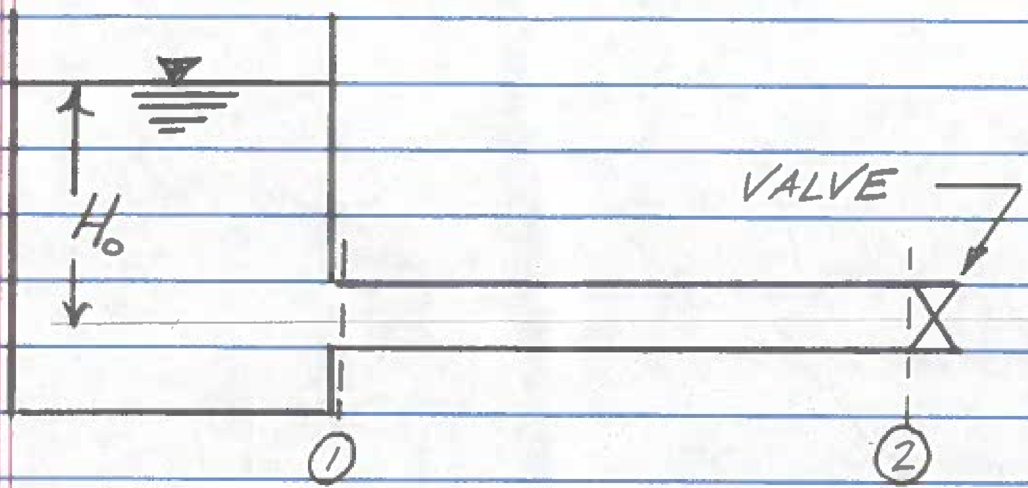
RIGID WATER COLUMN THEORY

VALVE CLOSURE OVER FINITE TIME



PIPELINE AS SHOWN, VALVE IS OPEN

THEN VALVE IS CLOSED QUICKLY (BUT NOT INSTANTLY) AND LIQUID BEGINS TO DECELERATE:

STARTING WITH EULER'S EQUATION

$$-\frac{1}{\gamma}\frac{\partial p}{\partial s} - \frac{\partial z}{\partial s} - \frac{fV^2}{2gD} = \frac{1}{g}\frac{dV}{dt}$$

INTEGRATE FROM 1 TO 2 TO OBTAIN

$$H_o - \frac{p_2}{\gamma} - \frac{fL}{2gD}V^2 = \frac{L}{g}\frac{dV}{dt}$$

NOW CONSIDER ENERGY ACROSS THE VALVE

$$\frac{p_2}{\gamma} = K_L \frac{V^2}{2g}$$

SUBSTITUTE INTO EQUATION OF MOTION

$$H_o - \left(K_L + \frac{fL}{D}\right)\frac{V^2}{2g} = \frac{L}{g}\frac{dV}{dt}$$

K_L IS A FUNCTION OF VALVE POSITION, SO ITS SCHEDULE WOULD NEED TO BE SPECIFIED. A SIMPLE FINITE-DIFFERENCE APPROACH IS

$$V(t+\Delta t) = V(t) + \frac{g\Delta t}{L}\left[H_o - \left(K_L(t) + \frac{f(t)L}{D}\right)\frac{V^2(t)}{2g}\right]$$

THERE WOULD BE PRACTICAL LIMITS OF APPLICABILITY, IF $K_L(t) \rightarrow$ BIG, FAST ; $\frac{p_2}{\gamma} \rightarrow$ BIG, FAST ;

$\frac{dV}{dt} \rightarrow$ BIG, FAST

R SCRIPT TO EXPLORE A SCHEDULED VALVE CLOSURE.

$K(t) = kt$ (WE WILL MAKE k AN ADJUSTIBLE PARAMETER)

INPUTS: $H_0, L, D, g, f, k, \Delta t$

OUTPUTS: $V_0, V(t), \frac{p_2}{\gamma}(t)$

ATTACHED CODE & EXAMPLE FOR

$K(t) = 9t$

NOTES: THE INTEGRATION "METHOD" IS CALLED "EULER'S" METHOD – NICE COINCIDENCE.

THE NUMERICAL METHOD IS SENSITIVE TO CHOICE OF k AND Δt

EXPERIMENTALLY $\frac{\Delta t}{k} = \frac{1}{2}$ PRODUCED STABLE SOLUTIONS – THE RELATIONSHIP IS PROBABLY SOME KIND OF COURANT NUMBER (BUT DON'T KNOW YET!)

```
# Finite-difference approximation for scheduled valve closure
########### Prototype Functions ############
Vsteady <- function(head,distance,diameter,gravity,friction){
  Vsteady <- sqrt((2.0*gravity*diameter*head)/(friction*distance))
  return(Vsteady)
}
######
Head2 <- function(valve_loss_coefficient,gravity,velocity){
  Head2 <- valve_loss_coefficient*((velocity^2)/(2*gravity))
  return(Head2)
}
######
steadyVelocity <- function(head,distance,diameter,gravity,friction){
  steadyVelocity <- sqrt((2.0*gravity*diameter*head)/(friction*distance))
  return(steadyVelocity)
}
#######
dVdt <-
 function(head,distance,diameter,gravity,friction,Vnow,deltat,valve_loss_coeffi
 cient){
  dVdt <- (valve_loss_coefficient+(friction*distance)/(diameter))
  dVdt <- dVdt * ((Vnow^2)/(2*gravity))
  dVdt <- head - dVdt
  dVdt <- ((gravity*deltat)/(distance))*dVdt
  return(dVdt)
}
##################################################
# Read Inputs
Ho <- as.numeric(readline(prompt = "Enter Reservoir Pool Elevation "))
L  <- as.numeric(readline(prompt = "         Enter Pipeline Length "))
D  <- as.numeric(readline(prompt = "           Enter Pipe Diameter "))
g  <- as.numeric(readline(prompt = "  Enter gravitational constant "))
f  <- as.numeric(readline(prompt = "   Enter darcy friction factor "))
k  <- as.numeric(readline(prompt = "   Intrinsic Valve Coefficient "))
dt  <- as.numeric(readline(prompt = "      Computational Time Step "))
HowManyTimeSteps  <- as.numeric(readline(prompt = "         How Many Time
 Steps "))
# Echo Inputs
message("   Reservoir Pool Elevation : ",Ho)
message("            Pipeline Length : ",L)
message("              Pipe Diameter : ",D)
message("     Gravitational Constant : ",g)
message("      Darcy Friction Factor : ",f)
message("Intrinsic Valve Coefficient : ",k)
message("    Computational Time Step : ",dt)
message("        How Many Time Steps : ",HowManyTimeSteps)
# Compute Some Constants
Vzero <- steadyVelocity(Ho,L,D,g,f)
# Report Some Constants
message("                         Vo : ",Vzero)
elapsed_time <- numeric(length = (HowManyTimeSteps+1))
```

```
valve_loss_coefficient <- numeric(length = (HowManyTimeSteps+1))
VofT <- numeric(length = (HowManyTimeSteps+1))
Pat2 <- numeric(length = (HowManyTimeSteps+1))
# Set Initial Values
elapsed_time[1] <- 0
valve_loss_coefficient[1] <- 0
VofT[1] <- Vzero
Pat2[1] <- Head2(valve_loss_coefficient[1],g,VofT[1])
# Begin Time-Stepping
for (i in 2:(HowManyTimeSteps+1)){
  elapsed_time[i] <- elapsed_time[i-1] + dt
  valve_loss_coefficient[i] <- k*elapsed_time[i]
  VofT[i] <- VofT[i-1] +
   dVdt(Ho,L,D,g,f,VofT[i-1],dt,valve_loss_coefficient[i])
  Pat2[i] <- Head2(valve_loss_coefficient[i],g,VofT[i])
}
par(mfrow=c(1,2))
plot(elapsed_time,VofT/Vzero,xlab="Time",ylab="V/
 Vo",type="l",tck=1,col="blue",lwd=3)
plot(elapsed_time,Pat2/Ho,xlab="Time",ylab="H/
 Ho",type="l",tck=1,col="red",lwd=3)
```

RStudio

Go to file/function | Addins | Project: (None)

ReservoirValveOpen.R | ReserviorValveClose.R

Source on Save | Run | Source

```r
# Finite-difference approximation for scheduled valve closure
########### Prototype Functions ############
Vsteady <- function(head,distance,diameter,gravity,friction){
  Vsteady <- sqrt((2.0*gravity*diameter*head)/(friction*distance))
  return(Vsteady)
}
######
Head2 <- function(valve_loss_coefficient,gravity,velocity){
  Head2 <- valve_loss_coefficient*((velocity^2)/(2*gravity))
  return(Head2)
}
######
steadyVelocity <- function(head,distance,diameter,gravity,friction){
  steadyVelocity <- sqrt((2.0*gravity*diameter*head)/(friction*distanc
  return(steadyVelocity)
```

66:74 | (Untitled) | R Script

Console ~/Dropbox/3-Research/NetworkSimulators/UnsteadyPipeNetwork/RigidWaterCol

```
        Enter Pipe Diameter 2
 Enter gravitational constant 32.2
  Enter darcy friction factor 0.018
  Intrinsic Valve Coefficient 9
      Computational Time Step 3
         How Many Time Steps 200
  Reservoir Pool Elevation : 100
           Pipeline Length : 10000
             Pipe Diameter : 2
      Gravitational Constant : 32.2
       Darcy Friction Factor : 0.018
Intrinsic Valve Coefficient : 9
     Computational Time Step : 3
         How Many Time Steps : 200
                          Vo : 8.45905169363301
>
```

Environment | History

Files | Plots | Packages | Help | Viewer

Zoom | Export | Publish