Dalton Gilpatrick

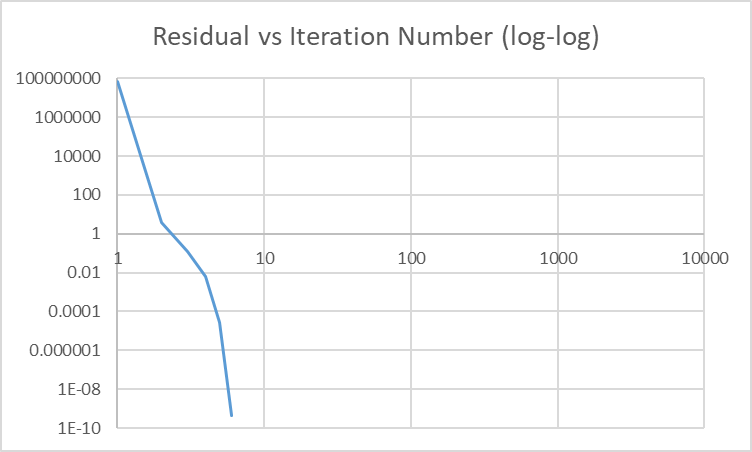
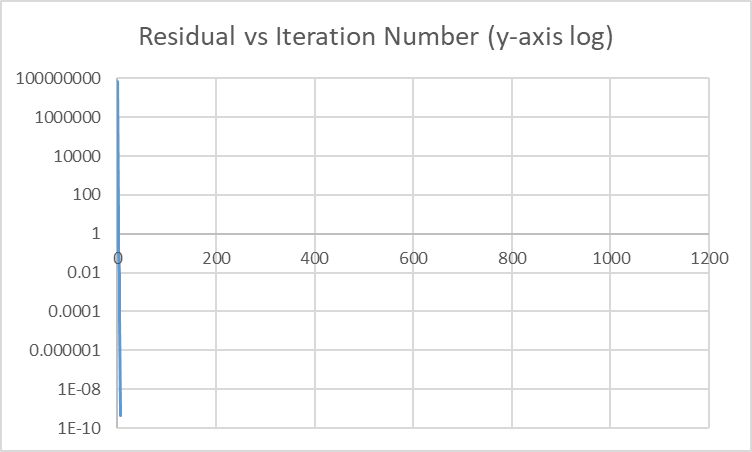
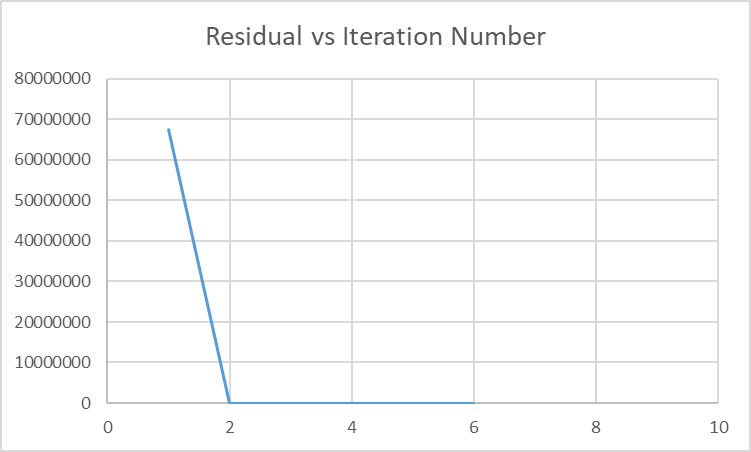
10/20/2017

Comp. Methods: Project #1

This project was to build a material balance simulator that takes inputs such as pressure, compressibilities, saturations, productions, formation volume, and formation volume factors at an initial time, and outputs pressure and saturations at a later time after production has taken place. This was accomplished by using Newton’s method and Jacobian matrices. I coded this project in Microsoft Excel’s VBA language.

Convergence typically happened very rapidly, provided a good initial guess was provided and a reasonable tolerance. I usually just used the initial conditions as the guess, which produced quick results. The tolerance level, if set with too high a precision, would cause the program to run through all of its iterations once it had reached its minimum continuous residual result. The following are various charts designed to show the convergence given different conditions.

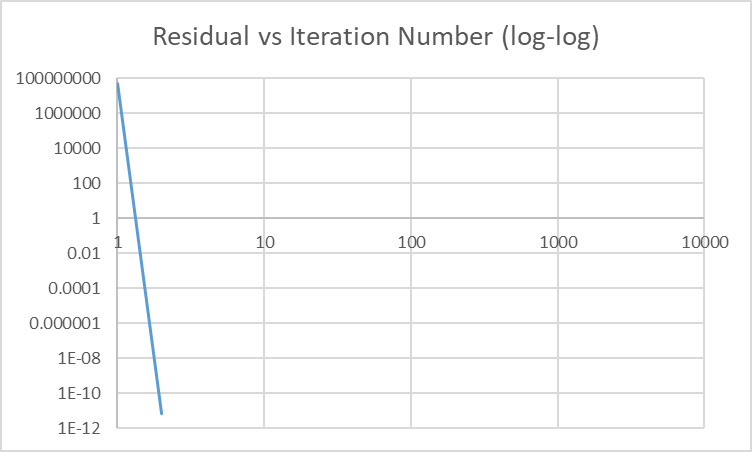
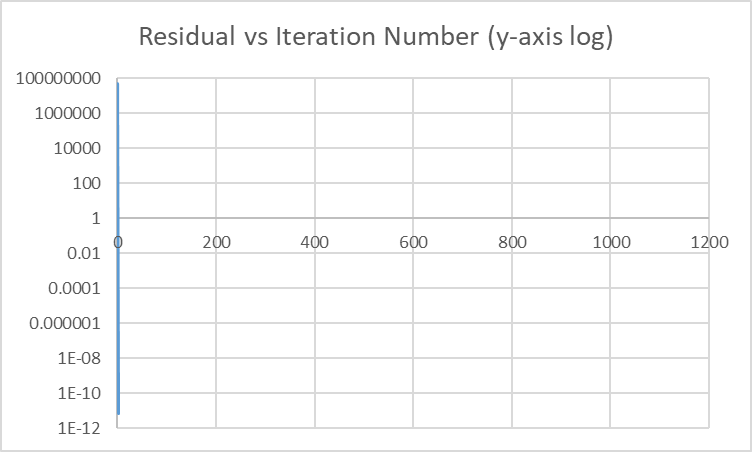
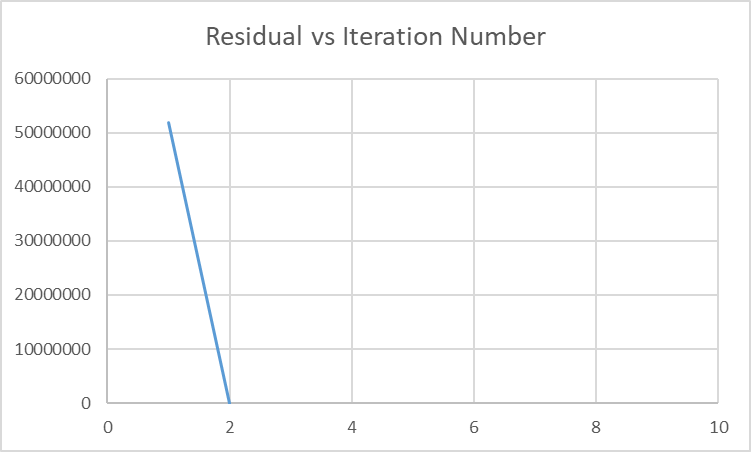
Typical chart set if the result converges:



This shows that the method usually converges very quickly. For most circumstances, as long as a vaguely reasonable guess was given, the program converged.

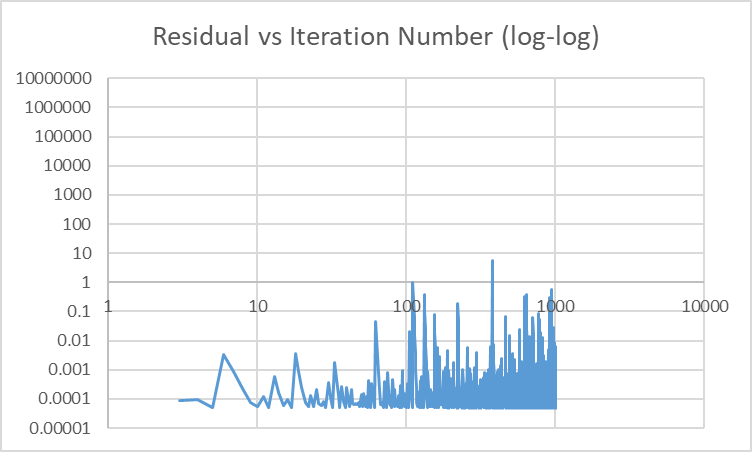
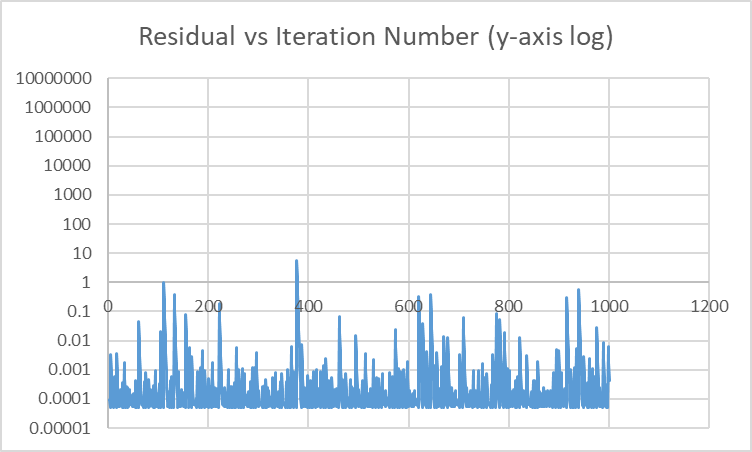
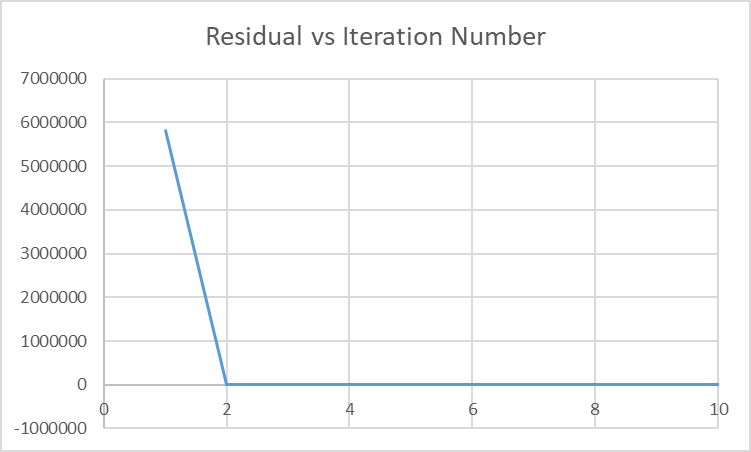
The prior results were for compressible cases. Assuming that water, oil, and rock were incompressible, the program still converged very rapidly.

Graphs showing the convergence with incompressible rock, oil, and water:



An interesting result happens when withdrawing almost all of what a reservoir has to offer. It is an extreme example, but it warrants attention. The program seems to be unable to arrive at a definitive conclusion where the residual would be below the tolerance threshold.

Graphs associated with extracting very large amounts of fluids:



Overall, this programming exercise was very difficult, required extensive amounts of time and research to complete, but was satisfying to see work in the end. I believe that I have a better knowledge of how these types of programs now than I did before the project.

My VBA Code:

Option Explicit

Sub Material\_Balance()

Const refcell = "C2"

'Volume is in ft^3, Qo in stb, Qg in scf, Qw in stb

Dim p1 As Double, sw1 As Double, so1 As Double, sg1 As Double, volume As Double

Dim cr As Double, co As Double, cw As Double, cg As Double

Dim Boref As Double, Bwref As Double, Bgref As Double

Dim qo As Double, qg As Double, qw As Double

Dim phi1 As Double, bg1 As Double, bo1 As Double, bw1 As Double

Dim yg1 As Double, yo1 As Double, yw1 As Double

volume = Range(refcell).Cells(1, 1).Value

p1 = Range(refcell).Cells(2, 1).Value

sw1 = Range(refcell).Cells(3, 1).Value

so1 = Range(refcell).Cells(4, 1).Value

sg1 = 1 - (so1 + sw1)

cr = Range(refcell).Cells(6, 1).Value

co = Range(refcell).Cells(7, 1).Value

cg = Range(refcell).Cells(8, 1).Value

cw = Range(refcell).Cells(9, 1).Value

Boref = Range(refcell).Cells(10, 1).Value

Bwref = Range(refcell).Cells(11, 1).Value

Bgref = Range(refcell).Cells(12, 1).Value

qo = Range(refcell).Cells(1, 3).Value \* -1

qg = Range(refcell).Cells(2, 3).Value \* -1

qw = Range(refcell).Cells(3, 3).Value \* -1

phi1 = 0.25 \* Exp(cr \* (p1 - 3000))

yg1 = cg \* (p1 - 3000)

yo1 = co \* (p1 - 3000)

yw1 = cw \* (p1 - 3000)

bg1 = Bgref / (1 + (yg1) + (0.5 \* (yg1 ^ 2)))

bo1 = Boref / (1 + (yo1) + (0.5 \* (yo1 ^ 2)))

bw1 = Bwref / (1 + (yw1) + (0.5 \* (yw1 ^ 2)))

Dim b1 As Double, b2 As Double, b3 As Double

b1 = qo + ((volume \* so1 \* phi1) / (bo1))

b2 = qw + ((volume \* sw1 \* phi1) / (bw1))

b3 = qg + ((volume \* sg1 \* phi1) / (bg1))

'"x" part of the matrix equation

Dim p2 As Double, so2 As Double, sw2 As Double

Dim p3 As Double, so3 As Double, sw3 As Double

Dim x() As Variant

ReDim x(1 To 3, 1 To 1)

p2 = Range(refcell).Cells(4, 3).Value

so2 = Range(refcell).Cells(5, 3).Value

sw2 = Range(refcell).Cells(6, 3).Value

'Newton Update part

Dim tol As Double

Dim fo As Double, fw As Double, fg As Double

Dim fosop As Double, foswp As Double, fopresp As Double

Dim fwsop As Double, fwswp As Double, fwpresp As Double

Dim fgsop As Double, fgswp As Double, fgpresp As Double

Dim phi2 As Double, bg2 As Double, bo2 As Double, bw2 As Double

Dim yg2 As Double, yo2 As Double, yw2 As Double

Dim phi2p As Double, bg2p As Double, bo2p As Double, bw2p As Double

tol = Range(refcell).Cells(7, 3).Value

phi2 = 0.25 \* Exp(cr \* (p2 - 3000))

yg2 = cg \* (p2 - 3000)

yo2 = co \* (p2 - 3000)

yw2 = cw \* (p2 - 3000)

bg2 = Bgref / (1 + yg2 + (0.5 \* (yg2 ^ 2)))

bo2 = Boref / (1 + yo2 + (0.5 \* (yo2 ^ 2)))

bw2 = Bwref / (1 + yw2 + (0.5 \* (yw2 ^ 2)))

phi2p = 0.25 \* cr \* Exp(cr \* (p2 - 3000))

bg2p = (-Bgref \* (cg + ((yg2) \* cg))) / ((1 + yg2 + (0.5 \* (yg2 ^ 2))) ^ 2)

bo2p = (-Boref \* (co + ((yo2) \* co))) / ((1 + yo2 + (0.5 \* (yo2 ^ 2))) ^ 2)

bw2p = (-Bwref \* (cw + ((yw2) \* cw))) / ((1 + yw2 + (0.5 \* (yw2 ^ 2))) ^ 2)

fo = ((phi2 \* volume \* so2) / (bo2)) - b1

fw = ((phi2 \* volume \* sw2) / (bw2)) - b2

fg = ((phi2 \* volume \* (1 - (so2 + sw2))) / (bg2)) - b3

'Jacobian Matrix Elements

'Row 1

fosop = (volume \* phi2) / bo2

foswp = 0

fopresp = volume \* (so2) \* (((bo2 \* phi2p) - (phi2 \* bo2p)) / (bo2 ^ 2))

'Row 2

fwsop = 0

fwswp = (volume \* phi2) / bw2

fwpresp = volume \* (sw2) \* (((bw2 \* phi2p) - (phi2 \* bw2p)) / (bw2 ^ 2))

'Row 3

fgsop = -((volume \* phi2) / bg2)

fgswp = -((volume \* phi2) / bg2)

fgpresp = volume \* (1 - (so2 + sw2)) \* (((bg2 \* phi2p) - (phi2 \* bg2p)) / (bg2 ^ 2))

'Creating the Matrix itself

Dim J() As Variant

ReDim J(1 To 3, 1 To 3)

J(1, 1) = fosop

J(1, 2) = foswp

J(1, 3) = fopresp

J(2, 1) = fwsop

J(2, 2) = fwswp

J(2, 3) = fwpresp

J(3, 1) = fgsop

J(3, 2) = fgswp

J(3, 3) = fgpresp

Dim jinv() As Variant

ReDim jinv(1 To 3, 1 To 3)

jinv = Application.WorksheetFunction.MInverse(J)

Dim F() As Variant

ReDim F(1 To 3, 1 To 1)

F(1, 1) = fo

F(2, 1) = fw

F(3, 1) = fg

'Iterative Part

Dim count As Double, iterations As Double, i As Double

count = 0

iterations = 1000

For i = 1 To iterations

Range("z1").Cells(i, 1).Value = Null

Range("z1").Cells(i, 2).Value = Null

Next i

Do While Abs(Application.WorksheetFunction.Max(fo, fw, fg)) >= tol

If count > iterations Then

MsgBox ("The maximum number of iterations was exceeded.")

Exit Do

End If

phi2 = 0.25 \* Exp(cr \* (p2 - 3000))

yg2 = cg \* (p2 - 3000)

yo2 = co \* (p2 - 3000)

yw2 = cw \* (p2 - 3000)

bg2 = Bgref / (1 + yg2 + (0.5 \* (yg2 ^ 2)))

bo2 = Boref / (1 + yo2 + (0.5 \* (yo2 ^ 2)))

bw2 = Bwref / (1 + yw2 + (0.5 \* (yw2 ^ 2)))

phi2p = 0.25 \* cr \* Exp(cr \* (p2 - 3000))

bg2p = (-Bgref \* (cg + ((yg2) \* cg))) / ((1 + yg2 + (0.5 \* (yg2 ^ 2))) ^ 2)

bo2p = (-Boref \* (co + ((yo2) \* co))) / ((1 + yo2 + (0.5 \* (yo2 ^ 2))) ^ 2)

bw2p = (-Bwref \* (cw + ((yw2) \* cw))) / ((1 + yw2 + (0.5 \* (yw2 ^ 2))) ^ 2)

fo = -(((phi2 \* volume \* so2) / (bo2)) - b1)

fw = -(((phi2 \* volume \* sw2) / (bw2)) - b2)

fg = -(((phi2 \* volume \* (1 - (so2 + sw2))) / (bg2)) - b3)

fosop = (volume \* phi2) / bo2

foswp = 0

fopresp = volume \* (so2) \* (((bo2 \* phi2p) - (phi2 \* bo2p)) / (bo2 ^ 2))

fwswp = (volume \* phi2) / bw2

fwsop = 0

fwpresp = volume \* (sw2) \* (((bw2 \* phi2p) - (phi2 \* bw2p)) / (bw2 ^ 2))

fgsop = -((volume \* phi2) / bg2)

fgswp = -((volume \* phi2) / bg2)

fgpresp = volume \* (1 - (so2 + sw2)) \* (((bg2 \* phi2p) - (phi2 \* bg2p)) / (bg2 ^ 2))

J(1, 1) = fosop

J(1, 2) = foswp

J(1, 3) = fopresp

J(2, 1) = fwsop

J(2, 2) = fwswp

J(2, 3) = fwpresp

J(3, 1) = fgsop

J(3, 2) = fgswp

J(3, 3) = fgpresp

jinv = Application.WorksheetFunction.MInverse(J)

F(1, 1) = fo

F(2, 1) = fw

F(3, 1) = fg

x = Application.WorksheetFunction.MMult(jinv, F)

so2 = so2 + x(1, 1)

sw2 = sw2 + x(2, 1)

p2 = p2 + x(3, 1)

count = count + 1

Range("z1").Cells(count, 1).Value = count

Range("z1").Cells(count, 2).Value = Application.WorksheetFunction.Max(fo, fw, fg)

Loop

Range(refcell).Cells(9, 3).Value = p2

Range(refcell).Cells(10, 3).Value = sw2

Range(refcell).Cells(11, 3).Value = so2

End Sub