Derivatives

Equation	Name	Parameters
$Q_t = aW_t^b$	At-a-station hydrology geometry (AHG)	$\boldsymbol{a}$ and $\boldsymbol{b}$
$Q_t = c(H_t - H_0)^d$	AHG for depth (rating curve)	$\boldsymbol{c}$, $\boldsymbol{d}$ and H_0
$Q_t = \frac{1}{n}(H_t - H_0)^{5/3}W_tS^{1/2}$	Manning's equation for height	$\boldsymbol{n}$ and H_0
$Q_t = \frac{1}{n}(A_0 + \delta A_t)^{5/3}W_t^{-2/3}S^{1/2}$	Manning's equation for area with constant n	$\boldsymbol{n}$ and A_0
$Q_t = \frac{1}{n_t}(A_0 + \delta A_t)^{5/3}W_t^{-2/3}S^{1/2}$ $n_t = n_0\left(\frac{A_0+\delta A_t}{W_t}\right)^p$	Manning's equation for area with power law n	n_0, $\boldsymbol{p}$ and A_0
$Q_t = \frac{1}{n_t}(A_0 + \delta A_t)^{5/3}W_t^{-2/3}S^{1/2}$ $n_t = n_\infty\left(1+\frac{5}{6}\left[\frac{\sigma_Z W_t}{A_0+\delta A_t}\right]^2\right)$	Manning's equation for are with spatial variability n	n_∞, σ_Z and A_0
$Q_t = \frac{1}{n_t}\left([H_t - B]\frac{r}{1+r}\right)^{5/3}W_tS^{1/2}$ $n_t = n_b\left(1+\log\left[\frac{H_b-B}{H_t-B}\right]\right)$	MOMMA	H_b, n_b, $\boldsymbol{B}$ and $\boldsymbol{r}$

At-a-station hydrology geometry (AHG)

(Q-aW^b)^2

FlowLawVariants['AHGW']=AHGW(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

At - a - station hydrology geometry (AHG) $Q_t = aW_t^b$

$$\varepsilon = \sum (Q_t - aW_t^b)^2 = (Q_1 - aW_1^b)^2 + (Q_2 - aW_2^b)^2 \ldots + \ldots (Q_{n_t} - aW_{n_t}^b)^2$$

$$\frac{d}{da}\left[(Q_1 - aW_1^b)^2\right] = 2(Q_1 - aW_1^b)\cdot\left(\frac{d}{da}\left[Q_1 - aW_1^b\right]\right)$$

$$= 2(Q_1 - aW_1^b)\cdot\left(\frac{d}{da}\left[Q_1\right] - (W_1^b)\right)$$

$$= 2(Q_1 - aW_1^b)\cdot(0 - W_1^b)$$

$$= -2(Q_1 - aW_1^b)(W_1^b)$$

$$\boxed{\frac{d\varepsilon}{da} = -2\cdot\sum_t (Q_t - aW_t^b)(W_t^b)}$$

$$\frac{d}{db}\left[(Q_1 - aW_1^b)^2\right] = 2(Q_1 - aW_1^b)\cdot\frac{d}{db}\left[Q_1 - aW_1^b\right]$$

$$= 2(Q_1 - aW_1^b)\cdot\frac{d}{db}\left[Q_1\right] - \frac{d}{db}\left[aW_1^b\right]$$

$$= 2(Q_1 - aW_1^b)\cdot(0 - a\cdot \ln(W_1)\cdot W_1^b)$$

$$= -2(Q_1 - aW_1^b)(aW_1^b\cdot \ln(W_1))$$

$$= -2aW_1^b \ln(W_1^b)(Q_1 - aW_1^b)$$

$$\boxed{\frac{d\varepsilon}{db} = -2\cdot\sum_t aW_t^b \ln(W_t^b)(Q_t - aW_t^b)}$$

Derivatives

A: params [0]

```
dydp[0]=-2*sum( (Qt-params[0]*self.W**params[1])*self.W**params[1] )
```

b: params [1]

```
dydp[1]=-2*sum( (Qt-params[0]*self.W**params[1])*(log(self.W**params[1]))*(params[0]*self.W**params[1]))
```

```
[8]: # use finite-difference to check jacobian at initial parameters
     flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496      983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584     1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.5, 0.25]
Jacobian at initial parameters= [-202306.52915697 -135486.54350111]
```

```
[10]: #look at deltaQ for first parameter, a
      Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.5, 0.25])
      ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
      da=.05
      Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.5+da, 0.25])
      ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

      dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

      print(dObjFunc_da)
```

```
-202275.8944106847
```

AHG for depth (rating curve)

(Q-(c(t-H)^d))^2

FlowLawVariants['AHGD']=AHGD(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

AHG for depth (rating curve) $Q_t = c(H_t - H_0)^d$

$$\varepsilon = \sum \left[Q_t - c(H_t - H_0)^d \right]^2$$

$$\frac{d}{dc}\left[\left(Q_t - c(H_t - H_0)^d\right)^2\right] = 2\left(Q_t - c(H_t - H_0)^d\right) \cdot \frac{d}{dc}\left[Q_t - c(H_t - H_0)^d\right]$$

$$= 2\left(Q_t - c(H_t - H_0)^d\right) \cdot \frac{d}{dc}\left[Q_t\right] - (H_t - H_0)^d \cdot \frac{d}{dc}\left[c\right]$$

$$= 2\left(Q_t - c(H_t - H_0)^d\right)\left(0 - (H_t - H_0)^d\right)$$

$$= -2\left(Q_t - c(H_t - H_0)^d\right)(H_t - H_0)^d$$

$$\boxed{\frac{d\varepsilon}{dc} = -2 \cdot \sum_t \left(Q_t - c(H_t - H_0)^d\right)(H_t - H_0)^d}$$

Derivatives

$$\frac{d}{dH_0}\left[\left(Q_1 - c(H_1 - H_0)^d\right)^2\right] = 2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot \frac{d}{dH_0}\left[Q_1 - c(H_1 - H_0)^d\right]$$

$$= 2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot \frac{d}{dH_0}\left[Q_1\right] - c \cdot \frac{d}{dH_0}\left[(H_1 - H_0)^d\right]$$

$$= 2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot \left(0 - c \cdot d(H_1 - H_0)^{d-1} \cdot \frac{d}{dH_0}\left[H_1 - H_0\right]\right)$$

$$= -2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot cd\left(H_1 - H_0\right)^{d-1} \cdot \left(\frac{d}{dH_0}\left[H_1\right] - \frac{d}{dH_0}\left[H_0\right]\right)$$

$$= -2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot cd\left(H_1 - H_0\right)^{d-1} \cdot (0 - 1)$$

$$= -2\left(Q_1 - c(H_1 - H_0)^d\right) \cdot -cd\left(H_1 - H_0\right)^{d-1}$$

$$= 2cd\left(Q_1 - c(H_1 - H_0)^d\right)\left(H_1 - H_0\right)^{d-1}$$

$$\boxed{\frac{d\varepsilon}{dH_0} = 2\sum_t cd\left(Q_t - c(H_t - H_0)^d\right)\left(H_t - H_0\right)^{d-1}}$$

$$\frac{d}{d\alpha}\left[\left(Q_1 - c(H_1 - H_0)^\alpha\right)^2\right] = 2\left(Q_1 - c(H_1 - H_0)^\alpha\right) \cdot \frac{d}{d\alpha}\left[Q_1 - c(H_1 - H_0)^\alpha\right]$$

$$= 2\left(Q_1 - c(H_1 - H_0)^\alpha\right) \cdot \left(\frac{d}{d\alpha}\left[Q_1\right] - c\frac{d}{d\alpha}\left[(H_1 - H_0)^\alpha\right]\right)$$

$$= 2\left(Q_1 - c(H_1 - H_0)^\alpha\right) \cdot \left(0 - c\ln(H_1 - H_0)(H_1 - H_0)^\alpha\right)$$

$$= -2\left(Q_1 - c(H_1 - H_0)^\alpha\right)\left(c\ln(H_1 - H_0)(H_1 - H_0)^\alpha\right)$$

$$= -2c\left(Q_1 - c(H_1 - H_0)^\alpha\right)\left(\ln(H_1 - H_0)(H_1 - H_0)^\alpha\right)$$

$$\boxed{\frac{d\varepsilon}{d\alpha} = -2\sum_t c\left(Q_t - c(H_t - H_0)^\alpha\right)\left(\ln(H_t - H_0)(H_t - H_0)^\alpha\right)}$$

c: params [0]

```
dydp[0]=-2*sum((self.H-params[1])**params[2]*(Qt-(self.H-
params[1])**params[2]*params[0]))
```

H: params[1]

```
dydp[1]=-2*sum((params[0]*params[2]*(Qt-params[0]*(self.H-
params[1])**params[2])*(self.H-params[1])**(params[2]-1)))
```

d: params [2]

```
dydp[2]=-2*sum((params[0]*(self.H-params[1])**params[2]*log(self.H-
params[1])*(Qt-params[0]*(self.H-params[1])**params[2])))
```

Derivatives

```
[12]: # use finite-difference to check jacobian at initial parameters
      flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496     983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [5.0, 594.2900519226707, 2.0]
Jacobian at initial parameters= [ -290658.28999986 -1176858.02282956 -1364564.51096613]
```

```
[14]: #look at deltaQ for first parameter, a
      Q_initparams=flow_law_cal.FlowLaw.CalcQ([5.0, 594.2900519226707, 2.0])
      ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
      da=.05
      Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([5.0+da, 594.2900519226707, 2.0])
      ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

      dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

      print(dObjFunc_da)
```

```
-290609.17974464595
```

Manning's equation for height
(Q-(1/n)(t-H)^(5/3)W(S^(1/2)))^2
FlowLawVariants['HeightManning']=MWHCN(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

manning's equation for height $Q_t = \frac{1}{n}(H_t - H_0)^{5/3} W_t S^{1/2}$

$$\varepsilon = \sum_t \left(Q_t - \frac{1}{n}(H_t - H_0)^{5/3} W_t S^{1/2}\right)^2$$

$$\frac{d}{dn}\left[\left(Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right)^2\right] = 2\left(Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right) \cdot \frac{d}{dn}\left[Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right]$$

$$= 2\left(Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right) \cdot \left(\frac{d}{dn}[Q_1] - S^{1/2} W_1 (H_1 - H_0)^{5/3} \cdot \frac{d}{dn}\left[\frac{1}{n}\right]\right)$$

$$= 2\left(Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right) \cdot \left(0 + S^{1/2} W_1 (H_1 - H_0)^{5/3} \cdot \frac{\frac{d}{dn}[n]}{n^2}\right)$$

$$= 2\left(Q_1 - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right) \cdot \left(S^{1/2} W_1 (H_1 - H_0)^{5/3} \cdot \frac{1}{n^2}\right)$$

$$= \frac{2}{n_2}\left(Q - \frac{1}{n}(H_1 - H_0)^{5/3} W_1 S^{1/2}\right)\left(S^{1/2} W_1 (H_1 - H_0)^{5/3}\right)$$

$$\boxed{\frac{d\varepsilon}{dn}\ 2\sum_t \frac{1}{n^2}\left(Q - \frac{1}{n}(H_t - H_0)^{5/3} W_t S^{1/2}\right)\left(S^{1/2} W_t (H_t - H_0)^{5/3}\right)}$$

Derivatives

$$\frac{d}{dH_0}\left[\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)^2\right]=2\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)\cdot\frac{d}{dH_0}\left[Q-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right]$$

$$=2\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)\left(\frac{d}{dH_0}[Q_1]-\frac{W_1S^{1/2}}{n}\cdot\frac{d}{dH_0}\left[(H_1-H_0)^{5/3}\right]\right)$$

$$=2\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)\left(0-\frac{W_1S^{1/2}}{n}\cdot\frac{5}{3}(H_1-H_0)^{5/3-1}\cdot\frac{d}{dH_0}[H_1-H_0]\right)$$

$$=\frac{-10}{3n}\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)\left(W_1S^{1/2}(H_1-H_0)^{2/3}\cdot(0-1)\right)$$

$$=\frac{10W_1S^{1/2}}{3n}\left(Q_1-\frac{1}{n}(H_1-H_0)^{5/3}W_1S^{1/2}\right)(H_1-H_0)^{2/3}$$

$$\boxed{\frac{d\mathcal{E}}{dH_0}=\frac{10}{3}\sum\frac{W_tS^{1/2}}{n}\left(Q_t-\frac{1}{n}(H_t-H_0)^{5/3}W_tS^{1/2}\right)(H_t-H_0)^{2/3}}$$

n: params [0]

```
dydp[0]=2*sum((sqrt(self.S)*self.W*(self.H-params[1])**(5/3)*(Qt-(sqrt(self.S)*self.W*(self.H-params[1])**(5/3))/params[0]))/params[0]**2)
```

H: params [1]

```
dydp[1]=10*sum((sqrt(self.S)*self.W*(Qt-(sqrt(self.S)*self.W*(self.H-params[1])**(5/3))/params[0])*(self.H-params[1])**(2/3))/(3*params[0]))
```

```
[16]: # use finite-difference to check jacobian at initial parameters
      flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496     983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.03, 594.2900519226707]
Jacobian at initial parameters= [2.28087672e+08 4.64618815e+06]
```

```
[17]: #look at deltaQ for first parameter, a
      Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.03, 594.2900519226707])
      ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
      da=.005
      Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.03+da, 594.2900519226707])
      ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

      dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

      print(dObjFunc_da)
```

```
258419843.51107785
```

Manning's equation for area with constant n
(Q-(1/n)(t+A)^(5/3)W^(-2/3) S^(1/2))^2

Derivatives

FlowLawVariants['Constant-n']=MWACN(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

manning's equation for area w/ constant n $Q_t = \frac{1}{n}(A_0 + \delta A_t)^{5/3} W_t^{-2/3} S^{1/2}$

$$\mathcal{E} = \sum_t \left(Q_t - \frac{1}{n}(A_0 + \delta A_t)^{5/3} W_t^{-2/3} S^{1/2}\right)^2$$

$$\frac{d}{dn}\left[\left(Q_1 - \frac{1}{n}(A_0 + \delta A_1)^{5/3} W_1^{-2/3} S^{1/2}\right)^2\right] = \frac{d}{dn}\left[\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right)^2\right]$$

$$= 2\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right) \cdot \frac{d}{dn}\left[Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right]$$

$$= 2\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right) \cdot \left(\frac{d}{dn}[Q_1] - \frac{(A_0 + \delta A_1)^{5/3} S^{1/2}}{W_1^{2/3}} \frac{d}{dn}\left[\frac{1}{n}\right]\right)$$

$$= 2\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right) \cdot \left(0 + \frac{(A_0 + \delta A_1)^{5/3} S^{1/2}}{W_1^{2/3}} \cdot \frac{\frac{d}{dn}[n]}{n^2}\right)$$

$$= 2\left(\frac{1}{W^{2/3} n^2}\right)\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right)(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}$$

$$= 2\left(S^{1/2}\right)\left(\frac{1}{W_1^{2/3} n^2}\right)(A_0 + \delta A_1)^{5/3}\left(Q - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right)$$

$$\boxed{\frac{d\mathcal{E}}{dn} = 2\sum_t \left(S^{1/2}\right)\left(\frac{1}{W_t^{2/3} n^2}\right)(A_0 + \delta A_t)^{5/3}\left(Q - \frac{(A_0 + \delta A_t)^{5/3} \cdot S^{1/2}}{n \cdot W_t^{2/3}}\right)}$$

$$\frac{d}{dA_0}\left[\left(Q_1 - \frac{1}{n}(A_0 + \delta A_1)^{5/3} W_1^{-2/3} S^{1/2}\right)^2\right] = \frac{d}{dA_0}\left[\left(Q_1 - \frac{(A_0 + \delta A_1)^{5/3} \cdot S^{1/2}}{n \cdot W_1^{2/3}}\right)^2\right]$$

$$= 2\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right) \cdot \frac{d}{dA_0}\left[Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right]$$

$$= 2\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right)\left(\frac{d}{dA_0}[Q] - \frac{S^{1/2}}{W_1^{2/3} n} \cdot \frac{d}{dA_0}\left[(A_0 + \delta A_1)^{5/3}\right]\right)$$

$$= 2\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right)\left(0 - \frac{S^{1/2}}{W_1^{2/3} n} \cdot \frac{5}{3}(A_0 + \delta A_1)^{5/3 - 1} \cdot \frac{d}{dA_0}[A_0 + \delta A_1]\right)$$

$$= \frac{-10}{3} \cdot \frac{1}{W_1^{2/3} \cdot n}\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right) S^{1/2}(A_0 + \delta A_1)^{2/3}\left(\frac{d}{dA_0}[A_0] + \frac{d}{dA_0}[\delta A_1]\right)$$

$$= \frac{-10}{3} \cdot \frac{1}{W_1^{2/3} \cdot n}\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right) S^{1/2}(A_0 + \delta A_1)^{2/3}(1 + 0)$$

$$= -\frac{10}{3} \cdot \frac{1}{W_1^{2/3} \cdot n} \cdot S^{1/2}(A_0 + \delta A_1)^{2/3}\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right)$$

$$= -\frac{10}{3} \cdot \frac{S^{1/2}}{W_1^{2/3} \cdot n}(A_0 + \delta A_1)^{2/3}\left(Q - \frac{S^{1/2}(A_0 + \delta A_1)^{5/3}}{W_1^{2/3} \cdot n}\right)$$

$$\boxed{\frac{d\mathcal{E}}{dA_0} = -\frac{10}{3}\sum_t \frac{S^{1/2}}{W_t^{2/3} \cdot n}(A_0 + \delta A_t)^{2/3}\left(Q - \frac{S^{1/2}(A_0 + \delta A_t)^{5/3}}{W_t^{2/3} \cdot n}\right)}$$

n: params [0]

```
(2*(self.dA+params[1])**(5/3)*(Qt*sqrt(self.S)*self.W^(2/3)*params[0]-
self.S*(self.dA+params[1])**(5/3)))/(self.W**(4/3)*params[0]**3)
```

A: params [1]

```
(10*(self.dA+params[1])**(2/3)*(self.S*(self.dA+params[1])**(5/3)-
Qt*sqrt(self.S)*self.W**(2/3)*params[0]))/(3*self.W**(4/3)*params[0]**2)
```

Derivatives

[19]:
```python
# use finite-difference to check jacobian at initial parameters
flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496      983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.03, 228.48576653932]
Jacobian at initial parameters= [ 3.73569842e+08 -5.05574979e+04]
```

[21]:
```python
#look at deltaQ for first parameter, a
Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.03, 228.48576653932])
ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
da=.001
Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.03+da, 228.48576653932])
ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

print(dObjFunc_da)
```

```
365733130.3663589
```

Manning's equation for area with power law n
(Q-(1/(n((A+R)/W)^p)(A+R)^(5/3)(W^(-2/3)(S^(1/2)))))^2
FlowLawVariants['PowerLaw-n']=MWAPN(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

manning's equation for area with power law n $\quad Q_t = \frac{1}{n_0\left(\frac{A_0+\delta A_t}{w_t}\right)^p}\left(A_0+\delta A_t\right)^{5/3} w_t^{-2/3} s^{1/2}$

$$\varepsilon = \sum_t \left(Q_t - \frac{1}{n_0\left(\frac{A_0+\delta A_t}{w_t}\right)^p}\left(A_0+\delta A_t\right)^{5/3} w_t^{-2/3} s^{1/2}\right)^2$$

$$\frac{d}{dn_0}\left[\left(Q_1 - \frac{1}{n_0\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\left(A_0+\delta A_1\right)^{5/3} w_1^{-2/3} s^{1/2}\right)^2\right] = \frac{d}{dn_0}\left[\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)^2\right]$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)\cdot\frac{d}{dn_0}\left[\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)^2\right]$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)\cdot\left(\frac{d}{dn_0}\left[Q_1\right] - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\cdot\frac{d}{dn_0}\left[\frac{1}{n_0}\right]\right)$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)\cdot\left(0 + \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\cdot\frac{\frac{d}{dn_0}\left[n_0\right]}{n_0^2}\right)$$

$$= 2\,\frac{\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot s^{1/2}}{n_0 w_1^{2/3}\left(\frac{A_0+\delta A_1}{w_1}\right)^p}\right)\left(A_0+\delta A_1\right)^{5/3}\cdot s^{1/2}}{\left(\frac{A_0+\delta A_1}{w_1}\right)^p\cdot w_t^{2/3}\cdot n_0^2}$$

$$\boxed{\frac{d\varepsilon}{dn_0} = 2\sum_t \frac{\left(Q_1 - \frac{(A_0+\delta A_t)^{5/3}\cdot s^{1/2}}{n_0 w_t^{2/3}\left(\frac{A_0+\delta A_t}{w_t}\right)^p}\right)\left(A_0+\delta A_t\right)^{5/3}\cdot s^{1/2}}{\left(\frac{A_0+\delta A_t}{w_t}\right)^p\cdot w_t^{2/3}\cdot n_0^2}}$$

Derivatives

$$\frac{d}{dA_0}\left[\left(Q_1 - \frac{1}{n_0\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\left(A_0+\delta A_1\right)^{5/3} W_1^{-2/3} S^{1/2}\right)^2\right] = \frac{d}{dA_0}\left[\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot S^{1/2}}{n_0 W_1^{2/3}\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right)^2\right]$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot S^{1/2}}{n_0 W_1^{2/3}\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right)\cdot\frac{d}{dA_0}\left[Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot S^{1/2}}{n_0 W_1^{2/3}\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right]$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot S^{1/2}}{n_0 W_1^{2/3}\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right)\cdot\left(\frac{d}{dA_0}\left[Q_1\right] - \frac{S^{1/2}}{n_0\cdot W^{2/3}}\cdot\frac{d}{dA_0}\left[\frac{(A_0+\delta A_1)^{5/3}}{\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right]\right)$$

$$= 2\left(Q_1 - \frac{(A_0+\delta A_1)^{5/3}\cdot S^{1/2}}{n_0 W_1^{2/3}\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right)\left(0 - \frac{S^{1/2}}{n_0\cdot W^{2/3}}\cdot\frac{d}{dA_0}\left[\frac{(A_0+\delta A_1)^{5/3}}{\left(\frac{A_0+\delta A_1}{W_1}\right)^P}\right]\right)$$

$$\boxed{\frac{d\mathcal{E}}{dA_0} = 2\sum_t \frac{S^{1/2}(3p-5)(A_0+\delta A_t)^{2/3}\left(Q_t\cdot W_t^{2/3}\cdot n\left(\frac{A_0+\delta A_t}{W_t}\right)^P - S^{1/2}(A_0+\delta A_t)^{5/3}\right)}{3\,W_t^{4/3}\cdot n^2\left(\frac{A_0+\delta A_t}{W_t}\right)^{2P}}}$$

$$\boxed{\frac{d\mathcal{E}}{dP} = 2\sum_t \frac{2(A_0-\delta A_t)^{5/3}\cdot S^{1/2}\cdot \ln\left(\frac{A_0+\delta A_t}{W_t}\right)\left(Q - \frac{(A_0+\delta A_t)^{5/3}\cdot S^{1/2}}{\left(\frac{A_0+\delta A_t}{W_t}\right)^P\cdot W_t^{2/3}\cdot n}\right)}{\left(\frac{A_0+\delta A_t}{W_t}\right)^P\cdot W_t^{2/3}\cdot n}}$$

n: params [0]

```
(2*(self.dA+params[1])**(5/3)*sqrt(self.S)*(Qt-((self.dA+params[1])**(5/3)*sqrt(self.S))/(((self.dA+params[1])/self.W)**params[2]*self.W**(2/3)*params[0])))/(((self.dA+params[1])/self.W)**params[2]*self.W**(2/3)*params[0]**2)
```

A: params [1]

```
2*((sqrt(self.S)*params[2]*(params[1]+self.dA)**(2/3))/(self.W**(2/3)*params[0]*((params[1]+self.dA)/self.W)**params[2])-(5*sqrt(self.S)*(params[1]+self.dA)**(2/3))/(3*self.W**(2/3)*params[0]*((params[1]+self.dA)/self.W)**params[2]))*(Qt-(sqrt(self.S)*(params[1]+self.dA)**(5/3))/(self.W**(2/3)*params[0]*((params[1]+self.dA)/self.W)**params[2]))
```

P: params [2]

```
(2*(self.dA+params[1])**(5/3)*sqrt(self.S)*log((self.dA+params[1])/self.W)*(Qt-((self.dA+params[1])**(5/3)*sqrt(self.S))/(((self.dA+params[1])/self.W)**params[2]*self.W**(2/3)*params[0])))/(((self.dA+params[1])/self.W)**params[2]*self.W**(2/3)*params[0])
```

Derivatives

[23]:
```
# use finite-difference to check jacobian at initial parameters
flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496     983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.03, 228.48576653932, 1]
Jacobian at initial parameters= [ 2.80927282e+08 -1.62141514e+04  4.30650240e+06]
```

[24]:
```
#look at deltaQ for first parameter, a
Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.03, 228.48576653932, 1])
ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
da=.001
Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.03+da, 228.48576653932, 1])
ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

print(dObjFunc_da)
```

```
273482242.071148
```

Manning's equation for area with spatial variability n
(Q-(((1/(n(1+(5/6)((ZW)/(A+R))^2))(A+R)^(5/3)W^(-2/3)S^(1/2)))))^2
FlowLawVariants['MWAVN']=MWAVN(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])

manning's equation with spatial variability n $Q_t = \frac{1}{n_\infty \left(1 + \frac{5}{6}\left[\frac{\sigma_z W_t}{\delta A_t + A_0}\right]^2\right)} \left(A_0 + \delta A_t\right)^{5/3} \cdot W_t^{-2/3} \cdot S^{1/2}$

$$\boxed{\frac{d\mathcal{E}}{d n_\infty} = 2 \sum_t \frac{(\delta A_t + A_0)^{5/3} \cdot S^{1/2} \left(Q_t - \frac{(\delta A_t + A_0)^{5/3} \cdot S^{1/2}}{W_t^{2/3} \cdot \left(\frac{5 W_t^2 \cdot \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right) \cdot n_\infty}\right)}{W_t^{2/3} \cdot \left(\frac{5 W_t^2 \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right) n_\infty^2}}$$

$$\boxed{\frac{d\mathcal{E}}{d\sigma_z} = \frac{10}{3} \sum_t \frac{S^{1/2} \cdot W_t^{4/3} \cdot \sigma_z \left(Q_t - \frac{(\delta A_t + A_0)^{5/3} \cdot S^{1/2}}{W_t^{2/3} \cdot n_\infty \left(\frac{5 W_t^2 \cdot \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right)}\right)}{(\delta A_t + A_0)^{1/3} \cdot n_\infty \left(\frac{5 W_t^2 \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right)^2}}$$

$$\boxed{\frac{d\mathcal{E}}{d A_0} = -2 \sum_t \frac{S^{1/2} \left(\frac{5 W_t^2 \sigma_z^2}{3(A_0 + \delta A_t)^{4/3}} + \frac{5(A_0 + \delta A_t)^{2/3}\left(\frac{5 W_t^2 \cdot \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right)}{3}\right) \left(Q_t - \frac{S^{1/2}(A_0 + \delta A_t)^{5/3}}{W^{2/3} \cdot n_\infty \left(\frac{5 W_t^2 \cdot \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right)}\right)}{W^{2/3} \cdot n_\infty \left(\frac{5 W_t^2 \sigma_z^2}{6(\delta A_t + A_0)^2} + 1\right)^2}}$$

Derivatives

n: params [0]

```
(2*(params[1]+self.dA)**(5/3)*sqrt(self.S)*(Qt-((params[1]+self.dA)**(5/3)*sqrt(self.S))/(self.W**(2/3)*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1)*params[0])))/(self.W**(2/3)*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1)*params[0]**2)
```

A: params [1]

```
2*(-(5*sqrt(self.S)*(params[1]+self.dA)**(2/3))/(3*self.W**(2/3)*params[0]*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1))-(5*sqrt(self.S)*self.W**(4/3)*params[2]**2)/(3*params[0]*(params[1]+self.dA)**(4/3)*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1)**2))*(Qt-(sqrt(self.S)*(params[1]+self.dA)**(5/3))/(self.W**(2/3)*params[0]*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1)))
```

Z: params [2]

```
(10*sqrt(self.S)*self.W**(4/3)*params[2]*(Qt-((params[1]+self.dA)**(5/3)*sqrt(self.S))/(self.W**(2/3)*params[0]*((5*self.W**2*params[2]**2)/(6*(params[1]+self.dA)**2)+1))))/(3*(params[1]+self.dA)**(1/3)*params[0]*((5*self.W**2* params[2]**2)/(6*(params[1]+self.dA)**2)+1)**2)
```

```
[26]: # use finite-difference to check jacobian at initial parameters
flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496     983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.03, 228.48576653932, 1]
Jacobian at initial parameters= [ 3.34416779e+08 -5.76537632e+04  4.46495672e+06]
```

```
[27]: #look at deltaQ for first parameter, a
Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.03, 228.48576653932, 1])
ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
da=.001
Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.03+da, 228.48576653932, 1])
ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

print(dObjFunc_da)
```

```
326153371.5985007
```

MOMMA
(Q-(((1/(n(1+log((H-B)/(t-B)))((t-B) r/(r+1)))^(5/3)WS^(1/2))))^2
FlowLawVariants['MOMMA']=MWAVN(ReachDict['dA'],ReachDict['w'],ReachDict['S'],ReachDict['H'])
***Work for this one was incorrect- code below is right derivative**

Derivatives

```
[29]: # use finite-difference to check jacobian at initial parameters
      flow_law_cal.CalibrateReach(verbose=False,suppress_warnings=True)
```

```
Qtrue= [1179.496     983.418189 1049.714979  589.417387  349.935483  250.048693
  281.160297  293.809213  280.033016  282.760474  266.773627  261.021762
  277.009424  272.616     262.584    1758.892711  751.331161  716.124809
  788.129459  721.922579  567.415925  463.411476  374.754097  349.201929
  324.541016  344.922189 1702.656     652.306139  617.033172  601.153676
  435.486041  384.291118  335.415967  355.723488 1827.51237   714.19614
  502.688192  381.650106  357.305159  348.941074 1546.541436  739.273459
  574.939049  487.412039]
width= [236.9919221  224.23808981 245.1335819  200.44040141 180.18319887
 168.41548937 175.99267721 150.7521055  176.63411365 169.72716522
 164.36749667 159.82622073 162.51055078 164.44875949 165.74179826
 239.44922007 221.12524074 209.5237898  211.72287739 206.18534676
 177.4674591  177.620412   187.23701307 181.07810814 179.18553736
 175.35080953 269.63381951 205.12988851 210.8995567  190.82102341
 195.94931663 177.86816793 181.92169271 177.26998045 267.49292103
 214.98942721 199.40748792 175.89974428 180.80838176 179.58413844
 260.25408611 193.50851918 193.58239075 189.65066555]
initial flow law parameters= [0.03, 228.48576653932, 1]
Jacobian at initial parameters= [ 3.34416779e+08 -5.76537632e+04  4.46495672e+06]
```

```
[30]: #look at deltaQ for first parameter, a
      Q_initparams=flow_law_cal.FlowLaw.CalcQ([0.03, 228.48576653932, 1])
      ObjFunc=sum( (ReachDict['Qtrue']-Q_initparams)**2 )
      da=.001
      Q_perturbed_params=flow_law_cal.FlowLaw.CalcQ([0.03+da, 228.48576653932, 1])
      ObjFunc_perturbed_params=sum( (ReachDict['Qtrue']-Q_perturbed_params)**2 )

      dObjFunc_da=(ObjFunc_perturbed_params-ObjFunc)/da

      print(dObjFunc_da)
```

```
326153371.5985007
```

n: params [0]

```
(2*sqrt(self.S)*self.W* params[3]**(5/3)*(self.H-params[2])**(5/3)*(Qt-(sqrt(self.S)*self.W*params[3]**(5/3)*(self.H-params[2])**(5/3))/((params[3]+1)**(5/3)*(log((params[1]- params[2])/(self.H-params[2]))+1)* params[0])))/(( params[3]+1)**(5/3)*(log((params[1]-params[2])/(self.H-params[2]))+1)*params[0]**2)
```

H: params [1]

```
(2*sqrt(self.S)*self.W* params[3]**(5/3)*(self.H- params[2])**(5/3)*(Qt-(sqrt(self.S)*self.W* params[3]**(5/3)*(self.H- params[2])**(5/3))/( params[0]*( params[3]+1)**(5/3)*(log((params[1]- params[2])/(self.H-params[2]))+1))))/(params[0]*( params[3]+1)**(5/3)*(params[1]-params[2])*(log((params[1]- params[2])/(self.H- params[2]))+1)**2)
```

B: params [2]

```
2*(Qt-(sqrt(self.S)*self.W* params[3]**(5/3)*(self.H-params[2])**(5/3))/(params[0]*(params[3]+1)**(5/3)*(log((params[1]-params[2])/(self.H- params[2]))+1)))*((sqrt(self.S)*self.W*params[3]**(5/3)*(( params[1]-params[2])/(self.H- params[2])**2-1/(self.H-params[2]))*(self.H- params[2])**(8/3))/(params[0]*(params[3]+1)**(5/3)*(log((params[1]- params[2])/(self.H-params[2]))+1)**2*(params[1]- params[2]))+(5*sqrt(self.S)*self.W*params[3]**(5/3)*(self.H-params[2])**(2/3))/(3*params[0]*( params[3]+1)**(5/3)*(log((params[1]-params[2])/(self.H-params[2]))+1)))
```

Derivatives

r: params[3]

```
2*((5*sqrt(self.S)*self.W*(self.H-params[2])**(5/3)*
params[3]**(5/3))/(3*params[0]*(log((params[1]-params[2])/(self.H-
params[2]))+1)*(params[3]+1)**(8/3))-(5*sqrt(self.S)*self.W*(self.H-
params[2])**(5/3)*params[3]**(2/3))/(3*params[0]*(log((params[1]-
params[2])/(self.H-params[2]))+1)*(params[3]+1)**(5/3)))*(Qt-
(sqrt(self.S)*self.W*(self.H-params[2])**(5/3)*
params[3]**(5/3))/(params[0]*(log((params[1]- params[2])/(self.H-
params[2]))+1)*(params[3]+1)**(5/3)))
```