

```
' 9-10-99
' Basic code of Moore's Transfer functions for correcting eddy fluxes
' CJ Moore. 1986. Boundary Layer Meteorology.37, 17-35
```

```
' ESPM 298, Advanced Topics in Biomet and Micromet
'Dennis Baldocchi
```

```
DECLARE SUB TRANSFER ()
DECLARE SUB SPECTRA ()
```

```
'
COMMON SHARED z, u, ZL, D1, D2, D3, PQ, PCO, PCC, PW, D5
COMMON SHARED PU, XT, XU, XCO, XCC, DC, NS, S1, S2, S3, S4, S5, S6
COMMON SHARED XQ, US, QS, CCS, XCOS, DCO2, XL, UL, RAD, CCD
COMMON SHARED S7, S8, S9, S10, S11, S12, C, n, LF, B, cr
COMMON SHARED X1, X2, CU, CW, X3, X4, X5, Y1, Y2, Y3
COMMON SHARED Y4, Y5, GV, VC, w, NU, WU, W2, GD, tr, qp, ur
COMMON SHARED up, WP, TS, ga, gx, WT, UW, TT, WW, UU
COMMON SHARED G1, G2, G3, G4, G5, G6, G7, G8, G9, G10, G11, G12
```

```
'OPEN "d:\microm~1\transfer.wbw" FOR OUTPUT AS #1
'OPEN "d:\microm~1\transfer.osu" FOR OUTPUT AS #1
OPEN "d:\microm~1\transfer.bor" FOR OUTPUT AS #1
```

```
' *****
```

```
' TRANSFNC
```

```
' 7-3-98 (DDB) Transfer functions for flux covariance
' computations. Program adapted from Moore (1986) BLM 37: 17-35
```

```
' This program corrects for signal attenuation due to a finite
' path of the sonics, hygrometer and CO2 irgas. It corrects for dynamic
' frequency response limitations in the T' sensor and for a finite
' residence time in the cell of the closed path irga. It corrects
' for the effects of sensor separation between w and x. Signal
' attenuation effects due to digital high pass filtering, aliasing
' are also considered.
```

```
' *****
```

```
' Z, height above zero plane, s
' U, wind speed, m/s
' ZL, Monin Obukhov stability parameter
' D1, thermometer time constant, s
' D2, sonic anemom time constant, s.
' (note in Moore's program this variable is
' distance constant (m) of a propellor
' anemometer).
' D3, time constant of low pass electronic filter, s
' D4, time constant of ref. volume of p'2 sensor
' D5, residence time in closed path irga, s
' PQ, path length of q sensor, m
' PCO, path length of open CO2 sensor, m
' PCC, path length of closed CO2 sensor, m
' PW, path length of sonic anemo w, m
```

```

'
' PU, path length of sonic anemo u, m
' XT, separation distance of w and T, m
' XQ, separation distance of w and Q, m
' XU, separation distance of w and u sensors, m
' XCO, separation distance of w and open CO2 sensor
' XCC, separation distance of w and closed CO2 sensor
' DC, time constant of digital high pass filter, s
' NS, sampling frequency, HZ
' DCO2, diffusivity of CO2
' XL, length of the irga sampling tube
' UL, flow velocity thru tube
' RAD, radius of tube
'

```

covariance	transfer function	correction factor
wt	G1	S1
wq	G2	S2
uw	G3	S3
tt	G4	S4
qq	G5	S5
uu	G6	S6
ww	G7	S7
wc open	G8	S8
cc open	G9	S9
wc closed	G10	S10
cc closed	G11	S11
p'2	G12	S12

```

'*****
'

```

```

' set up measurement parameters
'

```

```

' sampling rate
'
' 10.0 hz WBW, BOREAS, OSU
' Forest tower
'
'
NS = 10!
'
d = .75 * 26
z = 2 'Z = 36 - d 'tower
u = 1.5 'U = 4
ZL = .001
D1 = .025 'thermometer time constant
D2 = .025 'sonic time constant: over sampling is done
D3 = 1 / NS
D4 = .05
D5 = .56 ' 1/e residence time in spring is 0.56 s and .71 in fall
PQ = .15 'path size of water vapor sensor
PW = .15 'path of w
PU = .15 'path of u
PCO = .15 'path of open co2
PCC = .15 'path of closed co2
XT = .01
XQ = .4 'IRGA is 40 cm from sonic
XCO = .4
XCC = .5

```

```

XU = .1
DC = 400
'
'fall setup on IRGA Sampling through a tube
'
DCO2 = 14.7 ' mm^2/s
XL = 3400 ' mm      2500mm in spring, 3400mm in fall
UL = 1130 ' mm/s    1430mm/s in spring, 1130 mm/s in fall
RAD = 2.25 ' mm
'
LPRINT ""
LPRINT " Z-d(m) U (m/s) ZL samp. freq (Hz) digital filter"
LPRINT USING " ##.# ##.### ##.### ##.### #####.##"; z;
u; ZL; NS; DC

WRITE #1, " Z-d(m) U (m/s) ZL samp. freq (Hz) digital filter"
WRITE #1, z, u, ZL, NS, DC

LPRINT ""
LPRINT " time constants, s"
LPRINT " thermistor sonic anemo"
LPRINT USING " ##.##### ##.#####"; D1; D2
LPRINT ""
LPRINT " averaging path length of sensor, m"
LPRINT " sonic anemo hygrometer CO2 open CO2 closed"
LPRINT USING " #.## #.### #.### #.###"; PU; PQ;
PCO; PCC

LPRINT ""
LPRINT " separation between w and scalar sensor"
LPRINT " U T Q CO2 open CO2 closed"
LPRINT USING " #.### #.### #.### #.### #.###"; XU; XT; XQ;
XCO; XCC

LPRINT ""
LPRINT " wt wq wu tt qq uu ww wco cco
wcc ccc"
LPRINT " N G1 G2 G3 G4 G5 G6 G7 G8 G9
G10 G11"
IMAGE$ = "##.##### #.### #.### #.### #.### #.### #.### #.### #.###
#.### #.### #.### #.### #.###"

WRITE #1, " wt, wq, wu, tt, qq, uu, ww, wco, cco, wcc, ccc"
WRITE #1, " N, G1, G2, G3, G4, G5, G6, G7, G8,
G9, G10, G11"
'
'call SPECTRA, this subroutine computes the coefficient
' for use in the formulae of Kaimal
'
CALL SPECTRA
'
'integration of spectra with Simpson's rule.
'The logarithmic spectrum is integrated between
'the log frequency range of -5 to log(nyquist freq)
'over 19 equal intervals.
'
I3 = 0

```

```

'
FOR I1 = 1 TO 10
FOR I2 = 2 TO 4 STEP 2
I3 = I3 + 1
'
IF I3 > 19 THEN GOTO 500
'
I4 = I2 + (I3 = 1) + (I3 = 19)
'
' Call subroutine transfer.  Convolutes the transfer
' functions onto the spectral formulae
'
CALL TRANSFER
'
' It will be assumed the nSwt = nSwq = nSwc and
' nStt = nSqq = nScc, which
' is valid according to measurements by Anderson et al.
' over oak.
'
' integration with Simpson's rule
'
' Sn = (h/6)[f0+fn+2 sum(fi)+4sum(fi-.5)]
'
S1 = S1 + I4 * G1 * WT
S2 = S2 + I4 * G2 * WT
S3 = S3 + I4 * G3 * UW
S4 = S4 + I4 * G4 * TT
S5 = S5 + I4 * G5 * TT
S6 = S6 + I4 * G6 * UU
S7 = S7 + I4 * G7 * WW
S8 = S8 + I4 * G8 * WT
S9 = S9 + I4 * G9 * TT
S10 = S10 + I4 * G10 * WT
S11 = S11 + I4 * G11 * TT
'
S12 = S12 + I4 * G12 * PP
'
S13 = S13 + I4 * gx * TT
'
LPRINT USING IMAGE$; n; G1; G2; G3; G4; G5; G6; G7; G8; G9; G10; G11

WRITE #1, n, G1, G2, G3, G4, G5, G6, G7, G8, G9, G10, G11

'
n = n * LF
'
NEXT I2
NEXT I1
'
'compute transfer functions
'
500 S1 = C / S1
S2 = C / S2
S3 = C / S3
S4 = C / S4
S5 = C / S5

```

```

S6 = C / S6
S7 = C / S7
S8 = C / S8
S9 = C / S9
S10 = C / S10
S11 = C / S11
S13 = C / S13

'
LPRINT ""
LPRINT "          SPECTRAL AND CO-SPECTRAL CORRECTION FACTORS"
LPRINT "      WT          WQ          UW          TT          QQ          UU          WW          WCO          CCO
WCC      CCC"
LPRINT USING " #.### #.### #.### #.### #.### #.### #.### #.### #.###
#.### #.###"; S1; S2; S3; S4; S5; S6; S7; S8; S9; S10; S11
LPRINT ""
END

SUB SPECTRA STATIC
'
' computations for temperature sensor time constant
'
TC = D1 / (1 + 4.9 * (SQR(D1) * U) ^ .45)
'
      TC = D1
'
TC = TC * TC
'
' compute time constant of sonic anemometer
'
UC = D2
UC = UC * UC
'
'square time constant of electronic RC filter
'
VC = D3 * D3
'
C = 4.115
'
' initialize correction factors, S
'
S1 = 0
S2 = 0
S3 = 0
S4 = 0
S5 = 0
S6 = 0
S7 = 0
S8 = 0
S9 = 0
S10 = 0
S11 = 0
S12 = 0
S13 = 0
S14 = 0
S15 = 0
S16 = 0

```

```

'
' lowest frequency, log(N) = -5
'
n = .00001
'
LF = 2.073
'
' coefficient power, 5/3
'
B = 1.667
'
IF ZL >= 0 THEN GOTO 10
'
' compute coefficients for spectra under unstable conditions
'
nSwv(n) = (f/(1+5.3 f^1.667) + 16 f x1/(1+17f)^1.667) CW^-1
'
nSuv(n) = (210 f/(1+33f)^1.667) + f x1/(x2+ 2.2 f^1.667) CU^-1
'
X1 = (-ZL) ^ .667
'
' (z/zi)^(5/3)
'
X2 = (.001 * z) ^ B
'
CW = .7285 + 1.4115 * X1
CU = 9.546 + 1.235 * X1 / (X2 ^ .4)
'
GOTO 20
'
10 ' compute spectral coefficients from Kaimal for stable conditions
'
' normalized cospectrum form for heat and momentum transfer
'
nSwx(n) = f/(Awx + Bwx f^(2.1))
'
Awt
'
X1 = .285 * (1 + 6.4 * ZL) ^ .75
'
Auw
'
X2 = .124 * (1 + 7.9 * ZL) ^ .75
'
' Sxx for x=T,w and u
'
nS(n) = f/(Ax + Bx f^(5/3))
'
' Coefficients for stable conditions
'
At
'
X3 = .0961 + .644 * ZL ^ .6
'
Aw

```

```

'
'      X4 = .838 + 1.172 * ZL
'
' Au
'
'      X5 = .2 * X4
'
' Bwx for x= T and u
'
'      Y1 = 2.34 / X1 ^ 1.1
'      Y2 = 2.34 / X2 ^ 1.1
'
' Bx for x= T,w and u
'
'      Y3 = 3.124 / X3 ^ .667
'      Y4 = 3.124 / X4 ^ .667
'      Y5 = 3.124 / X5 ^ .667
'
20 END SUB

SUB TRANSFER STATIC
'
'      constants
'
'      omega, 2 pi n
'
'      w = 6.283 * n
'
'      wavenumber, n/u
'
'      NU = n / u
'
'      2 pi n/u
'
'      WU = w / u
'
'      omega squared, (2 pi n)^2
'
'      W2 = w * w
'
'      data acquisition sampling transfer function
'
'      T(n) = 0 if n > ns/2
'
'      T(n) = 1 +(n/(ns-n))^b if n <= ns/2
'
'      IF n >= NS / 2 THEN
'      G1 = 0
'      G2 = 0
'      G3 = 0
'      G4 = 0
'      G5 = 0
'      G6 = 0
'      G7 = 0
'      G8 = 0
'      G9 = 0
'      G10 = 0

```

```

G11 = 0
G12 = 0
'
'goto end of sub
'
GOTO 250
END IF
'
' Tranfer function for sampling and accounting for
' the effects of aliasing.
'
' if n < ns/2
'
'   Ga = 1 +(n/(n-ns))^3
'
X = n / (NS - n)
ga = 1 + X * X * X
'
'
' gain function for the digital filter low pass filter
'
'   Gd(n) = 2 pi n tau/(1 + (2 pi n tau)^2/alpha)^.5
'
GD = 1
'
' Gd=1 for 1/2 pi n > tau, ie DC
'
X = w * DC
ALPHA = EXP(-1 / (NS * DC))
'
'test if 1/n < tau and apply filter to higher
' freq. contributions
'
IF X < 6.283 THEN
DX = X * X
GD = X / ((1 + DX) / ALPHA) ^ .5
END IF
'
'NO DC FILTER APPLIED
'
GD = 1
'
' Electronic filter transform
'
'   (1 + (2 pi n tau)^2)^.5
'
'GV = W2 * VC
'GV = 1 + GV * GV
'
' no filter is applied
'
GV = 1
'
' Cospectral Transfer function of data acquisition system
'
'   Ga is the sampling rate and aliasing transform
'   Gd is the digital high pass filter transform,

```

```

'      as applied for mean removal.
'      Gv is the electronic filter transfor
'
gx = ga * GD * GD * GV * GV
'
' first order gain for dynamic frequency
' response of temperature probe
'
'      G(n) = TR(n)^-.5
'
'      TR(n) = 1 + (2 pi n tau)^2
'
tr = (1 + W2 * TC) ^ -.5
'
' first order gain for dynamic frequency
' response of residence time in closed path irga
'
'      G(n) = TR(n)^-.5
'
'      TR(n) = 1 + (2 pi n tau)^2
'
cr = (1 + W2 * D5 * D5) ^ -.5
'
'First order gain for dynamic frequency response of
'anemometer
'
ur = (1 + W2 * UC) ^ -.5
'
' Transform function for line averaging scalars
'
'      Tp(n) = 3+exp(-x) - 4(1-exp(-x)/x)
'
'      x =2 pi f, f = n p/u
'
' for the hygrometer
'
qp = 1
X = WU * PQ
IF X > .02 THEN
Y = EXP(-X)
qp = 3 + Y - 4 * (1 - Y) / X
qp = qp / X
END IF
'
' for the u component of the sonic
'
up = 1
X = WU * PU
IF X > .02 THEN
Y = EXP(-X)
up = 3 + Y - 4 * (1 - Y) / X
up = up / X
END IF
'
' for the w component of the sonic
'
WP = 1

```

```

X = WU * PW
IF X > .04 THEN
Y = EXP(-X)
WP = 1 + (Y - 3 * (1 - Y) / X) / 2
WP = 4 * WP / X
END IF
'
'for the open path CO2 sensor
'
cop = 1
X = WU * PCO
IF X > .02 THEN
Y = EXP(-X)
cop = 3 + Y - 4 * (1 - Y) / X
cop = cop / X
END IF
'
'for the closed path CO2 sensor
'
ccp = 1
X = WU * PCC
IF X > .02 THEN
Y = EXP(-X)
ccp = 3 + Y - 4 * (1 - Y) / X
ccp = ccp / X
END IF
'
' transfer function for lateral and longitudinal
' separation between a sensor and w.
'
' for the temperature sensor
'
TS = 1
X = NU * XT
IF X > .01 THEN
TS = EXP(-9.9 * X ^ 1.5)
END IF
'
' for the humidity sensor
'
QS = 1
X = NU * XQ
IF X > .01 THEN
QS = EXP(-9.9 * X ^ 1.5)
END IF
'
' for the open path CO2 sensor
'
XCOS = 1
X = NU * XCO
IF X > .01 THEN
XCOS = EXP(-9.9 * X ^ 1.5)
END IF
'
' for the closed path CO2 sensor
'
CCS = 1

```

```

X = NU * XCC
IF X > .01 THEN
CCS = EXP(-9.9 * X ^ 1.5)
END IF
'
' for the u component
'
US = 1
X = NU * XU
IF X > .01 THEN
US = EXP(-9.9 * X ^ 1.5)
END IF
'
'
' Compute fluctuation dampening via sampling thru a tube
'
CCD = EXP(-.021 * w * w * RAD * RAD * XL / (DCO2 * UL))
'
'
' compute the gains
'
G1 = gx * TS * SQR(WP) * tr
G2 = gx * QS * SQR(WP * qp)
G3 = gx * US * SQR(WP * up) * ur
G4 = gx * tr * tr
G5 = gx * qp
G6 = gx * up * ur * ur
G7 = gx * WP * ur * ur
G8 = gx * SQR(WP * cop) * XCOS
G9 = gx * cop
G10 = gx * SQR(WP * ccp * CCD * cr) * CCS
G11 = gx * ccp * CCD * cr
G12 = gx
'
' compute spectral estimates at each discrete frequency, f
'
F = NU * z
F1 = F ^ B
IF ZL <= 0 THEN
'
' if unstable conditions goto 710
'
710 ' nSww(n) = (f/(1+5.3 f^1.667) + 16 f x1/(1+17f)^1.667) CW^-1
'
X1 = (-ZL) ^ .667
X2 = (.001 * z) ^ B
WW = 16 * F * X1 / (1 + 17 * F) ^ B
WW = WW + F / (1 + 5.3 * F1)
'
' nSwt(n) =
'
' 12.92f/(1+26.7 f)^1.375, f< 0.54
'
' 4.378f/(1+3.8f)^2.4, f>= 0.54
'
IF F >= .54 THEN WT = 4.378 * F / (1 + 3.8 * F) ^ 2.4
'
IF F < .54 THEN WT = 12.92 * F / (1 + 26.7 * F) ^ 1.375

```

```

'
' It will be assumed the nSwt = nSwq = nSwc, which
' is valid according to measurements by Anderson et al.
' over oak.
'
IF F >= .2 THEN
UW = 12.66 * F / (1 + 9.6 * F) ^ 2.4
'
TT = 6.827 * F / (1 + 12.5 * F) ^ B
END IF
'
IF F < .2 THEN
UW = 20.78 * F / (1 + 31 * F) ^ 1.575
'
TT = 14.94 * F / (1 + 24 * F) ^ B
'
'It will be assumed that
' nStt = nSqq = nScc, which is valid according to
' measurements by Anderson et al over oak.
'
END IF
'
nSuu(n) = (210 f/(1+33f)^1.667) + f x1/(x2+ 2.2 f^1.667) CU^-1
'
UU = 210 * F / (1 + 33 * F) ^ B + F * X1 / (X2 * 2.2 * F1)
UU = UU / CU
WW = WW / CW
'

ELSE
'
'stable conditions
'
nSwx(n) = f/(Awx + Bwx f^(2.1))
'
nSxx(n) = f/(Ax + Bx f^(5/3))
'

F2 = F ^ 2.1
WT = F / (X1 + Y1 * F2)
UW = F / (X2 + Y2 * F2)
TT = F / (X3 + Y3 * F1)
WW = F / (X4 + Y4 * F1)
UU = F / (X5 + Y5 * F1)
END IF
'
250      END SUB

```