SNL-SWAN

Ari J. Posner, PhD

January, 2013

This module calculates the relative capture width (RCW) or ratio of incident wave power to captured wave power by a specific WEC device.

The RCW value is then returned to the main SWAN program (see changes below) where 1-RCW is used as the Power Transmission Coefficient (Ktp) for the obstacle.

The obstacle Transmission Coefficient (Kt) used by SWAN is squared in the spectral action balance equation, due to the fact that this coefficient represents the ratio of incident to lee-side wave heights at the obstacle. By contrast, Ktp represents the incident to lee-side power at the obstacle, and is therefore not squared in the spectral action balance equation.

SNL-SWAN has the options of using either the built in obstacle transmission coefficient options (constant Kt, Goda and Seelig, or d'Angremond and Van der Meer formulae (1996)), or there are two methods which can be used to represent WECs depending on available inputs. The WEC Module includes options for either the device Power Matrix or Relative Capture Width Curve to be used.

Additional files required for this simulation should be located in the same folder as the executable, along with the usual SWAN Input and bathymetry files.

WEC Module Options:

1. A Power Matrix for an individual device.
   1. The power matrix axes are Significant Wave Height (Hs)[m] and the Peak Period (Tp)[sec].
   2. Incident wave power is calcaluted using the general form :

Power = 0.42\*Hs\*Hs\*Tp

Which gives the incident wave power in kW/m.

* 1. Relative Capture Width (RCW) is given by:

RCW = Power Matrix Value/Incident Power/Device Width

* 1. Device power values are interpolated from the values in the power matrix, based on Hs and Tp values calculated in hydrodynamic simulation by SWAN. (See Numerical Recipes in Fortran Second Edition http://imf.ing.ucv.ve/\_1numerical\_recipe/Fortran77.html)
  2. Input files required for this method include:
     1. "Period.txt"
        1. This file contains a vertical array of values, the first being the number of values on the axis, followed by the axis values.

15

0.522232968

0.696310624

0.87038828

1.044465936

1.218543592

1.392621248

1.566698904

1.74077656

1.914854216

2.088931871

2.263009527

2.437087183

2.611164839

2.785242495

2.959320151

* + 1. "WaveHeight.txt"
       1. Same format as the “Period.txt” file with wave height axis values.

15

0.015151515

0.03030303

0.045454545

0.060606061

0.075757576

0.090909091

0.106060606

0.121212121

0.136363636

0.151515152

0.166666667

0.181818182

0.196969697

0.212121212

0.227272727

* + 1. "Power.txt"
       1. This is the power matrix file, produced by a developer of numerical simulation, given in kW. Matrix is made in MS Excel, then copy and pasted in to Notepad and saved as a .txt file.

2.83732E-05 3.24493E-05 5.09339E-05 7.76844E-05 0.000107231 0.000109568 7.63509E-05 5.85622E-05 4.20306E-05 2.8049E-05 2.55858E-05 1.91799E-05 1.82709E-05 1.24922E-05 1.09348E-05

0.000106481 0.000121511 0.000188493 0.000300165 0.00036195 0.000334902 0.00023747 0.000183693 0.000126868 0.000106289 8.27257E-05 5.96411E-05 4.66369E-05 4.72885E-05 2.87304E-05

0 0.000265635 0.000403722 0.000590594 0.000708065 0.000700113 0.00041536 0.000357481 0.000246137 0.000186015 0.000141083 0.000123147 8.14721E-05 7.16575E-05 7.35262E-05

0 0.000423889 0.000633225 0.000963377 0.001285004 0.001054441 0.00067307 0.000545382 0.000374907 0.000334498 0.000259333 0.000183918 0.00015488 0.000123439 0.000112327

0 0 0.001024499 0.001546218 0.001674163 0.001447323 0.001062693 0.000752242 0.0005313 0.000446748 0.000367487 0.000250919 0.000182267 0.000167498 0.000151712

0 0 0.001358882 0.002041379 0.002379185 0.002091922 0.001348915 0.000971752 0.000744481 0.000598844 0.000482213 0.000422594 0.00028649 0.000269115 0.00019715

0 0 0.001727349 0.002787942 0.003217128 0.002607183 0.001871579 0.001299376 0.000948431 0.00073843 0.00059228 0.000478351 0.000370643 0.000283037 0.000263205

0 0 0 0.003541131 0.003454458 0.003333397 0.002272846 0.001667092 0.00122546 0.000921966 0.000785083 0.000537329 0.000517952 0.000356775 0.000340433

0 0 0 0.0041271 0.004771376 0.003752225 0.002421569 0.001932138 0.001511662 0.00121255 0.000987311 0.00067703 0.000572796 0.000474809 0.000356663

0 0 0 0.005090611 0.005921715 0.004441716 0.003106903 0.00218089 0.001835534 0.001351786 0.001073121 0.000867781 0.000711084 0.000599831 0.000495721

0 0 0 0.006006452 0.00610458 0.005166089 0.003856342 0.00274697 0.002193336 0.00147823 0.001288333 0.000959987 0.000769162 0.000618626 0.000574836

0 0 0 0 0.00742613 0.006117013 0.00410519 0.003074302 0.002104196 0.001850868 0.001357189 0.00109833 0.000932372 0.000709041 0.000644862

0 0 0 0 0.009440585 0.006645156 0.004488882 0.003117891 0.002535858 0.001992097 0.001513204 0.001303598 0.000981042 0.000768972 0.00065379

0 0 0 0 0.01064565 0.007653976 0.00524805 0.003915699 0.002979503 0.002459063 0.001608882 0.001423929 0.001154415 0.000935317 0.000840432

0 0 0 0 0.010284463 0.009000298 0.005899326 0.004501255 0.003252303 0.002387992 0.002080942 0.001467368 0.001218243 0.000970465 0.000954391

* + 1. "Width.txt"
       1. This file has two values in it. First, and integer to identify which option will be used:
          1. 0=SWAN default option
          2. 1=Power Matrix option
          3. 2=Relative Capture Width option

Second, the device width value [m].

1

10

1. A Relative Capture Width curve for an individual device.
   1. This strategy inlvoves the interpolation of values from a given RCW curve, which is only period dependent.
   2. Input files for this strategy:
      1. Period="Period.txt"
         1. Vertical array, where first value is the number of values, followed by the actual axis values.

15

0.522232968

0.696310624

0.87038828

1.044465936

1.218543592

1.392621248

1.566698904

1.74077656

1.914854216

2.088931871

2.263009527

2.437087183

2.611164839

2.785242495

2.959320151

* + 1. RCW=“Relative Capture Width.txt”
       1. Same format as above.

15

0

0

0

0.427270503

0.423403813

0.291345301

0.167133482

0.120018246

0.085363306

0.06276644

0.04717585

0.030039248

0.023720157

0.018433494

0.013032215

* + 1. "Width.txt"
       1. This file has two values in it. First, and integer to identify which option will be used:
          1. 0=SWAN default option
          2. 1=Power Matrix option
          3. 2=Relative Capture Width option

Second, the device width value [m].

1

10

SNL-SWAN: Changes to SWAN4072ABCDE

The following files/subroutines/lines were changed or added:

swanmain.for

**SUBROUTINE** SWMAIN

275 **INTEGER** :: CASE\_N

276 **COMMON** CASE\_N

346! get the name of the file containing the data.

347 **open** (101, file="Width.txt", status='old')

348 **read** (101, \*) case\_n

349 **close** (101)

swanser.for

**SUBROUTINE** SWTRCF

2989 **USE** Interp\_v1

3204 **REAL** ::RCW,POWER

3205 **INTEGER** ::CASE\_N

3206 **COMMON** CASE\_N

NOTE: The algorithm to compute peak frequency below is copy and pasted from lines 3350-3364 from the original code, which have been commented out.

3254!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3255! compute peak frequency of incoming wave

3256 EMAX = 0.

3257 ISIGM = -1

3258 **DO** IS = 1, MSC

3259 ETD = 0.

3260 **DO** ID = 1, MDC

3261 ETD = ETD + SPCSIG(IS)\*AC2(ID,IS,KCGRD(ILINK+1))\*DDIR

3262 **END DO**

3263 **IF** (ETD.GT.EMAX) **THEN**

3264 EMAX = ETD

3265 ISIGM = IS

3266 **END IF**

3267 **END DO**

3268 **IF** (ISIGM.LE.0) ISIGM=MSC

3269 TP=2.\*PI/SPCSIG(ISIGM)

3270!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3282!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3283! Calling Power Matrix Subroutine

3284 **print**\*,case\_n

3285 **if** (case\_n.NE.0) **then**

3286 **call** P\_Matrix(TP,HSIN,RCW)

3287 **print**\*,RCW

3288! Compute incident wave power

3289 POWER=0.42\*HSIN\*HSIN\*TP

3290 **print**\*,TP

3291 **print**\*,HSIN

3292 **print**\*,POWER

3293! Use proportion of power taken off to estimate transmission coefficient

3294 **if** (RCW.EQ.0.0) **then**

3295 OBHKT = 1.0

3296 **else**

3297 OBHKT = 1-RCW

3298 **endif**

3299 **print**\*,OBHKT

3300 **WRITE** (PRINTF,\*) OBHKT

3301 **endif**

3302!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3303! constant transmission coefficient

3304!User defined transmission coefficient concerns ratio of40.09

3305! waveheights, so: 40.09

3306 **if** (case\_n.EQ.0) **then**

3307 OBHKT = COBST%TRCOEF(1) 40.31

3308 TRCF = OBHKT \* OBHKT 40.09

3309! If using power as ratio do not use square

3310 **else**

3311 TRCF = OBHKT

3312 **endif**

Added the file Bivariate\_1.F90

**MODULE** Interp\_v1

**contains**

**SUBROUTINE** P\_Matrix(TP,HSIN,RCW)

! Code input: Period and Wave Height from SWAN

! Code Body: Find power extracted by reading and interpolating

! power matrix

! Calculate RCW

!

! Record of Revisions

! Date Programmer Description of Change

! ----- ---------- ---------------------

! 12/21/12 Ari J Posner Original Code

!

**implicit none**

!Data Dictionary: Declare calling parameter types & definitions

**character** (len = 64 ) :: filein,fileout !input file

**character** (len = 64 ):: Period,Height,Power,Width\_val,RelCapWidth

**character** (len = 40 )::name,temp1(1000)

**real**, **allocatable**::x1a(:),x2a(:),ya(:,:)

**integer**::n\_Tp,n\_Hs,n\_RCW,ITRAS

**integer**::case\_n !case=1=power matrix; case=2=RCW

**COMMON** CASE\_N

**integer**::n,i,j,k,c

**REAL**:: dy,x1,x2,Width,p\_abs

! REAL, intent(in)::TP,HSIN

! REAL, intent(out)::y1

**REAL**::TP,HSIN,Incident,temp2,RCW

**REAL**::y1

! get the name of the file containing the data.

! write (\*, \*) 'Enter 1=Power Matrix Input, 2=RCW Input '

! read (\*, \*) case\_n

**select case** (case\_n)

**case**(1)

! write (\*, \*) 'Enter the period input filename: '

! read (101, \*) Period

! write (\*, \*) 'Enter the Wave Height Input filename: '

! read (102, \*) Height

! write (\*, \*) 'Enter the Power Matrix Input filename: '

! read (103, \*) Height

! write (\*, \*) 'Enter the Device Width Input filename: '

! read (104, \*) Width\_val

n\_Tp=0.0

n\_Hs=0.0

Width=0.0

n=0

i=0

j=0

k=0

c=0

Period="Period.txt"

Height="WaveHeight.txt"

Power="Power.txt"

Width\_val="Width.txt"

!open that file for reading.

**open** (101, file =Period, status='old')

**open** (102, file=Height, status='old')

**open** (103, file=Power, status='old')

**open** (104, file=Width\_val, status='old')

**read**(101,\*) n\_Tp

**read**(102,\*) n\_Hs

1001 **format** (1(/),F6.3)

**read**(104,1001) Width

**print**\*,Width

**print**\*,n\_Tp

**print**\*,n\_Hs

**allocate**(x1a(n\_Tp),x2a(n\_Hs),ya(n\_Hs,n\_Tp))

**read**(101,\*) (x1a(i),i=1,n\_Tp) !Period axis values [s]

**read**(102,\*) (x2a(i),i=1,n\_Hs) !Hs axis values [m]

**do** i=1,n\_Hs

**read**(103,\*) (ya(i,j),j=1,n\_Tp) !Power Matrix values [kW]

**enddo**

x1=TP !Input from SWAN model

x2=HSIN

!x1=9.4

!x2=4.3

!CALL polin2(x1a,x2a,ya,n\_Tp,n\_Hs,x1,x2,y1,dy)

**CALL** polin2(x1a,x2a,ya,n\_Tp,n\_Hs,x1,x2,y1,dy)

Incident=0.42\*HSIN\*HSIN\*TP !Estimate Incident wave power [kW/m]

RCW=y1/Incident/Width !Calculate RCW as Power (kW)/Device Width(m)/Incident(kW/m) []

**close** (101)

**close** (102)

**close** (103)

**close** (104)

!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**case**(2)

! write (\*, \*) 'Enter the period input filename: '

! read (101, \*) Period

! write (\*, \*) 'Enter the Relative Capture Width Input filename: '

! read (102, \*) RelCapWidth

n\_Tp=0.0

n\_Hs=0.0

RCW=0.0

n=0

i=0

j=0

k=0

c=0

Period="Period.txt"

RelCapWidth="Relative Capture Width.txt"

!open that file for reading.

**open** (101, file =Period, status='old')

**open** (102, file=RelCapWidth, status='old')

**read**(101,\*) n\_Tp

**read**(102,\*) n\_RCW

**allocate**(x1a(n\_Tp),x2a(n\_RCW))

**read**(101,\*) (x1a(i),i=1,n\_Tp)

**read**(102,\*) (x2a(i),i=1,n\_RCW)

**call** polint(x1a,x2a,n\_Tp,TP,RCW,dy)

**close** (101)

**close** (102)

**end select**

**endsubroutine**

**ENDMODULE**

!ENDPROGRAM

**SUBROUTINE** polin2(x1a,x2a,ya,m,n,x1,x2,y,dy)

**IMPLICIT NONE**

**INTEGER** m,n,NMAX,MMAX

**REAL** dy,x1,x2,y,x1a(m),x2a(n),ya(n,m)

**PARAMETER** (NMAX=100,MMAX=100)

! C USES polint

! Given arrays x1a(1:m) and x2a(1:n) of independent variables, and an m by n array of

! function values ya(1:m,1:n), tabulated at the grid points defined by x1a and x2a; and

! given values x1 and x2 of the independent variables; this routine returns an interpolated

! function value y, and an accuracy indication dy (based only on the interpolation in the x1

! direction, however).

**INTEGER** j,k

**REAL** ymtmp(MMAX),yntmp(NMAX)

!12 do j=1,m

!11 do k=1,n

!yntmp(k)=ya(j,k)

! enddo

!call polint(x2a,yntmp,n,x2,ymtmp(j),dy)

!! Interpolate answer into temporary storenddo 12 age.

! enddo

!call polint(x1a,ymtmp,m,x1,y,dy)

!return

**do** k=1,m

yntmp=0.0

**do** j=1,n

yntmp(j)=ya(j,k)

**enddo**

**do** j=1,m-1

**if**(x2>=x2a(j).AND.x2<(x2a(j+1))) **then**

**call** polint(x2a(j:j+1),yntmp(j:j+1),2,x2,ymtmp(k),dy)

**exit**

**endif**

**enddo**

**enddo**

! Interpolate answer into temporary storenddo 12 age.

**do** k=1,m-1

**if**(x1>=x1a(k).AND.x1<(x1a(k+1))) **then**

**call** polint(x1a(k:k+1),ymtmp(k:k+1),2,x1,y,dy)

**exit**

**endif**

**enddo**

**END**

**SUBROUTINE** polint(xa,ya,n,x,y,dy)

**IMPLICIT NONE**

**INTEGER** n, NMAX

**REAL** dy,x,y,xa(n),ya(n)

**PARAMETER** (NMAX=20)

! Given arrays xa and ya, each of lenth n, and given a value x, this routine returns a

! value y, and an error estimate dy. If P(x) is the polynomial of degree N-1 such that

! P(xa\_i)=ya\_i, i=1...,n, then returnd value y=P9x).

**INTEGER** i,m,ns

**REAL** den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)

ns=1

dif=abs(x-xa(1))

11 **do** i=1,n

dift=abs(x-xa(i))

**if** (dift.lt.dif) **then**

ns=i

dif=dift

**endif**

c(i)=ya(i)

d(i)=ya(i)

**enddo**

y=ya(ns)

ns=ns-1

13 **do** m=1,n-1

12 **do** i=1,n-m

ho=xa(i)-x

hp=xa(i+m)-x

w=c(i+1)-d(i)

den=ho-hp

**if** (den.eq.0) **pause** 'failure in polint'

! this error can occur only if 2 input xa's are (to within roundoff) identical.

den=w/den

d(i)=hp\*den

c(i)=ho\*den

**enddo**

**if** (2\*ns.lt.n-m) **then**

dy=c(ns+1)

**else**

dy=d(ns)

ns=ns-1

**endif**

y=y+dy

**enddo**

**return**

**END**