EXTRACT DUNE PKS

361

----------------------------------------------------------------

[pks,locs] = findpeaks(smoothed profile, minimum peak prominence is P)

[...] = findpeaks(...,'MinPeakProminence',MPP) finds peaks guaranteed to have a vertical drop of more than MPP from the peak on both sides without encountering either the end of the signal or a larger intervening peak. The default value of MPP is zero.

----------------------------------------------------------------

362

~~Pks > 1.2 m elevation found (Minimum dune ht)~~

Pks > minelev, defined on line 13 (minimum dune ht)

369

Set to nan if empty or if no elevations >1.2 m or minimum dune ht.

377

Dune crest is defined as the first peak fitting the criteria defined above, moving from the shoreline landward.

----------------------------------------------------------------

dune\_c = pks(dune\_loc\_i(1));

dune\_x = X(dune\_loc(1));

dune\_y = Y(dune\_loc(1))

----------------------------------------------------------------

The c, x, y are added to the array

----------------------------------------------------------------

dunec = horzcat(dunec,dune\_c);

dunex = horzcat(dunex,dune\_x);

duney = horzcat(duney, dune\_y);

----------------------------------------------------------------

388

The highest crest/ridge is located as the maximum of pks

duneHigh\_c = max(pks);

maxloc = find(pks == max(pks));

duneHigh\_x = X(locs(maxloc));

duneHigh\_y = Y(locs(maxloc));

----------------------------------------------------------------

Added to the array in the same way.

397

Dune toe.

The cross-shore-meters CSM vector is used, created ln. 353.

Converted from the lat/lon values associated with positive only elevations (above 0 navd88).

Add MHW elevation input value.

Run in UTM or other projected cords so no conversion needed.

Distance from shoreline to the foredune crest is determined:

----------------------------------------------------------------

crestDist = CSM(dune\_loc(1));

----------------------------------------------------------------

A search window is established (50 m in this case) from crest to look for toe in the direction of the shoreline.

405

New vectors indexing x, y, and z between the foredune crest and the end of the window is created.

----------------------------------------------------------------

toeSearchInd = find(abs(CSM - (crestDist - ToeWindow))<10, 1, 'first'); % --- find distance from shore to stop looking for toe

shore2crestX = X(toeSearchInd:dune\_loc(1)); % --- location of dune to end

shore2crestY = Y(toeSearchInd:dune\_loc(1)); % --- location of dune to end shore2crestZ = z2(toeSearchInd:dune\_loc(1));

----------------------------------------------------------------

416

The second derivative of z is calculated along this vector.

Peaks and troughs in d2 are found. D2 crosses 0 where the inflection point on the face of the dune exists (slope moves from decreasing to increasing, still negative),

421

The last peak (from shoreline towards dune) in D2 is located that is larger than 0.9.

Check this d2 peak parameter (ii, line 421)

Check test expression, make sure correct value set (ln. 434), run

If no peak is found above or at this value, peaks are examined between 0.9 and 0.001 with a step of -0.0001.

If still no peaks found, the most seaward trough in D2 is located.

441

If toe\_loc\_i is greater than one (not the most seaward) and the value of D2 at this trough location is more than 0.5\* the value of the peak chosen at the toe loc, the next seaward toe loc is chosen.

Check on this logic ^^ I guess the purpose is to make sure there are no more big slope changes seaward of where the toe is chosen?

458

Toe x y and z are set and concatenated.

516

Crest, toe, and high xyz are written to shapefile.