The DHLLDV Software Model

Development and Use

Robert C. Ramsdell
Great Lakes Dredge & Dock Company
Sape A. Miedema
Delft University of Technology

DID YOU KNOW

SAFETY IS NOT ONLY ABOUT TAKING PRECAUTIONS, IT'S ALSO ABOUT TAKING RESPONSIBILITY.

There's a catch phrase that's being heard more and more these days. "See it. Own it." That phrase is particularly applicable to safety.

If you see an unsafe situation, or even a potentially unsafe situation, don't just walk away. Take responsibility for getting it corrected.

Whether it's in the office, while you're traveling, or at the work site, wherever you see something that you believe is unsafe, or could lead to an adverse incident, speak up. If it's unsafe to actually do something about it yourself, keep others out of the unsafe zone and contact your supervisor.

Think how you'd feel if you did nothing, then heard later that someone was injured.

http://www.halliburton.com/en-US/about-us/hse-service-quality/hse-safety-moments.page

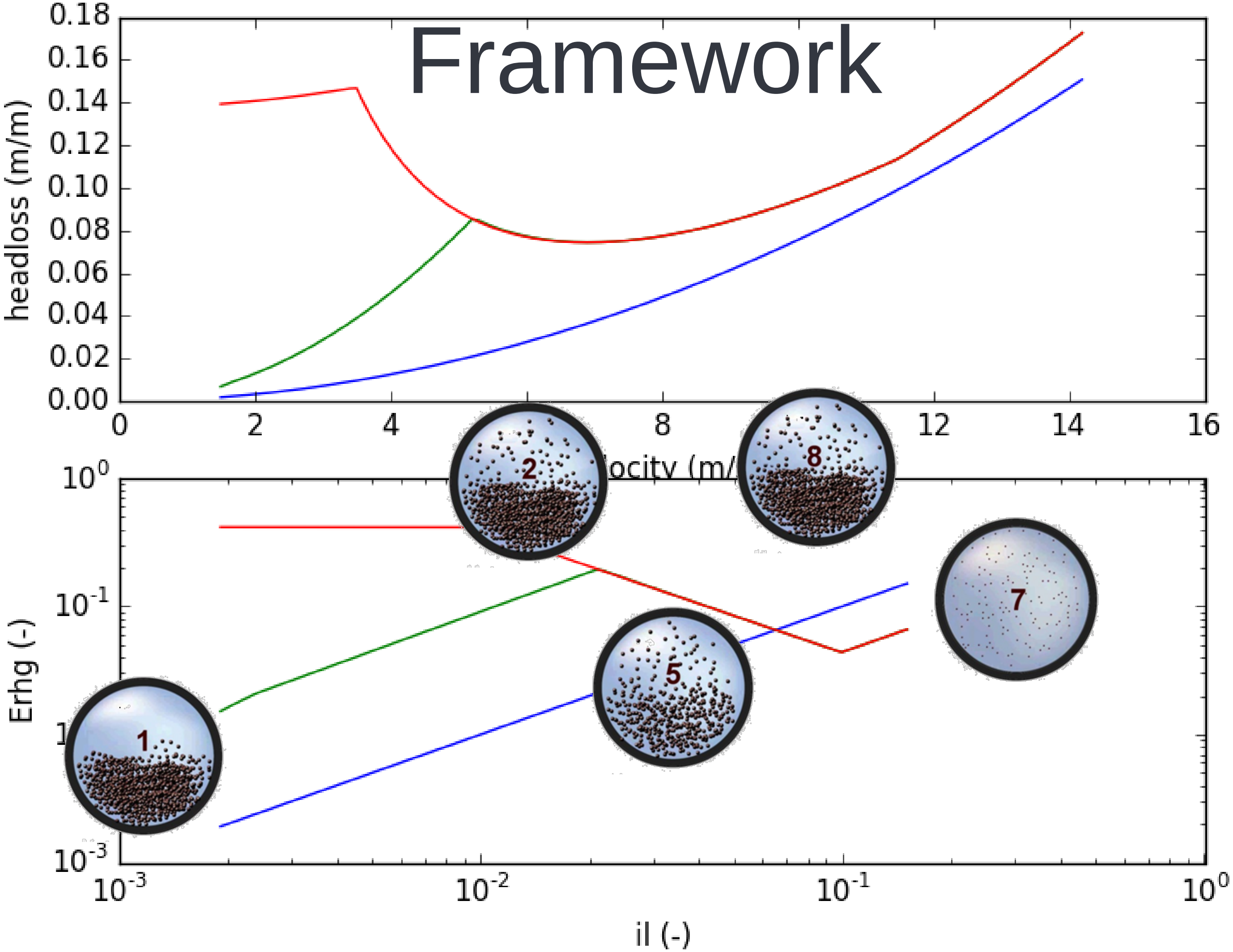
At Halliburton, solving customer challenges is second only to keeping everyone safe and healthy. You can find more safety tips at **www.halliburton.com/HSE**.

Safety Moment Subject suggested by: Allen McClure, *Halliburton Employee*

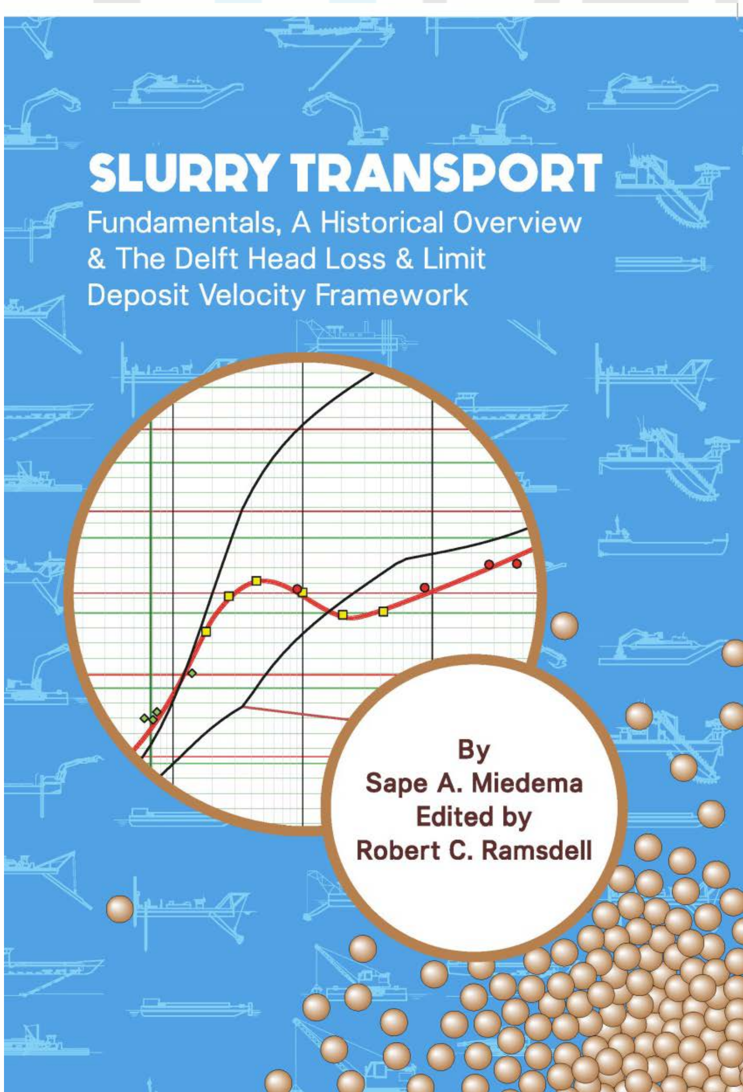
Introduction

- The authors developed the Delft Head Loss & Limit Deposit Velocity (DHLLDV) Framework, using commonly available parameters
- Has sub models for each flow regime, the limit deposit velocity, holdup function and graded sands and gravels

Overview – The DHLLDV Framework



Slurry Transport Fundamentals



- The book was completed in time for WODCON and handed out to attendees.
- Chapter 7 presents the theory behind DHLLDV
- Chapter 8 presents how to implement it

The Framework - Formulae

Sliding bed:

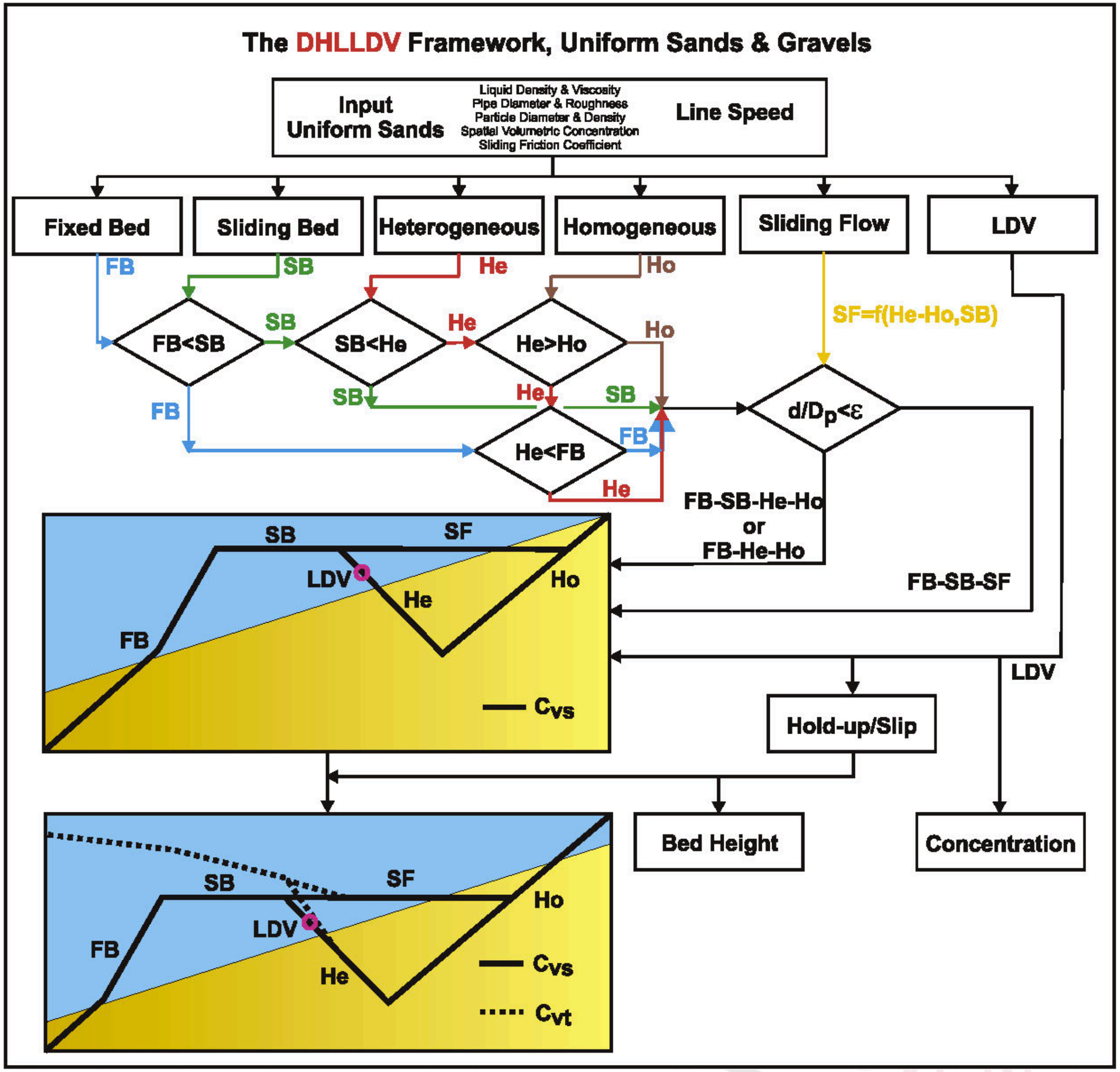
$$E_{rhg} = \frac{i_m - i_l}{R_{sd} \cdot C_{vs}} = \mu_{sf}$$

Heterogeneous:

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

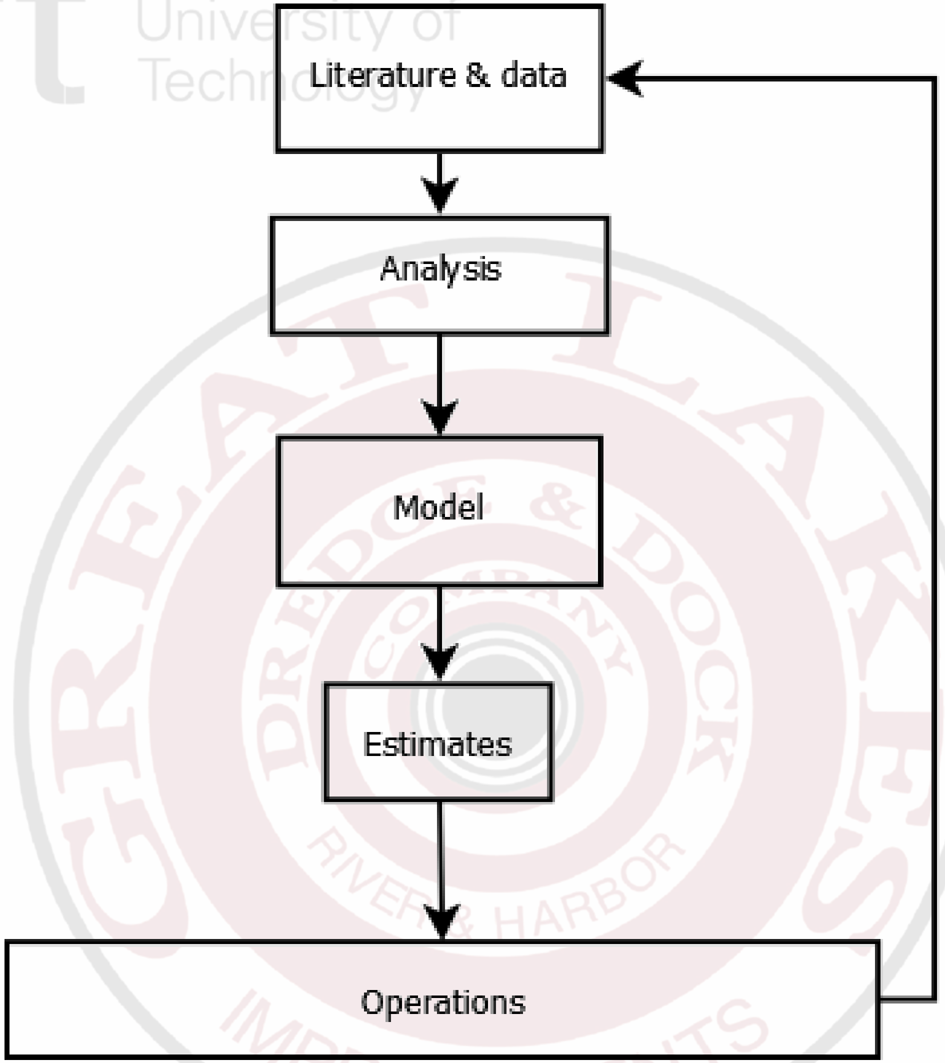
$$= \frac{v_t \cdot \left(1 - \frac{C_{vs}}{\kappa_C}\right)^{\beta}}{v_{ls}} + 8.5^2 \cdot \left(\frac{1}{\lambda_l}\right) \cdot \left(\frac{v_t}{\sqrt{g \cdot d}}\right)^{10/3} \cdot \left(\frac{(\nu_l \cdot g)^{1/3}}{v_{ls}}\right)^2$$

The Framework – Procedure



Why we need models

- Models apply scientific theory to real-life circumstances
  - Reasonable input and assumptions
  - Useful outputs



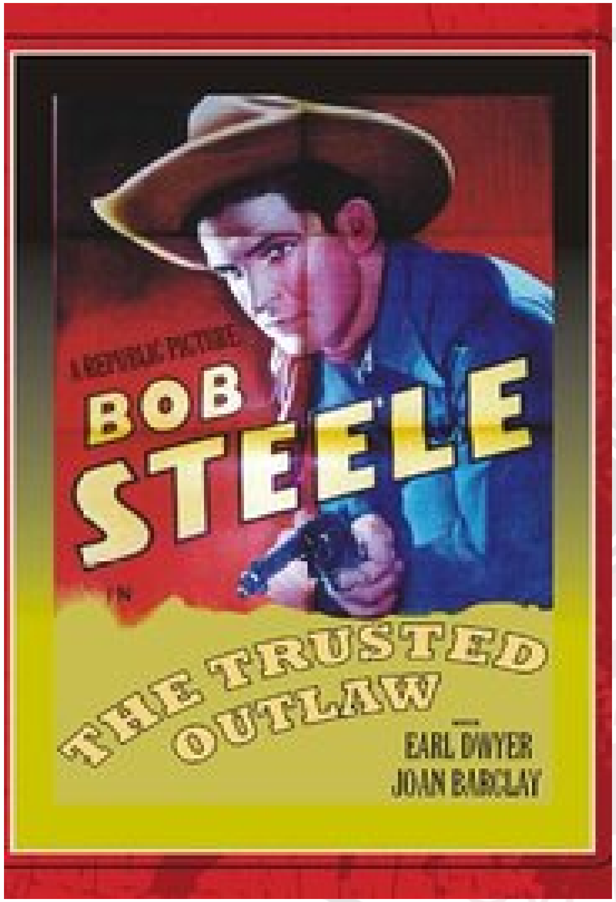
Implementing the model

- Useful to engineers
- There is a tradeoff between precision (accuracy) and complexity
- In engineering we often need to run many scenarios quickly

Implementing the model

The implementation in software must be:

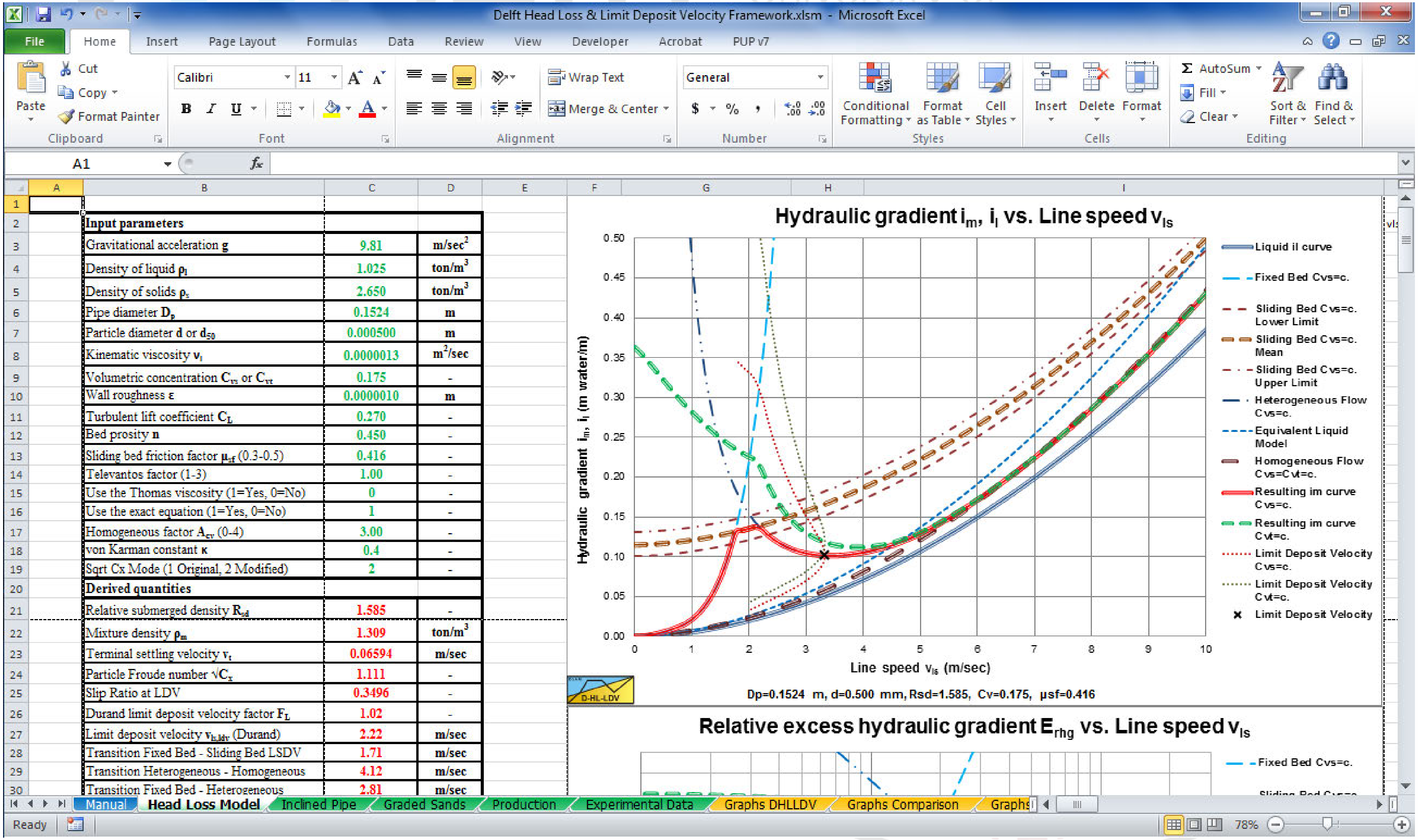
- Accessible
- Correct
- Trusted



2 implementations of the DHLLDV model

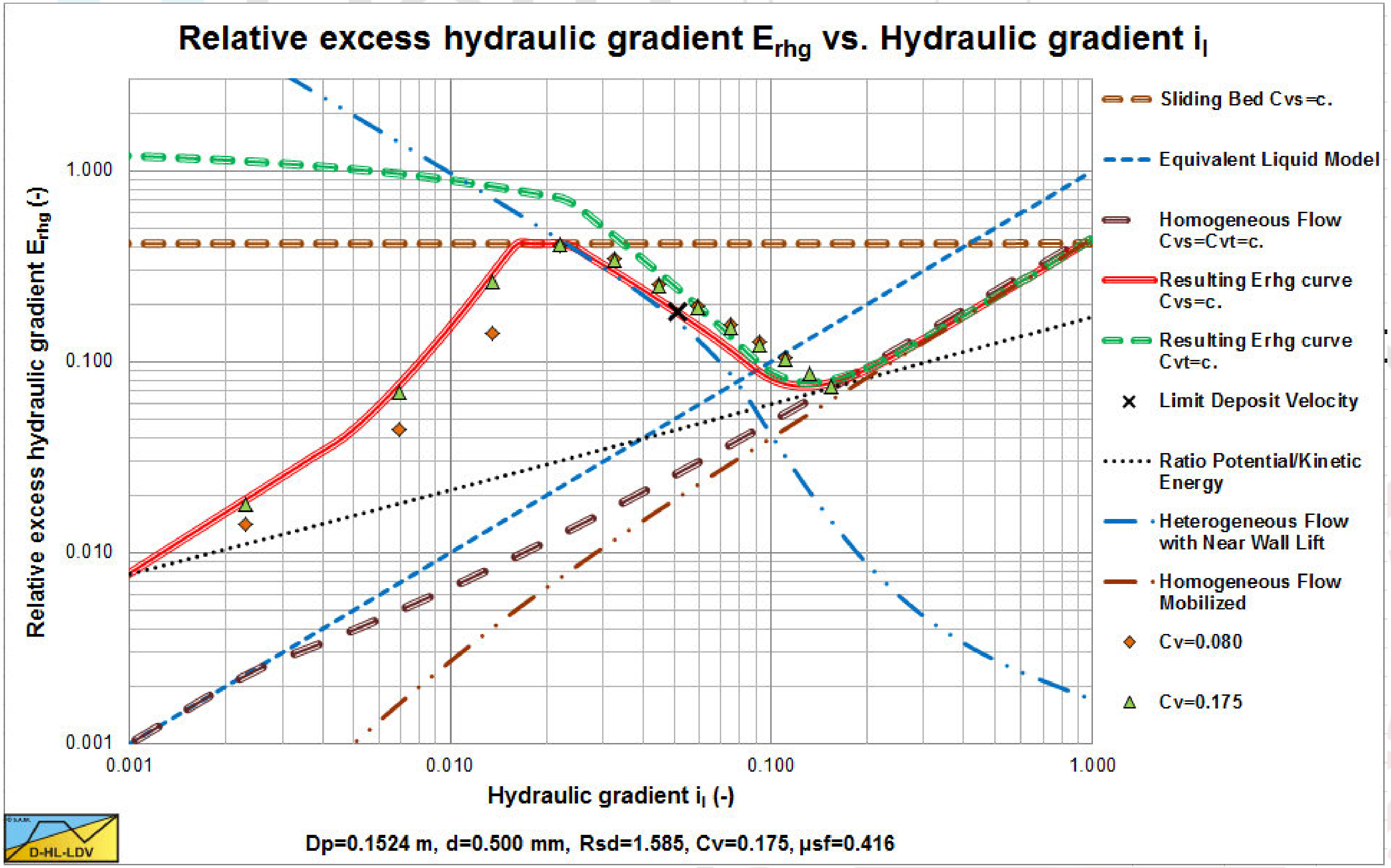
- The book and papers are the theory, the procedure in the book is the model
- The implementations supplement the book for users (including the authors)

Excel Spreadsheet



2	**Input parameters**		
3	Gravitational acceleration g	9.81	m/sec^2
4	Density of liquid ρ_l	1.025	ton/m^3
5	Density of solids ρ_s	2.650	ton/m^3
6	Pipe diameter D_p	0.1524	m
7	Particle diameter d or d_{50}	0.000500	m
8	Kinematic viscosity ν_l	0.0000013	m^2/sec
9	Volumetric concentration C_{vs} or C_{vt}	0.175	-
10	Wall roughness ε	0.0000010	m
11	Turbulent lift coefficient C_L	0.270	-
12	Bed prosity n	0.450	-
13	Sliding bed friction factor μ_{sf} (0.3-0.5)	0.416	-
14	Televantos factor (1-3)	1.00	-
15	Use the Thomas viscosity (1=Yes, 0=No)	0	-
16	Use the exact equation (1=Yes, 0=No)	1	-
17	Homogeneous factor A_{cv} (0-4)	3.00	-
18	von Karman constant κ	0.4	-
19	Sqrt Cx Mode (1 Original, 2 Modified)	2	-
20	**Derived quantities**		
21	Relative submerged density R_{sd}	1.585	-
22	Mixture density ρ_m	1.309	ton/m^3
23	Terminal settling velocity v_t	0.06594	m/sec
24	Particle Froude number $\sqrt{C_x}$	1.111	-
25	Slip Ratio at LDV	0.3496	-
26	Durand limit deposit velocity factor F_L	1.02	-
27	Limit deposit velocity $v_{ls,ldv}$ (Durand)	2.22	m/sec
28	Transition Fixed Bed - Sliding Bed LSDV	1.71	m/sec
29	Transition Heterogeneous - Homogeneous	4.12	m/sec
30	Transition Fixed Bed - Heterogeneous	2.81	m/sec

Excel Spreadsheet

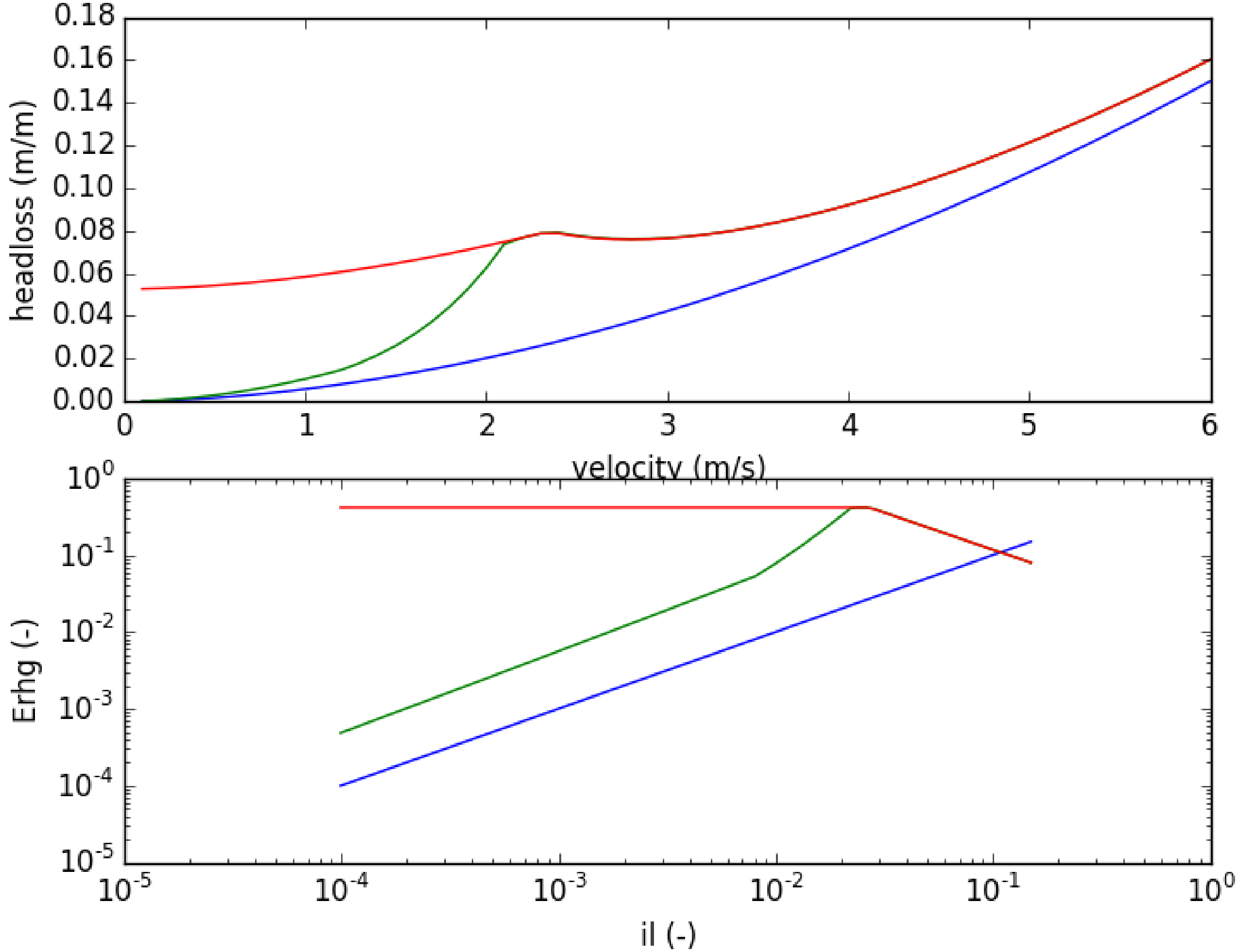


Python Code

homogeneous

```
'''
homogeneous.py - calculations of newtonian (fluid), homogeneous and pseudo-homogeneous flow.

Created on Oct 7, 2014

@author: RCRamsdell
'''

from math import log, exp
from DHLLDV_constants import gravity

Acv = 3.    #coefficient homogeneous regime, advised in section 8.7.
kvK = 0.4 #von Karman constant

def pipe_reynolds_number(vls, Dp, nu):
    """
    Return the reynolds number for the given velocity, fluid & pipe
    vls: velocity in m/sec
    Dp: pipe diameter in m
    nu: fluid kinematic viscosity in m2/sec
    """
    return vls*Dp/nu

def swamee_jain_ff(Re, Dp, epsilon):
    """
    Return the friction factor using the Swaamee-Jain equation.
    Re: Reynolds number
```

Python Code



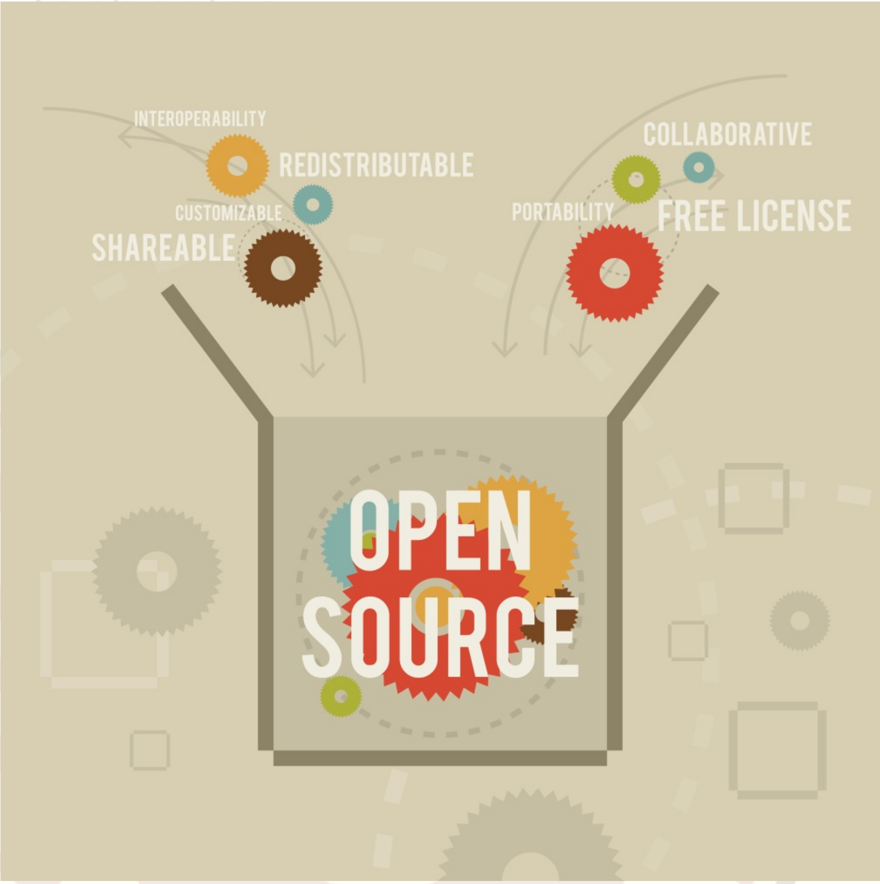
headloss (m/m)
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
1
2
3
4
5
6
velocity (m/s)
Erhg (-)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
il (-)

Why would you use my code?

- It must be accessible
  - Ease of access
  - Unencumbered
- It must be reliable
  - Correct
  - Secure
  - Verifiable
  - The code must go on!

Accessible – Open Source

- Free redistribution
- Source Code
- Derived works
- Non-discrimination
- Distribution of rights



Accessible – Python

- “Executable pseudocode”
- Simple, clear syntax, little boilerplate
- Quick write-run-rewrite cycle
- Lots of third-party tools to aid development

python powered

```
print("Hello, world!")
```

Example: Homogeneous Flow

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

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

Where:

$$\frac{\delta_V}{d} = \frac{11.6 \cdot v_l}{\sqrt{\frac{\lambda_l}{8}} \cdot v_{ls}.\, d}$$

Example: Homogeneous Flow

```python
def Erhg(vls, Dp, d, epsilon, nu, rhol, rhos, Cvs):
    """Return the Erhg value for homogeneous flow.
    Use the Talmon (2013) 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)
    lambda1 = swamee_jain_ff(Re, Dp, epsilon)
    Rsd = (rhos-rhol)/rhol
    rhom = rhol+Cvs*(rhos-rhol)
      deltav_to_d  =  min((11.6*nu)/((lambda1/8)**0.5*vls*d),  1) #eqn 8.7-7

    sb = ((Acv/kvK)*log(rhom/rhol)*(lambda1/8)**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
```

Accessible – Git Source control

- Online hosting: https://github.com/rcriii42/DHLLDV

```
c:\Data\workspace>git clone
https://github.com/rcriii42/DHLLDV
Cloning into 'DHLLDV'...
remote: Counting objects: 173, done.
remote: Compressing objects: 100% (3/3), done.
Receiving objects: 85% (148/173) (delta 0), pack-
reused 170
Receiving objects: 100% (173/173), 177.36 KiB | 0
bytes/s, done.
Resolving deltas: 100% (98/98), done.
Checking connectivity... done.
c:\Data\workspace>
```

Git – Status

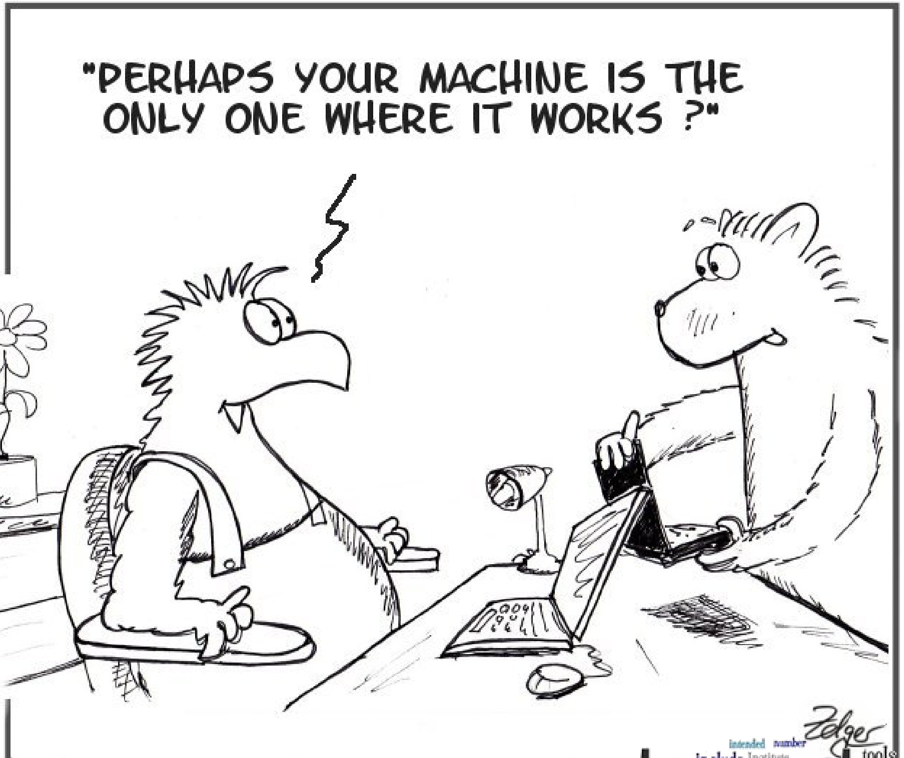
```
c:\Data\workspace\DHLLDV>git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean

c:\Data\workspace\DHLLDV>
```

```
c:\Data\workspace\DHLLDV_dev>git log --
pretty=format:"%h - %an, %ar : %s"
86c2eb2 - Robert Ramsdell, 14 minutes ago : More
cleanup of homogeneous.py
ab1721b - Robert Ramsdell, 9 hours ago : Fixed comments
and added eqn # to homogeneous.py
70fb6b8 - Robert Ramsdell, 3 weeks ago : Added
Cvs_Erhg_graded, runs with tests that don't pass
...
```

Git - Updating

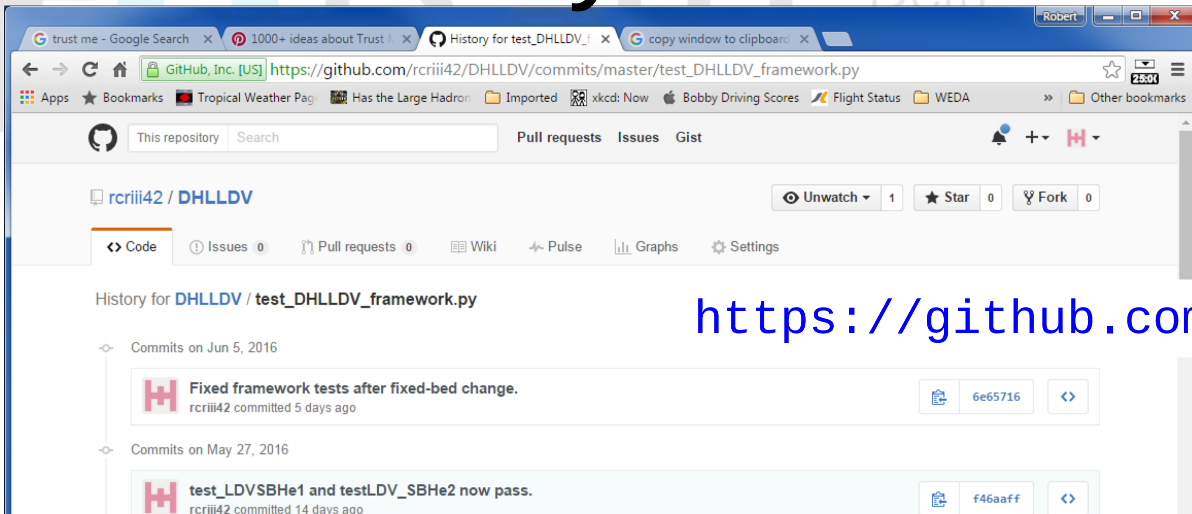
```
c:\Data\workspace\DHLLDV_dev>git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused
0
Unpacking objects: 100% (3/3), done.
From https://github.com/rcriii42/DHLLDV
ab1721b..86c2eb2 master -> origin/master
Updating ab1721b..86c2eb2
Fast-forward
homogeneous.py | 18 +++++++++---------
1 file changed, 9 insertions(+), 9 deletions(-)
c:\Data\workspace\DHLLDV_dev>
```

Reliability



It works on my machine

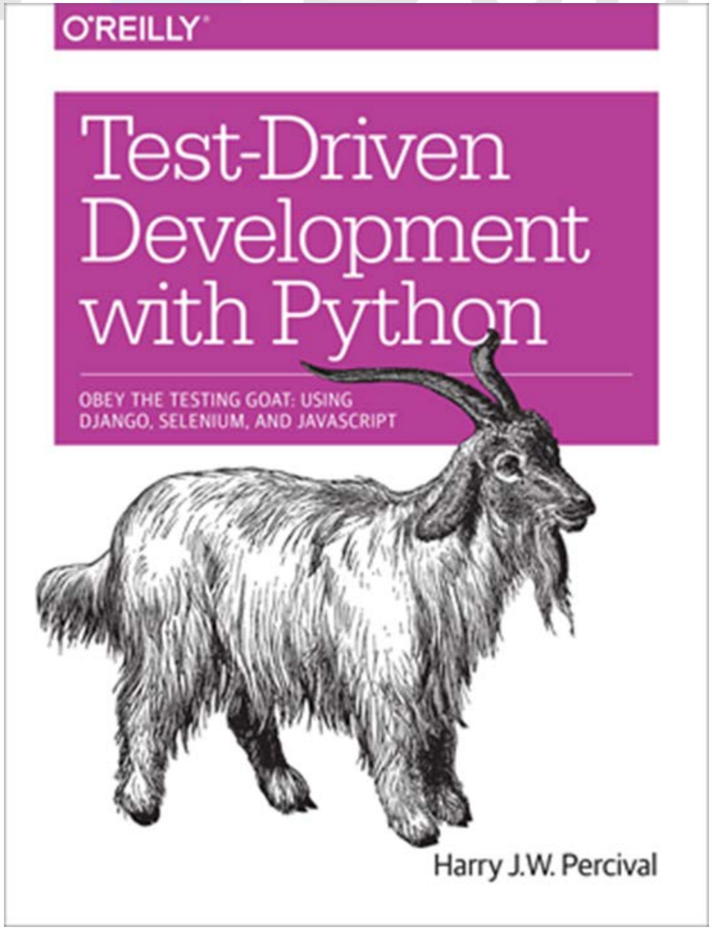
Reliability - Source Control



https://github.com/rcriii42/DHLLDV

```
c:\Data\workspace\DHLLDV_dev>git log --
pretty=format:"%h - %an, %ar : %s"
86c2eb2 - Robert Ramsdell, 14 minutes ago : More
cleanup of homogeneous.py
ab1721b - Robert Ramsdell, 9 hours ago : Fixed comments
and added eqn # to homogeneous.py
70fb6b8 - Robert Ramsdell, 3 weeks ago : Added
Cvs_Erhg_graded, runs with tests that don't pass
...
```

Reliability - Testing

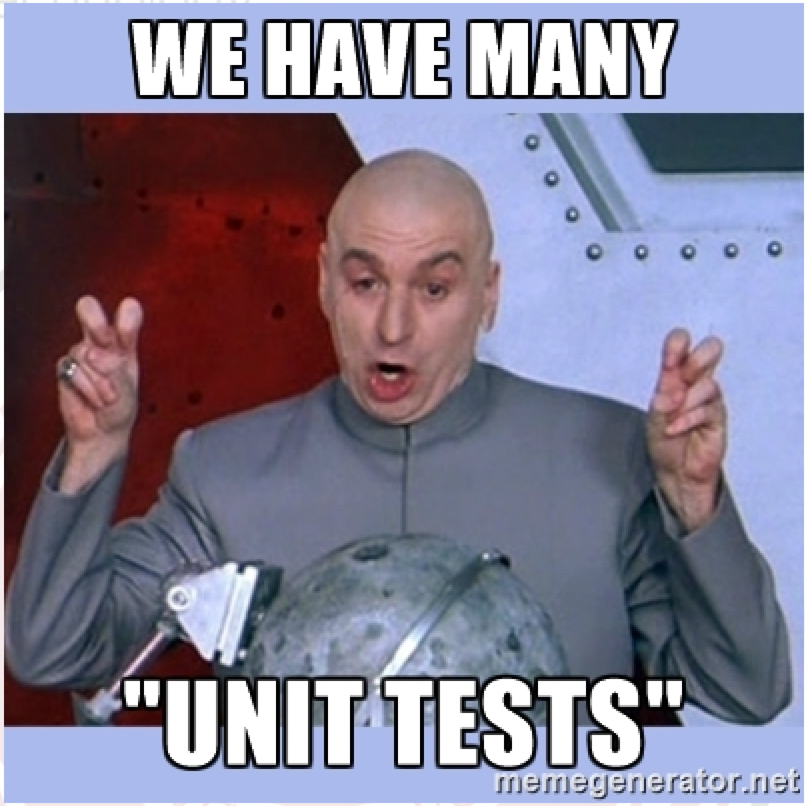


Test Driven Development:

A software **development** process that relies on the repetition of a very short **development** cycle: requirements are turned into very specific **test** cases, then the software is improved to pass the new **tests**, only.

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.



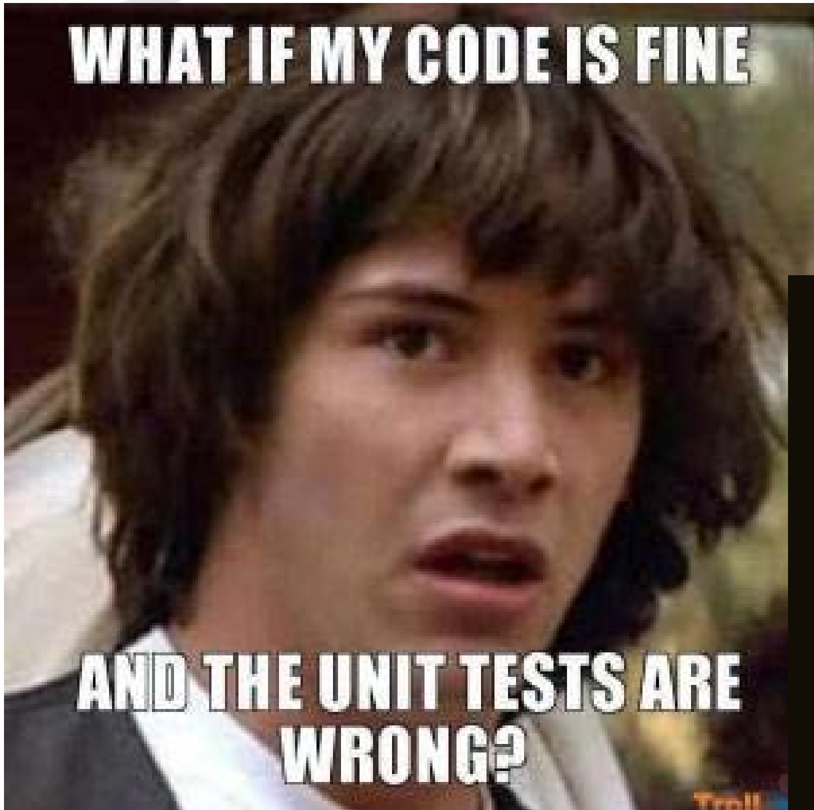
Example: Homogeneous Flow

```
'''
Created on Oct 7, 2014

@author: RCRamsdell

Testing the homogeneous module and water constants
'''
import unittest
import homogeneous
import DHLLDV_constants


...
def test_Erhg(self):
vls = 3.0
Dp =0.5
d=0.075/1000
epsilon = DHLLDV_constants.steel_roughness
rhol = DHLLDV_constants.water_density[20]
nu = DHLLDV_constants.water_viscosity[20]
rhos = 2.65
Cvs = 0.25
self.assertAlmostEqual(homogeneous.Erhg(vls, Dp, d, epsilon, nu, rhol,
                                          rhos, Cvs), 0.0118762)
```

Running Tests

```
c:\Data\workspace\DHLLDV_dev>test_homogeneous.py
.......
----------------------------------------------------------------------
Ran 7 tests in 0.012s

OK
```

```
c:\Data\workspace\DHLLDV_dev>nosetests
FF...........................................
======================================================================
FAIL: testCvs_Erhg_graded_result (test_DHLLDV_framework.Test)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "c:\Data\workspace\DHLLDV_dev\test_DHLLDV_framework.py", line 264, in
testCvs_Erhg_graded_result
    self.assertAlmostEqual(Erhg, 0.0528735)
AssertionError: 0.002939348191883322 != 0.0528735 within 7 places


----------------------------------------------------------------------
Ran 45 tests in 0.293s

FAILED (failures=2)
```

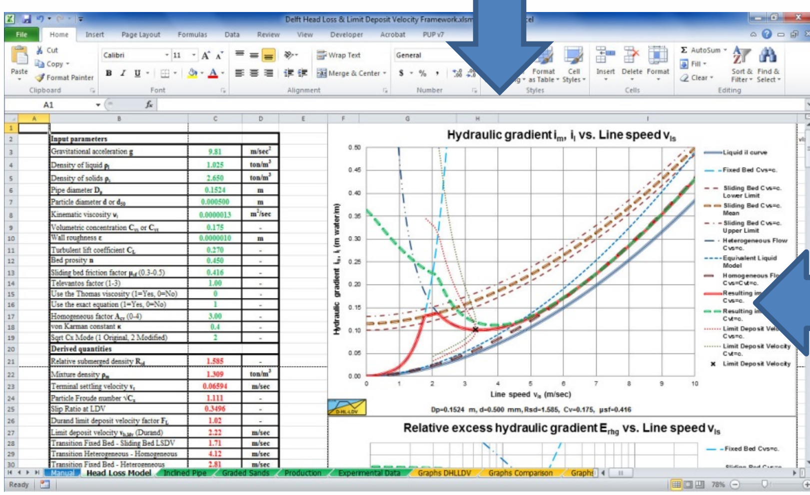

WHAT IF MY CODE IS FINE
AND THE UNIT TESTS ARE WRONG?


REFACTORED SOME STUFF
38
UNIT TESTS PENDING

What are we testing?

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

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




```
def Erhg(vls, Dp, d,      ilon, nu, rhol, rhos, Cvs):
    """Return the E       lue for homogeneous flow.
    Use the Talmon (20  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)
    lambda1 = swamee_jain_ff(Re, Dp, epsilon)
    Rsd = (rhos-rhol)/rhol
    rhom = rhol+Cvs*(rhos-rhol)
     deltav_to_d = min((11.6*nu)/((lambda1/8)**0.5*vls*d), 1)
#eqn 8.7-7

    sb = ((Acv/kvK)*log(rhom/rhol)*(lambda1/8)**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
```

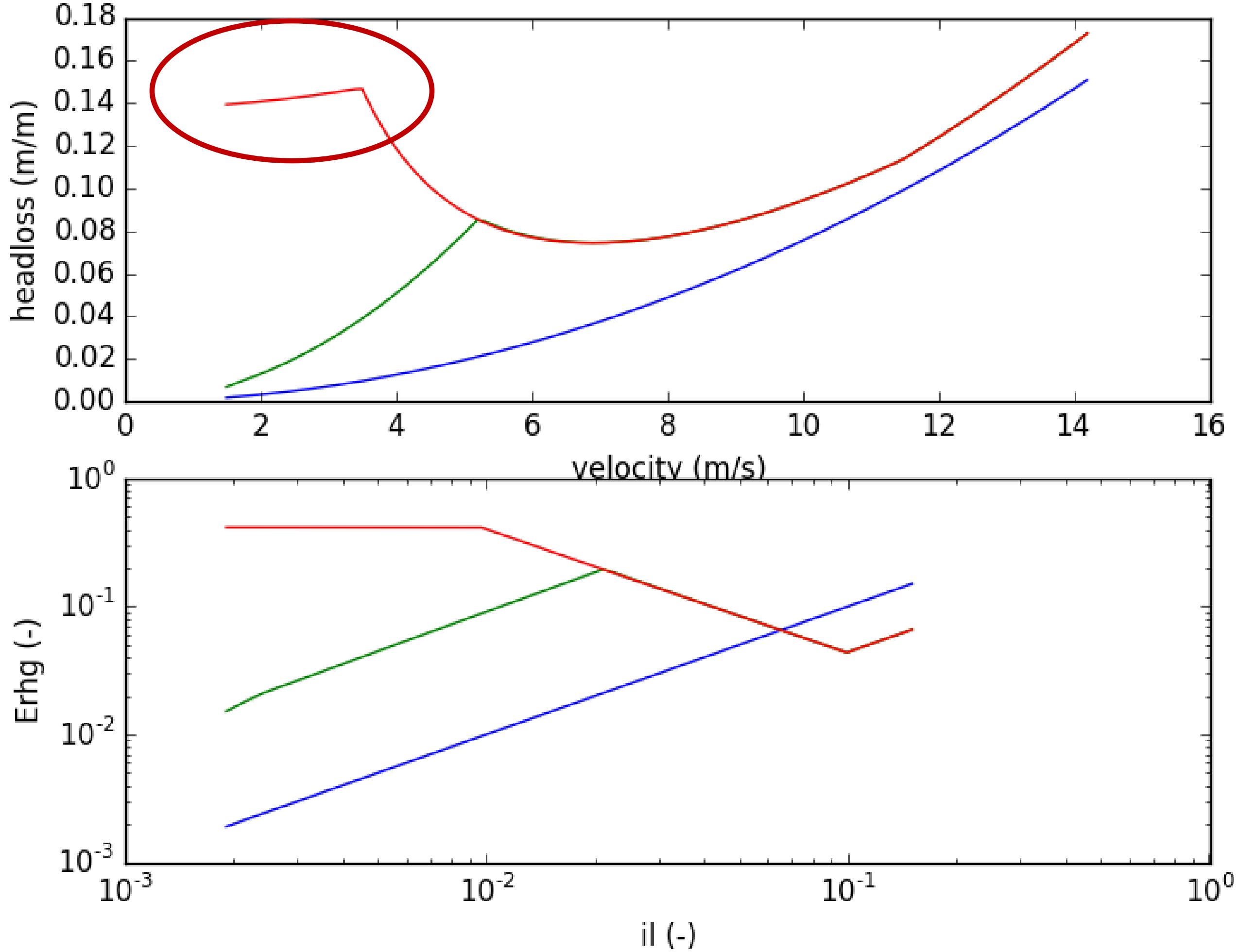
Code Coverage

- How to know testing is comprehensive!

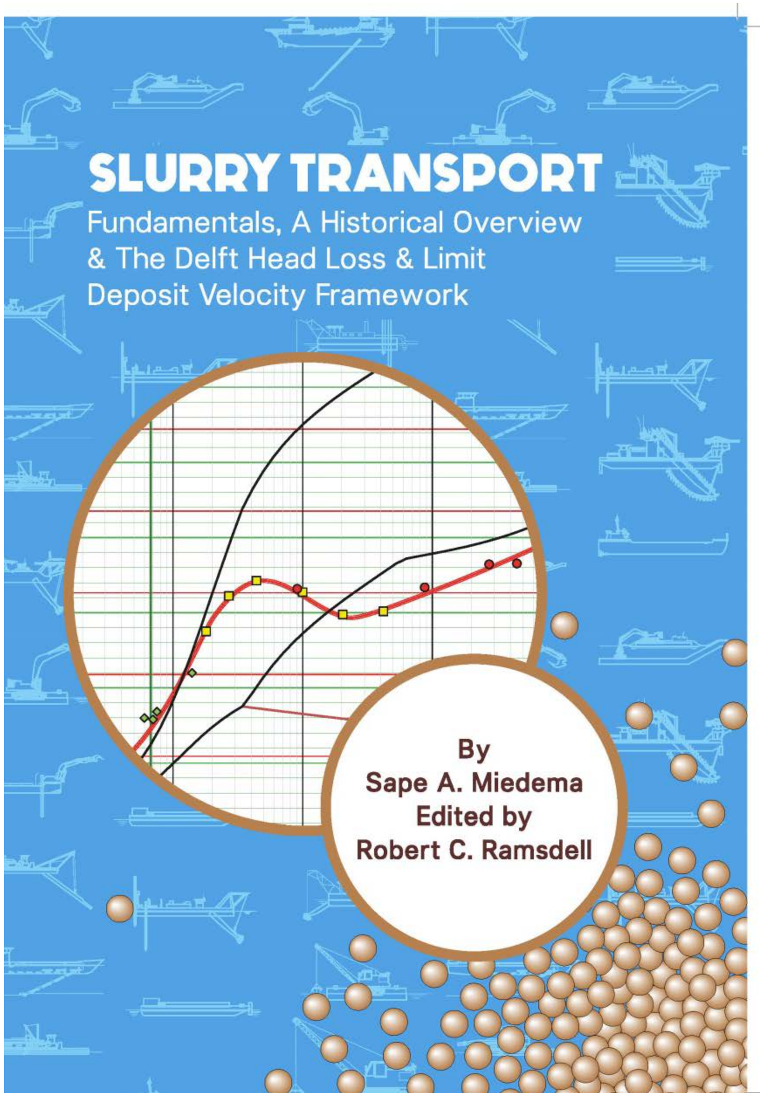
Code Coverage

```
c:\Data\workspace\DHLLDV_dev>nosetests --with-coverage

. . .

Name                      Stmts   Miss  Cover   Missing
---------------------------------------------------
DHLLDV_Utils.py              23      0   100%
DHLLDV_constants.py           7      0   100%
DHLLDV_framework.py         198     15    92%  40, 131, 237-248,
                                               262-263, 282, 369
heterogeneous.py             41      1    98%   96
homogeneous.py               39      4    90%   48-50, 72
stratified.py                68      1    99%   188
---------------------------------------------------
TOTAL                       376     21    94%
---------------------------------------------------
Ran 45 tests in 0.413s

FAILED (failures=2)
```

State of the Software



Conclusions



- Read the book
- Download the software: https://github.com/rcriii42/DHLLDV
- Run the tests: `nosetests`