In[1]:=
```
Remove["Global`*"]
Format[Continuation[_], FortranForm] := Format["      ", OutputForm]
Begin["Global`"]
```

Out[3]=
```
Global`
```

Conversions of climate data

Function to calculate downwards longwave radiation (after Roderick's notes):

Following the notes by Mike Roderick in response to the first draft of the second paper, downwards longwave radiation can be written as:

In[4]:=
```
rldown = kboltz ta^4 (1 - edash rdash);
edash = 0.34 - 0.14 √(vp / 1000 );
rdash = 1.35 sg / ((0.75 + 2 * 10^-5 z) so);
```

Where kboltz is the Boltzman constant (5.67032*10^-8 W m^2 K^{-4}, ta is air temperature at 2 m above surface in K, vp is vapour pressure in Pa, sg is global solar irradiance in W m^-2, z is height above mean sea level in m and so is the extra-terrestrial solar irradiance in W m^-2.

In[7]:=
```
FortranForm[FullSimplify[rldown]]
```

Out[7]//FortranForm=
```
        kboltz*ta**4*(1 + (sg*(-0.4590000000000001 + 0.005976704777718238*Sqrt(vp)))/
         (so*(0.75 + 0.00002*z)))
```

In[8]:=
```
FortranForm[SetAccuracy[FullSimplify[rldown], 7]]
Clear[rldown, edash, rdash]
```

Out[8]//FortranForm=
```
kboltz*ta**4*(1. + (sg*(-0.459 + 0.005977*Sqrt(vp)))/(so*(0.75 + 0.00002*z)))
```

■ **Alternative:**

In[10]:=
```
rldown = kboltz ta^4 (1 - edash rdash);
edash = 0.34 - 0.14 √(vp / 1000 );
rdash = 1.35 sgbyso / (0.75 + 2 * 10^-5 z);
```

Where kboltz is the Boltzman constant (1.381*10^-23 JK^-1, ta is air temperature at 2 m above surface in K, vp is vapour pressure in Pa, sg is global solar irradiance in W m^-2, z is height above mean sea level in m and so is the extra-terrestrial solar irradiance in W m^-2.

In[13]:=
```
FortranForm[FullSimplify[rldown]]
```

Out[13]//FortranForm=
```
        (kboltz*ta**4*(0.75 + sgbyso*
           (-0.4590000000000001 + 0.005976704777718238*Sqrt(vp)) + 0.00002*z))/
      (0.75 + 0.00002*z)
```

In[14]:=
```
FortranForm[SetAccuracy[FullSimplify[rldown], 7]]
Clear[rldown, edash, rdash]
```

Out[14]//FortranForm=
```
(kboltz*ta**4*(0.75 + sgbyso*(-0.459 + 0.005977*Sqrt(vp)) + 0.00002*z))/(0.75 + 0.00002*z)
```

Function to calculate irr from daily solar radiation measurements (after Ting 87):

In[16]:=
```
par = 2.0804 * srad;
```

In[17]:=
```
FortranForm[2.0804 * srad]
```
Out[17]//FortranForm=
```
2.0804*srad
```

where par in mol/m2, srad in MJ/m2

Function to calculate irr from hourly solar radiation measurements (after Ting 87):

In[18]:=
```
par = 2.0699 * srad;
```

In[19]:=
```
FortranForm[2.0699 * srad]
```
Out[19]//FortranForm=
```
2.0699*srad
```

where par in mol/m2, srad in MJ/m2

Function to calculate instantaneous irradiance from daily (from sunlight2a.nb)

In[20]:=
```
N[Degree]
```
Out[20]=
```
0.0174533
```

A simplified calculation of daylength after Spitter, where td was the day of the year and was changed to dayyear:

In[21]:=
```
daylength ==
 Simplify[12 + 24 / 180 ArcSin[Tan[lat Degree] * tandelta] * 180 / Pi /.
   tandelta → -0.3979486313076104`
     Cos[2/365 π (10 + dayyear)] / √(0.9208184434201998` - 0.07918155657980021` Cos[4/365 π (10 + td)])]
```
Out[21]=
$$\text{daylength} = 12 - \frac{24\,\text{ArcSin}\left[\frac{0.397949\,\text{Cos}\left[\frac{2}{365}(10+\text{dayyear})\,\pi\right]\,\text{Tan}[^\circ\,\text{lat}]}{\sqrt{0.920818-0.0791816\,\text{Cos}\left[\frac{4}{365}\,\pi\,(10+\text{td})\right]}}\right]}{\pi}$$

In[22]:=
```
FortranForm[SetAccuracy[%, 7]]
```
Out[22]//FortranForm=
```
      daylength.eq.
     12. - 7.639437*ArcSin((0.397949*Cos(0.017214*(10. + dayyear))*Tan(0.017453*lat))/
        Sqrt(0.920818 - 0.079182*Cos(0.034428*(10. + td))))
```

Spitters gives the following function for global radiation per second as a function of daily global radiation (td changed to dayyear):

In[23]:=
```
sindelta = -Sin[23.45 Degree] Cos[360 (td + 10) / 365 Degree];
cosdelta = √(1 - sindelta^2) ;
sinbeta = Sin[lat Degree] sindelta + Cos[lat Degree] cosdelta Cos[15 (th - 12) Degree];
so = ssc * sinbeta;
intsinbeta =
  3600
    (daylength Sin[lat Degree] sindelta + (24 / Pi) Cos[lat Degree] cosdelta
       (1 - Tan[lat Degree]^2 tandelta^2)^0.5) /. tandelta → sindelta / cosdelta;
sg = sinbeta * sgdaily / intsinbeta;
FullSimplify[sg];
FortranForm[SetAccuracy[%, 7]]
```

Out[30]//FortranForm=
```
      (-0.000873*sgdaily*Cos(0.017453*lat)*
       Sqrt(0.920818 - 0.079182*Cos(0.034428*(10. + td)))*Cos(0.2618*th) -
      0.000347*sgdaily*Cos(0.017214*(10. + td))*Sin(0.017453*lat))/
     (-1.250192*daylength*Cos(0.017214*(10. + td))*Sin(0.017453*lat) +
      24.*Cos(0.017453*lat)*Sqrt(0.920818 - 0.079182*Cos(0.034428*(10. + td)))*
       (1. - (0.158363*Cos(0.017214*(10. + td))**2*Tan(0.017453*lat)**2)/
           (0.920818 - 0.079182*Cos(0.034428*(10. + td))))**0.5)
```

In[31]:=
```
FullSimplify[so];
FortranForm[SetAccuracy[%, 7]]
```

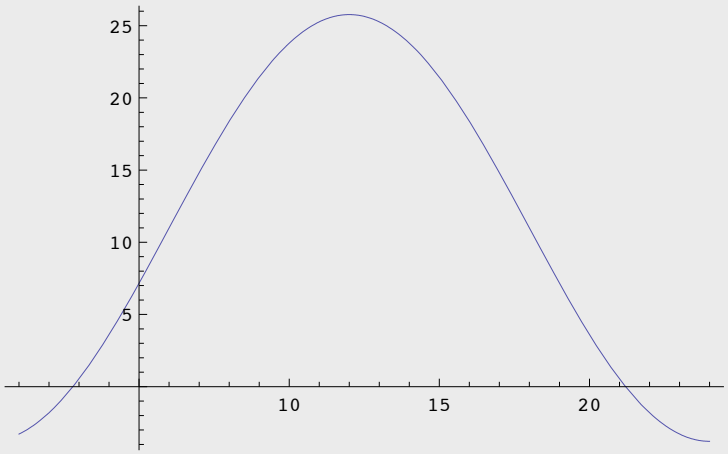
Out[32]//FortranForm=
```
      -1.*ssc*Cos(0.017453*lat)*Sqrt(0.920818 - 0.079182*Cos(0.034428*(10. + td)))*
     Cos(0.2618*th) - 0.397949*ssc*Cos(0.017214*(10. + td))*Sin(0.017453*lat)
```

In[33]:=
```
FullSimplify[sgdaily / intsinbeta / ssc];
sgbyso = FortranForm[SetAccuracy[%, 7]]
```

Out[34]=
```
sgbyso = (0.000873 * sgdaily) / (-1.250192 * daylength * ssc * Cos (0.017214 * (10. + td)) * Sin (0.017453 * lat) +
    24. * ssc * Cos (0.017453 * lat) * Sqrt (0.920818 - 0.079182 * Cos (0.034428 * (10. + td))) *
     (1. - (0.158363 * Cos (0.017214 * (10. + td)) ** 2 * Tan (0.017453 * lat) ** 2) /
         (0.920818 - 0.079182 * Cos (0.034428 * (10. + td)))) ** 0.5)
```

In[35]:=
```
td = 180; lat = 60; sgdaily = 10^6;
daylength = 12 + 24 ArcSin[Tan[lat °] tandelta] 180 / (180 π) /.
  tandelta → -0.3979486313076104` Cos[2/365 π (10 + td)] / √(0.9208184434201998` - 0.07918155657980021` Cos[4/365 π (10 + td)]);
Plot[sg, {th, 1, 24}]
Clear[td, lat, sgdaily, daylength]
```

Out[36]=



Function to calculate diurnal temperature variation from Tmax and Tmin (after VP2.nb)

In[38]:=
```
FortranForm[
 tahour ==
  tamean +
   dtr (0.0138` Cos[3.513` - 1/3 (-1 + hour) π] + 0.0168` Cos[0.822` - 1/4 (-1 + hour) π] +
      0.0984` Cos[0.36` - 1/6 (-1 + hour) π] + 0.4632` Cos[3.805` - 1/12 (-1 + hour) π])]
```

Out[38]//FortranForm=
```
      tahour.eq.tamean + dtr*(0.0138*Cos(3.513 - ((-1 + hour)*Pi)/3.) +
        0.0168*Cos(0.822 - ((-1 + hour)*Pi)/4.) +
        0.0984*Cos(0.36 - ((-1 + hour)*Pi)/6.) + 0.4632*Cos(3.805 - ((-1 + hour)*Pi)/12.))
```

where

dtr = diurnal temperature range

tamean = daily mean temperature in oC

hour = hour of day

Function to calculate VD from VP and RH or temperature (after VP2.nb)

Equations taken from http://www.fao.org/docrep/X0490E/x0490e07.htm#calculation%20procedures (Allen et al. (1998)).

In[39]:=
```
vps == 0.6108 * Exp[17.27 * temp / (temp + 237.3)]
```

Out[39]=
$$\text{vps} == 0.6108\, e^{\frac{17.27\,\text{temp}}{237.3+\text{temp}}}$$

where vps is saturated vapour pressure in kPa, temp in ^{o}C

Due to the non-linearity of the above equation, the mean saturation vapour pressure for a day, week, decade or month should be computed as the mean between the saturation vapour pressure at the mean daily maximum and minimum air temperatures for that period:

In[40]:=
```
vps == vpstmax + vpstmin / 2
```

Out[40]=
$$\text{vps} == \text{vpstmax} + \frac{\text{vpstmin}}{2}$$

Using mean air temperature instead of daily minimum and maximum temperatures results in lower estimates for the mean saturation vapour pressure. The corresponding vapour pressure deficit (a parameter expressing the evaporating power of the atmosphere) will also be smaller and the result will be some underestimation of the reference crop evapotranspiration. Therefore, the mean saturation vapour pressure should be calculated as the mean between the saturation vapour pressure at both the daily maximum and minimum air temperature.

■ Actual vapour pressure (vp) derived from relative humidity data

In[41]:=
```
vp == (vpstmin * rhmax / 100 + vpstmax * rhmin / 100) / 2
```

Out[41]=
$$\text{vp} == \frac{1}{2}\left(\frac{\text{rhmin}\,\text{vpstmax}}{100} + \frac{\text{rhmax}\,\text{vpstmin}}{100}\right)$$

The above is equivalent to Equation 17 from above resource.

The molar vapour deficit is

In[42]:=
```
vd == (vps - vp) / airpressure
```

Out[42]=
$$\text{vd} == \frac{-\text{vp} + \text{vps}}{\text{airpressure}}$$

where d is expressed in mol/mol.

With the above equations we get:

In[43]:=
```
vd == (vpstmax + vpstmin / 2 - (vpstmin * rhmax / 100 + vpstmax * rhmin / 100) / 2) / 101.325
```

Out[43]=
$$\text{vd} == 0.00986923 \left(\text{vpstmax} + \frac{\text{vpstmin}}{2} + \frac{1}{2}\left(-\frac{\text{rhmin}\ \text{vpstmax}}{100} - \frac{\text{rhmax}\ \text{vpstmin}}{100}\right)\right)$$

Now vps=0.6108*Exp[17.27*temp/(temp+237.3)], therefore

In[44]:=
```
vpstmin = 0.6108 * Exp[17.27 * tmin / (tmin + 237.3)]
```

Out[44]=
$$0.6108\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmin}}}$$

In[45]:=
```
vpstmax = 0.6108 * Exp[17.27 * tmin / (tmax + 237.3)]
```

Out[45]=
$$0.6108\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmax}}}$$

In[46]:=
```
vd == (vpstmax + vpstmin / 2 - (vpstmin * rhmax / 100 + vpstmax * rhmin / 100) / 2) / 101.325
```

Out[46]=
$$\text{vd} == 0.00986923 \left(0.6108\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmax}}} + 0.3054\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmin}}} + \frac{1}{2}\left(-0.006108\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmin}}}\ \text{rhmax} - 0.006108\ e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmax}}}\ \text{rhmin}\right)\right)$$

In[47]:=
```
FullSimplify[
 0.009869232667160128`
  (0.6108` e^(17.27` tmax / (237.3` + tmax)) + 0.3054` e^(17.27` tmin / (237.3` + tmin)) +
    1/2 (-0.006108000000000001` e^(17.27` tmin / (237.3` + tmin)) rhmax - 0.006108000000000001` e^(17.27` tmax / (237.3` + tmax)) rhmin))]
```

Out[47]=
$$e^{\frac{17.27\ \text{tmin}}{237.3+\text{tmin}}}\ (0.00301406 - 0.0000301406\ \text{rhmax}) + e^{\frac{17.27\ \text{tmax}}{237.3+\text{tmax}}}\ (0.00602813 - 0.0000301406\ \text{rhmin})$$

In[48]:=
```
Clear[vpstmax, vpstmin]
```

Originally I calculated d using the given vp and only rhmaxt:

In[49]:=
```
d == ((vp * 100) / (rhmaxt / 100) - vp * 100) / 101 325
```

Out[49]=
$$\text{d} == \frac{-100\ \text{vp} + \frac{10\,000\ \text{vp}}{\text{rhmaxt}}}{101\,325}$$

where d: wi-wa (water vapour conc saturated - atmoshperic water vapour conc) in mol/mol, vp=vapour pressure in hPa, rhmaxt: relative humidity at maximum temperature. This is less accurate and should not be used.

■ Derivation of VD from 9am VP and diurnal temperature

Knowing the saturated vapour pressure in kPa for any temperature from above, and assuming that the absolute vapour pressure as measured at 9am is constant, we can calculate the diurnal variation of VPD following VP2.nb as:

In[50]:=
```
vps = 0.6108 * Exp[17.27 * tair / (tair + 237.3)];
vd == FullSimplify[(vps * 1000 - vp) / 101 325]
FortranForm[SetPrecision[%, 7]]
Clear[vps]
```

Out[51]=
$$\text{vd} == 0.00602813\ e^{\frac{17.27\ \text{tair}}{237.3+\text{tair}}} - 9.86923 \times 10^{-6}\ \text{vp}$$

Out[52]//FortranForm=
```
vd.eq.0.006028127*2.718282**((17.27*tair)/(237.3 + tair)) - 9.869233e-6*vp
```

Multi-layer Water balance (waterbalancemultiveg7.nb)

Note that upward is positive and downward negative. Pcap is always positive.

Calculation of pcap and hydraulic conductivity

For capillary pressure and unsaturated hydraulic conductivity, we will use the van Genuchten model:

In[54]:=
```
pcap == FortranForm[ (-1 + su^-1/m)^1/n / alpha /. {m → mvg, n → nvg, alpha → alphavg}]
```

Out[54]=
```
pcap == (-1 + su ** (-1 / mvg)) ** (1 / nvg) / alphavg
```

In[55]:=
```
kunsat == FortranForm[ksat su^1/2 (-1 + (1 - su^1/m)^m)^2 /. m → mvg]
```

Out[55]=
```
kunsat == ksat * Sqrt (su) * (-1 + (1 - su ** (1 / mvg)) ** mvg) ** 2
```

Calculation of initial values for ys and pcapvec:

In[56]:=
```
Solve[pcapinit == (-1 + su^-1/m)^1/n / alpha, su]
```

Out[56]=
```
{{su → (1 + (alpha pcapinit)^n)^-m}}
```

We will now set the initial conditions for the soil moisture such that no fluxes would occur (ys=zr, and dpcap/dz=-1):

In[57]:=
```
FortranForm[ys == zr]
FortranForm[su == (1 + (alphavg pcapinit)^nvg)^-mvg /. pcapinit → depth - ys]
```

Out[58]//FortranForm=
```
su.eq.(1 + (alphavg*(depth - ys))**nvg)**(-mvg)
```

where depth-ys is the distance to the water table

In[59]:=
```
FortranForm[If[ys > zr, omgu == (cz - ys) / Sqrt[(cz - ys) (cz - zr)], 1]]
```

Out[59]//FortranForm=
```
If(ys.gt.zr,omgu.eq.(cz - ys)/Sqrt((cz - ys)*(cz - zr)),1)
```

Calculation of state variables and fluxes

■ Infiltration, seepage face flow and fluxes between layers

In[60]:=
```
inf == FortranForm[Min[rain * omgu, ksat * omgu (1 + pcapvec[[1]] / (0.5 delyuvec[[1]]))]]
```

Out[60]=
```
inf == Min (omgu * rain, ksat * omgu * (1 + (2. * Part (pcapvec, 1)) / Part (delyuvec, 1)))
```

In[61]:=
```
spgfcf = FortranForm[ksat * omgo / (Cosgo cgs) * 1 / 2.0 * (ys - zr)]
```

Out[61]=
```
spgfcf = (0.5 * ksat * omgo * (ys - zr)) / (cgs * Cosgo)
```

In[62]:=
```
qbli = FortranForm[-Max[kunsatveci, kunsatveciplus1] omgu (1 + (-pcapveci + pcapveciplus1) / (0.5 delyuveci + 0.5 delyuveciplusone))]
```

Out[62]=
```
qbli = -(omgu * (1 + (-pcapveci + pcapveciplus1) / (0.5 * delyuveci + 0.5 * delyuveciplusone)) *
    Max (kunsatveci, kunsatveciplus1))
```

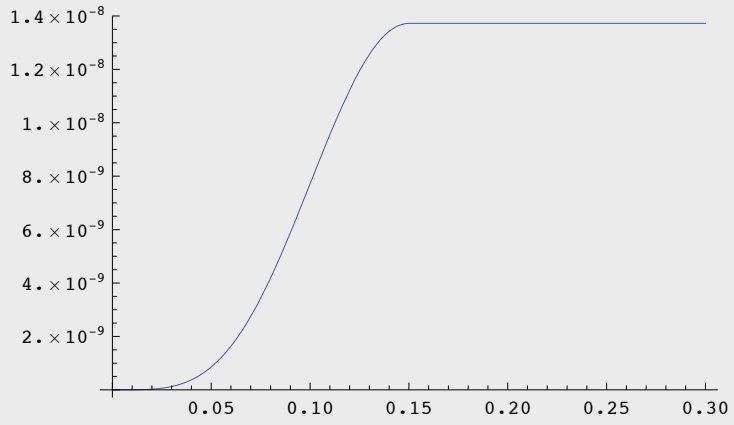
■ Soil evaporation

If soil temperature is known, we can calculate soil evaporation as derived in dataprep8a.nb::

In[63]:=
```
vpsoil = { beta vpssoil + (1 - beta) vp /. beta → 1/4 (1 - Cos[thetav π / thetafc])^2   thetav < thetafc
         { vpssoil                                                                      thetav ≥ thetafc ;
vpssoil = 610.8` ℯ^(17.27` tsoil / (237.3` + tsoil));
vdsoil = (vpsoil - vp) / pressure;
esoil = Simplify[0.03` × 18 vdsoil / 10^6]
Plot[esoil /. {pressure → 100 000, vp → 1702, thetafc → 0.15`, tsoil → 30}, {thetav, 0, 0.3`}]
Clear[vpsoil, vpssoil, vdsoil, esoil]
```

Out[66]=
$$\frac{1}{\text{pressure}}\left(-5.4\times10^{-7}\,\text{vp}+5.4\times10^{-7}\left(\begin{cases}\text{vp}+152.7\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}}\left(-1.+\cos\left[\frac{\pi\,\text{thetav}}{\text{thetafc}}\right]\right)^2-\text{vp}\,\sin\left[\frac{\pi\,\text{thetav}}{2\,\text{thetafc}}\right]^4 & \text{thetav}<\text{thetafc}\\ 610.8\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}} & \text{thetav}\geq\text{thetafc}\end{cases}\right)\right)$$

Out[67]=



In[69]:=
```
vpssoil == FortranForm[610.8` E^((17.27` tsoil)/(237.3` + tsoil))]
vpsoil == FortranForm[beta vpssoil + (1 - beta) vp /. beta -> 1 / 4 (1 - Cos[thetav / thetafc Pi]) ^ 2]
vpsoil == vpssoil
vdsoil == FortranForm[(vpsoil - vp) / pair]
```

Out[69]=
```
vpssoil = 610.8 * E ** ((17.27 * tsoil) / (237.3 + tsoil))
```

Out[70]=
```
vpsoil =
 vp * (1 - (1 - Cos ((Pi * thetav) / thetafc)) ** 2 / 4.) + (vpssoil * (1 - Cos ((Pi * thetav) / thetafc)) ** 2) / 4.
```

Out[71]=
```
vpsoil = vpssoil
```

Out[72]=
```
vdsoil = (-vp + vpsoil) / pair
```

The above could also be simplified to:

In[73]:=
```
vdsoil == FullSimplify[beta esat + (1 - beta) ea - ea]
```

Out[73]=
```
vdsoil = beta (-ea + esat)
```

We could equally espress vpsoil as a function of su, if we convert thetav to su

In[74]:=
```
Solve[su == (thetav - thetar) / (thetas - thetar), thetav]
sufc == (thetafc - thetar) / (thetas - thetar)
```

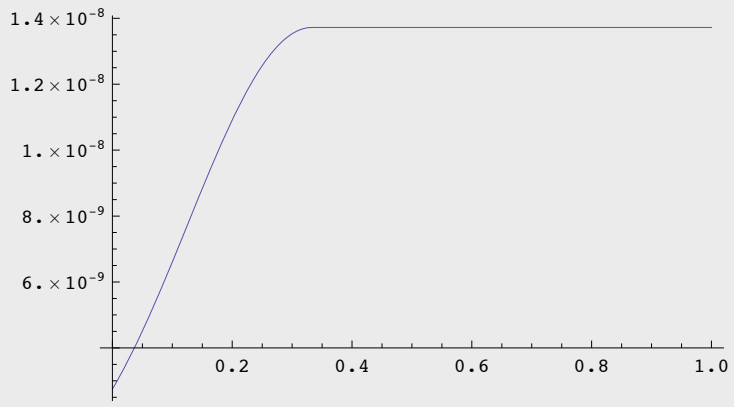
Out[74]=
```
{{thetav → thetar - su thetar + su thetas}}
```

Out[75]=
$$\text{sufc} = \frac{\text{thetafc} - \text{thetar}}{-\text{thetar} + \text{thetas}}$$

In[76]:=
```
vpsoil =
  {beta vpssoil + (1 - beta) vp /. beta → 1/4 (1 - Cos[thetav π / thetafc])^2 /.
     thetav → thetar - su thetar + su thetas                    su < (thetafc-the / -thetar+the
   vpssoil                                                      su ≥ (thetafc-the / -thetar+the
  ;
vpssoil = 610.8` ℯ^(17.27` tsoil / (237.3` + tsoil));
vdsoil = (vpsoil - vp) / pressure;
esoil = Simplify[0.03` × 18 vdsoil / 10^6]
Plot[esoil /. {pressure → 100 000, vp → 1702, thetafc → 0.15`, tsoil → 30, thetas → 0.31`, thetar → 0.07`},
 {su, 0, 1}, PlotRange → All]
Clear[vpsoil, vpssoil, vdsoil, esoil]
```

Out[79]=
$$\frac{1}{\text{pressure}}\left(-5.4\times10^{-7}\,\text{vp}+5.4\times10^{-7}\left(\begin{cases}\text{vp}+152.7\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}}\left(-1.+\cos\left[\frac{\pi\,(\text{thetar}-\text{su thetar}+\text{su thetas})}{\text{thetafc}}\right]\right)^2-\text{vp}\sin\left[\frac{\pi\,(\text{thetar}-\text{su thetar}+\text{su thetas})}{2\,\text{thetafc}}\right]^4 & \text{su} < \frac{-\text{thetafc}+\text{the}}{\text{thetar}-\text{thet}} \\ 610.8\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}} & \text{su} \geq \frac{-\text{thetafc}+\text{the}}{\text{thetar}-\text{thet}}\end{cases}\right)\right)$$

Out[80]=



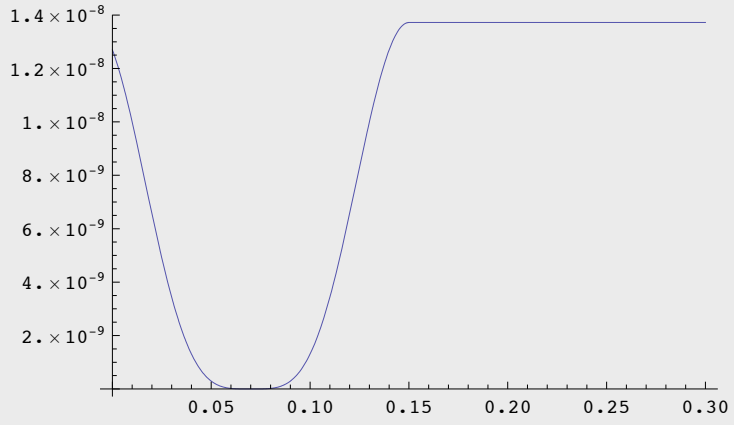
The above would results in positive soil evaporation even if su=0!

To fix this, we could subtract thetar from thetav and thetafc:

In[82]:=
```
vpsoil = { beta vpssoil + (1 - beta) vp /. beta → 1/4 (1 - Cos[(thetav-thetar) π / thetafc-thetar])^2   thetav - thetar < thetafc - thetar
         { vpssoil                                                                                       thetav - thetar ≥ thetafc - thetar ;
vpssoil = 610.8` ℯ^(17.27` tsoil / (237.3` + tsoil)) ;
vdsoil = (vpsoil - vp) / pressure ;
esoil = 0.03` × 18 vdsoil / 10^6
Plot[esoil /. {pressure → 100 000, vp → 1702, thetafc → 0.15`, tsoil → 30, thetar → 0.07`},
 {thetav, 0, 0.3`}]
esoil /. {pressure → 100 000, vp → 1702, thetafc → 0.15`, tsoil → 30, thetar → 0.07`, thetav → 0.07`}
Clear[vpsoil, vpssoil, vdsoil, esoil]
```

Out[85]=
$$\frac{1}{\text{pressure}}\,5.4\times10^{-7}\left(-\text{vp}+\begin{cases}\text{vp}\left(1-\frac{1}{4}\left(1-\text{Cos}\left[\frac{\pi\,(-\text{thetar}+\text{thetav})}{\text{thetafc}-\text{thetar}}\right]\right)^2\right)+ & -\text{thetar}+\text{thetav}<\text{thetafc}-\text{the}\\ 152.7\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}}\left(1-\text{Cos}\left[\frac{\pi\,(-\text{thetar}+\text{thetav})}{\text{thetafc}-\text{thetar}}\right]\right)^2 & \\ 610.8\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}} & -\text{thetar}+\text{thetav}\geq\text{thetafc}-\text{the}\end{cases}\right)$$

Out[86]=



Out[87]=
```
0.
```

In[89]:=
```
vpssoil = FortranForm[610.8` E^((17.27` tsoil)/(237.3` + tsoil))]
vpsoil =
 FortranForm[beta vpssoil + (1 - beta) vp /. beta -> 1 / 4 (1 - Cos[(thetav - thetar) / (thetafc - thetar) Pi]) ^ 2]
vpsoil = vpssoil
vdsoil = FortranForm[(vpsoil - vp) / pair]
```

Out[89]=
```
vpssoil = 610.8 * E ** ((17.27 * tsoil) / (237.3 + tsoil))
```

Out[90]=
```
vpsoil = vp * (1 - (1 - Cos ((Pi * (-thetar + thetav)) / (thetafc - thetar))) ** 2 / 4.) +
   (vpssoil * (1 - Cos ((Pi * (-thetar + thetav)) / (thetafc - thetar))) ** 2) / 4.
```

Out[91]=
```
vpsoil = vpssoil
```

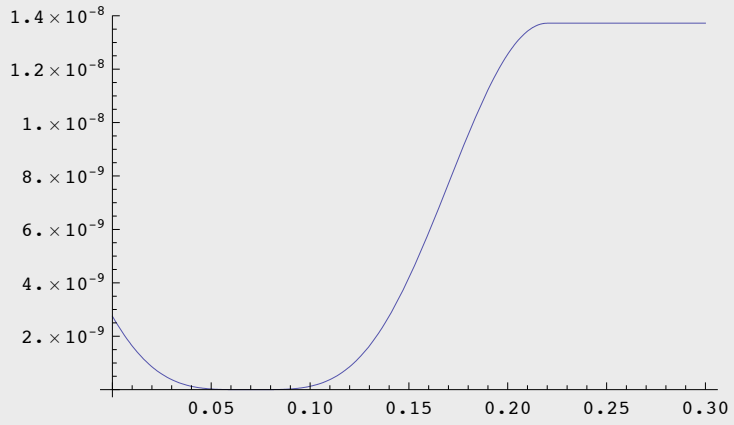
Out[92]=
```
vdsoil = (-vp + vpsoil) / pair
```

Alternatively, we coul replace only thetav by (thetav-thetar) in the above:

In[93]:=
```
vpsoil = {beta vpssoil + (1 - beta) vp /. beta → 1/4 (1 - Cos[((thetav-thetar) π)/thetafc])^2   thetav - thetar < thetafc
         {vpssoil                                                                               thetav - thetar ≥ thetafc ;
vpssoil = 610.8` E^((17.27` tsoil)/(237.3` + tsoil));
vdsoil = (vpsoil - vp)/pressure;
esoil = Simplify[(0.03` × 18 vdsoil)/10^6]
Plot[esoil /. {pressure → 100 000, vp → 1702, thetafc → 0.15`, tsoil → 30, thetar → 0.07`},
 {thetav, 0, 0.3`}]
Clear[vpsoil, vpssoil, vdsoil, esoil]
```

Out[96]=
$$\frac{1}{\text{pressure}}\left(-5.4\times10^{-7}\,\text{vp}+5.4\times10^{-7}\left(\begin{cases}\text{vp}+152.7\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}}\left(-1.+\cos\left[\frac{\pi\,(-\text{thetar}+\text{thetav})}{\text{thetafc}}\right]\right)^2-\text{vp}\,\sin\left[\frac{\pi\,(-\text{thetar}+\text{thetav})}{2\,\text{thetafc}}\right]^4 & \text{thetav}<\text{thetafc}+\text{thetar}\\ 610.8\,e^{\frac{17.27\,\text{tsoil}}{237.3+\text{tsoil}}} & \text{thetav}\geq\text{thetafc}+\text{thetar}\end{cases}\right)\right)$$

Out[97]=



In[99]:=
```
vpssoil = FortranForm[610.8` E^(17.27` tsoil/(237.3` +tsoil))]
vpsoil = FortranForm[beta vpssoil + (1 - beta) vp /. beta -> 1 / 4 (1 - Cos[(thetav - thetar) / thetafc Pi]) ^ 2]
vpsoil = vpssoil
vdsoil = FortranForm[(vpsoil - vp) / pair]
```

Out[99]=
```
vpssoil = 610.8 * E ** ((17.27 * tsoil) / (237.3 + tsoil))
```

Out[100]=
```
vpsoil = vp * (1 - (1 - Cos ((Pi * (-thetar + thetav)) / thetafc)) ** 2 / 4.) +
  (vpssoil * (1 - Cos ((Pi * (-thetar + thetav)) / thetafc)) ** 2) / 4.
```

Out[101]=
```
vpsoil = vpssoil
```

Out[102]=
```
vdsoil = (-vp + vpsoil) / pair
```

where vpssoil is the saturated vapour pressure at soil tempterature, vpsoil is the vapour pressure in the soil-air interface and vp is the actual atmospheric vapour pressure, all in Pa.

This would extend the 'unsaturated' evaporation beyond field capacity, so we will adopt the previous formulation, which is more in line with the original.

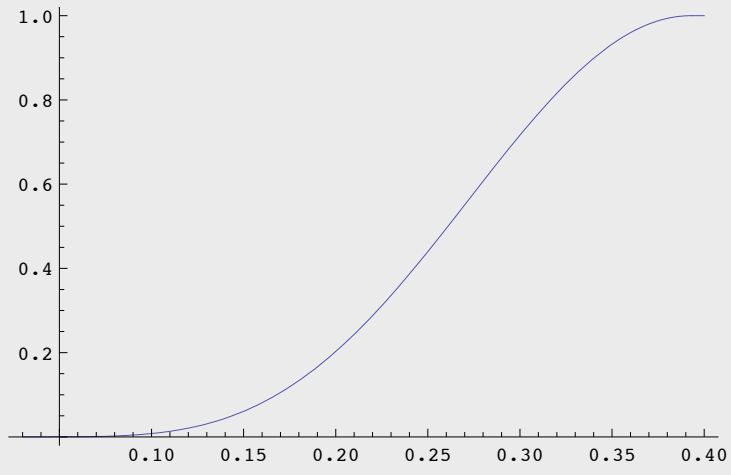
Yet another formulation for beta would depend on pcap:

In[103]:=
```
thetas = 0.4`; thetar = 0.03`; alpha = 0.827`; n = 4.549`; m = 0.0535`;
thetafc = thetar + (1 + alpha^n)^-m (-thetar + thetas)
pcap = ((-1 + ((thetar-thetas)/(thetar-thetav))^(1/m))^(1/n))/alpha;
beta = {1/4 (1 - Cos[((thetav-thetar) π)/(thetafc-thetar)])^2   thetav - thetar < thetafc - thetar
        1                                                       thetav - thetar ≥ thetafc - thetar};
betap = 1/(pcap + 1)
Plot[{beta}, {thetav, thetar, thetas}]
Plot[{betap}, {thetav, thetar, thetas}]
Plot[{betap, beta}, {thetav, thetar, thetas}]
Clear[thetas, thetar, alpha, n, m, thetafc, pcap, beta, betap]
```

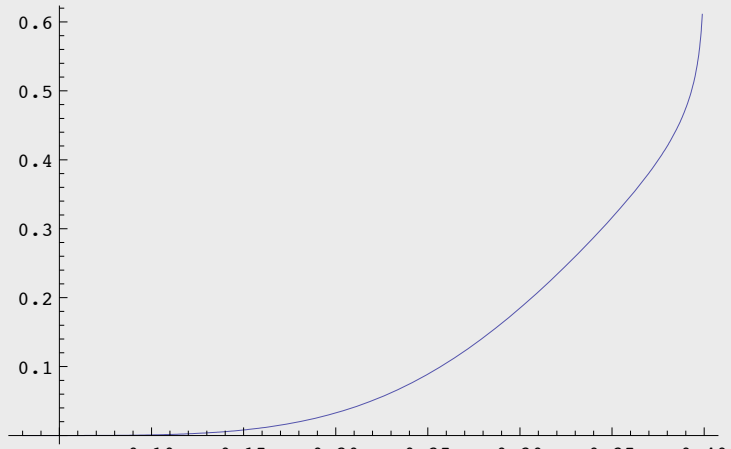
Out[104]=
```
0.393104
```

Out[107]=
$$\frac{1}{1 + 1.20919 \left(-1 + 8.49189 \times 10^{-9} \left(-\frac{1}{0.03-\text{thetav}}\right)^{18.6916}\right)^{0.219829}}$$

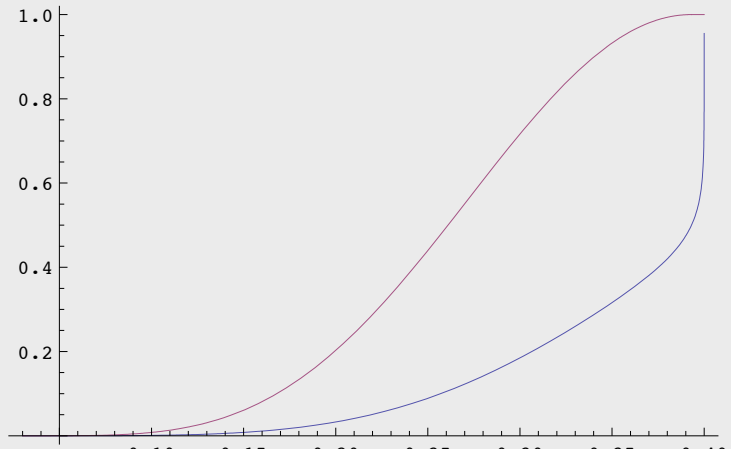
Out[108]=



Out[109]=



Out[110]=



In[112]:=

```
thetas = 0.41`; thetar = 0.065`; alpha = 7.5`; n = 1.89`; m = 1 - 1/n;
thetafc = thetar + (1 + alpha^n)^-m (-thetar + thetas)
pcap = ((-1 + ((thetar-thetas)/(thetar-thetav))^(1/m))^(1/n))/alpha;
beta = {(1/4 (1 - Cos[((thetav-thetar) π)/(thetafc-thetar)]))^2   thetav - thetar < thetafc - thetar
        1                                                         thetav - thetar ≥ thetafc - thetar};
betap = 1/(pcap + 1)
Plot[{beta}, {thetav, thetar, thetas}]
Plot[{betap}, {thetav, thetar, thetas}]
Plot[{betap, beta}, {thetav, thetar, thetas}]
Clear[thetas, thetar, alpha, n, m, thetafc, pcap, beta, betap]
```

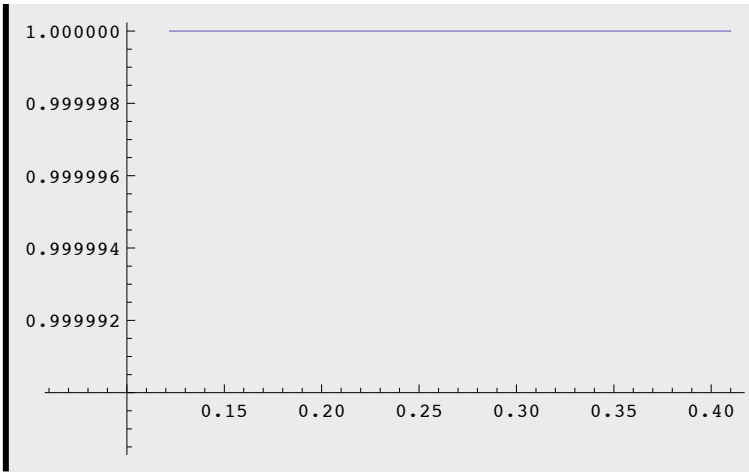
Out[113]=

```
0.121823
```

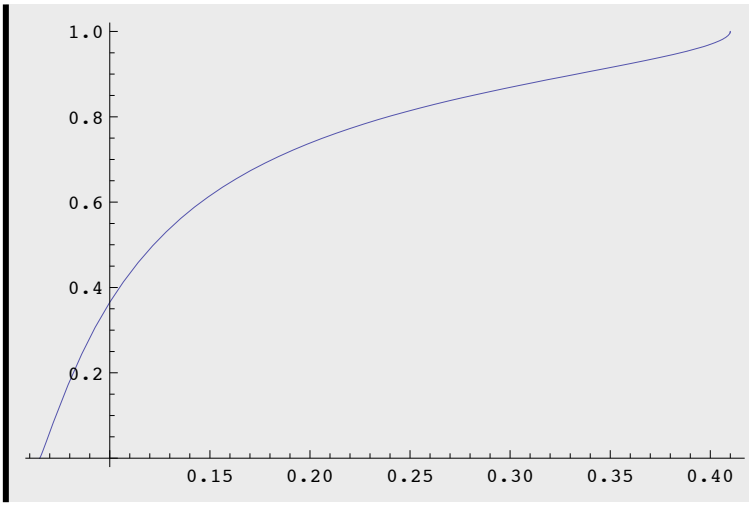
Out[116]=

$$\frac{1}{1 + 0.133333 \left(-1 + 0.104355 \left(-\frac{1}{0.065 - \text{thetav}}\right)^{2.1236}\right)^{0.529101}}$$

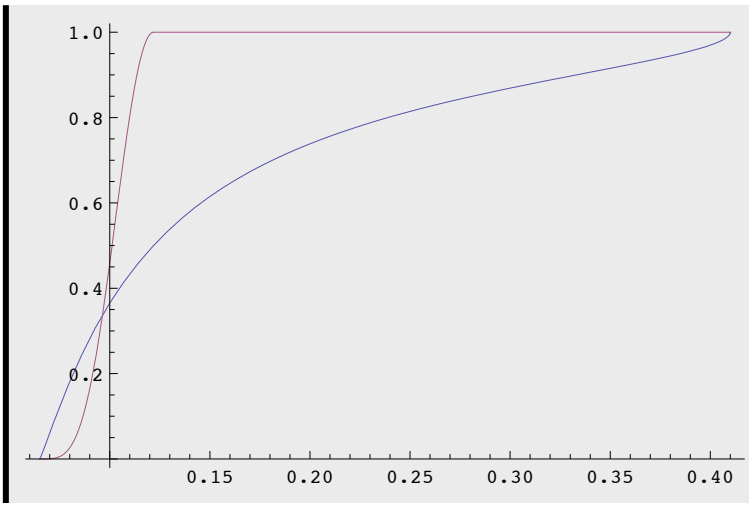
Out[117]=



Out[118]=



Out[119]=



■ No soil temperature

If soil temperature is not available, soil evaporation could be approximated using the beta from above and radiant heat flux. The latent heat of vaporisation (see dataprep6.nb) can be taken as 2.45 MJ/kg. If all incoming energy was converted to evaporation, we would have an evaporation of

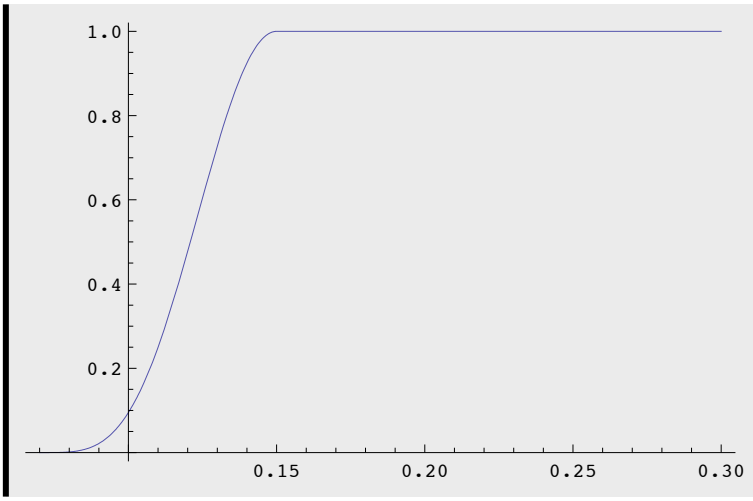
In[121]:=
```
epot = 1 / 2.45
```

Out[121]=
```
epot = 0.408163
```

kg water per MJ radiation.

In[122]:=
```
beta = { 1/4 (1 - Cos[ (thetav-thetar) π / thetafc-thetar ])^2   thetav - thetar < thetafc - thetar
       { 1                                                       thetav - thetar ≥ thetafc - thetar ;
Plot[beta /. {thetafc → 0.15`, thetar → 0.07`}, {thetav, 0.07`, 0.3`}]
Clear[beta]
```

Out[123]=



In[125]:=
```
beta = FortranForm[1 / 4 (1 - Cos[(thetav - thetar) / (thetafc - thetar) Pi])^2]
```

Out[125]=
```
beta = (1 - Cos ((Pi * (-thetar + thetav)) / (thetafc - thetar)))** 2 / 4.
```

Field capacity is thetav at p=1m:

In[126]:=
```
FullSimplify[Solve[ (-1 + su^(-1/m))^(1/n) / alpha == 1 /. su → (thetafc - thetar) / (thetas - thetar), thetafc]]
```

Out[126]=
```
{{thetafc → thetar + (1 + alpha^n)^-m (-thetar + thetas)}}
```

In[127]:=
```
thetafc = FortranForm[thetar + (1 + alpha^n)^-m (-thetar + thetas)]
```

Out[127]=
```
thetafc = thetar + (-thetar + thetas) / (1 + alpha ** n) ** m
```

In[128]:=
```
Solve[su == (thetav - thetar) / (thetas - thetar), thetav]
```

Out[128]=
```
{{thetav → thetar - su thetar + su thetas}}
```

In[129]:=
```
thetar - su thetar + su thetas /. {thetar → 0.078, thetas → 0.43, su → 10^-10}
```

Out[129]=
```
0.078
```

In[130]:=
```
thetav = FortranForm[thetar - su thetar + su thetas]
```

Out[130]=
```
thetav = thetar - su * thetar + su * thetas
```

With daily radiation in the range of 10-25 MJ, potential evaporation would equate to 4-10 mm. If we multiply this with a beta of around 0.01 to1 and a bare soil fraction of around 0.7 we would get a range of

In[131]:=
```
esoilmin = 10 / 2.45 * 0.01 * 0.7
esoilmax = 25 / 2.45 * 1 * 0.7
```

Out[131]=
```
esoilmin = 0.0285714
```

Out[132]=
```
esoilmax = 7.14286
```

Hutley estimated a range of 0.1 to 0.5 mm/day in the dry and 0.65 mm/day in wet season.

From PAR in mol/m2 we can get total radiation in MJ/m2 as shown above in the climate conversions:

In[133]:=
```
srad == par / 2.0699;
```

Now we will formulate esoil as a function of srad and surface soil moisture:

In[134]:=
```
esoil = FortranForm[baresoil * beta * par / 2.0699 / 2.45 / 1000]
```

Out[134]=
```
esoil = 0.00019718984748351244 * baresoil * beta * par
```

where esoil is in m/s if PAR is in mol/s.

However, the above would state that if baresoil->0, esoil->0, which would happen in the wet season. To prevent this, we will replace baresoil with 1-0.9pcveg:

In[135]:=
```
esoil = FortranForm[0.0002 * (1 - 0.9 * (pc + pcg)) * par * beta * omgu]
esoils = FortranForm[0.0002 * (1 - 0.9 * (pc + pcg)) * beta * omgo]
```

Out[135]=
```
esoil = 0.0002 * beta * omgu * par * (1 - 0.9 * (pc + pcg))
```

Out[136]=
```
esoils = 0.0002 * beta * omgo * (1 - 0.9 * (pc + pcg))
```

In[137]:=
```
esoildrymin = 15 / 2.45 * 0.01 * (1 - 0.9 * 0.3)
esoildrymax = 20 / 2.45 * 1 * (1 - 0.9 * 0.3)
esoilwetmin = 10 / 2.45 * (1 - 0.9 * 1)
esoilwetmax = 25 / 2.45 * (1 - 0.9 * 1)
```

Out[137]=
```
esoildrymin = 0.0446939
```

Out[138]=
```
esoildrymax = 5.95918
```

Out[139]=
```
esoilwetmin = 0.408163
```

Out[140]=
```
esoilwetmax = 1.02041
```

However, experiments showed that this use of beta over-estimates esoil at the beginning of the wet season at Howard springs, when grass cover has not picked up yet, but the surface gets wet. We achieved much better results when we replaced beta simply with su:

In[141]:=
```
esoil = FortranForm[0.0002 * (1 - 0.8 * (pc + pcg)) * par * suvec[1] * omgu]
esoils = FortranForm[0.0002 * (1 - 0.8 * (pc + pcg)) * suvec[1] * omgo]
```

Out[141]=
```
esoil = 0.0002 * omgu * par * (1 - 0.8 * (pc + pcg)) * suvec (1)
```

Out[142]=
```
esoils = 0.0002 * omgo * (1 - 0.8 * (pc + pcg)) * suvec (1)
```

Changes in state variables (taken from equations_VO_c_2a.nb)

In[143]:=
```
Solve[dys == (-Last[qblvec] - spgfcf - esoils + dys * epsln * suavg) / (epsln), dys]
```

Out[143]=
$$\left\{\left\{\text{dys} \to \frac{\text{esoils} + \text{spgfcf} + \text{Last}[\text{qblvec}]}{\text{epsln}\ (-1 + \text{suavg})}\right\}\right\}$$

In[144]:=
```
dys == FortranForm[(esoils + spgfcf + qblvec[nlayers]) / (epsln * (-1. + suavg))]
```

Out[144]=
```
dys = (esoils + spgfcf + qblvec (nlayers)) / (epsln * (-1. + suavg))
```

In[145]:=
$$\text{domgu} = \begin{cases} \frac{(cz-zr)\,(cz-ys)\,dys}{2\,((cz-zr)\,(cz-ys))^{3/2}} - \frac{dys}{\sqrt{(cz-zr)\,(cz-ys)}} & ys > zr \\ 0 & ys \le zr \end{cases};$$

```
domgu == FortranForm[FullSimplify[(cz - zr) (cz - ys) dys/(2 ((cz - zr) (cz - ys))^(3/2)) - dys/Sqrt[(cz - zr) (cz - ys)]]]
```

Out[146]=
```
domgu = -dys / (2. * Sqrt ((cz - ys) * (cz - zr)))
```

In[147]:=
$$\text{dyu} = \begin{cases} -\frac{dys\,(cz-zr)}{2\sqrt{(cz-ys)\,(cz-zr)}} & -ys + zr < 0 \\ -dys & \text{True} \end{cases};$$

```
dyu == FortranForm[FullSimplify[-(dys (cz - zr))/(2 Sqrt[(cz - ys) (cz - zr)])]]
```

Out[148]=
```
dyu = (dys * (-cz + zr)) / (2. * Sqrt ((cz - ys) * (cz - zr)))
```

In[149]:=
```
dsu1 = FortranForm[(-esoil + qbl1 + inf - ruptkvec1)/(epsln delyuvec1 omgu)]
```

Out[149]=
```
dsu1 = (-esoil + inf + qbl1 - ruptkvec1) / (delyuvec1 * epsln * omgu)
```

In[150]:=
```
dsui = FortranForm[(-qbliminus1 + qbli - rupktveci) / (epsln delyuveci omgu)]
```

Out[150]=
```
dsui = (qbli - qbliminus1 - rupktveci) / (delyuveci * epsln * omgu)
```

In[151]:=
```
dsunlayers =
 FortranForm[(-Q-1+nlayers[t] + Qnlayers[t] - Qr,nlayers[t])/(ε ωu δyu,nlayers) /.
   {ε → epsln, ωu → omgu, δyu,nlayers → delyuvecnlayers, Q-1+nlayers[t] → qblnlayersminus1,
    Qnlayers[t] → qblnlayers, Qr,nlayers[t] → ruptkvecnlayers, su,nlayers[t] → suvecnlayers,
    ωu'[t] → domgu, δyu,nlayers'[t] → dyu}]
```

Out[151]=
```
dsunlayers = (qblnlayers - qblnlayersminus1 - ruptkvecnlayers) / (delyuvecnlayers * epsln * omgu)
```

In[152]:=
```
wsnewvec[[i]] == sunewvec[[i]] * epsln * omgunew * delyunewvec[[i]];
wsnewvec[[nlayersnew]] == Total[wsoldvec] + ys epsln + io - Total[Take[wsnewvec, nlayersnew - 1]] -
   ysnew epsln;
```

In[154]:=
```
sunewvec[[nlayersnew]] == wsnewvec[[nlayersnew]] / (epsln delyunewvec[[nlayersnew]] * omgunew);
```

■ Prevention of su->1

If su gets close to 1, we must prevent dsu from being >1. To do this, we will decrease qbl accordingly.

In[155]:=
```
Solve[(-esoil + inf + qbl1 - ruptkvec1) / (delyuvec1 * epsln * omgu) == 0, qbl1]
```

Out[155]=
```
{{qbl1 → esoil - inf + ruptkvec1}}
```

In[156]:=
```
qbl1 == FortranForm[(esoil - inf + ruptkvec1)]
```

Out[156]=
```
qbl1 = esoil - inf + ruptkvec1
```

In[157]:=
```
Solve[-qbliminus1 + qbli - rupktveci == 0, qbli]
```

Out[157]=
```
{{qbli → qbliminus1 + rupktveci}}
```

In[158]:=
```
qbli = FortranForm[qbliminus1 + rupktveci]
```
Out[158]=
```
qbli = qbliminus1 + rupktveci
```

In[159]:=
```
Solve[qblnlayers - qblnlayersminus1 - ruptkvecnlayers == 0, qblnlayers]
```
Out[159]=
```
{{qblnlayers → qblnlayersminus1 + ruptkvecnlayers}}
```

In[160]:=
```
qblnlayers == FortranForm[qblnlayersminus1 + ruptkvecnlayers]
```
Out[160]=
```
qblnlayers = qblnlayersminus1 + ruptkvecnlayers
```

Vegetation parameters

Function to determine water transport costs (after wetdrystanmultilayerc11e.nb)

Water transport costs are assumed to be linearly dependent on rooting depth and projected cover (pc):

In[161]:=
```
FortranForm[cpcc = rootdepth * cpccf * pc]
```
Out[161]//FortranForm=
```
cpcc.eq.cpccf*pc*rootdepth
```

where cpccf has to be determined, and could be in the range of 4.86157×10^{-7}.

Cernusak et al. (2005) mentioned that specific leaf area at HS is 5.5 m2/kg, so with a LAI of 0.7 in the dry season, we get for the living biomass in the foliage:

In[162]:=
```
mdfol = 0.7 / 5.5 * 1000
```
Out[162]=
```
mdfol = 127.273
```

They also estimated the total above-ground sapwood density to be 0.0032 m^3/m2. Assuming a ratio of mqx to md of 1:1, we can approximate the sapwood density to

In[163]:=
```
mdwood = 0.0032 * 10 ^ 6
```
Out[163]=
```
mdwood = 3200.
```

So the cell walls in leaves contribute only by less than 5% to the total active cell walls.

Setting mass of cell walls and water storage capacity of living tissues:

In[164]:=
```
md = 0.7 / 5.5 * 1000 + 0.0032 * 10 ^ 6
```
Out[164]=
```
md = 3327.27
```

Cernusak et al. (2005) estimated total above-ground woody-tissue respiration of 297 g C m-2 ground area year-1. With 3200 g/m2 sapwood, this would amount to a maintenance respiration of the sapwood of:

In[165]:=
```
rsapf = N[297 / 12 / 3200 / (365 * 24 * 3600)]
```
Out[165]=
```
rsapf = 2.45255 × 10^-10
```

in mol/s per g sapwood.

Using the above formulation of cpcc==rootdepth*cpccf*pc, we can estimate cpccf:

In[166]:=
```
Solve[297 / 12 / (365 * 24 * 3600) == 2 * cpccf * 0.3, cpccf]
```

Out[166]=
$$\{\{\text{cpccf} \to 1.30803 \times 10^{-6}\}\}$$

This was derived from the above ground costs only. We could now assume that there is a certain amount of sapwood required for water transport per unit of pc and rootdepth, and any additional sapwood incurs additional costs.

In[167]:=
```
Solve[2.4525542237442924`*^-10 * sapwood == rootdepth * cpccf * pc /. cpccf → 1.0 * 10^-6, sapwood]
```

Out[167]=
$$\{\{\text{sapwood} \to 4077.38\ \text{pc rootdepth}\}\}$$

So a cpccf=10^-6 would be equivalent to assuming that 4000g *pc*rootdepth are necessary in sapwood for water transport. However, the measured respiration costs of sapwood apply only to above-ground sapwood. Below-ground, we could expect substantially higher maintenance costs due to the lack of oxygen and the necessity of anerobic respiration, which has only 2% of the efficiency of aerobic respiration. If we assumed that rsapf increases exponentially with soil depth, we would get for the water transport costs:

In[168]:=
```
Integrate[rsapf * Exp[depth], {depth, 0, rootdepth}]
```

Out[168]=
$$\left(-1 + e^{\text{rootdepth}}\right) \text{rsapf}$$

In[169]:=
```
cpcc == FullSimplify[0.5 transporttissue rsapf + 0.5 transporttissue (-1 + e^rootdepth) rsapf]
```

Out[169]=
$$\text{cpcc} == \left(0. + 0.5\ e^{\text{rootdepth}}\right) \text{rsapf transporttissue}$$

Now, the necessary transport tissue should depend linearly on pc*rootdepth, with an unknown proportionality constant (trtissf):

In[170]:=
```
cpcc == (0.5` e^rootdepth) rsapf transporttissue /. transporttissue → trtissf * rootdepth * pc
```

Out[170]=
$$\text{cpcc} == 0.5\ e^{\text{rootdepth}}\ \text{pc rootdepth rsapf trtissf}$$

Previously we assumed a cpccf of 10^-6, which would lead to cpcc for trees:

In[171]:=
```
cpcccalib == 10^-6 * 0.3 * 3
```

Out[171]=
$$\text{cpcccalib} == 9. \times 10^{-7}$$

So in order to get the same cpcc for the trees using the new formula, we would need to insert for trtissf:

In[172]:=
```
Solve[9 * 10^-7 == 0.5` trtissf e^rootdepth pc rootdepth rsapf /.
  {rootdepth → 2, pc → 0.3, rsapf -> 2.4525542237442924`*^-10}, trtissf]
```

Out[172]=
$$\{\{\text{trtissf} \to 1655.44\}\}$$

For grasses with 1m root depth we would then get:

In[173]:=
```
cpccgrass == 0.5` trtissf e^rootdepth pc rootdepth rsapf /.
 {trtissf → 406, rootdepth → 1, pc → 1, rsapf -> 1.3080289193302893`*^-6}
sapwoodgrass == trtissf * rootdepth * pc /. {trtissf → 406, rootdepth → 1, pc → 1}
sapwoodtrees == trtissf * rootdepth * pc /. {trtissf → 406, rootdepth → 3, pc → 0.3}
```

Out[173]=
$$\text{cpccgrass} == 0.000721785$$

Out[174]=
$$\text{sapwoodgrass} == 406$$

Out[175]=
$$\text{sapwoodtrees} == 365.4$$

If we however assume that rsapf increases exponentially with soil depth until it reaches a value of 50*rsapf (representing the decrease in efficiency to 2% efficiency of aerobic respirtaion), the maximum respiration rate would be reached at a depth of:

In[176]:=
```
Solve[rsapf * Exp[depth] == 50 * rsapf, depth]
```

Out[176]=
```
{{depth → Log[50]}}
```

In[177]:=
```
N[Log[50]]
```

Out[177]=
```
3.91202
```

We would then have to write for cpcc:

In[178]:=
```
Integrate[rsapf * Exp[depth], {depth, 0, Min[Log[50], rootdepth]}] + (Max[0, rootdepth - Log[50]]) * rsapf * 50
```

Out[178]=
$$\left(-1 + e^{\text{Min}[\text{rootdepth},\text{Log}[50]]}\right) \text{rsapf} + 50\ \text{rsapf}\ \text{Max}[0, \text{rootdepth} - \text{Log}[50]]$$

In[179]:=
```
cpcc ==
 FullSimplify[
  0.5 transporttissue rsapf +
    0.5 transporttissue ((-1 + e^Min[rootdepth,Log[50]]) rsapf + 50 rsapf Max[0, rootdepth - Log[50]]) /.
   transporttissue → trtissf * rootdepth * pc]
```

Out[179]=
$$\text{cpcc} == \begin{cases} \left(0. + 0.5\ e^{\text{rootdepth}}\right) \text{pc rootdepth rsapf trtissf} & \text{rootdepth} \le \text{Log}[50] \\ \text{pc rootdepth}\ (-72.8006 + 25.\ \text{rootdepth})\ \text{rsapf trtissf} & \text{True} \end{cases}$$

In[180]:=
```
cpcc == FortranForm[trtissf (0.5` e^rootdepth) pc rootdepth rsapf]
```

Out[180]=
```
cpcc == 0.5 * E ** rootdepth * pc * rootdepth * rsapf * trtissf
```

In[181]:=
```
cpcc == FortranForm[trtissf pc rootdepth (-72.80057513570365` + 25.` rootdepth) rsapf]
```

Out[181]=
```
cpcc == pc * rootdepth * (-72.80057513570365 + 25. * rootdepth) * rsapf * trtissf
```

Function to determine root respiration (rr) and root hydraulic conductivity or water uptake capacity

Rr: respiration rate [mol/s]

rsurf: fine root surface area per unit soil volume [m^2/m^3]

r: fine root radius

ltot: total length of fine roots per unit soil volume

The below values are taken from Eissenstat (1991): New Phytol. 118:63-68 and Bryla et al. (2001): Plant, Cell and Env. 24:781-790, and they apply to citrus roots. Bryla et al. give values of root respiration for a single fine root in the order of 10 nmol CO2 /(g DW)/s. The average fine root diameter of the measured roots was 0.6 mm.

In[182]:=
```
Rr == 10*^-9 mol / g / s
```

Out[182]=
$$\text{Rr} == \frac{\text{mol}}{100\,000\,000\ \text{g s}}$$

Eissenstat gave for the mass/vol relationships of different citrus roots values between 0.15 and 0.2 g/cm3. Thus we can calculate back from mass to volume. I m3 root volume should have 0.17E6 g dry weight. So the respiration rate for one m3 fine roots would be (in mol/s):

In[183]:=
```
Rr == 10*^-9 * 0.17*^6
```

Out[183]=
```
Rr == 0.0017
```

Using the information that the average root diameter is 0.6*10^-3m, we get the following relation between root volume and root surface:

In[184]:=
```
rsurf = 2 Pi r ltot;
rvol = Pi r^2 ltot;
rsurfperrvol == rsurf / rvol
Clear[rsurf, rvol]
```

Out[186]=
$$\text{rsurfperrvol} = \frac{2}{r}$$

which equates to rsurf=2/r*rvol or rvol=r/2*rsurf. So we get for Rr

In[188]:=
```
r = 0.3*^-3;
Rr == 0.0017 * rvol;
FortranForm[Rr == 0.0017 * (r / 2 * rsurf)]
Clear[r]
```

Out[190]//FortranForm=
```
Rr.eq.2.5499999999999994e-7*rsurf
```

And per year we get:

In[192]:=
```
Rr == 2.55*^-7 * 3600 * 24 * 365 * rsurf
```

Out[192]=
$$\text{Rr} = 8.04168\ \text{rsurf}$$

From the average root diameter of 0.6 mm, we can also calculate the maximum possible root surface area per soil volume, if all the voids in the soil were taken up by roots.

rsurfperrvol=2/r, so that if volume of voids = 0.3, we get:

In[193]:=
```
Solve[rsurfmax / rvol == 2 / r /. {rvol → 0.3, r → 0.3 * 10^-3}, rsurfmax]
```

Out[193]=
$$\{\{\text{rsurfmax} \to 2000.\}\}$$

In[194]:=
```
rsurfmax == 0.3 * 2 / (0.3 * 10^-3)
```

Out[194]=
$$\text{rsurfmax} = 2000.$$

Or in general:

In[195]:=
```
Solve[rsurfmax / rvol == 2 / r /. rvol → porosity, rsurfmax]
```

Out[195]=
$$\left\{\left\{\text{rsurfmax} \to \frac{2\ \text{porosity}}{r}\right\}\right\}$$

In[196]:=
```
rsurfmax == porosity * 2 / rrad;
```

Given that thicker transport roots must be present as well, and there has to be some space for air, setting the maximum root surface area per m^3 'soil' to 1000 m^2 should be reasonable.

Root water uptake

The below is taken from **E.** Raymond Hunt, Jr., Steven W. Running and C. Anthony Federer (1991): *Agricultural and Forest Meteorology,* 54 (1991) 169-195

In TABLE 3) the authors gave the following values for root resistance:

Rroot (MPa s m -3) = 790E6 (grass), 220E6 (shrubs), 11E6 (conifers), 7.2E6 (hardwood)

They then calculated root water uptake as

q-qstem=$\sum_i \left(\text{psisoil}_i - \text{psiroot}_i\right) \Big/ (\text{rsoil}_i + \text{rroot}_i)$

In the above, rsoil and rroot can either be resistivities (which would give us q as a flow density) or resistances (which would give us q as total flow rate). Resistance=resistivity*area, or Conductance=conductivity*area.

In the following, we will define the "r" as resistivities and hence q as a flow per unit root surface area.

psisoil and psiroot are expressed as negative potentials, with psiroot having to be more negative than psisoil to warrant water flow into the root.

Now, rsoil can be expressed in terms of the unsaturated hydraulic conductivity (kunsat), but we must take into account that rsoil is based on a potential difference (psisoil-psiroot), while kunsat was based on a potential gradient ((pcap1-pcap2)/dx).

In[197]:=
```
Solve[q == kunsat (p1 - p2) / dx, kunsat]
Solve[q == (p1 - p2) / rroot, rroot]
```

Out[197]=
$$\left\{\left\{\text{kunsat} \to \frac{\text{dx q}}{\text{p1 - p2}}\right\}\right\}$$

Out[198]=
$$\left\{\left\{\text{rroot} \to \frac{\text{p1 - p2}}{\text{q}}\right\}\right\}$$

Potential difference can be used if dx is fixed, as in the case of roots with a certain diameter. However, it does not seem to make sense to simply add a soil water resistivity term to the psiroot-driven flow, as this one is driven simply by the pressure difference between the inside and the outside of the root. If soil conductivity is low, water flow towards the root would be slow and hence psoil in the vicinity of the root would decline, which would feed back onto the root water uptake rate. Using the electrical analogue as in Hunt et al. (1991) however, this feedback could be simplified by assuming that the total pressure difference is between the inside of the root and the space between two roots. Thus, unsaturated flow could be expressed either as a function of the pressure difference between two points, or as a function of the pressure gradient, so that rsoil and kunsat are related as:

In[199]:=
```
Solve[(p1 - p2) / rsoil == kunsat (p1 - p2) / dx, rsoil]
```

Out[199]=
$$\left\{\left\{\text{rsoil} \to \frac{\text{dx}}{\text{kunsat}}\right\}\right\}$$

The flow of water through the soil towards the root can be compared with the flow of water towards a well,which is expressed following Marshall&Holmes,(eq.6.6) as:

Q=2*Pi K l r dH/dr,

where K is the hydraulic conductivity, l is the'thickness of the conducting zone of the aquifer', or the vertical root length in our case and r is the distance from the centre of the well. For the flow into a vertical root (r=rootrad), the above is equivalent to

Q=K*rsurf dH/dr,

as 2*Pi*l*r is the root surface area.

We will now express q=Q/rsurf and approximate dH/dx by (p1-p2)/dx, where p1 is the balance pressure in the root and p2 is the average soil balance pressure, while dx is half the average distance between roots:

In[200]:=
```
q == kunsat (p1 - p2) / dx
```

Out[200]=
$$\text{q} == \frac{\text{kunsat (p1 - p2)}}{\text{dx}}$$

The average distance between two roots (2*dx) can be calculated from the root surface area and root radius if we assume that all roots are straight lines at equal distances from each other. If n lines have length l and are a distance rdist apart, the volume of soil they would take up would be:

In[201]:=
```
svolume == l * n * rdist^2;
```

Now, l*n is the total root length in the volume of soil taken up by roots, so that we can write:

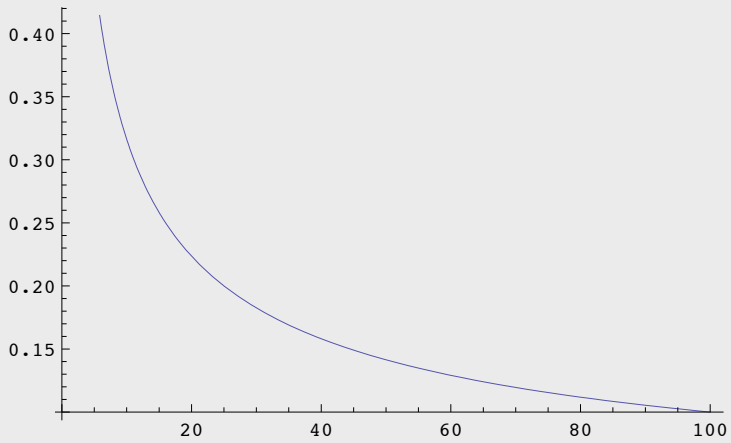
In[202]:=
```
Solve[svolume == l * n * rdist^2 /. {l * n → ltot}, rdist]
```

Out[202]=
$$\left\{\left\{\text{rdist} \to -\frac{\sqrt{\text{svolume}}}{\sqrt{\text{ltot}}}\right\}, \left\{\text{rdist} \to \frac{\sqrt{\text{svolume}}}{\sqrt{\text{ltot}}}\right\}\right\}$$

In[203]:=
```
rdist == Sqrt[svolume / ltot];
```

In[204]:=
```
Plot[Sqrt[svolume/ltot] /. svolume → 1, {ltot, 1, 100}]
```

Out[204]=



From rsurf and rradius we can get ltot and hence rdist

In[205]:=
```
Solve[rsurf == 2 Pi rradius ltot, ltot]
```

Out[205]=
$$\left\{\left\{\text{ltot} \to \frac{\text{rsurf}}{2\,\pi\,\text{rradius}}\right\}\right\}$$

In[206]:=
```
rdist == Sqrt[svolume / ltot] /. ltot -> rsurf/(2 π rradius)
```

Out[206]=
$$\text{rdist} == \sqrt{2\,\pi}\,\sqrt{\frac{\text{rradius svolume}}{\text{rsurf}}}$$

Following the Bryla et al. (2001), we will take an average fine root diameter of 0.6 mm, which would give us a radius of 0.3 mm.

In[207]:=
```
rdist == Sqrt[2 π] Sqrt[rradius svolume/rsurf] /. rradius → 0.3 * 10^-3
```

Out[207]=
$$\text{rdist} == 0.0434161\,\sqrt{\frac{\text{svolume}}{\text{rsurf}}}$$

Units:

In[208]:=
```
rdist == Sqrt[2 π] Sqrt[rradius svolume/rsurf] /. {rradius → m, svolume → m^3, rsurf → m^2}
```

Out[208]=
$$\text{rdist} == \sqrt{\text{m}^2}\,\sqrt{2\,\pi}$$

Taking dx as rdist/2 in the water uptake function, we get:

In[209]:=
```
rsoil == dx/kunsat /. dx → 0.04341607527349606` Sqrt[svolume/rsurf] / 2
```

Out[209]=
$$\text{rsoil} == \frac{0.021708\,\sqrt{\frac{\text{svolume}}{\text{rsurf}}}}{\text{kunsat}}$$

Or in general:

In[210]:=
```
rsoil == dx / kunsat /. dx -> Sqrt[2 π] Sqrt[rradius svolume / rsurf] / 2
```

Out[210]=
$$\text{rsoil} = \frac{\sqrt{\frac{\pi}{2}}\ \sqrt{\frac{\text{rradius svolume}}{\text{rsurf}}}}{\text{kunsat}}$$

kunsat is the conductivity per unit area (flow=kunsat*area*dP/dx)

In[211]:=
```
Solve[m^3 / s == kunsat m^2 Pa / m, kunsat]
```

Out[211]=
$$\left\{\left\{\text{kunsat} \to \frac{\text{m}^2}{\text{Pa s}}\right\}\right\}$$

or, if ΔP is expressed in m head:

In[212]:=
```
Solve[m^3 / s == kunsat m^2 m / m, kunsat]
```

Out[212]=
$$\left\{\left\{\text{kunsat} \to \frac{\text{m}}{\text{s}}\right\}\right\}$$

Or, using rsoil:

In[213]:=
```
rsoil == dx / kunsat /. {dx → m, kunsat → m / s}
```

Out[213]=
$$\text{rsoil} = \text{s}$$

Thus, if we express psoil and proot in m head as well, we need to convert rroot from [MPa s/m] to [m s/m].

In[214]:=
```
MPa = 10^6 kg / m^2 m / s^2
meter = m * 1000 kg / m^3 * 9.81 m / s^2
MPapmeter == meter / MPa
Clear[MPa, meter]
```

Out[214]=
$$\frac{1\,000\,000\ \text{kg}}{\text{m s}^2}$$

Out[215]=
$$\frac{9810.\ \text{kg}}{\text{m s}^2}$$

Out[216]=
$$\text{MPapmeter} = 0.00981$$

Rroot for citrus can be calculated from Kroot, as given by Huang&Eissenstat (2000), and used in rootcosts.nb:

In[218]:=
```
rroot == 1 / kroot == 1 / (10 * 10^-7 m / s / MPa)
```

Out[218]=
$$\text{rroot} = \frac{1}{\text{kroot}} = \frac{1\,000\,000\ \text{MPa s}}{\text{m}}$$

So we can convert the above to m s/m by dividing by 0.00981:

In[219]:=
```
rrootm == 1 000 000 MPa s / m / (0.00981 MPa / m)
```

Out[219]=
$$\text{rrootm} = 1.01937 \times 10^8\ \text{s}$$

In[220]:=
```
ruptk == (psoil - proot) / (rrootm + rsoil) * rsurf /. rsoil -> (Sqrt[2 π] Sqrt[(rradius*svolume)/rsurf])/kunsat / 2
```

Out[220]=
$$\text{ruptk} == \frac{(-\text{proot} + \text{psoil})\ \text{rsurf}}{\text{rrootm} + \frac{\sqrt{\frac{\pi}{2}}\ \sqrt{\frac{\text{rradius svolume}}{\text{rsurf}}}}{\text{kunsat}}}$$

where

rroot: root water uptake resistivity per MPa pressure difference between root and soil and per unit root surface area [MPa s m -1]

rrootm: same in [s]

ruptk: water uptake rate per unit soil volume [m^3/m^3/s]

p1, p2, psoil, proot: hydraulic head [m]

rsurf: root surface area per unit soil volume [m^2/m^3]

Units:

However, we express both psoil and proot as positive, so it should be proot-psoil:

In[221]:=
```
ruptk == (proot - psoil) rsurf / (rrootm + rsoil) /. {proot → 10 m, psoil → 5 m, rsurf → 1 m^2, rrootm → 1.0193679918450561`*^8 s,
  rsoil → 10^4 s}
```

Out[221]=
$$\text{ruptk} == \frac{4.90452 \times 10^{-8}\ \text{m}^3}{\text{s}}$$

Root water uptake in soil profile:

In the below, surfinit denotes the root surface per m^3 soil and rootdepth denotes the maximum rooting depth. All values stored in vectors (rsurfvec etc.) are totals per m^2 surface area of each sublayer.

In[222]:=
```
nlayers = 10; delyu = 1; surfinit = 0.2; rootdepth = 5.5; depth = 0; omgu = 1; rsurfvec = Table[0, {i, nlayers}];
Do[depth = depth + delyu; If[depth < rootdepth, rsurfvec[[i]] = surfinit * delyu * omgu,
   rsurfvec[[i]] = surfinit * rootdepth - Total[Take[rsurfvec, i - 1]]], {i, Ceiling[rootdepth / delyu]}];
rsurfvec
Total[rsurfvec]
surfinit * rootdepth
Clear[nlayers, delyu, surfinit, rootdepth, depth, rsurfvec]
```

Out[224]=
```
{0.2, 0.2, 0.2, 0.2, 0.2, 0.1, 0, 0, 0, 0}
```

Out[225]=
```
1.1
```

Out[226]=
```
1.1
```

We can now calculate root water uptake as:

In[228]:=
```
nlayers = 10; delyu = 1; delyuvec = Table[delyu, {i, nlayers + 1}]; delyuvec[[nlayers + 1]] = 0;
surfinit = 0.2; rootdepth = 5.5; mpbar = 10.2; md = 3327.27; mqx = md; mq = 0.95 mqx;
prootmvec = Table[0, {i, nlayers}];
depth = 0; ivec = Table[i, {i, 1, nlayers}];
ivec = Table[i, {i, 1, nlayers}];
phydrostaticvec = 0.5 * ((ivec - 1) delyu + 0.5 Take[delyuvec, nlayers]);
prootmvec = mpbar (-mq + mqx) (750 - 750 mqx/(md+mqx) + (md+mqx)/mqx) / (md + mqx) - phydrostaticvec;
prootmvec
Do[
  depth = depth + (delyuvec[[i]] + delyuvec[[i + 1]]) / 2;
  prootmvec[[i]] = mpbar (-mq + mqx) (750 - 750 mqx/(md+mqx) + (md+mqx)/mqx) / (md + mqx) - depth, {i, 1, nlayers}];
prootmvec
Clear[nlayers, delyu, delyuvec, surfinit, rootdepth, depth, rsurfvec, nlayers, prootmvec,
 depth, mpbar, md, mqx, mq, phydrostaticvec, ivec]
```

Out[234]=
```
{95.885, 95.385, 94.885, 94.385, 93.885, 93.385, 92.885, 92.385, 91.885, 91.385}
```

Out[236]=
```
{95.135, 94.135, 93.135, 92.135, 91.135, 90.135, 89.135, 88.135, 87.135, 86.635}
```

In[238]:=
```
FortranForm[phydrostaticvec[i] == 0.5 * ((i - 1) delyu + 0.5 delyuvec[i])]
```

Out[238]//FortranForm=
```
phydrostaticvec(i).eq.0.5*(delyu*(-1 + i) + 0.5*delyuvec(i))
```

In[239]:=
```
FortranForm[prootmvec == mpbar (-mq + mqx) (750 - 750 mqx/(md+mqx) + (md+mqx)/mqx) / (md + mqx) - phydrostaticvec]
```

Out[239]//FortranForm=
```
      prootmvec.eq.
     (mpbar*(-mq + mqx)*(750 - (750*mqx)/(md + mqx) + (md + mqx)/mqx))/(md + mqx) -
      phydrostaticvec
```

In[240]:=
```
FortranForm[prootmvec[i] == mpbar (-mq + mqx) (750 - 750 mqx/(md+mqx) + (md+mqx)/mqx) / (md + mqx) - phydrostaticvec[i] /.
  phydrostaticvec[i] -> 0.5 * ((i - 1) delyu + 0.5 delyuvec)]
```

Out[240]//FortranForm=
```
      prootmvec(i).eq.
     -0.5*(0.5*delyuvec + delyu*(-1 + i)) +
      (mpbar*(-mq + mqx)*(750 - (750*mqx)/(md + mqx) + (md + mqx)/mqx))/(md + mqx)
```

In[241]:=
```
rsoilvec = FortranForm[Sqrt[Pi / 2] Sqrt[(rradius*svolume)/rsurfvec] / kunsatvec]
```

Out[241]=
```
rsoilvec = (Sqrt (Pi / 2.) * Sqrt ((rradius * svolume) / rsurfvec)) / kunsatvec
```

In[242]:=
```
ruptkvec = FortranForm[(prootmvec - pcapvec) rsurfvec / (rrootm + rsoilvec)]
```

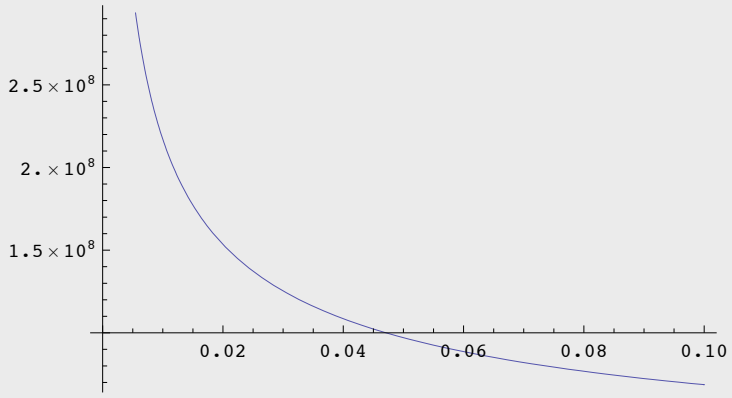
Out[242]=
```
ruptkvec = ((-pcapvec + prootmvec) * rsurfvec) / (rrootm + rsoilvec)
```

In[243]:=
```
ruptkvec = FortranForm[(prootm - pcapvec) rsurfvec / rrootm + rsoilvec /. rsoilvec -> Sqrt[π / 2] Sqrt[rradius*svolume / rsurfvec] / kunsatvec]
```

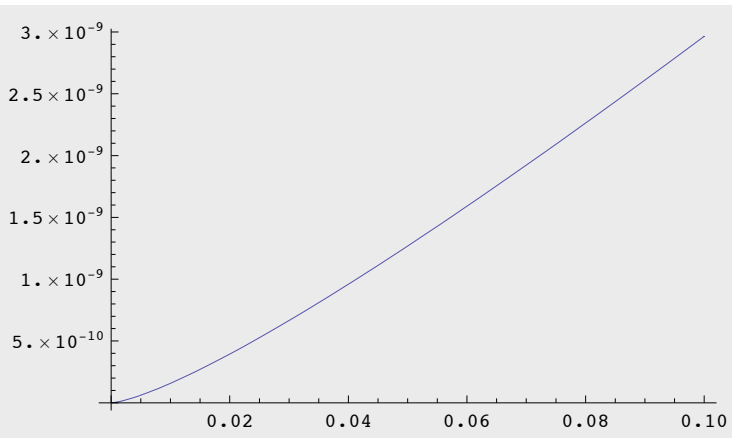
Out[243]=
```
ruptkvec = ((-pcapvec + prootm) * rsurfvec) /
  (rrootm + (Sqrt (Pi / 2.) * Sqrt ((rradius * svolume) / rsurfvec)) / kunsatvec)
```

In[244]:=
```
prootm = 10; pcapvec = 5; rrootm = 10^8; rradius = 3/10^6; kunsatvec = 1/10^10; svolume = 1;

Plot[Sqrt[π/2] Sqrt[rradius svolume / rsurfvec] / kunsatvec, {rsurfvec, 0, 0.1`}]

Plot[(prootm - pcapvec) rsurfvec / rrootm + rsoilvec /. rsoilvec → Sqrt[π/2] Sqrt[rradius svolume / rsurfvec] / kunsatvec, {rsurfvec, 0, 0.1`}]

Clear[prootm, pcapvec, rootm, rradius, kunsatvec, svolume];
```

Out[245]=



Out[246]=



With increasing rsurf, the increase in root water uptake becomes steaper, because rsurfvec/rsoilvec are more dominant at lower rsurf.

We have to limit transpiration to a fraction of Total[ruptkvec] when mq<=0.9mqss to prevent the plants from drying out. transpiration in mol/s can be converted to etm in m/s and vice versa:

In[248]:=
```
Solve[etm == transp * 18 / 10^-6, transp]
```

Out[248]=
$$\left\{\left\{\text{transp} \to \frac{\text{etm}}{18\,000\,000}\right\}\right\}$$

In[249]:=
```
transp = FortranForm[etm / 18 000 000]
```

Out[249]=
```
transp = etm / 1.8 e7
```

The change in tissue water storage can be calculated as:

In[250]:=
```
dmq == FortranForm[(Total[ruptkvec] - etm) * 10^6]
```

Out[250]=
```
dmq == 1 000 000 * (-etm + Total (ruptkvec))
```

The amount of water stored in tissues that would lead to steady-state (mqss) could then be written as (waterbalancemultiveg7a.nb):

In[251]:=
```
Clear[i];
Sum[ruptkvec[i], {i, nlayers}] == et /. ruptkvec[i] → (prootmvec[i] - psoilvec[i]) rsurfvec[i] / (rrootm + rsoilvec[i]) /.
 prootmvec[i] → ptissue - phydrostaticvec[i]
```

Out[252]=
$$\sum_{i=1}^{\text{nlayers}} \frac{(\text{ptissue} - \text{phydrostaticvec}[i] - \text{psoilvec}[i])\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} == \text{et}$$

We can re-write the above as

In[253]:=
```
Sum[Collect[((ptissue - phydrostaticvec[i] - psoilvec[i]) rsurfvec[i]) / (rrootm + rsoilvec[i]), ptissue] ==
 et
```

Out[253]=
$$\sum_{i=1}^{\text{nlayers}} \left(\frac{\text{ptissue}\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} + \frac{(-\text{phydrostaticvec}[i] - \text{psoilvec}[i])\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} \right) == \text{et}$$

Separating the sum:

In[254]:=
```
Sum[(ptissue rsurfvec[i]) / (rrootm + rsoilvec[i]), {i, 1, nlayers}] + Sum[((-phydrostaticvec[i] - psoilvec[i]) rsurfvec[i]) / (rrootm + rsoilvec[i]), {i, 1, nlayers}] == et
```

Out[254]=
$$\sum_{i=1}^{\text{nlayers}} \frac{\text{ptissue}\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} + \sum_{i=1}^{\text{nlayers}} \frac{(-\text{phydrostaticvec}[i] - \text{psoilvec}[i])\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} == \text{et}$$

Separating ptissue, inserting the function for ptissue and solving for mqss:

In[255]:=
```
FullSimplify[
 Solve[ptissue Sum[rsurfvec[i] / (rrootm + rsoilvec[i]), {i, 1, nlayers}] + Sum[((-phydrostaticvec[i] - psoilvec[i]) rsurfvec[i]) / (rrootm + rsoilvec[i]), {i, 1, nlayers}] == et /.
  ptissue -> (mpbar (-mqss + mqx) (750 - 750 mqx / (md + mqx) + (md + mqx) / mqx)) / (md + mqx), mqss]]
```

Out[255]=
$$\left\{\left\{\text{mqss} \to \left(\text{mqx}\left(\text{mpbar}\left(\text{md}^2 + 752\,\text{md}\,\text{mqx} + \text{mqx}^2\right)\sum_{i=1}^{\text{nlayers}} \frac{\text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]} - (\text{md} + \text{mqx})^2\left(\text{et} - \sum_{i=1}^{\text{nlayers}} \frac{(-\text{phydrostaticvec}[i] - \text{psoilvec}[i])\ \text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]}\right)\right)\right) \Big/ \left(\text{mpbar}\left(\text{md}^2 + 752\,\text{md}\,\text{mqx} + \text{mqx}^2\right)\sum_{i=1}^{\text{nlayers}} \frac{\text{rsurfvec}[i]}{100\,000\,000 + \text{rsoilvec}[i]}\right)\right\}\right\}$$

In[256]:=

In[257]:=
```
mqss =
 FortranForm[
  (mqx (mpbar (md^2 + 752 md mqx + mqx^2) Sum[rsurfvec[i]/(rrootm + rsoilvec[i]), {i, 1, nlayers}] -
       (md + mqx)^2 (et - Sum[((-phydrostaticvec[i] - psoilvec[i]) rsurfvec[i])/(rrootm + rsoilvec[i]), {i, 1, nlayers}])) /
   (mpbar (md^2 + 752 md mqx + mqx^2) Sum[rsurfvec[i]/(rrootm + rsoilvec[i]), {i, 1, nlayers}])]
```

Out[257]=
```
mqss =
 (mqx * (mpbar * (md ** 2 + 752 * md * mqx + mqx ** 2) * Sum (rsurfvec (i) / (100 000 000 + rsoilvec (i)), List (i, 1,
          nlayers)) - (md + mqx) ** 2 * (et - Sum (((-phydrostaticvec (i) - psoilvec (i)) * rsurfvec (i)) /
            (100 000 000 + rsoilvec (i)), List (i, 1, nlayers))))) /
  (mpbar * (md ** 2 + 752 * md * mqx + mqx ** 2) * Sum (rsurfvec (i) / (100 000 000 + rsoilvec (i)),
     List (i, 1, nlayers)))
```

Optimisation of root distribution

To optimise vertical root distribution, we will calculated daily root water uptake with two root surface distributions, the actual one and one with increased rsurf in each layer by growthmax. Then we will compare the potential increase in water uptake in each layer if rsurf was increased by growthmax, and standardise the increase in each layer by the maximum increase:

In[258]:=
```
ruptknewvec = {-0.1, 0.05, -0.2}
ruptkvec = {-0.08, 0.04, -0.13}
reff = (ruptknewvec - ruptkvec) / Max[ruptknewvec - ruptkvec]
Clear[ruptknewvec, ruptkvec]
```

Out[258]=
```
{-0.1, 0.05, -0.2}
```

Out[259]=
```
{-0.08, 0.04, -0.13}
```

Out[260]=
```
reff = {-2., 1., -7.}
```

To keep the values of reff between 0 and 1, we will need to set reff for all layers with negative water uptake to 0.

Assimilation model from Stanass6a.nb

Function to determine k from temperature (after Medlyn et al 2002)

In[262]:=
```
vomax / vcmax = 0.21;
```

after Badger & Andrews (1974)

In[263]:=
```
kc = 404.9 * Exp[79 430 (tk - 298) / (298 * 8.314 * tk)];
```

In[264]:=
```
ko = 278.4 * Exp[36 380 (tk - 298) / (298 * 8.314 * tk)];
```

from Bernacchi et al. 2001, where kc and ko are Michaelis-Menten coefficients of Rubisco, kc in micromol/mol and ko in mmol/mol, 8.314 is the universal gas constant in J/mol/K and tk is the leaf temperature in K.

To convert from tk to t in Celcius:

In[265]:=
```
tk = temp + 273;
```

Converting kc and ko to mol/mol, we can write:

In[266]:=
```
kc = 404.9 * Exp[79 430 (temp - 25) / (298 * 8.314 * (temp + 273))] * 10^-6
```

Out[266]=
$$0.0004049\, e^{\frac{32.0596\ (-25+temp)}{273+temp}}$$

In[267]:=
```
ko = 278.4 * Exp[36 380 (temp - 25) / (298 * 8.314 * (temp + 273))] * 10^-3
```

Out[267]=
$$0.2784\, e^{\frac{14.6837\ (-25+temp)}{273+temp}}$$

For later calculations it is convenient to lump (vomax kc)/(vcmax ko) into one variable kcko:

In[268]:=
```
kcko = 0.21 kc / ko;
```

From Stanass6a.nb, we know that gammastar is a linear function of kcko:

In[269]:=
```
gammastar == 0.5 * oa * kcko;
FortranForm[%]
Clear[kc, ko, kcko]
```

Out[270]//FortranForm=
```
gammastar.eq.0.00015271012931034484*E**((17.37588251723865*(-25 + temp))/(273 + temp))*oa
```

However, Medlyn (2002) cites Bernacchi et al. (2001) to give gammastar in micromol/mol as

In[272]:=
```
gammastar == 42.75 Exp[37 830 (tk - 298) / (298 r tk)];
```

Converting to mol/mol and temp in Celcius:

In[273]:=
```
gammastar == 10^-6 * 42.75 Exp[37 830 (tk - 298) / (298 r tk)] /. {tk → temp + 273}
FortranForm[%]
```

Out[273]=
$$\text{gammastar} == 0.00004275\, e^{\frac{18\,915\ (-25+temp)}{149\ r\ (273+temp)}}$$

Out[274]//FortranForm=
```
gammastar.eq.0.000042749999999999996*E**((18915*(-25 + temp))/(149.*r*(273 + temp)))
```

In[275]:=
```
FullSimplify[10^-6 * 42.75 Exp[37 830 (tk - 298) / (298 r tk)] /. {tk → temp + 273}]
```

Out[275]=
$$0.00004275\, e^{\frac{18\,915\ (-25+temp)}{149\ r\ (273+temp)}}$$

In[276]:=
```
gammastar == 0.000042749999999999996` e^(N[18 915/149] (-25+temp))/(r (273+temp))
```

Out[276]=
$$\text{gammastar} == 0.00004275\, e^{\frac{126.946\ (-25+temp)}{r\ (273+temp)}}$$

However, eq. (8) in Bernacchi (01) gives:

In[277]:=
```
gammastar == Exp[c - dHa / (r tk)]
```

Out[277]=
$$\text{gammastar} == e^{c-\frac{dHa}{r\ tk}}$$

with value at 25 degrees C: 42.75 umol/m2/s

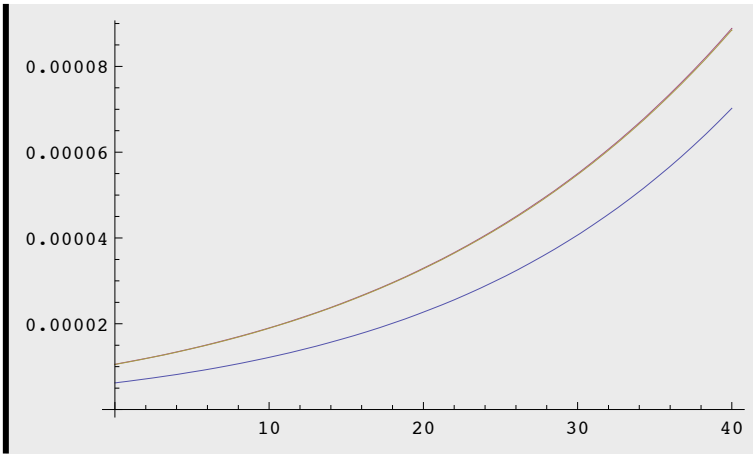
c=19.02

dHa=37.83 kJ/mol

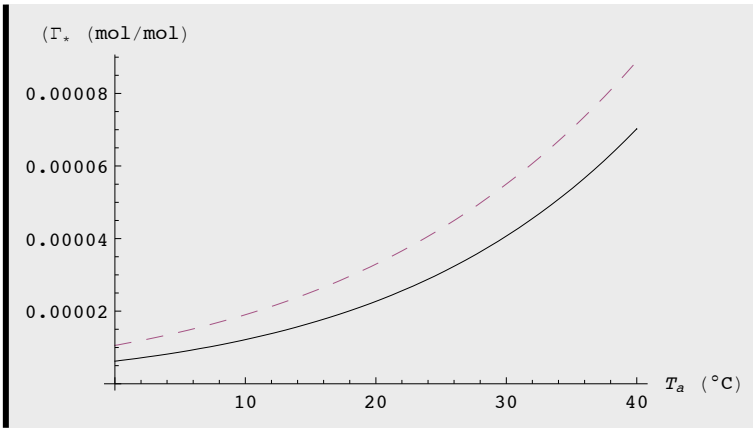
Both the original Bernacchi(2001) equation and the one quoted by Medlyn(2002) are very similar, but different from the one derived from the temperature dependencies of kc and ko:

In[278]:=
```
r = 8.314`;
oa = 0.2`;
Plot[{0.00015271012931034484` e^((17.37588251723865` (-25+temp))/(273+temp)) oa,
  0.00004274999999999996` e^((126.94630872483222` (-25+temp))/(r (273+temp))), (e^(19.02`-37 830/(8.314` (273+temp))))/10^6}, {temp, 0, 40}]
Plot[{0.00015271012931034484` e^((17.37588251723865` (-25+temp))/(273+temp)) oa, 0.00004274999999999996` e^((18 915 (-25+temp))/(149 r (273+temp)))},
 {temp, 0, 40}, PlotStyle → {GrayLevel[0], Dashing[{0.03`}]}, AxesLabel → {"Tₐ (°C)", "(Γ* (mol/mol)"}]
Clear[r, oa]
```

Out[280]=



Out[281]=



As K_c and K_o were measured in different plants, we will use the direct estimate of `gammastar` as in the last equation:

In[283]:=
```
gammastar == FortranForm[0.00004275 e^((18 915 (-25+temp))/(149 r (273+temp)))]
```

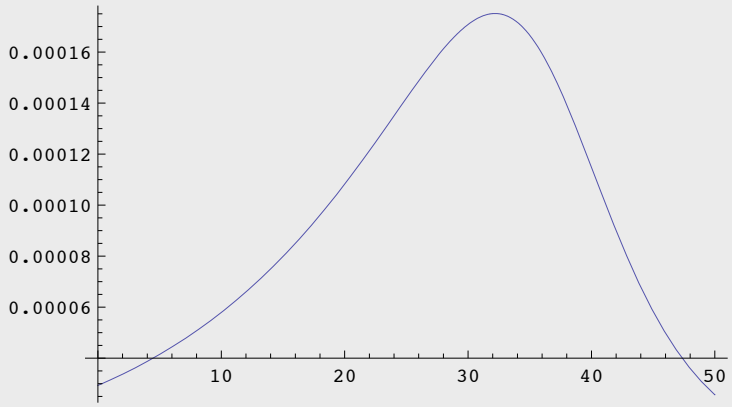
Out[283]=
```
gammastar == 0.00004275 * E ** ((18 915 * (-25 + temp)) / (149. * r * (273 + temp)))
```

Function to determine jmax(temp)

Parameters for E. pauciflora, from Medlyn 2002, Table 3.

In[284]:=
```
k25 = 141.94` / 10^6;
kopt = 175.13` / 10^6;
ha = 43.79` 10^3;
hd = 200 10^3;
topt = 32.19` + 273;
r = 8.314`;

Plot[kopt (hd E^(ha (tair+273-topt)/((tair+273) r topt))) / (hd - ha (1 - E^(hd (tair+273-topt)/((tair+273) r topt)))), {tair, 0, 50}]

Clear[k25, kopt, ha, hd, topt, r]
```

Out[290]=



In the above all parameters are transformed into SI units and topt is transformed to K, while tair is left in degrees celcius.

To calculate what a given jmax at t25 (jmax25) would be at tair, we need to normalise the above:

In[292]:=
```
jmax / jmax25 ==
 FullSimplify[
  kopt
   (hd Exp[ha (tair + 273 - topt) / ((tair + 273) r topt)] /
       (hd - ha (1 - Exp[hd (tair + 273 - topt) / ((tair + 273) r topt)]))) /
    (kopt
      (hd Exp[ha (25 + 273 - topt) / ((25 + 273) r topt)] /
         (hd - ha (1 - Exp[hd ((25 + 273) - topt) / ((25 + 273 ) r topt)])))))]
```

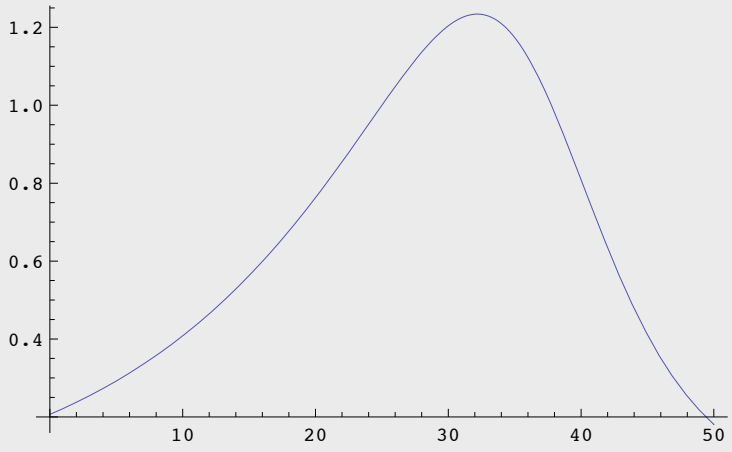
Out[292]=

$$\frac{\text{jmax}}{\text{jmax25}} == \frac{e^{\frac{\text{ha}\,(-25+\text{tair})}{298\,r\,(273+\text{tair})}}\left(\left(-1+e^{-\frac{\text{hd}\,(-298+\text{topt})}{298\,r\,\text{topt}}}\right)\text{ha}+\text{hd}\right)}{\left(-1+e^{\frac{\text{hd}\,(273+\text{tair}-\text{topt})}{r\,(273+\text{tair})\,\text{topt}}}\right)\text{ha}+\text{hd}}$$

In[293]:=

```
k25 = 141.94` / 10^6;
kopt = 175.13` / 10^6;
ha = 43.79` 10^3;
hd = 200 10^3;
topt = 32.19` + 273;
r = 8.314`;
Plot[ e^(ha (-25+tair)/(298 r (273+tair))) ((-1 + e^(-hd (-298+topt)/(298 r topt))) ha + hd) / ((-1 + e^(hd (273+tair-topt)/(r (273+tair) topt))) ha + hd), {tair, 0, 50}]
Clear[k25, kopt, ha, hd, topt, r]
```

Out[299]=

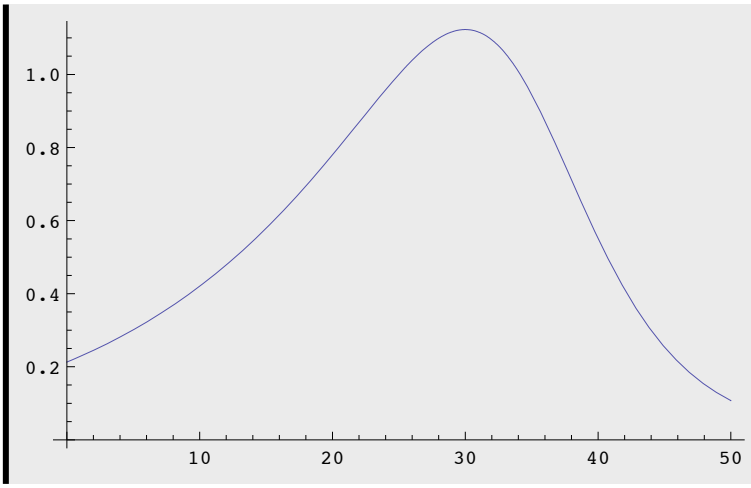


In[301]:=
```
k25 = 141.94` / 10^6;
kopt = 175.13` / 10^6;
ha = 43.79` 10^3;
hd = 200 10^3;
r = 8.314`;

jmax25 == kopt (hd E^(ha (25+273-topt)/((25+273) r topt))) / (hd - ha (1 - E^(hd (25+273-topt)/((25+273) r topt))))

jmaxoverjmax25 == FullSimplify[ E^(ha (-25+tair)/(298 r (273+tair))) ((-1 + E^(-hd (-298+topt)/(298 r topt))) ha + hd) / ((-1 + E^(hd (273+tair-topt)/(r (273+tair) topt))) ha + hd)]

Plot[ E^(ha (-25+tair)/(298 r (273+tair))) ((-1 + E^(-hd (-298+topt)/(298 r topt))) ha + hd) / ((-1 + E^(hd (273+tair-topt)/(r (273+tair) topt))) ha + hd) /. topt → 303, {tair, 0, 50}]

Clear[k25, kopt, ha, hd, topt, r]
```

Out[306]=
$$\text{jmax25} == \frac{35.026\, e^{\frac{17.6746\,(298-\text{topt})}{\text{topt}}}}{200\,000 - 43\,790.\left(1 - e^{\frac{80.7242\,(298-\text{topt})}{\text{topt}}}\right)}$$

Out[307]=
$$\text{jmaxoverjmax25} == \frac{e^{\frac{23\,613.9+17.6746\,\text{tair}}{273.+\text{tair}}}\left(156\,210. + 3.83092\times 10^{-31}\, e^{24\,055.8/\text{topt}}\right)}{156\,210.\, e^{\frac{24\,055.8}{273.+\text{tair}}} + 43\,790.\, e^{24\,055.8/\text{topt}}}$$

Out[308]=



In[310]:=
```
FortranForm[jmax == E^(ha (-25+tair)/(298 r (273+tair))) ((-1 + E^(-hd (-298+topt)/(298 r topt))) ha + hd) / ((-1 + E^(hd (273+tair-topt)/(r (273+tair) topt))) ha + hd)]
```

Out[310]//FortranForm=
```
      jmax.eq.(E**((ha*(-25 + tair))/(298.*r*(273 + tair)))*
       ((-1 + E**(-(hd*(-298 + topt))/(298.*r*topt)))*ha + hd))/
     ((-1 + E**((hd*(273 + tair - topt))/(r*(273 + tair)*topt)))*ha + hd)
```

Function to determine jact:

From wetdrystanbigleaf15.nb, we use for the electron transport rate jact:

In[311]:=
```
FortranForm[jact == (1 - E^(-(alpha par)/jmax)) pc jmax]
```

Out[311]//FortranForm=
```
jact.eq.(1 - E**(-(alpha*par)/jmax))*jmax*pc
```

where alpha=0.3, pc is the projected foliage cover = 1 and jmax is expressed in mol/m2 projected cover.

Function to determine rl:

In[312]:=
```
rl == ((ca - gammastar) jmax pc rlratio)/(4 (ca + 2 gammastar) (1 + rlratio))
```

Out[312]=
```
rl == ((ca - gammastar) jmax pc rlratio)/(4 (ca + 2 gammastar) (1 + rlratio))
```

In[313]:=
```
FortranForm[%]
```

Out[313]//FortranForm=
```
rl.eq.((ca - gammastar)*jmax*pc*rlratio)/(4.*(ca + 2*gammastar)*(1 + rlratio))
```

Calculate stomatal conductance (gstom) or et or ass as a function of lambda (after stanlambda2.nb)

```
glambda == (a (ca (j - 4 rl) - 4 gammastar (j + 2 rl)) vd (ca lambda + 2 gammastar lambda - a vd) +
    Sqrt[3] Sqrt[(a gammastar j (ca (j - 4 rl) - gammastar (j + 8 rl)) vd
        (ca lambda + 2 gammastar lambda - 2 a vd)^2 (ca lambda + 2 gammastar lambda - a vd))]) /
  (4 a (ca + 2 gammastar)^2 vd (ca lambda + 2 gammastar lambda - a vd))
```

In[314]:=
```
glambda ==
 FortranForm[
  (a (ca (j - 4.0 rl) - 4.0 gammastar (j + 2.0 rl)) vd (ca lambda + 2.0 gammastar lambda - a vd) +
     Sqrt[3.0] Sqrt[(a gammastar j (ca (j - 4.0 rl) - gammastar (j + 8.0 rl)) vd
         (ca lambda + 2.0 gammastar lambda - 2.0 a vd)^2.0 (ca lambda + 2.0 gammastar lambda - a vd))]) /
   (4.0 a (ca + 2.0 gammastar)^2 vd (ca lambda + 2.0 gammastar lambda - a vd))]
```

Out[314]=
```
glambda == (0.25 *
    (a * (ca * (j - 4. * rl) - 4. * gammastar * (j + 2. * rl)) * vd * (ca * lambda + 2. * gammastar * lambda - a * vd) +
      1.7320508075688772 * Sqrt (a * gammastar * j * (ca * (j - 4. * rl) - gammastar * (j + 8. * rl)) *
         vd * (ca * lambda + 2. * gammastar * lambda - 2. * a * vd) ** 2. *
         (ca * lambda + 2. * gammastar * lambda - a * vd)))) /
  (a * (ca + 2. * gammastar) ** 2 * vd * (ca * lambda + 2. * gammastar * lambda - a * vd))
```

glambda only has a solution if $\texttt{lambda} > \frac{2\,a\,vd}{ca+2\,gammastar}$ and $\texttt{j} > \frac{4\,ca\,rl+8\,gammastar\,rl}{ca-gammastar}$

In[315]:=
```
FortranForm[lambda > (2 a vd)/(ca + 2.0 gammastar)]
```

Out[315]//FortranForm=
```
lambda.gt.(2*a*vd)/(ca + 2.*gammastar)
```

In[316]:=
```
FortranForm[j > (4 ca rl + 8 gammastar rl)/(ca - gammastar)]
```

Out[316]//FortranForm=
```
j.gt.(4*ca*rl + 8*gammastar*rl)/(ca - gammastar)
```

In[317]:=
```
lambda.gt.(2 a*vd)/(ca + 2*gammastar)
```

Out[317]=
$$\frac{\texttt{lambda.gt.(2 a vd)}}{\texttt{ca + 2 gammastar}}$$

Function to determine ass:

In[318]:=
$$\texttt{ass} == \frac{1}{8}\left(4\,\texttt{ca gstom} + 8\,\texttt{gammastar gstom} + j - 4\,\texttt{rl} - \sqrt{\left((-4\,\texttt{ca gstom} + 8\,\texttt{gammastar gstom} + j - 4\,\texttt{rl})^2 + 16\,\texttt{gammastar gstom}\,(8\,\texttt{ca gstom} + j + 8\,\texttt{rl})\right)}\right);$$

In[319]:=
$$\texttt{FortranForm}\left[\frac{1}{8}\left(4\,\texttt{ca gstom} + 8\,\texttt{gammastar gstom} + \texttt{jact} - 4\,\texttt{rl} - \sqrt{\left(\texttt{(-4 ca gstom + 8 gammastar gstom + jact - 4 rl)}^2 + 16\,\texttt{gammastar gstom}\,\texttt{(8 ca gstom + jact + 8 rl)}\right)}\right)\right]$$

Out[319]//FortranForm=
```
      (4*ca*gstom + 8*gammastar*gstom + jact - 4*rl -
      Sqrt((-4*ca*gstom + 8*gammastar*gstom + jact - 4*rl)**2 +
        16*gammastar*gstom*(8*ca*gstom + jact + 8*rl)))/8.
```

Changes to Shuffled Complex algorithm

Neil Viney used the Weibull distribution to randomise the seed values in sce.f90.

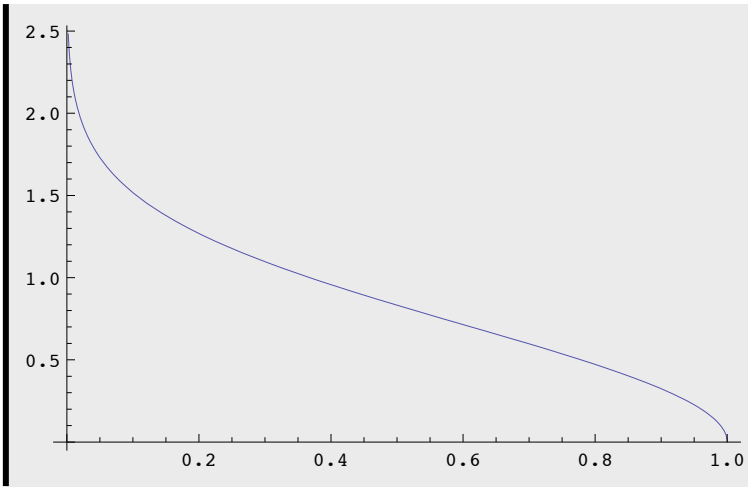
From Neil's email on 2/12/2004: "Random numbers with a Weibull distribution with mean b and shape

parameter c can be generated by: W(b,c) = b*(-log(R))**(1/c)

where R is a uniform rectangular random variate (i.e, a random number between 0 and 1). I chose the Weibull distribution because, for c>1, it gives the shape of the pdf that I'd expect. It is superficially similar to a normal distribution (with most of the weight near the mean), but it has a fixed lower bound of zero (the normal distribution has no lower bound). Having most of the weight of the pdf near the mean means that for each parameter set, most of the parameters will be reasonably close to the seed values. It is best to have only a few parameters diverging wildly from the seed, because having too many can often lead to model crashes which need to be re-run."

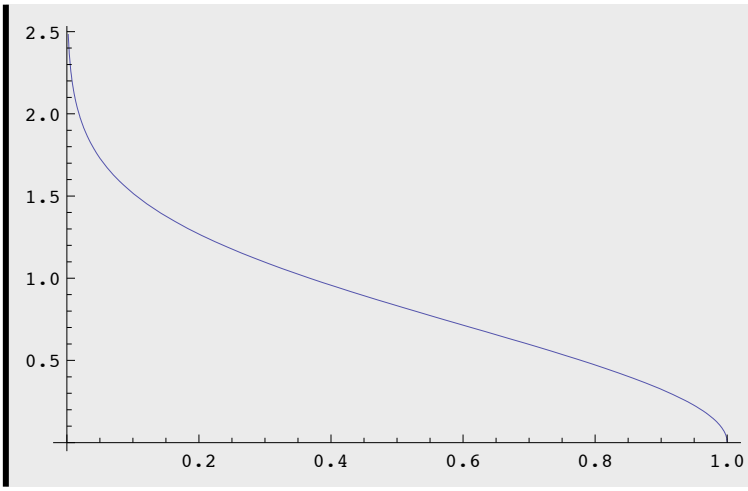
In[320]:=
```
b = 1;
weibul1 = Plot[b * (-Log[r]) ^ (1 / 2), {r, 0, 1}]
Clear[b]
```

Out[321]=

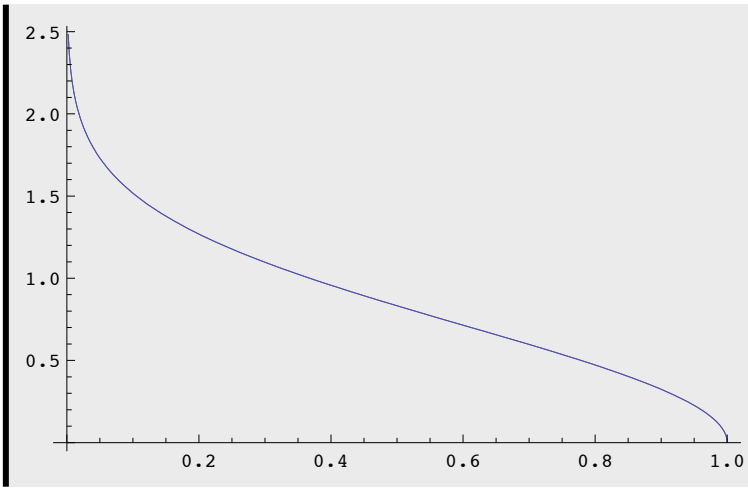


In[323]:=
```
b = 1;
weibul2 = Plot[b * (-Log[r * 0.99999 + 0.000005]) ^ (1 / 2), {r, 0, 1}]
Show[{weibul1, weibul2}]
Clear[b]
```

Out[324]=



Out[325]=



The way he coded it, leads to the problem that seed values of 0 do not get perturbed and that the sign of the values can never change, even if both signs were in the feasible range. To change this, I will use the logistic distribution, whose cdf is much simpler than the cdf of the normal distribution.

The cumulative distribution function of the logistic distribution is:

In[327]:=
```
Solve[cdf == 1 / (1 + Exp[-(x - a) / b]), x]
```

Out[327]=
$$\left\{\left\{x \to a - b \operatorname{Log}\left[-1 + \frac{1}{cdf}\right]\right\}\right\}$$

where a is any constant and represents the expected value, b>0, influencing the varation: Var[X]=(Pi^2*b^2/3), and x can take on values between -∞ and +∞.

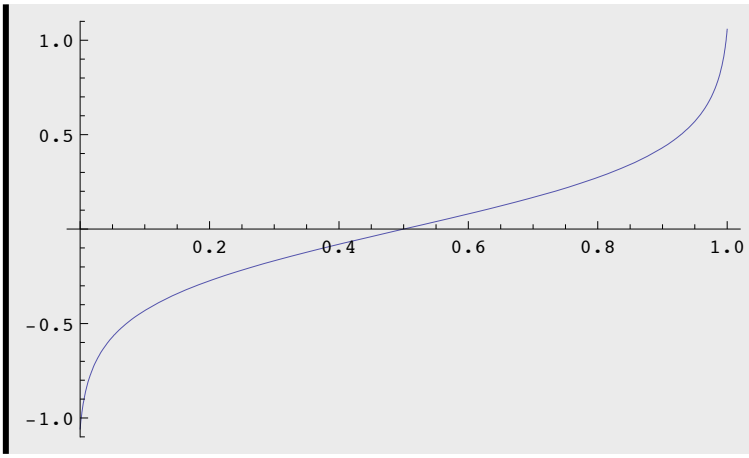
```
Solve::ifun : Inverse functions are being used by Solve, so some
   solutions may not be found; use Reduce for complete solution information. More…
```

$$\left\{\left\{x \to a - b \operatorname{Log}\left[-1 + \frac{1}{cdf}\right]\right\}\right\}$$

The random number generator in Fortran generates random numbers from a uniform distribution between 0 and 1. These can be inserted instead of cdf to obtain a factor of perturbation:

In[328]:=
```
a = 0;
b = 0.2;
Plot[a - b Log[-1 + 1/(cdf * 0.99 + 0.005)], {cdf, 0, 1}]
Clear[a, b]
```

Out[330]=



Inserting a=0 and cdf=0 or cdf=1 we can get a value for be to make our result be between 1 and -1:

In[332]:=
```
Solve[-b Log[-1 + 1/(1 * 0.99 + 0.005)] == 1, b]
Solve[-b Log[-1 + 1/(0 * 0.99 + 0.005)] == -1, b]
variable == -b Log[-1 + 1/(ranarr * 0.99 + 0.005)] /. %[[1]]
N[-b Log[-1 + 1/(1 * 0.99 + 0.005)] /. %%[[1]]]
Plot[-b Log[-1 + 1/(cdf * 0.99 + 0.005)] /. %%%, {cdf, 0, 1}, PlotRange → All]
```

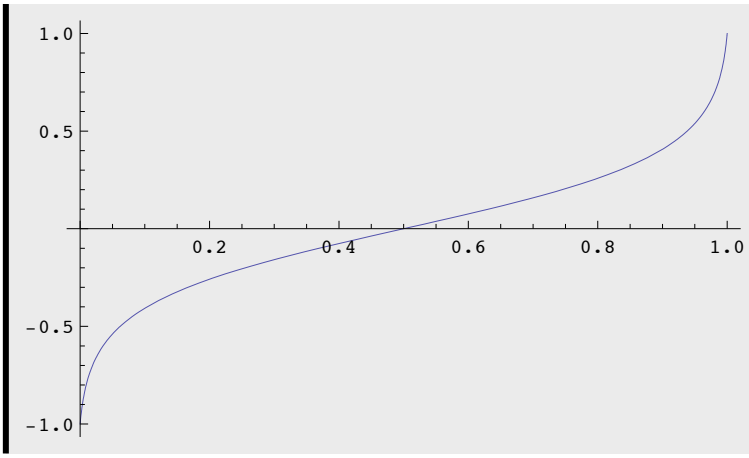
Out[332]=
```
{{b → 0.188918}}
```

Out[333]=
```
{{b → 0.188918}}
```

Out[334]=
```
variable == -0.188918 Log[-1 + 1/(0.005 + 0.99 ranarr)]
```

Out[335]=
```
1.
```

Out[336]=



Now, to perturb an original value (parval) lying within its feasible range of parmin<parval<parmax, I have to make sure that a positive outcome gets multiplied with the space between the origval and its upperbound, while a negative outcome gets multiplied with the space between teh origval and its lowerbound.:

In[337]:=
```
parval +
 (parval - parmin)
  (((-0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) -
     Abs[(-0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)])])/2 +
 (parmax - parval)
  (((-0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) +
     Abs[(-0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)])])/2
FortranForm[%]
```

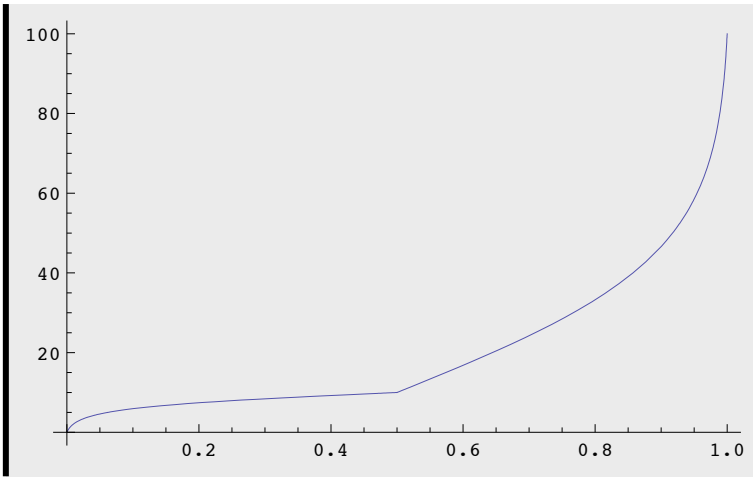
Out[337]=
$$\text{parval} + \frac{1}{2}(-\text{parmin} + \text{parval})\left(-0.188918\,\text{Abs}\left[\text{Log}\left[-1 + \frac{1}{0.005 + 0.99\,\text{ranarr}}\right]\right] - 0.188918\,\text{Log}\left[-1 + \frac{1}{0.005 + 0.99\,\text{ranarr}}\right]\right) + \frac{1}{2}(\text{parmax} - \text{parval})\left(0.188918\,\text{Abs}\left[\text{Log}\left[-1 + \frac{1}{0.005 + 0.99\,\text{ranarr}}\right]\right] - 0.188918\,\text{Log}\left[-1 + \frac{1}{0.005 + 0.99\,\text{ranarr}}\right]\right)$$

Out[338]//FortranForm=
```
      parval + ((-parmin + parval)*
       (-0.18891789404024137*Abs(Log(-1 + 1/(0.005 + 0.99*ranarr))) -
         0.18891789404024137*Log(-1 + 1/(0.005 + 0.99*ranarr))))/2. +
     ((parmax - parval)*(0.18891789404024137*Abs(Log(-1 + 1/(0.005 + 0.99*ranarr))) -
         0.18891789404024137*Log(-1 + 1/(0.005 + 0.99*ranarr))))/2.
```

In[339]:=

```
parval = 10;
parmin = 0;
parmax = 100;
Plot[
 parval +
  1/2 (-parmin + parval)
   (-0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
      0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) +
  1/2 (parmax - parval)
   (0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
      0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]), {ranarr, 0, 1}, PlotRange → All,
 AxesOrigin → {0, 0}]
result ==
 N[
  parval + 1/2 (-parmin + parval)
     (-0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
       0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) +
    1/2 (parmax - parval) (0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
       0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) /. ranarr → 0]
reduction ==
 N[
  (-parmin + parval)
    (-0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
       0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) / 2 /. ranarr → 0]
increase ==
 N[
  (parmax - parval)
    (0.18891789404024137` Abs[Log[-1 + 1/(0.005` + 0.99` ranarr)]] -
       0.18891789404024137` Log[-1 + 1/(0.005` + 0.99` ranarr)]) / 2 /. ranarr → 0]

Clear[parval, parmin, parmax]
```

Out[342]=



Out[343]=
```
result == 0.
```

Out[344]=
```
reduction == -10.
```

Out[345]=
```
increase == 0.
```

Alternatively, we could just get a totally random seed by writing:

In[347]:=
```
result = parmin + ranarr (parmax - parmin);
```