky) = fldyrvwatpr(ky) + NPateayr(kybeg) * patyrvwatpr( ky + 1
            - kybeg )
        fldyrvco2pr(ky) = fldyrvco2pr(ky) + NPateayr(kybeg) * patyrvco2pr( ky + 1
        – kybeg )
```

End Do

End Do

Where:

FldDevyr – number of years where patterns are implemented

kybeg – first CO_2 EOR operations year for a pattern (i.e., year when CO_2 injection begins at a pattern)

kyend – last CO_2 EOR operations year for a pattern (i.e., year when CO_2 injection ends at a pattern)

npatyr – number of years CO_2 EOR is operated at a pattern

ky – index associated with year when CO_2 operations occur (*ky* = 1 is first year when CO_2 EOR operations begin for the first patterns implemented)

fldyrvwatin(ky) – volume of water injected into oil field during CO_2 EOR operations year *ky* (MSTB)

NPateayr(kybeg) – number of patterns that begin CO_2 EOR operations in year *kybeg*

patyrvwatin(ky + 1 - kybeg) – volume of water injected during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MSTB)

fldyrvco2in(ky) – volume of CO_2 injected into oil field during CO_2 EOR operations year *ky* (MMscf)

patyrvco2in(ky + 1 - kybeg) – volume of CO_2 injected during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MMscf)

fldyrvoilpr(ky) – volume of oil produced from oil field during CO_2 EOR operations year *ky* (MSTB)

patyrvoilpr(ky + 1 - kybeg) – volume of oil produced during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MSTB)

fldyrvhcgpr(ky) – volume of hydrocarbon gas produced from oil field during CO_2 EOR operations year *ky* (MMscf)

patyrvhcgpr(ky + 1 - kybeg) – volume of hydrocarbon gas produced during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MMscf)

fldyrvwatpr(ky) – volume of water produced from oil field during CO_2 EOR operations year *ky* (MSTB)

patyrvwatpr(ky + 1 - kybeg) – volume of water produced during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MSTB)

fldyrvco2pr(ky) – volume of CO_2 produced from oil field during CO_2 EOR operations year *ky* (MMscf)

patyrvco2pr(ky + 1 - kybeg) – volume of CO_2 produced during CO_2 EOR operations year *ky* at a pattern that begins CO_2 EOR operations in year *kybeg* (MMscf)

The FE/NETL Onshore CO_2 EOR Cost Model also calculates cumulative field-level fluid flows. Cumulative fluid flows are generally not used in the cash flow calculations. However, cumulative fluid flows are very useful measures of the performance of the oil field with respect to CO_2 EOR. The summary fluid flows that are output by the FE/NETL Onshore CO_2 EOR Cost Model are typically cumulative fluid flows at the end of CO_2 EOR operations.

The arrays that store annual fluid flows store total fluid flows for a year. The arrays that store cumulative fluid flows store cumulative fluid flows associated with a time. The first element in each array stores the cumulative fluid flow at the beginning of CO_2 EOR operations, which is always a zero cumulative flow since this is the start of CO_2 EOR operations. Each successive element in the arrays stores the cumulative fluid flow at the end of a year. Thus, the second element in each array stores the cumulative fluid flow at the end of the first year, the third element in each array stores the cumulative fluid flow at the end of the second year, and so on. The algorithm used to calculate cumulative fluid flows at each time point accumulates annual fluid flows over time.

```
Do kt = 1, nfldtim
    ! Calculate time, in years, when cumulative fluid flow is determined
    fldtim(kt) = kt-1
    If( kt == 1 ) Then
        fldcvwatin(kt) = 0.0
        fldcvco2in(kt) = 0.0
        fldcvoilpr(kt) = 0.0
        fldcvhcgpr(kt) = 0.0
        fldcvwatpr(kt) = 0.0
        fldcvco2pr(kt) = 0.0
    Else
        fldcvwatin(kt) = fldcvwatin(kt - 1) + fldyrvwatin(kt - 1)
        fldcvco2in(kt) = fldcvco2in(kt - 1) + fldyrvco2in(kt - 1)
        fldcvoilpr(kt) = fldcvoilpr(kt - 1) + fldyrvoilpr(kt - 1)
        fldcvhcgpr(kt) = fldcvhcgpr(kt - 1) + fldyrvhcgpr(kt - 1)
        fldcvwatpr(kt) = fldcvwatpr(kt - 1) + fldyrvwatpr(kt - 1)
        fldcvco2pr(kt) = fldcvco2pr(kt - 1) + fldyrvco2pr(kt - 1)
    End If
End Do
```

Where:

nfldtim – number of time points in arrays storing cumulative fluid flows

kt – index for arrays storing cumulative fluid flows and time associated with each element in these arrays (*kt* = 1 is associated with the time when CO_2 EOR operations begins, which is time 0)

fldtim(kt) – time associated with the kt^{th} element in the arrays storing cumulative fluid flows (years)

fldcvwatin(kt) – cumulative volume of water injected into oil field during CO_2 EOR operations at time *fldtim(kt)* (MSTB)

fldyrvwatin(kt-1) – volume of water injected into oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MSTB)

fldcvco2in(kt) – cumulative volume of CO_2 injected into oil field during CO_2 EOR operations at time *fldtim(kt)* (MMscf)

fldyrvco2in(kt-1) – volume of CO_2 injected into oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MMscf)

fldcvoilpr(kt) – cumulative volume of oil produced from oil field during CO_2 EOR operations at time *fldtim(kt)* (MSTB)

fldyrvoilpr(kt-1) – volume of oil produced from oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MSTB)

fldcvhcgpr(kt) – cumulative volume of hydrocarbon gas produced from oil field during CO_2 EOR operations at time *fldtim(kt)* (MMscf)

fldyrvhcgpr(kt-1) – volume of hydrocarbon gas produced from oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MMscf)

fldcvwatpr(kt) – cumulative volume of water produced from oil field during CO_2 EOR operations at time *fldtim(kt)* (MSTB)

fldyrvwatpr(kt-1) – volume of water produced from oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MSTB)

fldcvco2pr(kt) –cumulative volume of CO_2 produced from oil field during CO_2 EOR operations at time *fldtim(kt)* (MMscf)

fldyrvco2pr(kt-1) – volume of CO_2 produced from oil field during CO_2 EOR operations year *kt – 1,* where *kt* is greater than 1 (MMscf)

4.4.5 Annual and Cumulative CO_2 Volumes and Masses

In addition to the field-level flows of injected CO_2 and produced CO_2, the FE/NETL Onshore CO_2 EOR Cost Model needs the volumes of CO_2 that are purchased, recycled and disposed each year. The model also needs the volume of CO_2 in the reservoir (i.e., the volume of CO_2 stored).

In each year, the volume of CO_2 purchased is the volume of CO_2 injected minus the volume of CO_2 recycled. The possible volume of CO_2 recycled is the volume of CO_2 produced minus any CO_2 that is lost to the atmosphere as incidental emissions during the processing of the produced fluids. Depending on the volume of CO_2 needed for injection, the volume available for

recycling can be greater than the volume of CO_2 injected. In that case, any CO_2 potentially available for recycling that is not needed for injection is disposed. The disposed CO_2 could be sent to another oil field for use in CO_2 EOR or, if no such oil field is available, the CO_2 is vented to the atmosphere. Thus, the term "disposed" does not necessarily mean the CO_2 is vented to the atmosphere.

The following algorithm calculates the volumes of CO_2 in the reservoir (i.e., stored), emitted to the atmosphere as incidental emissions during produced fluid processing, recycling, and disposal. The volumes are calculated on an annual basis. The arrays storing these annual volumes store the volumes for a CO_2 EOR operations year. The algorithm also calculates cumulative volumes. The arrays storing cumulative volumes are indexed so that each element is associated with a time stored in the array *fldtim(kt)* where *kt* is an index running from 1 to *nfldtim*. The array *fldtim(kt)* and variable *nfldtim* were defined previously. It should be noted that all the arrays storing cumulative volumes of CO_2 have the first element of the array set to zero (i.e., the cumulative volume of all fluid flows are always set to zero at the start of CO_2 EOR operations).

```
Do ky = 1, nfldyr
        !
        ! CO2 in reservoir
        fldcvco2res(ky + 1) = fldcvco2in(ky + 1) - fldcvco2pr(ky + 1)
        fldyrvco2res(ky) = fldcvco2res(ky + 1) - fldcvco2res(ky)
        !
        ! CO2 emitted
        fldyrvco2emit(ky) = fremissco2 * fldyrvco2pr(ky)
        fldcvco2emit(ky + 1) = fldcvco2emit(ky) + fldyrvco2emit(ky)
        !
        ! CO2avail is produced CO2 minus emitted CO2 that is
        ! potentially available for recycling
        CO2avail = fldyrvco2pr(ky) - fldyrvco2emit(ky)
        !
        ! Zero out CO2avail if it is a very small number
        If( CO2avail < 10^-10 ) CO2avail = 0.0
        !
        ! Calculate CO2 recycled, purchased and disposed
        If( fldyrvco2in(ky) > 10^-10 ) Then
                !
                ! CO2 is injected this year
                If( CO2avail >= fldyrvco2in(ky) ) Then
                        !
                        ! More CO2 is available for recycle than is needed for
                        !   injection. None is purchased.
                        ! The CO2 that cannot be injected must be disposed.
                        fldyrvco2recy(ky) = fldyrvco2in(ky)
                        fldyrvco2purch(ky) = 0.0
```

```
                fldyrvco2disp(ky) = CO2avail - fldyrvco2recy(ky)
            Else
                !
                ! Less CO2 is available for recycle than is needed for
                !   injection. All available CO2 is recycled.
                ! Some additional CO2 is purchased. None is disposed.
                fldyrvco2recy(ky) = CO2avail
                fldyrvco2purch(ky) = fldyrvco2in(ky) - CO2avail
                fldyrvco2disp(ky) = 0.0
            End If
        Else
            !
            ! No CO2 is injected this year, all available CO2 is disposed
            fldyrvco2recy(ky) = 0.0
            fldyrvco2purch(ky) = 0.0
            fldyrvco2disp(ky) = CO2avail
        End If
        !
        ! Calculate cumulative volumes of CO2 recycled, purchased and disposed
        fldcvco2recy(ky + 1) = fldcvco2recy(ky) + fldyrvco2recy(ky)
        fldcvco2purch(ky + 1) = fldcvco2purch(ky) + fldyrvco2purch(ky)
        fldcvco2disp(ky + 1) = fldcvco2disp(ky) + fldyrvco2disp(ky)
End Do
```

Where:

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

ky – index associated with year when CO_2 operations occur (*ky* = 1 is first year when CO_2 EOR operations begin for the first patterns implemented)

fldcvco2res(ky + 1) – volume of CO_2 in the reservoir at time *fldtim(ky+1)* (MMscf)

fldcvco2in(ky + 1) – cumulative volume of CO_2 injected into the oil field at time *fldtim(ky+1)* (MMscf)

fldcvco2pr(ky + 1) – cumulative volume of CO_2 produced from the oil field at time *fldtim(ky+1)* (MMscf)

fldyrvco2res(ky) – annual change in CO_2 volume in the reservoir during CO_2 EOR operations year *ky* (MMscf)

fldyrvco2emit(ky) – volume of CO_2 emitted to atmosphere as incidental emissions during fluid processing during CO_2 EOR operations year *ky* (MMscf)

fremissco2 – fraction of produced CO_2 that is assumed to be emitted to the atmosphere as incidental emissions. This is a user input

fldyrvco2pr(ky) – volume of CO_2 produced from oil field during CO_2 EOR operations year *ky* (MMscf)

fldcvco2emit(ky+1) – cumulative volume of CO_2 emitted to atmosphere as incidental emissions during fluid processing at time *fldtim(ky+1)* (MMscf)

CO2avail – volume of CO_2 potentially available for recycling during CO_2 EOR operations year *ky* (MMscf)

fldyrvco2in(ky) – volume of CO_2 injected into oil field during CO_2 EOR operations year *ky* (MMscf)

fldyrvco2recy(ky) – volume of CO_2 recycled during CO_2 EOR operations year *ky* (MMscf)

fldyrvco2purch(ky) – volume of CO_2 purchased during CO_2 EOR operations year *ky* (MMscf)

fldyrvco2disp(ky) – volume of CO_2 disposed during CO_2 EOR operations year *ky* (MMscf)

fldcvco2recy(ky + 1) – cumulative volume of CO_2 recycled at time *fldtim(ky+1)* (MMscf)

fldcvco2purch(ky + 1) – cumulative volume of CO_2 purchased at time *fldtim(ky+1)* (MMscf)

fldcvco2disp(ky + 1) – cumulative volume of CO_2 disposed at time *fldtim(ky+1)* (MMscf)

The volumes of CO_2 discussed above are all calculated as MMscf. These volumes can be converted to mass values by multiplying the volumes by the density of CO_2 at standard temperature and pressure. The variables storing annual masses of CO_2 are calculated with the following equations:

! Annual mass quantities of CO2
*fldyrmco2in = dengkt * fldyrvco2in*
*fldyrmco2pr = dengkt * fldyrvco2pr*
*fldyrmco2emit = dengkt * fldyrvco2emit*
*fldyrmco2res = dengkt * fldyrvco2res*
*fldyrmco2recy = dengkt * fldyrvco2recy*
*fldyrmco2purch = dengkt * fldyrvco2purch*
*fldyrmco2disp = dengkt * fldyrvco2disp*

Where:

fldyrmco2in(ky) – mass of CO_2 injected into the oil field during CO_2 EOR operations year *ky* (ktonne)

dengkt – density of CO_2 (ktonne/MMscf)

fldyrvco2in(ky) – volume of CO_2 injected into the oil field during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2pr(ky) – mass of CO_2 produced from the oil field during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2pr(ky) – volume of CO_2 produced from the oil field during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2emit(ky) – mass of CO_2 emitted to the atmosphere during processing of produced fluids during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2emit(ky) – volume of CO_2 emitted to the atmosphere during processing of produced fluids during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2res(ky) – annual change in CO_2 mass in the reservoir during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2res(ky) – annual change in CO_2 volume in the reservoir during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2recy(ky) – mass of CO_2 recycled during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2recy(ky) – volume of CO_2 recycled during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2purch(ky) – mass of CO_2 purchased during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2purch(ky) – volume of CO_2 purchased during CO_2 EOR operations year *ky* (MMscf)

fldyrmco2disp(ky) – mass of CO_2 disposed during CO_2 EOR operations year *ky* (ktonne)

fldyrvco2disp(ky) – volume of CO_2 disposed during CO_2 EOR operations year *ky* (MMscf)

The algorithm also calculates cumulative masses of CO_2. As with the arrays storing cumulative volumes of CO_2, the arrays storing cumulative masses are indexed so that each element is associated with a time stored in the array *fldtim(kt)* where *kt* is an index running from 1 to *nfldtim*. It should be noted that all the arrays storing cumulative masses of CO_2 have the first element of the array set to zero (i.e., the cumulative mass of all flows are always set to zero at the start of CO_2 EOR operations).

```
! Cumulative mass quantities of CO2
fldcmco2in = dengkt * fldcvco2in
fldcmco2pr = dengkt * fldcvco2pr
fldcmco2emit = dengkt * fldcvco2emit
fldcmco2res = dengkt * fldcvco2res
fldcmco2recy = dengkt * fldcvco2recy
fldcmco2purch = dengkt * fldcvco2purch
fldcmco2disp = dengkt * fldcvco2disp
```

Where:

fldcmco2in(kt) – cumulative mass of CO_2 injected into the oil field at time *fldtim(kt)* (ktonne)

dengkt – density of CO_2 (ktonne/MMscf)

fldcvco2in(kt) – cumulative volume of CO_2 injected into the oil field at time *fldtim(kt)* (MMscf)

fldcmco2pr(kt) – cumulative mass of CO_2 produced from the oil field at time *fldtim(kt)* (ktonne)

fldcvco2pr(kt) – cumulative volume of CO_2 produced from the oil field at time *fldtim(kt)* (MMscf)

fldcmco2emit(kt) – cumulative mass of CO_2 emitted to the atmosphere during processing of produced fluids at time *fldtim(kt)* (ktonne)

fldcvco2emit(kt) – cumulative volume of CO_2 emitted to the atmosphere during processing of produced fluids at time *fldtim(kt)* (MMscf)

fldcmco2res(kt) – mass of CO_2 in the reservoir at time *fldtim(kt)* (ktonne)

fldcvco2res(kt) – volume of CO_2 in the reservoir at time *fldtim(kt)* (MMscf)

fldcmco2recy(kt) – cumulative mass of CO_2 recycled at time *fldtim(kt)* (ktonne)

fldcvco2recy(kt) – cumulative volume of CO_2 recycled at time *fldtim(kt)* (MMscf)

fldcmco2purch(kt) – cumulative mass of CO_2 purchased at time *fldtim(kt)* (ktonne)

fldcvco2purch(kt) – cumulative volume of CO_2 purchased at time *fldtim(kt)* (MMscf)

fldcmco2disp(kt) – cumulative mass of CO_2 disposed at time *fldtim(kt)* (ktonne)

fldcvco2disp(kt) – volume of CO_2 disposed at time *fldtim(kt)* (MMscf)

4.4.6 Annual and Cumulative Water Volumes

In addition to the field-level flows of injected water and produced water, the FE/NETL Onshore CO_2 EOR Cost Model generates the volumes of water that are purchased, recycled, and disposed each year.

In each year, the volume of water purchased is the volume of water injected minus the volume of water recycled. The purchased water may come from water wells on site or the water may be imported from offsite. The possible volume of water recycled is the volume of water produced minus any water that is lost to the atmosphere as incidental emissions during the processing of the produced fluids. Depending on the volume of water needed for injection, the volume available for recycling can be greater than the volume of water injected. In that case, any water potentially available for recycling that is not needed for injection is disposed. The disposed water is assumed to be injected into the subsurface using water disposal wells.

The following algorithm calculates the volumes of water emitted to the atmosphere, recycled and disposed. The volumes are calculated on an annual basis. The arrays storing these annual volumes store the volumes for a CO_2 EOR operations year. The algorithm also calculates cumulative volumes. The arrays storing cumulative volumes are indexed so that each element is associated with a time stored in the array *fldtim(kt)* where *kt* is an index running from 1 to *nfldtim* as discussed previously. It should be noted that all the arrays storing cumulative volumes of water have the first element of the array set to zero (i.e., the cumulative volume of all fluid flows are always set to zero at the start of CO_2 EOR operations).

```
Do ky = 1, nfldyr
    !
    ! water emitted
    fldyrvwatemit(ky) = fremisswat * fldyrvwatpr(ky)
    fldcvwatemit(ky + 1) = fldcvwatemit(ky) + fldyrvwatemit(ky)
    !
    ! Calculate water potentially available for recycling
    watavail = fldyrvwatpr(ky) – fldyrvwatemit(ky)
    !
    ! if watavail is a very small number, set it equal to 0
    If( watavail < 10^-10 ) watavail = 0.0
    !
    ! water recycled, purchased and disposed
    If( fldyrvwatin(ky) > 10^-10 ) Then
        !
        ! water is injected this year
        If( watavail >= fldyrvwatin(ky) ) Then
            !
            ! More water is available for recycle than is needed for
            !   injection. None is purchased. Some must be disposed.
            fldyrvwatrecy(ky) = fldyrvwatin(ky)
            fldyrvwatpurch(ky) = 0.0
            fldyrvwatdisp(ky) = watavail - fldyrvwatrecy(ky)
        Else
            !
            ! Less water is available for recycle than is needed for
            ! injection. All available water is recycled.
            ! Some additional water is purchased. None is disposed.
            fldyrvwatrecy(ky) = watavail
            fldyrvwatpurch(ky) = fldyrvwatin(ky) - watavail
            fldyrvwatdisp(ky) = 0.0
        End If
    Else
        !
        ! No water is injected this year, all available water is disposed
```

```
        fldyrvwatrecy(ky) = 0.0
        fldyrvwatpurch(ky) = 0.0
        fldyrvwatdisp(ky) = watavail
    End If
    !
    ! Calculate cumulative volumes of water recycled, purchased and disposed
    fldcvwatrecy(ky + 1) = fldcvwatrecy(ky) + fldyrvwatrecy(ky)
    fldcvwatpurch(ky + 1) = fldcvwatpurch(ky) + fldyrvwatpurch(ky)
    fldcvwatdisp(ky + 1) = fldcvwatdisp(ky) + fldyrvwatdisp(ky)
End Do
```

Where:

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

ky – index associated with year when CO_2 operations occur (*ky = 1* is first year when CO_2 EOR operations begin for the first patterns implemented)

fldyrvwatemit(ky) – volume of water emitted to atmosphere as incidental emissions during fluid processing during CO_2 EOR operations year *ky* (MSTB)

fremisswat – fraction of produced water that is assumed to be emitted to the atmosphere as incidental emissions. This is a user input

fldyrvwatpr(ky) – volume of water produced from oil field during CO_2 EOR operations year *ky* (MSTB)

fldcvwatemit(ky+1) – cumulative volume of water emitted to atmosphere as incidental emissions during fluid processing at time *fldtim(ky+1)* (MSTB)

watavail – volume of water potentially available for recycling during CO_2 EOR operations year *ky* (MSTB)

fldyrvwatin(ky) – volume of water injected into oil field during CO_2 EOR operations year *ky* (MSTB)

fldyrvwatrecy(ky) – volume of water recycled during CO_2 EOR operations year *ky* (MSTB)

fldyrvwatpurch(ky) – volume of water purchased during CO_2 EOR operations year *ky* (MSTB)

fldyrvwatdisp(ky) – volume of water disposed during CO_2 EOR operations year *ky* (MSTB)

fldcvwatrecy(ky + 1) – cumulative volume of water recycled at time *fldtim(ky+1)* (MSTB)

fldcvwatpurch(ky + 1) – cumulative volume of water purchased at time *fldtim(ky+1)* (MSTB)

fldcvwatdisp(ky + 1) – cumulative volume of water disposed at time *fldtim(ky+1)* (MSTB)

4.4.7 Performance Measures for the CO_2 EOR Operation

The FE/NETL Onshore CO_2 EOR Cost Model calculates three measures of the field-level performance of the CO_2 EOR operation. One measure is the oil recovery factor, which is the fraction of the field-level OOIP that has been produced during CO_2 EOR operations. A second measure is the CO_2 utilization factor, which is the cumulative volume of purchased CO_2 divided by the cumulative volume of oil produced with that CO_2. Because recycled CO_2 supplements purchased CO_2, the purchased CO_2 is not the total volume of CO_2 injected. The CO_2 utilization factor is a measure of the effectiveness of the purchased CO_2 at producing oil. A third measure is the CO_2 retention factor, which is the fraction of the cumulative volume of CO_2 that has been injected that has been retained in the reservoir as pure phase CO_2 in the stream tubes, as CO_2 dissolved in water in the stream tubes or as pure phase CO_2 in the "other" portion of the reservoir. These three measures are calculated at the end of each year of CO_2 EOR operations using the following algorithm.

```
! field-level OOIP for developed portion of oil field
FldDevOOIP = Npattot * patOOIP
Do ky = 1, nfldyr
        !
        ! Oil recovery factor
        fldyroilrecf(ky) = fldcvoilpr(ky + 1) / FldDevOOIP
        !
        ! CO2 utilization factor
        If( fldcvoilpr(ky + 1) < 10^-10 ) Then
                fldyrco2utilf(ky) = 1000.0
        Else
                fldyrco2utilf(ky) = fldcvco2purch(ky + 1) / fldcvoilpr(ky + 1)
        End If
        If( fldyrco2utilf(ky) > 1000.0 ) fldyrco2utilf(ky) = 1000.0
        !
        ! CO2 retention factor
        If( fldcvco2in(ky + 1) < 10^-10 ) Then
                fldyrco2retf(ky) = 1.0
        Else
                fldyrco2retf(ky) = fldcvco2res(ky + 1) / fldcvco2in(ky + 1)
        End If
End Do
```

Where:

FldDevOOIP – field-level original oil in place for developed portion of oil field (MSTB)

Npattot – total number of patterns implemented at the oil field

patOOIP – pattern-level original oil in place (MSTB)

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

ky – index associated with year when CO_2 operations occur (*ky = 1* is first year when CO_2 EOR operations begin for the first patterns implemented)

fldyroilrecf(ky) – field-level oil recovery factor at end of CO_2 EOR operations year *ky*

fldcvoilpr(ky + 1) – cumulative volume of oil produced from oil field during CO_2 EOR operations at time *fldtim(ky + 1)* (MSTB)

fldyrco2utilf(ky) – field-level CO_2 utilization factor at end of CO_2 EOR operations year *ky* (Mscf/STB)

fldcvco2purch(ky + 1) – cumulative volume of CO_2 purchased for injection into the oil field at time *fldtim(ky + 1)* (MMscf)

fldyrco2retf(ky) – field-level CO_2 retention factor at end of CO_2 EOR operations year *ky*

fldcvco2res(ky + 1) – volume of CO_2 in the reservoir at time *fldtim(ky + 1)* (MMscf)

fldcvco2in(ky + 1) – cumulative volume of CO_2 injected into the oil field at time *fldtim(ky + 1)* (MMscf)

4.4.8 Pattern and Field Level Hydrocarbon Pore Volumes

The FE/NETL Onshore CO_2 EOR Cost Model calculates pattern-level and field-level hydrocarbon pore volumes (HCPVs) of fluid injected and produced. The pattern-level HCPVs of water injected, CO_2 injected, total fluid injected, oil produced, water produced, CO_2 produced, and total fluid produced are calculated. The algorithm uses the cumulative volumes of the fluids injected or produced, the formation volume factors relevant to a specific fluid and the hydrocarbon pore volume for the pattern:

Do jt = 1, npattim
*pathcpvwatin(jt) = patcvwatin(jt) * bwft3stb / pathcpv*
*pathcpvco2in(jt) = patcvco2in(jt) * bgft3mmcf / (pathcpv * 1000.0)*
pathcpvtotin(jt) = pathcpvwatin(jt) + pathcpvco2in(jt)
*pathcpvoilpr(jt) = patcvoilpr(jt) * boft3stb / pathcpv*
*pathcpvwatpr(jt) = patcvwatpr(jt) * bwft3stb / pathcpv*
*pathcpvco2pr(jt) = patcvco2pr(jt) * bgft3mmcf / (pathcpv * 1000.0 ft^3/Mscf)*
pathcpvtotpr(jt) = pathcpvoilpr(jt) + pathcpvwatpr(jt) + pathcpvco2pr(jt)
End Do

Where:

npattim – number of elements in arrays where cumulative volumes of fluid are stored. The times associated with each array element are stored in the array *pattim(jt)*

pathcpvwatin(jt) – cumulative HCPVs of water injected into the pattern at the end of the time associated with *pattim(jt)*

patcvwatin(jt) – cumulative volume of water injected into the pattern at the end of the time associated with *pattim(jt)* (MSTB)

bwft3stb – formation volume factor for water (res-ft^3/STB)

pathcpv - pattern hydrocarbon pore volume (1000 res-ft^3)

pathcpvco2in(jt) – cumulative HCPVs of CO_2 injected into the pattern at the end of the time associated with *pattim(jt)*

patcvco2in(jt) – cumulative volume of CO_2 injected into the pattern at the end of the time associated with *pattim(jt)* (MMscf)

bgft3mmcf – formation volume factor for CO_2 (res-ft^3/MMscf)

pathcpvtotin(jt) – cumulative HCPVs of water and CO_2 injected into the pattern at the end of the time associated with *pattim(jt)*

pathcpvoilpr(jt) – cumulative HCPVs of oil produced from the pattern at the end of the time associated with *pattim(jt)*. This includes oil and hydrocarbon gas produced

patcvoilpr(jt) – cumulative volume of oil produced from the pattern at the end of the time associated with *pattim(jt)* (MSTB)

boft3stb – formation volume factor for oil (res-ft^3/STB)

pathcpvwatpr(jt) – cumulative HCPVs of water produced from the pattern at the end of the time associated with *pattim(jt)*

patcvwatpr(jt) – cumulative volume of water produced from the pattern at the end of the time associated with *pattim(jt)* (MSTB)

pathcpvco2pr(jt) – cumulative HCPVs of CO_2 produced from the pattern at the end of the time associated with *pattim(jt)*

patcvco2pr(jt) – cumulative volume of CO_2 produced from the pattern at the end of the time associated with *pattim(jt)* (MMscf)

pathcpvtotpr(jt) – cumulative HCPVs of oil, water, and CO_2 produced from the pattern at the end of the time associated with *pattim(jt)*

In a similar manner, the FE/NETL Onshore CO_2 EOR Cost Model calculates the field-level HCPVs of water injected, CO_2 injected, total fluid injected, oil produced, water produced, CO_2 produced, and total fluid produced are calculated. The algorithm uses the cumulative volumes of the fluids injected or produced, the formation volume factors relevant to a specific fluid as defined above and the HCPV for the oil field:

*fldhcpv = Npattot * pathcpv*
Do jt = 1, nfldtim
*fldhcpvwatin(kt) = fldcvwatin(kt) * bwft3stb / fldhcpv*
*fldhcpvco2in(kt) = fldcvco2in(kt) * bgft3mmcf / (fldhcpv * 1000.0 ft^3/Mscf)*
fldhcpvtotin(kt) = fldhcpvwatin(kt) + fldhcpvco2in(kt)
*fldhcpvoilpr(kt) = fldcvoilpr(kt) * boft3stb / fldhcpv*
*fldhcpvwatpr(kt) = fldcvwatpr(kt) * bwft3stb / fldhcpv*
*fldhcpvco2pr(kt) = fldcvco2pr(kt) * bgft3mmcf / (fldhcpv * 1000.0 ft^3/Mscf)*
fldhcpvtotpr(kt) = fldhcpvoilpr(kt) + fldhcpvwatpr(kt) + fldhcpvco2pr(kt)
End Do

Where:

fldhcpv – oil field HCPV (1000 res-ft^3)

Npattot – number of patterns implemented at the oil field

nfldtim – number of elements in arrays where cumulative volumes of fluid are stored. The times associated with each array element are stored in the array *fldtim(kt)*

fldhcpvwatin(kt) – cumulative HCPVs of water injected into the oil field at the end of the time associated with *fldtim(kt)*

fldcvwatin(kt) – cumulative volume of water injected into the oil field at the end of the time associated with *fldtim(kt)* (MSTB)

fldhcpvco2in(kt) – cumulative HCPVs of CO_2 injected into the oil field at the end of the time associated with *fldtim(kt)*

fldcvco2in(kt) – cumulative volume of CO_2 injected into the oil field at the end of the time associated with *fldtim(kt)* (MMscf)

fldhcpvtotin(kt) – cumulative HCPVs of water and CO_2 injected into the oil field at the end of the time associated with *fldtim(kt)*

fldhcpvoilpr(kt) – cumulative HCPVs of oil produced from the oil field at the end of the time associated with *fldtim(kt)*. This includes oil and hydrocarbon gas produced

fldcvoilpr(kt) – cumulative volume of oil produced from the oil field at the end of the time associated with *fldtim(kt)* (MSTB)

fldhcpvwatpr(kt) – cumulative HCPVs of water produced from the oil field at the end of the time associated with *fldtim(kt)*

fldcvwatpr(kt) – cumulative volume of water produced from the oil field at the end of the time associated with *fldtim(kt)* (MSTB)

fldhcpvco2pr(kt) – cumulative HCPVs of CO_2 produced from the oil field at the end of the time associated with *fldtim(kt)*

fldcvco2pr(kt) – cumulative volume of CO_2 produced from the oil field at the end of the time associated with *fldtim(kt)* (MMscf)

fldhcpvtotpr(kt) – cumulative HCPVs of oil, water and CO_2 produced from the oil field at the end of the time associated with *fldtim(kt)*

4.4.9 Field-Level CO_2 Saturations

The FE/NETL Onshore CO_2 EOR Cost Model calculates the field-level saturations of CO_2 in the net pay (i.e., the portion of the reservoir modeled by stream tubes). This calculation uses the pattern-level CO_2 saturations in the net pay that are calculated by StrmtbFlow, the pattern pore volume and the patterns that are operational during CO_2 EOR operations. After a pattern ceases operation, it is assumed that the CO_2 saturation is constant at the CO_2 saturation at the end of pattern operations.

The algorithm used to calculate the field-level saturations of CO_2 in the net pay is provided below. This algorithm calculates the field-level saturations of CO_2 in the net pay in two steps. After the algorithm has been executed, the variable *fldsco2(ky)* stores the field-level saturations of CO_2 in the net pay at the end of CO_2 EOR operations year *ky*. However, in the first step, the algorithm stores the total volume of CO_2 in the net pay in the oil field at the end of CO_2 EOR operations year *ky*. This is the volume of CO_2 at reservoir temperature and pressure. At the end of the second step, this volume of CO_2 has been converted to a field-level CO_2 saturation by dividing the CO_2 volume by the pore volume of all patterns that have become operational at the end of CO_2 EOR operations year *ky*.

In the first step, the algorithm first loops through the years when groups of patterns begin operation. For each group of patterns, there is a second loop that cycles through the remaining years of CO_2 EOR operations. The algorithm uses pattern-level CO_2 saturations in the net pay, which are stored in the array variable *patsco2(jt).* However, this array variable stores pattern-level data at times associated with the array variable *pattim(jt)*. The time associated with *pattim(1)* is zero, the time when the pattern first begins operation, and there is no CO_2 in the pattern at that time, so *patsco2(1)* is also zero. Because groups of patterns become operational in different years, the volume of CO_2 in any given CO_2 EOR operations year will be different for different groups of patterns. The second loop adds the appropriate volume of CO_2 from each group to the sum of CO_2 in the oil field at the end of each CO_2 EOR operations year.

At the beginning of the second step, the array variable *fldsco2(ky)* stores the total volume of CO_2 in the net pay at the end of CO_2 EOR year *ky*. In the second step, this volume is divided by the total pore volume of all the patterns that have been implemented by the end of CO_2 EOR operations year *ky*. This translates the CO_2 volume to a saturation:

```
!
! First step: Calculate the total volume of CO2 in the net pay at the end of
! each CO2 EOR operations year
!
! First loop through the years when patterns become active
Do kybeg = 1, FldDevyr
```

```
        !
        ! Loop through all the years when the patterns started in year
        !   kybeg are active
        ! and extend to the years when the pattern ceases operation
        Do ky = kybeg, nfldyr
                jt1 = ky - kybeg + 2
                !
                ! The index jt points to a time when the pattern operates
                ! The time associated with this index is in the array pattim(jt)
                !   When jt = 1, the index points to time 0
                !   When jt > 1, the index points to the year (jt1 - 1) when
                !   pattern operations end
                If( jt1 <= npattim ) Then
                        !
                        ! The pattern is operational and the index jt2 points to
                        ! the pattern-level CO2 saturation associated with the
                        ! pattern time pattim(jt2)
                        jt2 = jt1
                        fldsco2(ky) = fldsco2(ky) + NPateayr(kybeg) * patsco2(jt2) * patpv
                Else
                        !
                        ! The pattern is no longer operating so the index j2
                        ! points to the pattern-level CO2 saturation associated with the
                        ! last time when the pattern operated, pattim(jnpattim)
                        jt2 = npattim
                        fldsco2(ky) = fldsco2(ky) + NPateayr(kybeg) * patsco2(jt2) * patpv
                End If
        End Do
End Do
!
! The array fldsco2(ky) stores the total volume of CO2 in the stream tubes
! in the oil field.
! The elements in the array fldsco2(ky) are divided by the pore volume of
! the patterns that have been implemented by the end of
! CO2 EOR operations year ky to calculate field-level CO2 saturations
! in the stream tubes.
Do ky = 1, nfldyr
        If( ky < FldDevyr ) Then
                fldsco2(ky) = fldsco2(ky) / (Npatacyr(ky) * patpv)
        Else
                fldsco2(ky) = fldsco2(ky) / (Npattot * patpv)
        End If
End Do
```

Where:

FldDevyr – number of years needed to implement all the patterns

kybeg – index associated with year when a set of patterns begin operations

ky – index associated with year when CO_2 operations occur (*ky = 1* is first year when CO_2 EOR operations begin for the first patterns implemented)

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

jt1, jt2 – indices pointing to location in arrays associated with the time of pattern operation. The time is in array *pattim(jt)* with the first element (*pattim(1)*) equal to 0, the time when the pattern begins operation

npattim – number of elements in arrays where array elements store values for the pattern associated with the time stored in the array *pattim(jt)*

fldsco2(ky) – field-level average CO_2 saturation at end of CO_2 EOR operations year *ky*. This array initially stores the total volume of CO_2 in the oil field at reservoir conditions (in 1000 res-ft^3). The array values are eventually adjusted to store the average CO_2 saturation in the oil field. Initially only patterns that have been implemented are included in the calculation but after *ky* reaches and exceeds *FldDevyr* all patterns are considered

Npateayr(kybeg) – array variable used to store the number of patterns that begin operation by the end of CO_2 EOR operation year *kybeg*

patsco2(jt) – average saturation of CO_2 in stream tubes in pattern at the time associated with *pattim(jt)*

Npatacyr(ky) – array variable storing the number of patterns active in CO_2 EOR operational year *ky*

patpv – pore volume of the pattern (1000 res-ft^3)

Npattot – total number of patterns implemented at the oil field

4.4.10 Field-Level Volumes of CO_2 in Different Aspects of Reservoir and Associated CO_2 Storage Coefficients

The FE/NETL Onshore CO_2 EOR Cost Model tracks CO_2 in different aspects of the reservoir. In the net pay (i.e., the stream tubes), CO_2 can be present as a pure phase or dissolved in water or brine. In the "other" portion of the reservoir, CO_2 can be present as a pure phase. The FE/NETL Onshore CO_2 EOR Cost Model utilizes an algorithm that uses the pattern-level volumes of pure phase CO_2 in the stream tubes, CO_2 dissolved in water in the stream tubes and pure phase CO_2 in the "other" portion of the reservoir. The algorithm also uses the patterns that are operational during CO_2 EOR operations. After a pattern ceases operation, it is assumed that the CO_2 volumes in different aspects of the reservoir at future times equal their values at the end of pattern operations.

The algorithm first loops through the years when groups of patterns begin operation. For each group of patterns, there is a second loop that cycles through the remaining years of CO_2 EOR operations. The algorithm uses pattern-level CO_2 volumes in different aspects of the reservoir to generate volumes at the field-level. The pattern-level array variables store pattern-level data at times associated with the array variable *pattim(jt)*. The time associated with *pattim(1)* is zero, the time when the pattern first begins operation, and there is no CO_2 in the pattern at that time, so the pattern-level arrays storing volumes of CO_2 in different aspects of the pattern are also zero. Because groups of patterns become operational in different years, the pattern-level volume of CO_2 in different aspects of the pattern in any given CO_2 EOR operations year will be different for different groups of patterns. The second loop adds the appropriate volume of CO_2 from each group to the sum of CO_2 in the oil field at the end of each CO_2 EOR operations year.

```
Do kybeg = 1, FldDevyr
    Do ky = kybeg, nfldyr
        jt1 = ky - kybeg + 2
        If( jt1 <= npattim ) Then
            jt2 = jt1
            fldVstbco2pp(ky) = fldVstbco2pp(ky) + NPateayr(kybeg) *
                patVstbco2pp(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVstbco2dw(ky) = fldVstbco2dw(ky) + NPateayr(kybeg) *
                patVstbco2dw(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVstbco2tot(ky) = fldVstbco2tot(ky) + NPateayr(kybeg) *
                patVstbco2tot(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVothco2pp(ky) = fldVothco2pp(ky) + NPateayr(kybeg) *
                patVothco2pp(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVco2tot(ky) = fldVstbco2tot(ky) + fldVothco2pp(ky)
        Else
            jt2 = npattim
            fldVstbco2pp(ky) = fldVstbco2pp(ky) + NPateayr(kybeg) *
                patVstbco2pp(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVstbco2dw(ky) = fldVstbco2dw(ky) + NPateayr(kybeg) *
                patVstbco2dw(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVstbco2tot(ky) = fldVstbco2tot(ky) + NPateayr(kybeg) *
                patVstbco2tot(jt2) * 10^6 res-ft^3/MM res-ft^3
            fldVothco2pp(ky) = fldVothco2pp(ky) + NPateayr(kybeg) *
                patVothco2pp(j2) * 10^6 res-ft^3/MM res-ft^3
            fldVco2tot(ky) = fldVstbco2tot(ky) + fldVothco2pp(ky)
        End If
    End Do
End Do
```

Where:

FldDevyr – number of years needed to implement all the patterns

kybeg – index associated with year when a set of patterns begin operations

ky – index associated with year when CO_2 operations occur (*ky = 1* is first year when CO_2 EOR operations begin for the first patterns implemented)

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

jt1, jt2 – indices pointing to location in arrays associated with the time of pattern operation. The time is in array *pattim(jt)* with the first element (*pattim(1)*) equal to 0, the time when the pattern begins operation

npattim – number of elements in arrays where array elements store values for the pattern associated with the time stored in the array *pattim(jt)*

fldVstbco2pp(ky) – volume of pure phase CO_2 in net pay (i.e., stream tubes) in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3)

Npateayr(kybeg) – array variable used to store the number of patterns that begin operation by the end of CO_2 EOR operation year *kybeg*

patVstbco2pp(jt) – volume of pure phase CO_2 in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

fldVstbco2dw(ky) – volume of CO_2 dissolved in water in stream tubes in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3); this is volume of CO_2 dissolved in water as an equivalent volume of pure phase CO_2 under reservoir conditions

patVstbco2dw(jt) – volume of CO_2 dissolved in water in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3); this is volume of CO_2 dissolved in water as an equivalent volume of pure phase CO_2 under reservoir conditions

fldVstbco2tot(ky) – volume of both pure phase CO_2 and CO_2 dissolved in water in stream tubes in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3)

patVstbco2tot(jt) – volume of both pure phase CO_2 and CO_2 dissolved in water in stream tubes in pattern in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

fldVothco2pp(ky) – volume of pure phase CO_2 in "other" portion of the reservoir in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3)

patVothco2pp(jt) – volume of pure phase CO_2 in "other" portion of the reservoir in reservoir volume at the time associated with *pattim(jt)* (MM res-ft^3)

fldVco2tot(ky) – volume of CO_2 in reservoir in the oil field (this volume includes the volumes of pure phase CO_2 in stream tubes, CO_2 dissolved in water in stream tubes and pure phase CO_2 in "other" portion of the reservoir) at the end of CO_2 EOR operations year *ky* (res-ft^3)

The volumes of CO_2 in different aspects of the oil field are converted into CO_2 storage coefficients that reflect the fraction of pore space volume occupied by CO_2. However, different

pore space volumes can be employed to calculate field-level CO_2 storage coefficients. In the algorithm presented below, first field-level pore volumes are calculated. The pattern-level pore volume of the "other" portion of the reservoir is stored in the variable *othpv*. In StrmtbFlow, only a fraction of the "other" portion of the reservoir can be used to store CO_2. The pore volume in the "other" portion of the reservoir that can store CO_2 is stored in the variable *patVothavpv*.

After the field-level pore space volumes are calculated, the following field-level CO_2 storage coefficients are calculated by the FE/NETL Onshore CO_2 EOR Cost Model:

- *fldstbco2scfpp(ky)* – field-level CO_2 storage coefficient for pure phase CO_2 in net pay (i.e., stream tubes) in CO_2 EOR operations year *ky*. This is the volume of pure phase CO_2 in the stream tubes at reservoir conditions divided by the pore space in the stream tubes across the oil field. This variable is the same as the average saturation of CO_2 in the stream tubes (the variable *fldsco2(ky)* calculated above)
- *fldstbco2scfdw(ky)* – field-level CO_2 storage coefficient for CO_2 dissolved in water in net pay (i.e., stream tubes) in CO_2 EOR operations year *ky*. This is the mass of CO_2 dissolved in water in the stream tubes converted to an equivalent volume of CO_2 at reservoir conditions and then divided by the total pore space in the stream tubes across the oil field
- *fldstbco2scftot(ky)* – field-level CO_2 storage coefficient for pure phase CO_2 and CO_2 dissolved in water in net pay (i.e., stream tubes) at the end of CO_2 EOR operations year *ky*. This is the volume of CO_2 in the pure phase and dissolved in water (at reservoir conditions) in the stream tubes divided by the pore space of the stream tubes across the oil field
- *fldothco2scfpp(ky)* – field-level CO_2 storage coefficient for pure phase CO_2 in "other" portion of the reservoir at the end of CO_2 EOR operations year *ky*. This is the volume of pure phase CO_2 in the "other" portion of the reservoir at reservoir conditions divided by the pore space in the "other" portion of the reservoir across the oil field
- *fldco2scftot(ky)* – field-level CO_2 storage coefficient for pure phase CO_2 and CO_2 dissolved in water in the reservoir (net pay and the "other" portion of the reservoir). This is the total volume of CO_2 in the reservoir at reservoir conditions divided by the pore space of the reservoir across the oil field. The volume of CO_2 includes the volume of CO_2 in the pure phase in the stream tubes and the "other" portion of the reservoir, and the volume of CO_2 dissolved in water in the stream tubes. The reservoir pore space is the pore space of the stream tubes and the pore space of the "other" portion of the reservoir

The following is the algorithm for calculating the applicable pore space volumes and the CO_2 storage coefficients:

```
!
! Calculate pattern-level pore volumes of stream tubes and the
! "other" portion of the reservoir
stbpv = patpv * 1000.0 res-ft3/M res-ft3
```

```
othpv = stbpv * (formthick - thick) / thick
!
! Calculate field-level pore volumes at the end of each CO2 EOR
!  operation year
Do ky = 1, nfldyr
    If( ky < FldDevyr ) Then
        fldstbpvol(ky) = stbpv * Npatacyr(ky)
        fldothpvol(ky) = othpv * Npatacyr(ky)
        fldothpvolav(ky) = patVothavpv * Npatacyr(ky) * 10^6 res-ft^3/MM res-ft^3
        fldpvol(ky) = fldstbpvol(ky) + fldothpvol(ky)
    Else
        fldstbpvol(ky) = stbpv * Npattot
        fldothpvol(ky) = othpv * Npattot
        fldothpvolav(ky) = patVothavpv * Npattot * 10^6 res-ft^3/MM res-ft^3
        fldpvol(ky) = fldstbpvol(ky) + fldothpvol(ky)
    End If
End Do
!
! Calculate field-level CO2 storage coefficients at the end of each CO2 EOR
!  operation year
Do ky = 1, nfldyr
    fldstbco2scfpp(ky) = fldVstbco2pp(ky) / fldstbpvol(ky)
    fldstbco2scfdw(ky) = fldVstbco2dw(ky) / fldstbpvol(ky)
    fldstbco2scftot(ky) = fldVstbco2tot(ky) / fldstbpvol(ky)
    fldothco2scfpp(ky) = fldVothco2pp(ky) / fldothpvol(ky)
    fldco2scftot(ky) = fldVco2tot(ky) / fldpvol(ky)
End Do
```

Where:

stbpv – pattern-level pore volume of the stream tubes (res-ft^3)

patpv – pattern-level pore volume of the stream tubes (M res-ft^3)

othpv – pore volume of the "other" portion of the reservoir (res-ft^3)

thick – thickness of the net pay (i.e., stream tubes) of the pattern (ft)

formthick – thickness of the reservoir (ft)

ky – index associated with year when CO_2 operations occur (*ky = 1* is first year when CO_2 EOR operations begin for the first patterns implemented)

nfldyr – number of years fluid is injected into and produced from the oil field during CO_2 EOR operations

FldDevyr – number of years needed to implement all the patterns

fldstbpvol(ky) – field-level pore volume of net pay (i.e., stream tubes) in CO_2 EOR operations year *ky* (ft^3)

Npatacyr(ky) – array variable storing the number of patterns active in CO_2 EOR operational year *ky*

fldothpvol(ky) – field-level pore volume of "other" portion of reservoir in CO_2 EOR operations year *ky* (ft^3)

fldothpvolav(ky) – field-level pore volume of "other" portion of reservoir that is available for storing CO_2 in CO_2 EOR operations year *ky* (ft^3)

patVothavpv – pattern-level volume in the "other" portion of the reservoir that is available for storing CO_2 (MM ft^3)

fldpvol(ky) – field-level pore volume of net pay and "other" portion of the reservoir (i.e., pore volume over the entire thickness of the reservoir) in CO_2 EOR operations year *ky* (ft^3)

Npattot – total number of patterns implemented at the oil field

fldstbco2scfpp(ky) – field-level CO_2 storage coefficient for pure phase CO_2 in net pay (i.e., stream tubes) in CO_2 EOR operations year *ky*; fraction of stream tube pore space for entire oil field that is filled with pure phase CO_2

fldVstbco2pp(ky) – volume of pure phase CO_2 in net pay (i.e., stream tubes) in oil field at the end of CO_2 EOR operations year ky (res-ft^3)

fldstbco2scfdw(ky) – field-level CO_2 storage coefficient for CO_2 dissolved in water in net pay (i.e., stream tubes) in CO_2 EOR operations year *ky*; fraction of stream tube pore space for entire oil field that is filled with CO_2 dissolved in water

fldVstbco2dw(ky) – volume of CO_2 dissolved in water in stream tubes in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3); this is volume of CO_2 dissolved in water as an equivalent volume of pure phase CO_2 under reservoir conditions

fldstbco2scftot(ky) – field-level CO_2 storage coefficient for pure phase CO_2 and CO_2 dissolved in water in net pay (i.e., stream tubes) at the end of CO_2 EOR operations year *ky*; fraction of stream tube pore space for entire oil field that is filled with pure phase CO_2 and CO_2 dissolved in water

fldVstbco2tot(ky) – volume of both pure phase CO_2 and CO_2 dissolved in water in stream tubes in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3)

fldothco2scfpp(ky) – field-level CO_2 storage coefficient for pure phase CO_2 in "other" portion of the reservoir at the end of CO_2 EOR operations year *ky*; fraction of pore space in the "other" portion of the reservoir for entire oil field that is filled with pure phase CO_2

fldVothco2pp(ky) – volume of pure phase CO_2 in "other" portion of the reservoir in oil field at the end of CO_2 EOR operations year *ky* (res-ft^3)

fldco2scftot(ky) – field-level CO_2 storage coefficient for pure phase CO_2 and CO_2 dissolved in water in the reservoir (net pay and the "other" portion of the reservoir); fraction of pore space in the reservoir for entire oil field that is filled with pure phase CO_2 and CO_2 dissolved in water

fldVco2tot(ky) – volume of CO_2 in reservoir in the oil field (this volume includes the volumes of pure phase CO_2 in stream tubes, CO_2 dissolved in water in stream tubes and pure phase CO_2 in "other" portion of the reservoir) at the end of CO_2 EOR operations year *ky* (res-ft^3)

4.5 Escalation and Discount Factors and Cash Flows

In the FE/NETL Onshore CO_2 EOR Cost Model, cash flows are calculated for many variables. All cash flows are initially calculated as constant dollars, with the base year for the constant dollars determined by the user. The expression "BY$" is used to refer to base year dollars. For the inputs currently provided with the model, the base year is 2018. The user must adjust the default prices and costs in the model if the user desires another year to be the base year. The base year is specified by the variable *basecstyr* in the model, which is a user input.

The various cash flows utilized in the model are illustrated with the example variable *excf*. For this variable, the constant dollar cash value in year *iy* is given by the variable *excfcn(iy)*. The letters "cn" at the end of *excfcn* indicate this is the constant dollar cash value for variable *excf* in year *iy*. The index *iy* equals 1 in the first year of the project.

The constant dollar cash flows are escalated to nominal dollars with a user-defined escalation rate (*escrate*) which should reflect the user's best estimate of how revenues and costs in the oil industry will increase (inflate) or decrease (deflate) over the next 30 to 50 years. The user can specify a start year (*startyr*) for the project that is different from the base year. The influence of a starting year different from the base year on the escalation factor for year *iy* (*escfac(iy)*) is

$$iiyr = iy + startyr - basecstyr + 1$$

$$escfac(iy) = (1 + escrate)^{iiyr}$$

When *iy* equals 1 and the starting year equals the base year, the escalation factor is 1. If the starting year is greater than the base year and the escalation rate is positive, then the escalation factor will be greater than 1 when *iy* equals 1.

For all values of *iy* greater than 1, the escalation factor is greater than 1, assuming the escalation rate is positive. For variable *excf*, the nominal dollar value in year *iy*, *excfnm(iy)*, is calculated with the following equation.

$$excfnm(iy) = excfcn(iy) \cdot escfac(iy)$$

The letters "nm" at the end of *excfnm* indicate this is the nominal dollar value for variable *excf* in year *iy*.

The nominal dollar cash flows are then discounted to give cash flows in present value dollars. The discount rate (*disc*) is a nominal discount rate and depends on several user defined inputs. The purpose of discounting is to generate cash flows that reflect the time value of money. In

general, a dollar available today is worth more than a dollar available in five or ten years. A dollar available today can, at a minimum, be invested in a relatively safe investment (such as government backed savings bonds) and earn interest for five or ten years. For year *iy* of a CO_2 EOR project, the discount factor (*discfac(iy)*) is:

$$discfac(iy) = \frac{1}{(1 + disc)^{iy-1}}$$

The variable *disc* is a function of user defined financial parameters, typically the weighted average cost of capital. These financial factors are discussed in more detail in a subsequent section. When *iy* equals 1, the discount factor is 1. For all values of *iy* greater than 1, the discount factor is less than 1. This reflects the concept that a dollar in year *iy* is worth less than a dollar in year 1. The present value dollars are relative to the starting year of the project (specified by the variable *startyr*). For variable *excf*, the present value in year *iy*, *excfnm(iy)*, is calculated with the following equation:

$$excfpv(iy) = excfnm(iy) \cdot discfac(iy)$$

The letters "pv" at the end of *excfpv* indicate this is the present value for variable *excf* in year *iy*.

In subsequent sections, equations are provided for calculating the cash flows for a variety of revenue and cost elements in constant dollars. In the FE/NETL Onshore CO_2 EOR Cost Model, cash flows are also calculated for these same revenue and cost elements in nominal dollars and present value dollars using the equations presented in this section. However, the equations used to calculate nominal and present value cash flows for individual revenues or cost elements are not presented.

4.6 REVENUES

There are two potential sources of revenue in the FE/NETL Onshore CO_2 EOR Cost Model. One source is oil, while the other is CO_2. Currently, CO_2 EOR operators must pay to obtain CO_2, so CO_2 is a cost and not a source of revenue. At some time in the future, emitters of CO_2 may be required to capture their CO_2 and store the captured CO_2 in the subsurface. In that situation, the emitters of CO_2 might pay CO_2 EOR operators to use and store their CO_2.

Additionally, the Internal Revenue Service has issued the Section 45Q tax credit, which provides up to $35 per tonne for CO_2 stored as part of CO_2 EOR operations. [3] Depending on the market price of CO_2, and given the possibility for applying additional state-sponsored carbon tax credits, CO_2 storage as part of CO_2 EOR could provide a negative cost.

4.6.1 Oil Revenue

The revenue from oil depends on the oil price. There are three oil prices used in the FE/NETL Onshore CO_2 EOR Cost Model:

1. *oilprice* – market oil price in constant dollars (BY$/STB). This is the price of oil that is publicly traded.
2. *gaoilprice* – API gravity adjusted oil price in constant dollars (BY$/STB). This price is dependent on the API oil gravity, which determines the overall energy content of the

produced oil. Lighter oils will have less energy content by volume, and thus the oil price will have a greater adjustment than a heavier oil with greater energy content per volume. As API gravity increases, the oil becomes lighter.

3. *loilprice* – lease oil price in constant dollars (BY$/STB). This is the price of oil at the lease, which is the price used to calculate the oil revenue that the CO_2 EOR operator experiences.

The variable *gaoilprice* is calculated with the following algorithm:

If(API > 45.0) Then
gaoilprice = oilprice - 0.15 · (API - 45.0)
Else If(API <= 45.0 and API > 40.0) Then
gaoilprice = oilprice
Else If(API <= 40.0 and API > 35.0) Then
gaoilprice = oilprice - 0.02 · (40.0 - API)
Else If(API <= 35.0 and API > 24.0) Then
gaoilprice = oilprice - (0.1 + 0.15 · (35.0 - API))
Else
gaoilprice = oilprice - (1.75 + 0.35 · (24.0 - API))
End If

Where:

API – API gravity of the crude oil (deg API)

The variable *loilprice* is given by the following equation:

$$loilprice = gaoilprice - transport(st)$$

Where:

transport(st) – cost associated with transporting the crude oil from the oil field to a refinery for processing (BY$/STB). This variable also includes the cost of operating and maintaining the oil connecting pipeline that transports the oil from the CO_2 EOR facility to an oil trunkline that transports the oil to a refinery. The variable *transport* is an array and the index *st* points to the location in *transport* that stores data for the state where the oil field is located

The first year of the CO_2 EOR project involves the implementation of equipment needed to start CO_2 EOR operations. Thus, no oil is produced in the first year of the CO_2 EOR project, so the revenue is zero in the first year. For years after the first year, the gross revenue generated from the sale of oil in year *iy* (where *iy* > 1) is given by:

$$fldoilgrevcn(iy) = fldyrvoilpr(iy - 1) \cdot loilprice$$

Where:

fldoilgrevcn(iy) – gross revenue at the lease generated from the sale of oil in CO_2 EOR project year *iy* in constant dollars (BY$)

fldyrvoilpr(iy-1) – oil produced in CO_2 EOR project year *iy* (STB/yr). This variable stores the oil produced each year of the CO_2 EOR operations, where the first element of this array stores the oil produced in the first year of CO_2 EOR operations. This is the second year of the CO_2 EOR project

loilprice – lease oil price (BY$/STB)

4.6.2 CO_2 Revenue

As noted above, CO_2 EOR operators must pay to obtain CO_2, so CO_2 is not a source of revenue. At some point in the future, CO_2 EOR operators may get paid to store CO_2, in which case CO_2 would be a source of revenue. As discussed above, there are no CO_2 EOR operations until the second year of the CO_2 EOR project (when *iy* is 2). For years after the first year, the gross revenue generated from the storage of CO_2 in year *iy* (where *iy* > 1) is given by:

$$fldco2grevcn(iy) = fldyrvco2purch(iy-1) \cdot co2pricerev$$

Where:

fldco2grevcn(iy) – gross revenue generated from the storage of CO_2 in CO_2 EOR project year *iy* (BY$)

fldyrvco2purch(iy-1) – CO_2 accepted for storage in operational year *iy* (MMscf/yr). This variable stores the CO_2 purchased each year of the CO_2 EOR operations, where the first element of this array stores the CO_2 purchased in the first year of CO_2 EOR operations. This is the second year of the CO_2 EOR project

co2pricerev – payment for storage of CO_2 if it is a source of revenue to the CO_2 EOR operator (BY$/Mscf or MBY$/MMscf)

The variable *co2pricerev* is determined by the user. The user inputs a value for the variable *co2price*. If *co2price* is negative (specifically, less than -1.0×10^{-6}), then it is assumed that CO_2 is a source of revenue. In this case, the value for *co2pricerev* is the negative of *co2price* as given by the following equation:

$$co2pricerev = -co2price$$

4.6.3 Total Revenue

The total gross revenue from the sale of oil and storage of CO_2 in CO_2 EOR project year *iy* is given by the following equation:

$$fldgrevcn(iy) = fldoilgrevcn(iy) + fldco2grevcn(iy)$$

Where:

fldgrevcn(iy) – total gross revenue generated from the sale of oil and storage of CO_2 in CO_2 EOR project year *iy* (BY$)

4.7 ROYALTIES

In the FE/NETL Onshore CO_2 EOR Cost Model, one of the costs incurred by the CO_2 EOR operator is the royalties that must be paid to the owner of the mineral rights to the oil. In the model, royalties are calculated as a percentage to the gross revenue at the lease generated by oil production. The fraction of the oil revenue that goes to royalties is input by the user. This factor is often state-specific.

The royalty payment in CO_2 EOR project year *iy* is given by the following equation:

$$fldroycn(iy) = fldoilgrevcn(iy) \cdot aroy(st)$$

Where:

fldroycn(iy) – royalty payment in constant dollars in CO_2 EOR project year *iy* (BY$)

aroy(st) – state-specific royalty rate (fraction). The variable *aroy* is an array and the index *st* points to the location in *aroy* that stores data for the state where the oil field is located

4.8 SEVERANCE AND AD VALOREM TAXES

The FE/NETL Onshore CO_2 EOR Cost Model applies state severance and ad valorem taxes as costs to the CO_2 EOR operator. These taxes are calculated as a fraction of the gross oil revenue at the lease less any royalty payment to the mineral owner. The severance and ad valorem taxes are state-specific, and are given by the following:

$$fldsevcn(iy) = (fldoilgrevcn(iy) - fldroycn(iy)) \cdot asev(st) \cdot 0.01$$

$$fldavtcn(iy) = (fldoilgrevcn(iy) - fldroycn(iy)) \cdot aadval(st) \cdot 0.01$$

Where:

fldsevcn(iy) – severance tax in constant dollars in CO_2 EOR project year *iy* (BY$)

asev(st) – state-specific severance tax rate (percent). The variable *asev* is an array and the index *st* points to the location in *asev* that stores data for the state where the oil field is located

fldavtcn(iy) – ad valorem tax in constant dollars in CO_2 EOR project year *iy* (BY$)

aadval(st) – state-specific ad valorem tax rate (percent). The variable *aadval* is an array and the index *st* points to the location in *aadval* that stores data for the state where the oil field is located

Some states have incentives for CO_2 EOR or the use of anthropogenic CO_2 that can be used to reduce the severance or ad valorem tax rates. The user can have these alternative tax rates applied by specifying specific values for the user variable *incentive*. The severance tax rates and ad valorem tax rates are given by the following algorithm.

```
If( incentive == 1 ) Then
    asev(st) = asev1(st)
    aadval(st) = aadval1(st)
```

Else If(incentive == 2) Then
asev(st) = asev2(st)
aadval(st) = aadval2(st)
Else
asev(st) = asev_b(st)
aadval(st) = aadval_b(st)
End If

Where:

incentive – variable controlling the severance and ad valorem tax rate used in the analysis

asev(st) – state-specific severance tax rate (percent). The variable *asev* is an array and the index *st* points to the location in *asev* that stores data for the state where the oil field is located

asev_b(st) – baseline state-specific severance tax rate (percent)

asev1(st) – state-specific severance tax rate if state has a reduced severance tax rate for implementing CO_2 EOR (percent)

asev2(st) – state-specific severance tax rate if state has a reduced severance tax rate for using anthropogenic CO_2 in a CO_2 EOR project (percent)

aadval(st) – state-specific ad valorem tax rate (percent). The variable *aadval* is an array and the index *st* points to the location in *aadval* that stores data for the state where the oil field is located

aadval_b(st) – baseline state-specific ad valorem tax rate (percent)

aadval1(st) – state-specific ad valorem tax rate if state has a reduced ad valorem tax rate for implementing CO_2 EOR (percent)

aadval2(st) – state-specific ad valorem tax rate if state has a reduced ad valorem tax rate for using anthropogenic CO_2 in a CO_2 EOR project (percent)

The variables *asev_b(st), asev1(st)*, *asev2(st)*, *aadval_b(st)*, *aadval1(st)*, *aadval2(st)*, and *incentive* are user inputs.

4.9 CAPITAL COSTS

This section presents the equations used to calculate capital costs for each cost element in the FE/NETL Onshore CO_2 EOR Cost Model. In the model, capital costs are incurred in the year before the element begins useful production. Some elements (such as the CO_2 delivery pipeline) are one-time expenses while others (such as the cost of drilling and completing a new injection well) are incurred over time as the cost elements are implemented.

4.9.1 Acquisition of Leases and Permits

This cost element is a greenfield cost and covers the cost of obtaining leases, paying lease bonuses and obtaining all permits needed for a new oil field. These costs are included in the cash flow only if the variable *grnfldcon* = 1.

The cost of obtaining leases and mineral rights, paying lease bonuses and obtaining selected permits in CO_2 EOR project year *iy* is given by the following equation which depends on the number of patterns implemented each year:

$$fldacqpatcapcn(iy) = NPateayr(iy) \cdot acqpattcap \cdot fac$$

Where:

fldacqpatcapcn(iy) – capital cost of acquiring leases, paying lease bonuses and obtaining selected permits in constant dollars (BY$)

NPateayr(iy) – number of patterns implemented in CO_2 EOR project year *iy*

acqpattcap – unit capital costs of acquiring leases, paying lease bonuses and obtaining selected permits (BY$/patt)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The unit cost variable *acqpattcap* is calculated with the following equation:

$$acqpattcap = leasbon \cdot patarea + (leaslab + unitlab + permothfix)/NPatTot + (permothvar/640.0) \cdot patarea$$

Where:

leasebon – lease bonus in constant dollars (BY$/ac)

patarea – area of pattern (ac)

leaslab – labor costs for obtaining lease, mostly landman in constant dollars (BY$)

unitlab – labor costs for unitization, mostly landman and lawyers in constant dollars (BY$)

permothfix – fixed site-wide costs for other permits in constant dollars (BY$)

NPatTot – total number of patterns implemented

permothvar – variable costs for other permits (BY$/$mi^2$)

The variables *leaslab, unitlab,* and *permothfix* are costs for the entire oil field. These have been divided by the total number of patterns (*NPatTot*) to provide a cost per pattern. Values for the variables *leasbon*, *leaslab, unitlab, permothfix,* and *permothvar* are input by the user.

4.9.2 Seismic Imaging for Characterization

This cost element is a greenfield cost and covers the cost of performing three-dimensional (3-D) seismic imaging, which provides reservoir characterization data that is used to design the CO_2 flood. Seismic imaging costs are incurred in the first year for a greenfield CO_2 EOR project

where no oil recovery has taken place. Seismic imaging costs are calculated for the entire oil field. These costs are included in the cash flow only if the variable *grnfldcon* = 1.

The cost for 3-D seismic imaging of the reservoir in year 1 is given by the following equation:

$$fldseiscapcn(1) = seiscap \cdot fac$$

Where:

fldseiscapcn(1) – capital cost of seismic imaging for characterization in CO_2 EOR project year 1 of a greenfield CO_2 EOR project (MBY$)

seiscap – capital cost for 3-D seismic imaging (BY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The variable *seiscap* is calculated assuming the oil field is a square and seismic imaging covers a greater area than the field itself to resolve the full field reservoir. It is assumed an area the size of the field plus a horizontal length equal to the depth of the field is used. This creates an area that will connect from the surface to the furthest reservoir edge by a 45-degree angle, which ensures the entirety of the reservoir is mapped by the 3-D seismic imaging.

The capital cost for 3-D seismic imaging, *seiscap*, is calculated using the following equations:

$$seiscap = seisarea \cdot seis3dcost \cdot (1.0 + seis3dprocmult)$$

$$seisarea = seisside^2/(5280^2)$$

$$seisside = fldside + 2 \cdot seisdepth$$

$$fldside = Sqrt(fldarea \cdot 43560)$$

$$seisdepth = depth + grosspay$$

Where:

seisarea – area needed for seismic imaging (mi^2)

seis3dcost – cost of 3-D seismic imaging (BY$/$mi^2$)

seis3dprocmult – cost of processing 3-D seismic imaging as a fraction of 3-D seismic imaging costs (equal to 0 unless otherwise specified by the user)

seisside – length of the side of the area needed for 3-D seismic imaging in order to image to depth *seisdepth* at edge of the oil field (ft)

fldside – length of oil field assuming it is a square (ft)

seisdepth – distance to deepest depth for seismic imaging (ft)

fldarea – area of the oil field (ac)

depth – reservoir depth (ft)

grosspay – gross reservoir pay thickness (ft)

The variables *seis3dcost* and *seis3dprocmult* are user inputs. The variables *fldarea*, *depth*, and *grosspay* are properties of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

4.9.3 Office Building and Access Road

This cost element is a greenfield cost and provides the construction costs for a new on-site field operation building, supply depot, and access road. These costs are incurred for all greenfield projects, but the costs are assumed to have already been incurred at brownfield projects. The model assumes that one new building and one new access road is constructed for each greenfield CO_2 EOR project. These costs are included in the cash flow only if the variable *grnfldcon* = 1.

The cost for constructing new on-site field office buildings and access roads in year 1 of the CO_2 EOR project is given by the following equation:

$$fldbldgroadcapcn\,(1) = (bldgcap + roadcap) \cdot fac$$

Where:

fldbldgroadcapcn(1) – capital cost of constructing office buildings and access roads in the first year of the CO_2 EOR project (MBY$)

bldgcap – capital cost for constructing office buildings (BY$)

roadcap – capital cost for access roads to the office building (BY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The capital cost of building office buildings (the variable *bldgcap*) is calculated with the following equation:

$$bldgcap = bldgarea \cdot bldgunitcap$$

Where:

bldgarea – area of the office building(s) (ft^2)

bldgunitcap – capital cost for the office building in (BY$/$ft^2$)

The variables *bldgarea* and *bldgunitcap* are user inputs.

The capital cost of constructing access roads (the variable *roadcap*) is calculated with the following equation:

$$roadcap = roadlen \cdot roadunitcap$$

Where:

roadlen – length of access road (mi)

roadunitcap – unit capital costs for access road (BY$/mi)

The variables *roadlen* and *roadunitcap* are user inputs.

4.9.4 Drilling and Completing New Wells

This cost element is applicable to both greenfield and brownfield sites. The cost element covers the cost of drilling and completing new wells. New wells can be production wells, injection wells, water disposal wells, or test characterization wells if this is a greenfield site. The costs of drilling and completing a new well are based on data in the API Joint Association survey of 2014 drilling costs. [4]

4.9.4.1 Basic Well Drilling and Completion Costs

The cost of drilling and completing new production and injection wells in CO_2 EOR project year *iy* is given by the following equation, which depends on the number of new wells drilled each year:

$$flddcwcapcn(iy) = dcwuncapby \cdot (Ninjwnewyr(iy) + Nprwnewyr(iy)) \cdot fac$$

Where:

flddcwcapcn(iy) – capital cost of drilling and completing new production and injection wells in CO_2 EOR project year *iy* (BY$)

dcwuncapby – unit cost of drilling a new production or injection well (BY$/well)

Ninjwnewyr(iy) – number of new injection wells implemented in CO_2 EOR project year *iy*

Nprwnewyr(iy) – number of new production wells implemented in CO_2 EOR project year *iy*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The unit cost variable *dcwuncapby* is calculated with the following equation:

$$dcwuncapby = dcwuncap1 \cdot dcwca$$

Where:

dcwuncap1 – unit cost of drilling a new production or injection well based on data in the API Joint Association survey of 2014 drilling costs (2014$/well)

dcwca – oil-price based cost adjustment factor that converts drilling costs in 2014 to drilling costs in the base year (BY$/2014$)

The variable *dcwuncap1* is given by the following equation based on the API Joint Association survey of well drilling costs:

$$dcwuncap1 = dcb0(rc) - dcb1(rc) \cdot depth1 + dcb2(rc) \cdot depth1^2$$

Where:

depth1 – depth from the surface to the bottom of the well (ft)

dcb0(rc), dcb1(rc), dcb2(rc) – coefficients in equation for well drilling cost as a function of depth from the API Joint Association well drilling costs. These variables are arrays and the index *rc* points to the location in each array that

stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

The variables *dcb0(rc), dcb1(rc)* and *dcb2(rc)* are user inputs.

The variable *depth1* is given by the following equation:

$$depth1 = depth + grosspay$$

Where:

depth – depth from the surface to the top of the gross pay (ft)

grosspay –thickness of reservoir gross pay (ft)

The variables *depth* and *grosspay* are properties of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

The variable *dcwca* uses the market price of oil to adjust the drilling and completion costs in 2014 dollars to drilling and completion costs associated with the user-specified oil price. The average oil price in 2014 was about $90/STB, so this value is the benchmark for the variable *dcwca*. The average drilling costs for a few years before 2014 and including the costs in 2014 did not change significantly even though oil price fluctuated from around $70/STB to $100/STB. [4] However, it is assumed that if long-term oil prices increased or decreased relative to the $90/STB oil price for 2014 then drilling and completion capital costs would increase or decrease, if only moderately.

The drilling and completion costs are believed to be constrained on the high end by competition. If drilling costs increase (i.e., the price demanded by drillers to drill new wells) then this business will have higher margins (assuming all other cost factors remain constant) and this will encourage new entrants to the well drilling market. The new entrants will increase competition, and this should constrain increases in the long-term price for drilling and completing new wells.

The drilling and completion costs are believed to be constrained on the low end by the costs of materials and labor that go into drilling a new well. In the short run, well drillers can lower prices to generate some revenue even if the revenue does not cover all their costs. However, over the long run, the cost of drilling and completing a well must cover the cost of inputs plus financing costs. The costs of inputs (e.g., equipment, materials, labor) can only be reduced so much over the long term. Thus, decreases in the cost of drilling and completing a well are believed to have a lower limit.

The variable *dcwca* is calculated with an equation that uses the variable *oilprice_eff* to adjust drilling costs based on the market price of oil. The variable *oilprice_eff* is set to the market price of oil, *oilprice*, but is not allowed to go above an upper bound oil price, *aoilpr_max*, or below a lower bound oil price *aoilpr_min*. The algorithm for calculating *oilprice_eff* is as follows:

```
If( oilprice > aoilpr_max ) Then
        oilprice_eff = aoilpr_max
Else If( oilprice < aoilpr_min ) Then
        oilprice_eff = aoilpr_min
```

Else

oilprice_eff = oilprice

End If

Where:

oilprice_eff – effective price of oil for adjusting drilling and completion capital costs (BY$/STB)

oilprice – market oil price (BY$/STB)

aoilpr_max – upper bound oil price for adjusting drilling and completion capital costs (BY$/STB)

aoilpr_min – lower bound oil price for adjusting drilling and completion capital costs (BY$/STB)

The variables *oilprice*, *aoilpr_max* and *aoilpr_min* are user inputs.

After calculating *oilprice_eff*, the variable *dcwca* is calculated by the following equation:

$$dcwca = (oilprice\_eff + 90)/(2 \cdot 90)$$

Where:

dcwca – oil price-based cost adjustment factor that converts drilling costs in 2014 dollars to drilling costs in the base year using the effective oil price (BY$/2014$)

oilprice_eff – effective price of oil (BY$/STB)

4.9.4.2 Drilling and Completing Test Characterization Wells

If this is a greenfield site, then costs associated with drilling wells for characterization are based on the costs drilling and completing new wells. The costs for drilling test characterization wells are assumed to be higher than for operational wells because of the additional effort to characterize and log data for the producing intervals during well drilling and completion.

The costs for drilling characterization wells are assumed to be incurred in the first year of operation:

$$flddcwcapcn(1) = flddcwcapcn(1) + fldtestwcapby \cdot fac$$

Where:

fldtestwcapby – capital costs associated with characterization wells (BY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The variable *fldtestwcapby* is given by the following equation:

$$fldtestwcapby = ntestwellchar \cdot dcwuncapby \cdot testwcostmult + ntestwellused \cdot dcwuncapby \cdot (testwcostmult - 1.0)$$

Where:

ntestwellchar – number of test characterization wells installed only for characterization; these test wells are plugged after installation

testwcostmult – cost multiplier for test characterization wells; basic unit costs for drilling and completing a well (*dcwuncapby*) are multiplied by this factor to give test well characterization costs

ntestwellused – number of test wells that are completed as injection or production wells after installation

The user specifies values for the variables *ntestwellchar*, *testwcostmult*, and *ntestwellused*. For test characterization wells that are completed as injection or production wells, the cost of drilling and completing these wells have already been calculated in the equation for *flddcwcapcn(1)* provided in the previous section. Thus, only the cost associated with additional testing is included in the variable *fldtestwcapby*, which is why the expression (*testwcostmult – 1.0*) is used in the equation for *fldtestwcapby.*

4.9.4.3 Water Disposal Wells

At brownfield sites, it is assumed that water disposal wells have already been drilled since water disposal wells are needed for primary and secondary production. However, greenfield CO_2 EOR sites will require capital costs for water disposal wells, if needed.

Capital costs for water disposal wells in the FE/NETL Onshore CO_2 EOR Cost Model are an optional cost determined by the user. Depending on the design of the CO_2 EOR operation, and the volume of produced water from the reservoir, water disposal may not be required at the site. Produced water, to the extent possible, is recycled and used for injection. Produced water may also be transported for use at another nearby oil field.

The user determines whether water disposal well costs are included in the cash flow analysis by the variable *watinjcon*. Setting the variable equal to 1 includes the costs while setting the variable equal to 0 does not include the costs.

The capital costs for water disposal wells in CO_2 EOR project year *iy* is given by the following equation:

$$fldwatinjcapcn(iy) = Nwatinjwnewyr(iy) \cdot watinjwuncapby \cdot fac$$

Where:

fldwatinjcapcn(iy) – capital costs for water injection wells in CO_2 EOR project year *iy* (MBY$)

Nwatinjnewyr(iy) – number of water injection wells installed in CO_2 EOR project year *iy*

watinjwuncapby – unit capital costs for a water disposal well (BY$/well)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The unit capital cost for water disposal wells uses the standard drilling and completion cost for a new well in the oil field discussed earlier (the variable *dcwuncapby*) plus a cost multiplier

determined by the user. The cost multiplier option is designed to reflect additional costs required to drill and complete separate water disposal wells. The unit capital cost is given by the following equation:

$$watinjwuncapby = dcwuncapby \cdot watinjmult$$

Where:

dcwuncapby – unit capital costs for drilling and completing a well in base year dollars (BY$/well)

watinjmult – multiplier applied to conventional well costs to get water injection well costs

The variable *watinjmult* is a user input.

4.9.5 Recompleting Existing Wells

This cost element applies only to brownfield sites. For such sites, there may be existing wells in the field available for use, which reduces the capital costs for drilling and completing new wells. However, these wells require additional costs to recomplete the well casing and upgrade other equipment to be fully functional and reliable.

The cost for recompleting existing injection and production wells in year *it* is given by the following equation:

$$fldrecwcapcn(iy) = recwuncapby \cdot (Ninjwrecyr(iy) + Nprwrecyr(iy)) \cdot fac$$

Where:

fldrecwcapcn(iy) – capital cost for recompleting old wells in CO_2 EOR project year *iy* (MBY$)

recwuncapby – unit cost for recompleting an old well in base year dollars (BY$/well)

Ninjwrecyr(iy) – number of old injection wells available for recompletion in CO_2 EOR project year *iy*

Nprwrecyr(iy) – number of old production wells available for recompletion in CO_2 EOR project year *iy*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The variable *recwuncapby* is determined using region-specific calculations developed from the 2014 JAS drilling and completion survey. The unit cost for recompletion is given by the following set of equations:

$$recwuncapby = recwuncap1 \cdot byca$$

$$recwuncap1 = rcb0(rc) + depth1 \cdot rcb1(rc)$$

$$depth1 = depth + grosspay$$

Where:

recwuncap1 – unit capital cost of recompleting a well (2011$/well)

byca – base year cost adjustment factor (BY$/2011$). This variable translates the costs in 2011 dollars to base year dollars

depth1 – total well depth (ft)

rcb0(rc), rcb1(rc) – coefficients in equation for well recompletion cost as a function of depth from the API Joint Association well drilling costs. These variables are arrays and the index *rc* points to the location in each array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

depth – distance from surface to oil reservoir (ft)

grosspay – thickness of gross pay (ft)

The variables *byca, rcb0(rc),* and *rcb1(rc)* are user inputs. The variables *depth* and *grosspay* are properties of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

4.9.6 Surface Equipment for New Production and Injection Wells

This cost element applies to both greenfield and brownfield sites. Surface equipment for oil field operations are installed after drilling and completion of new wells. Surface equipment components include tanks, pumps, heaters, and blowout preventers. Costs for surface equipment are calculated per well, which includes a fixed cost and a variable cost that depends on the length of the wells. Surface equipment costs are not region-specific.

The cost of installing surface equipment for new wells in CO_2 EOR project year *iy* is given by the following equations:

$$fldseprwcapcn(iy) = seprwuncapby \cdot Nprwnewyr(iy) \cdot fac$$

$$fldseinwcapcn(iy) = seinwuncapby \cdot Ninjwnewyr(iy) \cdot fac$$

Where:

fldseprwcapcn(iy) – capital costs for surface equipment associated with new production wells in CO_2 EOR project year *iy* (MBY$)

seprwuncapby – unit capital costs for surface equipment for new production well in base year dollars (BY$/well)

Nprwnewyr(iy) – number of new production wells installed in CO_2 EOR project year *iy*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

fldseinwcapcn(iy) – capital costs for surface equipment associated with new injection wells in CO_2 EOR project year *iy* (MBY$)

seinwuncapby – unit capital costs for surface equipment for new injection well in base year dollars (BY$/well)

Ninjwnewyr(iy) – number of new injection wells installed in CO_2 EOR project year *iy*

The unit capital cost equations for surface equipment for new production and injection wells are given by the following equations:

$$seprwuncapby = seprwuncap1 \cdot byca$$

$$seinwuncapby = seiwuncap1 \cdot byca$$

$$seprwuncap1 = prfixed + depthprd \cdot prvar$$

$$seinwuncap1 = infixed + depthinj \cdot invar$$

Where:

seprwuncap1 – unit capital costs for surface equipment for new production well (2011$/well)

byca – base year cost adjustment factor (BY$/2011$)

seinwuncap1 – unit capital costs for surface equipment for new injection well (2011$/well)

prfixed – fixed capital cost for production well surface equipment (BY$/well)

depthprd – modified length of production well (ft)

prvar – variable cost based on well length for production well surface equipment (BY$/well)

infixed – fixed capital cost for injection well surface equipment (BY$/well)

depthinj – modified length of injection well (ft)

invar – variable cost based on well length for injection well surface equipment (BY$/well)

These equations depend on the length of the well only if the well exceeds certain lengths. The algorithm for calculating *depthprd* and *depthinj* is:

```
depth1 = depth + grosspay
!
! Calculate depthprd
If( depth1 < 2000.00 ) Then
        depthprd = 0.0
Else
        depthprd = depth1 – 2000.0
End If
!
! Calculate depthinj
If( depth1 < 4000.00 ) Then
        depthinj = 0.0
Else
```

depthinj = depth1 – 4000.0
End If

Where:

depth1 – total well length (ft)

depth – distance from surface to oil reservoir (ft)

grosspay – thickness of gross pay (ft)

depthprd – modified length of production well (ft)

depthinj – modified length of injection well (ft)

The variables *byca, prfixed, prvar, infixed,* and *invar* are user inputs. The variables *depth* and *grosspay* are properties of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

4.9.7 CO_2 Recycling Plant

This cost element applies to both greenfield and brownfield sites. The CO_2 recycling plant collects CO_2 produced from the field and prepares the CO_2 for reinjection. The CO_2 recycling plant includes separators for extracting CO_2 from the produced water and hydrocarbons, dehydrators for dewatering the CO_2, and compressors to pressurize the CO_2 for reinjection. The CO_2 recycling plant also receives purchased CO_2 and combines purchased and recycled CO_2 for distribution to the injection wells.

In the early years of the CO_2 EOR project, virtually all the injected CO_2 will be purchased. Toward the end of the project, much of the injected CO_2 will be recycled. Over the life of the CO_2 EOR project, roughly half of the injected CO_2 will be purchased and half will be recycled.

The model calculates CO_2 recycling plant costs based on the maximum annual volume of CO_2 that the plant will recycle, which is stored in the variable *recmaxrate* and the year when this maximum rate occurs, *recymaxyr*. While this rate is a maximum annual rate, it is expressed as an equivalent daily rate (MMscf/day). The algorithm for determining these two variables is the following:

recymaxrate = 0.0
Do ky = 1, nfldyr
If(fldyrvco2recy(ky) / 365.0 > recymaxrate) Then
recymaxrate = fldyrvco2recy(ky) / 365.0
recymaxyr = ky
End If
End Do

Where:

recymaxrate – maximum rate of CO_2 production for recycling (MMscf/day); this is a maximum annual rate expressed as its equivalent average daily rate

fldyrvco2recy(ky) – volume of CO_2 produced for recycling in year *ky* (MMscf/yr)

recymaxyr – year when maximum rate of CO_2 production for recycling occurs

nfldyr – number of years the CO_2 EOR operation lasts (i.e., the CO_2 EOR operation time frame)

In this algorithm, the variable *fldyrvco2recy* is an array storing the volume of CO_2 produced for recycling each year. This variable is divided by 365.0 to convert it to an equivalent volume on an average daily basis. The variable *fldyrvco2recy* stores results for each year of CO_2 EOR operations, which is why the Do Loop cycles through *nfldyr* years.

The cost of the CO_2 recycling plant is determined using the following algorithm, which depends on the maximum volume of CO_2 that is processed:

If (recymaxrate < = 30.0) Then

co2plantcap1 = recyle30fixed + recymaxrate · recyle30var

Else

co2plantcap1 = recygt30fixed + (recymaxrate - 30.0) · recygt30var

End If

co2plantcapby = co2plantcap1 · byca

Where:

co2plantcap1 – capital costs for a CO_2 recycling plant in 2011 dollars (2011$)

recyle30fixed – fixed capital costs for a CO_2 recycling plant that processes 30 MMscf/day of CO_2 or less (2011$)

recyle30var – variable capital costs for a CO_2 recycling plant that processes 30 MMscf/day of CO_2 or less (2011$/(MMscf/day))

recygt30fixed – fixed capital costs for a CO_2 recycling plant that processes more than 30 MMscf/day of CO_2 (2011$)

recygt30var – variable capital costs for a CO_2 recycling plant that processes more than 30 MMscf/day of CO_2 (2011$/(MMscf/day))

co2plantcapby – capital costs for a CO_2 recycling plant in base year dollars (BY$)

byca – base year cost adjustment factor (BY$/2011$)

The variables *recyle30fixed*, *recyle30var*, *recygt30fixed*, and *recygt30var* are user inputs.

The CO_2 recycling plant can be implemented in two phases. In the first phase, the equivalent of half the plant is installed and this portion of the plant can process up to half the maximum CO_2 that is recycled (i.e., half of the value stored in the variable *recmaxrate*). This portion of the CO_2 recycling plant is constructed the year before CO_2 begins to be produced. The year when CO_2 begins to be produced is stored in the variable *recybegyr*, which is calculated using the following algorithm:

Do ky = 1, nfldyr
If(fldyrvco2recy(ky) > 0.0) Then

```
            recybegyr = ky
            Exit Do Loop
        End If
    End Do
```

This Do Loop cycles through each year of CO_2 EOR operations and once CO_2 production is a positive value (i.e., the variable *fldyrvco2recy(ky)* is positive), the value for the year is stored in the variable *recybegyr* and cycling through the Do Loop is ended. The year stored in *recybegyr* is the year in the CO_2 EOR operations time frame. However, this is also the year before CO_2 begins production in the CO_2 EOR project time frame.

In the second phase, the rest of the CO_2 recycling plant is installed the year before the production of CO_2 for recycling exceeds more than half of its maximum value (i.e., the quantity stored in the variable *recmaxrate*). The algorithm used to determine this year (stored in the variable *recyhalfmaxyr*) is the following:

```
Do ky = 1, nfldyr
    If( fldyrvco2recy(ky) /365.0 > 0.5 · recymaxrate ) Then
        recyhalfmaxyr = ky
        Exit Do Loop
    End If
End Do
```

This Do Loop cycles through each year of CO_2 EOR operations and once CO_2 production exceeds half of the value stored in the variable *recymaxrate*, the value for the year is stored in the variable *recyhalfmaxyr* and cycling through the Do Loop is stopped. The year stored in *recyhalfmaxyr* is the year in the CO_2 EOR operations time frame. However, this is also the year before CO_2 production exceeds half its maximum value in the CO_2 EOR project time frame.

The cost of constructing the CO_2 recycling plant in CO_2 EOR project year *iy* is 0 in all years except the years stored in the variables *recybegyr* and *recyhalfmaxyr*. The algorithm is:

```
Do iy = 1, nfldeconyr
    If( iy == recybegyr or iy == recyhalfmaxyr ) Then
        fldco2pltcapcn(iy) = 0.5 · co2plantcapby · fac
    Else
        fldco2pltcapcn(iy) = 0.0
    End If
End Do
```

Where:

fldco2pltcapcn(iy) – capital costs for constructing the CO_2 recycling plant incurred in CO_2 EOR project year *iy* (MBY$)

co2plantcapby – capital costs for a CO_2 recycling plant in base year dollars (BY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

4.9.8 CO_2 Connecting Pipeline

This cost element is applicable to both brownfield and greenfield sites. The CO_2 connecting pipeline delivers CO_2 from the CO_2 delivery trunk line to the CO_2 recycling plant.

The capital cost of a CO_2 connecting pipeline depends on the diameter of the pipe carrying the CO_2 and the length of the pipeline. The equation used to calculate this cost is the following:

$$co2pipcapby = (pipfixed + co2pipvar) \cdot dtoco2$$

Where:

co2pipcapby – capital cost for CO_2 connecting pipeline in base year dollars (BY$)

pipfixed – fixed pipeline installation and equipment costs per mile, independent of pipeline diameter (BY$/mi)

co2pipvar – variable CO_2 connecting pipeline costs per mile, depending on the diameter of the pipeline (BY$/mi)

dtoco2 – distance from the CO_2 trunk line to the CO_2 EOR recycling plant (mi)

The variables *pipfixed* and *dtoco2* are user inputs.

The variable *co2pipvar* depends on the diameter of the pipeline used to transport the CO_2 and this diameter depends on the maximum annual volume of purchased CO_2. The variable *purmaxrate* stores the maximum annual volume of purchased CO_2 expressed as its equivalent average daily flow rate of purchased CO_2 in MMscf/day. The algorithm for calculating *purmaxrate* is:

```
purmaxrate = 0.0
Do ky = 1,  nfldyr
    If( ( fldyrvco2purch(ky) / 365.0 ) > purmaxrate ) Then
        purmaxrate = fldyrvco2purch(ky) / 365.0
    End If
End Do
```

Where:

purmaxrate – maximum annual rate of purchased CO_2 (MMscf/day); this is a maximum annual rate expressed as its equivalent average daily rate

fldyrvco2purch(ky) – volume of CO_2 purchased in year *ky* (MMscf/yr)

nfldyr – number of years the CO_2 EOR operation lasts (i.e., the CO_2 EOR operation time frame)

In this algorithm, the variable *fldyrvco2purch* is an array storing the volume of CO_2 purchased each year. This variable is divided by 365.0 to convert it to an equivalent volume on an average daily basis. The variable *fldyrvco2purch* stores results for each year of CO_2 EOR operations, which is why the Do Loop cycles through *nfldyr* years.

The diameter of the pipeline needed to transport the CO_2 (*co2pipsize*) and its associated variable cost (*co2pipvar*) are given by the following algorithm:

```
If( purmaxrate < 15.0 ) Then
    ! purmaxrate < 15.0 MMscf/day
    co2pipsize = 4.0
    co2pipvar = pipvar4
Else If( purmaxrate < 35.0 ) Then
    ! purmaxrate >= 15.0 MMscf/day and purmaxrate < 35.0 MMscf/day
    co2pipsize = 6.0
    co2pipvar = pipvar6
Else If( purmaxrate < 60.0 ) Then
    ! purmaxrate >= 35.0 MMscf/day and purmaxrate < 60.0 MMscf/day
    co2pipsize = 8.0
    co2pipvar = pipvar8
Else If( purmaxrate < 120.0 ) Then
    ! purmaxrate >= 60.0 MMscf/day and purmaxrate < 120.0 MMscf/day
    co2pipsize = 10.0
    co2pipvar = pipvar10
Else If( purmaxrate < 480.0 ) Then
    ! purmaxrate >= 120.0 MMscf/day and purmaxrate < 480.0 MMscf/day
    co2pipsize = 15.0
    co2pipvar = pipvar15
Else
    ! purmaxrate >= 480.0 MMscf/day
    co2pipsize = 20.0
    co2pipvar = pipvar20
End If
```

Where:

co2pipsize – diameter of the pipe (in)

co2pipvar – variable CO_2 connecting pipeline costs per mile, depending on the diameter of the pipeline (BY$/mi)

pipvar4, pipvar6, pipvar8, pipvar10, pipvar15, pipvar20 – variable CO_2 connecting pipeline cost per mile for different diameter pipelines (BY$/mi)

The variables *pipvar4, pipvar6, pipvar8, pipvar10, pipvar15, pipvar20* are user inputs.

The variable *fldco2pipcapcn(iy)* is an array that stores the cost of constructing the CO_2 connecting pipeline in CO_2 EOR project year *iy*. This pipeline is assumed to be built the first year of the CO_2 EOR project. Thus, values for *fldco2pipcapcn(iy)* are 0 except when *iy* = 1.

$$fldco2pipcapcn(1) = fac \cdot co2pipcapby$$

Where:

fldco2pipcapcn(1) – capital cost of CO_2 connecting pipeline in year 1 of the CO_2 EOR project (MBY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

co2pipcapby – capital cost for CO_2 connecting pipeline in BY$

4.9.9 Oil Connecting Pipeline

This cost element is only applicable to greenfield sites. The oil connecting pipeline carries produced oil from the oil field to a large oil trunk line, where it is transported to a refinery. These costs are included in the cash flow only if the variable *grnfldcon* = 1.

The procedure used to calculate the oil connecting pipeline is analogous to the procedure used to calculate the CO_2 connecting pipeline. The algorithms for calculating the capital costs for the oil connecting pipeline use some of the same cost variables that are used to calculate the capital costs of the CO_2 connecting pipeline.

The capital cost of the oil connecting pipeline depends on the diameter of the pipe carrying the oil and the length of the pipeline. The equation used to calculate this cost is the following:

$$oilpipcapby = (pipfixed + oilpipvar) \cdot dtooil$$

Where:

oilpipcapby – capital cost for oil connecting pipeline in BY$

pipfixed – fixed pipeline installation and equipment costs per mile, independent of pipeline diameter (BY$/mi)

oilpipvar – variable oil connecting pipeline costs per mile, depending on the diameter of the pipeline (BY$/mi)

dtooil – distance from the oil trunk line to the oil separation and storage operations (mi)

The variables *pipfixed* and *dtooil* are user inputs. The variable *pipfixed* is the same variable used for the CO_2 connecting pipeline costs.

The variable *oilpipvar* depends on the diameter of the pipeline used to transport the oil and this diameter depends on the maximum annual mass of oil produced. The variable *oilmaxrate* stores the maximum annual mass of oil produced expressed as its equivalent average daily flow rate of produced oil in tonne/day. The algorithm for calculating *oilmaxrate* is:

```
oilmaxrate1 = 0.0
Do ky = 1,  nfldyr
    If( ( fldyrvoilpr(ky) / 365.0 ) > oilmaxrate1 ) Then
        oilmaxrate1 = fldyrvoilpr(ky) / 365.0
    End If
End Do
oilmaxrate = oilmaxrate1 · 1000 STB/MSTB · denosurfbbl · 0.001 tonnes/kg
```

Where:

oilmaxrate1 – maximum annual rate of oil produced (MSTB/day); this is a maximum annual rate expressed as its equivalent average daily rate

fldyrvoilpr(ky) – volume of oil produced in year *ky* (MSTB/yr)

nfldyr – number of years the CO_2 EOR operation lasts (i.e., the CO_2 EOR operation time frame)

oilmaxrate – maximum annual mass rate of oil produced (tonnes/day); this is a maximum annual mass rate expressed as its equivalent average daily rate

denosurfbbl – density of oil (kg/STB)

In this algorithm, the variable *fldyrvoilpr* is an array storing the volume of oil produced each year. This variable is divided by 365.0 to convert it to an equivalent volume on an average daily basis. The variable *fldyrvoilpr* stores results for each year of CO_2 EOR operations, which is why the Do Loop cycles through *nfldyr* years. This volumetric rate of oil produced is converted to a mass rate using the density of oil, *denosurfbbl.*

The diameter of the pipeline needed to transport the oil (*oilpipsize*) and its associated variable cost (*oilpipvar*) are given by the following algorithm.

```
If( oilmaxrate < 790.0 ) Then
    ! oilmaxrate < 790.0 tonnes/day
    oilpipsize = 4.0
    oilpipvar = pipvar4
Else If( oilmaxrate < 1,850.0 ) Then
    ! oilmaxrate >= 790.0 tonnes/day and oilmaxrate < 1,850.0 tonnes/day
    oilpipsize = 6.0
    oilpipvar = pipvar6
Else If( oilmaxrate < 3,175.0 ) Then
    ! oilmaxrate >= 1,850.0 tonnes/day and oilmaxrate < 3,175.0 tonnes/day
    oilpipsize = 8.0
    polpipvar = pipvar8
Else If( oilmaxrate < 6,350.0 ) Then
    ! oilmaxrate >= 3,175.0 tonnes/day and oilmaxrate < 6,350.0 tonnes/day
    oilpipsize = 10.0
    oilpipvar = pipvar10
Else If( oilmaxrate < 25,400.0 ) Then
    ! oilmaxrate >= 6,350.0 tonnes/day and oilmaxrate < 25,400.0 tonnes/day
    oilpipsize = 15.0
    oilpipvar = pipvar15
Else
    ! oilmaxrate >= 25,400.0 tonnes/day
    oilpipsize = 20.0
    oilpipvar = pipvar20
End If
```

Where:

oilpipsize – diameter of the pipe (in)

oilpipvar – variable oil connecting pipeline costs per mile, depending on the diameter of the pipeline (BY$/mi)

pipvar4, pipvar6, pipvar8, pipvar10, pipvar15, pipvar20 – variable connecting pipeline cost per mile for different diameter pipelines (BY$/mi)

The variables *pipvar4, pipvar6, pipvar8, pipvar10, pipvar15, pipvar20* are user inputs. These are the same variables that are used to calculate the capital costs of the CO_2 connecting pipeline.

The variable *fldoilpipcapcn(iy)* is an array that stores the cost of constructing the oil connecting pipeline in CO_2 EOR project year *iy*. This pipeline is assumed to be built the first year of the CO_2 EOR project. Thus, values for *fldoilpipcapcn(iy)* are 0 except when *iy* = 1.

$$fldoilpipcapcn(1) = fac \cdot oilpipcapby$$

Where:

fldoilpipcapcn (1) – capital cost of oil connecting pipeline in year 1 of the CO_2 EOR project (MBY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

oilpipcapby – capital cost for oil connecting pipeline in base year dollars (BY$)

4.9.10 CO_2 and Water Distribution Systems and Produced Fluid Gathering System

There are three cost elements described in this section: the CO_2 distribution system, the water distribution system, and the produced fluids gathering system. A new CO_2 distribution system must be built at both a brownfield and greenfield site. The water distribution system and produced fluids gathering system only need to be installed at a greenfield site (i.e., when the variable *grnfldcon* = 1)

The CO_2 and water distribution systems originate at the CO_2 recycling plant. These distribution systems transport CO_2 and water to the injection wells. The gathering system transports produced fluids from the production wells to the recycling plant where the produced fluids are sent to separation tanks where oil, water, and gases (primarily CO_2 at most facilities) are separated. The oil is sent to a refinery via the oil connecting pipeline and an oil trunkline. The gas stream is processed further in the CO_2 recycling plant. Depending on the composition of the gas, separation of CO_2 from other gaseous components (such as hydrocarbon gases, nitrogen, hydrogen sulfide) may occur. The CO_2 stream is compressed and dewatered and combined with purchased CO_2, if necessary, and then sent to injection wells for injection into the oil field. All or some of the water will be reinjected if a water alternating gas flood is being conducted. Any excess water is sent to water disposal wells for subsurface disposal.

Greenfield sites will incur capital costs for the two distribution systems and the gathering system, while brownfield sites will only need to install the CO_2 distribution system. At brownfield sites, it is assumed that water distribution and fluid gathering systems are already in place.

The distribution and gathering systems are assumed to be identical in their design, construction, and materials. The capital costs for distribution and gathering systems in the model are calculated by multiplying a unit cost per well by the number of injection and production wells in the oil field that require injection and production system installation.

The capital cost for installing distribution and gathering systems in CO_2 EOR project year *iy* is given by the following equation:

$$fldgathcapcn(iy) = gathuncapby \cdot fac \cdot$$
$$(distsysmult \cdot Ninjwtotyr(iy) + gathsysmult \cdot Nprwtotyr(iy))$$

Where:

fldgathcapcn(iy) – capital costs for implementing distribution and gathering systems in CO_2 EOR project year *iy* (MBY$)

gathuncapby – unit capital cost for implementing a distribution or gathering system for one injection or production well (BY$/well)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

distsysmult – variable that determines the number of distribution systems installed at each injection well. For greenfield sites, this variable equals 2. For brownfield sites, this variable equals 1

Ninjwtotyr(iy) – the total number of injection wells implemented in the oil field in CO_2 EOR project year *iy*; this includes both new and recompleted injection wells

gathsysmult – variable that determines the number of gathering systems installed at each production well. For greenfield sites, this variable equals 1. For brownfield sites, this variable equals 0

Nprwtotyr(iy) – the total number of production wells implemented in the oil field in CO_2 EOR project year *iy*; this includes both new and recompleted production wells

The equation for the unit capital cost, *gathuncapby*, is based on the weight of the distribution or gathering pipe system, the price of steel for the pipe and a labor cost multiplier. The weight of the distribution or gathering pipe system depends on the length of the pipe and the thickness of the pipe. includes the calculated distribution and gathering system pipe length and thickness, the current steel price, and a labor multiplier:

$$gathuncapby = gathpipweight \cdot steelprice \cdot laborcm$$

$$gathpipweight = gathpiplen \cdot gathpiparea \cdot \frac{490}{2000}$$

$$gathpiplen = 1.2 \cdot Sqrt(patarea) \cdot 208.7103$$

$$gathpiparea = 3.1416 \cdot \frac{\left((pdia \cdot 0.5)^2 - \left((pdia - 2 \cdot pthick) \cdot 0.5\right)^2\right)}{144}$$

Where:

gathpipweight – weight of pipe (ton); 490 is the approximate density of steel (lbs/ft^3); 2,000 is pounds (lbs) in a ton

steelprice – price of steel (BY$/ton)

laborcm – labor cost multiplier

gathpiplen – length of pipe for distribution/gathering system (ft); 208.7103 converts square root of acres to feet; 1.2 is a tortuosity factor

gathpiparea – cross-sectional area of steel portion of pipe (ft^2); 3.1416 is pi; 144 is the number of square inches in a square foot

patarea – area of the pattern (acres)

pdia – outer diameter of pipe (in)

pthick – thickness of pipe wall (in)

The variables *steelprice*, *laborcm*, *pdia*, and *pthick* are user inputs.

4.9.11 Water Disposal Wells

This cost element is applicable to greenfield sites since water disposal wells are assumed to have already been installed at brownfield sites. These costs are included in the cash flow analysis only if the variable *grnfldcon* = 1.

Capital costs for water disposal wells in the FE/NETL Onshore CO_2 EOR Cost Model are an optional cost determined by the user. Depending on the design of the CO_2 EOR operation, and the volume of produced water from the reservoir, water disposal may not be required at the site. Produced water, to the extent possible, is recycled and used for injection. Produced water may also be transported for use at another nearby oil field. The user determines whether water disposal well costs are included in the cash flow analysis by the variable *watinjcon*. Setting this variable equal to 1 includes the costs while setting the variable equal to 0 does not include the costs.

The capital costs for water disposal wells in CO_2 EOR project year *iy* are given by the following equation:

$$fldwatinjcapcn(iy) = Nwatinjwnewyr(iy) \cdot watinjwuncapby \cdot fac$$

Where:

fldwatinjcapcn(iy) – capital costs for water injection wells in CO_2 EOR project year *iy* (MBY$)

Nwatinjnewyr(iy) – number of water injection wells installed in CO_2 EOR project year *iy*

watinjwuncapby – unit capital costs for a water disposal well (BY$/well)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The unit capital cost for water disposal wells uses the equations presented earlier for calculating the standard drilling and completion cost for a new well in the oil field plus a cost multiplier determined by the user. The cost multiplier allows the user to modify this standard well drilling and completion cost to reflect costs associated with a water disposal well. The unit capital cost is given by the following equation:

$$watinjwuncapby = dcwuncapby \cdot watinjmult$$

Where:

dcwuncapby – unit capital costs for drilling and completing a well in base year dollars (BY$/well)

watinjmult – multiplier applied to unit well drilling and completion costs to get water injection well costs

The variable *watinjmult* is a user input.

4.9.12 Pipelines for Water Disposal Wells

If the capital costs for water disposal wells are included in the analysis (i.e., this is a greenfield site and the variable *watinjcon* is set to 1), then each water disposal well is assumed to have pipeline that transports excess produced water from the produced fluids processing facilities to the water disposal well for injection. These costs are included in the cash flow analysis only if the variable *grnfldcon* = 1.

The capital costs for the water disposal pipeline in CO_2 EOR project year *iy* is given by the following equation:

$$fldwatpipcapcn(iy) = Nwatinjwnewyr(iy) \cdot watpipwuncapby \cdot fac$$

Where:

fldwatpipcapcn(iy) – capital costs for water disposal pipelines in CO_2 EOR project year *iy* (MBY$)

Nwatinjnewyr(iy) – number of water injection wells installed in CO_2 EOR project year *iy*

watpipwuncapby – unit capital costs for one water disposal pipeline to a water disposal well (BY$/well)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The equations for calculating the unit capital cost for water disposal pipelines are like the equations used to calculate the unit costs for distribution/gathering pipelines. The variables *steelprice*, *laborcm*, and *gathpiparea* used in the equations for the distribution/gathering pipelines are used in the equations for the water disposal pipelines. The unit water disposal pipeline costs are given by the following equations:

$$watpipuncapby = watpipweight \cdot steelprice \cdot laborcm$$

$$watpipweight = watpiplen1 \cdot watpiparea \cdot 490/2{,}000$$

$$watpiplen1 = watpiplen \cdot 5{,}280$$

$$watpiparea = gathpiparea$$

Where:

watpipweight – weight of pipe (ton); 490 is the approximate density of steel (lbs/ft^3); 2,000 is lbs in a ton

steelprice – price of steel (BY$/ton)

laborcm – labor cost multiplier

watpiplen1 – length of pipe for water disposal pipeline (ft); 5,280 converts miles to feet

watpiplen – length of pipe for water disposal pipeline (mi)

watpiparea – cross-sectional area of steel portion of pipe (ft^2)

gathpiparea – cross-sectional area of steel portion of pipe used in the gathering or distribution pipeline system (ft^2)

The variables *steelprice*, *laborcm*, and *watpiplen* are user inputs.

4.9.13 Monitoring Technologies for MRV Plans

This cost element is applicable to both brownfield and greenfield sites. If the operator of a CO_2 EOR facility decides to get credit for CO_2 stored during CO_2 EOR operations, then the operator must develop and implement a Monitoring, Reporting and Verification (MRV) Plan. The variable *co2moncon* controls whether these costs are included in the analysis. If the variable *co2moncon* is set to 1 by the user, then the costs of developing and implementing an MRV Plan are included. If the variable *co2moncon* equals anything else, the costs for developing and implementing an MRV Plan are not included in the analysis.

The costs for developing and implementing an MRV Plan has several components.

4.9.13.1 MRV Plan Development Capital Costs

The user specifies the capital cost for developing the MRV Plan, submitting the plan to the appropriate regulatory authority, revising the plan and obtaining approval to implement the plan. This cost is stored in the variable *co2mon_plan* (in BY$). This cost is incurred in the first year of the CO_2 EOR project.

4.9.13.2 CO_2 Monitoring Technology Capital Costs

The MRV Plan involves the implementation and operation of CO_2 monitoring technologies. The FE/NETL Onshore CO_2 EOR Cost Model allows up to 10 CO_2 monitoring technologies to be implemented. The capital cost of each CO_2 monitoring technology has two components. One component is the number of units of the technology that are implemented each year. The second component is the capital cost of each unit.

The variables storing information on each CO_2 monitoring technology are stored in arrays. The index *mt* is used to indicate the location of a particular CO_2 monitoring technology in an array. The variable *co2mon_lab(mt)* is user supplied text that names or describes the mt^{th} CO_2 monitoring technology. The variable *co2mon_stat(mt)* is a user-supplied variable that is the status of the mt^{th} CO_2 monitoring technology. If *co2mon_stat(mt)* equals 1, then the mt^{th} CO_2 monitoring technology is active. For any other value of *co2mon_stat(mt)*, the mt^{th} CO_2 monitoring technology is not active.

Number of Units Implemented Each Year for Each CO_2 Monitoring Technology

The number of units of the mt^{th} CO_2 monitoring technology that are implemented can be specified three ways.

First, the user can specify one or more unit(s) site-wide for the mt^{th} CO_2 monitoring technology with the variable *co2mon_sitewunits(mt)*. All units specified with this variable are assumed to be implemented in the first year of the CO_2 EOR project.

Second, the user can specify the implementation of units on an areal basis for the mt^{th} CO_2 monitoring technology with the variable *co2mon_areaperunit(mt)*. This variable provides the areal coverage (in acres) for each unit of the mt^{th} CO_2 monitoring technology. The number of units implemented for the mt^{th} CO_2 monitoring technology is stored in the variable *co2mon_areaunits* using the following equation.

$$co2mon\_areaunits = Ceiling(flddevarea/co2mon\_areaperunit(mt))$$

Where:

co2mon_areaunits – number of units of the mt^{th} CO_2 monitoring technology implemented based on areal coverage

flddevarea – area of the oil field that is developed for CO_2 EOR (acres)

co2mon_areaperunit(mt) – area associated with one unit of the mt^{th} CO_2 monitoring technology (acres)

In this equation, the Fortran function *Ceiling(x)* rounds the result in parentheses up to the nearest integer (e.g., the value 3.1 is rounded up to 4). All units specified with the variable *co2mon_pattunits* are assumed to be implemented in the first year of the CO_2 EOR project. The variable *co2mon_areaperunit(mt)* is a user input.

Third, the user can specify the implementation of units on a pattern basis for the mt^{th} CO_2 monitoring technology with the variable c*o2mon_unitsperpatt(mt)*. This variable provides the number of units implemented for each pattern. The number of units implemented for the mt^{th} CO_2 monitoring technology is stored in the variable *co2mon_pattunits* using the following equation.

$$co2mon\_pattunits = Ceiling(Npatteayr(iy) \cdot co2mon\_unitsperpatt(mt))$$

Where:

co2mon_pattunits – number of units of the mt^{th} CO_2 monitoring technology implemented in CO_2 EOR project year *iy* based on the number of patterns implemented in CO_2 EOR project year *iy*

Npatteayr(iy) – number of patterns implemented in CO_2 EOR project year *iy*

co2mon_unitsperpatt(mt) – number of units of the mt^{th} CO_2 monitoring technology implemented per pattern

The number of units of the mt^{th} CO_2 monitoring technology that are implemented in CO_2 EOR project year *iy* of the CO_2 project is stored in the array variable *co2mon_unitsinstall(mt, iy)*. This variable is specified with the following algorithm, which loops through all the possible CO_2 monitoring technologies and all CO_2 EOR project years.

```
Do mt = 1, maxco2mon
    ! Is this CO2 monitoring technology active?
    If( co2mon_stat(mt) == 1 ) Then
        ! Technology is active
        ! Calculate number of units installed each year iy
        Do iy = 1, nfldeconyr
            If( iy == 1 ) Then
                If( co2mon_areaperunit(mt) > 0.0 ) Then
                    co2mon_areaunits = Ceiling( flddevarea /
                        co2mon_areaperunit(mt) )
                Else
                    co2mon_areaunits = 0
                End If
                co2mon_pattunits = Ceiling( Npateayr(iy) ·
                    co2mon_unitsperpatt(mt) )
                co2mon_unitsinstall(mt, iy) = co2mon_sitewunits(mt) +
                    co2mon_areaunits + co2mon_pattunits
            Else If( iy <= flddevyr ) Then
                co2mon_pattunits = Ceiling( NPateayr(iy) ·
                    co2mon_unitsperpatt(mt) )
                co2mon_unitsinstall(mt, iy) = co2mon_pattunits
            Else
                co2mon_unitsinstall(mt, iy) = 0
            End If
        End Do ! iy loop
    End If
End Do ! mt loop
```

Where:

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_stat(mt) – status of the mt^{th} CO_2 monitoring technology. *co2mon_stat(mt)* equals 1 if technology is active, otherwise technology is inactive

nfldeconyr – number of years for the CO_2 EOR project

co2mon_sitewunits(mt) – number of units implemented site-wide for the mt^{th} CO_2 monitoring technology

co2mon_areaunits – number of units of the mt^{th} CO_2 monitoring technology implemented based on areal coverage

flddevarea – area of the oil field that is developed for CO_2 EOR (acres)

co2mon_areaperunit(mt) – area associated with one unit of the mt^{th} CO_2 monitoring technology (acres)

co2mon_pattunits – number of units of the mt^{th} CO_2 monitoring technology implemented in CO_2 EOR project year *iy* based on the number of patterns implemented in CO_2 EOR project year *iy*

Npatteayr(iy) – number of patterns implemented in year *iy*

co2mon_unitsperpatt(mt) – number of units of the mt^{th} CO_2 monitoring technology implemented per pattern

co2mon_unitsinstall(mt, iy) – number of units of the mt^{th} CO_2 monitoring technology that are implemented in CO_2 EOR project year *iy* of the CO_2 EOR project

Number of Units Active Each Year for Each CO_2 Monitoring Technology

The number of units of the mt^{th} CO_2 monitoring technology that are active in CO_2 EOR project year *iy* is stored in the variable *co2mon_unitsactive(mt, iy)*. Each unit becomes active the year after it is installed. The variable *co2mon_unitsactive(mt, iy)* is specified with the following algorithm, which loops through all the possible CO_2 monitoring technologies and all CO_2 EOR project years:

```
Do mt = 1, maxco2mon
    ! Is this CO2 monitoring technology active?
    If( co2mon_stat(mt) == 1 ) Then
        ! Technology is active
        ! Calculate number of units installed each year iy
        Do iy = 1, nfldeconyr
            If( iy == 1 ) Then
                co2mon_unitsactive(mt, iy) = 0
            Else If( iy > 1 and iy <= (flddevyr + 1) ) Then
                nunits = 0
                Do iyk = 1, iy – 1
                    nunits = nunits + co2mon_unitsinstall(mt, iyk)
                End Do ! iyk loop
```

```
                    co2mon_unitsactive(mt, iy) = nunits
                Else
                    nunits = 0
                    Do iyk = 1, flddevyr
                        nunits = nunits + co2mon_unitsinstall(mt, iyk)
                    End Do ! iyk loop
                    co2mon_unitsactive(mt, iy) = nunits
                End If
            End Do ! iy loop
        End If
    End Do ! mt loop
```

Where:

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_stat(mt) – status of the mt^{th} CO_2 monitoring technology. *co2mon_stat(mt)* equals 1 if technology is active, otherwise technology is inactive

nfldeconyr – number of years for the CO_2 EOR project

co2mon_unitsactive(mt, iy) – number of units active for the mt^{th} CO_2 monitoring technology in CO_2 project year *iy*

nunits – cumulative number of units of the mt^{th} CO_2 monitoring technology that have been implemented by CO_2 project year *iy*

co2mon_unitsinstall(mt, iy) – number of units of the mt^{th} CO_2 monitoring technology that are implemented in CO_2 EOR project year *iy* of the CO_2 EOR project

Once a CO_2 EOR monitoring technology has been implemented, it is assumed to be active for at least the duration of the CO_2 EOR project. At some CO_2 EOR projects, the government authority overseeing the implementation of the MRV Plan may require CO_2 monitoring after the CO_2 EOR project ends (i.e., after fluid production ends).

Unit Capital Costs for Each CO_2 Monitoring Technology

The unit capital costs for each CO_2 monitoring technology are specified by the user. These capital costs can be a function of a length associated with the CO_2 monitoring technology or they can be fixed costs. The variables *co2mon_lenstat(mt)* and *co2mon_lenvar(mt)* are used to define how the length and length-related capital costs are calculated:

- If the variable *co2mon_lenstat(mt)* equals 1, then the CO_2 monitoring technology is assumed to be a deep monitoring well and the length is the distance to the reservoir (the variable *depth*) as modified by the variable *co2mon_lenvar(mt)*.
- If the variable *co2mon_lenstat(mt)* equals 2, then the CO_2 monitoring technology is assumed to be associated with a deep well and the length is the distance to the reservoir (the variable *depth*) as modified by the variable *co2mon_lenvar(mt)*. For

example, the CO_2 monitoring technology could be a fiber optic monitoring technology deployed in a deep well.

- If the variable *co2mon_lenstat(mt)* equals anything other than 1 or 2, then the length variable associated with the CO_2 monitoring technology is given by the variable *co2mon_lenvar(mt)*.

The length associated with the mt^{th} CO_2 monitoring technology is stored in the variable *co2mon_len(mt)* and the unit capital cost is stored in the variable *co2mon_cap(mt)*. The algorithm used to calculate these two variables is as follows:

```
Do mt = 1, maxco2mon
    ! Is this CO2 monitoring technology active?
    If( co2mon_stat(mt) == 1 ) Then
        ! Technology is active
        ! Calculate unit capital costs
        If( co2mon_lenstat(mt) == 1 ) Then
            !  Deep monitoring well
            length1 = depth + co2mon_lenvar(mt)
            co2mon_len(mt) = length1
            dcwuncap_cm = dcb0(rc) - dcb1(rc) · length1 + dcb2(rc) · length1²
            dcwca_cm = (oilprice + 90.0) / (90.0 · 2.0)
            dcwuncap_cm_by = dcwuncap_cm · dcwca_cm
            co2mon_cap(mt) = dcwuncap_cm_by + length1 ·
                (co2mon_lencap1(mt) + co2mon_lencap2(mt)) +
                co2mon_fixcap1(mt) + co2mon_fixcap2(mt)
        Else If( co2mon_lenstat(mt) == 2 ) Then
            ! CO2 monitoring technology with costs that depend
            ! on distance to reservoir
            length1 = depth + co2mon_lenvar(mt)
            co2mon_len(mt) = length1
             co2mon_cap(mt) = length1 · ( co2mon_lencap1(mt) +
                co2mon_lencap2(mt) ) + co2mon_fixcap1(mt) +
                co2mon_fixcap2(mt)
        Else
            ! CO2 monitoring technology with costs that do not depend
            ! on distance to the reservoir
            length1 = co2mon_lenvar(mt)
            co2mon_len(mt) = length1
            co2mon_cap(mt) = length1 · ( co2mon_lencap1(mt) +
                co2mon_lencap2(mt) ) + co2mon_fixcap1(mt) +
                co2mon_fixcap2(mt)
        End If
    End If
End Do ! mt loop
```

Where:

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_stat(mt) – status of the mt^{th} CO_2 monitoring technology. *co2mon_stat(mt)* equals 1 if technology is active, otherwise technology is inactive

co2mon_lenstat(mt) – variable controlling how the length associated with the mt^{th} CO_2 monitoring technology is calculated and how length dependent capital costs are calculated

length1 – length associated with mt^{th} CO_2 monitoring technology (ft)

depth – distance from surface to reservoir (ft)

co2mon_lenvar(mt) – length variable associated with mt^{th} CO_2 monitoring technology (ft). This variable can be negative so when it is added to the variable *depth* to calculate the variable *length1*, the variable *length1* is less than *depth*. For a deep monitoring well that is completed above the reservoir, this variable should be negative

co2mon_len(mt) – length associated with mt^{th} CO_2 monitoring technology (ft)

co2mon_cap(mt) – unit capital costs for mt^{th} CO_2 monitoring technology (BY$/unit)

dcwuncap_cm – unit cost of drilling a deep monitoring well based on data in the API Joint Association survey of 2014 drilling costs (2014$/unit)

dcwca_cm – oil-price based cost adjustment factor that converts drilling costs in 2014 to drilling costs in the base year (BY$/2014$)

dcb0(rc), dcb1(rc), dcb2(rc) – coefficients in equation for well drilling and completion cost as a function of depth from the API Joint Association well drilling costs. These variables are arrays and the index *rc* points to the location in each array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

co2mon_lencap1(mt), co2mon_lencap2(mt) – length dependent capital costs for the mt^{th} CO_2 monitoring technology (BY$/ft-unit)

co2mon_fixcap1(mt), co2mon_fixcap2(mt) – fixed capital costs for the mt^{th} CO_2 monitoring technology (BY$/unit)

The variables *co2mon_stat(mt)*, *co2mon_lenstat(mt)*, *co2mon_lenvar(mt)*, *co2mon_lencap1(mt), co2mon_lencap2(mt)*, *co2mon_fixcap1(mt)*, and *co2mon_fixcap2(mt)* are user inputs. The procedure for calculating the capital costs of drilling and completing deep wells was discussed in more detail in an earlier section.

4.9.13.3 Capital Cost Cash Flows for CO_2 Monitoring Technologies

The capital costs for implementing CO_2 monitoring technologies in each year *iy* is stored in the array *fldco2moncapcn(iy)*. The following algorithm is used to assign costs to each item in this array:

```
Do iy = 1, nfldeconyr
    ! For year iy, add all the capital costs for all CO2 monitoring technologies
    co2mon_temp = 0.0
    Do mt = 1, maxco2mon
        co2mon_temp = co2mon_temp + co2mon_unitsinstall(mt, iy) ·
            co2mon_cap(mt)
    End Do ! mt loop
    If( iy == 1 ) Then
        ! If this is first year, add cost of developing MRV Plan
        co2mon_temp = co2mon_temp + co2mon_plan
    End If
    fldco2moncapcn(iy) = co2mon_temp · fac
End Do ! iy loop
```

Where:

nfldeconyr – number of years for the CO_2 EOR project

co2mon_temp – variable storing total capital costs incurred in CO_2 EOR project year *iy* for implementing CO_2 monitoring technologies (BY$)

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_unitsinstall(mt, iy) – number of units of the mt^{th} CO_2 monitoring technology that are implemented in CO_2 EOR project year *iy* of the CO_2 EOR project

co2mon_cap(mt) – unit capital costs for mt^{th} CO_2 monitoring technology (BY$/unit)

co2mon_plan – capital costs for developing, submitting, and finalizing the MRV Plan (BY$)

fldco2moncapcn (iy) – capital costs for implementing CO_2 monitoring technologies in CO_2 EOR project year *iy* (MBY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

4.9.14 Process Contingency Capital Costs

This cost element is applicable to both brownfield and greenfield sites. Process contingencies are included in cost estimates to account for unknown costs that are omitted or unforeseen due to a lack of complete project definition and engineering. Contingencies are added because experience has shown that such costs are likely, and expected, to be incurred even though they cannot be explicitly determined at the time the cost estimate is prepared.

Process contingency is intended to compensate for uncertainty in cost estimates caused by performance uncertainties associated with the development status of a technology. Process contingencies are applied to each technology based on its development status. The array *fldprocconcapcn(iy)* provides the process contingency for capital costs in CO_2 EOR operation year *iy*. Since CO_2 EOR is a mature technology, a process contingency has not been applied to any of the cost elements in the FE/NETL Onshore CO_2 EOR Cost Model. Therefore, each element in the array *fldprocconcapcn(iy)* is set to zero.

4.9.15 Design and General & Administrative Capital Costs

This cost element is applicable to both brownfield and greenfield sites. The design and general and administrative (G&A) capital costs are included in the FE/NETL Onshore CO_2 EOR Cost Model as a single cost. The design costs are the costs for developing the design of the CO_2 EOR project from initial conception to detailed design. The G&A capital costs include in-house accounting, human resources, legal, administrative, technical, and operational costs that are essential for installing the equipment and systems necessary for a successful CO_2 EOR project. The FE/NETL Onshore CO_2 EOR Cost Model calculates the design and G&A costs by calculating a partial sum of all capital costs and multiplying this sum by the variable *capganda*, which is input by the user.

The total design and G&A cost calculation in CO_2 EOR project year *iy* is stored in the array *fldgaacapcn(iy).* The values in this array are calculated using the following equation:

$$fldgaacapcn(iy) = fldtot2capcn(iy) \cdot capganda$$

Where:

fldgaacapcn(iy) – design and G&A costs that are part of capital costs (MBY$)

fldtot2capcn(iy) – partial sum of capital costs in CO_2 EOR project year *iy* (MBY$)

capganda – fraction of partial sum of capital costs that represent the additional cost of design and G&A

The variable *capganda* is a user input.

The variable *fldtot2capcn(iy)* is the sum of all the capital costs described in previous sections. The sum is a partial total of all capital costs since this sum does not include the cost of design, G&A, and project contingency. The variable *fldtot2capcn(iy)* is calculated with the following equation:

$$\begin{aligned} fldtot2capcn(iy) &= fldacqpatcapcn(iy) + fldseiscapcn(iy) + fldbldgroadcapcn(iy) \\ &+ flddcwcapcn(iy) + fldrecwcapcn(iy) + fldseprwcapcn(iy) \\ &+ fldseinwcapcn(iy) + fldco2pltcapcn(iy) + fldco2pipcapcn(iy) \\ &+ fldoilpipcapcn(iy) + fldgathcapcn(iy) + fldwatinjcapcn(iy) \\ &+ fldwatpipcapcn(iy) + fldco2moncapcn(iy) + fldprocconcapcn(iy) \end{aligned}$$

Where:

fldtot2capcn(iy) – partial sum of capital costs in CO_2 EOR project year *iy* (MBY$)

fldacqpatcapcn(iy) – capital costs for acquiring leases, mineral rights, and obtaining some permits in CO_2 EOR project year *iy* (MBY$)

fldseiscapcn(iy) – capital costs for seismic imaging for characterization in CO_2 EOR project year *iy* (MBY$)

fldbldgroadcapcn(iy) – capital costs for constructing office buildings and access roads in CO_2 EOR project year *iy* (MBY$)

flddcwcapcn(iy) – capital costs for drilling and completing new wells in CO_2 EOR project year *iy* (MBY$)

fldrecwcapcn(iy) – capital costs for recompleting old wells in CO_2 EOR project year *iy* (MBY$)

fldseprwcapcn(iy) – capital costs for surface equipment associated with new production wells in CO_2 EOR project year *iy* (MBY$)

fldseinwcapcn(iy) – capital costs for surface equipment associated with new injection wells in CO_2 EOR project year *iy* (MBY$)

fldco2pltcapcn(iy) – capital costs for CO_2 recycling plant in CO_2 EOR project year *iy* (MBY$)

fldco2pipcapcn(iy) – capital costs for CO_2 delivery pipeline in CO_2 EOR project year *iy* (MBY$)

fldoilpipcapcn(iy) – capital costs for oil delivery pipeline in CO_2 EOR project year *iy* (MBY$)

fldgathcapcn(iy) – capital costs for implementing CO_2 and water distribution systems and produced fluid gathering systems in CO_2 EOR project year *iy* (MBY$)

fldwatinjcapcn(iy) – capital costs for water disposal wells in CO_2 EOR project year *iy* (MBY$)

fldwatpipcapcn(iy) – capital costs for pipelines carrying excess water to water disposal wells in CO_2 EOR project year *iy* (MBY$)

fldco2moncapcn(iy) – capital costs for implementing CO_2 monitoring technologies in CO_2 EOR project year *iy* (MBY$)

fldprocconcapcn(iy) – process contingency capital costs in CO_2 EOR project year *iy* (MBY$)

4.9.16 Project Contingency Capital Costs

This cost element is applicable to both brownfield and greenfield sites. Project contingencies are included in cost estimates to account for unknown costs that are omitted or unforeseen due to a lack of complete project definition and engineering. Contingencies are added because experience has shown that such costs are likely, and expected, to be incurred even though they cannot be explicitly determined at the time the cost estimate is prepared.

The project contingency is calculated based on the level of detail of the capital cost estimates and the data used to estimate each capital cost element. The more detailed the design and the better the data used in the cost estimate, the lower the project contingency cost. The project contingency cost in CO_2 EOR project year *iy* is stored in the array *fldgaacapcn(iy).* The values in this array are calculated using the following equation:

$$fldprojconcapcn(iy) = fldtot3capcn(iy) \cdot ProjContFac$$

Where:

fldprojconcapcn(iy) – project contingency cost in CO_2 EOR project year *iy* (MBY$)

fldtot3capcn(iy) – partial sum of all project capital costs in CO_2 EOR project year *iy* (MBY$). This variable includes all capital costs except project contingency costs

ProjContFac – project contingency factor

The variable *ProjContFac* is a user input.

The variable *fldtot3capcn(iy)* stores the sum of all capital costs in CO_2 EOR project year *iy* except project contingency capital costs. This variable is calculated using the following equation:

$$fldtot3capcn(iy) = fldtot2capcn(iy) + fldgaacapcn(iy)$$

Where:

fldtot3capcn(iy) – partial sum of all project capital costs in CO_2 EOR project year *iy* (MBY$). This variable includes all capital costs except project contingency costs

fldtot2capcn(iy) – partial sum of project capital costs in CO_2 EOR project year *iy* (MBY$). This variable includes all capital costs except design, G&A, and project contingency costs

fldgaacapcn(iy) – design and G&A capital costs in CO_2 EOR project year *iy* (MBY$)

4.9.17 Total Capital Costs

Total capital costs for the CO_2 EOR project include capital costs for implementing all the physical items described previously (such as constructing roads and offices, drilling and completing new wells, recompleting old wells, and installing a CO_2 recycling plant), process contingency costs, design and G&A costs, and project contingency costs.

Total capital costs in CO_2 EOR project year *iy* are stored in the variable *fldtotcapcn(iy)* and calculated using the following equation:

$$fldtotcapcn(iy) = fldtot3capcn(iy) + fldprojconcapcn(iy)$$

Where:

fldtotcapcn(iy) – sum of all project capital costs in CO_2 EOR project year *iy* (MBY$)

fldtot3capcn(iy) – partial sum of all project capital costs in year *iy* (MBY$). This variable includes all capital costs except project contingency costs

fldprojconcapcn(iy) – project contingency cost in CO_2 EOR project year *iy* (MBY$)

4.10 Operation and Maintenance Costs

This section presents the equations used to calculate O&M costs in the FE/NETL Onshore CO_2 EOR Cost Model. All O&M costs are incurred at brownfield and greenfield CO_2 EOR operations except for plugging old wells.

4.10.1 Well Plugging Costs

For brownfield sites, there may be existing wells that are too old to be recompleted in a cost-effective manner. Any well that is no longer used must be plugged and abandoned. Because a plugged well has no future value to the CO_2 EOR site operator, well plugging costs are treated as expenses and not capital costs.

The cost for plugging existing wells in CO_2 EOR project year *iy* is given by the following equation:

$$fldplugwexpcn(iy) = plugwunexpby \cdot Nplugwyr(iy) \cdot fac$$

Where:

fldplugwexpcn(iy) – cost for plugging wells in CO_2 EOR project year *iy* (MBY$)

plugwunexpby – unit cost for plugging old wells in base year dollars (BY$/well)

Nplugwyr(iy) – number of plugged wells in CO_2 EOR project year *iy*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The variable *plugwunexpby* is determined using the region-specific calculation:

$$plugwunexpby = plugcost(rc) \cdot byca$$

Where:

plugcost(rc) – region-specific unit cost for plugging a well (2011$/well). The variable *plugcost* is an array and the index *rc* points to the location in this array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

byca – base year cost adjustment factor (BY$/2011$). This is a user-defined variable

The variables *plugcost(rc)* and *byca* are user inputs.

4.10.2 Cost of Purchased CO_2

Purchased CO_2 cost is calculated using the total amount of CO_2 purchased in CO_2 EOR project year *iy* multiplied by the CO_2 purchase price. Purchased CO_2 volume is determined by the total amount of CO_2 injected less the volume of recycled CO_2 available for injection. Purchased CO_2 amounts are relatively higher in the earlier years of the CO_2 flood before recycled CO_2 becomes

available. Purchased CO_2 generally decreases over the life of the CO_2 flood as recycled CO_2 increases.

The purchased CO_2 cost in CO_2 EOR project year *iy* of the CO_2 EOR project is given by the following equation:

$$fldco2puropcn(iy) = co2pricecost \cdot fldyrvco2purch(iy - 1)$$

Where:

fldco2puropcn(iy) – costs for purchasing CO_2 when CO_2 is a cost to the CO_2 EOR operator in CO_2 EOR project year *iy* (MBY$)

co2pricecost – price of CO_2 if CO_2 is a cost to the CO_2 EOR operator (BY$/Mscf or MBY$/MMscf)

fldyrvco2purch(iy - 1) – volume of purchased CO_2 in CO_2 EOR operation year *iy - 1* (MMscf)

The CO_2 EOR facility does not become operational until the second year of the project, so no CO_2 is purchased in the first year of the CO_2 EOR project (i.e., *fldco2puropcn(1)* = 0).

The price of CO_2 is determined by the user based on user inputs for four variables, *co2price*, *co2prmult*, *co2prmin*, and *co2prmax*.

- If *co2price* is positive (specifically, greater than $1.0x10^{-6}$), then the value of *co2price* is assigned to the variable *co2pricecost*.
- If *co2price* is 0 (more specifically, if *co2price* is between $-1.0x10^{-6}$ and $1.0x10^{-6}$), then the value for *co2pricecost* is determined by multiplying the market price of oil by the variable *co2prmult*. In this case, the value for *co2pricecost* is constrained to be between a minimum CO_2 price (*co2prmin*) and a maximum CO_2 price (*co2prmax*). The minimum and maximum prices are set by the user. A useful reference value for the minimum cost is the cost of capturing CO_2 from an ethanol plant, since these plants produce an almost pure stream of CO_2 that only requires compression and dehydration. A useful reference value for the maximum cost is the cost of capturing CO_2 from the air, which is likely to be the most expensive technology for capturing CO_2.
- If *co2price* is less than $-1.0x10^{-6}$ then it is assumed that CO_2 is a source of revenue (i.e., the CO_2 EOR operator is paid to accept CO_2 for use in CO_2 EOR) and not a cost. In this case, the variable *co2pricecost* is set to zero. The variable *co2pricerev* is used to store the price of CO_2 when CO_2 is a source of revenue as discussed in an earlier section.

The algorithm for determining the variables *co2pricecost* and *co2pricerev* is as follows:

If(co2price > $1.0x10^{-6}$) Then
    co2pricecost = co2price
    co2pricerev = 0.0
Else If ((co2price >= $-1.0x10^{-6}$) and (co2price <= $1.0x10^{-6}$)) Then
    co2pricecost = oilprice · co2prmult
    If(co2pricecost < co2prmin) Then

```
            co2pricecost = co2prmin
        End If
        If( co2pricecost > co2prmax ) Then
            co2pricecost = co2prmax
        End If
        co2pricerev = 0.0
    Else
        co2pricecost = 0.0
        co2pricerev = - co2price
    End If
```

Where:

co2price – price of CO_2 input by user (BY$/Mscf)

co2pricecost – price of CO_2 if CO_2 is a cost to the CO_2 EOR operator (BY$/Mscf)

co2pricerev – price of CO_2 if CO_2 is a source of revenue for the CO_2 EOR operator (BY$/Mscf)

oilprice – market price of oil (BY$/STB)

co2prmult – multiplier input by user that is used to calculate the price of CO_2 from the market price of oil (STB/Mscf)

co2prmin – minimum price of CO_2 input by user (BY$/Mscf)

co2prmax – maximum price of CO_2 input by user (BY$/Mscf)

The variables *co2price, oilprice, co2prmult, co2prmin,* and *co2prmax* are user inputs.

4.10.3 Well and Lease O&M Costs

Well and lease O&M costs are calculated for each pattern. These are comprehensive O&M costs for a water flood and include the costs of operating and maintaining the following items:

- injection wells
- surface equipment for injection wells
- production wells
- surface equipment for production wells
- water distribution system
- produced fluid gathering system
- office buildings and access roads
- water disposal wells
- pipelines carrying excess produced water to water disposal wells

O&M costs not included in the well and lease O&M costs are the cost of lifting produced fluid, the cost of operating the CO_2 recycling plant, the cost of operating the CO_2 connecting pipeline and the cost of operating the CO_2 distribution system. The cost of operating and maintaining the oil connecting pipeline is also not included in the well and lease O&M costs.

The well and lease O&M costs are region-specific and are calculated using a fixed cost plus a variable cost depending on well depth. The FE/NETL Onshore CO_2 EOR Cost Model calculates well operating cost on a per-pattern basis. The well and lease O&M cost in CO_2 EOR project year *iy* is given by the following equation:

$$fldwellopcn(iy) = wellunopby \cdot Npatacyr(iy-1) \cdot fac$$

Where:

fldwellopcn(iy) – well and lease O&M cost in CO_2 EOR project year *iy* (MBY$)

wellunopby – annual unit well and lease O&M cost (BY$/pattern/yr)

Npatacyr(iy-1) – number of active patterns in CO_2 EOR operational year *iy-1*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldwellopcn(1)* = 0).

The annual unit well and lease O&M cost is a region-specific calculation that includes a fixed annual operating cost plus a variable operating cost depending on well depth. The annual unit well and lease O&M cost is given by solving the following equations:

$$wellunopby = (wocb0(rc) + wocb1(rc) \cdot depth1) \cdot byca$$

$$depth1 = depth + grosspay$$

Where:

wocb0(rc), wocb1(rc) – coefficients in equation for well and lease O&M costs as a function of depth. These variables generate well and lease O&M costs in 2011$. These variables are arrays and the index *rc* points to the location in the arrays that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

depth1 – depth from the surface to the bottom of the well (ft)

byca – base year cost adjustment factor (BY$/2011$). This is a user-defined variable

depth – depth from the surface to the reservoir (ft)

grosspay – gross pay (ft)

The variables *wocb0(rc), wocb1(rc),* and *byca* are user inputs. The variables *depth* and *grosspay* are properties of the oil field extracted from the StrmtbFlow output file "ystrmtbflow_outyr.csv".

4.10.4 Fluid Lifting O&M Costs

Fluid lifting costs reflect the cost of the energy needed to power the pumps that bring produced fluids to the surface at the production wells. Fluid lifting costs include a fixed cost component and a variable cost component that depends on the market price of oil. The equation includes region-specific coefficients. The lifting cost is dependent on the volumes of oil and water produced.

The fluid lifting cost in year CO_2 EOR project year *iy* is given by the equation:

$$fldliftopcn(iy) = liftunopby \cdot (fldyroilpr(iy-1) + fldyrwatpr(iy-1))$$

Where:

fldliftopcn(iy) – cost for fluid lifting in CO_2 EOR project year *iy* (MBY$)

liftunopby – unit operating cost for lifting produced water and oil (BY$/STB or MBY$/MSTB)

fldyroilpr(iy-1) – total volume of oil produced in operational year *iy-1* (MSTB)

fldwatpr(iy-1) – total volume of water produced in operational *iy-1* (MSTB)

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldliftopcn(1)* = 0).

The unit O&M cost for fluid lifting reflects the fixed and variable cost of fluid lifting on a per-barrel basis. The equation for the fluid lifting unit cost includes a fixed cost plus a variable cost that depends on the market price of oil. The variable cost was developed relative to an oil price of $50/STB. The fixed and variable cost components are region-specific. The unit fluid lifting cost is given by the following equation:

$$liftunopby = fluidliftf(rc) + ((\frac{oilprice}{50}) \cdot fluidliftv(rc))$$

Where:

fluidliftf(rc) – fixed cost of fluid lifting per barrel (BY$/STB). This variable is an array and the index *rc* points to the location in the array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

oilprice – market price of oil (BY$/STB)

fluidliftv(rc) – variable cost for fluid lifting that depends on price of oil (BY$/STB). This variable is an array and the index *rc* points to the location in the array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

The variables *fluidliftf(rc)*, *oilprice*, and *fluidliftv(rc)* are user inputs.

4.10.5 CO_2 Recycling Plant O&M Costs

The CO_2 recycling plant O&M costs include the costs for separating the CO_2 from other fluids, compressing the CO_2 and dehydrating the CO_2 in preparation for combining this CO_2 with

purchased CO_2 for eventual injection into the oil field. The cost is calculated using the total volume of CO_2 recycled in operational year *iy - 1* and a CO_2 recycling cost factor ($/Mscf) based on the market price of oil.

The CO_2 recycling plant operating cost in CO_2 EOR project year *iy* is given by the following equation:

$$fldco2pltopcn(iy) = recyco2 \cdot fldyrvco2recy(iy - 1)$$

Where:

fldco2pltopcn(iy) – cost of operating the CO_2 recycling plant in CO_2 project year *iy* (MBY$)

recyco2 – unit costs to operate and maintain the CO_2 recycling plant (BY$/Mscf or MBY$/MMscf)

fldyrvco2recy(iy-1) – total volume of recycled CO_2 in CO_2 EOR operating year *iy-1* (MMscf)

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldco2pltopcn(1)* = 0).

The variable *recyco2* is input by user. If this variable is less than or equal to $1.0x10^{-6}$, then *recyco2* is calculated based on the market oil price. The algorithm for calculating *recyco2* is the following:

If(recyco2 > $1.0x10^{-6}$) Then
! use the value for recyco2 input by the user
Else
recyco2 = recyco2opmult · oilprice
End If

Where:

recyco2opmult – variable used to convert the market price of oil into the unit O&M cost for the CO_2 recycling plant (STB/Mscf)

oilprice – market price of oil (BY$/STB)

Values for the variables *recyco2* and *recyco2opmult* are input by the user.

4.10.6 CO_2 Connecting Pipeline O&M Costs

The annual O&M cost of the CO_2 delivery pipeline is calculated as a fraction of the capital cost for installing this pipeline. The CO_2 connecting pipeline O&M cost in CO_2 EOR project year *iy* is given by the following equation:

$$fldco2pipopcn(iy) = co2pipopby \cdot fac$$

Where:

fldco2pipopcn(iy) – cost of operating the CO_2 connecting pipeline in CO_2 EOR project year *iy* (MBY$)

co2pipopby – total annual O&M costs for the CO_2 connecting pipeline (BY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldco2pipopcn(1)* = 0).

The unit O&M cost for the CO_2 connecting pipeline is given by the following equation:

$$co2pipopby = co2pipcapby \cdot pipeoandm(rc)$$

Where:

co2pipcapby – total capital cost of installing CO_2 connecting pipeline in base year dollars (BY$)

pipeoandm(rc) – region-specific fraction of total pipeline capital cost that represents annual pipeline O&M cost (BY$/BY$). This variable is an array and the index *rc* points to the location in the array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

Values for the array variable *pipeoandm(rc)* are input by the user. This variable is also used to calculate the O&M costs for CO_2 distribution system.

4.10.7 CO_2 Distribution System O&M Costs

The annual O&M cost of the CO_2 distribution system is calculated as a fraction of the capital cost for installing a CO_2 distribution pipeline system for one injection well in the distribution system. The O&M costs for the water distribution system and the fluid gathering system are included in the well and lease O&M costs. The CO_2 distribution system O&M cost in CO_2 EOR project year *iy* is given by the following equation:

$$fldgathopcn(iy) = gathunopby \cdot Ninjwacyr(iy-1) \cdot fac$$

Where:

fldgathopcn(iy) – cost of operating the CO_2 distribution system in CO_2 EOR project year *iy* (MBY$)

gathunopby – unit O&M cost for CO_2 distribution pipeline for one injection well (BY$/well)

Ninjwacyr(iy-1) – number of active injection wells in CO_2 EOR operating year *iy-1*

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldgathopcn(1)* = 0).

The unit O&M cost for the CO_2 distribution system is given by the following equation:

$$gathunopby = gathuncapby \cdot pipeoandm(rc)$$

Where:

gathuncapby – unit capital cost for installing a CO_2 distribution pipeline for one injection well (BY$/well)

pipeoandm(rc) – region-specific fraction of total pipeline capital cost that represents annual pipeline O&M cost (BY$/BY$). This variable is an array and the index *rc* points to the location in the array that stores data for the region where the oil field is located. The regions are defined in Exhibit 4-3

Values for the array variable *pipeoandm(rc)* are input by the user.

4.10.8 O&M Costs for Monitoring Technologies for MRV Plans

The annual O&M costs for CO_2 monitoring technologies implemented as part of an MRV Plan consist of the annual O&M costs for each implemented CO_2 monitoring technology and an annual cost for reporting the results of the monitoring to the appropriate regulatory authority. For each CO_2 monitoring technology, the O&M cost is the O&M cost for a unit multiplied by the number of active units. In addition to these routine annual O&M costs, there can be additional costs when the CO_2 monitoring technology is first implemented because many CO_2 monitoring technologies require background or baseline sampling before sampling during CO_2 EOR operations begins. Also, the regulatory authority that approves the MRV Plan may require the CO_2 monitoring technologies to operate for a number of years after the CO_2 EOR operations end.

4.10.8.1 Unit O&M Costs for Each CO_2 Monitoring Technology

The unit annual O&M costs for each CO_2 monitoring technology are specified by the user. These unit annual O&M costs can be a function of the capital cost for a unit, a length associated with the CO_2 monitoring technology or they can be fixed costs.

The unit annual O&M cost associated with the mt^{th} CO_2 monitoring technology is stored in the variable *co2mon_om(mt)*. The algorithm used to calculate these annual unit O&M costs is as follows.

```
Do mt = 1, maxco2mon
    ! Is this CO2 monitoring technology active?
    If( co2mon_stat(mt) == 1 ) Then
        ! Technology is active
        ! Calculate unit annual O&M costs
        length1 = co2mon_len(mt)
        co2mon_om(mt) = co2mon_cap(mt) · co2mon_capmultom(mt) + length1 ·
            (co2mon_lenom1(mt) + co2mon_lenom2(mt)) + co2mon_fixom1(mt)
            + co2mon_fixom2(mt)
    End If
End Do ! mt loop
```

Where:

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_stat(mt) – status of the mt^{th} CO_2 monitoring technology *co2mon_stat(mt)* equals 1 if technology is active, otherwise technology is inactive

co2mon_len(mt) – length associated with mt^{th} CO_2 monitoring technology (ft)

length1 – length associated with mt^{th} CO_2 monitoring technology (ft)

co2mon_om(mt) – unit annual O&M costs for mt^{th} CO_2 monitoring technology (BY$/unit)

co2mon_cap(mt) – unit capital costs for mt^{th} CO_2 monitoring technology (BY$/unit)

co2mon_capmultom(mt) – unit cost of drilling a deep monitoring well based on data in the API Joint Association survey of 2014 drilling costs (2014$/unit)

co2mon_lenom1(mt), co2mon_lenom2(mt) – length dependent annual O&M costs for the mt^{th} CO_2 monitoring technology (BY$/ft-unit)

co2mon_fixom1(mt), co2mon_fixom2(mt) – fixed annual O&M costs for the mt^{th} CO_2 monitoring technology (BY$/unit)

The variables *co2mon_stat(mt)*, *co2mon_capmultom(mt)*, *co2mon_lenom1(mt), co2mon_lenom2(mt)*, *co2mon_fixom1(mt)*, and *co2mon_fixom2(mt)* are user inputs.

4.10.8.2 O&M Cost Cash Flows for CO_2 Monitoring Technologies

The O&M costs for CO_2 monitoring technologies in each year *iy* is stored in the array *fldco2monopcn(iy)*. The following algorithm is used to assign costs to each item in this array:

```
Do iy = 1, nfldeconyr
    ! For year iy, add all the O&M costs for all CO2 monitoring technologies
    ! Store sum in co2mon_temp
    co2mon_temp = 0.0
    Do mt = 1, maxco2mon
        ! O&M costs for CO2 monitoring technology in year iy
        co2mon_temp = co2mon_temp + co2mon_unitsactive(mt, iy) ·
            co2mon_om(mt)
        ! Add costs for baseline or background monitoring for all
        ! units that were installed the previous year
        If( iy >= 2 and iy <= (flddevyr+1) ) Then
            co2mon_temp = co2mon_temp + co2mon_unitsinstall(mt, iy-1)) ·
                co2mon_om(mt) · ( co2mon_yronemult – 1.0 )
        End If
        ! If this is the last year of operation of the CO2 EOR facility,
        ! add a lumpsum of the O&M costs for CO2 monitoring for the
        ! number of years of monitoring after CO2 EOR ends that is
        ! required by the MRV Plan
        If( iy == nfldeconyr ) Then
```

```
                co2mon_temp = co2mon_temp + co2mon_unitsactive(mt, iy) ·
                    co2mon_om(mt) · co2mon_postinjdur
            End If
        End Do ! mt loop
        ! Add cost for annual reporting of monitoring results
        ! Include reporting cost for each year of CO2 monitoring
        ! after CO2 EOR ends that is required by the MRV Plan
        co2mon_temp = co2mon_temp + co2mon_rep
        If( iy == nfldeconyr ) Then
            co2mon_temp = co2mon_temp + co2mon_rep · co2mon_postinjdur
        End If
        fldco2monopcn(iy) = co2mon_temp · fac
End Do ! iy loop
```

Where:

nfldeconyr – number of years for the CO_2 EOR project

co2mon_temp – variable storing total annual O&M costs incurred in CO_2 EOR project year *iy* (BY$)

maxco2mon – maximum number of CO_2 monitoring technologies

co2mon_unitsactive(mt, iy) – number of units of the mt^{th} CO_2 monitoring technology that are active or operating in CO_2 EOR project year *iy* of the CO_2 EOR project

co2mon_om(mt) – unit annual O&M costs for mt^{th} CO_2 monitoring technology (BY$/unit)

flddevyr – number of years needed to implement CO_2 EOR at all of the patterns

co2mon_unitsinstall(mt, iy) – number of units of the mt^{th} CO_2 monitoring technology that are implemented in CO_2 EOR project year *iy* of the CO_2 EOR project

co2mon_yronemult – multiplier for first-year O&M costs to account for the cost of baseline or background monitoring

co2mon_postinjdur – years of CO_2 monitoring required after the CO_2 EOR facility ceases CO_2 EOR operations

co2mon_rep – annual cost of assembling CO_2 monitoring data, analyzing this data, writing required reports and submitting the reports to the appropriate regulatory authority (BY$)

fldco2monopcn (iy) – O&M costs for CO_2 monitoring technologies in CO_2 EOR project year *iy* (MBY$)

fac – conversion factor, *fac* = 0.001 MBY$/BY$

The CO_2 EOR facility does not become operational until the second year (first operational year) of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldco2monopcn(1)* = 0). The variables *co2mon_yronemult*, *co2mon_postinjdur*, and *co2mon_rep* are user inputs.

4.10.9 Process Contingency O&M Costs

The process contingency O&M costs are intended to compensate for uncertainty in O&M cost estimates caused by performance uncertainties associated with the development status of a technology. Process contingencies are applied to each technology based on its development status. The array *fldprocconopcn(iy)* provides the process contingency for O&M costs in CO_2 EOR operation year *iy*. Since CO_2 EOR is a mature technology, a process contingency has not been applied to any of the cost elements in the FE/NETL Onshore CO_2 EOR Cost Model. Therefore, each element in the array *fldprocconopcn(iy)* is set to zero.

4.10.10 General & Administrative Costs for Operations

The G&A cost represents the additional costs (including in-house accounting, human resources, legal, administrative, technical, operational, and development costs) that are essential for operating a successful CO_2 EOR project. The FE/NETL Onshore CO_2 EOR Cost Model calculates G&A O&M costs as a fraction of all O&M costs except for the cost of purchasing CO_2.

The G&A O&M cost in CO_2 project year *iy* is given by the following equation:

$$fldgaaopcn(iy) = fldtot2opcn(iy) \cdot omganda$$

Where:

fldgaaopcn(iy) – G&A O&M costs in CO_2 project year *iy* (MBY$)

fldtot2opcn(iy) – partial sum of O&M costs, excluding cost of purchased CO_2 and G&A O&M costs, in CO_2 project year *iy* (MBY$)

omganda – fraction of partial sum of O&M costs used to calculate G&A O&M costs

The CO_2 EOR facility does not become operational until the second year of the project, so there are no O&M costs in the first year of the CO_2 EOR project (i.e., *fldgaaopcn (1)* = 0). The variable *omganda* is a user input.

The variable *fldtot2opcn(iy)* is the sum of all O&M costs described in previous sections. The sum is a partial total of all O&M costs since this sum does not include the G&A O&M cost. The variable *fldtot2opcn(iy)* is calculated with the following equation:

$$\begin{aligned} fldtot2opcn(iy) &= fldplugwexpcn(iy) + fldwellopcn(iy) + fldliftopcn(iy) \\ &+ fldco2pltopcn(iy) + fldco2pipopcn(iy) + fldgathopcn(iy) \\ &+ fldco2monopcn(iy) + fldprocconopcn(iy) \end{aligned}$$

Where:

fldplugwexpcn(iy) – O&M costs for plugging wells in CO_2 EOR project year *iy* (MBY$)

fldwellopcn(iy) – well and lease O&M costs in CO_2 EOR project year *iy* (MBY$)

fldliftopcn(iy) – O&M costs for lifting produced fluid in CO_2 EOR project year *iy* (MBY$)

fldco2pltopcn(iy) – O&M costs for CO_2 recycling plant in CO_2 EOR project year *iy* (MBY$)

fldco2pipopcn(iy) – O&M costs for CO_2 connecting pipeline in CO_2 EOR project year *iy* (MBY$)

fldgathopcn(iy) – O&M costs for CO_2 delivery system in CO_2 EOR project year *iy* (MBY$)

fldco2monopcn (iy) – O&M costs for CO_2 monitoring technologies in CO_2 EOR project year *iy* (MBY$)

fldco2monopcn (iy) – process contingency costs related to O&M costs in CO_2 EOR project year *iy* (MBY$)

4.10.11 Total O&M Costs

Total O&M costs for the project include all base operating costs (well operating, fluid lifting, field equipment CO_2 recycling plant, distribution/gathering systems, pipelines, etc.), purchased CO_2 costs, and G&A O&M costs.

Total O&M costs in CO_2 EOR project year *iy* are given by the following equation:

$$fldtotopcn(iy) = fldco2puropcn(iy) + fldtot2opcn(iy) + fldgaaopcn(iy)$$

Where:

fldtotopcn(iy) – sum of all project O&M costs in CO_2 EOR project year *iy* (MBY$)

fldco2puropcn(iy) – cost of purchased CO_2 in CO_2 EOR project year *iy* when CO_2 is a cost to the operator (MBY$)

fldtot2opcn(iy) – partial sum of all project O&M costs in CO_2 EOR project year *iy* (MBY$). Does not include cost of purchasing CO_2 and G&A O&M costs

fldgaaopcn(iy) – G&A O&M costs in CO_2 EOR project year *iy* (MBY$)

4.11 Financing Parameters and the Discount Rate

This section discusses financing parameters used in the model and how the discount rate (*disc*) is calculated.

4.11.1 Weighted Average Cost of Capital

The weighted average cost of capital (WACC) is the rate that a company is expected to pay on average to all its security holders to finance its assets. The WACC represents the minimum return that a company must earn on an existing asset base to satisfy creditors and investors.

The WACC is given by the following equations:

$$wacc = freqty \cdot eqtyrate + (1 - freqty) \cdot adjdebtrate$$

$$adjdebtrate = (1 - fitrate) \cdot debtrate$$

Where:

wacc – weighted average cost of capital; this is annual interest rate

freqty – fraction of financing using equity (1 – *freqty* = amount of debt financing)

eqtyrate – minimum desired return on equity, after taxes

adjdebtrate – annual interest rate on debt adjusted for federal income tax effects

fitrate – federal annual income tax rate

debtrate – annual interest rate on debt

The variables *freqty*, *eqtyrate*, *fitrate*, and *debtrate* are user inputs. Values for *eqtyrate* and *debtrate* will depend on the escalation rate (*escrate*) as discussed in the next subsection. The user can do an analysis using an expected long-term escalation or inflation rate or the user can set the escalation rate to zero and do a constant dollar analysis. In either case, the values for *eqtyrate* and *debtrate* need to take the escalation rate into account. For an analysis using an expected long-term escalation rate (i.e., a positive value for *escrate*), the values for *eqtyrate* and *debtrate* should be higher than for a constant dollar analysis where the escalation rate is zero. In actual investment situations, investors have an expected rate of inflation in mind when they set the minimum desired return on equity and this value will be higher than minimum desired return on investment for a constant dollar analysis. Similarly, when banks lend money to a company, they have an expected rate of inflation in mind when they set the interest rate on the money they will lend.

4.11.2 Discount Rate for Net Present Value Calculations

As discussed earlier, the FE/NETL Onshore CO_2 EOR Cost Model generates three types of cash flows. Revenues and costs are first generated in constant dollars in a base year determined by the user. This base year is currently 2018 for cost data provided in the model. These constant dollar cash flows are then converted into nominal dollars by applying an escalation factor based on the escalation rate (*escrate*). The variable *escrate* is a user input that should reflect the user's best estimate of how revenues and costs in the oil industry will increase (inflate) or decrease (deflate) over the next 30 to 50 years. The nominal dollar cash flows are then discounted using a discount rate (*disc*) that converts nominal dollar cash flows to present value cash flows.

If the escalation rate is greater than zero, then the FE/NETL Onshore CO_2 EOR Cost Model will set the discount rate to the WACC. The values for *eqtyrate* and *debtrate* used to calculate the WACC should incorporate the influence of the positive escalation rate. If the escalation rate is equal to zero, then a constant dollar evaluation is performed, and the model can use the WACC or an interest rate supplied by the user (*rdisc*). If the WACC is used, then the values for *eqtyrate* and *debtrate* used to calculate the WACC should incorporate the influence of a zero escalation rate.

The algorithm used to calculate the discount rate, *disc*, is as follows:

```
If( escrate <= 10^-10 ) Then
    If( rdisc <= -0.99 ) Then
        disc = wacc
    Else
        disc = rdisc
    End If
Else
    disc = wacc
End If
```

Where:

escrate – annual escalation rate or long-term inflation rate for the oil industry

rdisc – annual real discount rate; used when escalation rate is 0

disc – annual discount rate

wacc – annual weighted average cost of capital rate

The variables *escrate* and *rdisc* are user inputs.

4.12 Earnings Before Federal Income Taxes

This section discusses how the earnings before federal income taxes and the internal rate of return for the project are calculated in the FE/NETL Onshore CO_2 EOR Cost Model.

4.12.1 Earnings Before Federal Income Taxes

The earnings before federal income taxes (*ebt*) is the gross revenue minus capital costs, O&M costs, royalties, severance taxes, and ad valorem taxes.

The earnings before federal income taxes in CO_2 EOR project year *iy* is given by the equation:

$$fdlebtcn(iy) = fldgrevcn(iy) - fldtotcapcn(iy) - fldtotopcn(iy) - fldroycn(iy) - fldsevcn(iy) - fldavtcn(iy)$$

Where:

fldebtcn(iy) – earnings before federal income taxes in CO_2 EOR project year *iy* (MBY$)

fldgrevcn(iy) – total gross revenue for the project in CO_2 EOR project year *iy* (MBY$)

fldtotcapcn(iy) – total capital costs for the project in CO_2 EOR project year *iy* (MBY$)

fldtotopcn(iy) – total operating costs for the project in CO_2 EOR project year *iy* (MBY$)

fldroycn(iy) – royalties paid to mineral rights holders in CO_2 EOR project year *iy* (MBY$)

fldsevcn(iy) – severance taxes paid in CO_2 EOR project year *iy* (MBY$)

fldavtcn(iy) – ad valorem taxes paid in CO_2 EOR project year *iy* (MBY$)

Earnings before federal taxes in nominal dollars is given by the equation:

$$fdlebtnm(iy) = fldebtcn(iy) \cdot escfac(iy)$$

Where:

fldebtnm(iy) – earnings before federal income taxes in CO_2 EOR project year *iy* in nominal dollars (M$)

escfac(iy) – escalation factor in CO_2 EOR project year *iy*

Earnings before federal taxes in present value dollars is given by the equation:

$$fdlebtpv(iy) = fldebtnm(iy) \cdot discfac(iy)$$

Where:

fldebtpv(iy) – earnings before federal income taxes in CO_2 EOR project year *iy* in present value dollars (M$)

discfac(iy) – discount factor in CO_2 EOR project year *iy*

None of the variables used to calculate earnings before federal income taxes are user inputs.

4.12.2 Internal Rate of Return Before Federal Income Taxes

In the discussion above, the discount rate (*disc*) is set by an algorithm (it will typically be the WACC) and the discount rate is used to calculate discount factors and present value cash flows for the earnings before federal income taxes. If all the values in this cash flow are summed, the result can be positive or negative. If this sum is zero, then the cash flows will generate enough earnings to cover all costs including the interest on debt and the investors minimum return on equity (assuming federal income taxes are ignored).

Given present value cash flows for earnings before federal income taxes, the discount rate that results in the sum of these cash flows equaling zero is called the internal rate of return (IRR). The IRR is the value for the discount rate, *disc*, that satisfies the following equation:

$$\sum_{iy=1}^{iy=nfldecony} fldebtnm(iy) \cdot discfac(iy) = 0$$

In this equation, the variable *discfac* is given by

$$discfac(iy) = \frac{1}{(1+disc)^{iy-1}}$$

The first equation can be re-written as

$$fldebtsumpv(disc) = \sum_{iy=1}^{iy=nfldeconyr} \frac{fldebtnm(iy)}{(1+disc)^{iy-1}})$$

The function *fldebtsumpv(disc)* is a function of *disc* and is the sum of the discounted nominal earnings before federal income taxes. The value of *disc* that makes *fldebtsumpv(disc)* equal to zero is the IRR. In the FE/NETL Onshore CO_2 EOR Cost Model, this value for *disc* is given by the variable *RORbt* where the letters *ROR* refer to "rate of return" and the letters *bt* refer to "before taxes".

The function *fldebtsumpv(disc)* is nonlinear with respect to the variable *disc*, so to find *RORbt*, an iterative algorithm must be employed. The FE/NETL Onshore CO_2 EOR Cost Model employs the interval bisection method to estimate *RORbt*. The algorithm used in the model is as follows:

```
! Initialize High ROR and LowROR
HighROR = 10.0
LowROR = 0.0
! Initialize itcount and iexit
itcount = 0
iexit = 0
Do While ( iexit == 0 )
    ! Check for two extreme cases before executing calculations for
    ! first iteration
    ! - First case is HighROR covering all costs including interest
    ! on debt and minimum return on equity even though HighROR
    ! is ridiculously high
    ! - The second case is LowROR not covering all costs even
    ! though LowROR is zero
    If( itcount == 0 ) Then
        Do iy = 1, nfldeconyr
            If (iy == 1) Then
                dischigh = 1.0
                fldebtcumhighpv = fldebtnm(iy) · dischigh
                disclow = 1.0
                fldebtcumlowpv = fldebtnm(iy) · disclow
            Else
                dischigh = 1.0 / (1.0 + HighROR) iy - 1
```

```
                    fldebtcumhighpv = fldebtcumhighpv + fldebtnm(iy) ·
                        dischigh
                    disclow = 1.0 / (1.0 + LowROR) iy - 1
                    fldebtcumlowpv = fldebtcumlowpv + fldebtnm(iy) · disclow
                End If
            End Do
            !  First case: Check for fldebtcumhighpv exceeding zero even
            !  with a ridiculously high HighROR
            !  Set MidROR to HighROR and Exit Do Loop
            If ( fldebtcumhighpv >= 0.0 ) Then
                MidROR = HighROR
                iexit = 1
                Exit
            End If
            !
            ! Second case: Check for fldebtcumlowpv being less than zero
            ! even with a value of LowROR equal to 0. This oil field does
            ! not have a positive discount rate that will cover all costs
            ! including interest on debt and return on equity not including
            ! federal income taxes.
            ! Set MidROR to LowROR and Exit Do Loop
            If ( fldebtcumlowpv <= 0.0 ) Then
                MidROR = LowROR
                iexit = 1
                Exit
            End If
        End If
        ! Extreme cases do not apply
        ! Execute calculations for an iteration
        MidROR = (LowROR + HighROR) · 0.5
        Do iy = 1, nfldeconyr
            If (iy == 1) Then
                discmid = 1.0
                fldebtcummidpv = fldebtnm(iy) · discmid
            Else
                discmid = 1.0 / (1.0 + MidROR) iy - 1
                fldebtcummidpv = fldebtcummidpv + fldebtnm(iy) · discmid
            End If
        End Do
        If ( fldebtcummidpv < 0.0 ) Then
             HighROR = MidROR
        Else
            LowROR = MidROR
```

End If
tolerance = HighROR - LowROR
itcount = itcount + 1
! Check for tolerance being less than criteria or the number
! of iterations exceeding limit
! If either is true, exit Do Loop (end iterations)
If(tolerance <= 10^{-5}) iexit = 1
If(itcount > 1000) iexit = 1
End Do
RORbt = MidROR

Where:

HighROR – high estimate of rate of return before federal income taxes

LowROR – low estimate of rate of return before federal income taxes

itcount – number of iterations

iexit – control variable. Set to 0 initially. Iterations stop when *iexit* is set to 1

nfldeconyr – number of years for the CO_2 EOR project

MidROR – mid-range estimate of rate of return before federal income taxes (mid-point between *HighROR* and *LowROR*)

dischigh – discount factor associated with *HighROR* in CO_2 EOR project year *iy*

fldebtcumhighpv – cumulative discounted nominal earnings before federal income taxes for *HighROR* (M$)

fieldebtnm(iy) – earnings before federal taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

disclow – discount factor associated with *LowROR* in CO_2 EOR project year *iy*

fldebtcumlowpv – cumulative discounted nominal earnings before federal income taxes for *LowROR* (M$)

discmid – discount factor associated with *MidROR* in CO_2 EOR project year *iy*

fldebtcummidpv – cumulative discounted nominal earnings before federal income taxes for *MidROR* (M$)

tolerance – difference between *HighROR* and *LowROR*. This variable is used to determine when to stop iterations

RORbt – estimated IRR before federal income taxes

The eventual value for *RORbt* should be between 0 and 1. A rate of return of 0 or less means the project will not be viable at any reasonable discount rate, since a negative discount rate implies a negative return on debt and equity. A rate of return greater than 1 is unlikely but is possible if costs are low relative to a very high price of oil (such as $1000/STB).

At the beginning of the first iteration, the value for *HighROR* is 10 and *LowROR* is 0. In the first iteration, the algorithm tests if either of these values is an extreme case. For the first case, the algorithm calculates a value for the variable *fldebtcumhighpv* (which is the value for the function *fldebtsumpv(HighROR)*). If *fldebtcumhighpv* is greater than or equal to 0, then this CO_2 EOR operation is profitable even with a ridiculously high return on investment. The variable *MidROR* is set to *HighROR* and the algorithm stops additional iterations.

For the second case, the algorithm calculates a value for the variable *fldebtcumlowpv* (which is the value for the function *fldebtsumpv(LowROR)*). If *fldebtcumlowpv* is less than or equal to 0, then this CO_2 EOR operation will not be profitable with any positive discount rate. The variable *MidROR* is set to *LowROR* and the algorithm stops additional iterations.

If neither of the extreme cases applies, the algorithm proceeds with the actual iterations. Each iteration involves the following calculations:

- The variable *MidRoR* is calculated as the average of *LowROR* and *HighROR*.
- The algorithm calculates a value for the variable *fldebtcummidpv* (which is the value for the function *fldebtsumpv(MidROR)*).
- If *fldebtcummidpv* is less than 0, then a discount rate equal to *MidROR* is not profitable. Because *MidROR* is less than *HighROR*, *HighROR* is too high, so *HighROR* is set to *MidROR*.
- If *fldebtcummidpv* is greater than or equal to 0, then a discount rate equal to *MidROR* is profitable. Because *MidROR* is greater than *LowROR*, *LowROR* is too low, so *LowROR* is set to *MidROR*.
- Last, the algorithm calculates the variable *tolerance*, which is the difference between *HighROR* and *LowROR*. If *tolerance* is less than or equal to 10^{-5}, then the iteration process has converged on values for *HighROR* and *LowROR* that are considered close enough, so the iterations are stopped. If *tolerance* exceeds 10^{-5}, then the algorithm proceeds through another iteration with the updated value for *HighROR* or *LowROR*.

After the iterations are stopped, the variable *RORbt* is set equal to *MidROR*.

None of the variables used to calculate *RORbt* are user inputs.

4.12.3 Year When Cumulative Nominal Earnings Become Positive

The FE/NETL Onshore CO_2 EOR Cost Model determines the year when the cumulative nominal earnings before federal income taxes becomes positive and stores this year in the variable *ipayoutbt*. The algorithm used to determine *ipayoutbt* is the following:

```
ipayoutbt = -1
rsum = 0.0
Do iy = 1, nfldeconyr
        rsum = rsum + fldebtnm(iy)
        If ( ( rsum >= 0.0 ) and ( ipayoutbt < 0 ) ) Then
                ipayoutbt = iy
```

Exit
End If
End Do
If(ipayoutbt < 0) ipayoutbt = 1000

Where:

ipayoutbt – year in CO_2 EOR project time frame when cumulative nominal earnings before federal income taxes become positive. This variable is set to -1 before entering the Do Loop. If this variable is still negative when it exits the Do Loop, then cumulative nominal earnings before federal income taxes are never positive during the CO_2 EOR project time frame. In this case, *ipayoutbt* is set to 1000, indicating that the cumulative nominal earnings before federal income taxes never become positive

rsum – variable that stores the cumulative sum of the nominal earnings before federal income taxes in CO_2 EOR project year *iy* (M$)

nfldeconyr – number of years for the CO_2 EOR project

fieldebtnm(iy) – earnings before federal taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

None of the variables used to calculate *ipayoutbt* are user inputs.

4.13 EARNINGS AFTER FEDERAL INCOME TAXES

This section discusses how the earnings after federal income taxes and the IRR for the project including federal income taxes are calculated in the FE/NETL Onshore CO_2 EOR Cost Model. To determine federal income taxes, capital costs must be depreciated. In addition, well drilling and completion costs must be divided between tangible and intangible costs.

4.13.1 Tangible and Intangible Well Drilling Costs.

The federal tax code allows well drilling costs to be divided between tangible and intangible costs. Tangible costs are depreciated while intangible costs can be expensed immediately. More precisely, intangible costs must be expensed entirely in the year the costs are incurred just like O&M costs, royalties, and state taxes. The decision by the owners of a CO_2 EOR operation to declare costs tangible or intangible depends on the financial situation of their entire enterprise, not just the specific CO_2 EOR field. To simplify things in the FE/NETL Onshore CO_2 EOR Cost Model, intangible costs are treated the same as any other cost that is expensed.

The tangible and intangible costs are calculated in a series of steps. First, the process contingency costs are calculated for well drilling and completion (*capcostproccont1*). Since well drilling and completion are mature technologies, the variable *capcostproccont1* is set equal to zero.

Second, capital costs for well drilling and completion in CO_2 EOR project year *iy* including the cost for acquiring leases and well-related permits and process contingency costs are calculated and stored in the variable *capcost1*.

$$capcost1 = flddcwcapnm(iy) + fldacqpatcapnm(iy) + capcostproccont1$$

Where:

capcost1 – capital costs for well drilling and completion including lease acquisition costs and process contingency costs (M$)

flddcwcapnm(iy) – well drilling and completion capital costs in CO_2 EOR project year *iy* in nominal dollars (M$)

fldacqpatcapnm(iy) – capital costs for lease acquisition and permitting in CO_2 EOR project year *iy* in nominal dollars (M$)

Third, the design and G&A costs associated with the well drilling capital costs and the process contingency costs are calculated for CO_2 EOR project year *iy*.

$$capcostga1 = capcost1 \cdot capganda$$

$$capcostprojcont1 = (capcost1 + capcostga1) \cdot ProjContFac$$

Where:

capcostga1 – design and G&A capital costs related to well drilling and completion (M$)

capcost1 – capital costs for well drilling and completion including lease acquisition costs and process contingency costs (M$)

capganda – fraction of capital costs that represents the additional cost of design and G&A

capcostprojcont1 – project contingency capital costs related to well drilling and completion (M$)

ProjContFac – project contingency factor

The variables *capganda* and *ProjContFac* are user inputs.

Fourth and last, total capital costs related to well drilling and completion are calculated and divided between tangible and intangible costs for CO_2 EOR project year *iy*.

$$capcosttot1 = capcost1 + capcostga1 + capcostproccont1 + capcostprojcont1$$

$$flddcwccaptangm(iy) = capcosttot1 \cdot dccapfractang$$

$$flddcwccapintgm(iy) = capcosttot1 \cdot (1 - dccapfractang)$$

Where:

capcosttot1 – sum of well drilling and completion related capital costs including project contingency costs, design and G&A costs, and project contingency costs (M$)

capcost1 – capital costs for well drilling and completion including lease acquisition costs and process contingency costs (M$)

capcostga1 – design and G&A capital costs related to well drilling and completion (M$)

capcostprojcont1 – project contingency capital costs related to well drilling and completion (M$)

flddcwccaptangnm(iy) – tangible well drilling and completion costs in nominal dollars in CO_2 EOR project year *iy* (to be depreciated) (M$)

dccapfractang – fraction of total well drilling and completion capital costs that are classified as tangible costs

flddcwccapintgnm(iy) – intangible well drilling and completion costs in nominal dollars in CO_2 EOR project year *iy* (to be expensed) (M$)

The variable *dccapfractang* is a user input.

4.13.2 Depreciation of Capital Costs

The FE/NETL Onshore CO_2 EOR Cost Model calculates depreciation of capital costs by classifying the capital costs in the model according to Internal Revenue Service guidance. [5] Based on this guidance, all costs in the model were divided into two groups. The first group of capital costs is depreciated using a 5-year, 200 percent depreciation schedule. The second group of capital costs is depreciated using a 7-year, 200 percent depreciation schedule.

4.13.2.1 Depreciation of Group 1 Capital Costs

The Group 1 capital costs are 3-D seismic survey costs and the tangible well drilling and completion costs. The total capital costs for 3-D seismic surveys in CO_2 EOR project year *iy* including process contingency costs, design and G&A costs, and project contingency costs are calculated using the following set of equations:

$$capcostproccont1 = 0.0$$

$$capcost1 = fldseiscapnm(iy) + capcostproccont1$$

$$capcostga1 = capcost1 \cdot capganda$$

$$capcostprojcont1 = (capcost1 + capcostga1) \cdot ProjContFac$$

$$capcosttot1 = capcost1 + capcostga1 + capcostprojcont1$$

Where:

capcostproccont1 – process contingency costs on capital costs for 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* (M$)

capcost1 – capital costs for 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* including process contingency costs (M$)

fldseiscapnm(iy) – capital cost for 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* (M$)

capcostga1 – design and G&A costs related to capital costs for 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* including process contingency costs (M$)

capganda – fraction of capital costs that represents the additional cost of design and G&A

capcostprojcont1 – project contingency costs related to capital costs for 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* including process contingency costs and design and G&A costs (M$)

ProjContFac – project contingency factor

capcosttot1 – sum of all capital costs associated with 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* (M$)

The variables *capganda* and *ProjContFac* are user inputs.

The total capital costs incurred in CO_2 EOR project year iy that will be depreciated using a 5-year 200 percent schedule are these total seismic imaging costs plus the tangible well drilling and completion costs. The equation for calculating these costs in CO_2 EOR project year *iy* is

$$fldcap5\_200(iy) = flddcwcaptangnm(iy) + capcosttot1$$

Where:

fldcap5_200(iy) – total capital costs subject to 5-year, 200% depreciation schedule in CO_2 EOR project year *iy* (M$)

flddcwccaptangnm(iy) – tangible well drilling and completion costs in nominal dollars in CO_2 EOR project year *iy* (M$)

capcosttot1 – sum of all capital costs associated with 3-D seismic survey in nominal dollars in CO_2 EOR project year *iy* (M$)

The capital costs that will be depreciated in each CO_2 EOR project year *iy* according the 5-year, 200 percent depreciation schedule is stored in the array variable *flddeprec5_200(iy)*. Depreciation is assumed to start in the second CO_2 EOR project year since CO_2 injection and oil production related to CO_2 EOR begins in this year. In the first CO_2 EOR project year, no oil is produced so there can be no revenue to potentially pay federal income taxes. The following algorithm is used to calculate values for the array variable *flddeprec5_200(iy)*.

```
! zero out the flddeprec5_200 array
flddeprec5_200 = 0.0
! Loop through all the CO2 EOR project years
Do iy = 1, nfldeconyr
    ! Capital costs incurred in CO2 EOR project year iy will be depreciated
    ! over a six year interval that starts in year jbeg and ends in year jend
    ! The indices jbeg and jend refer to CO2 EOR project years
```

```
!
! Calculate jbeg and jend
If( iy <= 1 ) Then
    jbeg = 2  ! start depreciation in first year of CO2 EOR operations
Else
     jbeg = iy
End If
! Calculate jend
jend = jbeg + 5
!
! Capital costs incurred in CO2 EOR project year iy are
! depreciated across a six year period according to the 5 year 200%
! depreciation schedule stored in the array macrs5_200(icnt)
icnt = 0
Do j = jbeg, jend
    icnt = icnt+1
    If ( j <= nfldeconyr ) Then
        flddeprec5_200(j) = flddeprec5_200(j) + fldcap5_200(iy) · macrs5_200(icnt)
    End If
End Do
End Do
```

Where:

flddeprec5_200(iy) – total depreciated capital costs subject to 5-year, 200% depreciation schedule in CO_2 EOR operating year *iy* (M$)

nfldeconyr – number of years the CO_2 EOR project lasts

jbeg, jend – indices storing the CO_2 EOR project years when depreciation begins and ends. Capital costs incurred in CO_2 EOR project year iy will be depreciated across these years. These are also indices pointing to locations in the array variable *flddeprec5_200*

icnt – index variable used to point to a location in the array variable *macrs5_200*

fldcap5_200(iy) – total capital costs subject to 5-year, 200% depreciation schedule in CO_2 EOR project year *iy* (M$)

macrs5_200(icnt) – array variable storing the depreciation schedule for 5-year 200% depreciation of capital. There are 6 values in this array and they are used to spread the capital cost incurred in CO_2 EOR project year *iy* across 6 years. The values in *macrs5_200* are 0.200, 0.320, 0.192, 0.1152, 0.1152, and 0.0576. These values sum to 1

4.13.2.2 Depreciation of Group 2 Capital Costs

The Group 2 capital costs are depreciated using a 7-year 200 percent depreciation schedule. The capital costs included in Group 2 include the costs for

- Constructing office buildings and access roads
- Recompleting existing wells
- Installing surface equipment at new production and injection wells
- Constructing a CO_2 recycling plant
- Installing a CO_2 connecting pipeline
- Installing an oil connecting pipeline
- Implementing CO_2 and water distribution systems and produced fluid gathering systems
- Installing water disposal wells
- Implementing water pipelines to the water disposal wells
- Installing CO_2 monitoring technologies for implementing an MRV Plan

The total capital costs for these items in CO_2 EOR project year *iy* including process contingency costs, design and G&A costs, and project contingency costs are stored in the array variable *fldcap7_200(iy)*. The elements of this array are calculated using the following set of equations:

$$capcostproccont1 = 0.0$$

$$\begin{aligned} capcost1 = {} & fldbldgroadcapnm(iy) + fldrecwcapnm(iy) + fldseprewcapnm(iy) \\ & + fldseinwcapnm(iy) + fldco2pltcapnm(iy) + fldco2pipcapnm(iy) \\ & + fldoilpipcapnm(iy) + fldgathcapnm(iy) + fldwatinjcapnm(iy) \\ & + fldwatpipcapnm(iy) + fldco2moncapnm(iy) + capcostproccont1 \end{aligned}$$

$$capcostga1 = capcost1 \cdot capganda$$

$$capcostprojcont1 = (capcost1 + capcostga1) \cdot ProjContFac$$

$$fldcap7\_200(iy) = capcost1 + capcostga1 + capcostprojcont1$$

Where:

capcostproccont1 – process contingency costs on capital costs for items in Group 2 in nominal dollars in CO_2 EOR project year *iy* (M$)

capcost1 – capital costs for the items in Group 2 including process contingency costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldbldgroadcapnm(iy) – capital costs for constructing office buildings and access roads in nominal dollars in CO_2 EOR project year *iy* (M$)

fldrecwcapnm(iy) – capital costs for recompleting old wells in nominal dollars in CO_2 EOR project year *iy* (M$)

fldseprwcapnm(iy) – capital costs for surface equipment associated with new production wells in nominal dollars in CO_2 EOR project year *iy* (M$)

fldseinwcapnm(iy) – capital costs for surface equipment associated with new injection wells in nominal dollars in CO_2 EOR project year *iy* (M$)

fldco2pltcapnm(iy) – capital costs for CO_2 recycling plant in nominal dollars in CO_2 EOR project year *iy* (M$)

fldco2pipcapnm(iy) – capital costs for CO_2 delivery pipeline in nominal dollars in CO_2 EOR project year *iy* (M$)

fldoilpipcapnm(iy) – capital costs for oil delivery pipeline in nominal dollars in CO_2 EOR project year *iy* (M$)

fldgathcapcn(iy) – capital costs for capital costs for implementing CO_2 and water distribution systems and produced fluid gathering systems in nominal dollars in CO_2 EOR project year *iy* (M$)

fldwatinjcapnm(iy) – capital costs for water disposal wells in nominal dollars in CO_2 EOR project year *iy* (M$)

fldwatpipcapnm(iy) – capital costs for pipelines carrying excess water to water disposal wells in nominal dollars in CO_2 EOR project year *iy* (M$)

fldco2moncapnm(iy) – capital costs for implementing CO_2 monitoring technologies in nominal dollars in CO_2 EOR project year *iy* (M$)

capcostga1 – design and G&A costs related to capital costs for items in Group 2 including process contingency costs in nominal dollars in CO_2 EOR project year *iy* (M$)

capganda – fraction of capital costs that represents the additional cost of design and G&A

capcostprojcont1 – project contingency costs related to capital costs for items in Group 2 including process contingency costs and design and G&A costs in nominal dollars in CO_2 EOR project year *iy* (M$)

ProjContFac – project contingency factor

fldcap7_200(iy) – sum of all capital costs for items in Group2 including process contingency costs, design and G&A costs, and project contingency costs in nominal dollars in CO_2 EOR project year *iy* (M$)

The variables *capganda* and *ProjContFac* are user inputs.

The capital costs that will be depreciated in each CO_2 EOR project year *iy* according the 7-year, 200 percent depreciation schedule are stored in the array variable *flddeprec7_200(iy)*. Depreciation is assumed to start in the second CO_2 EOR project year since CO_2 injection and oil production related to CO_2 EOR begins in this year. In the first CO_2 EOR project year, no oil is produced so there can be no revenue to potentially pay federal income taxes. The following algorithm is used to calculate values for the array variable *flddeprec7_200(iy)*:

```
! zero out the flddeprec7_200 array
flddeprec7_200 = 0.0
! Loop through all the CO2 EOR project years
Do iy = 1, nfldeconyr
```

```
! Capital costs incurred in CO2 EOR project year iy will be depreciated
! over an eight year interval that starts in year jbeg and ends in year jend
! The indices jbeg and jend refer to CO2 EOR project years
!
! Calculate jbeg and jend
If( iy <= 1 ) Then
    jbeg = 2  ! start depreciation in first year of CO2 EOR operations
Else
     jbeg = iy
End If
! Calculate jend
jend = jbeg + 7
!
! Capital costs incurred in CO2 EOR project year iy are
! depreciated across an eight year period according to the 7 year 200%
! depreciation schedule stored in the array macrs7_200(icnt)
icnt = 0
Do j = jbeg, jend
    icnt = icnt+1
    If ( j <= nfldeconyr ) Then
        flddeprec7_200(j) = flddeprec7_200(j) + fldcap7_200(iy) · macrs7_200(icnt)
    End If
End Do
End Do
```

Where:

flddeprec7_200(iy) – total depreciated capital costs subject to 7-year, 200% depreciation schedule in CO_2 EOR operating year *iy* (M$)

nfldeconyr – number of years the CO_2 EOR project lasts

jbeg, jend – indices storing the CO_2 EOR project years when depreciation begins and ends. Capital costs incurred in CO_2 EOR project year iy will be depreciated across these years. These are also indices pointing to locations in the array variable *flddeprec7_200*

icnt – index variable used to point to a location in the array variable *macrs7_200*

fldcap7_200(iy) – total capital costs subject to 7-year, 200% depreciation schedule in CO_2 EOR project year *iy* (M$)

macrs7_200(icnt) – array variable storing the depreciation schedule for 7-year 200% depreciation of capital. There are 8 values in this array and they are used to spread the capital cost incurred in CO_2 EOR project year *iy* across 8 years. The values in *macrs7_200* are 0.1429, 0.2449, 0.1749, 0.1249, 0.0893, 0.0892, 0.0893, and 0.0446. These values sum to 1

4.13.2.3 Total Depreciated Capital Costs

The total depreciated capital costs in nominal dollars in CO_2 EOR project year *iy* are stored in the array variable *flddeprectot(iy)*. This variable is the sum of the depreciated capital costs in each year *iy* using the two depreciation schedules for the two groups of capital costs:

$$flddeprectot(iy) = flddeprec5\_200(iy) + flddeprec7\_200(iy)$$

Where:

flddeprectot(iy) – total depreciated capital costs in nominal dollars in CO_2 EOR operating year *iy* (M$)

flddeprec5_200(iy) – total depreciated capital costs subject to 5-year, 200% depreciation schedule in nominal dollars in CO_2 EOR operating year *iy* (M$)

flddeprec7_200(iy) – total depreciated capital costs subject to 7-year, 200% depreciation schedule in nominal dollars in CO_2 EOR operating year *iy* (M$)

4.13.3 Earnings Subject to Federal Income Tax Before Net Operating Loss

Earnings subject to the federal income tax before net operating loss are the total revenue generated by the CO_2 EOR project minus depreciated capital costs, intangible capital costs, O&M costs, royalties, severance taxes, and ad valorem taxes. It should be noted that these are tax-basis earnings and not actual earnings. The tax basis earnings before net operating loss use depreciated capital costs incurred in a given year which are different and often lower than actual capital costs incurred in that year. Also, the tax basis earnings before net operating loss can be positive or negative. In the early years of a CO_2 EOR project when capital expenses are high and oil revenues are low (or nonexistent), tax basis earnings before net operating loss are often negative. When tax basis earnings before net operating loss are negative, the earnings in that year are a net operating loss.

Tax basis earnings subject to federal income tax before net operating loss in nominal dollars in CO_2 EOR project year *iy* are given by the following equation:

$$\begin{aligned} &flderntxbnol(iy) \\ &\quad = fldgrevnm(iy) - flddeprectot(iy) - flddcwcapintgnm(iy) \\ &\quad - fldtotopnm(iy) - fldroynm(iy) - fldsevnm(iy) - fldavtnm(iy) \end{aligned}$$

Where:

flderntxbnol(iy) – total tax basis earnings for calculating federal income tax before net operating losses in nominal dollars in CO_2 EOR project year *iy* (M$)

fldgrevnm(iy) – gross revenue in nominal dollars in CO_2 EOR project year *iy* (M$)

flddeprectot(iy) – total depreciated capital costs in nominal dollars in CO_2 EOR project year *iy* (M$)

flddcwcapintgnm(iy) – intangible capital costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldtotopnm(iy) – total O&M costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldroynm(iy) – royalties paid to mineral rights holders in nominal dollars in CO_2 EOR project year *iy* (M$)

fldsevnm(iy) – severance taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldavtnm(iy) – ad valorem taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

4.13.4 Federal Income Taxes

As noted previously, the tax basis earnings before net operating loss can be positive or negative and when they are negative then the operation has a net operating loss in that year. Federal income tax is only paid in any given year if the tax basis earnings are positive, so if the operation has a net operating loss in a year then no federal income taxes are owed. In addition, net operating losses can be accumulated over several years and used to offset the tax basis earnings before net operating loss. This will reduce or eliminate the federal income taxes owed to the federal government.

The FE/NETL Onshore CO_2 EOR Cost Model calculates the net operating loss in CO_2 EOR project year *iy* (*fldnolyr(iy)*), the tax basis earnings after net operating loss in CO_2 EOR project year *iy* (*fldernaplnol(iy))* and the cumulative net operating loss at the beginning (*fldcumnolbeg(iy)*) and end (*fldcumnolend(iy)*) of year *iy* using the following algorithm. The comments in the algorithm text below should clarify what each step in the algorithm is accomplishing. The algorithm has two main parts. The part that is executed depends on whether the tax basis earnings before net operating loss in CO_2 EOR project year *iy* (*flderntxbnol(iy)*) is positive or negative.

```
! Determine cumulative net operating loss at beginning of year iy
! For iy >0, this is the cumulative net operating loss at the end of the previous year
If( iy == 1 ) Then
        fldcumnolbeg(iy) = 0.0
Else
        fldcumnolbeg(iy) = fldcumnolend(iy - 1)
End If
!
! Determine net operating loss in year iy and tax basis earnings after net operating
! loss (which is basis for calculating federal income taxes)
If( flderntxbnol(iy) > 0.0 ) Then
        !
        ! Tax basis earnings before net operating loss are positive in this year
        !   so federal income tax could be owed
        ! The net operating loss in this year is zero since the
        !    tax basis earnings before net operating loss are positive
        ! Earnings applied against cumulative net operating loss in this year
        !    are the tax basis earnings before net operating loss in this year
```

```
        ! The cumulative net operating loss at end of year before applying
        !    earnings is the same as at the beginning of year
        fldnolyr(iy) = 0.0
        fldernaplnol(iy) = flderntxbnol(iy)
        fldcumnolmid(iy) = fldcumnolbeg(iy) + fldnolyr(iy)
        !
        ! Are the tax basis earnings before net operating loss is greater than
        !    the cumulative net operating loss?
        If( fldernaplnol(iy) > fldcumnolmid(iy) ) Then
                !
                ! The tax basis earnings before net operating loss are greater than
                !    the cumulative net operating loss
                ! Therefore, the tax basis earnings after net operating loss is
                !    tax basis earnings before net operating loss minus
                !      the cumulative net operating loss
                ! The cumulative net operating loss at the end of the year is
                !    zero since is has all been used to offset positive tax basis earnings
                fldernaftnol(iy) = fldernaplnol(iy) - fldcumnolmid(iy)
                fldcumnolend(iy) = 0.0
        Else
                !
                ! The tax basis earnings before net operating loss are less than
                !    or equal to the cumulative net operating loss
                ! Therefore, the tax basis earnings after net operating loss is zero
                ! The cumulative net operating loss at the end of the year is
                !    the cumulative net operating loss at the beginning of the year
                !      minus the tax basis earnings before net operating loss
                fldernaftnol(iy) = 0.0
                fldcumnolend(iy) = fldcumnolmid(iy) - fldernaplnol(iy)
        End If
Else
        !
        ! Tax basis earnings before net operating loss are negative in this year
        !    so federal income tax will not be owed
        ! The net operating loss in this year is the negative of the
        !    tax basis earnings before net operating loss
        ! Earnings applied against cumulative net operating loss in this year
        !    are zero since the tax basis earnings before net operating loss
        !    are negative
        ! The cumulative net operating loss at the end of the year is
        !    the cumulative net operating loss at the beginning of the year
        !      plus the net operating loss incurred in this year
        fldnolyr(iy) = -flderntxbnol(iy)
        fldernaplnol(iy) = 0.0
```

fldcumnolmid(iy) = fldcumnolbeg(iy) + fldnolyr(iy)
fldernaftnol(iy) = 0.0
fldcumnolend(iy) = fldcumnolmid(iy)
End If

Where:

flderntxbnol(iy) – tax basis earnings for calculating federal income tax before net operating losses in nominal dollars in CO_2 EOR project year *iy* (M$)

fldcumnolbeg(iy) – cumulative net operating loss in nominal dollars at the beginning of CO_2 EOR project year *iy* (M$)

fldcumnolend(iy) – cumulative net operating loss in nominal dollars at the end of CO_2 EOR project year *iy* (M$)

fldnolyr(iy) – net operating loss incurred in CO_2 EOR project year *iy* (M$)

fldernaplnol(iy) – earnings applied against net operating loss in CO_2 EOR project year *iy* (M$)

fldcumnolmid(iy) – cumulative net operating losses at end of CO_2 EOR project year *iy* before earnings are applied (M$)

fldernaftnol(iy) – tax basis earnings for calculating federal income tax (i.e., after net operating losses) in nominal dollars in CO_2 EOR project year *iy* (M$)

The main output from this algorithm is the tax basis earnings for calculating federal income taxes in CO_2 EOR project year *iy* (*fldernaftnol(iy)*). The FE/NETL Onshore CO_2 EOR Cost Model calculates the federal income tax before a tax credit and also after applying a tax credit. The user determines if a tax credit is applied.

The model calculates the federal income tax before tax credits in CO_2 EOR project year *iy* (*fldfedinctxbfcr(iy)*) using the following algorithm:

If(fldernaftnol(iy) <= 0.0) Then
fldfedinctxbfcr(iy) = 0.0
Else
fldfedinctxbfcr(iy) = fldernaftnol(iy) · fitrate
End If

Where:

fldernaftnol(iy) – tax basis earnings for calculating federal income tax (i.e., after net operating losses) in nominal dollars in CO_2 EOR project year *iy* (M$)

fldfedinctxbfcr(iy) – federal income taxes in nominal dollars in CO_2 EOR project year *iy* before applying any tax credits (M$)

fitrate – federal income tax rate

The variable *fitrate* is a user input.

The model then determines the tax credit in CO_2 EOR project year *iy* using the following algorithm, which depends on the total capital investment, the cost of purchasing CO_2, and the tax credit rates, which are input by the user:

fldtxcr(iy) = ftxcrrate_cap · fldtotcapnm(iy) + ftxcrrate_co2 · fldco2puropnm(iy)

Where:

ftxcrrate_cap – federal income tax credit rate for CO_2 EOR on capital costs (fraction)

ftxcrrate_co2 – federal income tax credit rate for CO_2 EOR on the cost of purchased CO_2 (fraction)

fldtxcr(iy) – federal tax credits in nominal dollars in CO_2 EOR project year *iy* (M$)

fldtotcapnm(iy) – total capital costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldco2puropnm(iy) – purchased CO_2 cost in nominal dollars in CO_2 EOR project year *iy* (M$)

The variables *ftxcrrate_cap* and *ftxcrrate_co2* are user inputs.

Finally, the model determines the federal taxes owed in CO_2 EOR project year *iy* including tax credits using the following algorithm:

```
If( fldfedinctxbfcr(iy) <= fldtxcr(iy) ) Then
    fldfedtxafcr(iy) = 0.0
Else
    fldfedtxafcr(iy) = fldfedinctxbfcr(iy) - fldtxcr(iy)
End If
```

Where:

fldfedinctxbfcr(iy) – federal income taxes in nominal dollars in CO_2 EOR project year *iy* before applying any tax credits (M$)

fldtxcr(iy) – federal tax credits in nominal dollars in CO_2 EOR project year *iy* (M$)

fldfedtxafcr(iy) – federal income taxes in nominal dollars owed in CO_2 EOR project year *iy* after applying tax credits (M$)

None of these variables are user inputs.

4.13.5 Earnings After Federal Income Taxes

The FE/NETL Onshore CO_2 EOR Cost Model calculates earnings after federal income taxes based on the gross revenue minus all capital costs, total O&M costs, royalties, state taxes, and federal income taxes. The model calculates the earnings using total capital costs in each year, not depreciated capital costs. The model calculates earnings in nominal and present value dollars and also calculates the total earnings across all years for the CO_2 EOR project.

The model first calculates the earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* using the following equation:

$$fldernafttxnm(iy) = fldgrevnm(iy) - fldtotcapnm(iy) - fldtotopnm(iy) - fldroynm(iy) - fldsevnm(iy) - fldavtnm(iy) - fldfedtxafcr(iy)$$

Where:

fldernafttxnm(iy) – earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldgrevnm(iy) – gross revenue in nominal dollars in CO_2 EOR project year *iy* (M$)

fldtotcapnm(iy) – total capital costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldtotopnm(iy) – total O&M costs in nominal dollars in CO_2 EOR project year *iy* (M$)

fldroynm(iy) – royalty payments to mineral rights owners in nominal dollars in CO_2 EOR project year *iy* (M$)

fldsevnm(iy) – severance taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldavtnm(iy) – ad valorem taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldfedtxafcr(iy) – federal income taxes after applying tax credits in nominal dollars in CO_2 EOR project year *iy* (M$)

The model next calculates the earnings after federal income taxes in present value dollars in CO_2 EOR project year *iy* by applying the following equation:

$$fldernafttxpv(iy) = fldernafttxnm(iy) \cdot discfac(iy)$$

Where:

fldernafttxnm(iy) – earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldernafttxpv(iy) – earnings after federal income taxes in present value dollars in CO_2 EOR project year *iy* (M$)

discfac(iy) – discount factor for CO_2 EOR project year *iy*

The model next calculates the total earnings after federal income taxes in nominal and present value dollars using the following equations:

$$fldernaftxtotnm = Sum(fldernafttxnm)$$

$$fldernaftxtotpv = Sum(fldernafttxpv)$$

Where:

fldernaftxtotnm – total earnings after federal income taxes in nominal dollars for all CO_2 EOR project years (M$)

fldernafttxnm(iy) – earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

fldernaftxtotpv – total earnings after federal income taxes in present value dollars for all CO_2 EOR project years (M$)

fldernafttxpv(iy) – earnings after federal income taxes in present value dollars (M$). The function *Sum(x)* is a built-in Fortran function that sums all the elements in the array variable x

If the variable *fldernaftxtotpv* is positive, then the CO_2 EOR project will cover all costs including interest on debt and the minimum return on equity desired by the investors.

4.13.6 Internal Rate of Return After Federal Income Taxes

The procedure presented earlier for calculating the IRR for nominal earnings before federal income taxes is used to calculate the IRR for nominal earnings after federal income taxes. Given present value cash flows for earnings after federal income taxes, the discount rate that results in the sum of these cash flows equaling zero is the IRR. The IRR is the value for the discount rate, *disc*, that satisfies the following equation:

$$\sum_{iy=1}^{iy=nfldecon} fldernafttxnm(iy) \cdot discfac(iy) = fldernaftxtotpv = 0$$

In this equation, the variable *discfac* is given by

$$discfac(iy) = \frac{1}{(1+disc)^{iy-1}}$$

The first equation can be re-written as

$$fldernaftxtotpv(disc) = \sum_{iy=1}^{iy=nfldeconyr} \frac{fldernafttxnm(iy)}{(1+disc)^{iy-1}})$$

The function *fldernaftxtotpv(disc)* is a function of *disc* and is the sum of the discounted nominal earnings after federal income taxes. The value of *disc* that makes *fldernaftxtotpv(disc)* equal to zero is the IRR. In the FE/NETL Onshore CO_2 EOR Cost Model, this value for *disc* is given by the variable *RORat* where the letters *ROR* refer to "rate of return" and the letters *at* refer to "after taxes".

The function *fldernaftxtotpv(disc)* is nonlinear with respect to the variable *disc*, so to find *RORat*, an iterative algorithm must be employed. The FE/NETL Onshore CO_2 EOR Cost Model employs the interval bisection method to estimate *RORat*. The algorithm used in the model is as follows:

! Initialize High ROR and LowROR

```
HighROR = 10.0
LowROR = 0.0
! Initialize itcount and iexit
itcount = 0
iexit = 0
Do While ( iexit == 0 )
        ! Check for two extreme cases before executing calculations for
        ! first iteration
        ! - First case is HighROR covering all costs including principal and interest
        ! on debt and minimum return on equity even though HighROR
        ! is ridiculously high
        ! - The second case is LowROR not covering all costs even
        ! though LowROR is zero
        If( itcount == 0 ) Then
                Do iy = 1, nfldeconyr
                        If (iy == 1) Then
                                dischigh = 1.0
                                fldeatcumhighpv = fldernafttxnm(iy) · dischigh
                                disclow = 1.0
                                fldeatcumlowpv = fldernafttxnm(iy) · disclow
                        Else
                                dischigh = 1.0 / (1.0 + HighROR) ^ iy - 1
                                fldeatcumhighpv = fldeatcumhighpv + fldernafttxnm(iy) ·
                                    dischigh
                                disclow = 1.0 / (1.0 + LowROR) ^ iy - 1
                                fldeatcumlowpv = fldeatcumlowpv + fldernafttxnm(iy) ·
                                    disclow
                        End If
                End Do
                !  First case: Check for fldeatcumhighpv exceeding zero even
                !  with a ridiculously high HighROR
                !  Set MidROR to HighROR and Exit Do Loop
                If ( fldeatcumhighpv >= 0.0 ) Then
                        MidROR = HighROR
                        iexit = 1
                        Exit
                End If
                !
                ! Second case: Check for fldeatcumlowpv being less than zero
                ! even with a value of LowROR equal to 0. This oil field does
                ! not have a positive discount rate that will cover all costs
                ! including principal and interest on debt and return on
                ! equity including federal income taxes.
```

```
            ! Set MidROR to LowROR and Exit Do Loop
            If ( fldeatcumlowpv <= 0.0 ) Then
                  MidROR = LowROR
                  iexit = 1
                  Exit
            End If
      End If
      ! Extreme cases do not apply
      ! Execute calculations for an iteration
      MidROR = (LowROR + HighROR) · 0.5
      Do iy = 1, nfldeconyr
            If (iy == 1) Then
                  discmid = 1.0
                  fldeatcummidpv = fldernafttxnm(iy) · discmid
            Else
                  discmid = 1.0 / (1.0 + MidROR) iy - 1
                  fldeatcummidpv = fldeatcummidpv + fldernafttxnm(iy) · discmid
            End If
      End Do
      If ( fldeatcummidpv < 0.0 ) Then
             HighROR = MidROR
      Else
            LowROR = MidROR
      End If
      tolerance = HighROR - LowROR
      itcount = itcount + 1
      ! Check for tolerance being less than criteria or the number
      ! of iterations exceeding limit
      ! If either is true, exit Do Loop (end iterations)
      If( tolerance <= 10-5 ) iexit = 1
      If( itcount > 1000 ) iexit = 1
End Do
RORat = MidROR
```

Where:

HighROR – high estimate of rate of return before federal income taxes

LowROR – low estimate of rate of return before federal income taxes

itcount – number of iterations

iexit – control variable. Set to 0 initially. Iterations stop when *iexit* is set to 1

nfldeconyr – number of years for the CO_2 EOR project

MidROR – mid-range estimate of rate of return before federal income taxes (mid-point between *HighROR* and *LowROR*)

dischigh – discount factor associated with *HighROR* in CO_2 EOR project year *iy*

fldeatcumhighpv – cumulative discounted nominal earnings after federal income taxes for *HighROR* (M$)

fldernafttxnm(iy) – earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

disclow – discount factor associated with *LowROR* in CO_2 EOR project year *iy*

fldeatcumlowpv – cumulative discounted nominal earnings after federal income taxes for *LowROR* (M$)

discmid – discount factor associated with *MidROR* in CO_2 EOR project year *iy*

fldeatcummidpv – cumulative discounted nominal earnings after federal income taxes for *MidROR* (M$)

tolerance – difference between *HighROR* and *LowROR*. This variable is used to determine when to stop iterations

RORat – estimated IRR after federal income taxes

The eventual value for *ROabt* should be between 0 and 1. A rate of return of 0 or less means the project will not be viable at any reasonable discount rate, since a negative discount rate implies a negative return on debt and equity. A rate of return greater than 1 is unlikely but is possible if costs are low relative to a very high price of oil (such as $1000/STB).

At the beginning of the first iteration, the value for *HighROR* is 10 and *LowROR* is 0. In the first iteration, the algorithm tests if either of these values is an extreme case. For the first case, the algorithm calculates a value for the variable *fldeatcumhighpv* (which is the value for the function *fldernaftxtotpv(HighROR)*). If *fldeatcumhighpv* is greater than or equal to 0*,* then this CO_2 EOR operation is profitable even with a ridiculously high return on investment. The variable *MidROR* is set to *HighROR* and the algorithm stops additional iterations.

For the second case, the algorithm calculates a value for the variable *fldeatcumlowpv* (which is the value for the function *fldernaftxtotpv(LowROR)*). If *fldeatcumlowpv* is less than or equal to 0*,* then this CO_2 EOR operation will not be profitable with any positive discount rate. The variable *MidROR* is set to *LowROR* and the algorithm stops additional iterations.

If neither of the extreme cases applies, the algorithm proceeds with the actual iterations. Each iteration involves the following calculations:

- The variable *MidRoR* is calculated as the average of *LowROR* and *HighROR*.
- The algorithm calculates a value for the variable *fldabtcummidpv* (which is the value for the function *fldernaftxtotpv(MidROR)*).
- If *fldeatcummidpv* is less than 0, then a discount rate equal to *MidROR* is not profitable. Because *MidROR* is less than *HighROR*, *HighROR* is too high, so *HighROR* is set to *MidROR*.

- If *fldeatcummidpv* is greater than or equal to 0, then a discount rate equal to *MidROR* is profitable. Because *MidROR* is greater than *LowROR*, *LowROR* is too low, so *LowROR* is set to *MidROR*.
- Last, the algorithm calculates the variable *tolerance*, which is the difference between *HighROR* and *LowROR*. If *tolerance* is less than or equal to 10^{-5}, then the iteration process has converged on values for *HighROR* and *LowROR* that are considered close enough, so the iterations are stopped. If *tolerance* exceeds 10^{-5}, then the algorithm proceeds through another iteration with the updated value for *HighROR* or *LowROR*.

After the iterations are stopped, the variable *RORat* is set equal to *MidROR*.

None of the variables used to calculate *RORat* are user inputs.

4.13.7 Year When Cumulative Nominal Earnings Become Positive

The FE/NETL Onshore CO_2 EOR Cost Model determines the year when the cumulative nominal earnings after federal income taxes become positive and stores this year in the variable *ipayoutat*. The algorithm used to determine *ipayoutat* is the following:

```
ipayoutat = -1
rsum = 0.0
Do iy = 1, nfldeconyr
        rsum = rsum + fldernafttxnm(iy)
        If ( ( rsum >= 0.0 ) and ( ipayoutat < 0 ) ) Then
                ipayoutat = iy
                Exit
        End If
End Do
If( ipayoutat < 0 ) ipayoutat = 1000
```

Where:

ipayoutat – year in CO_2 EOR project time frame when cumulative nominal earnings after federal income taxes become positive. This variable is set to -1 before entering the Do Loop. If this variable is still negative when it exits the Do Loop, then cumulative nominal earnings after federal income taxes are never positive during the CO_2 EOR project time frame. In this case, *ipayoutat* is set to 1000, indicating that the cumulative nominal earnings after federal income taxes never become positive

rsum – variable that stores the cumulative sum of the nominal earnings after federal income taxes in CO_2 EOR project year *iy* (M$)

nfldeconyr – number of years for the CO_2 EOR project

fldernafttxnm(iy) – earnings after federal income taxes in nominal dollars in CO_2 EOR project year *iy* (M$)

None of the variables used to calculate *ipayoutat* are user inputs.

4.14 DETERMINING THE BREAK-EVEN OIL PRICE

4.14.1 Description of Algorithms and Pseudo Code

The break-even oil price is the market oil price where the sum of the present value earnings after federal income taxes across all CO_2 EOR project years (the variable *fldernaftxtotpv*) is zero. The FE/NETL Onshore CO_2 EOR Cost Model determines the break-even oil price using an iterative procedure where the oil price is changed in each iteration and the variable *fldernaftxtotpv* is re-calculated until this variable is essentially zero. Because the revenues and many costs depend on the market price of oil, the procedure requires all the cash flows to be re-calculated each time the oil price is changed. In the FE/NETL Onshore CO_2 EOR Cost Model, two subroutines perform all the revenue and cost calculations. The subroutine CostCalcs calculates a variety of intermediate revenue and cost related values, such as oil prices, CO_2 prices, unit capital costs, unit O&M costs and the weighted average cost of capital. The subroutine FieldEcon calculates all cash flows, including revenues, royalties, state taxes, capital costs, O&M costs, and federal income taxes. The cash flows are initially in constant dollars, then escalated to nominal dollars and, finally, discounted to present value dollars. The variable *fldernaftxtotpv* (i.e., the sum of the present value earnings after federal income taxes across all CO_2 EOR project years) is calculated in this subroutine.

The break-even oil price is calculated using an iterative process very similar to the one described earlier for determining the IRR. The variable *fldernaftxtotpv* is a monotonically increasing function of the oil price (i.e., as the oil price increases, the variable *fldernaftxtotpv* increases). The goal of the process is to find the oil price, rounded up to the nearest integer value, that makes the variable *fldernaftxtotpv* slightly positive. The break-even oil price is stored in the variable *oilpriceatbrev* (with units of BY$/STB). The oil price associated with this variable generates a value of *fldernaftxtotpv* that is equal to or greater than zero. If the oil price stored in *oilpriceatbrev* is reduced by 1 BY$/STB then the value of *fldernaftxtotpv* associated with this oil price will be negative.

The first step in this process is to find two oil prices that give values for the variable *fldernaftxtotpv* that bracket 0 and, consequently, bracket the break-even oil price. The variable *ioilpricelow* gives a value for *fldernaftxtotpv* that is negative while the variable *ioilpricehigh* gives a value for *fldernaftxtotpv* that is positive. These prices are determined through a process that begins by setting the oil price to the base case oil price input by the user (stored in the variable *oilprbase*), calculating *fldernaftxtotpv* for this oil price, and proceeding based on the value for *fldernaftxtotpv*:

- If *fldernaftxtotpv* is greater than or equal to 0, then the base case oil price is higher than the break-even oil price. In this instance, the variable *ioilpricehigh* is initially set to the base case oil price and an iterative process is used to estimate a value for *ioilpricelow*. The iterative process is as follows:
    - The variable *ioilpricelow* is set equal to *ioilpricehigh*.
    - Step A1: The value for *ioilpricelow* is set to *ioilpricelow* divided by the variable *mult1* (which equals 1.5) and *fldernaftxtotpv* is calculated.

  - If *fldernaftxtotpv* is less than zero, then this value for *ioilpricelow* is less than the break-even oil price and values for *ioilpricelow* and *ioilpricehigh* have been found that bracket the break-even oil price. The iterative process is stopped.
  - If *fldernaftxtotpv* is greater than zero, then this value for *ioilpricelow* is greater than the break-even oil price, so values for *ioilpricelow* and *ioilpricehigh* have not been found that bracket the break-even oil price. In this instance, *ioilpricehigh* is set equal to *ioilpricelow* and the process returns to Step A1 for another iteration.

- Otherwise, *fldernaftxtotpv* is less than 0, so the base case oil price is less than the break-even oil price. In this instance, the variable *ioilpricelow* is initially set to the base case oil price and an iterative process is used to estimate a value for *ioilpricehigh*. The iterative process is as follows:
  - The variable *ioilpricehigh* is set equal to *ioilpricelow.*
  - Step A2: The value for *ioilpricehigh* is set to *ioilpricehigh* multiplied by the variable *mult1* (which equals 1.5) and *fldernaftxtotpv* is calculated.
  - If *fldernaftxtotpv* is greater than zero, then this value for *ioilpricehigh* is greater than the break-even oil price and values for *ioilpricelow* and *ioilpricehigh* have been found that bracket the break-even oil price. The iterative process is stopped.
  - If *fldernaftxtotpv* is less than zero, then this value for *ioilpricehigh* is less than the break-even oil price, so values for *ioilpricelow* and *ioilpricehigh* have not been found that bracket the break-even oil price. In this instance, *ioilpricelow* is set equal to *ioilpricehigh* and the process returns to Step A2 for another iteration.

The following pseudo code illustrates the process described above. The variables *ioilpricelow* and *ioilpricehigh* are integers. The Fortran function *Floor* rounds a real number down to its nearest integer value and the Fortran function *Ceiling* rounds a real number up to its nearest integer value. The definitions of the variables used in this code are provided in the next subsection.

```
!
! Set factor for increasing or decreasing estimates of the oil price
mult1 = 1.5
!
! Start with base case oil price input by user and call the subroutines
! CostCalcs and FieldEcon to calculate the variable fldernaftxtotpv
! (the sum of the present value earnings after federal income taxes
! across all CO2 EOR project years)
oilprice = oilprbase
Call CostCalcs
Call FieldEcon
!
! Find values for both ioilpricelow and ioilpricehigh based on initial
```

```
! value for fldernaftxtotpv
!   For iolpricehigh, fldernaftxtotpv is greater than or equal to 0
!   For iolpricelow, fldernaftxtotpv is less than 0
If( fldernaftxtotpv >= 0 ) Then
        !
        ! Base case oil price is above break-even oil price
        ! Initialize ioilpricehigh and ioilpricelow
        ioilpricehigh = Ceiling(oilprice)
        ioilpricelow = ioilpricehigh
        !
        ! Iterate to find ioilpricelow
        Do i = 1, 1000
                !
                ! Decrease oil price until fldernaftxtotpv is less than zero
                ioilpricelow = Floor( ioilpricelow / mult1 )
                !
                ! Check for ioilpricelow being less than or equal to 1
                If( ioilpricelow <= 1) Then
                        ioilpricelow = 1
                        Exit
                End If
                !
                ! Call the subroutines CostCalcs and FieldEcon
                ! to calculate the variable fldernaftxtotpv
                oilprice = ioilpricelow
                Call CostCalcs
                Call FieldEcon
                !
                ! Check if fldernaftxtotpv exceeds 0
                ! If yes, set ioilpricehigh to ioilpricelow and
                !    go through loop again until fldernaftxtotpv is less than 0
                ! If no, then ioilpricelow is value for further refinement
                If( fldernaftxtotpv > 0.0 ) Then
                        ioilpricehigh = ioilpricelow
                Else
                        Exit
                End If
        End Do
Else
        !
        ! Base case oil price is below break-even oil price
        ! Initialize ioilpricelow and ioilpricehigh
        ioilpricelow = Floor(oilprice)
        ioilpricehigh = ioilpricelow
```

```
        !
        ! Iterate to find ioilpricehigh
        Do i = 1, 1000
            !
            ! Increase oil price until fldernaftxtotpv is greater than zero
            ioilpricehigh = Ceiling( mult1 * ioilpricehigh )
            !
            ! Call the subroutines CostCalcs and FieldEcon
            ! to calculate the variable fldernaftxtotpv
            oilprice = ioilpricehigh
            Call CostCalcs
            Call FieldEcon
            !
            ! Check if fldernaftxtotpv is less than 0
            ! If yes, set ioilpricelow to ioilpricehigh and
            !    go through loop again until fldernaftxtotpv is greater than 0
            ! If no, then ioilpricehigh is value for further refinement
            If( fldernaftxtotpv < 0.0 ) Then
                ioilpricelow = ioilpricehigh
            Else
                Exit
            End If
            !
            ! Check if ioilpricehigh is unrealistically high
            ! If so, exit Do Loop
            If( ioilpricehigh >= 1000 ) Exit
        End Do
    End If
```

The variable *oilprbase* is input by the user.

The second step in this process checks for two extreme cases:

- In the first case, if *ioilpricehigh* exceeds 1000, then *ioilpricehigh* is set to 1000 BY$/STB and *fldernaftxtotpv* is calculated for this oil price. If *fldernaftxtotpv* is less than zero, then even this unrealistically high oil price is not high enough to allow the CO_2 EOR operation to be financially viable at this oil field. The variable *oilpriceatbrev*, which stores the break-even oil price, is set to *ioilpricehigh* and some additional results are stored in variables (see subsection below) and the process is halted. When the output from the FE/NETL Onshore CO_2 EOR Cost Model for an oil field displays a break-even oil price of 1000 BY$/STB this indicates that CO_2 EOR is not financially viable at this oil field.
- In the second case, if *ioilpricelow* is equal to or less than 1, then *ioilpricelow* is set to 1 and *fldernaftxtotpv* is calculated for this oil price. If *fldernaftxtotpv* is equal to or greater than zero, then CO_2 EOR is financially viable at this oil field even at an unrealistically low oil price. The variable *oilpriceatbrev* is set to *ioilpricelow* and some additional results are

stored in variables (see subsection below) and the process is halted. A break-even oil price of 1 BY$/STB indicates that this oil field is a great candidate for CO_2 EOR since CO_2 EOR is financially viable even at an oil price of 1 BY$/STB.

The third step in this process uses the bisection method to find the break-even oil price, which will be a value in between the values stored in *ioilpricelow* and *iolpricehigh*. The bisection method is an iterative method that hones in on the break-even oil price by adjusting the values for *ioilpricelow* and *iolpricehigh* until the difference between the values in these two variables is 1. The variables *ioilpricelow* and *iolpricehigh* are integers, so the smallest difference between the two (without being the same number) is 1. The value for *fldernaftxtotpv* associated with *ioilpricelow* is negative and the value for *fldernaftxtotpv* associated with *ioilpricehigh* is positive. Thus, when the difference between *ioilpricelow* and *ioilpricehigh* is 1, *ioilpricehigh* is the lowest integer oil price that gives a positive value for *fldernaftxtotpv*.

In the bisection method, an oil price that is halfway between *ioilpricelow* and *iolpricehigh* is calculated and stored in the variable *ioilpricemid*. The value for *fldernaftxtotpv* associated with *ioilpricemid* is calculated. If *fldernaftxtotpv* is positive, then this oil price is above the break-even oil price, so *iolpricehigh* is set equal to *iolpricemid*. If *fldernaftxtotpv* is negative, then this oil price is below the break-even oil price, so *iolpricelow* is set equal to *iolpricemid*. This process is repeated until the difference between the values stored in the variables *ioilpricelow* and *iolpricehigh* is 1.

The following pseudo code illustrates the process described above.

```
!
! Find break even oil price (rounded up to an integer dollar per barrel)
Do i = 1, 1000
        !
        ! Check if the difference between the high and low oil price is 1.
        ! If yes, then ioilpricehigh is break even oil price
        idif = ioilpricehigh - ioilpricelow
        If( idif == 1 ) Then
                Exit
        End If
        !
        ! Calculate an oil price between the high and low oil prices
        ioilpricemid = Floor( 0.5 * (ioilpricehigh + ioilpricelow) )
        If( ioilpricemid == ioilpricelow ) ioilpricemid = ioilpricelow + 1
        !
        ! Call the subroutines CostCalcs and FieldEcon
        ! to calculate the variable fldernaftxtotpv
        oilprice = ioilpricemid
        Call CostCalcs
        Call FieldEcon
        !
        ! Check for unlikely case that fldernaftxtotpv equals 0
```

```
        If( fldernaftxtotpv == 0.0 ) Then
            ioilpricehigh = ioilpricemid
            Exit
        End If
        !
        ! Reset high or low oil price based on fldernaftxtotpv
        If( fldernaftxtotpv < 0.0 ) Then
            ioilpricelow = ioilpricemid
        Else
            ioilpricehigh = ioilpricemid
        End If
    End Do
    oilpriceatbrev = ioilpricehigh
    … store additional results in variables (see subsection below)
```

None of these variables are user inputs.

4.14.2 Definition of Variables Used in Pseudo Code

The variables used in the pseudo code are defined below:

mult1 – factor used to increase or decrease oil price

oilprice – market oil price used in Fortran subroutines CostCalcs and FieldEcon (BY$/STB)

oilprbase – base case oil price input by the user (BY$/STB)

fldernaftxtotpv – total earnings after federal income taxes in present value dollars for all CO_2 EOR project years (M$)

ioilpricelow – estimate of market oil price that is an integer value and is below the break-even oil price (i.e., the value stored in *ioilpricelow* generates a value for *fldernaftxtotpv* that is negative) (BY$/STB)

ioilpricehigh – estimate of market oil price that is an integer value and is above the break-even oil price (i.e., the value stored in *ioilpricehigh* generates a value for *fldernaftxtotpv* that is positive) (BY$/STB)

i – integer index in iteration Do Loop used to limit number of iterations so an infinite loop does not occur

idif – integer variable that is difference between *ioilpricehigh* and *ioilpricelow*. When *idif* equals 1, the iterations are halted and the break-even oil price is set equal to *ioilpricehigh* (BY$/STB)

ioilpricemid – estimate of market oil price that is halfway between *ioilpricelow* and *ioilpricehigh* (BY$/STB)

oilpriceatbrev – break-even oil price or, more specifically, the oil price rounded up to the nearest integer value that generates a value for *fldernaftxtotpv* that is slightly positive (BY$/STB)

4.14.3 Additional Variables Calculated and Stored by Algorithms

In addition to the break-even oil price, a number of other variables that depend on the market oil price are also stored:

gaoilpriceatbrev – gravity adjusted oil price (BY$/STB)

loilpriceatbrev – lease oil price (BY$/STB)

dcwuncapbyatbrev – unit capital costs for drilling and completing a well in base year dollars (BY$/well)

co2pricecostatbrev – price of CO_2 if CO_2 is a cost to the CO_2 EOR operator (BY$/Mscf)

co2pricerevatbrev – price of CO_2 if CO_2 is a source of revenue for the CO_2 EOR operator (BY$/Mscf)

recyco2atbrev – unit costs to operate and maintain the CO_2 recycling plant (BY$/Mscf or MBY$/MMscf)

liftunopbyatbrev – unit operating cost for lifting produced water and oil (BY$/STB or MBY$/MSTB)

RORbtbrev – estimated IRR before federal income taxes

ipayoutbt – year in CO_2 EOR project time frame when cumulative nominal earnings before federal income taxes become positive. If *ipayoutbt* equals 1000, this indicates that the cumulative nominal earnings before federal income taxes never become positive

RORatbrev – estimated IRR after federal income taxes

ipayoutat – year in CO_2 EOR project time frame when cumulative nominal earnings after federal income taxes become positive. If *ipayoutat* equals 1000, this indicates that the cumulative nominal earnings after federal income taxes never become positive

At the break-even oil price, the estimated IRR after federal income taxes (*RORatbrev*) should be very close to the discount rate (*disc*) which is typically the WACC (*wacc*).

4.15 Determining the Break-Even CO_2 Price

4.15.1 Description of Algorithms and Pseudo Code

The break-even CO_2 price is the CO_2 price for a fixed market oil price where the sum of the present value earnings after federal income taxes across all CO_2 EOR project years (the variable *fldernaftxtotpv*) is zero. The FE/NETL Onshore CO_2 EOR Cost Model determines the break-even CO_2 price using an iterative procedure that is similar to the one used to calculate the break-even oil price. In the procedure, the CO_2 price is changed in each iteration and the variable *fldernaftxtotpv* is re-calculated until this variable is essentially zero. The subroutines CostCalcs and FieldEcon are called each time the CO_2 price is changed so that updated cash flows are used in each iteration.

Unlike the oil price, which is always positive, the CO_2 price (the value for the variable *co2price*) can be positive or negative. When the CO_2 price is positive, then CO_2 is treated as a cost to the CO_2 EOR operator (i.e., the CO_2 EOR operator pays for the CO_2 supplied to the operation). When the CO_2 price is negative, then CO_2 is treated as a source of revenue for the CO_2 EOR operation (i.e., the CO_2 EOR operator is paid to accept the CO_2 for use in CO_2 EOR). There is currently no market or regulatory mechanism for CO_2 to be a source of revenue for a CO_2 EOR operation, but this capability is in the model.

The procedure used to calculate the break-even CO_2 price depends on whether this price is positive or negative. In the FE/NETL Onshore CO_2 EOR Cost Model, three variables are used to store CO_2 prices. The first is the variable *co2price*, which can be positive or negative. The second is the variable *co2pricecost* which is positive if CO_2 is a cost to the CO_2 EOR operation and zero otherwise. If the variable *co2price* is positive then the variable *co2pricecost* is set equal to *co2price*. The third is the variable *co2pricerev*, which is positive if CO_2 is a source of revenue to the CO_2 EOR operation and zero otherwise. If the variable *co2price* is negative, the variable *co2pricerev* is set equal to the absolute value of *co2price*.

The variable *fldernaftxtotpv* is calculated with cash flows for the cost of purchasing CO_2 using the variable *co2pricecost* and separate cashflows for revenue incurred when the CO_2 EOR operator is paid to take CO_2 using the variable *co2pricerev*. Only one of these cash flows has nonzero values since the variable *co2pricecost* is zero when the variable *co2pricerev* is nonzero and vice versa. When *co2pricecost* is nonzero, the variable *fldernaftxtotpv* is a monotonically decreasing function of the variable *co2pricecost* (i.e., as the variable *co2pricecost* increases, the variable *fldernaftxtotpv* decreases). Thus, the goal of the iteration process is to find the highest value for *co2pricecost* that makes *fldernaftxtotpv* zero or slightly positive. In contrast, when *co2pricerev* is nonzero, the variable *fldernaftxtotpv* is a monotonically increasing function of the variable *co2pricerev* (i.e., as the variable *co2pricerev* increases, the variable *fldernaftxtotpv* increases). Thus, the goal of the iteration process is to find the lowest value for *co2pricerev* that makes *fldernaftxtotpv* zero or slightly positive. Since the goal of the iteration procedure is different depending on whether or not *co2pricecost* or *co2pricerev* is nonzero, it is important to first determine whether *fldernaftxtotpv* is positive when *co2pricecost* is zero.

To make the iterative process more computationally efficient, the process finds integer values for *co2pricecost* or *co2pricerev* that satisfy the goals discussed above. In the model calculations, the variables *co2pricecost* and *co2pricerev* have units of BY$/Mscf. In saline storage of CO_2, the cost of storing CO_2 is often expressed in units of $/tonne. For example, the break-even cost of storing CO_2 that is calculated by the FE/NETL CO_2 Saline Storage Cost Model is given in $/tonne. [6] To make the break-even price of CO_2 for CO_2 EOR easily comparable to conventional costs for storing CO_2, the integer values for calculating *co2pricecost* or *co2pricerev* are first expressed in BY$/tonne and then converted to their equivalent values in BY$/Mscf for the cash flow calculations. The integer values used in the iterative procedure are stored in the variables *ico2pricelow*, *ico2pricemid*, and *ico2pricehigh* with units of BY$/tonne. These variables are defined in the discussions below. The calculated break-even CO_2 prices are stored in the variables *co2pricecostatbrev*, which stores the break-even CO_2 price if CO_2 is a cost to the CO_2 EOR operation (and is zero otherwise), and *co2pricerevatbrev*, which stores the break-even CO_2

price if CO_2 is a source of revenue for the CO_2 EOR operation (and is zero otherwise). These latter two variables store values with units of BY$/Mscf.

The algorithm for determining the break-even CO_2 price comprises three major steps. The steps are described in the remainder of this subsection.

4.15.1.1 Step 1: Determine If CO_2 is a Cost or a Source of Revenue at the Break-Even CO_2 Price

In the first step, the price of CO_2 (the variable *co2price*) is set to zero and the variable *fldernaftxtotpv* (the present value earnings after federal income taxes across all CO_2 EOR project years) is calculated. If *fldernaftxtotpv* is positive then a price for CO_2 greater than zero will make *fldernaftxtotpv* negative. Since a positive value for *co2price* indicates that CO_2 is a cost to the CO_2 EOR operation, this indicates that the break-even CO_2 price represents a cost to the CO_2 EOR operation, not a source of revenue. Alternatively, if *fldernaftxtotpv* is negative then a price for CO_2 less than zero will make *fldernaftxtotpv* positive. Since a negative value for *co2price* indicates that CO_2 is a source of revenue to the CO_2 EOR operation, this indicates that the break-even CO_2 price represents a source of revenue to the CO_2 EOR operation, not a cost.

When *co2price* is set to 0 (actually a value greater than or equal to -10^{-6} BY$/Mscf and less than or equal to 10^{-6} BY$/Mscf) the FE/NETL Onshore CO_2 EOR Cost Model treats the price of CO_2 as a multiple of the oil price using the variable *co2prmult* to convert the oil price (in BY$/STB) to a CO_2 price (in BY$/Mscf). Thus, the algorithm sets the value of *co2price* to 10^{-4} BY$/Mscf and calculates *fldernaftxtotpv*. As discussed above, if *fldernaftxtotpv* is positive, this indicates the break-even CO_2 price is positive so CO_2 will be a cost to the CO_2 EOR operation at the break-even CO_2 price. The algorithm proceeds to Step 2. If *fldernaftxtotpv* is negative, this indicates the break-even CO_2 price is negative so CO_2 will be a source of revenue for the CO_2 EOR operation at the break-even CO_2 price. The algorithm proceeds to Step 3.

4.15.1.2 Step 2: Determine the Break-Even Price of CO_2 When CO_2 is a Cost for the CO_2 EOR Operation

The second step provides the algorithms for calculating the break-even price of CO_2 when CO_2 is a cost to the CO_2 EOR operation. The second step comprises several smaller steps.

Step 2A checks for a special case. The CO_2 price is set to 1 BY$/tonne (0.0529 BY$/Mscf) and the variable *fldernaftxtotpv* is calculated. The CO_2 price of 1 BY$/tonne is the lowest positive integer CO_2 price evaluated by this algorithm.

- If *fldernaftxtotpv* is negative, a CO_2 price of 10^{-4} BY$/Mscf is the break-even CO_2 price. The algorithm proceeds to Step 2B where the break-even CO_2 price and other variables are stored and the algorithm ends.

  Note: The Fortran code for calculating the break-even price of CO_2 is contained in a subroutine. A subroutine is a self-contained set of Fortran code that is called by the main program, another subroutine, or a function. The Return statement within a subroutine indicates that code for a section of the subroutine is completed and control

of the program returns to the main program, subroutine, or function that called the subroutine.

- If *fldernaftxtotpv* is positive, a CO_2 price of 10^{-4} BY$/Mscf is not the break-even price of CO_2 (the break-even price is a higher value) and the algorithm proceeds to Step 2C.

The following pseudo code illustrates the process described above:

```
!
! --- Step 2A: Check to see if a CO2 price of 1 BY$/tonne gives
! a negative value for fldernaftxtotpv. If yes, then a CO2 price
! around 0 BY$/tonne is the break-even CO2 price.
ico2pricehigh = 1   ! BY$/tonne
co2price = ico2pricehigh * dengsurfft3  ! BY$/Mscf
!
! Calculate fldernaftxtotpv
Call CostCalcs
Call FieldEcon
If( fldernaftxtotpv < 0.0 ) Then
        !
        ! --- Step 2B: The break-even CO2 price is around 0 BY$/Mscf
        co2price = 1.0e-4  ! BY$/Mscf
        !
        ! Calculate fldernaftxtotpv
        Call CostCalcs
        Call FieldEcon
        !
        ! Record values for selected variables in base year dollars
        co2pricecostatbrev = co2pricecost
        co2pricerevatbrev = co2pricerev
        … store additional results in variables (see subsection below)
        !
        ! Exit subroutine
        Return
End If
```

Step 2C finds two CO_2 prices, *ico2pricelow* and *ico2pricehigh* that bracket the break-even CO_2 price. At the conclusion of the Step 2C algorithm, the variable *ico2pricelow* will generate a positive value for the variable *fldernaftxtotpv*, while the variable *ico2pricehigh* will generate a negative value for the variable *fldernaftxtotpv*. The break-even CO_2 price will be between these two values or, possibly, equal to *ico2pricelow*. The algorithm starts by setting *ico2pricelow* to 1 BY$/tonne and then calculates a value for *ico2pricehigh* by multiplying *ico2pricelow* by the variable *mult1* and rounding the result up to the next highest integer. The variable *fldernaftxtotpv* is then calculated for the variable *ico2pricehigh*. If *fldernaftxtotpv* is negative, then acceptable values have been found for *ico2pricelow* and *ico2pricehigh* and the algorithm proceeds to Step 2E. If the variable *fldernaftxtotpv* is positive, then the variable *ico2pricehigh* is

too low, so the variable *ico2pricelow* is set to *ico2pricehigh* and the process is repeated with the variable *ico2pricehigh* increased until the variable *fldernaftxtotpv* is negative. When a negative value is calculated for *fldernaftxtotpv* then acceptable values have been found for *ico2pricelow* and *ico2pricehigh* and the algorithm proceeds to Step 2E.

A check is made at each iteration to see if the value for *ico2pricehigh* gives a positive value for the variable *fldernaftxtotpv* and *ico2pricehigh* also exceeds an upper limit stored in the variable *ico2priceuplim*. If *ico2pricehigh* exceeds this value, then the algorithm proceeds to Step 2D. In this step, the break-even CO_2 price is set to the upper limit (*ico2priceuplim*). The break-even CO_2 price and other variables are stored and the algorithm ends. The variable *ico2priceuplim* is currently set to 300 BY$/tonne, which is believed to be on the upper end of costs for direct air capture. It is unlikely that a CO_2 EOR operator would ever pay more than this value for CO_2, since the operator could, theoretically, run their own direct air capture system to obtain the CO_2 they need for their CO_2 EOR operations.

The following pseudo code illustrates the process described above:

```
!
! --- Step 2C: Find a high CO2 price that has a negative value for fldernaftxtotpv
! Initialize ico2pricelow
ico2pricelow = 1   ! BY$/tonne
mult1 = 1.5
!
! Set ico2priceuplim, an upper limit on ico2pricehigh
ico2priceuplim = 300  ! BY$/tonne
!
! Iterate to find ico2pricehigh
Do i = 1, 1000
        !
        ! Increase CO2 price until fldernaftxtotpv is less than zero
        ico2pricehigh = Ceiling(ico2pricelow * mult1)
        !
        ! Check if ico2pricehigh exceeds CO2 price upper limit (ico2priceuplim)
        ! If so, set ico2pricehigh to ico2priceuplim (in BY$/tonne)
        ! This is the highest break-even CO2 price allowed in this analysis.
        If( ico2pricehigh > ico2priceuplim ) Then
                ico2pricehigh = ico2priceuplim
        End If
        co2price = ico2pricehigh * dengsurfft3  ! BY$/Mscf
        !
        ! Calculate fldernaftxtotpv
        Call CostCalcs
        Call FieldEcon
        If( fldernaftxtotpv > 0.0 ) Then
                !
                ! The high CO2 price is still too low.
```

```
                !
                ! First check to see if ico2pricehigh equals or exceeds
                ! CO2 price upper limit (ico2priceuplim)
                ! If so, set the break-even price to ico2priceuplim (in BY$/tonne)
                If( ico2pricehigh >= ico2priceuplim ) Then
                        !
                        ! --- Step 2D: A CO2 price of ico2priceuplim gives a
                        ! positive value for fldernaftxtotpv.
                        ! This CO2 price is an upper bound on the
                        ! break-even CO2 price and results are stored for this price.
                        ico2pricehigh = ico2priceuplim  ! BY$/tonne
                        co2price = ico2pricehigh * dengsurfft3  ! BY$/Mscf
                        !
                        ! Calculate fldernaftxtotpv
                        Call CostCalcs
                        Call FieldEcon
                        !
                        ! Record values for selected variables in base year dollars
                        co2pricecostatbrev = co2pricecost
                        co2pricerevatbrev = co2pricerev
                         … store additional results in variables (see subsection below)
                        !
                        ! Exit subroutine
                        Return
                Else
                        !
                        ! Set low CO2 price to high CO2 price and go through
                        ! Do Loop again
                         ico2pricelow = ico2pricehigh
                End If
        Else
                !
                ! The high CO2 price produces a negative value for
                ! fldernaftxtotpv. Appropriate low and high CO2
                ! prices have been found.
                ! Exit the Do Loop and use the bisection method to find
                ! the break-even CO2 price which will be between the
                ! low and high CO2 price.
                Exit
        End If
End Do
```

In Step 2E, the bisection method is used to find the break-even CO_2 price, which will be a value in between the values stored in *ico2pricelow* and *ico2pricehigh*. The bisection method is an

iterative method that hones in on the break-even CO_2 price by adjusting the values for *ico2pricelow* and *ico2pricehigh* until the difference between the values in these two variables is 1. The variables *ico2pricelow* and *ico2pricehigh* are integers, so the smallest difference between the two (without being the same number) is 1. The value for *fldernaftxtotpv* associated with *ico2pricelow* is positive and the value for *fldernaftxtotpv* associated with *ico2pricehigh* is negative. Thus, when the difference between *ico2pricelow* and *ico2pricehigh* is 1, *ico2pricelow* is the highest integer CO_2 price that gives a positive value for *fldernaftxtotpv*.

In the bisection method, a CO_2 price that is halfway between *ico2pricelow* and *ico2pricehigh* is calculated and stored in the variable *ico2pricemid*. The value for *fldernaftxtotpv* associated with *ico2pricemid* is calculated. If *fldernaftxtotpv* is positive, then this CO_2 price is below the break-even CO_2 price, so *ico2pricelow* is set equal to *ico2pricemid*. If *fldernaftxtotpv* is negative, then this CO_2 price is above the break-even CO_2 price, so *ico2pricehigh* is set equal to *ico2pricemid*. This process is repeated until the difference between the values stored in the variables *ico2pricelow* and *ico2pricehigh* is 1. At this point, the algorithm proceeds to Step 2F where the break-even CO_2 price is set to *ico2pricelow.* The break-even CO_2 price and other variables are stored and the algorithm ends.

The following pseudo code illustrates the process described above.

```
!
! -- Step 2E: Use the bisection method to find the break-even
! CO2 price. The break-even CO2 price is a CO2 price between the low
! and high CO2 price where the CO2 price makes fldernaftxtotpv zero. In practice,
! the break-even CO2 price is the lower of two CO2 prices that differ by
! 1 BY$/tonne where the low CO2 price gives a zero or positive value
! for fldernaftxtotpv and the high CO2 price gives a negative value for
! fldernaftxtotpv.
! The bisection method keeps halving the difference between the low and high
! CO2 prices until the difference is 1 $BY/tonne.
! The low and high CO2 prices generated previously are
! appropriate starting values for the bisection method.
Do i = 1, 1000
        !
        ! Check if the difference between the high and low CO2 price is 1.
        ! If yes, then ico2pricelow is break even CO2 price
        idif = ico2pricehigh - ico2pricelow
        If( idif == 1 ) Then
                Exit
        End If
        !
        ! Calculate a CO2 price between the high and low CO2 prices
        ico2pricemid = Floor( 0.5 * (ico2pricehigh + ico2pricelow) )
        If( ico2pricemid == ico2pricelow ) Then
                ico2pricemid = ico2pricelow + 1
        End If
```

```
            !
            ! Calculate cash flows for this mid-range CO2 price
            co2price = ico2pricemid * dengsurfft3  ! BY$/Mscf
            !
            ! Calculate fldernaftxtotpv
            Call CostCalcs
            Call FieldEcon
            !
            ! Check for unlikely case that fldernaftxtotpv equals 0
            If( fldernaftxtotpv == 0.0 ) Then
                  ico2pricelow = ico2pricemid
                  Exit
            End If
            !
            ! Reset high or low CO2 price based on fldernaftxtotpv
            If( fldernaftxtotpv .LT. 0.0d0 ) Then
                  ico2pricehigh = ico2pricemid
            Else
                   ico2pricelow = ico2pricemid
            End If
            !
      End Do
      !
      ! --- Step 2F: Set break-even CO2 price to the low CO2 price
      co2price = ico2pricelow * dengsurfft3  ! BY$/Mscf
      !
      ! Calculate fldernaftxtotpv
      Call CostCalcs
      Call FieldEcon
      !
      ! Record values for selected variables in base year dollars
      co2pricecostatbrev = co2pricecost
      co2pricerevatbrev = co2pricerev
      … store additional results in variables (see subsection below)
      !
      ! Exit subroutine
      Return
```

4.15.1.3 Step 3: Determine the Break-Even Price of CO_2 When CO_2 is a Source of Revenue for the CO_2 EOR Operation

The third step provides the algorithms for calculating the break-even price of CO_2 when CO_2 is a source of revenue for the CO_2 EOR operation. The third step comprises several smaller steps.

Step 3A checks for a special case. The CO_2 price is set to a negative value slightly less than 0, -10^{-4} BY$/Mscf, and the variable *fldernaftxtotpv* is calculated. If the value for *fldernaftxtotpv* is positive, then a CO_2 price that is slightly less than 0 is the break-even CO_2 price. The algorithm proceeds to Step 3B where the break-even CO_2 price and other variables are stored and the algorithm ends. If *fldernaftxtotpv* is positive, then a CO_2 price of -10^{-4} BY$/Mscf is not the break-even price of CO_2 (the break-even price is a higher value) and the algorithm proceeds to Step 3C.

The following pseudo code illustrates the process described above:

```
!
! --- Step 3A: Check to see if a CO2 price slightly less than zero gives
! a positive value for fldernaftxtotpv. If yes, then a CO2 price around
! 0 BY$/tonne is the break-even CO2 price.
co2price = -1.0e-4  ! BY$/Mscf
!
! Calculate fldernaftxtotpv
Call CostCalcs
Call FieldEcon
If( fldernaftxtotpv > 0.0 ) Then
    !
    ! --- Step 3B: The break-even CO2 price is around 0 BY$/Mscf
    !
    ! Record values for selected variables in base year dollars
    co2pricecostatbrev = co2pricecost
    co2pricerevatbrev = co2pricerev
    … store additional results in variables (see subsection below)
    !
    ! Exit subroutine
    Return
End If
```

Step 3C checks for another special case. The CO_2 price is set to -1 BY$/tonne (-0.0529 BY$/Mscf) and the variable *fldernaftxtotpv* is calculated:

- If *fldernaftxtotpv* is positive, a CO_2 price of -1 BY$/tonne, which represents a positive price of CO_2 of 1 BY$/tonne that is paid to the CO_2 EOR operator to accept the CO_2, is the break-even CO_2 price. The algorithm proceeds to Step 3D where the break-even CO_2 price and other variables are stored and the algorithm ends.
- If *fldernaftxtotpv* is negative, a CO_2 price of -1 BY$/tonne is not the break-even price of CO_2 and the algorithm proceeds to Step 3E. In other words, the CO_2 EOR operator must be paid more than 1 BY$/tonne to accept CO_2 and meet their desired minimum return on investment.

The following pseudo code illustrates the process described above:

```
!
```

```
! --- Step 3C: Check to see if a CO2 price equal to -1 BY$/tonne gives a positive
! value for fldernaftxtotpv. If yes, then a CO2 price around -1 BY$/tonne is
! the break-even CO2 price.
ico2pricelow = -1  ! BY$/tonne
co2price = ico2pricelow * dengsurfft3  ! BY$/Mscf
!
! Calculate fldernaftxtotpv
Call CostCalcs
Call FieldEcon
If( fldernaftxtotpv > 0.0 ) Then
        !
        ! --- Step 3D: The break-even CO2 price is around -1 BY$/tonne
        !
        ! Record values for selected variables in base year dollars
        co2pricecostatbrev = co2pricecost
        co2pricerevatbrev = co2pricerev
        … store additional results in variables (see subsection below)
        !
        ! Exit subroutine
        Return
End If
```

Step 3E finds two CO_2 prices, *ico2pricelow* and *ico2pricehigh* that bracket the break-even CO_2 price. At the conclusion of the Step 3E algorithm, the variable *ico2pricelow* will generate a positive value for the variable *fldernaftxtotpv*, while the variable *ico2pricehigh* will generate a negative value for the variable *fldernaftxtotpv*. The break-even CO_2 price will be between these two values or, possibly, equal to *ico2pricelow*. The algorithm starts by setting *ico2pricehigh* to -1 BY$/tonne and then calculates a value for *ico2pricelow* by multiplying *ico2pricehigh* by the variable *mult1* and rounding the result down to the next lowest integer. The variable *fldernaftxtotpv* is then calculated for the variable *ico2pricelow*. If *fldernaftxtotpv* is positive, then acceptable values have been found for *ico2pricelow* and *ico2pricehigh* and the algorithm proceeds to Step 3G. If the variable *fldernaftxtotpv* is negative, then the variable *ico2pricelow* is too high, so the variable *ico2pricehigh* is set to *ico2pricelow* and the process is repeated with the variable *ico2pricelow* decreased until the variable *fldernaftxtotpv* is positive. When a positive value is calculated for *fldernaftxtotpv* then acceptable values have been found for *ico2pricelow* and *ico2pricehigh* and the algorithm proceeds to Step 3G.

A check is made at each iteration to see if the value for *ico2pricelow* gives a negative value for the variable *fldernaftxtotpv* and *ico2pricelow* is below a lower limit stored in the variable *ico2pricelowlim*. If *ico2pricelow* is below this value, then the algorithm proceeds to Step 3F. In this step, the break-even CO_2 price is set to the lower limit (*ico2pricelowlim*). The break-even CO_2 price and other variables are stored and the algorithm ends. The variable *ico2pricelowlim* is currently set to -100 BY$/tonne. This value indicates that a CO_2 EOR operator would have to be paid at least 100 BY$/tonne to accept CO_2 and operate the CO_2 EOR operation profitably. At 100 BY$/tonne, the source of the CO_2 is likely to have other opportunities, such as saline

storage, for disposing of their captured CO_2. Thus, the algorithm halts when the break-even CO_2 price is -100 BY$/tonne or less.

The following pseudo code illustrates the process described above:

```
!
! --- Step 3E: Find a low CO2 price that has a positive value for fldernaftxtotpv
! Initialize ico2pricehigh
ico2pricehigh = -1   ! BY$/tonne
mult1 = 1.5
!
! Set ico2pricelowlim, a lower limit on ico2pricelow
ico2pricelowlim = -100  ! BY$/tonne
!
! Iterate to find ico2pricelow
Do i = 1, 1000
        !
        ! Decrease CO2 price until fldernaftxtotpv is greater than zero
        ico2pricelow = Floor( ico2pricehigh * mult1 )
        !
        ! Check if ico2pricelow is less than CO2 price lower limit (ico2pricelowlim)
        ! If so, set ico2pricelow to ico2pricelowlim (in BY$/tonne)
        ! This is the lowest break-even CO2 price allowed in this analysis.
        If( ico2pricelow < ico2pricelowlim ) Then
                ico2pricelow = ico2pricelowlim
        End If
        co2price = ico2pricelow * dengsurfft3  ! BY$/Mscf
        !
        ! Calculate fldernaftxtotpv
        Call CostCalcs
        Call FieldEcon
        If( fldernaftxtotpv < 0.0 ) Then
                !
                ! The low CO2 price is still too high.
                !
                ! First check to see if ico2pricelow is equal to
                ! or less than CO2 price lower limit (ico2pricelowlim)
                ! If so, set the break-even CO2 price to ico2pricelowlim
                ! (in BY$/tonne)
                If( ico2pricelow <= ico2pricelowlim ) Then
                        !
                        ! --- Step 3F: A CO2 price of ico2pricelowlim gives a negative
                        ! value for fldernaftxtotpv. This CO2 price is a lower
                        ! bound on the break-even CO2 price and results are
                        ! stored for this price.
```

```
                    ico2pricelow = ico2pricelowlim  ! BY$/tonne
                    !
                    ! Calculate fldernaftxtotpv
                    Call CostCalcs
                    Call FieldEcon
                    !
                    ! Record values for selected variables in base year dollars
                    co2pricecostatbrev = co2pricecost
                    co2pricerevatbrev = co2pricerev
                    … store additional results in variables (see subsection below)
                    !
                    ! Exit subroutine
                    Return
                Else
                    !
                    ! Set high CO2 price to low CO2 price and go through
                    ! Do Loop again
                    ico2pricehigh = ico2pricelow
                End If
            Else
                !
                ! The low CO2 price produces a positive value for fldernaftxtotpv.
                ! Appropriate low and high CO2 prices have been found.
                ! Exit the Do Loop and use the bisection method to find the
                ! break-even CO2 price which will be between the low and
                ! high CO2 price.
                Exit
            End If
        End Do
```

In Step 3G, the bisection method is used to find the break-even CO_2 price, which will be a value in between the values stored in *ico2pricelow* and *ico2pricehigh*. The bisection method is an iterative method that hones in on the break-even CO_2 price by adjusting the values for *ico2pricelow* and *ico2pricehigh* until the difference between the values in these two variables is 1. The variables *ico2pricelow* and *ico2pricehigh* are integers, so the smallest difference between the two (without being the same number) is 1. The value for *fldernaftxtotpv* associated with *ico2pricelow* is positive and the value for *fldernaftxtotpv* associated with *ico2pricehigh* is negative. Thus, when the difference between *ico2pricelow* and *ico2pricehigh* is 1, *ico2pricelow* is the highest integer CO_2 price that gives a positive value for *fldernaftxtotpv*.

In the bisection method, a CO_2 price that is halfway between *ico2pricelow* and *ico2pricehigh* is calculated and stored in the variable *ico2pricemid*. The value for *fldernaftxtotpv* associated with *ico2pricemid* is calculated. If *fldernaftxtotpv* is positive, then this CO_2 price is below the break-even CO_2 price, so *ico2pricelow* is set equal to *ico2pricemid*. If *fldernaftxtotpv* is negative, then this CO_2 price is above the break-even CO_2 price, so *ico2pricehigh* is set equal to *ico2pricemid*.

This process is repeated until the difference between the values stored in the variables *ico2pricelow* and *ico2pricehigh* is 1. At this point, the algorithm proceeds to Step 3H where the break-even CO_2 price is set to *ico2pricelow.* The break-even CO_2 price and other variables are stored and the algorithm ends.

The following pseudo code illustrates the process described above:

```
!
! --- Step 3G: Use the bisection method to find the break-even CO2 price.
! The break-even CO2 price is a CO2 price between the low and high CO2
! price where the CO2 price makes fldernaftxtotpv zero. In practice,
! the break-even CO2 price is the lower of two CO2 prices that differ
! by 1 BY$/tonne where the low CO2 price gives a zero or positive value
! for fldernaftxtotpv and the high CO2 price gives a negative value for
! fldernaftxtotpv. The bisection method keeps halving the difference
! between the low and high CO2 prices until the difference is 1 $BY/tonne.
! The low and high CO2 prices generated previously are appropriate
! starting values for the bisection method.
Do i = 1, 1000
        !
        ! Check if the difference between the high and low CO2 price is 1.
        ! If yes, then ico2pricelow is break even CO2 price
        idif = ico2pricehigh - ico2pricelow
        If( idif == 1 ) Then
                Exit
        End If
        !
        ! Calculate a CO2 price between the high and low CO2 prices
        ico2pricemid = Floor( 0.5 * (ico2pricehigh + ico2pricelow) )
        If( ico2pricemid == ico2pricehigh ) Then
                ico2pricemid = ico2pricehigh – 1
        End If
        !
        ! Calculate cash flows for this mid-range CO2 price
        co2price = ico2pricemid * dengsurfft3  ! BY$/Mscf
        !
        ! Calculate fldernaftxtotpv
        Call CostCalcs
        Call FieldEcon
        !
        ! Check for unlikely case that fldernaftxtotpv equals 0
        If( fldernaftxtotpv == 0.0 ) Then
                ico2pricelow = ico2pricemid
                Exit
        End If
```

```
        !
        ! Reset high or low oil price based on fldernaftxtotpv
        If( fldernaftxtotpv < 0.0 ) Then
                ico2pricehigh = ico2pricemid
        Else
                ico2pricelow = ico2pricemid
        End If
End Do
!
! --- Step 3 H: Set break-even CO2 price to the low CO2 price
co2price = ico2pricelow * dengsurfft3  ! BY$/Mscf
!
! Calculate fldernaftxtotpv
Call CostCalcs
Call FieldEcon
!
! Record values for selected variables in base year dollars
co2pricecostatbrev = co2pricecost
co2pricerevatbrev = co2pricerev
… store additional results in variables (see subsection below)
!
! Exit subroutine
Return
```

4.15.2 Definition of Variables Used in Pseudo Code

The variables used in the pseudo code are defined below:

co2price – CO_2 price used in Fortran subroutines CostCalcs and FieldEcon; *co2price* can be positive (the CO_2 EOR operation pays for CO_2) or negative (the CO_2 EOR operation is paid to accept CO_2) (BY$/Mscf)

ico2pricelow – estimate of CO_2 price that is an integer value and is below the break-even CO_2 price (i.e., the value stored in *ico2pricelow* generates a value for *fldernaftxtotpv* that is positive) (BY$/tonne)

ico2pricehigh – estimate of CO_2 price that is an integer value and is above the break-even CO_2 price (i.e., the value stored in *ico2pricehigh* generates a value for *fldernaftxtotpv* that is negative) (BY$/tonne)

fldernaftxtotpv – total earnings after federal income taxes in present value dollars for all CO_2 EOR project years (M$)

co2pricecostatbrev – break-even CO_2 price when CO_2 is a cost to the CO_2 EOR operation (the CO_2 EOR operation pays for CO_2). In this case, *co2pricecostatbrev* is the highest positive integer that gives a value for *fldernaftxtotpv* that is positive. If CO_2 is a

source of revenue for the CO_2 EOR operation (the CO_2 EOR operation is paid to accept CO_2) then *co2pricecostatbrev* is zero (BY$/Mscf)

co2pricerevatbrev – break-even CO_2 price when CO_2 is a source of revenue for the CO_2 EOR operation (the CO_2 EOR operation is paid to accept CO_2). In this case, *co2pricerevatbrev* is the lowest positive integer that gives a value for *fldernaftxtotpv* that is positive. If CO_2 is a cost to the CO_2 EOR operation (the CO_2 EOR operation pays for CO_2) then *co2pricerevatbrev* is 0 (BY$/Mscf)

mult1 – factor used to increase or decrease oil price

i – integer index in iteration Do Loop used to limit number of iterations so an infinite loop does not occur

ico2priceuplim – integer value that is the highest CO_2 price allowed in the analysis (BY$/tonne)

ico2pricelowlim – integer value that is the lowest CO_2 price allowed in the analysis (BY$/tonne)

idif – integer variable that is difference between *ico2pricehigh* and *ico2pricelow*. When *idif* equals 1, the iterations are halted and the break-even CO_2 price is set equal to *ioilpricelow* (BY$/tonne)

ico2pricemid – estimate of CO_2 price that is halfway between *ico2pricelow* and *ico2pricehigh* (BY$/STB)

4.15.3 Additional Variables Calculated and Stored by Algorithms

In addition to the break-even CO_2 price, a number of other variables are also stored. These variables are

oilpriceatbrev – market oil price (BY$/STB)

gaoilpriceatbrev – gravity adjusted oil price (BY$/STB)

loilpriceatbrev – lease oil price (BY$/STB)

dcwuncapbyatbrev – unit capital costs for drilling and completing a well in base year dollars (BY$/well)

recyco2atbrev – unit costs to operate and maintain the CO_2 recycling plant (BY$/Mscf or MBY$/MMscf)

liftunopbyatbrev – unit operating cost for lifting produced water and oil (BY$/STB or MBY$/MSTB)

RORbtbrev – estimated IRR before federal income taxes

ipayoutbt – year in CO_2 EOR project time frame when cumulative nominal earnings before federal income taxes become positive. If *ipayoutbt* equals 1000, this indicates that the cumulative nominal earnings before federal income taxes never become positive

RORatbrev – estimated IRR after federal income taxes

ipayoutat – year in CO_2 EOR project time frame when cumulative nominal earnings after federal income taxes become positive. If *ipayoutat* equals 1000, this indicates that the cumulative nominal earnings after federal income taxes never become positive

At the break-even CO_2 price, the estimated IRR after federal income taxes (*RORatbrev*) should be very close to the discount rate (*disc*) which is typically the WACC (*wacc*).

5 INPUTS TO THE FE/NETL ONSHORE CO_2 EOR COST MODEL

This section briefly describes the input files that are used to run the FE/NETL Onshore CO_2 EOR Cost Model. There are three input files:

- "ystrmtbflow_outyr.csv" is a text file generated by the Fortran program StrmtbFlow, which is part of the FE/NETL CO_2 Prophet Model. StrmtbFlow is a stream tube reservoir simulation program. The contents of the file "ystrmtbflow_outyr.csv" are described in Section 4.4.1 in considerable detail. Additionally, this file is described in the "User's Manual for StrmtbFlow."
- "Cost_in.dat" is a text file that contains values for many of the variables used to calculate capital and O&M costs. It is anticipated that these values will not differ significantly from oil field to oil field so that the variables can be specified once by the user and then applied to many oil fields with no modification.
- "Gen_in.dat" is a text file that contains values for variables that may be oil field specific. Compared to the values input for variables in the "Cost_in.dat" file, the user is more likely to modify values for variables in this file to either reflect differences between oil fields or, more likely, to examine the influence of changing specific variables on key outputs, such as the break-even price of oil.

When the FE/NETL Onshore CO_2 EOR Cost Model reads the contents of these files, the program ignores any line that begins with the letters CC. Any line beginning with CC is a comment line that is used to provide explanations of the input variables in the files. The letters CC can be upper or lower case (i.e., CC, cc, cC or Cc) but they must occur in the first two spaces of a line to be treated by the FE/NETL Onshore CO_2 EOR Cost Model as a comment line.

The "Cost_in.dat" and "Gen_in.dat" files that come with the Fortran files that are distributed with the FE/NETL Onshore CO_2 EOR Cost Model are heavily commented. These comments should provide enough information for the user to understand the input variables in these files and how to modify the variables. Thus, the user should read the comments in these files to understand the input variables. The comments indicate where in this user's manual (i.e., in Section 4) these variables are discussed or used in equations.

There are two sets of inputs and one specific input variable in the "Gen_in.dat" file that have a significant effect on the output generated by the FE/NETL Onshore CO_2 EOR Cost Model. The user specifies a base market oil price and up to 10 additional market oil prices. The user also specifies a base CO_2 price and up to 10 additional CO_2 prices. The FE/NETL Onshore CO_2 EOR Cost Model generates a number of cases for different combinations of oil prices and CO_2 prices. Some of the cases involve calculating break-even oil prices or break-even CO_2 prices. Other cases involve calculating the project economics for an oil price and a CO_2 price that are specified by the user. The variable *casecon*, which is a user input, controls which cases are generated by the model with results for each case reported in the output file. Section 6.1 describes the variable *casecon* and the cases that are generated and reported in more detail.

It is recommended that the user create backup copies of the original versions of "Cost_in.dat" and "Gen_in.dat" files before making any modifications to the input variables in these files.

Many of the input variables in the "Cost_in.dat" are coefficients that were developed from fitting a function to data. Thus, care should be taken before these variables are modified. In contrast, many of the variables in the "Gen_in.dat" file, such as the market price of oil, CO_2 cost, and financial variables are intended to be modified to determine how changing these variables affects the financial viability of a CO_2 EOR project.

The FE/NETL Onshore CO_2 EOR Cost Model does very little checking to make sure the values specified by the user for input variables are appropriate. If the user specifies inappropriate input values (such as text values where a number is expected), then the program is likely to unceremoniously crash. Thus, the user must carefully review the values they specify for input variables to ensure the values are appropriate.

6 OUTPUTS FROM THE FE/NETL ONSHORE CO_2 EOR COST MODEL

The FE/NETL Onshore CO_2 EOR Cost Model generates one output file, "CO2EORCMOn_det_out.csv". This file is a very large text file in csv format. The output is divided into two sections. The first section provides a summary of the results generated by the FE/NETL CO_2 EOR Cost Model. The second section provides detailed output for the base oil price and the base CO_2 price input by the user.

Each section begins with a comment section with text beginning with the "CC" comment designation. The comment section describes the output presented in that section. The actual output comprises values for specific variables in the FE/NETL Onshore CO_2 EOR Cost Model. The comment section provides definitions or descriptions for each of the output variables. At the end of the comment section, the name of each output variable is presented along with value for the variable and the units for the variable. The purpose of the comment section is to include in the file the documentation needed for the user to understand the output variables.

As noted above, the output file is a csv file intended to be opened directly in Excel. Any text or numbers between commas is placed within a single cell of Excel. In the text within comments, no commas are used so that the comments occur within a single cell within Excel (the first cell in a row). If a comma is needed within the comment text, a semicolon (;) is used instead within the comment text.

Because the documentation for the output for the FE/NETL Onshore CO_2 EOR Cost Model is contained within the output file, this section of the user's manual describes the structure of the output file. In particular, the sections within the output file are described. However, a detailed description of the variables output within each section of the output file is not provided since the comments in the output file describe each variable.

6.1 OUTPUT SECTION 1: SUMMARY OF EVALUATION

This section provides a summary of the evaluation with output in subsections A through H. Some of the subsection headings below have semicolons in the text because semicolons replace commas in the comment text within the output file as discussed above.

A. Oil field identifiers

B. Oil field characteristics

C. Pattern; field development and well information

D. Oil field fluid volumes and flows

E. Oil field performance measures

F. CO_2 saturation and storage coefficients for oil field

G. Financial parameters

H. Revenues; costs and return on investment for different oil and CO_2 prices (cases)

The output in the last section deserves some explanation. In the input file "Gen_in.dat", the user supplies a base case oil price and up to 10 additional oil prices to evaluate. In this file, the user also supplies a base case CO_2 price and up to 10 additional CO_2 prices. The total number of oil prices is stored in the variable *noilprres* and the total number of CO_2 prices is stored in the variable *nco2prres*. These variables have a minimum value of 1 if only the base case prices are provided and a maximum value of 11.

The FE/NETL Onshore CO_2 EOR Cost Model uses the oil prices and CO_2 prices to generate cases. There are four types of cases in the model:

- Case type 1: Break-even oil price for a user specified CO_2 price
- Case type 2: Break-even CO_2 price for a user specified oil price
- Case type 3: Project economics for base case oil price and base case CO_2 price
- Case type 4: Project economics for an oil price specified by the user and a CO_2 price specified by the user. This can be the base case oil price as long as the CO_2 price is not the base case CO_2 price. Case type 4 includes all combinations of oil prices and CO_2 prices except the base case oil price and base case CO_2 price since that is case type 3

Depending on the number of oil prices and CO_2 prices input by the user, a very large number of cases can be generated. If the user inputs the maximum of 11 oil prices and 11 CO_2 prices, then the model can calculate 11 break-even oil prices (one for each CO_2 price) and 11 break-even CO_2 prices (one for each oil price). The model can calculate up to 121 additional cases where the project economics are calculated for different combinations of oil prices and CO_2 prices. This is a total of 133 possible cases.

Since the user may only desire results for specific case types, the user can control the case types that are generated and reported by selecting an appropriate value for the variable *casecon*. The variable *casecon* is an integer variable that can take on values from 1 to 4:

- *casecon* = 1 for case types 1, 2, and 3 to be evaluated and reported. This means break-even oil prices, break-even CO_2 prices, and project economics for the base case oil price and base case CO_2 price are evaluated and reported.
- *casecon* = 2 for case types 1 and 3 to be evaluated and reported. This means break-even oil prices and project economics for the base case oil price and base case CO_2 price are evaluated and reported.
- *casecon* = 3 for case types 2 and 3 to be evaluated and reported. This means break-even CO_2 prices and project economics for the base case oil price and base case CO_2 price are evaluated and reported.
- *casecon* = 4 for all case types to be evaluated and reported. This means break-even oil prices, break-even CO_2 prices and project economics for all combinations of oil prices and CO_2 prices are evaluated and reported.

If the user specifies a value for *casecon* that is less than 1 or greater than 4, *casecon* is set to the default value of 1.

In Section H, the cases are reported by case type. If break-even oil prices are requested by the user (*casecon* is 1, 2, or 4), then break-even oil prices (case type 1) are reported for all CO_2 prices (i.e., a total of *nco2prres* break-even oil prices). The break-even oil price is the minimum oil price (rounded up to the nearest dollar per STB) where the IRR after federal income taxes (*RORat*) equals the discount rate (*disc*). The discount rate is usually equal to the WACC (*wacc*). Since the break-even oil price is rounded up, the variable *RORat* may be slightly higher than *disc*. The variable *fldernaftxtotpv*, which is the total earnings after federal income taxes in discounted or present value dollars, is equal to zero at the break-even price of oil. Since the break-even price of oil is rounded up, this variable will be greater than zero. However, since this number is expressed in millions of present value dollars, it is often still a large number, just not as large as it would be if the oil price were much higher than the break-even oil price.

If break-even CO_2 prices are requested by the user (*casecon* is 1, 3, or 4), then break-even CO_2 prices (case type 2) are reported for all oil prices (i.e., a total of *noilprres* break-even CO_2 prices). The break-even CO_2 price (given by the variable *co2price*) is the highest CO_2 price (rounded down to the nearest dollar per tonne) where the IRR after federal income taxes equals the discount rate. The CO_2 price variable *co2price* can be positive or negative, where a positive CO_2 price indicates CO_2 is a cost to the CO_2 EOR operation (i.e., the variable *co2pricecost* is positive and the variable *co2pricerev* is zero) and a negative CO_2 price indicates the CO_2 EOR operation is paid to accept CO_2 (i.e., the variable *co2pricecost* is zero and the variable *co2pricerev* is positive). The variable fldernaftxtotpv is equal to zero at the break-even price of CO_2. Since the break-even price of CO_2 is rounded down, this variable will be greater than zero, but not as large as it would be if the CO_2 price were much lower than the break-even CO_2 price.

The project economics are then reported for case type 3, the base case oil price and base case CO_2 price. Case type 3 is reported for all values for the variable *casecon*.

If requested by the user (*casecon* equals 4), then the project economics are reported for case type 4, which are all combinations of oil prices and CO_2 prices with one exception. The case where the oil price is the base case oil price and the CO_2 price is the base case CO_2 price is not reported because this is case type 3.

While Output Section 1 is a summary of the output from the FE/NETL Onshore CO_2 EOR Cost Model, it is not particularly short. It can have between 300 and 500 lines of output depending on the number of cases that are evaluated and reported. The lines of output are mostly comment lines that define the variables that have output values.

6.2 Output Section 2: Detailed Results for Base Case

This section provides the details of the project economics for the base case oil price and the base case CO_2 price. The break-even oil price is provided for the base case CO_2 price.

This section is quite long with approximately 1750 lines of output. This section is divided into six parts plus a last part (labeled Part 99) that provides timing information for the run. The parts and their sub-parts are listed below. Some of the headings below have semicolons in the text

because semicolons replace commas in the comment text within the output file as discussed above.

Part 1: Run info and oil field info and day and time

Part 2: Basic data about oil field

- 2.1 Geologic characteristics
- 2.2 Fluid saturations
- 2.3 Temperatures and pressures
- 2.4 Oil properties
- 2.5 Hydrocarbon gas properties
- 2.6 Brine properties
- 2.7 Pure water properties
- 2.8 CO_2 properties
- 2.9 Pattern properties

Part 3: Pattern and well information

- 3.1 Oil field development information
- 3.2 Pattern information
- 3.3 Well information

Part 4: Injection and production volumes and masses

- 4.1 Pattern fluid flows; fluid saturations and CO_2 volumes
- 4.2 Oil field fluid flows
- 4.3 CO_2 volumetric flows for the oil field (assumes emissions to atmosphere are nonzero)
- 4.4 CO_2 mass flows for the oil field (assumes emissions to atmosphere are nonzero)
- 4.5 CO_2 volumetric flows for the oil field (assumes emissions to atmosphere are zero)
- 4.6 CO_2 mass flows for the oil field (assumes emissions to atmosphere are zero)
- 4.7 Water volumetric flows for the oil field (assumes fugitive emissions are nonzero)
- 4.8 Performance measures for oil field
- 4.9 CO_2 saturations and storage coefficients for oil field
- 4.10 Cumulative hydrocarbon pore volumes (HCPV) of water and CO_2 injected and oil; water and CO_2 produced for pattern and field

7 REFERENCES

[1] D. Morgan, D. Remson and T. McGuire, "Conceptual and Mathematical Foundation for the FE/NETL CO2 Prophet Model for Simulating CO2 Enhanced Oil Recovery, Version 2," U.S. Department of Energy, National Energy Technology Laboratory, DOE/NETL-2020/2630, Pittsburgh, PA, United States, 2020.

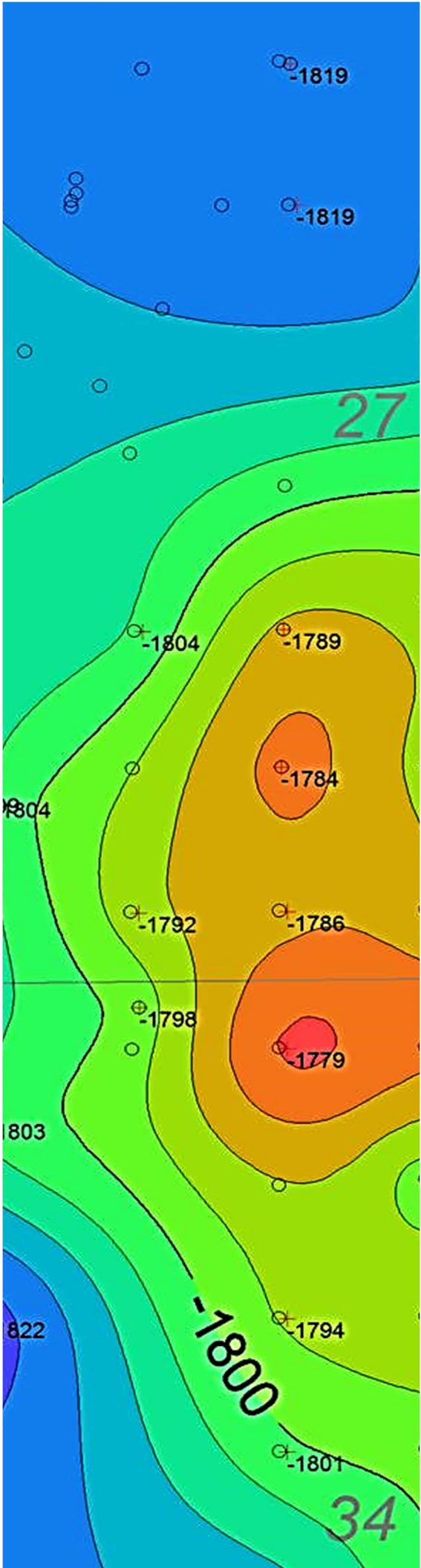
[2] D. Morgan, D. Remson and T. McGuire, "User's Manual for StrmtbFlow, the Stream Tube Multiphase Flow Part of the FE/NETL CO2 Prophet Model, Version 2," U.S. Department of Energy, National Energy Technology Laboratory, DOE/NETL-2020/2631, Pittsburgh, PA, United States, 2020.

[3] Internal Revenue Service, "§45Q. Credit for carbon oxide sequestration," [Online]. Available: https://uscode.house.gov/view.xhtml?req=(title:26%20section:45Q%20edition:prelim). [Accessed 21 February 2020].

[4] American Petroleum Institute, "Joint Association Survey on 2014 Drilling Costs," IHS under license with API, Washington, DC, United States, 2015.

[5] Internal Revenue Service, "How to Depreciate Property, Publication 946," Internal Revenue Service, U.S. Department of Treasury, Washington, DC, 2016.

[6] National Energy Technology Laboratory, "FE/NETL CO2 Saline Storage Cost Model," U.S. Department of Energy, National Energy Technology Laboratory, Pittsburgh, 2019.



NATIONAL ENERGY TECHNOLOGY LABORATORY

www.netl.doe.gov

Albany, OR • Anchorage, AK • Morgantown, WV • Pittsburgh, PA • Sugar Land, TX

(800) 553-7681