**Carbon (dry matter) partitioning in the root model.**

These variables are used:

*pcrl* contains the minimum carbohydrate

*pcrq* the maximum carbohydrate

*pcrts* is a counter for the total carbon actually used when distributing carbon among the elements. It is compared to pcrs to make sure the total allocated carbon is not exceeded.

*pcrs* a holding variable for the available carbon. It is calculated in the partitioning component of 2dsoil. It is assigned either pcrl (when leaf turgor pressure larger than 2 bars) or pcrq (when turgor pressure less than 2 bars)

Note that a fraction of pcrq can be added to pcrs (after assigning pcrl) if turgor pressure is less than 5 bars and decreasing.

 Therefore at the end of carbon partitioning, the amount carbohydrate to partition to the roots is in pcrs (at this point we can forget about pcrq and pcrl).

Remember, these are all RATES, (g dry matter per day)

PDWR(element) is the potential carbohydrate accretion rate needed in the element to grow the roots to the necessary weight (defined by the potential rate of increase x current weight x root growth correction factors. This array is calculated in the carbon partitioning routine.

PDWR(e)=RTWT(e)\*awr

where *awr* is the potential percentage increase in root weight in a cell as a function of current weight. This value is input in the root file.

**Dimensional analysis** - *RTWT* is gr root, pcrl is grams root per day. *PDWR* must have similar units if *pcrl* is additive to it. Thus the units of awr are day-1.

Not all roots with a value of pdwr will grow, depending on the amount of carbohydrate available.

Program Logic:

The variables pcrl and pcrq are assigned in the plant model (Crop object) via the SHOOTR structure.  The subroutine Carbon\_Partitioning calculates the root growth correction factors and the potential root growth rate (PDWR) evaluates the water status of the plant to determine how much carbon to allocate. The needed carbohydrate is assigned to the variable pcrs as described above.

The root growth subroutine (rootgrow) allocates carbon among the elements based on available carbohydrate. It uses PDWR, and pcrs and input. The array ADWR holds the actual carbohydrate assigned to the roots. The units are the same as PDWR. While all elements of PDWR that have roots will have a value, not all elements of ADWR will have values. For example if RTWT(25)=0.1 then PDWR(25) will be 0.1\*awr. ADWR(25) could be anywhere from 0 to 0.1\*awr depending on how much carbohydrate was available. The variable pcrts holds the cumulative carbo assigned to the roots and is used to compare to pcrs to decide when the carbon was used up. If the roots don't use all the carbon allocated then pcrts will be less than pcrs. If the roots use all the carbon then the code will overshoot pcrs slightly. That is, the carbohydrate in the last element will be too high since the code adds increments determined by PDWR. The code goes back and removes the fraction of PDWR that is too high from ADWR but does not adjust the summation counter pcrts. I added the following line of code to account for this (about line 94 in root\_growC):

PCRS=AMIN1(PCRTS,PCRS)

Thus PCRS will hold the actual carbohydrate used in that TIME STEP.

The actual mass of root in the soil element is calculated as AWR=ADWR(e)\*step. The variable step is time and gives an actual mass of root in that time step.

Here is where the problem is. Note that the root grow and carbon partitioning routines will operate for several loops before the crop model executes. thus when we return pcrs to the crop model it will be the value of pcrs determined at the **last** execution of the root subroutines before the crop model is called. Thus we never get an accurate accounting of carbon used. I also found that carbohydrate is not being completely used by the plant when it is young. As the correct value of total carbohydrate is never returned to the plant model the actual carbon use by the roots cannot be determined accurately. When actual root growth is compared to carbohydrate sent actual root growth is always lower (since all the carbon is not being used).

Several changes need to be done to fix this (Note – Sep 20, 2014 – DT I believe these have all been addressed).

1. need to integrate the rates for carbohydrate sent to the soil model over the 1 hour time and send the integrated carbohydrate used back to the plant model.
2. need to increase carbohydrate use by the root models when the roots are small and cannot use all the carbohydrate sent by the plant. There are several considerations/approaches to use here.
   1. should we change the time loop for the root growth and partitioning to the same scale as the plant (one hour)
   2. loop over the root growth code several times until the carbohydrate is used up
   3. increase the root growth rate (awr) when the plant is young to increase the carbohydrate sink or make awr a dynamic variable that responds to carbon availability. We could also make awr an array and assign different values to young and old roots.
   4. Roots should grow at night but don't (because there is no assimilation of co2 at night. If we distributed carbohydrate over 24 hours we would be more likely to use all the carbo partitioned/.

#1 can be addressed easily and will give a better accounting but no change in root growth.

#2 can be addressed with different levels of programming complexity and there are long and short term solutions. Any solution will increase root growth and help the roots grow deeper.

The short term solutions include

1. increasing root growth rate when the plant is young and decrease it as the plant ages using a linear function (this doesn't guarantee that all the carbo will be used but it will increase root growth),
2. loop the root growth routine (this will give precedence to the first elements in the root array) until carbon is used up,
3. use a PID to adjust the root growth rate on the cumulative and instantaneous carbo used as is done for nitrogen,
4. adjust awr dynamically to use all the carbo or
5. pool the leftover carbo not partitioned to roots so it can be distributed over the night (I wound up doing this approach – required minimal changes DT)

I like the last two best since they are more likely to distribute carbo uniformly among the elements with growing roots and don't require too many changes in the code. If you make awr too big, you can distort root growth so only the elements toward the top of the sorted array grow.

At some point it will be useful to have a variable to hold a root age category for each element. Of course elements can have a range of ages based on size so this will only be an average.