Time-dependent modelling of the temperature profile near the sea surface

Report from the Norwegian Research Council Project 'Measurement and modelling of controlling factors for air–sea fluxes'

Alastair D. Jenkins
Norwegian Meteorological Institute
Allégaten 70, 5007 Bergen, Norway
(Now at:
Bjerknes Climate Research Centre, Geophysical Institute
Allégaten 70, 5007 Bergen, Norway
e-mail: alastair.jenkins@gfi.uib.no)

Brian Ward
NOAA/AOML/OCD
4301 Rickenbacker Causeway
Miami, 33149 Florida, USA
e-mail: ward@aoml.noaa.gov
(Now at:
Applied Ocean Physics and Engineering
Woods Hole Oceanographic Institution
Woods Hole, MA 02543
U.S.A.
e-mail: bward@whoi.edu)

Version 2.1, 2003 June 1

Abstract

This report describes a simple time-dependent numerical model for the time-dependent computation of the temperature profile in the upper layer of the ocean, and the comparison of the results from the model with measurements from the SkinDeEP surface profiling instrument made during the MOCE-5 field experiment. The model is based upon conservation laws for heat and momentum, and employs an eddy diffusion parameterisation which is dependent on the wind speed and wind stress applied at the sea surface. Other parameters such as the bulk−skin surface temperature difference and CO_2 flux are determined by application of the Molecular Oceanic Boundary LAyer Model (MOBLAM) of Schlüssel and Soloviev.

Given that no tuning was performed the simple time-dependent model, and that the model does not take account of stratification, the results of the model runs are in rather good agreement with the observations. The model may be suitable as an interface between time-independent models for processes very near the surface, and larger-scale three-dimensional time-dependent ocean circulation models. A straightforward extension of the model should also be capable of making time-dependent computations of CO_2 concentration in the near-surface layer of the ocean.

Acknowledgements

The high-resolution upper-ocean temperature profiling device referred to in this report was developed and deployed by Dr. Brian Ward, whose salary was covered under a European Commission Marie Curie Fellowship, Contract No. ERBFMBICT983162. The Norwegian Research Council funded the instrument development under the current project. The observational data presented in this report are from the Marine Optical Characterisation Experiment (MOCE-5), which was conducted by the U.S. National Aeronautic and Space Administration and National Oceanic and Atmospheric Administration.

The computer code MOBLAM was made available on the Internet at address `http://www.nova.edu/ocean/gasex/Moblam8.f90` by Dr. Peter Schlüssel.

Contents

1 Introduction

The project *Measurement and modelling of controlling factors for air–sea fluxes,* funded by the Norwegian Research Council, has the following objectives:

1. The development and testing of a high-resolution profiling device for the near-surface temperature profiler, and its deployment on field cruises in Norwegian waters and abroad;

2. The application of a numerical model for the computation of the temperature profiles under the conditions prevailing during the deployment of the above instrument, for the computation of air-sea fluxes of heat, momentum and mass (e.g. gas species).

The high-resolution profiling device SkinDeEP was developed by Dr. Brian Ward under a European Commission Marie Curie Postdoctoral Fellowship, during 1999–2000. It was manufactured by Precision Measurement Engineering in the U.S.A. It is an autonomous profiler, carrying high-resolution temperature sensors to provide a record of the bulk temperature. It was deployed during the Marine Optical Characterisation Experiment (MOCE-5), which was conducted during 1999 October 1–21 in the waters around Baja California. A report on the instrument and its deployment is given in the Marie Curie Fellowship Final Report [17].

The motivation behind the development of this profiling instrument is to provide high-resolution measurements of water temperature very near the sea surface, in order to obtain good estimates of the bulk−skin temperature difference, or 'skin effect'. The *skin temperature* of the ocean or other water body is in general a few tenths of a degree cooler than the temperature of the bulk of the fluid, 0.1–1 mm lower down, primarily as a result of the cooling of the water by outgoing longwave radiation emitted in a thin layer of the order of 1μm deep. The transition zone from this skin layer to the bulk is controlled primarily by molecular heat conduction and has a thickness of the order of 1 mm [14]. Reliable estimates of the bulk−skin temperature difference are necessary in order to interpret ocean temperature measurements by satellite radiometer, and a good understanding of the processes occurring very near the sea surface is necessary in order to obtain accurate estimates of such parameters as the air–sea flux of CO_2 and other gas species.

2 Measurements

The SkinDeEP temperature profiling instrument is illustrated in Fig. 1. A detailed description of the instrument, which uses an FP07 thermistor and a high-resolution platinum wire (Pt) temperature sensor, is given by Ward and Minnett [18]. The instrument is able to rise and sink autonomously, by inflating and deflating a neoprene bladder.

The instrument was deployed on the cruise of the MOCE-5 experiment, in the waters around Baja California (see Fig. 2). Data were acquired from nine stations: Table 1 summarises the deployment information for the three stations where the results are shown in this report. The measurements reported here are from the FP07 thermistor, as data from the Pt sensor were not available for this cruise.

Table 1: SkinDeEP deployment information for the measurements presented in this report

Date	Stat. no.	Station name	Times UTC	No. of profiles	Position
1999-10-04	4	Punta Magdalena	18:18–20:42	118	25° 09.42′N, 113° 00.00′W
1999-10-10	10	T/S Irwin	18:16–21:17	161	22° 31.80′N, 109° 36.00′W
1999-10-14	14	Isla San Esteban 2	13:29–14:59	78	28° 36.24′N, 112° 29.40′W

Other observations made include the following:

- Sea-surface skin temperature by the M-AERI passive infrared radiometric interferometer, using the 500–3000 cm^{-1} wavelength range, which has an accuracy of better than 0.05° [10, 11];
- Bulk water temperature measurements from the ship intake and from a floating sensor (an inverted hard plastic helmet filled with foam with a thermistor just below the waterline);
- Air temperature, wind speed, and net heat flux.

All three measurement series presented here are for periods with rather light winds, between 0.7 and 4.5 $\mathrm{m\,s}^{-1}$. The air temperature is less than the sea surface temperature (measured by SkinDeEP) on October 4, is greater than the surface temperature on October 14, and on October 10 the air and sea surface temperatures are quite similar. The net heat flux is positive (from air to sea) in all three cases. The net heat flux is primarily incident short-wave radiation, and is mostly absorbed in the upper 0.1 m of the water column [14].

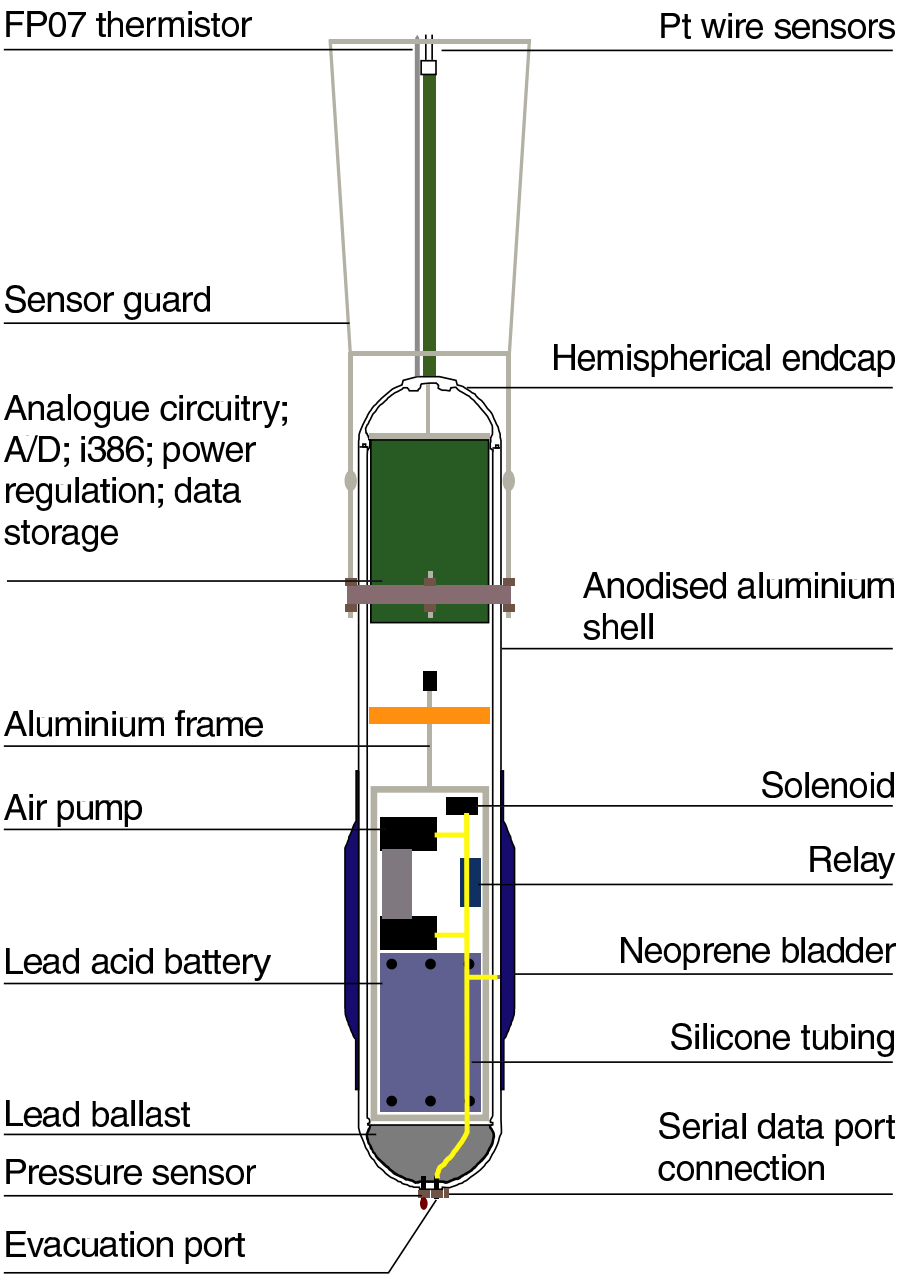

Figure 1: The SkinDeEP ocean temperature profiler, showing its main components

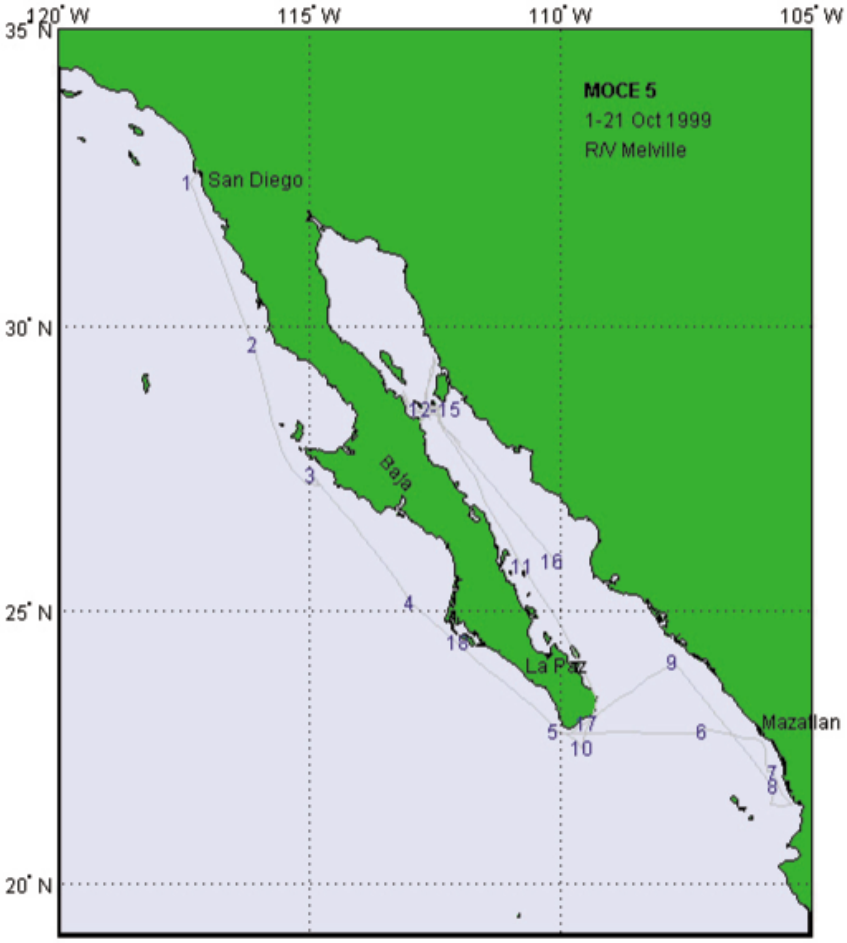

Figure 2: The MOCE-5 cruise showing the station numbers

3 Time-dependent model for temperature profile

3.1 Background

A number of models exist for computing the bulk−skin temperature difference ΔT_{B-S} and other parameters and fluxes within the upper centimetre of the water column [6, 7, 9, 16]. These models can be quite complex, since they compute many interrelated parameters, and since they consider processes with a small time scale, are generally in time-independent form. Alternatively one can use such techniques as neural network methods to derive empirically the relation between the forcing parameters and ΔT_{B-S} [17, 19].

In the upper 10–50 m layer of the water column, time-dependent three-dimensional ocean circulation models may employ simple algebraic parameterisations for turbulent fluxes of momentum, heat, and mass, or may, at greater computational expense, use some form of turbulence closure technique (e.g. [3]). Even more expensive computationally are direct numerical simulations of vortex structures (e.g. [12]).

In this report we employ a time-dependent model which should be applicable to the zone intermediate between the surface millimetre layer where the skin effect occurs and the ≈10 m level which can be resolved by oceanic circulation models. The model may thus be useful in describing or parameterising the coupling between the different model scales. It uses a simple parameterisation of turbulent flux processes, so that it should be numerically stable and computationally inexpensive.

3.2 Model formulation and specification

The basic premise underlying the model is that the combined effect of wind and wave forcing at the water surface gives rise to turbulent eddy motions which cause quantities such as heat or momentum to diffuse vertically. In addition, in the case of the diffusion of momentum, it has been observed empirically, e.g. by the observation of drifting buoys, oil slicks etc., that the current at the sea surface is approximately a constant fraction λ of the wind speed U, which we will assume here to take a value of 2 per cent.

3.2.1 Current profile and its possible effect on waves

For an ocean initially at rest, we consider at first the following evolution of the current $u(z,t)$ at time t and depth $-z$:

$$u(z,t) = \lambda U \exp\left[\lambda U z/({u_*}^2 t)\right], \tag{1}$$

where u_* is the friction velocity within the water column, $\tau = \rho_w {u_*}^2 = \rho_a {U_*}^2$ being the wind stress. U_*, ρ_a, and ρ_w are the friction velocity in the atmosphere, the air density and the water density respectively. Equation 1 satisfies the conservation of momentum:

$$\rho_w \frac{d}{dt} \int_{-\infty}^{0} u(z,t)\, dz = \rho_a {U_*}^2, \tag{2}$$

where for simplicity we neglect the Coriolis force.

The current $u(z,t)$ in (1) obeys the equation

$$\frac{\partial u}{\partial t} = -\frac{{u_*}^2 z}{\lambda U} \frac{\partial^2 u}{\partial z^2}, \tag{3}$$

with $-{u_*}^2 z/(\lambda U)$ playing the role of an eddy viscosity, similar to that of turbulent boundary layer flow near a rigid wall. If the wind speed is allowed to vary with time we can alter (1) so that it becomes

$$u(z,t) = \int_0^t \lambda \frac{dU(t')}{dt'} \exp\left[\frac{\lambda U(t')\, z}{(u_*(t'))^2\ (t-t')}\right] dt'. \tag{4}$$

In this case, the partial differential equation (3) does not hold, as the system retains a 'memory' of the wind forcing at previous points in time. The memory is effectively longer at large depths than near the surface.

Sample calculations of current profiles and their effect on waves

Figure 3 shows the current induced by a wind of 20 m/s blowing for 2 hours followed by an increased wind of 30 m/s. The current profiles are calculated every 5 minutes, and it can be seen that the current responds very rapidly to wind changes.

Wave spectra have also been calculated by using in the first instance the Donelan *et al.* [5] formulation for limited fetch, transformed by means of the wave group velocity to apply to winds blowing for limited times. Calculations have also been made of the effect of the computed vertical current shear on the wave height required for wave breaking, according to the theory of Banner and Phillips [2]. Results of these calculations indicate that a rapid increase in wind speed will tend to increase the intensity of wave breaking so that the wave height will decrease by a small factor, of order 6% according to linear wave theory, but this factor may be as much as 25% according to time-dependent nonlinear numerical simulations by Banner *et al.* [1].

The above results are published in the refereed proceedings of the conference WAVES 2001 [8].

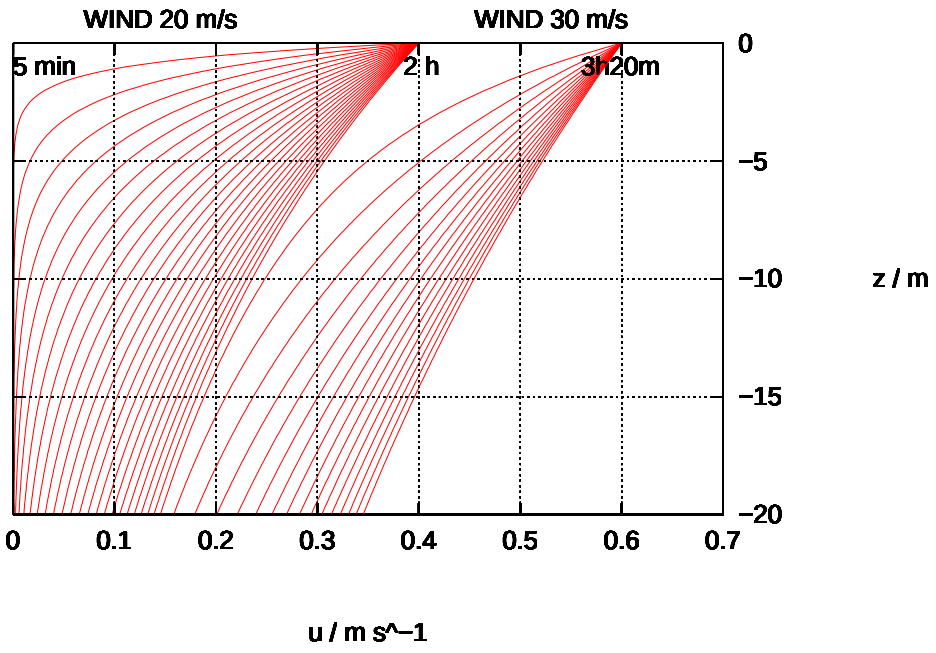

Figure 3: Example of the evolution of a current profile using the present model.

3.2.2 Heat flux

If we now consider the diffusion of heat within the water column, we obtain a corresponding formula for the evolution of the temperature T:

$$T(z,t) = T_0 + \int_0^t \lambda \frac{dT(0^-,t')}{dt'} \exp\left[\frac{\lambda U(t')\, z}{(u_*(t'))^2\ (t-t')}\right] dt', \tag{5}$$

where T_0 is the initial temperature, and $T(0^-,t)$ is the bulk temperature just below the surface skin layer. Here we have not taken into account the possibility that the turbulent diffusion coefficients for heat and momentum may be different, but we may make such an adjustment by altering the value of the parameter λ. The model is assumed to be horizontally homogeneous, and the effect of density stratification on the diffusion of heat is neglected.

3.2.3 Detailed specification of the model

The model codes and documentation are specified in Appendix A

4 Model results

Figures 4–6 show SkinDeEP measurements from three of the MOCE-5 stations, together with background atmospheric parameters and net heat flux. Results of model simulations from equation 5 are shown at the bottom of each figure. The model is started with a uniform temperature at the time corresponding to the left-hand side of the plot, and the surface boundary condition $T(0^-, t)$ is set equal to the uppermost temperature measured by the SkinDeEP profiler. The ocean is thus assumed to be horizontally homogeneous, and neither the effect of stratification nor of the Coriolis force are taken into account. The current and waves were not calculated in this case, since no observations of current or waves were available.

1999 October 4 In this situation the air temperature is lower than the sea temperature, but the net positive heat flux should tend to increase the bulk temperature near the sea surface. The observed warming of the surface layer at around 12:30 is likely to be due to horizontal advection. Nevertheless, the model does predict an increase in depth of the warm surface layer ($T > 22°\mathrm{C}$) to 2 m by the end of the observation period, which is of the same order as the depth increase which is actually observed.

1999 October 10 Here the modelled increase in the warm surface layer depth during the observation period is quite similar to that actually observed. The drop in the observed surface temperature at 19:00 may be associated with the decrease in net heat flux in the previous 15 minutes, but the associated changes in the depth of the warm surface layer are not reproduced in the model. The greater wind speed in the period around 20:00, together with the positive air−sea temperature difference, tend to increase the temperature in the upper few centimetres of the ocean, and this is reproduced in both measurements and model results. Again, the agreement between model results and measurements is good, considering the fact that the model does not take stratification into account.

1999 October 14 In this case the model appears to make a poor prediction of the temperature evolution of the surface layer of the ocean, when compared with the measurements. In this case, however, the temperature in the water column is rather uniform, varying by less than 0.15 K over the whole domain. The turbulent diffusion predicted by the model suppresses any developing non-uniformity in the temperature profile, an effect which would be mitigated if stratification were taken into account.

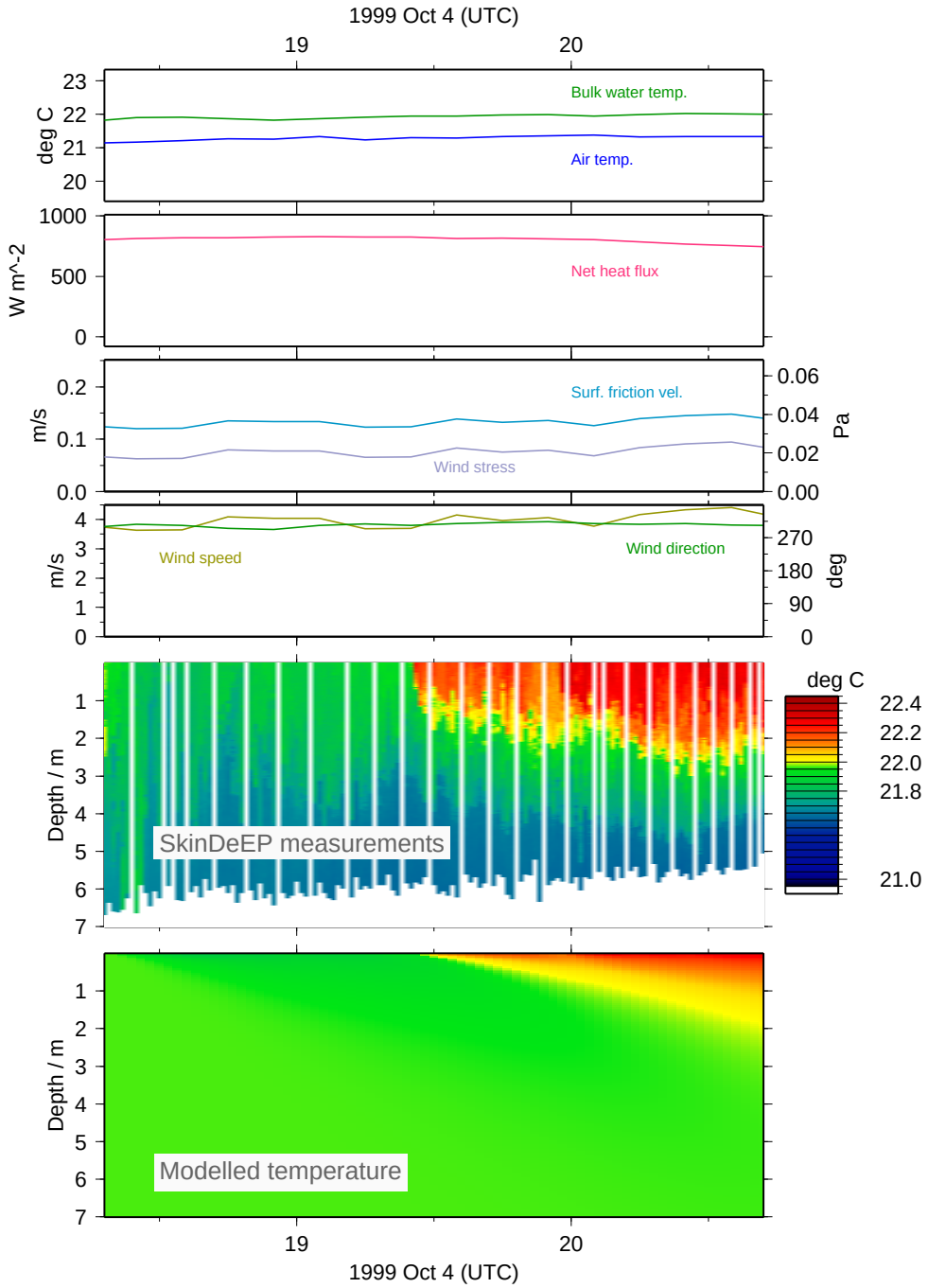

Figure 4: Observations and model results from Station Punta Magdalena, 1999 October 4.

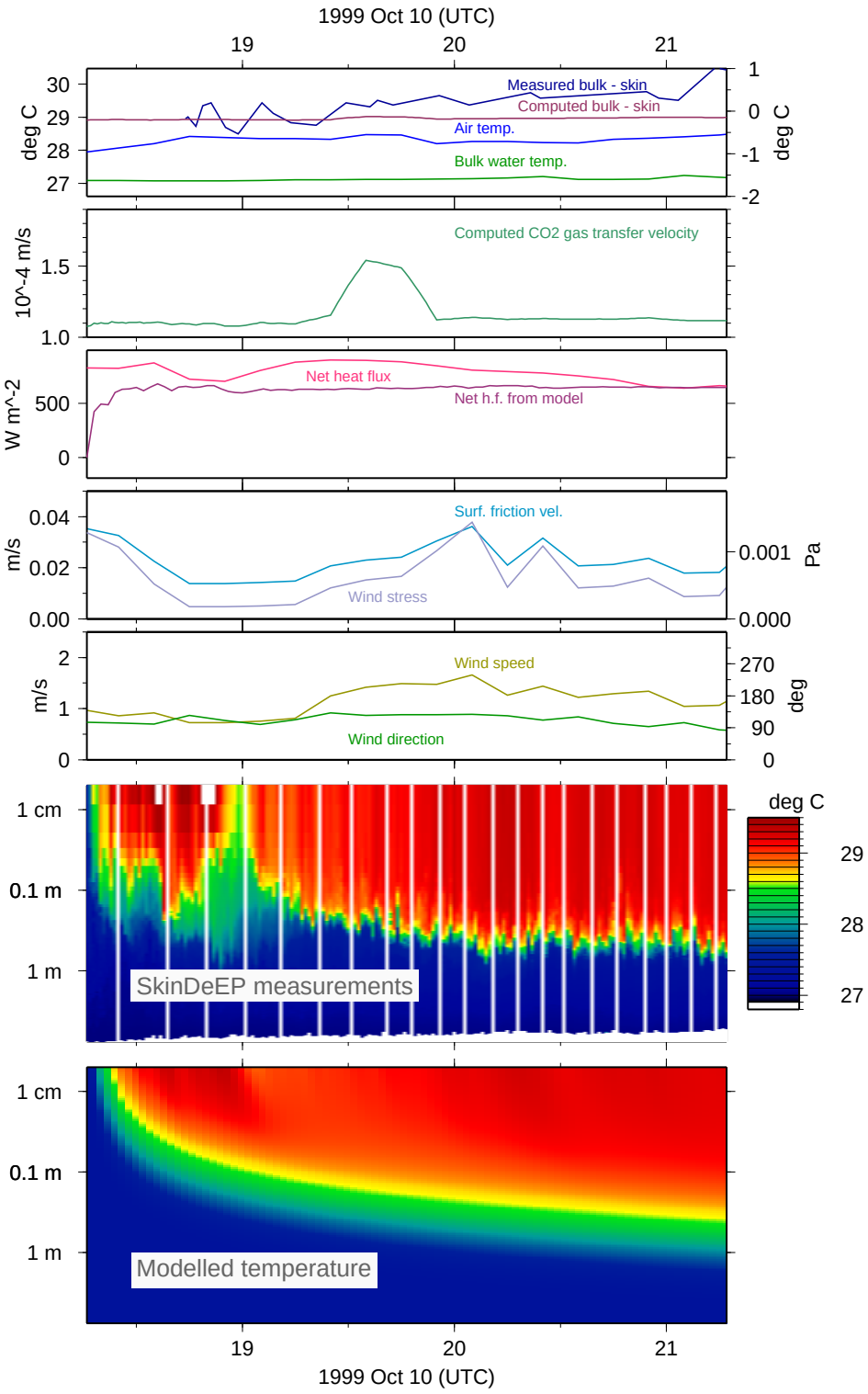

Figure 5: Observations and model results from Station T/S Irwin, 1999 October 10.

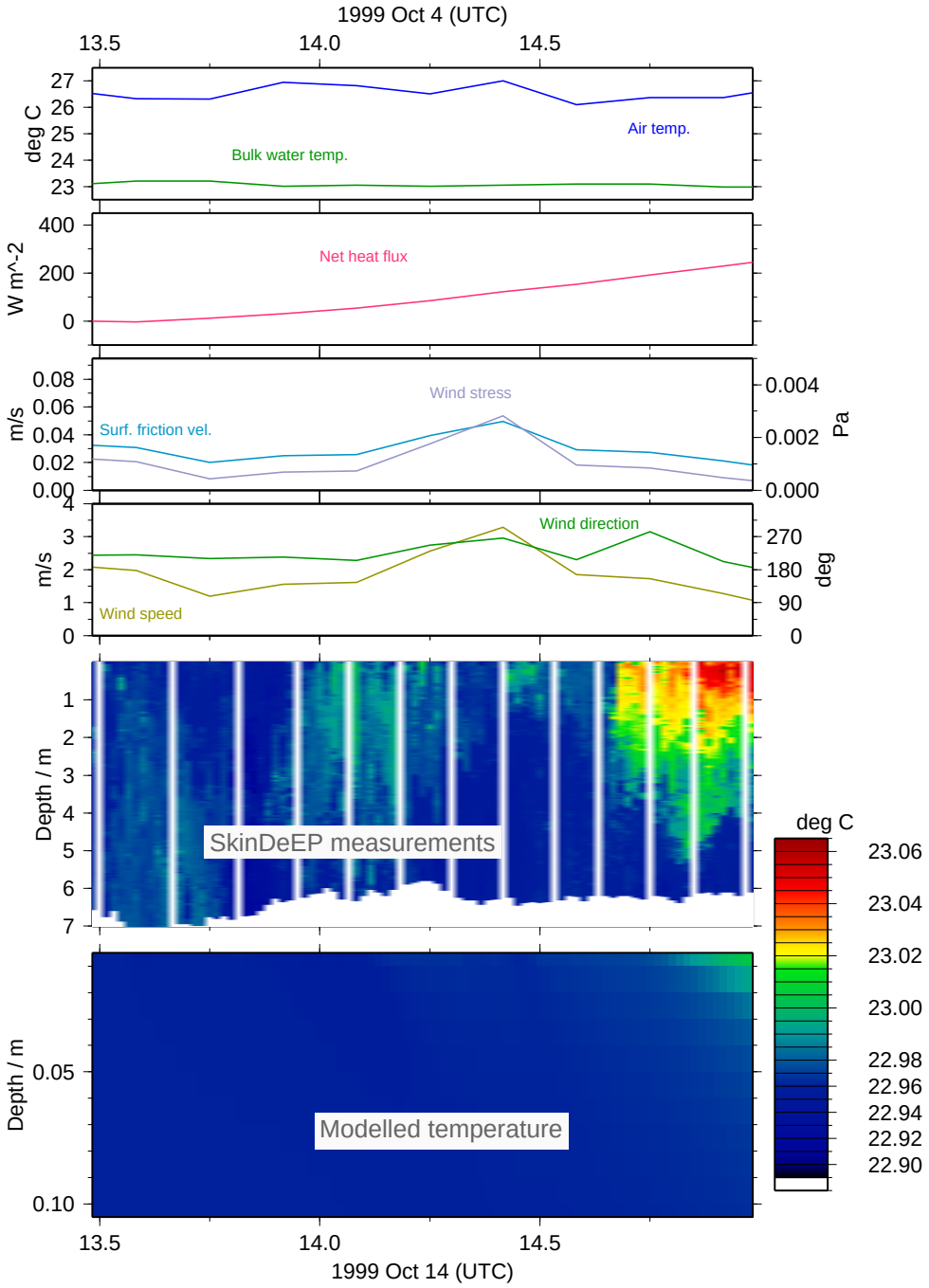

Figure 6: Observations and model results from Station Isla San Esteban 2, 1999 October 14

Additional measurements and model results

For the October 10 data set, a number of additional measurements and model results are presented. The downward heat flux $Q_m(t)$ through the surface computed from the present model is given by:

$$Q_m(t) = \rho_w C \int_0^t \frac{dT(0^-, t')}{dt'} \left[\frac{(u_*(t'))^2}{\lambda U(t')} \right] dt'. \tag{6}$$

The bulk−skin temperature difference and gas flux are computed from the background parameters (wind speed, air temperature, humidity, heat fluxes, precipitation) and SkinDeEP surface (bulk) temperature using the Molecular Oceanic Boundary LAyer Model (MOBLAM)[1], by Peter Schlüssel and A. V. Soloviev. The MOBLAM model is a surface renewal model, which takes account of solar and long wave radiation, turbulent and diffusive fluxes of sensible and latent heat, and impact of rain on the cool skin [4, 13, 15, 16].

Heat flux The heat flux computed by the present model, after a short period of adjustment, settles down to a fairly constant level which is roughly equal to the net heat flux deduced from observations. Given the simplicity of the model and the fact that no parameter adjustment has been made, this is remarkably good agreement.

Bulk−skin temperature difference The bulk−skin temperature difference calculated from the MOBLAM model gives values of between $-0.2\,\mathrm{K}$ and $-0.15\,\mathrm{K}$, which are considerably smaller than the measured values using SkinDeEP and the M-AERI interferometer. This considerable discrepancy requires further investigation.

Gas flux The CO_2 gas transfer velocity, computed by MOBLAM, shows rather interesting behaviour. It is fairly constant, except for enhanced values between 19:20 and 20:00. The increasing values are associated by an increase in wind stress, which will increase the amount of turbulent diffusion, whereas the decreasing values are associated with an increase in the sea surface temperature and consequent decrease in CO_2 solubility.

[1] MOBLAM is available on the Internet from `http://www.nova.edu/ocean/gasex/Moblam8.f90`, this particular version being from 1999 February 10. The model is described on `http://www.nova.edu/ocean/gasex/micro.html`.

5 Conclusion

The time-dependent model presented in this report for the evolution of the temperature and heat flux in the near-surface layer of the ocean does, obviously, have its limitations, as it does not take account of stratification or of the Earth's rotation. Nevertheless, given the simplicity of the model, and the fact that no tuning was performed of the model parameters, it does provide useful results on the time evolution of the temperature distribution in the oceanic surface layer over the measurement periods of the order of 2 hours, and its computed net surface heat flux values are largely consistent with observed values. A suitable application for the model would be as an interface between complex time-independent models for interfacial flux of momentum, heat, and mass, and time-dependent three-dimensional numerical models of ocean circulation which employ turbulence closure schemes and account for stratification and the Coriolis force.

The flux of CO_2 and other gas species through the sea surface can be computed using MOBLAM or similar time-independent surface models. A straightforward extension of the present time-dependent model should be capable of extending these flux predictions and computing the concentration and flux of CO_2 further into the water column.

References

[1] M. L. Banner, A. V. Babanin, and I. R. Young. Breaking probability for dominant waves on the sea surface. *Journal of Physical Oceanography*, 30(12):3145–3160, 2000.

[2] M. L. Banner and O. M. Phillips. On the incipient breaking of small scale waves. *Journal of Fluid Mechanics*, 65(4):647–656, 1974.

[3] H. Burchard, O. Petersen, and T. P. Rippeth. Comparing the performance of the Mellor–Yamada and the k–ϵ two-equation turbulence models. *Journal of Geophysical Research*, 103(C5):10543–10554, 1998.

[4] C. Craeye and P. Schlüssel. Rainfall on the sea: Surface renewal and wave damping. *Boundary-Layer Meteorology*, 89:349–355, 1998.

[5] M. A. Donelan, J. Hamilton, and W. H. Hui. Directional spectra of wind-generated waves. *Philosophical Transactions of the Royal Society of London, Series A*, 315:509–562, 1985.

[6] W. Eifler. Modelling the skin–bulk temperature difference near the sea–atmosphere interface for remote sensing applications. In *Proceedings of Conference, Space in the Service of a Changing Earth, Munich, 1992 March 30 – April 4*, pages 335–341, 1992.

[7] W. Eifler. A hypothesis on momentum and heat transfer near the sea–atmosphere interface and a related simple model. *Journal of Marine Systems*, 64:133–153, 1993.

[8] A. D. Jenkins. Do strong winds blow waves flat? In B. L. Edge and J. M. Hemsley, editors, *Ocean Wave Measurement and Analysis*, volume 1, pages 494–500. American Society of Civil Engineers, 2001. Proceedings, WAVES 2001, San Francisco.

[9] A. T. Jessup, C. J. Zappa, V. Hesany, M. R. Loewen, and M. G. Skafel. Dependence of the skin layer recovery rate on heat flux and temperature. In B. Jähne and E. C. Monahan, editors, *Selected Papers from the Third International Symposium on Air–Water Gas Transfer, July 24–27, 1995, Heidelberg University*, pages 601–610, 63453 Hanau, Germany, 1995. AEON Verlag & Studio.

[10] P. J. Minnett, W. McKeown, J. A. Hanafin, and O. Brown. Measurement of the ocean skin temperature using Fourier transform interferometric radiometers, 2000. Submitted to *J. Rem. Sens. Env.*

[11] P. J. Minnett and B. Ward. Measurement of near-surface ocean temperature variability—consequences on the validation of AATSR on ENVISAT, 2000. Submitted to *Proceedings of the ERS ENVISAT Symposium*, Gothenburg, Sweden.

[12] R. Nagaosa. Direct numerical simulation of vortex structures and turbulent scalar transfer across a free surface in a fully developed turbulence. *Physics of Fluids*, 11(6):1581–1595, 1999.

[13] P. Schlüssel, A. V. Soloviev, and W. J. Emery. Cool and freshwater skin of the ocean during rainfall. *Boundary-Layer Meteorology*, 82:437–472, 1997.

[14] P. Schlüssel, W. J. Emery, H. Grassl, and T. Mammen. On the bulk skin temperature difference and its impact on satellite remote sensing of sea surface temperature. *Journal of Geophysical Research*, 95:13 341—13 356, 1990.

[15] A. V. Soloviev and P. Schlüssel. Evolution of cool skin and direct air-sea gas transfer coefficient during daytime. *Boundary-Layer Meteorology*, 77:45–68, 1996.

[16] A. V. Soloviev and P. Schlüssel. Parameterization of the cool skin of the ocean and of the air–ocean gas transfer on the basis of modelling surface renewal. *Journal of Physical Oceanography*, 24:1339, 1994.

[17] B. Ward. Thermometric and radiometric measurements of sea surface temperature and their application to a neural network–based parameterisation of the bulk–skin temperature difference, 2000. Marie Curie Fellowship Final Report to the European Commission, Contract No. ERBFMBICT983162. Nansen Environmental and Remote Sensing Center, Bergen, Norway.

[18] B. Ward and P. J. Minnett. An autonomous profiler for near surface temperature measurements, 2000. In *Gas Transfer at Water Surfaces*, American Geophysical Union monograph, in press.

[19] B. Ward and S. Redfern. A neural network model for predicting the bulk−skin temperature difference at the sea surface. *International Journal of Remote Sensing*, 20:3533–3548, 1999.

A Model program code and documentation

A.1 Code for computing current profile and effect on waves (*blowflat.m*)

The code is written to be run by the open-source package *Octave*, available from `http://www.octave.org/`, and uses a syntax similar to that of programs for the commercial package *Matlab*. It plots the results by calling the freeware software package *Gnuplot*, available from `http://www.gnuplot.info`.

The code may be run on a Unix, Linux etc. system by typing the comand:

```
./blowflat.m
```

No arguments are required—to alter the parameters of the program run, the user should alter the source code directly.

The following output files are produced by the program:

`wind.eps` Plot file for wind time-series

`current.eps` Plot file for evolution of current profile with time

`waves.eps` Plot file evolution of wave spectra

`reduction.eps` Plot file showing the predicted fractional reduction of the wave spectrum due to the effect of near-surface current gradient

A.1.1 Source code for *blowflat.m*

```
#!/usr/bin/octave
#
# Revision 1.3  2001/10/18 13:06:18  vpvaj
# Several .eps files
#
# Revision 1.2  2001/10/18 12:59:21  vpvaj
# No titles (B+W version for Waves2001 proceedings)
#
# Revision 1.1  2001/10/18 12:57:04  vpvaj
# Initial revision
#

# Note: Capital and small letters define different variables.
```

```
global g
global t_1
global U_0
global U_1


# Physical data setup:

g = 9.80665 # m/s2
U_0 = 20 # m/s
U_1 = 30 # m/s

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3

A = 0.8e-3
B = 0.065e-3 # m-1 s

C_D0 = A + B * U_0
C_D1 = A + B * U_1

global lambda = 0.02 # ("surface eulerian drift factor")

Ustar0 = sqrt(C_D0) * U_0
Ustar1 = sqrt(C_D1) * U_1

global ustar0 = Ustar0 * sqrt(rho_a/rho_w)
global ustar1 = Ustar1 * sqrt(rho_a/rho_w)

global u_0s = lambda * U_0
global u_1s = lambda * U_1

global t_1 = 7200 # s  (2 h is approximately the inertial period / 2 pi at
               #  60 deg N or so)

function UU = UU(t)
  global t_1
  global U_0
  global U_1
  UU = (U_0 + U_1)/2 + sign(t-t_1)*(U_1 - U_0)/2
endfunction

function mu_0 = mu_0(t)
```

```
  global u_0s
  global ustar0
  mu_0 = u_0s ./ (ustar0 .* ustar0 * t)
endfunction

#
# Plotting setup:

gset terminal postscript eps
gset encoding iso_8859_1
gset grid


t = [(0:30:10800)]
data = [t'./3600, UU(t)'] # time axis in hours

gset output "wind.eps"
gset xlabel "time / h"
gplot [0:3] [0:40] data title "Wind speed U / m s^-1"

data2 = [t'./3600, -1./mu_0(t)']
gplot  [0:3] [-100:0] data2 title "-mu_0^-1 / m"

# The current as a function of depth and time:
gset output "current.eps"
gset multiplot

function u = u(z,t)
  global lambda
  global t_1
  global U_0
  global U_1
  global ustar0
  global ustar1
  u = lambda * (U_0 * exp( (lambda * U_0 * z') * (1 ./ (ustar0 * ustar0 .* t)))\
     + (t > t_1) * (U_1 - U_0) * exp( (lambda * (U_1 - U_0) * z' *  \
                          (1 ./ ((ustar1*ustar1 - ustar0*ustar0) \
                                * (t - t_1))))) \
     ) ;
endfunction

# Plot the current as a function of time:
```

```
z = (-20:0.1:0)
gset parametric
gset xrange [0:0.7]
gset yrange [-20:0]
gset zrange [-1:1] # dummy value
gset view 0,0,1,1
gset zlabel ""
gset noztics
gset xlabel "u / m s^-1"
gset ylabel "z / m"
gset nokey

gset label "WIND 20 m/s" at graph 0.1,1.05
gset label "WIND 30 m/s" at graph 0.65,1.05
gset label "5 min" at graph 0,0.95
gset label "2 h" at graph 0.55,0.95
gset label "3h20m" at graph 0.8,0.95
for t = 300:300:12000
  data3 = [u(z,t), z', zeros(size(z'))];
  gsplot data3
endfor

gset nomultiplot

gset nolabel

# The wave spectrum FF(f) as a function of peak frequency fp, fetch X
# and wind speed U10
# (source p 187 of Komen et al Dynamics and modelling of ocean waves,
# CUP 1994, ISBN 0-521-47047-1)
# Note that the figure shown on p186 gives fp as a function of fetch (in m)
# according to the following data and functions:
#
# fetch = [9500  20000  37000  52000  80000]
# fp = [0.39000  0.32000  0.29000  0.24000  0.22000]
#
# NB dimensionless energy and peak frequency as a function of fetch
# are given on p181
#
# If we non-dimensionalize using U10 and g, we have
# fp U10**2 / g = function( g X / U10**2)
#
# Now their Eq. 2.226b gives
```

```
#   nup = 2 pi fp U10/g = 13.7 (g X / U10**2)**(-0.27)
#   so   fp = (13.7 g / (2 pi U10)) * (g X / U10**2)**(-0.27)

function retval = fp_(X,U10)
  global g;
  retval = (13.7*g/(2*pi*U10)) * (g*X/(U10**2))**(-0.27);
endfunction

function retval = FF(f,fp,X,U10)
  global g
  alphaj = 0.076*(X*g/(U10*U10))**(-0.22)
  gammaj = 3.3
  sigmaja = 0.07
  sigmajb = 0.09
  sigmaj = sigmaja*(f<fp) + sigmajb*(f>=fp)
  GGamma = exp(-(f-fp)**2/(2*sigmaj**2*fp**2))

  retval = alphaj*g*g*((2*pi)**(-4)) * \
             (f**(-5))*exp(-1.25*(fp/f)**4)*gammaj**GGamma
endfunction

# Donelan version:
# This has no explicit fetch dependence, the dependence on fetch being
# integrated into the behaviour of omegap
#    Uc is basically U10 (but resolved along the main wave direction)
#
function retval = omegapf(X,U10)
  global g;
  retval = (13.7*g./U10) .* (g*X./(U10.**2)).**(-0.27);
endfunction

function retval = FF_D(omega,omegap,Uc)
  global g;

  cp = g./omegap;
  inv_age = Uc./cp;
  inv_age_alpha = min(max(0.83,inv_age),5);
  alphad = 0.006*inv_age_alpha.**0.55;
  inv_age_sigma =  min(max(1.0,inv_age),5);
  sigmad = 0.08*(1+4./inv_age_sigma.**3);
  inv_age_gamma2 =  min(inv_age,5);
  gammad = 1.7+(inv_age>=1)*6.0.*log(inv_age_gamma2);
  GGamma =  exp(-(omega-omegap).**2./(2*sigmad.**2.*omegap.**2));
```

```
  retval = \
    alphad.*g*g.*omega.**(-5).*(omega./omegap) \
      .*exp(-(omegap./omega).**4).*gammad.**GGamma;
endfunction

# Changing the fetch relation to the time relation:
#
# If we have   omegap = (13.7g/U10) * (g X/U10**2)**(-0.27)
#  we can determine cp = g/omegap
#               - group velocity  vg = g/(2 omegap)
#               - time  t = X / vg
#                         = 2 omegap X / g
#               - so    X = g t / (2 omegap)
#
#   Thus we have  omegap = (13.7g/U10)*(g**2 t / ( 2 omegap U10**2))**(-0.27)
#   =>     omegap = ((13.7g/U10)*(g**2 t / ( 2 U10**2))**(-0.27))**(1/1.27)

function retval = omegapt(t,U10)
  global g;
  retval = ((13.7*g./U10).*(g**2.*t ./ ( 2*U10.**2)).**(-0.27)).**(1/1.27);
endfunction

# Plot spectrum for 5-100 km fetch

Uc = 10;
freq = (0.01:0.01:1.0);
omega = 2*pi*freq;
gset output "waves.eps"
gset multiplot
gset parametric
gset xrange [0:0.7]  # use Hz
gset yrange [0:2]    # m2/Hz
gset zrange [-1:1]   # dummy
gset view 0,0,1,1
gset noztics
gset xlabel "Hz"
gset ylabel "m2/Hz"

# Plot spectrum for 15 min -2h time

Uc = 20;
freq = (0.001:0.001:1.0);
```

```
omega = 2*pi*freq;
gset parametric

gset xrange [0:0.5]  # use Hz
gset yrange [0:30]    # m2/Hz
gset zrange [-1:1]   # dummy
gset view 0,0,1,1
gset noztics
gset xlabel "Hz"
gset ylabel "m^2/Hz"
gset label "15min @ 20m/s" at graph 0.52,0.1
gset label "3h @ 20m/s" at graph 0.32,0.43

for t = [900:900:10800]
#for t = [14400]
 data3 = [omega'/(2*pi),\
   2*pi*FF_D(omega,omegapt(t,Uc),Uc)',zeros(size(omega'))];
 gsplot data3 with lines 1
endfor

Uc = 30;
freq = (0.001:0.001:1.0);
omega = 2*pi*freq;
gset parametric
gset zrange [-1:1]   # dummy
gset view 0,0,1,1
gset noztics

gset label "15min @ 30m/s" at graph 0.45,0.2
gset label "1h @ 30m/s" at graph 0.34,0.55
gset label "2h @ 30m/s" at graph 0.30,0.9


for t = [900:900:7200]
#for t = [14400]
 data3 = [omega'/(2*pi),\
   2*pi*FF_D(omega,omegapt(t,Uc),Uc)',zeros(size(omega'))];
 gsplot data3 with lines 3
endfor
gset nomultiplot

# Now compute the breaking crest elevation at various frequencies
```

```
function retval = redfac(omega_,t_)
  global g
  k = omega_'.*omega_'/g;
  c = g./omega_';
  us = u(0,t_);
  um2km1 = u(-0.5./k', t_) ;
  retval = (1 .- (us - um2km1)./c).**2;
#  retval = 0;
endfunction

# Plot the reduction factor:
gset nolabel
gset yrange [0.85:1.0]
gset ylabel "reduction"
gset output "reduction.eps"
gset multiplot
gset label "t = 5 min (20 m/s)" at graph 0.7,0.15
gset label "10 min" at graph 0.7,0.45
gset label "15 min" at graph 0.7,0.59
gset label "2 h" at graph 0.4,0.90
for t = 300:300:7200
  data3 = [omega'/(2*pi), redfac(omega,t), zeros(size(omega'))];
  gsplot data3 with lines 1
endfor

gset label "2h 1m" at graph 0.4,0.5
gset label "2h 5m (30 m/s)" at graph 0.15,0.65
gset label "3 h" at graph 0.25,0.90
for t = [7200:60:7500 7500:300:12000]
  data3 = [omega'/(2*pi), redfac(omega,t), zeros(size(omega'))];
  gsplot data3 with lines 3
endfor

gset nomultiplot
```

A.2 Code for computing time evolution of temperature (*surfflux.py*)

The code is written in the scripting language Python, which is freely available from `http://www.python.org/`. The following extra Python modules are required:

`mx.DateTime` Available freely from `http://www.egenix.com/files/python/mxDateTime.html` under the open source eGenix Public License Agreement (`http://www.egenix.com/files/python/mxLicense.html#Public`).

`Numeric.py` Available freely from `http://www.pfdubois.com/numpy`.

`Scientific.py` Available freely from `http://starship.python.net/~hinsen/ScientificPython/`.

The code may be run on a Unix, Linux etc. system by typing the comand:

```
./surfflux.py input_data_file output_file_prefix
```

where `input_data_file` is the name of the input data file, and `output_file_prefix` is the first part of the name of the output plot files.

The input data file consists of ASCII characters, each record containing the date and UTC time in ISO format, followed by the wind speed and the temperature in degrees Celsius just below the surface skin layer.

The following output file is produced by the program:

`<outprefix>_Tgrid.z` A two-dimensional ASCII array of temperatures, arranged according to time and depth.

A.2.1 Source code for *surfflux.py*

```
#! /usr/bin/env python

# Plots a time evolution of temperature based on forcing from surface
#
# The time is in seconds but may be converted to real times using
# mx.DateTime routines

# Usage: ./surfflux.py <input data file> <first part of output file name
# (without_<plot-type>.z>
```

```
import sys
import string
import os
import re
import fpformat
import gzip
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds

infile = sys.argv[1]
outprefix = sys.argv[2]

# Physical data setup:

g = 9.80665 # m/s2

f = open(infile,'r')
fr = f.readlines()

t_DT = []
U = zeros(len(fr),Float)
T0 = zeros(len(fr),Float)
t_s = zeros(len(fr),Float)
for i in range(len(fr)):
  t_DT.append(ISO.ParseDateTime(fr[i][:24]))
  if i==0: (da0,se0) = t_DT[i].absvalues()
  (da,se) = t_DT[i].absvalues()
  t_s[i] = DAY*(da-da0) + (se-se0)
  UT0_str = string.split(fr[i][24:])
  U[i] = string.atof(UT0_str[0])
  T0[i] = string.atof(UT0_str[1])

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3

A = 0.8e-3
B = 0.065e-3 # m-1 s
```

```
C_D = A + B * U

lambd = 0.02 # ("surface eulerian drift factor")

Ustar = sqrt(C_D) * U

ustar = Ustar * sqrt(rho_a/rho_w)

def UU(t):
  UU = 0.0
  for i in range(len(t_s)-1):
    UU = UU + U[i] * greater_equal(t,t_s[i]) * less(t,t_s[i+1])
  UU = UU + U[-1] * greater_equal(t,t_s[-1])
  return UU

def nu_exp(x):
  # exponential function, avoiding underflow
  MINARG = -100.0
  return where(greater(x,MINARG),exp(max(x,MINARG)),0.0)

def u(z,t):
  MIN_DT = 0.001 # seconds
  u = 0.0
  if t > t_s[0]: u = u + lambd * U[0] * \
      nu_exp((lambd*U[0]*z)/(ustar[0]*ustar[0]*t))
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: u =  u + lambd * (U[it+1]-U[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return u

def uT(t_h,z): return u(z,t_h*HOUR)

def T(z,t):
  MIN_DT = 0.001 # seconds
  T = T0[0]      # put in other initial conditions if more appropriate
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: T =  T + (T0[it+1]-T0[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return T

def TT(t_h,z): return T(z,t_h*HOUR)
```

```
# Plotting setup

t = arrayrange(10800/30 + 1)*30.0

UU_arr = UU(t)
t_h_arr = t/3600.0

z = (-0.01 * exp(arrayrange(51)*log(1000.0)/50))[::-1]


t_h_arr = arrayrange(0.0, 3.0 + 4.0/60, 2.0/60) # 3h 4min at 2 min intervals

of = open(outprefix+'_Tgrid.z','w')
TT_arr = zeros([len(t_h_arr),len(z)],Float)
for iz in arrayrange(len(z)):
  for it_h in arrayrange(len(t_h_arr)):
   TT_arr[it_h,iz] = TT(t_h_arr[it_h],z[iz])
   of.write(`TT_arr[it_h,iz]`+'\n')
of.close()
```

A.2.2 Source code for *temp-prof-hflux.py*

```
#! /usr/bin/env python

# Computes a time evolution of heat flux based on forcing from surface
#
# Usage: ./temp-prof-hflux.py <input data file (temperature)>
#   <input data file (wind)>
#   <first part of output file name>

# The input data file has the time in hours

import sys
import string
import os
import re
import fpformat
import gzip
#import Gnuplot
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds
#DUMMY = -999.0

infile = sys.argv[1]
windfile = sys.argv[2]
outprefix = sys.argv[3]

# Physical data setup:

g = 9.80665 # m/s2

# Read time and temperature
f = open(infile,'r')
fr = f.readlines()
f.close()

T0tt = zeros(len(fr),Float)
tt_s = zeros(len(fr),Float)
```

```
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  if i==0: t_s0 = HOUR*string.atof(thUT0_str[0])
  tt_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  T0tt[i] = string.atof(thUT0_str[2])
  # (note that thUT0_str[2] contains the (surface) depth)

# Read time, wind speed and friction velocity
# (We have already used awk to extract the relevant columns from a larger file)
f = open(windfile,'r')
fr = f.readlines()
f.close()
Utu = zeros(len(fr),Float)
Ustar_tu = zeros(len(fr),Float)
tu_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  tu_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  Utu[i] = string.atof(thUT0_str[1])
  Ustar_tu[i] = string.atof(thUT0_str[2])

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3
cp = 4200.0 # J kg-1 K-1

A = 0.8e-3
B = 0.065e-3 # m-1 s

# C_D = A + B * U

lambd = 0.02 # ("surface eulerian drift factor")

# Ustar = sqrt(C_D) * U

ustar_tu = Ustar_tu * sqrt(rho_a/rho_w)

# Generate an array of surface temperature, at regular intervals
dt = tt_s[1] - tt_s[0]
nt = int((tt_s[-1]-tt_s[0])/dt + 1.5)
newdt = ((tt_s[-1]-tt_s[0])/(nt-1))
t_s = arrayrange(tt_s[0],tt_s[-1]+newdt,newdt)
T0 = zeros(len(t_s),Float)
for i in range(len(t_s)):
```

```
  T0[i] = 0.0
  for it in range(len(tt_s)-1):
    if tt_s[it] <= t_s[i] < tt_s[it+1]:
      p = (t_s[i]-tt_s[it])/(tt_s[it+1]-tt_s[it])
      T0[i] = (1-p)*T0tt[it] + p*T0tt[it+1]
  if tt_s[-1] <= t_s[i]:
    T0[i] = T0tt[it]

# Generate an array of U and an array of ustar, at the times specified in the
# interpolated T file:

U = zeros(len(t_s),Float)
ustar = zeros(len(t_s),Float)
for i in range(len(t_s)):
  U[i] = 0.0
  ustar[i] = 0.0
  for iu in range(len(tu_s)-1):
     if tu_s[iu] <= t_s[i] < tu_s[iu+1]:
       p = (t_s[i]-tu_s[iu])/(tu_s[iu+1]-tu_s[iu])
       U[i] = (1-p)*Utu[iu] + p*Utu[iu+1]
       ustar[i] = (1-p)*ustar_tu[iu] + p*ustar_tu[iu+1]
  if tu_s[-1] <= t_s[i]:
    U[i] = Utu[-1]
    ustar[i] = ustar_tu[-1]

def UU(t):
  UU = 0.0
  for i in range(len(t_s)-1):
    if t_s[i] >= 0.0: UU = UU + Utu[i] * greater_equal(t,tu_s[i]) * \
      less(t,tu_s[i+1])
  UU = UU + Utu[-1] * greater_equal(t,tu_s[-1])
  return UU

def nu_exp(x):
  # exponential function, avoiding underflow
  MINARG = -100.0
  return where(greater(x,MINARG),exp(max(x,MINARG)),0.0)

def T(z,t):
  MIN_DT = 0.001 # seconds
  T = T0[0]      # put in other initial conditions if more appropriate
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: T =  T + (T0[it+1]-T0[it]) * \
```

```
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return T

def TT(t_h,z): return T(z,t_h*HOUR)

# Heat flux:

def dHdt(t):
  dHdt = 0.0
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: dHdt = dHdt + (T0[it+1]-T0[it]) * rho_w * cp * \
            ustar[it+1]**2 / (lambd * U[it+1])
  return dHdt

def dHdth(t_h): return dHdt(t_h*HOUR)

t_h_arr = arrayrange(0.0, 3.0 + 4.0/60, 2.0/60) # 3h 4min at 2 min intervals

of = open(outprefix+'_dHdt.dat','w')
dHdt_arr = zeros(len(t_h_arr),Float)
for it_h in arrayrange(len(t_h_arr)):
 dHdt_arr[it_h] = dHdth(t_h_arr[it_h])
 of.write(`(t_s0/HOUR)+t_h_arr[it_h]`+' '+`dHdt_arr[it_h]`+'\n')
of.close()
```

A.2.3 Source code for *temp-profile-oct04.py*

```
#! /usr/bin/env python

# Computes a time evolution of temperature based on forcing from surface
#
# Usage: ./temp-profile-oct04.py <input data file (temperature)>
#   <input data file (wind)> <first part of output file name>

# The input data file has the time in hours

import sys
import string
import os
import re
import fpformat
```

```
import gzip
#import Gnuplot
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds
#DUMMY = -999.0

infile = sys.argv[1]
windfile = sys.argv[2]
outprefix = sys.argv[3]

# Physical data setup:

g = 9.80665 # m/s2

# Read time and temperature
f = open(infile,'r')
fr = f.readlines()
f.close()

T0tt = zeros(len(fr),Float)
tt_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  if i==0: t_s0 = HOUR*string.atof(thUT0_str[0])
  tt_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  T0tt[i] = string.atof(thUT0_str[2])
  # (note that thUT0_str[2] contains the (surface) depth)

# Read time, wind speed and friction velocity
# (We have already used awk to extract the relevant columns from a larger file)
f = open(windfile,'r')
fr = f.readlines()
f.close()
Utu = zeros(len(fr),Float)
Ustar_tu = zeros(len(fr),Float)
tu_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
```

```
  tu_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  Utu[i] = string.atof(thUT0_str[1])
  Ustar_tu[i] = string.atof(thUT0_str[2])

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3

A = 0.8e-3
B = 0.065e-3 # m-1 s

# C_D = A + B * U

lambd = 0.02 # ("surface eulerian drift factor")

# Ustar = sqrt(C_D) * U

ustar_tu = Ustar_tu * sqrt(rho_a/rho_w)

# Generate an array of surface temperature, at regular intervals
dt = tt_s[1] - tt_s[0]
nt = int((tt_s[-1]-tt_s[0])/dt + 1.5)
newdt = ((tt_s[-1]-tt_s[0])/(nt-1))
t_s = arrayrange(tt_s[0],tt_s[-1]+newdt,newdt)
T0 = zeros(len(t_s),Float)
for i in range(len(t_s)):
  T0[i] = 0.0
  for it in range(len(tt_s)-1):
    if tt_s[it] <= t_s[i] < tt_s[it+1]:
      p = (t_s[i]-tt_s[it])/(tt_s[it+1]-tt_s[it])
      T0[i] = (1-p)*T0tt[it] + p*T0tt[it+1]
  if tt_s[-1] <= t_s[i]:
    T0[i] = T0tt[it]

# Generate an array of U and an array of ustar, at the times specified in the
# interpolated T file:

U = zeros(len(t_s),Float)
ustar = zeros(len(t_s),Float)
for i in range(len(t_s)):
  U[i] = 0.0
  ustar[i] = 0.0
  for iu in range(len(tu_s)-1):
     if tu_s[iu] <= t_s[i] < tu_s[iu+1]:
```

```
      p = (t_s[i]-tu_s[iu])/(tu_s[iu+1]-tu_s[iu])
      U[i] = (1-p)*Utu[iu] + p*Utu[iu+1]
      ustar[i] = (1-p)*ustar_tu[iu] + p*ustar_tu[iu+1]
  if tu_s[-1] <= t_s[i]:
    U[i] = Utu[-1]
    ustar[i] = ustar_tu[-1]

def UU(t):
  UU = 0.0
  for i in range(len(t_s)-1):
    if t_s[i] >= 0.0: UU = UU + Utu[i] * greater_equal(t,tu_s[i]) * \
      less(t,tu_s[i+1])
  UU = UU + Utu[-1] * greater_equal(t,tu_s[-1])
  return UU

def nu_exp(x):
  # exponential function, avoiding underflow
  MINARG = -100.0
  # print x,MINARG
  return where(greater(x,MINARG),exp(max(x,MINARG)),0.0)

def T(z,t):
  MIN_DT = 0.001 # seconds
  T = T0[0]      # put in other initial conditions if more appropriate
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: T =  T + (T0[it+1]-T0[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return T

def TT(t_h,z): return T(z,t_h*HOUR)


# Plotting setup

#z = (-0.005 * 10.0**(arrayrange(151)*(0.88167+2.0+log10(2.0))/150))[::-1]
z = (-0.005 - arrayrange(151)*7.00/151)[::-1] # linear scale this time.

#t_h_arr = arrayrange(0.0, 3.0 + 4.0/60, 2.0/60) # 3h 4min at 2 min intervals
t_h_arr = arrayrange(73)*120.0/3600.0
t_s_arr = 3600.0*t_h_arr

of = open(outprefix+'_Tgrid.z','w')
```

```
TT_arr = zeros([len(t_h_arr),len(z)],Float)
for iz in arrayrange(len(z)-1,-1,-1):
  print 'iz =', iz
  for it_h in arrayrange(len(t_h_arr)):
   TT_arr[it_h,iz] = TT(t_h_arr[it_h],z[iz])
#  of.write(`-z[iz]`+' '+`t_h_arr[it_h]`+' '+`TT_arr[it_h,iz]`+'\n')
   of.write(`TT_arr[it_h,iz]`+'\n')
of.close()
```

A.2.4 Source code for *temp-profile-oct14.py*

```
#! /usr/bin/env python

# Computes a time evolution of temperature based on forcing from surface
#
# Usage: ./temp-profile-oct04.py <input data file (temperature)>
#   <input data file (wind)> <first part of output file name>

# The input data file has the time in hours

import sys
import string
import os
import re
import fpformat
import gzip
#import Gnuplot
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds
#DUMMY = -999.0

infile = sys.argv[1]
windfile = sys.argv[2]
outprefix = sys.argv[3]

# Physical data setup:
```

```
g = 9.80665 # m/s2

# Read time and temperature
f = open(infile,'r')
fr = f.readlines()
f.close()

T0tt = zeros(len(fr),Float)
tt_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  if i==0: t_s0 = HOUR*string.atof(thUT0_str[0])
  tt_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  T0tt[i] = string.atof(thUT0_str[2])
  # (note that thUT0_str[2] contains the (surface) depth)

# Read time, wind speed and friction velocity
# (We have already used awk to extract the relevant columns from a larger file)
f = open(windfile,'r')
fr = f.readlines()
f.close()
Utu = zeros(len(fr),Float)
Ustar_tu = zeros(len(fr),Float)
tu_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  tu_s[i] = HOUR*string.atof(thUT0_str[0]) - t_s0
  Utu[i] = string.atof(thUT0_str[1])
  Ustar_tu[i] = string.atof(thUT0_str[2])

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3

A = 0.8e-3
B = 0.065e-3 # m-1 s

# C_D = A + B * U

lambd = 0.02 # ("surface eulerian drift factor")

# Ustar = sqrt(C_D) * U

ustar_tu = Ustar_tu * sqrt(rho_a/rho_w)
```

```
# Generate an array of surface temperature, at regular intervals
dt = tt_s[1] - tt_s[0]
nt = int((tt_s[-1]-tt_s[0])/dt + 1.5)
newdt = ((tt_s[-1]-tt_s[0])/(nt-1))
t_s = arrayrange(tt_s[0],tt_s[-1]+newdt,newdt)
T0 = zeros(len(t_s),Float)
for i in range(len(t_s)):
  T0[i] = 0.0
  for it in range(len(tt_s)-1):
    if tt_s[it] <= t_s[i] < tt_s[it+1]:
      p = (t_s[i]-tt_s[it])/(tt_s[it+1]-tt_s[it])
      T0[i] = (1-p)*T0tt[it] + p*T0tt[it+1]
  if tt_s[-1] <= t_s[i]:
    T0[i] = T0tt[it]

# Generate an array of U and an array of ustar, at the times specified in the
# interpolated T file:

U = zeros(len(t_s),Float)
ustar = zeros(len(t_s),Float)
for i in range(len(t_s)):
  U[i] = 0.0
  ustar[i] = 0.0
  for iu in range(len(tu_s)-1):
     if tu_s[iu] <= t_s[i] < tu_s[iu+1]:
       p = (t_s[i]-tu_s[iu])/(tu_s[iu+1]-tu_s[iu])
       U[i] = (1-p)*Utu[iu] + p*Utu[iu+1]
       ustar[i] = (1-p)*ustar_tu[iu] + p*ustar_tu[iu+1]
  if tu_s[-1] <= t_s[i]:
    U[i] = Utu[-1]
    ustar[i] = ustar_tu[-1]

def UU(t):
  UU = 0.0
  for i in range(len(t_s)-1):
    if t_s[i] >= 0.0: UU = UU + Utu[i] * greater_equal(t,tu_s[i]) * \
      less(t,tu_s[i+1])
  UU = UU + Utu[-1] * greater_equal(t,tu_s[-1])
  return UU

def nu_exp(x):
  # exponential function, avoiding underflow
```

```
  MINARG = -100.0
  # print x,MINARG
  return where(greater(x,MINARG),exp(max(x,MINARG)),0.0)

def T(z,t):
  MIN_DT = 0.001 # seconds
  T = T0[0]      # put in other initial conditions if more appropriate
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: T =  T + (T0[it+1]-T0[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return T

def TT(t_h,z): return T(z,t_h*HOUR)


# Plotting setup

#z = (-0.005 * 10.0**(arrayrange(151)*(0.88167+2.0+log10(2.0))/150))[::-1]
z = (-0.005 - arrayrange(101)*1.00/100)[::-1] # linear scale this time.

#t_h_arr = arrayrange(0.0, 3.0 + 4.0/60, 2.0/60) # 3h 4min at 2 min intervals
t_h_arr = arrayrange(91)*60.0/3600.0
t_s_arr = 3600.0*t_h_arr

of = open(outprefix+'_Tgrid.z','w')
TT_arr = zeros([len(t_h_arr),len(z)],Float)
for iz in arrayrange(len(z)-1,-1,-1):
  print 'iz =', iz
  for it_h in arrayrange(len(t_h_arr)):
   TT_arr[it_h,iz] = TT(t_h_arr[it_h],z[iz])
#  of.write(`-z[iz]`+' '+`t_h_arr[it_h]`+' '+`TT_arr[it_h,iz]`+'\n')
   of.write(`TT_arr[it_h,iz]`+'\n')
of.close()
```

A.3 Code for computing and plotting time evolution of temperature (*surfflux_gnuplot.py*)

The code is written in the scripting language Python, which is freely available from `http://www.python.org/`. The following extra Python modules are required:

`mx.DateTime` Available freely from `http://www.egenix.com/files/python/mxDateTime.html` under the open source eGenix Public License Agreement (`http://www.egenix.com/files/python/mxLicense.html#Public`).

`Numeric.py` Available freely from `http://www.pfdubois.com/numpy`.

`Scientific.py` Available freely from `http://starship.python.net/~hinsen/ScientificPython/`.

`Gnuplot.py` Available freely under the GNU General Public License (`http://www.fsf.org/copyleft/gpl.html`) from `http://gnuplot-py.sourceforge.net/`

It plots the results by calling the freeware software package *Gnuplot*, available from `http://www.gnuplot.info`.

The code may be run on a Unix, LInux etc. system by typing:

```
./surfflux_gnuplot.py <input data file> <first part of plot output
      file name>
```

The input data file consists of ASCII characters, each record containing the date and UTC time in ISO format, followed by the wind speed and the temperature in degrees Celsius just below the surface skin layer.

The following output files are produced by the program:

`<outprefix>_Tgrid.z` A two-dimensional ASCII array of temperatures, arranged according to time and depth.

`<outprefix>_cur.eps` A plot file of predicted near-surface current, computed by the same algorithm as in the program

`<outprefix>_cgrid.eps` A contour plot of predicted near-surface current.

`<outprefix>_Tgrid.ps` A contour plot of predicted temperature within the water column.

A.3.1 Source code for *surfflux_gnuplot.py*

```
#! /usr/bin/env python

# Plots a time evolution of temperature based on forcing from surface
#
# The time is in seconds but may be converted to UTC times using
# mx.DateTime routines

# Usage: ./surfflux_gnuplot.py <input data file>
# <first part of plot output file name
# (without_<plot-type>.[e]ps)>

import sys
import string
import os
import re
import fpformat
import gzip
import Gnuplot
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds

infile = sys.argv[1]
outprefix = sys.argv[2]

# Physical data setup:

g = 9.80665 # m/s2

f = open(infile,'r')
fr = f.readlines()

t_DT = []
U = zeros(len(fr),Float)
T0 = zeros(len(fr),Float)
t_s = zeros(len(fr),Float)
for i in range(len(fr)):
```

```
  t_DT.append(ISO.ParseDateTime(fr[i][:24]))
  if i==0: (da0,se0) = t_DT[i].absvalues()
  (da,se) = t_DT[i].absvalues()
  t_s[i] = DAY*(da-da0) + (se-se0)
  UT0_str = string.split(fr[i][24:])
  U[i] = string.atof(UT0_str[0])
  T0[i] = string.atof(UT0_str[1])

rho_a = 1.25 # kg/m3
rho_w = 1000 # kg/m3

A = 0.8e-3
B = 0.065e-3 # m-1 s

C_D = A + B * U

lambd = 0.02 # ("surface eulerian drift factor")

Ustar = sqrt(C_D) * U

ustar = Ustar * sqrt(rho_a/rho_w)

# t_1 = 7200.0 # s  (2 h is approximately the inertial period / 2 pi at
               #  60 deg N or so)

def UU(t):
  UU = 0.0
  for i in range(len(t_s)-1):
    UU = UU + U[i] * greater_equal(t,t_s[i]) * less(t,t_s[i+1])
  UU = UU + U[-1] * greater_equal(t,t_s[-1])
  return UU

def nu_exp(x):
  # exponential function, avoiding underflow
  MINARG = -100.0
  return where(greater(x,MINARG),exp(max(x,MINARG)),0.0)

def u(z,t):
  MIN_DT = 0.001 # seconds
  u = 0.0
  if t > t_s[0]: u = u + lambd * U[0] * \
      nu_exp((lambd*U[0]*z)/(ustar[0]*ustar[0]*t))
  for it in range(len(t_s)-1):
```

```
    if t > t_s[it+1]: u =  u + lambd * (U[it+1]-U[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return u

def uT(t_h,z): return u(z,t_h*HOUR)

def T(z,t):
  MIN_DT = 0.001 # seconds
  T = T0[0]      # put in other initial conditions if more appropriate
# if t > t_s[0]: T = T + T0[0] * \
#     nu_exp((lambd*U[0]*z)/(ustar[0]*ustar[0]*t))
  for it in range(len(t_s)-1):
    if t > t_s[it+1]: T =  T + (T0[it+1]-T0[it]) * \
            nu_exp((lambd*U[it+1]*z)/ \
            (ustar[it+1]**2 * (t - t_s[it+1])))
  return T

def TT(t_h,z): return T(z,t_h*HOUR)


# Plotting setup

g = Gnuplot.Gnuplot(debug=1,persist=0)

#g('set terminal postscript eps color')
g('set terminal postscript')
g('set encoding iso_8859_1')
g('set grid')
g('set data style lines')

t = arrayrange(10800/30 + 1)*30.0

UU_arr = UU(t)
t_h_arr = t/3600.0

g_wind = Gnuplot.Data(t_h_arr, UU_arr, title = "Wind speed U / m s^-1")

g('set output "'+outprefix+'_wind.eps"')
g('set xlabel "time / h"')

g.title('Wind speed U / m s^-1')
g('set xrange [11.0:14.5]')
```

```
g('set yrange [0:5]')
g.plot(g_wind)

# Now compute the current:

g('set output "'+outprefix+'_cur.eps"')
g('set multiplot')

z = (-0.01 * exp(arrayrange(51)*log(1000.0)/50))[::-1]

g('set parametric')
g('set xrange [0:0.7]')
g('set yrange [-20:0]')
g('set zrange [-1:1]') # dummy value
g('set zlabel ""')
g('set noztics')
g('set xlabel  "u / m s^-1"')
g('set ylabel "z / m"')
g('set nokey')
g('set label "WIND 20 m/s" at graph 0.1,1.05')
g('set label "WIND 30 m/s" at graph 0.65,1.05')
g('set label "5 min" at graph 0,0.95')
g('set label "2 h" at graph 0.55,0.95')
g('set label "3h20m" at graph 0.8,0.95')

for t in arrayrange(300.0, 12300.0, 300.0):
#for t in arrayrange(300.0, 600.0, 300.0):
  u_arr = u(z,t)
  g_u = Gnuplot.Data(u_arr,z)
  g.plot(g_u)

# Plot the current on a grid:
g('set nomultiplot')
#g('clear')
g.title('Contour plot of current')
g('set nolabel')
g('set key')
g('set output "'+outprefix+'_cgrid.eps"')

t_h_arr = arrayrange(0.0, 4.5, 0.16667)

# Plot the temperature on a grid:
g('set nomultiplot')
```

```
g.title('Contour plot of temperature')
g('set nolabel')
g('set key')
g('set output "'+outprefix+'_Tgrid.ps"')

# t_h_arr is already defined
TT_arr = zeros([len(t_h_arr),len(z)],Float)
for it_h in arrayrange(len(t_h_arr)):
 for iz in arrayrange(len(z)):
   TT_arr[it_h,iz] = TT(t_h_arr[it_h],z[iz])
g_Tgrid = Gnuplot.GridData(TT_arr,t_h_arr,-z)

g('set noparametric')
g('set nosurface')
g('set view 0.0,0.0,1.0,1.0')
g('set contour base')

g('set zrange[*:*]')
g('set grid')
g('set xlabel "time / h"')
g('set ylabel "z / m"')
g('set logscale y')
g('set xrange ['+`t_h_arr[0]`+':'+`t_h_arr[-1]`+']')
g('set yrange [10:0.009] reverse')
g('set cntrparam linear')
g('set cntrparam levels incremental 27.0,0.1,30.0')
g.splot(g_Tgrid)
```

A.4 Code for putting data on a regular grid (*gridit.py*)

The code is written in the scripting language Python, which is freely available from `http://www.python.org/`. The following extra Python modules are required:

`Numeric.py` Available freely from `http://www.pfdubois.com/numpy`.

The code may be run on a Unix, LInux etc. system by typing:

```
./gridit.py <input data file> <number of points in x-direction>
  <start coordinate in x-direction> <end coordinate in x-direction>
  <number of points in y-direction> <start coordinate in y-direction>
  <end coordinate in y-direction>
```

An output file is created, with the name `<input file name>gridded`.

A.4.1 Source code for *gridit.py*

```
#!/usr/bin/env python

# Usage: ./gridit.py filename nx xstart xend ny ystart yend

from Numeric import *
import string
import sys

DUMMY = -9999.0
TOL = 0.001  # Tolerance in y

filename = sys.argv[1]
nx = string.atoi(sys.argv[2])
xstart = string.atof(sys.argv[3])
xend = string.atof(sys.argv[4])
ny = string.atoi(sys.argv[5])
ystart = string.atof(sys.argv[6])
yend = string.atof(sys.argv[7])

outputfile = filename+'gridded'

if xend >= xstart:
  def firstx(xa,xb):
```

```
    return xa
  def lastx(xa,xb):
    return xb
else:
  def firstx(xa,xb):
    return xb
  def lastx(xa,xb):
    return xa

if yend >= ystart:
  def firsty(ya,yb):
    return ya
  def lasty(ya,yb):
    return yb
else:
  def firsty(ya,yb):
    return yb
  def lasty(ya,yb):
    return ya

def ixf(x):
  return int((nx-1)*(x-xstart)/(xend-xstart) + 0.5)

def iyf(y):
  return int((ny-1)*(y-ystart)/(yend-ystart) + 0.5)

dx = (xend-xstart)/(nx-1)
dy = (yend-ystart)/(ny-1)

def x(ix):
  return xstart + ix*dx

def y(iy):
  return ystart + iy*dy

f=open(sys.argv[1],'r')
lines = f.readlines()
f.close()

xyz = zeros([len(lines),3],Float)
z = zeros([nx,ny],Float) + DUMMY
for i in range(len(lines)):
  line = string.split(lines[i][:-1])
```

```
  for j in range(3):
    xyz[i,j] = string.atof(line[j])

for i in range(len(xyz)):
  if (i <= 0) or (abs(xyz[i,0] - xyz[i-1,0]) > TOL) :
    # New x
    # Determine the "lowest" y value
    yl = firsty(ystart,xyz[i,1])
    # Determine the "highest" y value
    yh = (xyz[i,1]+xyz[i+1,1])/2
    print "x =", xyz[i,0], "   yl = ", yl, "   yh = ", yh, \
      "   ix = ", ixf(xyz[i,0])
  elif (i+1 >= len(xyz)) or (abs(xyz[i+1,0] - xyz[i,0]) > TOL) :
    # Last of this x
    yl = (xyz[i-1,1]+xyz[i,1])/2
    yh = lasty(xyz[i,1],yend)
  else:
    if abs(xyz[i+1,0]-xyz[i-1,0]) > TOL : raise 'Error in xyz arrangement'
    yl = (xyz[i-1,1]+xyz[i,1])/2
    yh = (xyz[i,1]+xyz[i+1,1])/2

  # Now fill in the regular grid:
  ix = ixf(xyz[i,0])
  if 0 <= ix <= nx-1:
    for iy in range(iyf(yl),iyf(yh)+1):
      if 0 <= iy <= ny-1:
        z[ix,iy] = xyz[i,2]

# Write output file
f=open(outputfile,'w')
for j in arrayrange(ny)[::-1]:
  for i in range(nx):
    f.write(`z[i,j]`[:8]+'\n')
    if i < 3: print `x(i)`[:8]+' '+`y(j)`[:9]+' '+`z[i,j]`[:8]
f.close()
```

A.5 Code for controlling the MOBLAM model and handling the results (*tomoblam.py*, *runmoblam.sh*, *outmoblam.sh*)

A.5.1 Source code for *tomoblam.py*

```
#! /usr/bin/env python

# Computes a time evolution of temperature based on forcing from surface
#
# Usage: ./tomoblam.py

# The input data file has the time in hours

import sys
import string
import os
import re
import fpformat
import gzip
#import Gnuplot
from Numeric import *
from Scientific import *
from mx.DateTime import *

MINUTE = 60.0
HOUR = 3600.0
DAY = 86400.0 # seconds
#DUMMY = -999.0

datadir = '/usr/people/alastairj/projects/nersc-NFR-brian/DATA'
prefix = 'oct10'

infile = datadir+'/flux_'+prefix+'.dat.gz'
tempfile = prefix+'.surf'
outprefix = prefix

# Read the background data

f = gzip.open(infile,'r')
fr = f.readlines()
f.close()

background_data = zeros([len(fr),len(string.split(fr[0]))],Float)
```

```
for i in range(len(fr)):
  line = string.split(fr[i])
  for j in range(len(line)):
    background_data[i,j] = string.atof(line[j])
t_background_s = DAY*(background_data[:,-1]-int(background_data[0,-1]))
t_background_h = t_background_s/HOUR

# Read the surface temperature data

f = open(prefix+'.surf')
fr = f.readlines()
f.close()

T0tt = zeros(len(fr),Float)
tt_s = zeros(len(fr),Float)
for i in range(len(fr)):
  thUT0_str = string.split(fr[i])
  tt_s[i] = HOUR*string.atof(thUT0_str[0])
  T0tt[i] = string.atof(thUT0_str[2])
  # (note that thUT0_str[2] contains the (surface) depth)

# Generate an array of surface temperature, at regular intervals
dt = tt_s[1] - tt_s[0]
nt = int((tt_s[-1]-tt_s[0])/dt + 1.5)
newdt = ((tt_s[-1]-tt_s[0])/(nt-1))
t_s = arrayrange(tt_s[0],tt_s[-1]+newdt,newdt)
T0 = zeros(len(t_s),Float)
for i in range(len(t_s)):
  T0[i] = 0.0
  for it in range(len(tt_s)-1):
    if tt_s[it] <= t_s[i] < tt_s[it+1]:
      p = (t_s[i]-tt_s[it])/(tt_s[it+1]-tt_s[it])
      T0[i] = (1-p)*T0tt[it] + p*T0tt[it+1]
  if tt_s[-1] <= t_s[i]:
    T0[i] = T0tt[it]

# Interpolate the background data, at the times specified in the
# interpolated T file:

background_interp = zeros([len(t_s),len(background_data[0,:])],Float)
for i in range(len(t_s)):
  for iu in range(len(t_background_s)-1):
     if t_background_s[iu] <= t_s[i] < t_background_s[iu+1]:
```

```
        p = (t_s[i]-t_background_s[iu])/(t_background_s[iu+1]-t_background_s[iu])
        background_interp[i,:] = (1-p)*background_data[iu,:] + \
                                  p*background_data[iu+1,:]
  if t_background_s[-1] <= t_s[i]:
    background_interp[i,:] = background_data[-1,:]

# Write time to file:
t_h = t_s/HOUR
of = open(outprefix+'.hours','w')
for i in range(len(t_h)):
  of.write(`t_h[i]`+'\n')
of.close()

# Output parameters (input for moblam8):
ws = background_interp[:,16] # wind speed
Ta = background_interp[:,3] + 273.15  # air temperature

# Humidity
rh =  background_interp[:,13]
esdb = exp((16.78*(Ta-273.15) - 116.9)/(Ta - 273.15 + 237.3)) * 10.0
                       # saturation vapour pressure (hPa)
ed = rh * esdb / 100.0 # actual vapour pressure
p = background_interp[:,2]
q = ed * 16.0 / (ed * 16.0 + (p - ed) * 28.96)
tsurf = T0 + 273.15
qs = q
qsol = background_interp[:,10]
qlong = background_interp[:,11]
rr = background_interp[:,12]
sal = background_interp[:,14]
zu = 10.0
zt = 10.0
zq = 10.0
#skin_flag = 0
skin_flag = 0

# write to file:
of = open(outprefix+'_moblam.in','w')
for i in range(len(background_interp[:,0])):
  of.write(`ws[i]`[:8]+' '+`Ta[i]`[:8]+' '+`q[i]`[:8]+' '+\
    `tsurf[i]`[:8]+' '+`qs[i]`[:8]+' '+\
    `qsol[i]`[:8]+' '+`qlong[i]`[:8]+' '+`rr[i]`[:8]+' '+`sal[i]`[:8]+' '+\
    `zu`[:8]+' '+\
```

```
    ‘zt‘+’ ’+‘zq‘+’ ’+‘p[i]‘[:8]+’ ’+’\n’+‘skin_flag‘+’\n’)

#raise ’temp.stop’
```

A.5.2 Source code for *runmoblam.sh*

```
#!/bin/bash

while read line1
do
 read line2
 echo -e "$line1" "\n" "$line2" "\n" | \
    rsh eurus "~/projects/nersc-NFR-brian/programs/moblam/moblam"
done
```

A.5.3 Source code for *outmoblam.sh*

```
#!/bin/bash

# This script takes the moblam .out file and converts it into a file containing
# just dt_skin = tc + ts + trs + trv,  gc, and bc  ( = skin - bulk )

while read line1
do
  if test "${line1::5}" = "tmax,"
  then
    read line2
    read line3
    read line4
    tc=‘echo -e "$line3" | awk ’{print $2+$3+$4+$5}’‘
    gc=‘echo -e "$line4" | awk ’{print $3}’‘
    bc=‘echo -e "$line4" | awk ’{print $4}’‘
    echo $tc ’ ’ $gc ’ ’ $bc
  fi
done

# read line2
# echo -e "$line1" "\n" "$line2" "\n" | \
#    rsh eurus "~/projects/nersc-NFR-brian/programs/moblam/moblam"
```

A.6 Code for plotting the data from the MOCE-5 experiment (*plot-oct04.bash*, *plot-oct10.bash*, and *plot-oct14.bash*)

The code is written to be run by the *bash* shell in Unix or similar operating systems. It runs routines from the Generic Mapping Tools (GMT) package, available free under the GPL licence from `http://gmt.soest.hawaii.edu/`. It is derived from GMT example code, © 1991-2002, P. Wessel & W. H. F. Smith.

The code produces an output file in encapsulated postscript format.

A.6.1 Source code for *plot-oct04.bash*

```
#!/bin/bash
#
# Usage: ./plot-oct04.bash
#
# Script based upon the following script from GMT (GPL licence):
#
# GMT EXAMPLE 03
#
# @(#)job03.bash 1.10  11/30/99
#
# Purpose: Resample track data, do spectral analysis, and plot
# GMT progs: filter1d, fitcircle, gmtset, minmax, project, sample1d,
#  spectrum1d, trend1d, pshistogram, psxy, pstext
# Unix progs: $AWK, cat, echo, head, paste, rm, tail


# The minmax utility will report the minimum and maximum values for all columns.

#set -x

gmtset LABEL_FONT_SIZE 14p
gmtset MEASURE_UNIT CM
AWK=awk
FILEPREFIX=oct04

datadir=$HOME/projects/nersc-NFR-brian/DATA
echo "datadir = " $datadir

tmpfile="$FILEPREFIX.tmp"
tmpfile2="$FILEPREFIX.tmp2"
tmptextfile="$FILEPREFIX_txt.tmp"
```

```
gunzip -c $datadir/flux_$FILEPREFIX.dat.gz | dos2unix > $tmpfile
cat $tmpfile | minmax > $FILEPREFIX.minmax

declare -a xminmax
declare -a xmin
declare -a xmax
i=0
while [ $i -lt 19 ]
do
  cat > $FILEPREFIX.awk <<EOF
{print \$$((i+5))}
EOF
  $AWK -f $FILEPREFIX.awk $FILEPREFIX.minmax > $FILEPREFIX.awkresult
  xminmax[i]=`cat $FILEPREFIX.awkresult`
  xmin[i]=`cat $FILEPREFIX.awkresult | sed -e 's/<//' | sed -e 's/\/.*$//'`
  xmax[i]=`cat $FILEPREFIX.awkresult | sed -e 's/>//' | sed -e 's/^.*\///'`
  echo "xmin["$i"] = " ${xmin[i]}   "xmax["$i"] = " ${xmax[i]} \
   "xminmax["$i"] = " ${xminmax[i]}
  i=$((i+1))
done

# The array variables:
declare -a text
Hl=0;     text[$Hl]="Latent heat flux (W/m2)"                # $1 for awk
Hs=1;     text[$Hs]="Sensible heat flux (W/m2)"              # ...
Pa=2;     text[$Pa]="Barometric pressure (HPa)"
Ta=3;     text[$Ta]="Air temperature (degC)"
Tb=4;     text[$Tb]="Bulk water temperature (degC)"
U_star=5; text[$U_star]="Sea surface friction velocity (m/s)"
dlw=6;    text[$dlw]="Downwelling longwave (W/m2)"
dsw=7;    text[$dsw]="Downwelling shortwave (W/m2)"
lat=8;    text[$lat]="Latitude (deg)"
lon=9;    text[$lon]="Longitude (deg)"
nsw=10;   text[$nsw]="Net shortwave (W/m2)"
qlw=11;   text[$qlw]="Net longwave (W/m2)"
rain=12;  text[$rain]="Precipitation (mm)"
rh=13;    text[$rh]="Relative humidity (%)"
sal=14;   text[$sal]="Salinity (ppt)"
stress=15; text[$stress]="Wind stress (N/m2)"
ur=16;    text[$ur]="Wind speed at 10m (m/s)"
wd=17;    text[$wd]="Wind direction (deg)"                   # ...
yd=18;    text[$yd]="Day of year"                            # $19 for awk
```

```
i=0
while [ $i -lt 19 ]
do
  echo $i: ${text[$i]}
  i=$((i+1))
done

# We now should plot:
# - Air temperature (3), Bulk water temperature (4)
# - Latent heat flux (0), Sensible heat flux (1), Downwelling longwave (6),
#     Downwelling shortwave (7), Net shortwave (10), Net longwave (11)
# - Sea surface friction vel (5), Wind stress (15), Wind speed (16), Wind
#     direction (17)
# - against Hour of day (18):

# OK so we plot the air temperature first:
xplotmin="18.300000"
xplotmax="20.700000"
#xplotmin="0.0"
#xplotmax="24.0"
#yplotmin=`echo ${xmin[3]}-0.1 | bc -l`
#yplotmax=`echo ${xmax[3]}+0.1 | bc -l`
yplotmin=`python -c "print min(${xmin[3]},${xmin[4]})-0.1"`
yplotmax=`python -c "print max(${xmax[3]},${xmax[4]})+0.1"`
echo "xplotmin =" $xplotmin "   " "xplotmax =" $xplotmax
echo "yplotmin =" $yplotmin "   " "yplotmax =" $yplotmax
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
ranges="-R$xplotmin/$xplotmax/22.5/25.9"
projection_pars="$ranges -JX15.0/3.0"
filename="$FILEPREFIX.eps"
tmptextfile="$FILEPREFIX.txt.tmp"

d1=${1#o}; d1=${d1%10}; d2=${1#oct}; mmdd="O$d1 $d2"

psbasemap $projection_pars -B1f0.5:"1999 Oct 4 (UTC)"\
:/1f1:"deg C":WesN -Y19.8 -K  > $filename

$AWK '{print ($19-277.0)*24.0 " " $4}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/0/255p -O -K >> $filename
echo "20 20.5 10 0 0 0 Air temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/0/255 -O -K >> $filename
```

```
# Now plot the bulk water temperature (4 = $5)
$AWK '{print ($19-277.0)*24.0 " " $5}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/150/0p -O -K >> $filename
echo "20 22.5 10 0 0 0 Bulk water temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Now plot the heat fluxes:

yplotmin=`python -c "print min(${xmin[0]},${xmin[1]},${xmin[6]},${xmin[7]},\
${xmin[10]})*1.1"`
yplotmax=`python -c "print max(${xmax[0]},${xmax[1]},${xmax[6]},${xmax[7]},\
${xmax[10]})*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
#ranges="-R$xplotmin/$xplotmax/-100.0/150.0"
projection_pars="$ranges -JX15.0/3.0"

psbasemap $projection_pars -B1f0.5/500f1000:"W m^-2":Wesn -Y-3.3 -K -O  \
  >> $filename

# Latent heat flux:
#$AWK '{print ($19-277.0)*24.0 " " $1}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/150/p -O -K >> $filename
#echo "20 100 10 0 0 0 Latent ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename
#
## Sensible heat flux:
#$AWK '{print ($19-277.0)*24.0 " " $2}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/0/0p -O -K >> $filename
#echo "13.75 15 10 0 0 0 Sensible ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/0/0 -O -K >> $filename
#
## Net long-wave radiation:
#$AWK '{print ($19-277.0)*24.0 " " $12}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/125/0p -O -K >> $filename
#echo "20 200   10 0 0 0 Net long-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/125/0 -O -K >> $filename
#
## Net short-wave radiation:
#$AWK '{print ($19-277.0)*24.0 " " $11}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/125/255p -O -K >> $filename
#echo "20 500 10 0 0 0 Net short-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/125/255 -O -K >> $filename
```

```
# Net heat flux (sum of above)
$AWK '{print ($19-277.0)*24.0 " " $1+$2+$12+$11}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/255/50/125p -O -K >> $filename
echo "20 500 10 0 0 0 Net heat flux" > $tmptextfile
pstext $tmptextfile $projection_pars -G255/50/125 -O -K >> $filename


#Now plot the friction velocity:
yplotmin=0
yplotmax=`python -c "print ${xmax[5]}*1.1"`
#ranges="-R$xplotmin/$xplotmax/$yplotmin/0.095"
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-277.0)*24.0 " " $6}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/0.1f0.05:m/s:Wsn -Y-3.3 \
  -W1/0/150/200p -O -K >> $filename
echo "20  0.18 10  0 0 0 Surf. friction vel." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/200 -O -K >> $filename

#Wind stress:
yplotmin=0
yplotmax=`python -c "print ${xmax[15]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-277.0)*24.0 " " $16}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1.0f0.5/0.02f0.01:"Pa":Esn -Y0.0 \
  -W1/150/150/200p -O -K >> $filename
echo "19.5 0.01 10 0 0 0 Wind stress" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/200 -O -K >> $filename

#Wind speed:
yplotmin=0
yplotmax=`python -c "print ${xmax[16]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/4.5"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-277.0)*24.0 " " $17}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/1f0.5:"m/s":Wsn -Y-3.3 \
  -W1/150/150/0p -O -K >> $filename
echo "18.5 2.5 10 0 0 0 Wind speed" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/0 -O -K >> $filename
```

```
#Wind direction:
yplotmin=0
yplotmax=‘python -c "print ${xmax[17]}*1.1"‘
ranges="-R$xplotmin/$xplotmax/0.0/359.9"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK ’{print ($19-277.0)*24.0 " " $18}’ $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/90f45:deg:Esn \
  -W1/0/150/0p -Y0.0 -O -K >> $filename
echo "20.2 225.0 10 0 0 0 Wind direction" > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Skindeep results

datafile=$FILEPREFIX.dat
gunzip -c $datadir/skindeep_$FILEPREFIX.dat.gz  | dos2unix |\
  $AWK ’{print $3 " " $1 " " $2}’  > $datafile
./gridit.py $datafile 145 18.300000 20.700000 301 7.005 0.005
ranges="-R18.300000/20.700000/0.005/7.005"
projection_pars="$ranges -JX15.0/-6.0"
psbasemap $projection_pars -B1f0.5/1f1:"Depth / m":Wesn \
  -Y-6.6 -O -K >> $filename

# Generate axis labels
#axmin=‘echo "$xmin - 0.16" | bc -l‘
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -2 14 0 0 MR 1 cm
EOF

xyz2grd ${datafile}gridded -G$datafile.grd -I0.01666667/0.023333333 -ZBa \
  $ranges -D= -V
grdimage $projection_pars $datafile.grd -CTgrid_oct04.cpt \
  -O -K -V >> $filename
# pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "18.5 5.0 16 0 0 0 SkinDeEP measurements" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O -K  >> $filename
psscale -D15.5/-2/8.5/1.2 -L -B/:"        deg C": -CTgrid_oct04.cpt \
   -N300 -V -O -K >> $filename
```

```
# make a file containing the time, velocity and surface temperature:
#
cat $datafile | sed -e "/5\.00*e-0*3/! d" > $FILEPREFIX.surf
$AWK '{print ($19-277)*24 " " $17 " " $6}' $FILEPREFIX.tmp > \
$FILEPREFIX.UUstar
./temp-profile-oct04.py $FILEPREFIX.surf $FILEPREFIX.UUstar \
  $FILEPREFIX
xyz2grd ${FILEPREFIX}_Tgrid.z -G$datafile.grd -I0.0333333/0.04666666666 -ZBa \
  -R18.300000/20.7000000/0.005/7.005 -D= -V
grdimage $projection_pars $datafile.grd -B1f0.5:"1999 Oct 4 (UTC)":/1f1\
:"Depth / m":WeSn -CTgrid_oct04.cpt -Y-6.6 \
  -O -K -V >> $filename
# Generate axis labels
#axmin=`echo "$xmin - 0.16" | bc -l`
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -2 14 0 0 MR 1 cm
EOF
#pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "18.5 6.0 16 0 0 0 Modelled temperature" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O >> $filename

cp $filename $filename.tmp
cat $filename.tmp | \
  sed -e "/%%BoundingBox:/s/^.*$/%%BoundingBox: -100 0 761 668/" \
  > $filename
```

A.6.2 Source code for *plot-oct10.bash*

```
#!/bin/bash
#
# Usage: plot-oct10.bash
#
# Script based upon the following script from GMT (GPL licence):
#
# GMT EXAMPLE 03
#
```

```
# @(#)job03.bash 1.10  11/30/99
#
# Purpose: Resample track data, do spectral analysis, and plot
# GMT progs: filter1d, fitcircle, gmtset, minmax, project, sample1d,
#  spectrum1d, trend1d, pshistogram, psxy, pstext
# Unix progs: $AWK, cat, echo, head, paste, rm, tail


# The minmax utility will report the minimum and maximum values for all
# columns.

#set -x

gmtset LABEL_FONT_SIZE 14p
gmtset MEASURE_UNIT CM
AWK=awk
FILEPREFIX=oct10

if [[ `hostname` == kitlet ]]
then
  datadir=../DATA
else
  datadir=/usr/people/alastairj/projects/nersc-NFR-brian/DATA
fi
echo "datadir = " $datadir

tmpfile="$FILEPREFIX.tmp"
tmpfile2="$FILEPREFIX.tmp2"
tmptextfile="$FILEPREFIX_txt.tmp"

gunzip -c $datadir/flux_$FILEPREFIX.dat.gz | dos2unix > $tmpfile
cat $tmpfile | minmax > $FILEPREFIX.minmax

declare -a xminmax
declare -a xmin
declare -a xmax
i=0
while [ $i -lt 19 ]
do
  cat > $FILEPREFIX.awk <<EOF
{print \$$((i+5))}
EOF
  $AWK -f $FILEPREFIX.awk $FILEPREFIX.minmax > $FILEPREFIX.awkresult
```

```
  xminmax[i]=`cat $FILEPREFIX.awkresult`
  xmin[i]=`cat $FILEPREFIX.awkresult | sed -e 's/<//' | sed -e 's/\/.*$//'`
  xmax[i]=`cat $FILEPREFIX.awkresult | sed -e 's/>//' | sed -e 's/^.*\///'`
  echo "xmin["$i"] = " ${xmin[i]}   "xmax["$i"] = " ${xmax[i]} \
   "xminmax["$i"] = " ${xminmax[i]}
  i=$((i+1))
done

# The array variables:
declare -a text
Hl=0;     text[$Hl]="Latent heat flux (W/m2)"                # $1 for awk
Hs=1;     text[$Hs]="Sensible heat flux (W/m2)"              # ...
Pa=2;     text[$Pa]="Barometric pressure (HPa)"
Ta=3;     text[$Ta]="Air temperature (degC)"
Tb=4;     text[$Tb]="Bulk water temperature (degC)"
U_star=5; text[$U_star]="Sea surface friction velocity (m/s)"
dlw=6;    text[$dlw]="Downwelling longwave (W/m2)"
dsw=7;    text[$dsw]="Downwelling shortwave (W/m2)"
lat=8;    text[$lat]="Latitude (deg)"
lon=9;    text[$lon]="Longitude (deg)"
nsw=10;   text[$nsw]="Net shortwave (W/m2)"
qlw=11;   text[$qlw]="Net longwave (W/m2)"
rain=12;  text[$rain]="Precipitation (mm)"
rh=13;    text[$rh]="Relative humidity (%)"
sal=14;   text[$sal]="Salinity (ppt)"
stress=15; text[$stress]="Wind stress (N/m2)"
ur=16;    text[$ur]="Wind speed at 10m (m/s)"
wd=17;    text[$wd]="Wind direction (deg)"                   # ...
yd=18;    text[$yd]="Day of year"                            # $19 for awk

i=0
while [ $i -lt 19 ]
do
  echo $i: ${text[$i]}
  i=$((i+1))
done

# We now should plot:
# - Air temperature (3), Bulk water temperature (4)
# - Latent heat flux (0), Sensible heat flux (1), Downwelling longwave (6),
#     Downwelling shortwave (7), Net shortwave (10), Net longwave (11)
# - Sea surface friction vel (5), Wind stress (15), Wind speed (16), Wind
#     direction (17)
```

```
# - against Hour of day (18):

# OK so we plot the air temperature first:
xplotmin="18.266667"
xplotmax="21.283333"
#xplotmin="0.0"
#xplotmax="24.0"
#yplotmin=`echo ${xmin[3]}-0.1 | bc -l`
#yplotmax=`echo ${xmax[3]}+0.1 | bc -l`
yplotmin=`python -c "print min(${xmin[3]},${xmin[4]})-0.1"`
yplotmax=`python -c "print max(${xmax[3]},${xmax[4]})+0.1"`
echo "xplotmin =" $xplotmin "   " "xplotmax =" $xplotmax
echo "yplotmin =" $yplotmin "   " "yplotmax =" $yplotmax
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
filename="$FILEPREFIX.eps"
tmptextfile="$FILEPREFIX.txt.tmp"

d1=${1#o}; d1=${d1%10}; d2=${1#oct}; mmdd="O$d1 $d2"

psbasemap $projection_pars \
-B1f0.5:"1999 Oct 10 (UTC)":/1f1:"deg C":WsN -Y19.8 -K  > $filename

$AWK '{print ($19-283.0)*24.0 " " $4}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/0/255p -O -K >> $filename
echo "20 28.5 10 0 0 0 Air temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/0/255 -O -K >> $filename

# Now plot the bulk water temperature (4 = $5)
$AWK '{print ($19-283.0)*24.0 " " $5}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/150/0p -O -K >> $filename
echo "20 27.5 10 0 0 0 Bulk water temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Now plot the bulk - skin temperature difference
yplotmin="-2.0"
yplotmax="1.0"
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
psbasemap $projection_pars -B1f0.5/1f0.5:"deg C":E -Y0.0 -K -O >> $filename
inputfile="${FILEPREFIX}.measDT"
$AWK '{print $1+7 " " $2}' $inputfile > $tmpfile2 # Time in input file is local
psxy $tmpfile2 $projection_pars -W1/0/0/150p -O -K >> $filename
```

```
echo "20.25 0.5 10 0 0 0 Measured bulk - skin" > $tmptextfile
pstext $tmptextfile $projection_pars -G0/0/150 -O -K >> $filename
inputfile="${FILEPREFIX}_DT_gc_bc.dat"
timefile="${FILEPREFIX}.hours"
paste $timefile $inputfile | $AWK '{print $1 " " (-$2)}' > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/150/50/100p -O -K >> $filename
echo "20.3 0.0 10 0 0 0 Computed bulk - skin" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/50/100 -O -K >> $filename

#Now plot the computed gas transfer velocity (m/s)  Range 1-2 x 10**-4

yplotmin="1.0"
yplotmax="1.9"
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
psbasemap $projection_pars -B1f0.5/0.5f0.1:"10^-4 m/s":Wesn -Y-3.3 -K -O \
  >> $filename
inputfile="${FILEPREFIX}_DT_gc_bc.dat"
timefile="${FILEPREFIX}.hours"
paste $timefile $inputfile | $AWK '{print $1 " " (($3+$4)*10000.0)}' > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/50/150/100p -O -K >> $filename
echo "20.0 1.7 10 0 0 0 Computed CO2 gas transfer velocity" > $tmptextfile
pstext $tmptextfile $projection_pars -G50/150/100 -O -K >> $filename

#Now plot the heat fluxes:

yplotmin=`python -c \
"print min(${xmin[0]},${xmin[1]},${xmin[6]},${xmin[7]},${xmin[10]})*1.1"`
yplotmax=`python -c \
"print max(${xmax[0]},${xmax[1]},${xmax[6]},${xmax[7]},${xmax[10]})*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
#ranges="-R$xplotmin/$xplotmax/-100.0/150.0"
projection_pars="$ranges -JX15.0/3.0"

psbasemap $projection_pars -B1f0.5/500f1000:"W m^-2":Wesn -Y-3.3 -K -O  \
  >> $filename

# Latent heat flux:
#$AWK '{print ($19-283.0)*24.0 " " $1}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/150/p -O -K >> $filename
#echo "20 100 10 0 0 0 Latent ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename
#
```

```
## Sensible heat flux:
#$AWK '{print ($19-283.0)*24.0 " " $2}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/0/0p -O -K >> $filename
#echo "13.75 15 10 0 0 0 Sensible ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/0/0 -O -K >> $filename
#
## Net long-wave radiation:
#$AWK '{print ($19-283.0)*24.0 " " $12}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/125/0p -O -K >> $filename
#echo "20 200   10 0 0 0 Net long-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/125/0 -O -K >> $filename
#
## Net short-wave radiation:
#$AWK '{print ($19-283.0)*24.0 " " $11}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/125/255p -O -K >> $filename
#echo "20 500 10 0 0 0 Net short-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/125/255 -O -K >> $filename

# Net heat flux (sum of above)
$AWK '{print ($19-283.0)*24.0 " " $1+$2+$12+$11}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/255/50/125p -O -K >> $filename
echo "19.3 700 10 0 0 0 Net heat flux" > $tmptextfile
pstext $tmptextfile $projection_pars -G255/50/125 -O -K >> $filename
psxy ${FILEPREFIX}_dHdt.dat $projection_pars -W1/155/50/125p -O -K >> $filename
echo "20 500 10 0 0 0 Net h.f. from model" > $tmptextfile
pstext $tmptextfile $projection_pars -G155/50/125 -O -K >> $filename


#Now plot the friction velocity:
yplotmin=0
yplotmax=`python -c "print ${xmax[5]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/0.05"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-283.0)*24.0 " " $6}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars \
-B1f0.5/0.02f0.01:m/s:Wsn -Y-3.3 -W1/0/150/200p \
  -O -K >> $filename
echo "20  0.04 10  0 0 0 Surf. friction vel." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/200 -O -K >> $filename

#Wind stress:
yplotmin=0
```

```
yplotmax=‘python -c "print ${xmax[15]}*1.1"‘
ranges="-R$xplotmin/$xplotmax/$yplotmin/0.0019"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK ’{print ($19-283.0)*24.0 " " $16}’ $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/0.001f0.0005f0.1:"Pa":Esn -Y0.0 \
  -W1/150/150/200p -O -K >> $filename
echo "19.5 0.00025 10 0 0 0 Wind stress" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/200 -O -K >> $filename

#Wind speed:
yplotmin=0
yplotmax=‘python -c "print ${xmax[16]}*1.1"‘
ranges="-R$xplotmin/$xplotmax/$yplotmin/2.5"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK ’{print ($19-283.0)*24.0 " " $17}’ $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/1f0.5:"m/s":Wsn -Y-3.3 \
  -W1/150/150/0p -O -K >> $filename
echo "20.0 1.8 10 0 0 0 Wind speed" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/0 -O -K >> $filename

#Wind direction:
yplotmin=0
yplotmax=‘python -c "print ${xmax[17]}*1.1"‘
ranges="-R$xplotmin/$xplotmax/0.0/359.9"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK ’{print ($19-283.0)*24.0 " " $18}’ $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1f0.5/90f45:deg:Esn \
  -W1/0/150/0p -Y0.0 -O -K >> $filename
echo "19.5 45.0 10 0 0 0 Wind direction" > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Skindeep results

datafile=$FILEPREFIX.dat
gunzip -c $datadir/skindeep_$FILEPREFIX.dat.gz  | dos2unix |\
  $AWK ’{print $3 " " log($1)/log(10) " " $2}’  > $datafile
#./gridit.py $datafile  182 18.266667 21.283333 301 0.88167 -2.30103
ranges="-R18.266667/21.283333/-2.30103/0.88167"
projection_pars="$ranges -JX15.0/-6.0"
psbasemap $projection_pars -B1f0.5/1f1wesn \
```

```
  -Y-6.6 -O -K >> $filename

# Generate axis labels
#axmin=‘echo "$xmin - 0.16" | bc -l‘
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -2 14 0 0 MR 1 cm
EOF

xyz2grd ${datafile}gridded -G$datafile.grd -I0.01666667/0.010609 -ZBLa \
  $ranges -D= -V
grdimage $projection_pars $datafile.grd -CTgrid.cpt \
  -O -K -V >> $filename
pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "18.5 0.3 16 0 0 0 SkinDeEP measurements" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O -K  >> $filename
psscale -D15.5/3/4.5/1.2 -L -B/:"        deg C": -CTgrid.cpt -N300 -V -O -K \
   >> $filename

# Next, produce the plot of the modelled results from the surface data and the
# friction velocity info.

# First make a file containing the time, velocity and surface temperature:
#
cat $datafile | sed -e "/-2\.30103/! d" > $FILEPREFIX.surf
$AWK ’{print ($19-283)*24 " " $17 " " $6}’ $FILEPREFIX.tmp > $FILEPREFIX.UUstar
#./temp-profile.py $FILEPREFIX.surf $FILEPREFIX.UUstar $FILEPREFIX
xyz2grd ${FILEPREFIX}_Tgrid.z -G$datafile.grd -I0.0333333/0.021218 -ZBLa \
  -R18.266667/21.3333306/-2.30103/0.88167 -D= -V
grdimage $projection_pars $datafile.grd -B1f0.5:"1999 Oct 10 (UTC)":/1f1weSn \
   -CTgrid.cpt -Y-6.6 \
  -O -K -V >> $filename
# Generate axis labels
#axmin=‘echo "$xmin - 0.16" | bc -l‘
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
```

```
$axmin -2 14 0 0 MR 1 cm
EOF
pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "18.5 0.3 16 0 0 0 Modelled temperature" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O >> $filename

#cat $filename | \
#  sed -e 's/^%%BoundingBox: .*$/%%BoundingBox: -95 0 827 568/' > \
#  ${filename%%.eps}_new.eps
```

A.6.3 Source code for *plot-oct14.bash*

```
#!/bin/bash
#
# Usage: ./plot-oct14.bash
#
# Script based upon the following script from GMT (GPL licence):
#
# GMT EXAMPLE 03
#
# @(#)job03.bash 1.10  11/30/99
#
# Purpose: Resample track data, do spectral analysis, and plot
# GMT progs: filter1d, fitcircle, gmtset, minmax, project, sample1d,
#  spectrum1d, trend1d, pshistogram, psxy, pstext
# Unix progs: $AWK, cat, echo, head, paste, rm, tail


# The minmax utility will report the minimum and maximum values for all columns.

#set -x

gmtset LABEL_FONT_SIZE 14p
gmtset MEASURE_UNIT CM
AWK=awk
FILEPREFIX=oct14

if [[ `hostname` == kitlet ]]
then
  datadir=../DATA
else
```

```
  datadir=/usr/people/alastairj/projects/nersc-NFR-brian/DATA
fi
echo "datadir = " $datadir

tmpfile="$FILEPREFIX.tmp"
tmpfile2="$FILEPREFIX.tmp2"
tmptextfile="$FILEPREFIX_txt.tmp"

gunzip -c $datadir/flux_$FILEPREFIX.dat.gz | dos2unix > $tmpfile
cat $tmpfile | minmax > $FILEPREFIX.minmax

declare -a xminmax
declare -a xmin
declare -a xmax
i=0
while [ $i -lt 19 ]
do
  cat > $FILEPREFIX.awk <<EOF
{print \$$((i+5))}
EOF
  $AWK -f $FILEPREFIX.awk $FILEPREFIX.minmax > $FILEPREFIX.awkresult
  xminmax[i]=`cat $FILEPREFIX.awkresult`
  xmin[i]=`cat $FILEPREFIX.awkresult | sed -e 's/<//' | sed -e 's/\/.*$//'`
  xmax[i]=`cat $FILEPREFIX.awkresult | sed -e 's/>//' | sed -e 's/^.*\///'`
  echo "xmin["$i"] = " ${xmin[i]}   "xmax["$i"] = " ${xmax[i]} \
   "xminmax["$i"] = " ${xminmax[i]}
  i=$((i+1))
done

# The array variables:
declare -a text
Hl=0;     text[$Hl]="Latent heat flux (W/m2)"                # $1 for awk
Hs=1;     text[$Hs]="Sensible heat flux (W/m2)"              # ...
Pa=2;     text[$Pa]="Barometric pressure (HPa)"
Ta=3;     text[$Ta]="Air temperature (degC)"
Tb=4;     text[$Tb]="Bulk water temperature (degC)"
U_star=5; text[$U_star]="Sea surface friction velocity (m/s)"
dlw=6;    text[$dlw]="Downwelling longwave (W/m2)"
dsw=7;    text[$dsw]="Downwelling shortwave (W/m2)"
lat=8;    text[$lat]="Latitude (deg)"
lon=9;    text[$lon]="Longitude (deg)"
nsw=10;   text[$nsw]="Net shortwave (W/m2)"
qlw=11;   text[$qlw]="Net longwave (W/m2)"
```

```
rain=12;  text[$rain]="Precipitation (mm)"
rh=13;    text[$rh]="Relative humidity (%)"
sal=14;   text[$sal]="Salinity (ppt)"
stress=15; text[$stress]="Wind stress (N/m2)"
ur=16;    text[$ur]="Wind speed at 10m (m/s)"
wd=17;    text[$wd]="Wind direction (deg)"                    # ...
yd=18;    text[$yd]="Day of year"                             # $19 for awk

i=0
while [ $i -lt 19 ]
do
  echo $i: ${text[$i]}
  i=$((i+1))
done

# We now should plot:
# - Air temperature (3), Bulk water temperature (4)
# - Latent heat flux (0), Sensible heat flux (1), Downwelling longwave (6),
#     Downwelling shortwave (7), Net shortwave (10), Net longwave (11)
# - Sea surface friction vel (5), Wind stress (15), Wind speed (16), Wind
#     direction (17)
# - against Hour of day (18):

# OK so we plot the air temperature first:
xplotmin="13.483333"
xplotmax="14.983333"
#xplotmin="0.0"
#xplotmax="24.0"
#yplotmin=`echo ${xmin[3]}-0.1 | bc -l`
#yplotmax=`echo ${xmax[3]}+0.1 | bc -l`
yplotmin=`python -c "print min(${xmin[3]},${xmin[4]})-0.1"`
yplotmax=`python -c "print max(${xmax[3]},${xmax[4]})+0.1"`
echo "xplotmin =" $xplotmin "   " "xplotmax =" $xplotmax
echo "yplotmin =" $yplotmin "   " "yplotmax =" $yplotmax
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
ranges="-R$xplotmin/$xplotmax/22.5/27.5"
projection_pars="$ranges -JX15.0/3.0"
filename="$FILEPREFIX.eps"
tmptextfile="$FILEPREFIX.txt.tmp"

d1=${1#o}; d1=${d1%10}; d2=${1#oct}; mmdd="O$d1 $d2"

psbasemap $projection_pars \
```

```
-B0.5f0.25:"1999 Oct 4 (UTC)":/1f1:"deg C":WesN -Y19.8 -K  > $filename

$AWK '{print ($19-287.0)*24.0 " " $4}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/0/255p -O -K >> $filename
echo "14.7 25.0 10 0 0 0 Air temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/0/255 -O -K >> $filename

# Now plot the bulk water temperature (4 = $5)
$AWK '{print ($19-287.0)*24.0 " " $5}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/0/150/0p -O -K >> $filename
echo "13.8 24.0 10 0 0 0 Bulk water temp." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Now plot the heat fluxes:

yplotmin=`python -c \
"print min(${xmin[0]},${xmin[1]},${xmin[6]},${xmin[7]},${xmin[10]})*1.1"`
yplotmax=`python -c \
"print max(${xmax[0]},${xmax[1]},${xmax[6]},${xmax[7]},${xmax[10]})*1.1"`
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
ranges="-R$xplotmin/$xplotmax/-100.0/450.0"
projection_pars="$ranges -JX15.0/3.0"

psbasemap $projection_pars -B0.5f0.25/200f100:"W m^-2":Wesn -Y-3.3 -K -O  \
  >> $filename

# Latent heat flux:
#$AWK '{print ($19-287.0)*24.0 " " $1}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/150/p -O -K >> $filename
#echo "20 100 10 0 0 0 Latent ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename
#
## Sensible heat flux:
#$AWK '{print ($19-287.0)*24.0 " " $2}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/0/0p -O -K >> $filename
#echo "13.75 15 10 0 0 0 Sensible ht. flux" > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/0/0 -O -K >> $filename
#
## Net long-wave radiation:
#$AWK '{print ($19-287.0)*24.0 " " $12}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/255/125/0p -O -K >> $filename
#echo "20 200   10 0 0 0 Net long-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G255/125/0 -O -K >> $filename
```

```
#
## Net short-wave radiation:
#$AWK '{print ($19-287.0)*24.0 " " $11}' $tmpfile > $tmpfile2
#psxy $tmpfile2 $projection_pars -W1/0/125/255p -O -K >> $filename
#echo "20 500 10 0 0 0 Net short-wave rad." > $tmptextfile
#pstext $tmptextfile $projection_pars -G0/125/255 -O -K >> $filename

# Net heat flux (sum of above)
$AWK '{print ($19-287.0)*24.0 " " $1+$2+$12+$11}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -W1/255/50/125p -O -K >> $filename
echo "14 250 10 0 0 0 Net heat flux" > $tmptextfile
pstext $tmptextfile $projection_pars -G255/50/125 -O -K >> $filename


#Now plot the friction velocity:
yplotmin=0
yplotmax=`python -c "print ${xmax[5]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/0.095"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-287.0)*24.0 " " $6}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B0.5f0.25/0.02f0.01:m/s:Wsn -Y-3.3 \
  -W1/0/150/200p -O -K >> $filename
echo "13.5 0.04  10  0 0 0 Surf. friction vel." > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/200 -O -K >> $filename

#Wind stress:
yplotmin=0
yplotmax=`python -c "print ${xmax[15]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/0.005"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-287.0)*24.0 " " $16}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B1.0f0.5/0.002f0.001:"Pa":Esn -Y0.0 \
  -W1/150/150/200p -O -K >> $filename
echo "14.25 0.0035 10 0 0 0 Wind stress" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/200 -O -K >> $filename

#Wind speed:
yplotmin=0
yplotmax=`python -c "print ${xmax[16]}*1.1"`
ranges="-R$xplotmin/$xplotmax/$yplotmin/4.0"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
```

```
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-287.0)*24.0 " " $17}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B0.5f0.25/1f0.5:"m/s":Wsn -Y-3.3 \
  -W1/150/150/0p -O -K >> $filename
echo "13.5 0.50 10 0 0 0 Wind speed" > $tmptextfile
pstext $tmptextfile $projection_pars -G150/150/0 -O -K >> $filename

#Wind direction:
yplotmin=0
yplotmax=`python -c "print ${xmax[17]}*1.1"`
ranges="-R$xplotmin/$xplotmax/0.0/359.9"
#ranges="-R$xplotmin/$xplotmax/$yplotmin/$yplotmax"
projection_pars="$ranges -JX15.0/3.0"
$AWK '{print ($19-287.0)*24.0 " " $18}' $tmpfile > $tmpfile2
psxy $tmpfile2 $projection_pars -B0.5f0.25/90f45:deg:Esn \
  -W1/0/150/0p -Y0.0 -O -K >> $filename
echo "14.5 290.0 10 0 0 0 Wind direction" > $tmptextfile
pstext $tmptextfile $projection_pars -G0/150/0 -O -K >> $filename

#Skindeep results

datafile=$FILEPREFIX.dat
gunzip -c $datadir/skindeep_$FILEPREFIX.dat.gz  | dos2unix |\
  $AWK '{print $3 " " $1 " " $2}'  > $datafile
./gridit.py $datafile 91 13.483333 14.983333 301 7.005 0.005
ranges="-R13.483333/14.983333/0.005/7.005"
projection_pars="$ranges -JX15.0/-6.0"
psbasemap $projection_pars -B0.5f0.25/1f1:"Depth / m":Wesn \
  -Y-6.6 -O -K >> $filename

# Generate axis labels
#axmin=`echo "$xmin - 0.16" | bc -l`
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -2 14 0 0 MR 1 cm
EOF

xyz2grd ${datafile}gridded -G$datafile.grd -I0.01666667/0.023333333 -ZBa \
  $ranges -D= -V
grdimage $projection_pars $datafile.grd -CTgrid_oct14.cpt \
```

```
  -O -K -V >> $filename
# pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "13.75 5.0 16 0 0 0 SkinDeEP measurements" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O -K  >> $filename
psscale -D15.5/-2/8.0/1.2 -L -B/:"        deg C": -CTgrid_oct14.cpt \
   -N300 -V -O -K >> $filename

# Model run:
# make a file containing the time, velocity and surface temperature:
#
cat $datafile | sed -e "/5\.00*e-0*3/! d" > $FILEPREFIX.surf
$AWK '{print ($19-287)*24 " " $17 " " $6}' $FILEPREFIX.tmp > $FILEPREFIX.UUstar
python ./temp-profile-oct14.py $FILEPREFIX.surf $FILEPREFIX.UUstar \
#  $FILEPREFIX
ranges="-R13.483333/14.983333/0.005/1.005"
projection_pars="$ranges -JX15.0/-6.0"
xyz2grd ${FILEPREFIX}_Tgrid.z -G$datafile.grd \
  -I0.0166666667/0.010 -ZBa \
  $ranges -D= -V
ranges="-R13.483333/14.983333/0.005/0.105"
projection_pars="$ranges -JX15.0/-6.0"
grdimage $projection_pars $datafile.grd \
  -B0.5f0.25:"1999 Oct 14 (UTC)":/0.05f0.01\
:"Depth / m":WeSn -CTgrid_oct14.cpt -Y-6.6 \
  -O -K -V >> $filename
# Generate axis labels
#axmin=`echo "$xmin - 0.16" | bc -l`
axmin=18.15
cat > $tmptextfile <<EOF
$axmin 0  14 0 0 MR 1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -1 14 0 0 MR 0.1 m
$axmin -2 14 0 0 MR 1 cm
EOF
#pstext $tmptextfile $projection_pars -G0/0/0 -N -O -K >> $filename
echo "14.0 0.075 16 0 0 0 Modelled temperature" > $tmptextfile
pstext $tmptextfile $projection_pars -G100/100/100 -W250/250/250 \
  -N -O >> $filename

cp $filename $filename.tmp
cat $filename.tmp | \
  sed -e "/%%BoundingBox:/s/^.*$/%%BoundingBox: -100 0 761 668/" \
```

```
> $filename
```