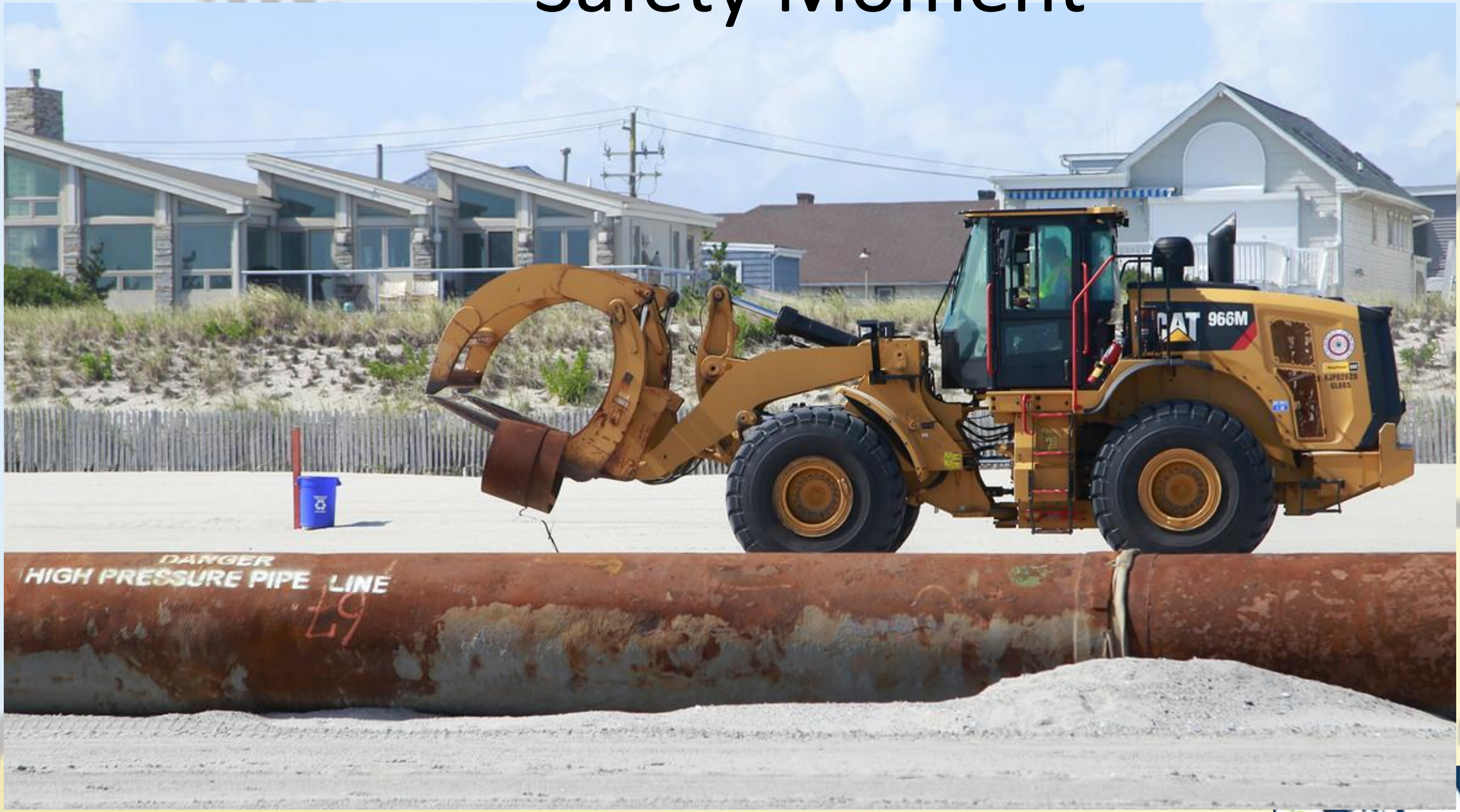
Slurry Transport Using the DHLLDV Software

Robert Ramsdell

Robert.Ramsdell@DredgingResources.net

DREDGING RESOURCES

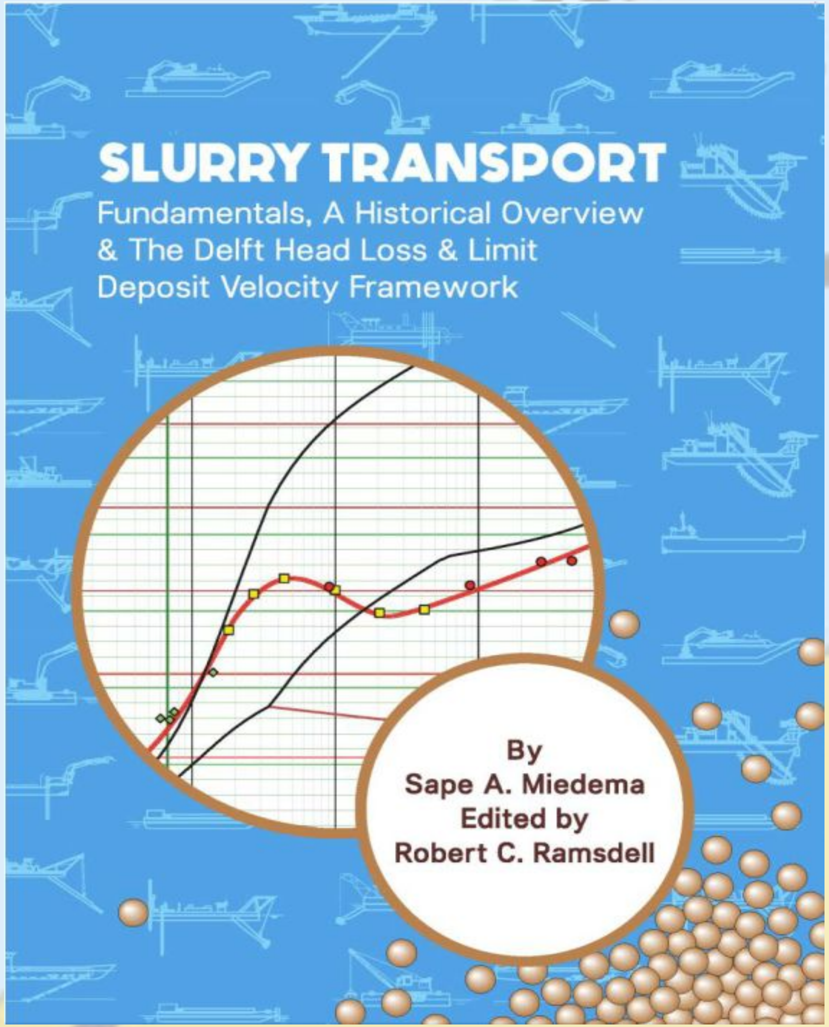
Safety Moment



Western Dredging Association

Introduction

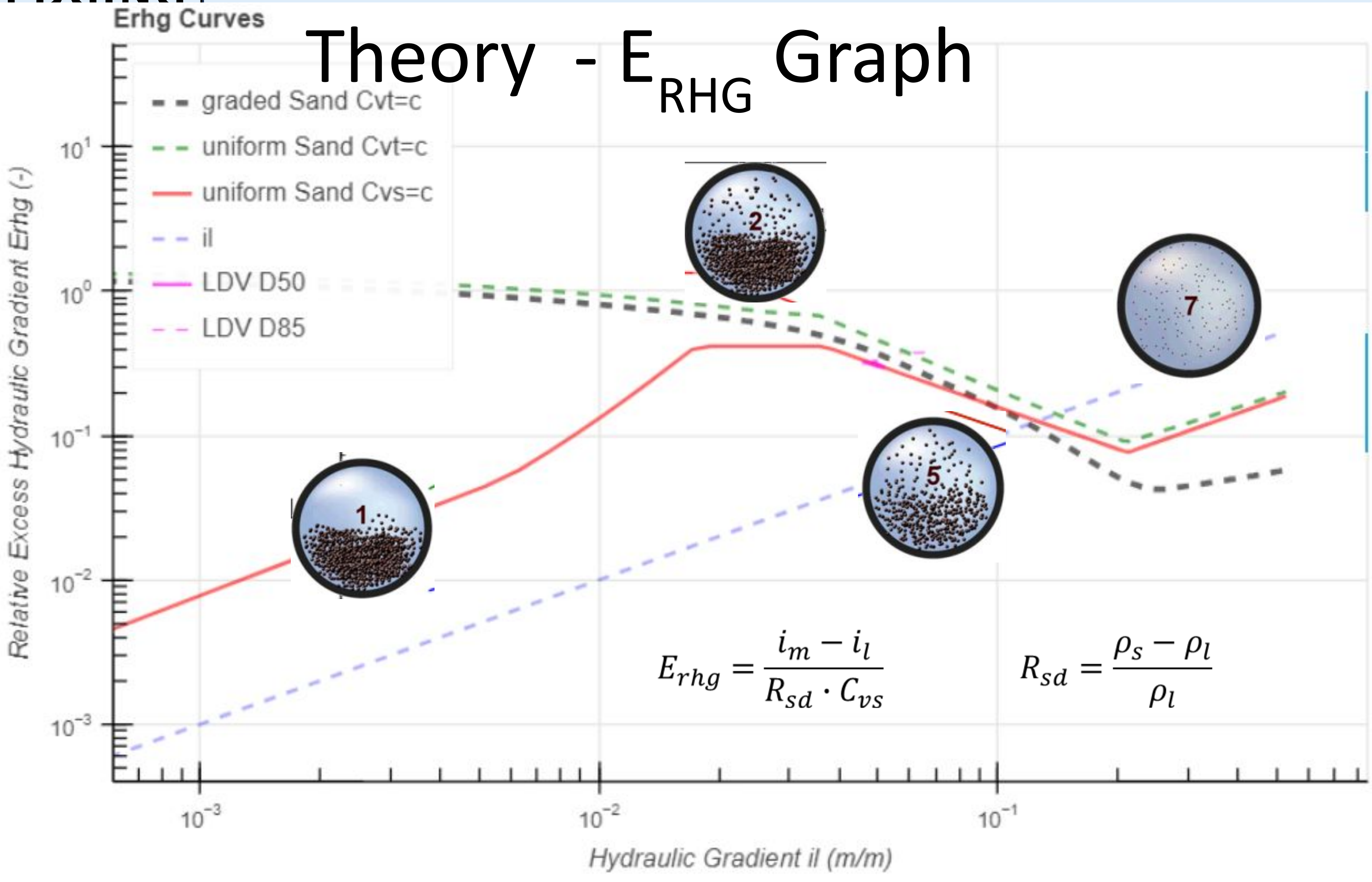
- Theory Refresher
- Dredge Pumps
- Dredge Pumping System
- The Excel Spreadsheet
- DHLLDV Software – Overview & Installation
- Using the Software
- How Can You Help?



- Chapters 1 – 5 are the theoretical background
  - Pressure losses in pipe (Darcy-Weisbach etc)
  - Particle settling velocities
  - Initiation of motion & sediment transport
  - Sand/Shell mixtures
- Chapter 6 is an overview of historical models
  - Durand & Condolios, Fuhrboter, Wasp, Wilson, and many more!
- Chapter 7 derives the DHLLDV theory
- Chapter 8 presents how to implement the model

https://www.researchgate.net/publication/330753872_The_Delft_Head_Loss_Limit_Deposit_Velocity_Framework_2nd_Edition

Theory - E_{RHG} Graph



$$E_{rhg} = \frac{i_m - i_l}{R_{sd} \cdot C_{vs}}$$

$$R_{sd} = \frac{\rho_s - \rho_l}{\rho_l}$$

Theory - Concentrations

- $C_v = \frac{\rho_m - \rho_l}{\rho_s - \rho_l}$
  - ρ_m is the slurry density (ton/m^3)
  - ρ_l is the fluid density (0.998 for water, 1.025 for salt water)
  - ρ_s is the solids density (2.65 for sand & gravel)
- C_{vs} = In-pipe Concentration
- C_{vt} = Transported Concentration, relative to solids density
- $C_{vi} = \frac{\rho_m - \rho_l}{\rho_i - \rho_l}$ = Insitu volume concentration
  - ρ_i is the insitu (in place) density including porewater (1.85 – 2.05 for sand)
- Production = Q * C_{vi} where Q (m^3/hour) is the total flow rate

Theory – Head and Pressure

- Head is a measure of pressure with the units of length (??)

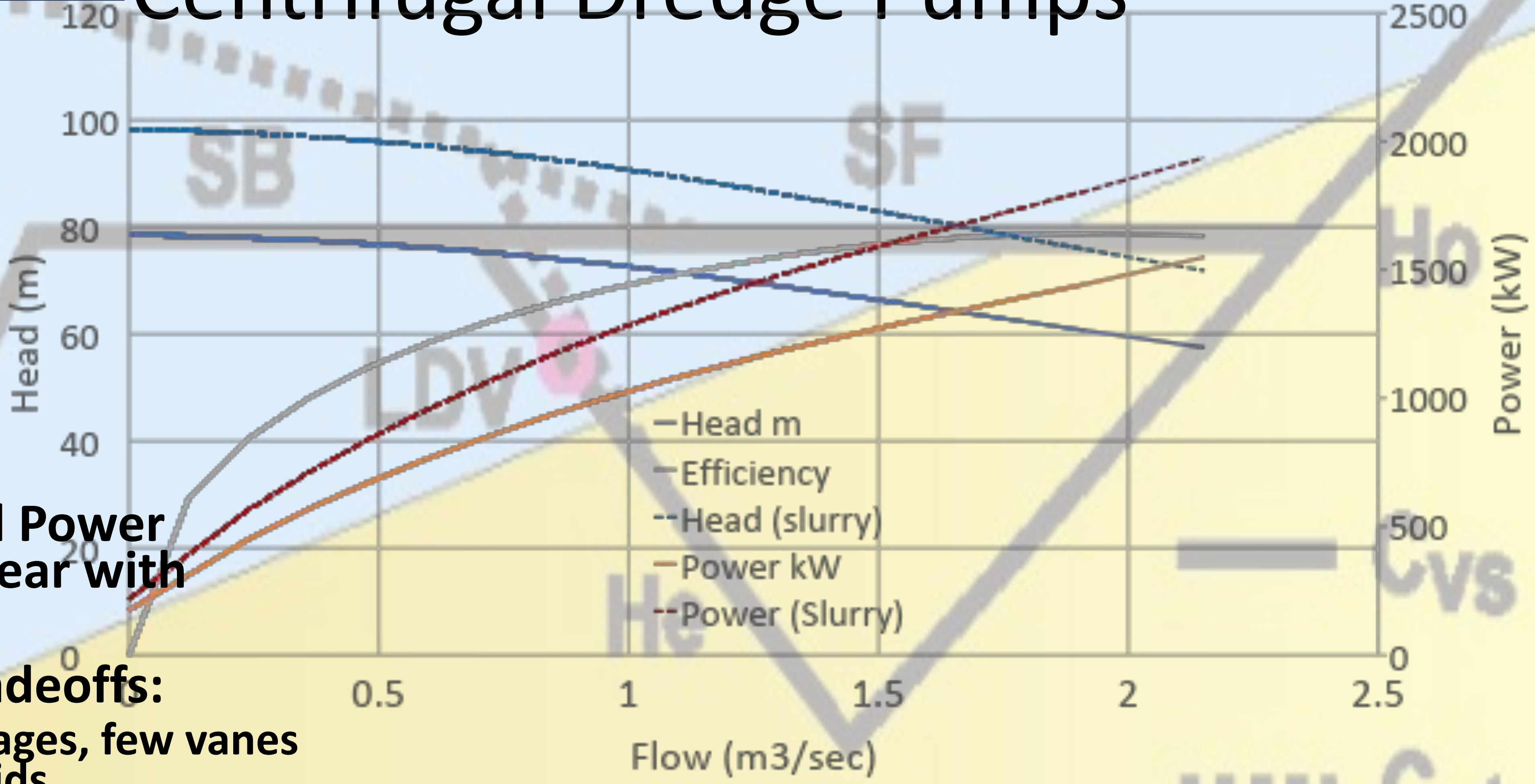
$$H = \frac{p}{\rho_l g}$$

It is convenient for talking about the energy
(potential & kinetic) in a piece of pipe

1 psi = 2.31 ‘ of head (of what?)

0.098 bar = 9.8 kPa = 1m head

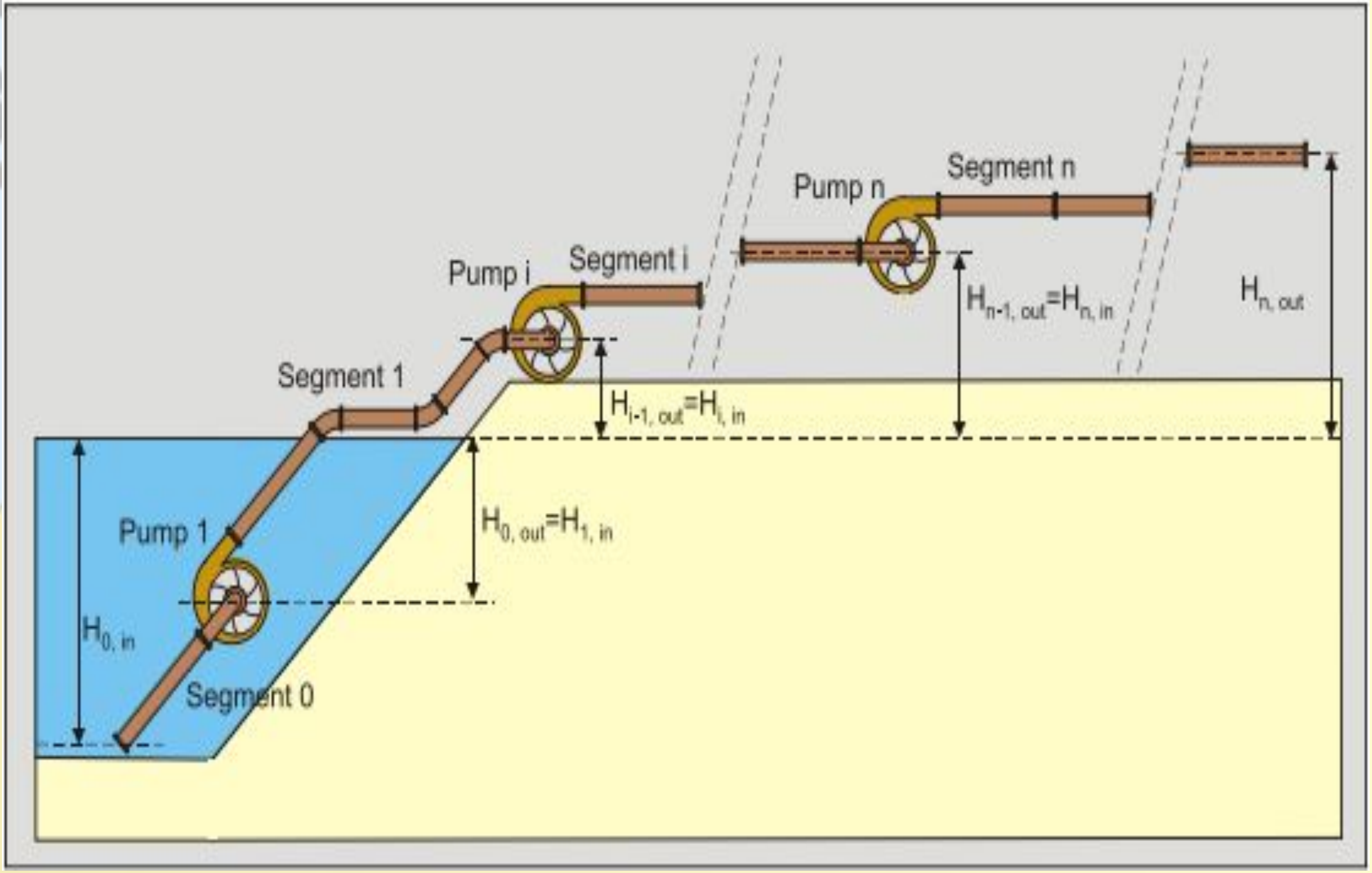
Centrifugal Dredge Pumps



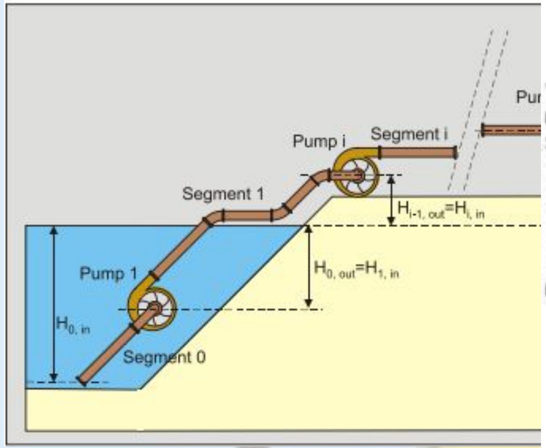
- **Pressure and Power demand ~linear with density**
- **Efficiency tradeoffs:**
  - **Large passages, few vanes to pass solids**
  - **Thick parts for wear life and shock tolerance**

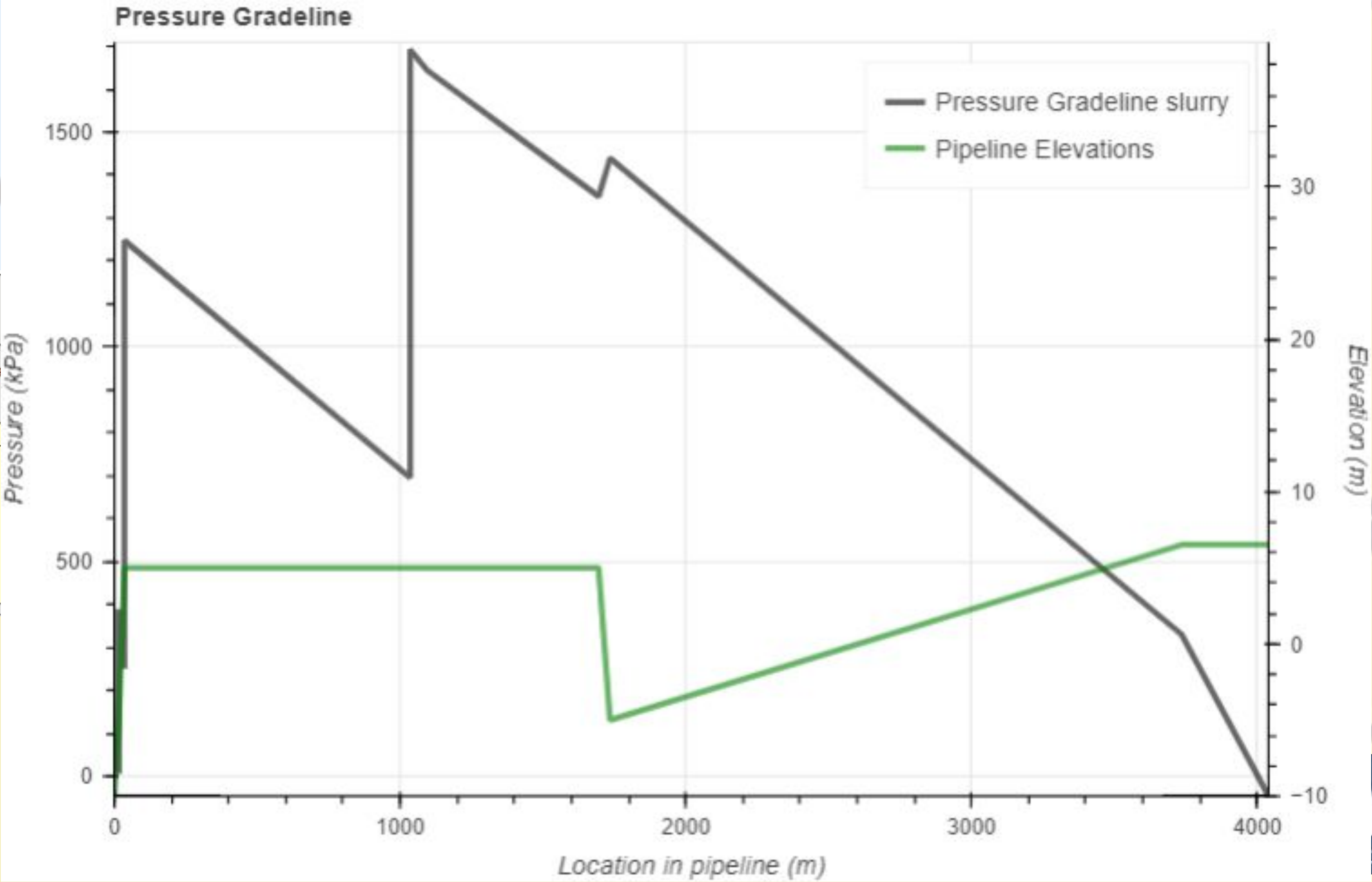
The Pipeline System

- Starts with an entrance
- Consists of alternating pipe sections and pumps
- Pipe segments have varied diameters and fittings
- Pumps may be the same or different
- Elevations must be taken into account



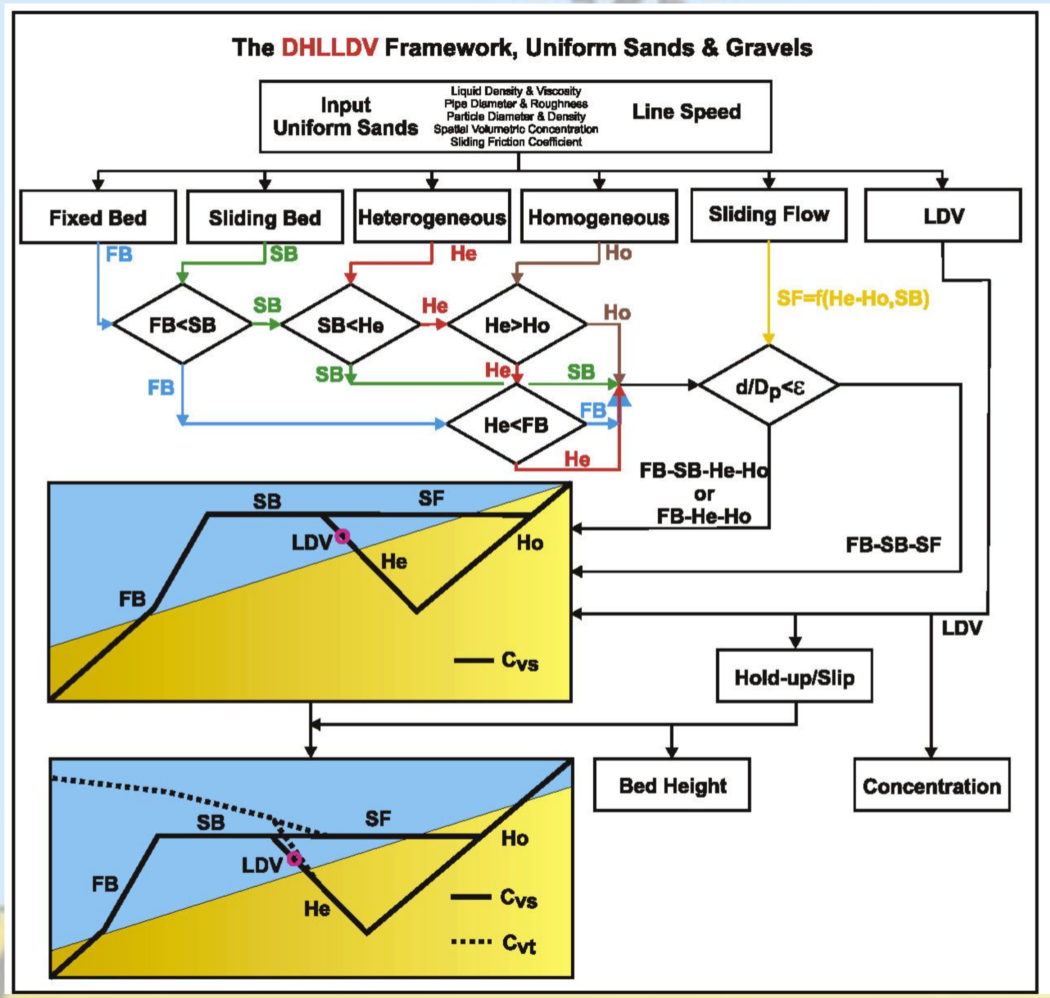
Pressure in the Pipe

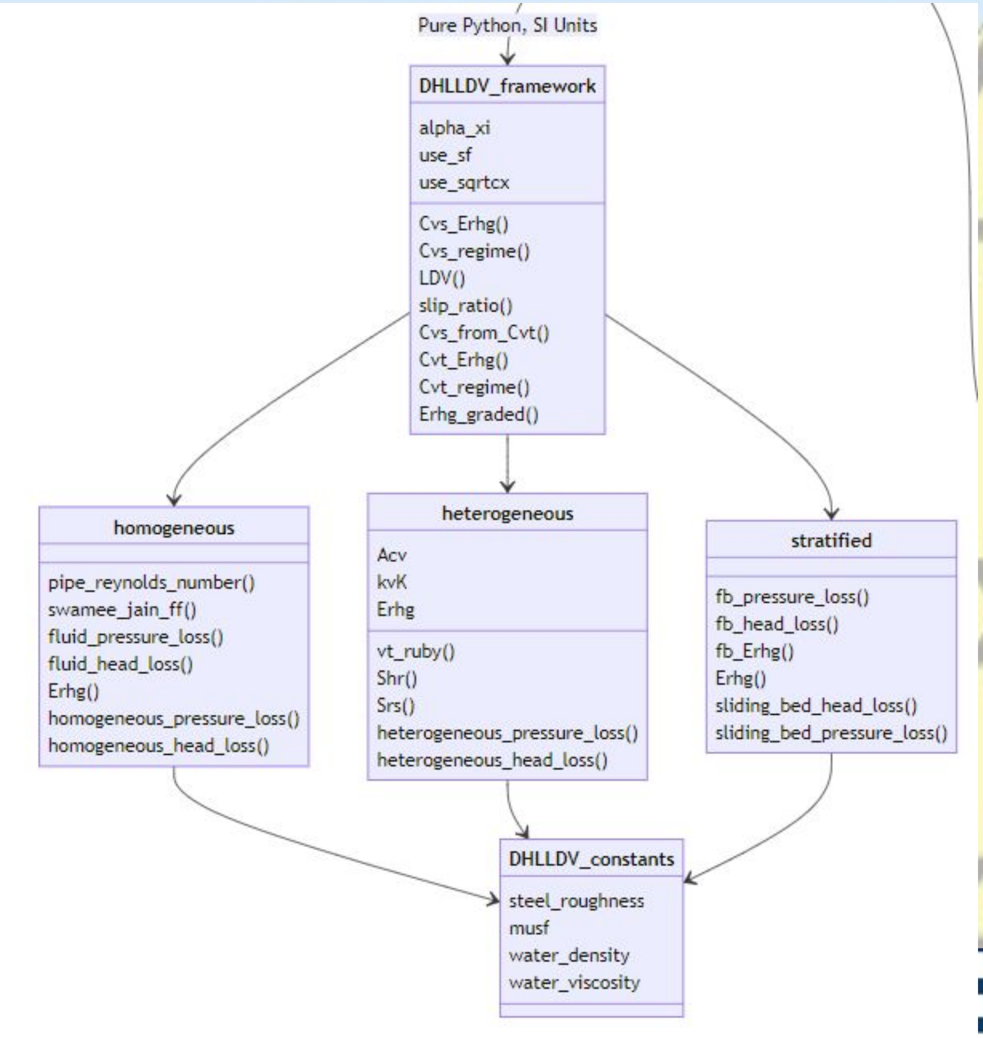




DREDGING RESOURCES

From Model to Software





Western Dredging Association

DHLLDV_Framework.xlsm

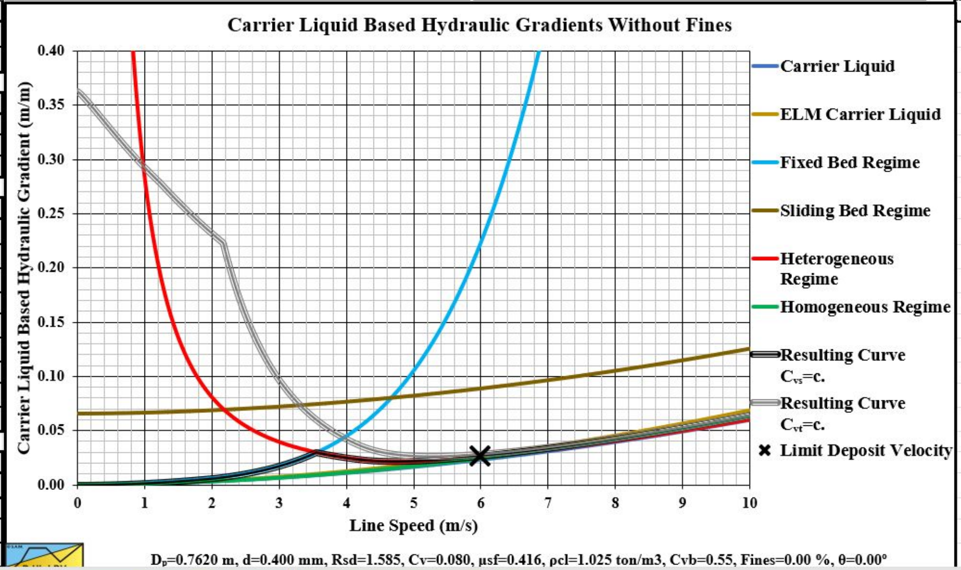
DHLLDV Framework.xlsm

C68 | 3.35454454527339

Head Losses Single Particle Inputs & Calculations

Manual	Calculate Graded Solids	Goto Graphs Other Models	Export Inputs	Import Inputs
Defaults	Calculate Single Particle	Goto Graphs Single Particle	Goto Graphs Flow Regime Transitions	

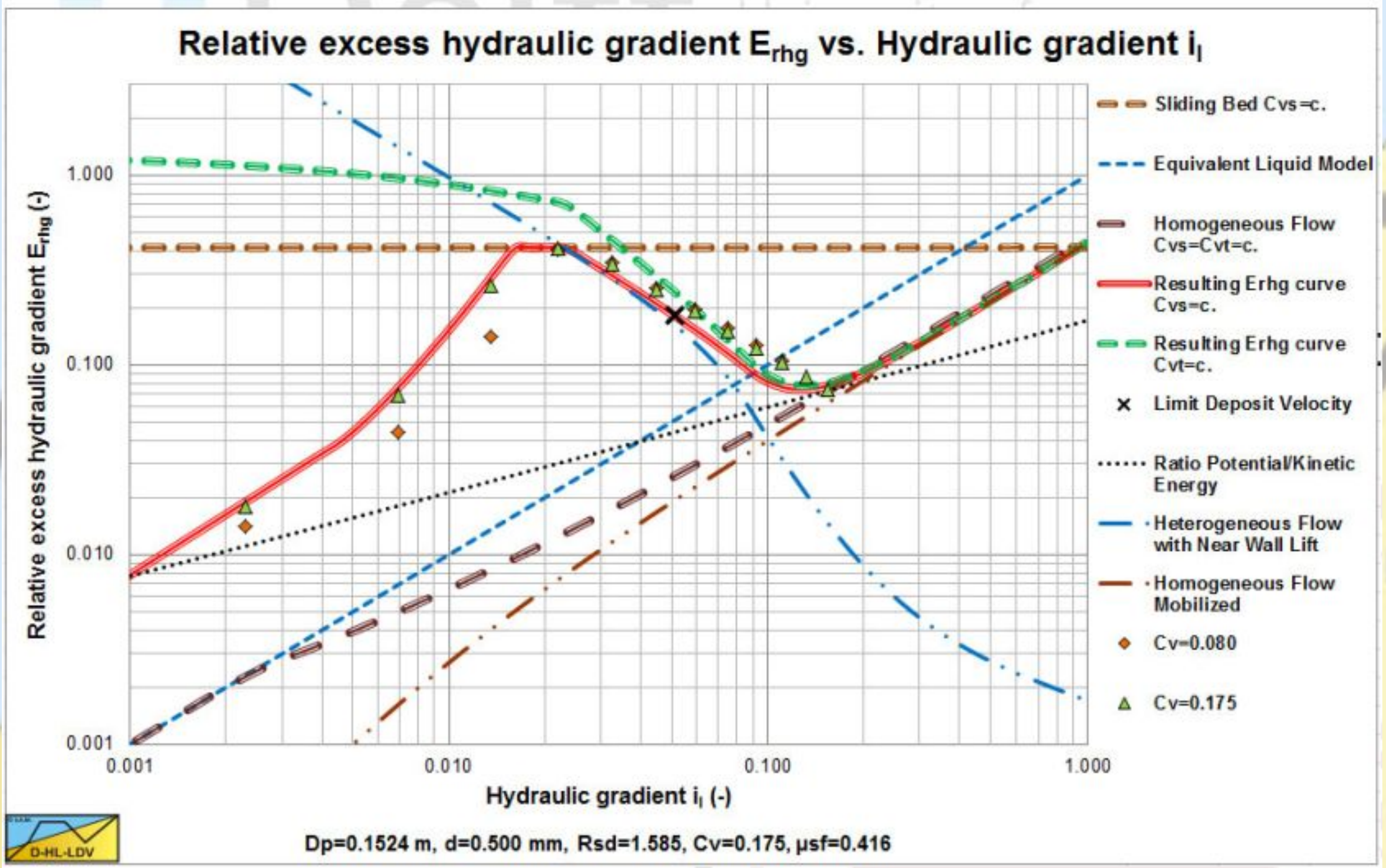
	Carrier liquid input parameters		
1.025	Density of carrier liquid ρ_l	1.025	ton/m^3
0.0000013	Kinematic viscosity ν_l	0.0000013	m^2/sec
	Pipe input parameters		
0.7620	Pipe diameter D_p	0.7620	m
0.0000010	Wall roughness ε	0.0000010	m
0.00	Pipe inclination angle	0.00	°
	Solids input parameters		
2.650	Density of solids ρ_s	2.650	ton/m^3
0.001000	Particle diameter d or d_{50}	0.000400	m
0.270	Turbulent lift coefficient C_L	0.270	-
0.450	Bed prosity n	0.450	-
0.416	Sliding bed friction factor μ_{sf} (0.3-0.5)	0.416	-
0.300	Volumetric concentration C_{vs} or C_{vt}	0.080	-
0.100	Fines fraction	0.100	-
2.718	Ratio d_{50}/d_{15} graded sands	5.000	-
2.718	Ratio d_{85}/d_{50} graded sands	5.000	-
	DHLLDV Framework model parameters		
3.00	Homogeneous factor A_{cv} (0-4)	3.00	-
6.80	Heterogeneous factor c (5-8)	6.80	-
0.015	Sliding flow d/D_p ratio	0.015	-
3.400	Limit Deposit Velocity factor α_p	3.400	-



Manual | Calculations Menu | Head Losses Single Particle | Experimental Data | Concentration Distribution | Experimental Data CD | Ma

https://github.com/rcriii42/DHLLDV/tree/master/DHLLDV_Framework_xlsm

DHLLDV_Framework.xlsm



https://github.com/rcriii42/DHLLDV/tree/master/DHLLDV_Framework_xlsm

DREDGING RESOURCES

The DHLLDV Software

- Open Source
- Written in Python
- Hosted on GitHub
- Tested

WEDA
Western Dredging Association

Goals
(Why use MY code?)

- Accessible
  - Easy to Use
  - Importable
  - Unincumbered
  - Long-lived
- Reliable
  - Correct
  - Secure
  - Verifiable

Accessibility

Written in Python (3.10+)

- Simple clear syntax
- “The programmer’s time is as important as the computer’s”
- Quick write-run-rewrite cycle
- “Batteries included”
- Lots of third party libraries and tools

https://www.python.org/

Hosted on Github

- Public and Private repositories
- Source control
- Issue Tracker
- Multi user workflow
- Security updates
- Wiki & Discussions

https://github.com/rcriii42/DHLLDV

Accessible – Open Source

- Free redistribution
- Source code available
- Derivatives allowed
- No limitations on use
- Gives back to the community!

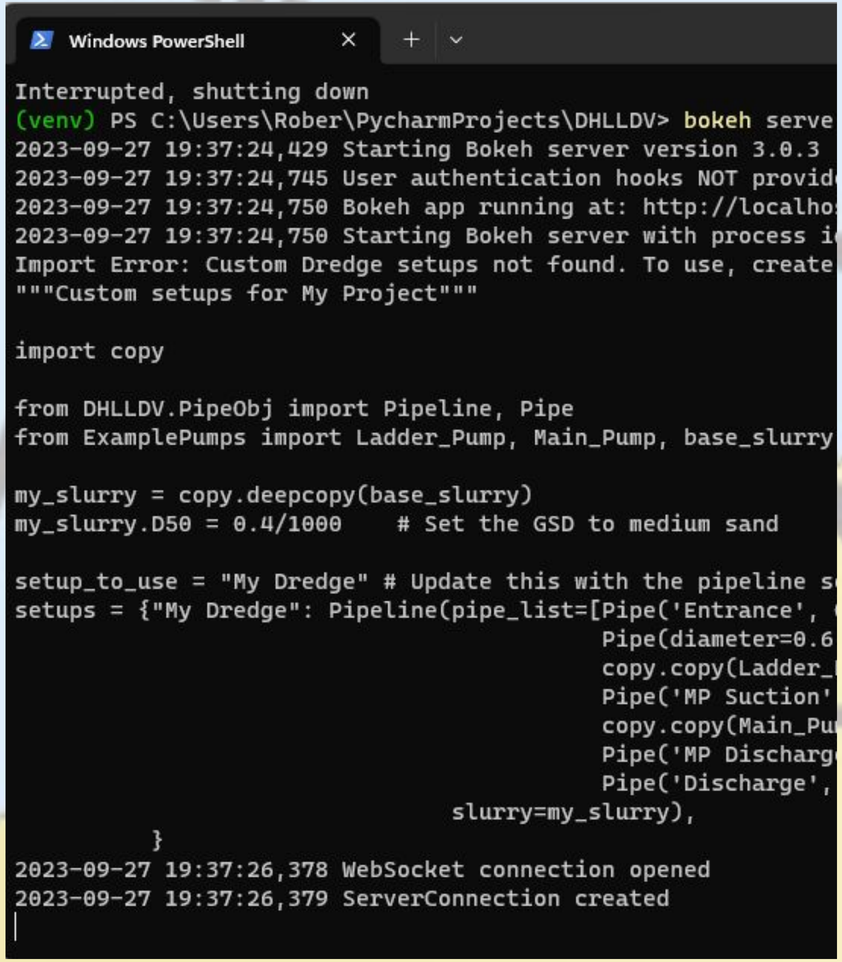
Coding Standards

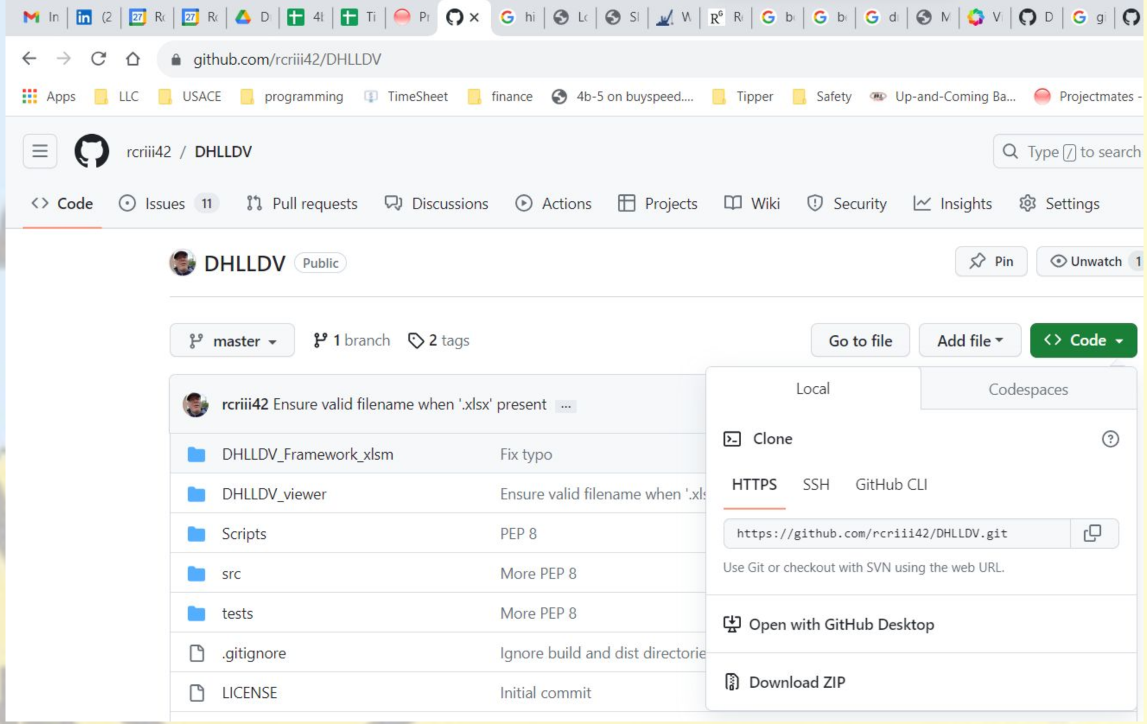
```
PS C:\Users\Rober> python
Python 3.11.5 (tags/v3.11.5:cce6ba9, Aug 24 2023, 14:38:34) [MSC v.1936
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

- Underlying framework in SI units, viewer does conversions
- PEP 8 style guidelines: https://peps.python.org/pep-0008/
- PEP 257 Docstring Conventions: https://peps.python.org/pep-0257/
- No type annotations

DREDGING RESOURCES

What you'll need:





https://github.com/rcriii42/DHLLDV

WEDA Western Dredging Association

Installation (follow along!)

Assuming you have Python installed!

1. Open command window
2. Navigate to your desired location (root is fine)
3. Clone the repository:

```
git clone https://github.com/rcriii42/DHLLDV.git DHLLDV2
```

4. Create the virtual environment: `python -m venv .\venv`
5. Start the virtual environment: `venv\scripts\activate.ps1`
6. Download requirements: `pip install -r requirements.txt`
7. Run tests: `pytest --cov-report term-missing --cov=DHLLDV`
8. Start the viewer: `bokeh serve .\DHLLDV_viewer\ --show`

Code Sample

The equation for the terminal settling

$$v_t = \frac{10 \cdot \nu_l}{d} \cdot \left(\sqrt{1 + \frac{R_{sd} \cdot g \cdot d^3}{100 \cdot \nu_l^2}} - 1 \right)$$

heterogeneous.py

```python
'''
heterogeneous.py - heterogeneous transport model.
Created on Feb 21, 2015

@author: rcriii
'''

from .DHLLDV_constants import gravity, Arel_to_beta, musf, particle_ratio
from . import homogeneous


14 usages   Robert Ramsdell
def vt_ruby(d, Rsd, nu, K=0.26):
    """vt_ruby - terminal settling velocity via the Ruby & Zanke formula (eqn 8.2-2)
       d particle diameter (m)
       Rsd relative solids density
       nu fluid kinematic viscosity in m2/sec
       k particle shape factor (sand = 0.26) (not used, included for compatibility
    """
    right = 10 * nu / d
    left = (1 + (Rsd * gravity * d**3) / (100 * nu**2))**0.5 - 1
    return right * left  # Eqn 8.2-2
```

Using the Software

Top Bar

- Pipeline chooser
- Unit conversions
- Open/save to Excel
- Stop Button

Slurry Tab

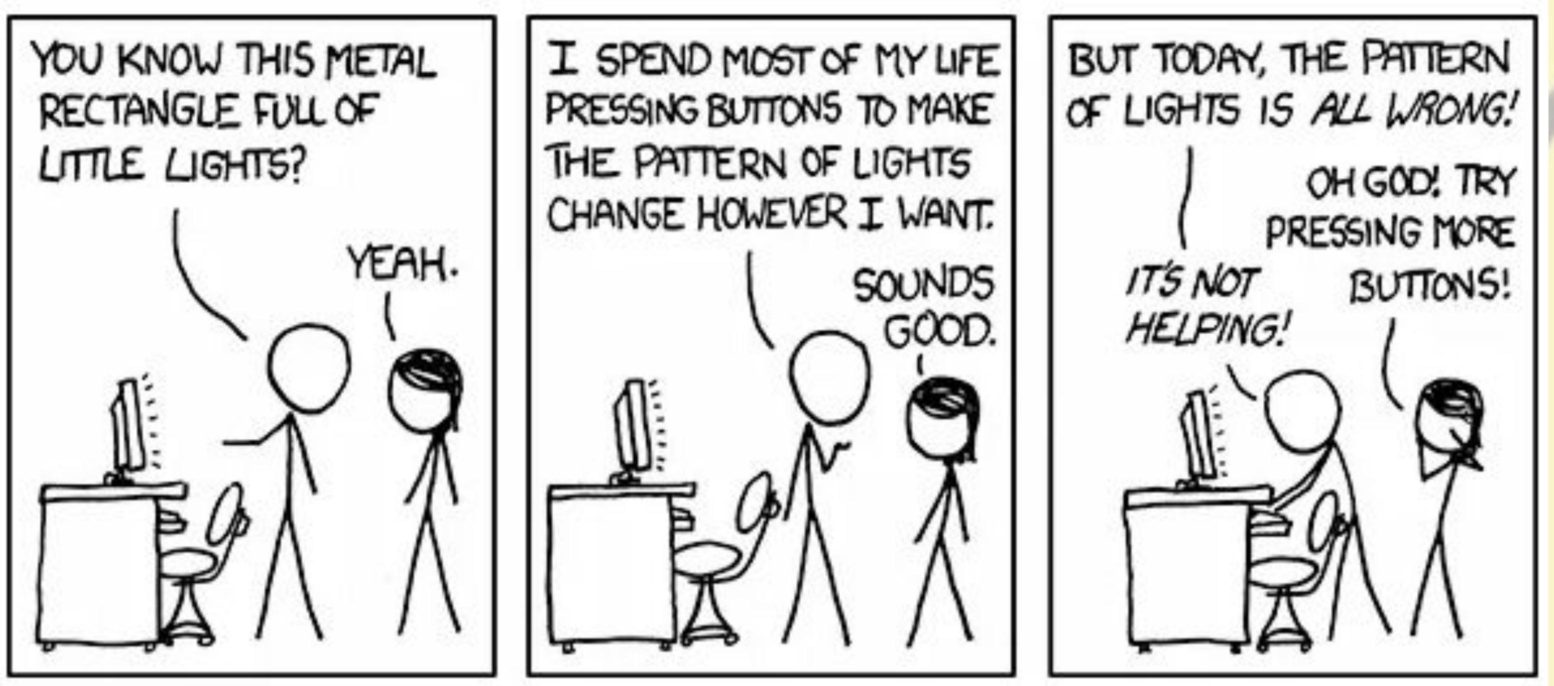
- Pipe Diameter
- Fluid
- Grain size distribution
- Density and Concentration
- i_m graph
- E_{rhg} Graph

Pipeline Tab

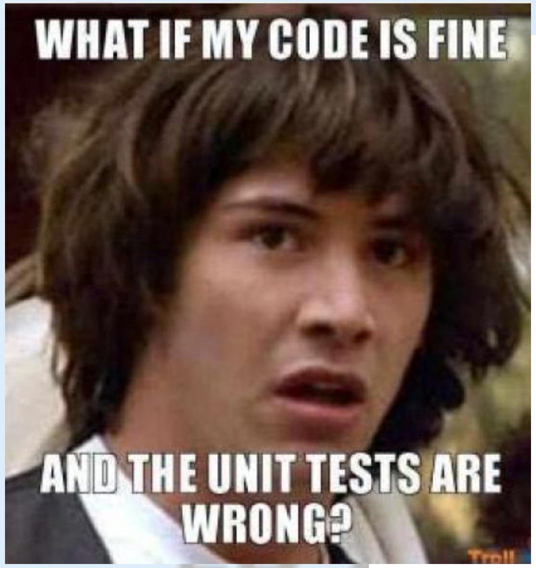
- Pipeline layout
- Slurry details
- System Head plot
- Minimum Friction Point
- Operating Point
- Pressure Gradeline

Unit Tests

- Isolate each part of the program and show that individual parts are correct
- Write tests before underlying code
- Tests pass before moving on.

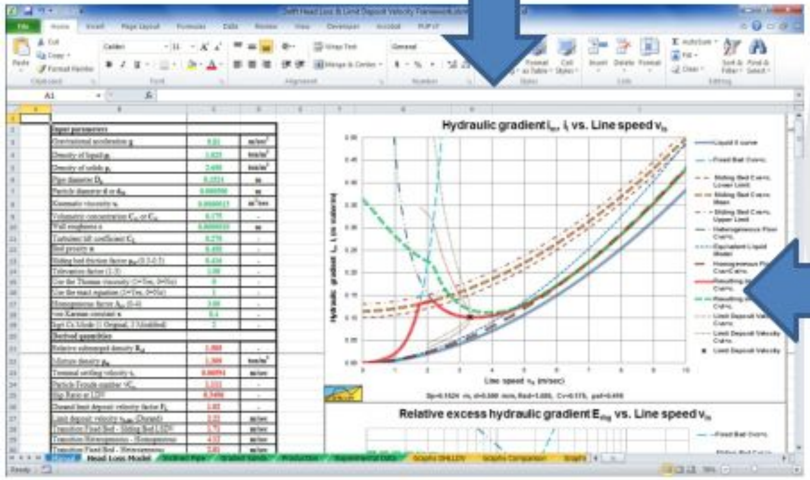


What are we testing?

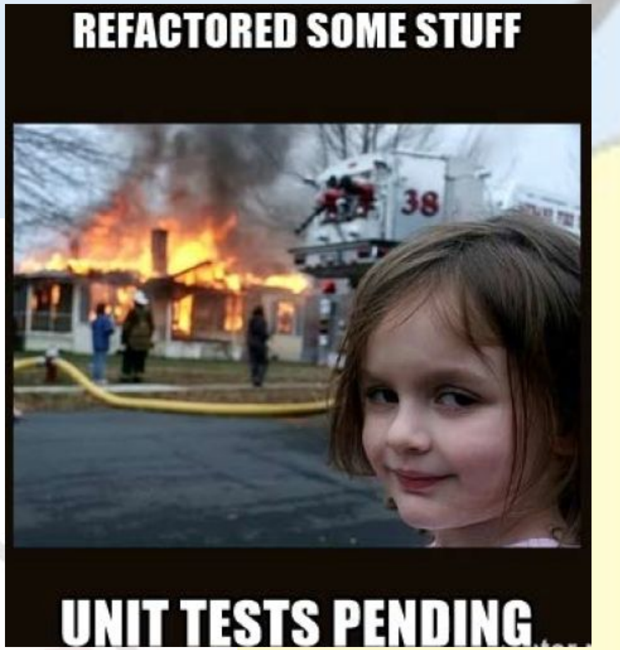


$$E_{rhg} = \frac{i_m - i_l}{R_{sd} \cdot C_{vs}} =$$

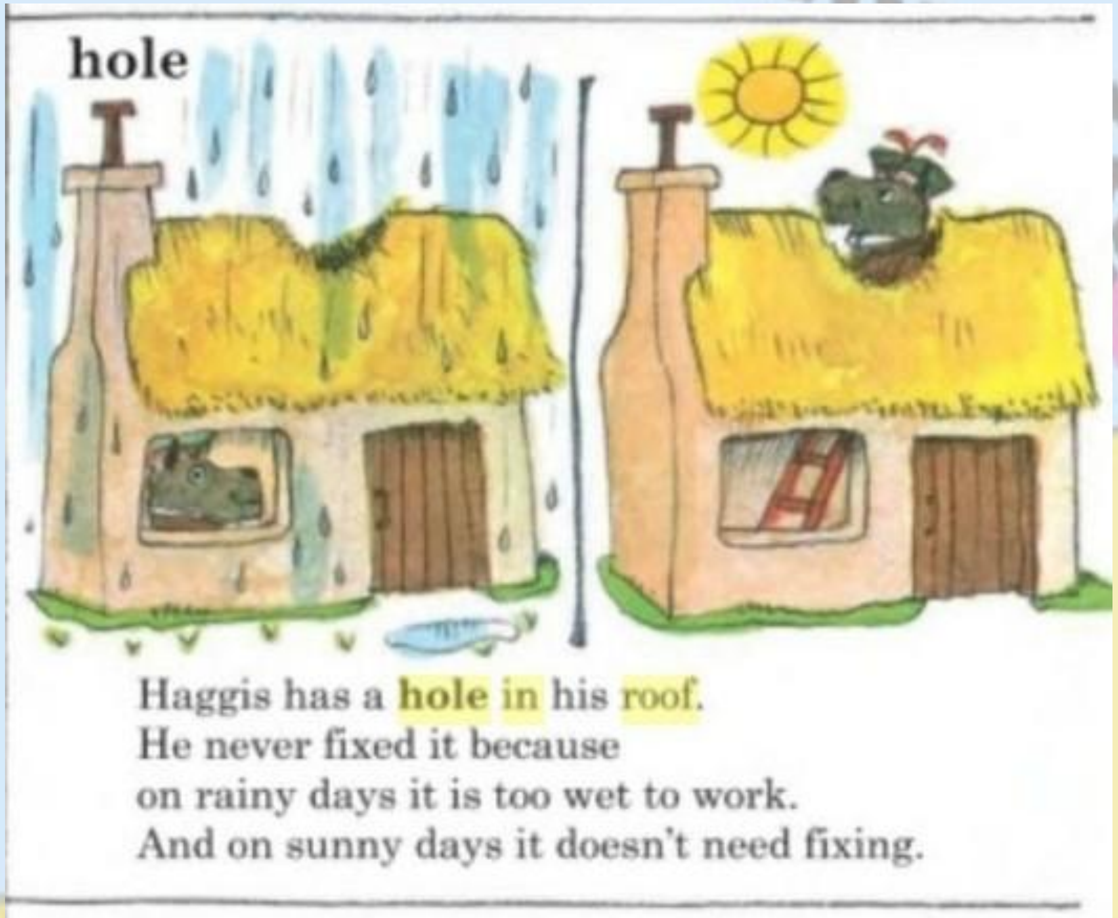
$$i_l \cdot \left[1 - \left[1 - \frac{1 + R_{sd} \cdot C_{vs} - \left(\frac{A_{C_v}}{\kappa} \cdot ln\left(\frac{\rho_m}{\rho_l}\right) \cdot \sqrt{\frac{\lambda_l}{8}} + 1 \right)^2}{R_{sd} \cdot C_{vs} \cdot \left(\frac{A_{C_v}}{\kappa} \cdot ln\left(\frac{\rho_m}{\rho_l}\right) \cdot \sqrt{\frac{\lambda_l}{8}} + 1 \right)^2} \right] \left[1 - \left[\frac{\delta_v}{d} \right] \right] \right]$$




```
def Erhg(vls, Dp, d.     ilon, nu, rhol, rhos, Cvs):
    """Return the E    alue for homogeneous flow.
    Use the Talmon (2 3) correction for slurry density.
    vls: line speed in m/sec
    Dp: Pipe diameter in m
        d: Particle diameter in m (not used, here for
consistency)
    epsilon: pipe absolute roughness in m
    nu: fluid kinematic viscosity in m2/sec
    rhol: fluid density in ton/m3
    Cvs - spatial (insitu) volume concentration of solids
    """
    Re = pipe_reynolds_number(vls, Dp, nu)
    lambdal = swamee_jain_ff(Re, Dp, epsilon)
    Rsd = (rhos-rhol)/rhol
    rhom = rhol+Cvs*(rhos-rhol)
     deltav_to_d = min((11.6*nu)/((lambdal/8)**0.5*vls*d), 1)
#eqn 8.7-7

    sb = ((Acv/kvK)*log(rhom/rhol)*(lambdal/8)**0.5+1)**2
    top = 1+Rsd*Cvs - sb
    bottom = Rsd*Cvs*sb
    il = fluid_head_loss(vls, Dp, epsilon, nu, rhol)
    return il*(1-(1-top/bottom)*(1-deltav_to_d)) #eqn 8.7-8
```



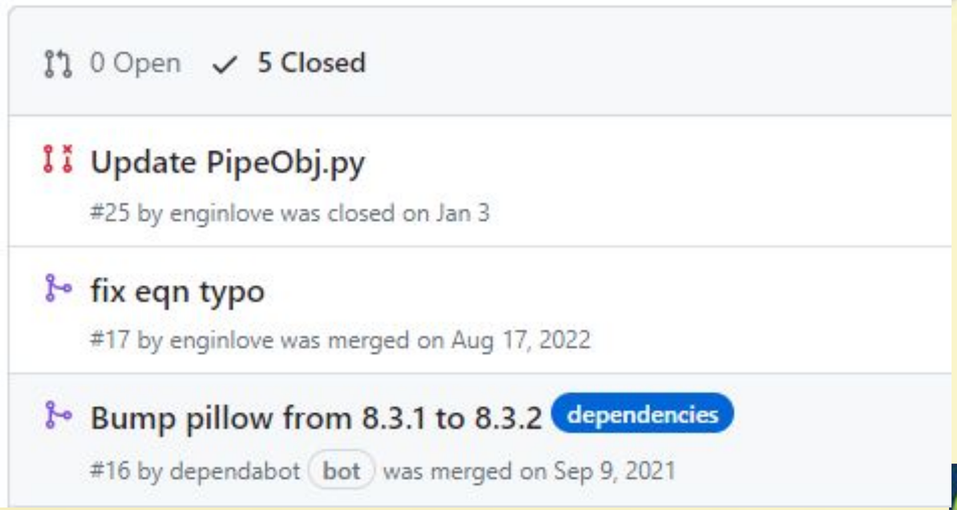
Code Coverage



- Make sure testing is comprehensive
- Find lines that are not tested
- Currently at 87%
- Low % due to heavy development of:
  - DriverObj
  - PipeObj
  - PumpObj

How can You Help?

- Use the Library
- Email problems, requests, and ideas
  Robert.Ramsdell@dredgingresources.net
- Better: Raise issues: https://github.com/rcriii42/DHLLDV/issues
- Best: Submit patches!

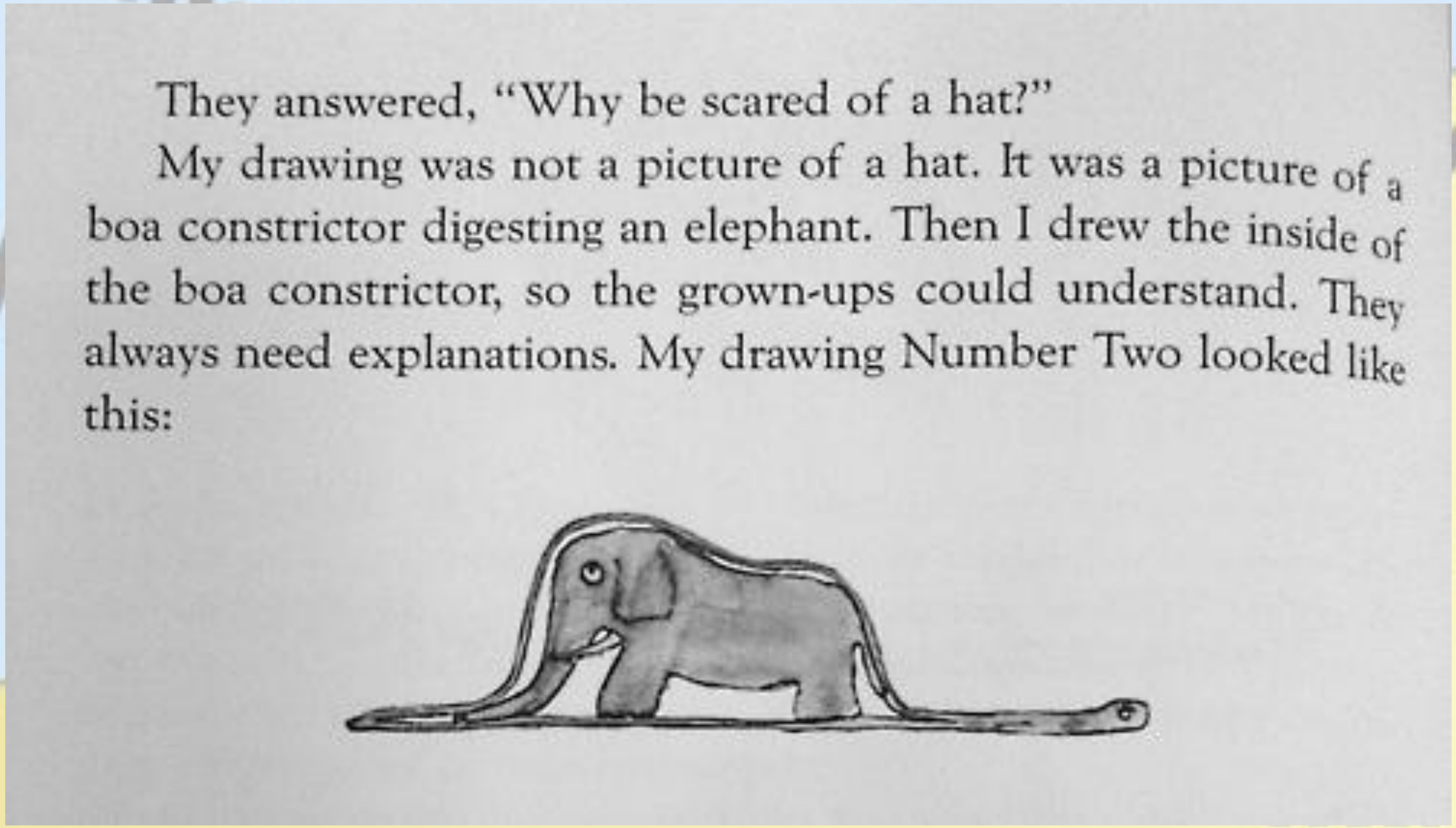


DREDGING RESOURCES

Patch - Fixed Error

```
4 src/DHLLDV/DHLLDV_framework.py

@@ -211,7 +211,7 @@ def slip_ratio(vls, Dp,  d, epsilon, nu, rhol, rhos, Cvt):

      Re = homogeneous.pipe_reynolds_number(vls, Dp, nu)
      lambda_l = homogeneous.swamee_jain_ff(Re, Dp, epsilon)
-     Xi_HeHo = 8.5*(1/lambda_l)*(vt/(gravity*d)**0.5)**5/3*((nu*gravity)**(1/3)/vls)*(vt/vls)  # Eqn 8.12-1
+     Xi_HeHo = 8.5*(1/lambda_l**0.5)*(vt/(gravity*d)**0.5)**(5./3)*((nu*gravity)**(1/3)/vls)*(vt/vls)  # Eqn 8.12-1

      alpha = 0.58*Cvr**-0.42
      ex1 = -(0.83 + stratified.musf/4 + (Cvr - 0.5 - 0.075*Dp)**2 + (0.025*Dp))
@@ -221,7 +221,7 @@ def slip_ratio(vls, Dp,  d, epsilon, nu, rhol, rhos, Cvt):
      vls_t = (5 * exp(ex1 * ex2)) ** 0.25 * vls_ldv  # Eqn 8.12-7

      Kldv = 1/(1 - Xi_ldv)         # Eqn 7.9-14
-     Xi_fb = 1-((Cvt*vls_ldv)/(Cvb-Kldv*Cvt)*(vls_ldv-vls)+Kldv*Cvt*vls_ldv)  # Eqn 8.12-3
+     Xi_fb = 1-((Cvt*vls_ldv)/((Cvb-Kldv*Cvt)*(vls_ldv-vls)+Kldv*Cvt*vls_ldv))  # Eqn 8.12-3

      ex2 = Dp ** 0.025 * (vls / vls_lsdv) ** alpha * Cvr ** 0.65 * (Rsd / 1.585) ** 0.1
      Xi_3LM = (1 - Cvr) * exp(ex1 * ex2)  # Eqn 8.12-4
```

DREDGING RESOURCES


They answered, "Why be scared of a hat?"

My drawing was not a picture of a hat. It was a picture of a boa constrictor digesting an elephant. Then I drew the inside of the boa constrictor, so the grown-ups could understand. They always need explanations. My drawing Number Two looked like this:

WEDA Western Dredging Association