One-stream heat evacuation from the planet.

Joseph Reynen
Le Ducal U111
Port Marina Baie des Anges
06270 Villeneuve-Loubet
France
jwreynen@aol.com

In this paper the MATLAB listing is presented of the one-stream stack model for the heat evacuation from the planet. In the introduction the links are given for the description of the model including results of runs with the present pro

```
%layersnov2015.m

%      INTRODUCTION
%      The present MATLAB program gives a numerical analysis of the heat
%      transport by the one-way  radiation through the atmosphere.
%      A stack of semi-transparent grids represents the IR-active
%      trace gases.
%      It turns out that the radiation from the surface to the atmosphere
%      is rather small: heat goes by convection to the higher layers, and
%      only the higher layers emit by radiation the heat to outer space.

%      In the listing comments are given to follow up the procedures.
%      Description of the underlying mathematics and of the results
%      are given in the following articles, references
%      [1],[2],[3],[4],[5],[6],[7],[8]
%
%      [1] http://www.tech-know-group.com/papers/IR-absorption_updated.pdf
%          This is the earliest paper in which the equations are written by a
%          finite difference technique.
%
%      [2] http://principia-scientific.org/publications/
%             PROM/PROM_REYNEN_Finite_Element.pdf
%           More or less the same as [1] but the equations are
%          implemented with a finite element technique.
%           The present MATLAB listing is based on the FEM formulation.
%          The paper is also available at the site of Hans Scheuder:
%          http://www.tech-know-group.com/papers/Finite_Element.pdf
%
%      [3] http://tech-know-group.com/papers/planckabsorption.pdf
%          In this paper the stack model is compared to the Schwarzschild way
%          of solution. It turns out that the Schwarzschild proposal is the
%          same as the earlier engineering proposal of looking for pairs
%          of emitters and absorbers which are communicating with each other.
%          In appendix 4 of this reference the results of [1] and [2] are
%          compared as concerns the view factors.
%
%           OR ALSO IN
%          http://principia-scientific.org/publications/
%              PROM/PRO M_REYNEN_Planck_absorption.pdf

%      [4] http://tech-know-group.com/papers/Prevost_no_back-radiation.pdf
%          This paper repeats the work of Christiansen from back in 1883.
%          It defines the heat transport by radiation between two plates
%          with a different emission coefficient.
%          It clearly shows that the net exchange of heat between two
%          plates is zero when the plates have the same temperature.
```

```
%         The Prevost statement that
%         q(1-->2) = eps1*sigma*T1^4 and q(2-->1) = eps2*sigma*T2^4
%         cannot be true for eps1 not equal to eps2.
%         We should write for T1>T2: q(1-->2)=eps12*sigma*(T1^4-T2^4) with
%         1/eps12=1/eps1+1/eps2 -1
%         In practical terms:
%              "look for pairs of surfaces to exchange heat by radiation"
%         See also Wikipedia, emission.
%
%      [5]http://tech-know-group.com/papers/Transient.pdf
%         In this paper the transient option 4 is described
%         OR ALSO IN
%            http://principia-scientific.org/publications/
%                  PROM/PROM-REYNEN_Transient.pdf
%      [6]http://tech-know-group.com/papers/vacuum.pdf
%         In this paper examples of stacks in vacuum are given
%         showing that in vacuum radiation dominates and in
%         the atmosphere convection
%         OR ALSO IN
%            http://principia-scientific.org/publications/
%                  PROM/PROM-REYNEN_vacuum.pdf
%      [7] http://tech-know-group.com/papers/sensitivity.pdf
%          OR ALSO IN
%          http://principia-scientific.org/publications/
%                  PROM/PROM-REYNEN_sensitivity.pdf

%          In this paper the sensitivity of doubling the CO2 concentration
%          is calculated with a model with a height of 30 km.
%          The doubling of CO2 gives rise to a surface temperature increase
%          of 0.04 K
%
%      [8] http://tech-know-group.com/papers/surface_emission.pdf
%          OR ALSO IN
%          http://principia-scientific.org/publications/
%                  PROM/PROM-REYNEN_surface_emission.pdf
%
%          In this paper the stack model is validated by means of a
%          comparison with two-stream papers:
%          Ferenc Miskolczi, K&T, KNMI Rob van Dorland and NASA Norman Loch
%
%
%     Further information can be obtained from the author: jwreynen@aol.com
%


%
%     Various analyses can be carried out by the MATLAB program:
%
%  option (1)
%     Straightforward resolution to define temperatures due to radiation
%     of heat in vacuum,assuming outer space at zero K and surface at
temperature
%     TsC, defned by input. The heat absorption and re-emission by IR-active
%     molecules is modelled by a stack of grids, giving rise to an
%     absorption matrix K depending on absorption factors,
%     defined by the cross-section of the grid.
%     In reference [1] the finite difference equations are derived, in
%     reference [2] a finite element approach is given.
%     The present MATLAB program uses the finite element description.
%     For a stack in vacuum (or no heat exchange with the air) K*theta=0,
%     where theta represents sigma*T^4 and sigma the Stefan-Boltzmann constant.
%     Boundary conditions: theta(1)=epssurf*sigma*TsK^4 and
%     theta(nods)= sigma*zeroK^4=0, epssurf=emission of the surface.
%     These boundary conditions are introduced by means of Lagrange multipliers.
%     In the listing the concept of Lagrange multipliers is explained with
```

```
%     green comments.
%     Also in ref [1] the technique of Lagrange multipliers is explained.
%
%     Figures of the temperature distribution are given as function of ftot.
%     Besides, the surface flux qs is defined, which for this option of a
%     stack in vacuum is equal to the outgoing longwave radiation (OLR),
%     because there is no convection in vacuum. This case is not very realistic.
%     The reason that the analysis is included is for demonstration purposes
%     only:the IR-active gases are not heating up the atmosphere as is
%     advocated by IPCC authors.
%     When it is assumed that the trace gases are isolated from the bulk
%     of the atmosphere, the trace gases are rather cold while the
%     atmosphere has a temperature according to the environmental lapse rate.
%     The trace gases do not heat the atmosphere.
%     It is the other way around,when the trace gases are not isolated from the
%     19% oxygen O2 and 80% nitrogen N2, represening 99% of the atmosphere,
%     the bulk of the atmosphere is keeping the traces of IR-active gases
%     warm by molecular collisions, also called diffusion,despite the heat
%     loss to outer space by emission.
%
%
%
%
%  option(2)
%     Reversed technique q = K*theta with theta = sigma*(TsK+LR*z)^4,
%     where TsK+LR*z is the air temperature distribution, with lapse rate LR.
%     It is assumed that the IR-active trace gases do not cool to the low
%     temperatures as calculated in option (1) but they are kept warm by
%     molecular collision -also called conduction- with the 99% bulk of O2
%     and N2 in the air, with a temperature defined by the environmental
%     lapse rate ELR = -6.5 K/km.
%     The calculated fluxes q are:
%
%     q(1):    the first component is the LW surface flux, including the flux
%              through the atmospheric window, directly towards outer space
%     q(nods): the last component represents the outgoing LW radiation: -OLR
%
%     The remaining components are the necessary heat deposits by mechanisms
%     other than LW radiation to compensate for the loss of heat due to LW
%     emission by the traces of IR-active gases!
%     The air temperature distribution is given by the lapse rate LR and
%     the surface temperature.
%     Figures are given of LW surface flux and OLR as function of ftot.
%     The difference between OLR and qsurf is the heat deposited in the
%     atmosphere by convection of active and latent heat and the heat
%     due to absorption of SW incoming radiation.
%
%     Results of the steady state equation q=K*theta are given for 3
%     different temperature distibutions:
%     Case 1: The atmospheric temperature distibution remains as defined by
%             the lapse rate with an original surface temperature TsK0=288.
%             However results are calculated for surface temperatures varying
%             from Ts= TsK0-deltaT to Ts=TsK0+deltaT.
%
%     Case 2: In the higher atmosphere the distibution remains as defined by
%             the lapse rate with an original surface temperature TsK0=288.
%             However results are calculated for near surface temperatures
%             varying from T(z)= TsK0-deltaT*exp(-z/mixlair) to
%             T(z)= TsK0+deltaT*exp(-z/mixlair).
%
%     Case 3: The atmospheric temperature is defined by the lapse rate and
%             a modified surface temperature: T(z)=TsK +deltaT LR*z.
%
%     Case 1 and case 3 are the extremes for transient temperature variations:
```

```
%            case 1 for very fast variations where the atmospheric temperature
%            variations do not change, and
%            case 3 for slow variations where the complete column of air is
involved.
%
%     Input variations fot lapse rates.
%        (1)  dry adiabatic lapse rate  DALR = -g/cp= -9.81 K/km
%        (2)  saturated adiabiatic lapse rate SALR= -5 K/km
%        (3)  isothermal atmosphere with lapse LR = -0.0001 K/km
%
%     Different surface temperature can be used with Ts=15C=288K as
%     default value.
%     For an hypothetical isothermal atmosphere the stacks, representing
%     the IR-active molecules, do not absorb radiation from the surface
%     nor from radiation within the atmosphere, because of the supposed
%     isothermal temperature distribution.
%     The stack radiates only to outer space!
%     This heat should arrive at the axial stations by mechanisms other
%     than LW radiation which would be difficult in this hypothetical
%     case of an isothermal atmosphere!
%
%
%  option(3)
%      For a givenftot = 0.849 the daily variation of OLR, of the surface flux
and of the
%      flux through the window are defined corresponding to daily surface
temperatures
%      variations around 288 C and corresponding atmospheric temperatures close
%      to the surface are estimated..
%      The diffusion of heat in the sub-surface (sea or land) together with
%      the near surface air is modeled by finite elements,
%      Nze nodes (input, typical 10) nodes in the sub-surface and nods nodes
%      (input, typical 40) in the atmosphere.
%      For the sea surface, the incoming sunlight is absorbed in the first
%      meter of the sea, in the pogram it is an input variable.
%      To simulate the effect of waves the conductivity  in the upperlayers
%      is taken big.
%      In the same way for the air above the surface, in the lower elements
%      near the surface conductivity is taken high.
%
%      An inversion occurs in the early morning and a physical back-radiation
results,
%      since the atmosphere is under such circumstances warmer than the surface.
%
%  option(4)
%     The various analyses of this option are described in appendix 2 of [3]
%     It is the so-called Schwarzschild procedure, which is a two-stream
%     formulation but wrongly interpreted by IPCC authors.%the splitting up
%     of the radiation in upward and downward fluxes was done in the
%     beginning  of 1900 by Schwarzschild in order to be able toformulate
%     analyticcal solutions, although formulated as integrals which were
%     evaluated by quadrature techniques available at the time.
%
%  option(5)
%     Reversed solution technique for the sensitivity analysis for the surface
temperature.
%     for doubling CO2 concentration.
%     We take the height of the model for the sensitivity analysis 30 km,
%     because CO2 continues to be present at those heights at a volumetric
%     concentration with the same percentage as the surface concentration
%     that means proportional to the density.
```

```
% Make a choice achoice of the options 1,2,3,4,5



option=0.1;

while option >0                % outer loop N
clear
%loop over the total program, until option =0 or > 5
disp('                 LIST OF OPTIONS')
disp('option == 1:     Temperature calculation for a stack in vacuum')
disp('option == 2:     Reverse solution for absorption concentration from fmin')
disp('                 to fmax for various lapse rates and surface temperatures')
disp('option == 3:     OLR, qsurface and qwindow flux as function of ')
disp('                 ELR temperature distibution with daily average')
disp('                 surface temperature TsK. Sub-surface soil(or
sea)temperatures')
disp('                 and the near surface air temperatures daily variations
are')
disp('                 defined by transient FEM analyses')
disp('option == 4:     Schwarzschild type of solutions')
disp('option == 5:     Sensitivity analysis for doubling CO2 concentration.
Height 30 km' )
disp('  ')


iter=1;
while iter>0
option=input('Make a choice for an option, finish with 0:')
  if option < 6
    if option>-1
        iter=0;
    end
  end
     if iter==2
     iter=0;
     option=0;
     end
     if iter==1
     disp('Option should be between 0 and 5')
     disp('Hit any key and try once more')
     pause
     iter=iter+1
     end

end

% physical properties

R  = 287.058;          % ideal gas  constant per J/kg
cp = 1000;             % specific heat of air at constant pressure J/kg
g  = 9.81;             % gravity acceleration m/s^2

%  radiation  properties
windowMatrix=0 %1;
windowfactor = 'windowV';
windowfactor
%         Alternative values can be given for the different options:
%
%         (windowMatrix=1)
%          the viewfactors are defined for each pair (i,j) of the mesh
%          This option is recommended, according to FEM formulation
%
```

```
%          (windowMatrix==0)
%           only the viewfacors between nodes (i,nods) are defined, it are
%           the windows from node i to outer space.
%           It was the viewfactor used in the finite difference formulation.
%


% initial surface temperatures
TsC= 15;              % surface temperature in C
TsK= TsC+273;         % surface temperature in K
TsK0=TsK;             % initial surface temperature
sigma=5.67*10^-8;     % Stefan-Bolzmann constant  Watt/m^2/K^4
Planck=[TsK^3,TsK^2,TsK,1,1/TsK];
Planck=Planck*sigma;
                  %play with different temperature dependences
                  %P;anck(5):  theta=sigma/TsK*T^5
                  %Planck(4):  theta=sigma*T^4            SB relation
                  %Planck(3):  theta=sigma*TsK*T^3
                  %Planck(2):  theta=sigma*TsK^2*T^2
                  %Planck(1):  theta=sigma*Tsk^3*T linear, like conduction
Planckcase=4;     %default value, according to Stefan Boltzmann


%  lapse rates,
ELR  = -6.5/1000;       % environmental lapse rate K/km,
DALR = -g/cp;           % dry adiabatic lapse rate, just to show the
                        % the influence of the lapse rate,
                        %see [2] for a derivation from the ideal gas law
                        %and Newtons law for gravitation.
SALR = -5/1000;         % saturated lapse rate
LR   = ELR;             %default value
 %   solar flux
qtoa  =  240;           % flux at top of atmosphere according to NASA, W/m^2
                        % used in plots for finding reference point, see K&T
                        % diagrams in reference [1,2,3]

%geometry and meshes
heightkm=11;            % default height of model in km, can be changed
toakm=heightkm                      % in particular for sensitivity analysis
option 5
                        % where it is taken as 30 km.
height=heightkm*1000;% convert data from km into meters
toa=height
nods=40;                % nodes in the atmosphere,  including surface node=1
                        % and outer space node = nods
                        % we take it higher in option 5 for CO2 sensitivity
                        % computer time is not an issue!


depthsea =  -5;          % sea depth in meters for the diffusion analyses
depthland  =  -0.15;      % land depth in meters the diffusion anlyses
Nze =10;                 % nodes in the sub surface analyses option 7
Ntot=nods+Nze-1;

%mesh generation parameters
ratioatm=1.2 ;              % ratio for the geometrical series of distribution
of nods.
                           % mesh size of order of cm near surface, of order of
km
                           % at TOA
ratiosea=1.5;


%distribution of IR-active trace gases
```

```
m= 7;                    % coefficient for the exponential distribution of water
vapour
                         % users are invited to play with it
                         % m=7 turned out to give results corresponding to
                         % experimental results of Miskolczi publications see
ref[7]
epssurf=0.923;           % according to data of Miskolczi, emissivity of 70%
probably better
                         % but taken into account by albedo giving a SW
reflection
                         % users are invited to play with it

givenftot= 0.8348;       % this value is to start an iteration in order to
                          % obtain aN OLR=qtoa.
                          % absorption coefficient by comparison to
                          % Ferenc Miskolczi see reference
                          % [1],[6], [7]


ftotco2=givenftot/1000
ftoth2o=givenftot-ftotco2;

                           % sensitivity around givenftot, this is the absorption
of the stack
                           % giving rise to results corresponding to the
                           % published K&T  diagrams discussed in references
[1,2,3,6,7]
                           % the flux through the atmospheric window becomes
                           % (1-ftot)*epssurf*sigma*Ts^4
                           % contribution of CO2, taken as 0.1% of the absorption
of the
                           % average atmosphere
                           % mainly governed by water vapour: giventot

schwarz=1;                 % only in option 3 we make schwarz=2 for
                           % the Schwarzschild viewfactors
                           % if windowMatrix==0 then viewfactor=ones(nods) except
last
                           % column which becomes windowF see [1] or
                           % appendix 4 of ref [3]
                           % if windowMatrix =1 then viewfactorF see ref 2 and
                           % appendix 4 of [3]


  %alternative input for options 1, 2 ,3,4
if option>0
 if option <5               % for option 1,2,3,4


     disp('For this option you can specify alternative input')
     input0=input('Alternative input Click NE 0, click 0 for standard:')
     %atmospheric temperature at surface z=0

     if input0==0
         disp('Standard input')
     else
     disp('windowMatrix=1 means the complete viewfactorF MATRIX is used ')
     disp('windowMatrix=0 means only the windowF VECTOR is used')
     windowMatrix= input('Make a choice for windowMatrix, 1 or 0 :')

                           % if windowMatrix==0 then viewfactor=ones(nods) except
last
```

```
                            % column which becomes windowF see {1} or
                            % appendix 4 of ref [3]
                            % if windowMatrix ==1 then full viewfactorF matrix,
see ref 2 and
                            % ref 3
                            % Note of june 2015: comparison with two-stream
                            % formulations, with the huge "back-radiation" it
                            % seems that IPCC authors as well as Miskolsci
                            % have used the version windowMatrix=0. But they
                            % do not give explicitly the window flux.
                            % For that reason the default value is now taken
                            % as windowMatrix=0.
    if windowMatrix<=0
     windowMatrix=0;
     windowfactor='windowV';
    end
    if windowMatrix>=1
     windowMatrix=1;
     windowfactor='windowM';
    end
    windowfactor
    disp('Planckcase=5 indicates            theta=sigma/TsK*T^5')
    disp('Planckcase=4 indicates SB is used theta=sigma*T^4, DEFAULT')
    disp('Planckcase=3 indicates            theta=sigma*TsK*T^3')
    disp('Planckcase=2 indicates            theta=sigma*TsK^2*T^2')
    disp('Planckcase=1 indicates            theta=sigma*TsK^3*T, LINEAR')
    default=4
     input0=input('Give Planckcase, default click 0:');

     if input0==0
         Planckcase=4
     else
       Planckcase=input0
     end
     if Planckcase<0
        Planckcase=4;
     else
        if Planckcase>5
         Planckcase=4
        end
     end
    disp('')
    disp('ALTERNATIVE  epssurf MENU')
    disp('0:LWepssurf=0.923 according to Miskolczi, default')
    disp('1:LWepssurf=1     IPCC authors')
    disp('2:LWepssurf= SWepssurf=0.878 Miskolczi')
    disp('3:LWepssurf= SWepssurf= 0.848 K&T')
    disp('4:LWepssurf=0.7 to show the effect')
    disp('5:LWepssurf=0.61526, gives OLR=240 for ftot=0')
    disp('6:value to be given')
    default=epssurf
    input0=input('Make your choice between 0 and 6, default click 0:');
    if input0<=0
        input0=0
    else
        if input0>=6
            input0=6
        end
    end
    switch input0
        case  {0}
            epssurf=default;
            givenftot=0.8327;  % to avoid iterations
        case {1}
```

```
            epssurf=1;
            givenftot=0.84822; % to avoid iterations
        case {2}
            epssurf=0.878;
            givenftot=0.8221; % to avoid iterations
        case {3}
            epssurf=0.848;
            givenftot=0.814; % to avoid iterations
        case {4}
            epssurf=0.7;
            givenftot=0.757;    % to avoid iterations
        case {5}
            epssurf=0.61526
            givenftot=0.6984
        case {6}
            epssurf=input('Give a value between 0 and 1 :')
            if epssurf<=0
                epssurf=default;
                givenfot=0.8327; % to avoid iterations
            else
                if epssurf>1
                    epssurf=1;
                    givenftot=0.84622; % to avoid iterations
                end
            end
    end
    epssurf
    givenftot
    ftotco2=givenftot/1000
    ftoth2o=givenftot-ftotco2



      %disribution of watervapor
         disp('Give m parameter for absorber distribution')
        default = m              %m=7 this value is obtained from Miskolczi
data.
        input0=input('for default value click 0: ')

        if input0==0
         m=default;
        else
         m=input0;
         if m<0.1
             m=0;     %homogeneous distribution for demonstration purposes only
         end
        end

      disp('Give atmospheric temperature at surface in Celsius')
      default=TsC
      input0=input('For default value click 0: ')

      if input0==0
         TsC=default
      else
         TsC = input0
      end
      TsK = TsC+273;         % surface temperature in K
      TsK0= TsK;             % initial surface temperature

      disp('Give number of nodes')
      default=nods
      input0=input('For default value click 0: ')
```

```
    if input0==0
       nods=default
    else
       nods = input0          %40 will be sufficient
       if nods<4
           nods=4
       else
           if nods >100
               nods=100
           end
       end
    end

    %height

    disp('Give height of model in km')
    default = heightkm
    input0=input('for default value click 0: ');

    if input0==0
        heightkm=default;
    else
        heightkm=input0
        if heightkm<1
            heightkm=1;
        end
        if heightkm>11
            heightkm=11;
        end
    end
    height= heightkm*1000;



  %lapse rates

  default= ELR;    %ELR is measured environmental lapse rate
  disp('Make a choice for a lapse rate')
  disp('default value ELR= - 6.5 K/km                type: 0')
  disp('Dry adiabatic lapse rate DALR=-9.81 K/km     type: 1')
  disp('Saturated adiabatic lapse rate SALR=-5 K/km type: 2')
  disp('Isothrmal atmosphere  (lapse rate=0          type; 3')

  input0=input('Make a choice between 0 and 3:');

   if input0==0
     LR=ELR;
   else
     if input0==1
      LR= DALR;
     else
     if input0==2
     LR=SALR;
     else
      LR=-0.00001;
     end
     end
    end
  end



dftot=givenftot/5 ;   % defines the number of absorption levels fmin, +dftot,
```

```
+2*dftot,..fmax
fmin=ftotco2;            % for demonstration purposes we start with ftot nearly
zero,
                     % only CO2 like in polar regions, no water vapour



                     %maximum total absorption,for the time being avoid to use
ftot>1


 %define absorption levels between fmin and fmax for which an analysis is
wanted.
 %the last value is ftot=givenftot = 0.849, which is the absorption level which
gave
 %results close to IPCC values, without back-radiation see reference [1]

fmax = givenftot;
ftot = fmin;
jf=1;
fp(1)=fmin;
qtoav(1)=qtoa;
while ftot < fmax
    ftot = ftot + dftot;
    if ftot>fmax
       ftot=fmax;
    end
   jf=jf+1;
   fp(jf)=ftot;
end

nabsorb=jf;           %nabsorb=total number of absorption levels to be analyzed
fp(nabsorb)=givenftot;                        %in analyses (1),(2),
                     %For option (5) zoom and sensitivity analysis nabsorb=2.
if option==4              %input related to option 3, Schwarzschild procedure
    schwarz=2;
end



%mesh generation for options 1, 2, 3, 4 with max height up to 10 km


%if option<5
% mesh generation for the lower atmosphere
% for all options except option  5,
% sensitivity analysis
% taking into account heights up to 30 km.
% z according to geometrical series with r=ratioatm
% r<=1 gives homogeneous distribution

                     %thickness registrated at centre of layer
znod=zeros(nods,1);  %ground level at znod=0
if ratioatm>1
    r=ratioatm;
    dz=(r-1)/(r^(nods-1)-1)*height;
                % increase of dz according to geometrical
                % series with ratio r
else
   dz=height/nods;
   r=1;
end

for i=2:nods
```

```
    dz=dz*r;
    znod(i)=znod(i-1)+dz;
end


for i=1:nods
    znod(i)=znod(i)*height/znod(nods-1);
                        %normalisation in order z(nods-1)=height
    qtoaz(i)=qtoa;      %for plotting only, to draw a reference line.
end
dzv=diff(znod);
for i=1:nods-1
    cor(i)=1/dzv(i);            %correction for plotting of nodal Watt/m^2
                                %in volumetric Watt/m^3
end



%temperature distribution according to lapse rates LR

TLR=zeros(nods,1);
dTLR=zeros(nods,1);
TLR0=zeros(nods,1);
theta=zeros(nods,1);
TDALR=zeros(nods,1);

for i=1:nods-1
    iv(i)=i;                    % for nodal value distribution
    znodi=znod(i);
    TLR(i)=TsK+LR*znod(i);      % temperature distribution  for TsK and LR
    TDALR(i)=TsK+DALR*znod(i); % temperature according to DALR
    theta(i)=Planck(Planckcase)*TLR(i)^Planckcase;   % theta are convenient
parameters for temperature

end


TLR(nods)=TLR(nods-1);
TDALR(nods)=TDALR(nods-1);
theta(nods)=0;
TLR0=TLR;
zeroK=0;                    % outer space taken as zero, can be taken as 2
degree K
theta(nods)=0;              %sigma*zeroK^4;
znod(nods)=znod(nods-1);


%input and analysis preparation related to option 3,diurnal transients
if option==3            %diurnal transients
    nabsorb=2;          %we only want a single K for ftot= givenftot
    schwarz=1;
    ftot=givenftot;
    fp(2)=givenftot;
    fp(1)=givenftot/2;


disp('Give mixing lenght in meters in air boundary')
default=30

input0=input('Mixing lenght in meters, default value click 0: ');

if input0 == 0
    mixlair=default;
else
```

```
    if  input0<0.1
    mixlair=default;
    else
    mixlair=input0;
    end
end

disp('Make your choise between sea and soil surface.')
input0 = input('for sea click 1, for land click  0: ');

if input0 <0
    sea=1
else
    if input0>1
    sea=1
    else
    sea=input0
    end
end



if sea==1
  disp('Give  mixing lenght in meters in sea at atmospheric boundary')
                    %to take into account waves in the afternoon
                    % and   upwelling convection
                    % in the morning
  default=0.5
input0=input('Give mixing length in sea, default value in meters click 0:');

if input0 ==0
    mixlsea=default;
elseif input0<0
    mixlsea=default;
else
    mixlsea=input0;
end

 disp('Give penetration depth in meters of sun SW radiation into sea')
                        %to take into account that SW is absorbed below
                        %the surface
  default=2
input0=input('Give penetration depth in sea, default value in meters click 0:');

if input0 ==0
    pendepth=default;
elseif input0<0
    pendepth=default
else
    pendepth=input0
end
if pendepth>abs(depthsea/2)
    pendepth=abs(depthsea/2);
end
end   %end loop if sea


disp('Which hypothesis for convection?')
if sea ==1
    default=1;
else
    default=3
end
```

```
default
disp(' 1   qconv equal to SS of global and annual mean')
disp(' 2   Proportional to 2-exp(-dTsK/20) and for dTsK<0 to exp(dTsK/20)')
disp(' 3   Proportional to 2-exp(-dTsK) and for dTsK<0 to exp(dTsK)')
disp(' 4   Proportional to sunpower with average global and annual mean')
input0=input('Make your choise 1 to 4, click 0 for default :');

if input0 ==0
    convhyp=default;
elseif input0<0
    convhyp=default;
elseif input0>4
    convhyp=default;
else
    convhyp=input0;
end

disp('Give timeconstant in hours for convection ')
default=1

input0=input(' default click 0,and a negative value for no time delay:');


if input0 == 0
    tauconvh =default;
elseif  input0<0.
    tauconvh=0;

else
    tauconvh=input0;
end
tauconvh
tauconv=tauconvh*3600;
qav=240;
q0=pi*qav;             %maximum sunpower for an average of 240

OLRav=0;

omega=2*pi/3600/24;     % angular speed of earth rad/sec
deltath=0.15;
default=deltath % timestep in hours
deltath =input('Give timestep in hours, default click 0 :')
if deltath<=0
    deltath = default;
end
deltath
deltat=3600*deltath;    % timestep in seconds



if sea==1
    disp('water properties')
  lambda=0.6;        %make lambda in upper layers of sea bigger
                  %to take into account waves and convection in sea

  rhoc=4.2*10^6
  depth=depthsea
else
% sand  properties
    disp('land properties')
  lambda=0.4
  rhoc=1.36*10^6
  depth=depthland
end
```

```
disp(' air properties')
rhocair=1.2e3
rhocairH=10^7                    %=integral from 0 to height(rhocairdz)=p*c/g
lambdaair=0.025

ze=zeros(Nze,1);
ze(1)=0;
r=ratiosea;
dz=abs(depth/(Nze-1));
for i=2:Nze
    ze(i)=ze(i-1)-dz;
    dz=dz*r;
end
zeNze=ze(Nze);
for i=2:Nze                    %normalize to depth.
    ze(i)=ze(i)*depth/zeNze;
end

if sea == 1
    Npen=Nze;
   for i = 2:Nze
    if abs(ze(i))< pendepth
        Npen=i
    end
   end

pendepth=abs(ze(Npen))
fdepth=zeros(Npen,1);
fdepthe=zeros(Npen,1);



for i=1:Npen
    ratio= (abs(ze(i)/pendepth))
    fdepthe(i)=(1-ratio)^4;
end
fdepth=zeros(Npen,1);
for i=1:Npen-1
   fthickness= abs(ze(i)-ze(i+1))*fdepthe(i)/2;
   fdepth(i)=fdepth(i)+fthickness;
   fdepth(i+1)=fdepth(i+1)+fthickness;
end
fdepth(1)=3*fdepth(1)             %if heat goes too far into the sea
                                  %it cannot come out anymore by conduction


sumfdepth=sum(fdepth);
fdepth=fdepth/sumfdepth           % normalized to sum =1

end  %end loop if==sea

disp ('Duration of transient analysis?')
default=3
input0 =input('Give number of days, default click 0 :');


if input0 ==0
    days=default;
elseif input0<0
    days=default;
elseif input0>20
     days=20;
else
```

```
 days=input0;
end
days


timetoth=24*days;
timetot=24*3600*days;  %total time in seconds
Nplot=timetot/deltat; %length of vector for plotting
timep=zeros(Nplot,1);   %time vector for plotting
OLRp=zeros(Nplot,1);    %vector of OLR, outgoing LW radiation
TE1p=zeros(Nplot,1);    %surface temperatures
qsunp=zeros(Nplot,1);   %incoming sun power

TE=ones(Nze,1);         %temperature under surface, soil or sea
TE=TsK*TE;              %initial condition

Tmaxd = zeros(Nze,1);        %extreme maximum sub surface distribution
Tmind = zeros(Nze,1);        %extreme minimum sub surface distribution
TLRmaxd=zeros(nods,1);       %extreme maximum air distribution
TLRmind=zeros(nods,1);       %extreme minimum air distribution


Tday=zeros(Nze,1);           %daily surface temperature distribution of
                             %air and soil (or sea)

ME =zeros(Ntot);       %heat capacity matrix of sub-surface and air
KE=zeros(Ntot);        %conductivity matrix of sub-surface and air
KWC=zeros(Ntot);        %conductivity matrix correction of sub-surface
                        % sea for convection of warmer water to surface.

MED=zeros(Ntot);       % space-time equivalent heat capacity matrix
                       % of sub-surface and air


znod(nods)=znod(nods-1)+10;  % znod(nods) not in outer space but near the
                             %the node = nods-1. Not equal to it which would
                             %give an infinitely term in KE
   % we assemble the matrix from the node=1 down in
   % the depth of land or sea up to the top node
   % of the atmosphere: Ntot = nods+Nze-1, i=Ntot  to i=2.
    % The corresponding unknown is TX a vector containing both sub-surface
    % temperatures TE and air temperatures TLR in such a way that the
    % surface temperature TE(1) coincides with the surface air temparture TLR(1)
 %The structure of TX transposed becomes avector of (nods + Nze-1)  variables
 %TX' = [ TLR(nods).....TLR(1)=TE(1).....TE(Nze)]

 % definittion of the corresponding zx cordinates

 zx=zeros(Ntot,1);       %coordinates in FEM model, atmosphere plus soil or sea
 for i=nods:Ntot         %subsurface nodes
     zx(i)=ze(i-nods+1);
 end
 for i=1:nods-1             % zx atmospheric nodes in FEM, from TOA down to
surface
     zx(i)=znod(nods-i+1); % znod atmospheric nodes in radiation model,from
surface up to TOA
 end


  for i=1:Ntot-1
    if i<nods     %air
        k0=abs(lambdaair/(zx(i+1)-zx(i)));
           if zx(i)<mixlair/4    %boundary layer of air
              k=25*k0;          %artificial high conductivity near surface
```

```
            elseif zx(i)<mixlair/2
                k=20*k0;
            elseif zx(i)<mixlair
                k=10*k0;
            else
                k=k0;
            end
          me=abs(rhocair*(zx(i+1)-zx(i)))/6;
      end
      if i>nods-1        %sea or soil
        k0=abs(lambda/(zx(i+1)-zx(i)));
        if sea == 1
            if zx(i)>-mixlsea/4
              k=25*k0;     %surface waves of sea simulated by bigger conductivity
            elseif zx(i)>-mixlsea/2
              k=20*k0;
            elseif zx(i)>-mixlsea
             k=10*k0;
            else
                k=k0;
            end
        KWC(i,i)=KWC(i,i)+k;                 %conductivity matrix correction for
                                                  %convection of warmer
                                                  %water to surface
        KWC(i+1,i)=KWC(i+1,i)-k;
        KWC(i,i+1)=KWC(i,i+1)-k;
        KWC(i+1,i+1)=KWC(i+1,i+1)+k;
        end


        me=abs(rhoc*(zx(i+1)-zx(i)))/6;
      end


      KE(i,i)=KE(i,i)+k;                    %conductivity matrix
      KE(i+1,i)=KE(i+1,i)-k;
      KE(i,i+1)=KE(i,i+1)-k;
      KE(i+1,i+1)=KE(i+1,i+1)+k;

      ME(i,i)=ME(i,i)+2*me;                 %heat capacity matrix
      ME(i+1,i)=ME(i+1,i)+me;
      ME(i,i+1)=ME(i,i+1)+me;
      ME(i+1,i+1)=ME(i+1,i+1)+2*me;
      end

    MED=ME+2/3*deltat*KE;       %Space time finite elements dynamic heat capacity
matrix
    invMED=inv(MED);            % with the inverse,

   %insert TLR and TE in TX as initial conditions
      TX =zeros(Ntot,1);
      TX0 =zeros(Ntot,1);
      TXmaxd =zeros(Ntot,1);    % to record the extremes, maximum distribution
      TXmind =zeros(Ntot,1);    % and mimimum distribution

      for i=1:Ntot
      j=nods-i+1;
       if i<nods+1               %atmosphere according to lapse rate
          TX0(j)=TLR(i);
       else
          TX0(i)=TsK0;  %subsurface temperatures for soil and sea
       end
      end
      TX=TX0;
```

```
    znod(nods)=znod(nods-1);     %make znod(nods) equal to znod(nods-1)
              %end of option 4, preparation for transient analyses

end   %end option==3

%preparation of analyses for options 1,2, 3,4



f=zeros(nods,1);
f(1)=epssurf;    %emission of the surface , K&T diagrams fot 0.96
f(nods)=1;       %outer space f=1.


% distribution of absorption at nods = znod(i)
% exponential decay of absorption with height defined by m
% for m=0 homogeneous distribution, m=7 gave the best comparison with the
% experimental results of the K&T diagrams see ref [1]


for i=2:nods-1
    znodi=znod(i);
    ratio=exp(-m*znodi/5000);
    fdH2O(i)=dzv(i-1)*ratio;
    ratioCO2=(TsK/(TsK+LR*znodi))^(g/R/LR+1);
    fdCO2(i)=dzv(i-1)*ratioCO2;
end
fdH2O(1)=0;                        % at node =1 (surface) and at node =nods
(outerspace)
                                    % fd=0 for both H2O and CO2
fdCO2(1)=0;
fdH2O(nods)=0;
fdCO2(nods)=0;

%normalized distribution of absorption, total normalized = 1
nods
totfdH2O=sum(fdH2O);
fdH2O=fdH2O/totfdH2O;
totfdCO2=sum(fdCO2);
fdCO2=fdCO2/totfdCO2;
end    %end loop option<5


%input data and preparation for sensitivity analysis
%for CO2 contribution maximun height is taken as 30km.

if option ==5                % sensitivity analysis lower atmosphere with 40
nodes
                             % and 20 other nodes higher up

    nabsorb=3;               %jf=1 only CO2 jf=nabsorb-1 and jf=nabsorb only H20
    nods=40;                 %number of nods in troposfere
     disp('windowMatrix=1 means the complete viewfactorF MATRIX is used ')
     disp('windowMatrix=0 means only the windowF VECTOR is used')
     windowMatrix= input('Make a choice for windowMatrix, 1 or 0 :')

                           % if windowMatrix==0 then viewfactor=ones(nods) except
last
                           % column which becomes windowF see {1} or
                           % appendix 4 of ref [3]
                           % if windowMatrix ==1 then full viewfactorF matrix,
see ref 2 and
                           % ref 3
                           % Note of june 2015: comparison with two-stream
```

```
                        % formulations, with the huge "back-radiation" it
                        % seems that IPCC authors as well as Miskolsci
                        % have used the version windowMatrix=0. But they
                        % do not give explicitly the window flux.
                        % For that reason the default value is now taken
                        % as windowMatrix=0.
if windowMatrix<=0
 windowMatrix='windowF';
end
if windowMatrix>=1
 windowfactor='windowM';
end
windowfactor
disp('ALTERNATIVE epssurf MENU')
disp('0:LWepssurf=0.923 according to Miskolczi, default')
disp('1:LWepssurf=1     IPCC authors')
disp('2:LWepssurf= SWepssurf=0.878 Miskolczi')
disp('3:LWepssurf= SWepssurf= 0.848 K&T')
disp('4:LWepssurf=0.7 to show the effect')
disp('5:LWepssurf=0.61526, gives OLR=240 for ftot=0')
disp('6:value to be given')
default=epssurf
input0=input('Make your choice between 0 and 4, default click 0:');
if input0<0
    input0=0
else
    if input0>6
        input0=6
    end
end
 switch input0
    case  {0}
        epssurf=default;
        givenftot=0.8327;  % to avoid iterations
    case {1}
        epssurf=1;
        givenftot=0.84822; % to avoid iterations
    case {2}
        epssurf=0.878;
        givenftot=0.8221; % to avoid iterations
    case {3}
        epssurf=0.848;
        givenftot=0.814; % to avoid iterations
    case {4}
        epssurf=0.7
        givenftot=0.7    % to avoid iterations
    case {5}
         epssurf=0.61526
         givenftot=0.6984
    case {6}
        epssurf=input('Give a value between 0 and 1: ');
        if epssurf<=0
            epssurf=default;
            givenfot=0.8327; % to avoid iterations
        else
            if epssurf>1
                epssurf=1;
                givenftot=0.84622; % to avoid iterations
            end
        end
end
 epssurf
 givenftot
 ftotco2=givenftot/1000
```

```
 ftoth20=givenftot-ftotco2


ratio=0.001;
default = ratio
ratio=input('Give ratio ftotco2/givenftot, default click 0:');
if ratio<=0
    ratio=default
else
   if ratio>0.5
    ratio=0.5
   end
end
epssurf
givenftot
ftotco2=givenftot*ratio
ftoth2o=givenftot-ftotco2;
fp(1)= ftotco2;
fp(2)= givenftot/2;
fp(3) =givenftot;
dftotco2=ftotco2;            % indeed a doubling of CO2 concentration

disp( 'Make a choice between different climate zones')
disp ( ' 0: standard, see reference ')
disp ( ' 1: tropical')
disp ( ' 2: polar')
climatezone=input('Give climate zone, default click 0: ')
if climatezone <=0
    climatezone=0;
else
    if climatezone>2
        climatezone=0;
    end
end
climatezone
nods2=nods;
nods3=nods2+10;
nods4=nods3+10;
switch climatezone                  %standard
    case {0}
        znod2=11500;
        znod3=20000;
    case {1}                        %tropical
        znod2=16500;
        znod3=22000;
    case {2}                        %polar
        znod2= 9000;
        znod3=26000;
   end
znod4=30000
% mesh generation for the lower atmosphere
h2o=zeros(nods4,1);
co2=zeros(nods4,1);

                  %thickness registrated at centre of layer
znod=zeros(nods4,1);  %ground level at znod=0
height=znod2;
if ratioatm>1
 r=ratioatm;
 dz=(r-1)/(r^(nods-1)-1)*height;
              % increase of dz according to geometrical
              % series with ratio r
else
   dz=height/nods;
```

```
   r=1;
end
%lower atmosphere
znod(1)=0;
dz=dz/r;
for i=2:nods
    dz=dz*r;
    znod(i)=znod(i-1)+dz;
end
for i=1:nods
    znod(i)=znod(i)*height/znod(nods);
                        %normalisation in order z(nods-1)=height
    qtoaz(i)=qtoa;      %for plotting only, to draw a reference line.
end

TLR=zeros(nods4,1);  %temperature distribution according to laps rates LR in
lower atmosphere
theta=zeros(nods4,1);



ELR1=ELR;


 switch climatezone
        case {0}                              %standard
          TsK=288;
          TsK0=TsK;
          Tnod2=TsK+ELR1*znod2;
          Tnod3=Tnod2+0.1; %otherwise ELR2 will be zero
          Tnod4=230;
        case {1}                             %tropical
          TsK=305;
          TsK0=TsK0;
          Tnod2=TsK+ELR1*znod2;
          Tnod3=215;
          Tnod4=240.
        case {2}                             %polair
          TsK=270    %no inversion
          TsK0=TsK0
          Tnod2=TsK+ELR1*znod2;
          Tnod3=210
          Tnod4=210.1
 end
 for i=1:nods
    iv(i)=i;                    % for nodal value distribution
  znodi=znod(i);
  TLR(i)=TsK+ELR*znodi;         % temperature distribution  for TsK and LR
 end

    ELR1=ELR;
    ELR2=(Tnod3-Tnod2)/(znod3-znod2);
    ELR3=(Tnod4-Tnod3)/(znod4-znod3);
    zeroK=0;                      % outer space taken as zero, can be taken as 2
degree K

    h2o(1)=1;
    co2(1)=1;
    dTLR(1)=1;

for i=2:nods2
    znodi=znod(i);
    ratioH2O=exp(-m*znodi/5000);
    h2o(i)=ratioH2O;
```

```
    dzv(i-1)=znodi-znod(i-1);
    fdH2O(i)=dzv(i-1)*ratioH2O;
    Tznodi=TsK+ELR1*znodi;
    ratioCO2=(Tznodi/TsK)^-(g/R/ELR1+1);
    co2(i)=ratioCO2;
    fdCO2(i)=dzv(i-1)*ratioCO2;
    TLR(i)=Tznodi;
    dTLR(i)=1;
end
ratioCO2old=ratioCO2;
%temperatures between nods2 and nod3
dz=(znod3-znod2)/(nods3-nods2);
for i=nods2+1:nods3
    znod(i)=znod(i-1)+dz;
    znodi=znod(i);
    ratioH2O=exp(-m*znodi/5000);
    h2o(i)=ratioH2O;
    dzv(i-1)=dz;
    fdH2O(i)=dzv(i-1)*ratioH2O ;
    Tznodi=Tnod2+ELR2*(znodi-znod2);
    ratioCO2=ratioCO2old*(Tznodi/Tnod2)^-(g/R/ELR2+1);
    co2(i)=ratioCO2;
    fdCO2(i)=dzv(i-1)*ratioCO2;
    TLR(i)=Tznodi;
    dTLR(i)=0;
end
ratioCO2old=ratioCO2;

%temperatures between nods3 and nods4
dz=(znod4-znod3)/(nods4-nods3);
for i=nods3+1:nods4
    znod(i)=znod(i-1)+dz;
    znodi=znod(i);
    ratioH2O=exp(-m*znodi/5000);
    h2o(i)=ratioH2O;
    dzv(i-1)=dz;
    fdH2O(i)=dzv(i-1)*ratioH2O;
    Tznodi=Tnod3+ELR3*(znodi-znod3);
    ratioCO2=ratioCO2old*(Tznodi/Tnod3)^-(g/R/ELR3+1);
    co2(i)=ratioCO2;
    fdCO2(i)=dzv(i-1)*ratioCO2;
    TLR(i)=Tznodi;
    dTLR(i)=0;
end
ratioCO2old=ratioCO2;


nods=nods4;
m
nods
height=znod4

fdH2O(1)=0;                         % at node =1 (surface) and at node =nods
(outerspace)
                                     % fd=0 for both H2O and CO2
fdCO2(1)=0;
fdH2O(nods)=0;
fdCO2(nods)=0;
totfdH2O=sum(fdH2O);
fdH2O=fdH2O/totfdH2O;
totfdCO2=sum(fdCO2);
fdCO2=fdCO2/totfdCO2;

end   %end loop if option==5
```

```
%input for all options done

f(1)=epssurf;              % surface emission , one can play with it by changing
                           % epssurf.
                           % In ref [1] a study is made of different values
                           % and epssurf = 0.96 was retained. In polar
                           % regions eps is lower, uses are invited to play
                           % with the parameter

qsurf=zeros(nabsorb,1);    %LW surface flux, LW absorption from surface into
atmosphere,
qwindow=zeros(nabsorb,1);  %it includes qabsorb and qwindow
OLR=zeros(nabsorb,1);      %Outgoing longwave radiation
qintotot=zeros(nabsorb,1); % absorption in atmosphere, one or more times
qouttot =zeros(nabsorb,1); %emitted LW radiation by atmosphere
                            % qouttot-qintotot: heat by convection and
                            % SW emission into atmosphere
qtoav = zeros(nabsorb,1);   %make a vector of qtoa for plotting purposes
qintov=zeros(nabsorb,nods);
qoutv=zeros(nabsorb,nods);
qzv = zeros(nabsorb,nods);
qv  = zeros(nabsorb,nods);
backrad = zeros(nabsorb,1);

dzv=diff(znod);
fdH2O(1)=0;                        % at node =1 (surface) and at node =nods
(outerspace)
                                    % fd=0 for both H2O and CO2
fdCO2(1)=0;
fdH2O(nods)=0;
fdCO2(nods)=0;

%normalized distribution of absorption, total normalized = 1


totfdH2O=sum(fdH2O);
fdH2O=fdH2O/totfdH2O;
totfdCO2=sum(fdCO2);
fdCO2=fdCO2/totfdCO2;
cor=zeros(nods-1,1);
for i=1:nods-1
    cor(i)=1/dzv(i);           %correction for plotting of nodal Watt/m^2
                               %in volumetric Watt/m^3
    theta(i)=Planck(Planckcase)*TLR(i)^Planckcase;       %sigma*T^4
end
knodsco2 = zeros(nods,1);
knodsh2o = zeros(nods,1);
knods = zeros(nods,1);     % two different OLR are calculated for two CO2
concentrations
K=zeros(nods);
KS=zeros(nods);

% loop through the absorption levels from jf=1 to nabsorb for
%the different options.

option
nabsorb
ftotco2
fp(nabsorb)=givenftot;
jf=0;

while jf<nabsorb               % loop through absorption levels
qN=0;
```

```
qNz=0;                                  % loop N-1
  jf=jf+1;
  qtoav(jf)=qtoa;                 %for plotting a horizontal line=qtoa

  f=zeros(nods,1);
  f(1)=0;                          % wil be put to epssurf later
    for i=2:nods-1
      if option == 3                       %option=3 transient analysis
          f(i)=fdH2O(i)*fp(jf);
      else
                     %for sensitivity analysis, treat CO2 separately at

        if jf==1
         f(i)=fdCO2(i)*ftotco2;      %only CO2 for jf=1

        else
        f(i)=fdH2O(i)*(fp(jf));
        end
      end
    end

 f(1)=epssurf;        % surface emission , one can play with it by changing
epssurf.
                     % in ref [1] a study is made of different values
                     % epssurf is important, the lower epssurf the less
                     % influence have the traces of IR-active gases.

  f(nods)=1;         % outer space f(nods)=1
  qwindow(jf)= (1-fp(jf))*epssurf*theta(1);
  if qwindow(jf)<0
      qwindow(jf)=0;
  end


       viewfactorF=ones(nods);
       viewfactorS=ones(nods);
       windowF= ones(nods,1);
       windowS= ones(nods,1);
      for i=1:nods-1
          windowFi= 1;
          windowSi=1;

       for j=i+1:nods
           windowSi=windowSi-f(j);
           windowFi=windowFi-f(j);
           viewfactorFij=viewfactorF(i,j);
           viewfactorSij=viewfactorS(i,j);

            for k=i+1:j-1
               viewfactorFij=viewfactorFij-f(k);
                             % FEM viewfactor(i,j)=1-f(i+1)-f(i+2))...-f(j-1)
               viewfactorSij=viewfactorSij*(1-f(k));
                          % Schwarzschild
                        % viewfactor(i,j)=(1-f(i+1))*(1-f(i+2))...*(1-f(j-1))

            end

            if viewfactorFij<0
              viewfactorFij=0;
            end
            viewfactorF(i,j)= viewfactorFij;
            viewfactorF(j,i)=0;
            if viewfactorSij<0
              viewfactorSij=0;
```

```
            end
            viewfactorS(i,j)=viewfactorSij;
            viewfactorS(j,i)=0;
        end
        windowF(i)=windowFi+1;  %windowF was uses in finite difference version[1]
        windowS(i)=windowSi+1;

      end



      viewfactor = viewfactorF + viewfactorF';
      viewfactorF= viewfactor;
      viewfactor = viewfactorS+viewfactorS';
      viewfactorS= viewfactor;

      for i=1:nods
          viewfactorF(i,i)=1;
          viewfactorS(i,i)=1;
      end
       windowF=viewfactorF(:,nods);
       windowS=viewfactorS(:,nods);
    for schw =1:schwarz    %in case schwarz=2 (option 4) we make two system
matrices
                             % K for the viewfactorsF with windows finite
elements
                             % KS for the viewfactorS of Schwarzschild concept
      OUT = zeros(nods);
     INTO = zeros(nods);



      if schw==1
          if windowMatrix==0                %this option is introduced to show
                                            %the influence of windowF as
                                            %was used in [1], based on a
                                            %finite difference
                                            %implementation

              viewfactorF=ones(nods);
              viewfactorF(:,nods)=windowF;
          end
      end


    % assemblage of system matrix
   if schw==1
       viewfactor=viewfactorF;
   else
       viewfactor=viewfactorS;
   end
   backradjf=0;
   qsurfPrevostjf=0;
   olravjf=0;

    for i=1:nods-1
      fi=f(i) ;
      for j=i+1:nods

        fe = fi*viewfactor(i,j)*f(j);

        OUT(i,i)=  OUT(i,i)+fe;          %assemble OUT and INTO
        OUT(i,j)=  OUT(i,j)-fe;
        INTO(j,i)= INTO(j,i)-fe;
```

```
        INTO(j,j)= INTO(j,j)+fe;
        if i==1
            backradjf=backradjf+fe*theta(j);
                                % to show that back-radiation is just
                                % the sum of the Prevost element
                                % flux of pairs for which the lower
                                % node is the surface node
        end
      end
    end


   if schw == 1
    backrad(jf)=backradjf;    %store it for plotting
    K=INTO+OUT;

       % for schwarz=1 the naked  system matrix K without boundary conditions,
       % to be used for analyses option 2,3
       % for option=4 two system matrices are made, the classical K for the
       % stack model with FEM viewfactors including windows to outer space
       % and a second if one KS with viewfactors according to the
       % Schwarzschild concept (schw=2)

   else
   backradSvf(jf)=backradjf;     % store it for plotting
   KS=INTO+OUT;    %system matrix with viewfactors according to Schwarzschild

   end
 end        %radiation matrices are assembled from OUT and INTO

 % system matrices have been assembled,
 % now analyses for the different options are carried out

 if option ==1
       % For analyses of a stack in vacuum, so to speak on the moon(option 1)
       % boundary conditions for the surface nod=1: theta(1) and at
       % outer-space nods boundary condition
       % theta(nods)=zeroK by means of lagrangian multipliers lambda
       % at position nods+1 and nods+2
       % It are additional unknowns
       % The additional unknown x(nods+1)=lambda1
       % corresponds to the negative ofthe surface flux, -qs.
       % The additional unknown x(nods+2)=lambda2 corresponds to
       % outgoing longwave radiation, OLR.

%  AUGMENTED MATRIX KL
%  For "nods" unknown theta and 2 prescribed surface
%  theta01=theta1 and outer space thetaN = zeroK,
%  by means of Lagrange Multipliers lambda1 and lambda2
%  dimension of KL (nods+2,nods+2)

%          <-----nods+2------>
%   1    / . . . . . .    . 1 0//  theta1 / = /  rhs1     /
%   2    / . . . .        . 0 0//  theta2 /   /  rhs2    /
%   3    / .   .          . 0 0//  theta3 /   /  rhs3    /
%   .    / .     .        . 0 0//   .     /   /    .     /
%   .    / .       .      . 0 0//   .     /   /    .     /
%   .    / .         .    . 0 0//   .     /   /    .     /
%   .    / .           . . 0 0//   .     /   /    .     /
% nods   / . . . . . . . . 0 1//  thetaN /   /  rhsN    /
% nods+1 / 1 0 0 0 0 0 0 0 0 0// lambda1 /   /  theta1 /
% nods+2 / 0 0 0 0 0 0 0 1 0 0// lambda2 /   /  thetaN /

  %introduction of Lagrange multipliers.
```

```
  LAG=zeros(nods,2);
  LAG(1,1)=1;
  LAG(nods,2)=1;
  KL= [K,LAG;LAG',zeros(2,2)]; % KL matrix of order (nods+2)x(nods+2)
  invKL=inv(KL);
  theta1=theta(1);
  thetaN=zeroK;
          %serve as BC at node 1
          %at outerspace node= nods: theta(nods)= zeroK
  rhs=zeros(nods,1);
  temp=zeros(nods,1);
  rhsL=zeros(nods+2,1);
  rhsL=[rhs;theta1;thetaN];
             % right hand side including prescribed values theta1
             % respectively zeroK at node = nods at outer space thetaN
  x=invKL*rhsL;
  for i=1:nods
      theta(i)=x(i);
            % the first result x( 1 to nods) represent theta
            % which can be converted to temperature
            % in degree K including outer space = zero K,
            %imposed by BC through Lagrange multipliers
      temp(i)=(theta(i)/Planck(Planckcase))^(1/Planckcase);
  end

  temp(nods)=temp(nods-1);  %plotting purpose only,to avoid temp(nods) = zeroK.
                            %znod(nods) is not at infinity but at TOA
  qsurf(jf)= -x(nods+1);    %lambda1 = -surface flux
  OLR(jf) =  x(nods+2);     %lambda2 = OLR outgoing longwave radiation
  tempv(jf,:)= temp;        %preparation of plots for temperatures for
                            %a stack in vacuum due to ground flux,
                            %analysis option 1. It is inside the loop
                            %jf=1:nabsorb in order to make comparisons
                            %for different absorption levels.

  q=K*theta;
  qout = OUT*theta;   %vector of fluxes emitted by the nodes, nodal values
  qinto=INTO*theta;     %vector of fluxes absorbed by the nodes, nodal values
  qinto(nods)=0;          %eliminate the nodal value at nods , outer space
  qout(nods)=0;
  qintotot(jf)=sum(qinto);  % make the sum of heat absorbed and
  qouttot(jf)=sum(qout);    % the sum of heat emitted
  backradjf=0;
  for i=2:nods
      backradjf=backradjf-K(1,i)*theta(i);  %it is not the backradiation of heat
                                            %but only the second term in the
                                            %SB relation
  end
  backrad(jf)=backradjf;
  thetanods=theta(nods);
  theta(nods)=theta(nods-1);
  thetav(jf,:)= theta;
  theta(nods)=thetanods;
 end                        %end of Lagrangian multiplier option 1, actual
plotting outside nabsorb loop

 qN=0;
 qNz=0;
 for i=1:nods
 qN=qN+K(i,nods)*theta(i);
 qNz=qNz+K(i,nods)*theta(i)*znod(i);
 end
```

```
    %start of analyses option 2, 3,4,5  using the method of inverse analysis.
    %absorption and emission for temperature distribution with imposed lapse
rate
    %This is the most important result of this computer program
    %emission and absorption for temperature distribution with imposed lapse
rate

    theta(nods)=zeroK;  %outer space temperatrure at zeroK, just to be sure  in
a
                        %previous plotting option the value is set to
                        %theta(nods-1) for plotting reasons
    q=K*theta;       % q is includes contribution from mechanisms other than
radiation
                      %except first term=qsurf and last term=-OLR

    qsurf(jf)=q(1);     %outgoing radiation due to LW surface flux,
                        %including flux through window: qwindow(jf)
    OLR(jf)=-q(nods);   % total outgoing including mechanisms
                        %other than surface radiation
    qabsorb(jf)=qsurf(jf)-qwindow(jf);
    qout = OUT*theta;  %LW radiation from atmosphere
    qinto=INTO*theta;  %LW radiation absorbed by atmosphere, one or more times
    qinto(nods)=0;
    qout(nods)=0;
    qintotot(jf)=sum(qinto);
    qouttot(jf)=sum(qout);

        %correct for plotting in a non-homogeneous mesh, qinto and qout are
        %nodal values, W/m^2, to be converted to volumetric values W/m^3.
        %The difference is due to mechanisms other than LW radiation

    qz =zeros(nods,1);
    qz(1)=q(1);
    for i=2:nods-1
        qz(i)=qz(i-1)+q(i);  % total flux upwards
        q(i)=q(i)*cor(i-1); % convert the nodal values q, qinto and qout from
W/m^2
                            % towards W?m^3 to give a distribution
        qinto(i) = qinto(i)*cor(i-1);
        qout(i) = qout(i)*cor(i-1);
    end
    qout(1)=qout(2);    %for plotting purposes make value at staion 1 equal to 2
    qinto(1)=qinto(2);
    qintov(jf,:)=qinto; %assemble in a matrix for plotting as function of
                        %the height for the different absorption levels ftot
    qoutv(jf,:)=qout;

    %disregard surface flux in plotting

    q(1)=q(2);
    q(nods)=q(nods-1);
    qz(nods)=qz(nods-1);

              % plotting preparation for analysis for  analysis  option 2
              % emitted flux by atmosphere with imposed lapse raet
    qv(jf,:)=q;         % assemble in a matrix to plot distributions of heat
                        % deposit mechanisms other than LW radiation
                        % due to imposed lapserate
    qzv(jf,:)= qz;    %z-distribution of LW for different absorbtions ftot
```

```
   q = K*theta;
   OLR(jf) = -q(nods);
   qsurf(jf) = q(1);
   qwindowjf=qwindow(jf);
   qabsorb(jf)= q(1)-qwindowjf;



  %Schwarzchild procedure on the stack.

   if option ==4

     theta(nods)=zeroK;
     f(nods)=1;      %in case that for plotting in other
                 %options f(nods)is changed

  % FEM with Schwarzschild viewfactors. Results *Svf

 q = KS*theta;

 OLRSvf(jf) = -q(nods);
 qsurfSvf(jf) = q(1);
 qabsorbSvf(jf)= q(1)-qwindow(jf);
 nodsm1=nods-1;



%Schwarzschild original analytical solution with
%integrals to be determined numerically

fpjf=fp(jf);
nodsm1=nods-1;
tau=zeros(nodsm1,1);         %optical thickness tau(1)=ftot
tau(1)=fpjf;                 %optical thickness per wavelenght
tau1=tau(1);
emtau1=exp(-tau1);
Prevost=epssurf*theta(1);    % Prevost surface flux
OLRjf=Prevost*emtau1;
backradjf=0;
qsurfjf=OLRjf;
for i=2:nodsm1
    fi=f(i);
    tau(i)=tau(i-1)-fi;
    OLRjf=OLRjf + fi*theta(i)*exp(-tau(i));
    backradjf= backradjf + fi*theta(i)*exp(-(tau(2)-tau(i)));
    qsurfjf=OLRjf-backradjf;
end
backradS0(jf)=backradjf;
OLRS0(jf)=OLRjf;
qsurfS0(jf)=qsurfjf;
for modif=1:3
    if modif==1
       OLRjf =(Prevost-backradjf)*emtau1;
    end
    if modif==2
       OLRjf =(Prevost-qwindowjf)*emtau1;
    end
    if modif==3
      OLRjf =(Prevost-backradjf - qwindowjf)*emtau1;
    end
                %modification of Schwarzschild by JR
                %alternative starting value for upward flux.
                %Prevost-backrad-qwindow
```

```
                %for modification 1 qwindowjf=0
                %for modification 2 qwindowjf=qwindow(jf)
                %for modification =3 both
    for i=2:nodsm1
       fi=f(i);
       tau(i)=tau(i-1)-fi;
       OLRjf=OLRjf + fi*theta(i)*exp(-tau(i));
    end
    if modif==1
      OLRS0MBR(jf)=OLRjf;
    end
    if modif==2
      OLRS0MW(jf)=OLRjf +qwindowjf;
    end
    if modif ==3
        OLRS0M(jf)=OLRjf +qwindowjf;
    end


end
 %add the parallel flux through atmospheric window

 %forward stepping procedure to solve Schwarzschild equation for a stack
 %Start with backward radiation in order to use the result as BC for the
 %upgoing flux in the modified procedure

 DS=zeros(nodsm1,1);            % nods is at outerspace at zeroK
 DS(nodsm1) =0;                 % incoming downward LW is zero .
 for i=nodsm1-1:-1:2            % standard Schwarzschild equation
    fi=f(i);
    DS(i) = DS(i+1)*(1-fi)+fi*theta(i);
                    % dDS = -beta*DS*(-dz)+beta*theta*(-dz)
                    % f=beta*dz
 end

 DS(1)=DS(2);
 backradjf = DS(1);
 backradS1(jf)= backradjf;
 qwindowjf=qwindow(jf);
 Prevost=epssurf*theta(1);    %standard Schwarzschild starting value for
                              %upward flux=Prevost flux
 US=zeros(nods-1,1);
 US(1)= Prevost;          %standard BC for Schwarzschild

 for i=2:nodsm1
    fi=f(i);
    US(i) = US(i-1)*(1-fi) +fi*theta(i);
                         % standard Schwarzschild equation
                         % dUS = -beta*US*dz+beta*theta*dz
                         % f=beta*dz
 end

 OLRS1(jf)=US(nodsm1);                %Outgoing LW flux according to
Schwarzschild
 qsurfS1(jf)=Prevost-backradjf;       %surface flux according to Schwarzschild
 US(1)= Prevost-backradjf-qwindowjf;  %modified Schwarzschild solution
                                      % US(1) = Prevost -backradjf -qwindowjf
 for i=2:nodsm1
    fi=f(i);
    US(i) = US(i-1)*(1-fi) +fi*theta(i);
                    %this is the standard Schwarzschild equation
                    % dUS = -beta*US*dz+beta*theta*dz
                    % f=beta*dz
 end
```

```
 OLRS1M(jf)=US(nodsm1)+qwindowjf; %add qwindow to OLR

   % modified Schwarzschild formulation with an implicit solution based
   % on least square variational method


   %1 upward radiation
    A1=zeros(nods-1);      %reserve space for matrices A1,B1
    B1=zeros(nods-1);
    thetam1=zeros(nods-1,1);
    rhs = zeros(nods-1,1);
                  %because f(1) belongs to the surface not to the atmosphere
    for i=1:nods-2
      thetam1(i)=theta(i);
      fi=f(i+1);
      A1(i,i)=A1(i,i)+1-fi+fi^2/3;
      A1(i,i+1) =A1(i,i+1) -1+fi^2/6;
      A1(i+1,i)=A1(i+1,i)-1+fi^2/6;
      A1(i+1,i+1)=A1(i+1,i+1)+1+fi+fi^2/3;

      B1(i,i)=B1(i,i)-fi/2+fi^2/3;
      B1(i,i+1) =B1(i,i+1) -fi/2 +fi^2/6;
      B1(i+1,i)=B1(i+1,i)+fi/2+fi^2/6;
      B1(i+1,i+1)=B1(i+1,i+1)+fi/2+fi^2/3;
    end
    thetam1(nods-1)=theta(nods-1);

    big=10^10;
    A1(1,1)=big;        %BC for upward at node 1, make diagonal big
    invA1=inv(A1);


    %2 downward radiation
    A2=zeros(nods-1);   %reserve space for A2 and B2
    B2=zeros(nods-1);
                        %because f(1) belongs to the surface not to the
atmosphere
    for i=1:nods-2

      fi=f(i+1);        % for the implicit solution we take f as average of the
nodal values.
      A2(i,i)=A2(i,i)+1+fi+fi^2/3;
      A2(i,i+1) =A2(i,i+1) -1+fi^2/6;
      A2(i+1,i)=A2(i+1,i)-1+fi^2/6;
      A2(i+1,i+1)=A2(i+1,i+1)+1-fi+fi^2/3;

      B2(i,i)=B2(i,i)+fi/2+fi^2/3;
      B2(i,i+1) =B2(i,i+1) +fi/2 +fi^2/6;
      B2(i+1,i)=B2(i+1,i)-fi/2+fi^2/6;
      B2(i+1,i+1)=B2(i+1,i+1)-fi/2+fi^2/3;
    end

      A2(nods-1,nods-1)=big;  %BC for downward is at node nods-1,
                              %make diagonal big
      invA2=inv(A2);
      rhs=B2*thetam1;
      rhs(nods-1)=0;              % boundary condition for back-radiation
      DS=invA2*rhs;
      backradjf = DS(1);
      backradS2(jf)= backradjf;  % ready with back radiation,

      rhs=B1*thetam1;
      Prevost=epssurf*thetam1(1); % standard Schwarzschild with US(1)=Prevost
      US=zeros(nods-1,1);
```

```
      rhs(1)=big*(Prevost);
      US=invA1*rhs ;
      OLRS2(jf)=US(nodsm1);      % Outgoing LW flux according to Schwarzschild
      qsurfS2(jf)=US(1);
      rhs(1)=big*(Prevost- backradjf-qwindowjf);
                          % modified Schwarzschild Prevost-backradjf- qwindowjf
      US=invA1*rhs ;
      OLRS2M(jf)=US(nodsm1)+qwindowjf;
                          % add qwindow to OLRS2M
   end %end of option==4 Schwarzschild
   qNv(jf)=-qN;
   zEmission(jf)=qNz/qN;
   if option==5
   if jf==1
       knodsco2=K(:,nods);   %last row of K for jf=1
                             %only CO2
       k1co2=K(:,1);         %first row of K for jf=1
                             %only CO2
       viewfactorFco2=viewfactorF;
       viewfactorSco2=viewfactorS;
       windowFco2=windowF;
       windowSco2=windowS;
   end
   end
 if jf==nabsorb-1
     OLRpr=OLR(jf);
     fppr=fp(jf);
 end
 if jf<nabsorb
     iter=0;
 end
 if jf==nabsorb
    disp('Iterations to obtain OLR=qtoa ')
    iter=iter+1
    OLRjf=OLR(jf)
    mismatch=OLRjf-qtoa;
    if abs(mismatch/qtoa)>0.001
    mismatch =mismatch/(OLRjf-OLRpr);
    OLRpr=OLRjf;
    fpjf=fp(jf);
    dfpfj= -(fpjf-fppr)*mismatch;
    fppr=fpjf;
    fp(nabsorb)=fpjf+dfpfj;
    jf=jf-1;
    else
        jf=nabsorb;
    end
   if iter>9
       disp('More than 10 iterations for qtoa')
       jf=nabsorb;
   end
  end
 maxftot=fp(nabsorb);
end
 maxftot
 maxftot= round(maxftot*10000)/10000;
 fp(nabsorb)=maxftot;
 ftoth2o=maxftot-ftotco2
 knodsh2o = K(:,nods);         %last row of K for jf=nabsorb
 k1h2o = K(:,1);                                % H2O


if option <3
```

```
disp('Outgoing LW radiation OLR=qNv')
disp('from an average emission height zEmission')
disp('as function of ftot=fp')
Prevost=epssurf*theta(1);
Prevost
fp1=fp(1);
disp('For first fp only CO2 the others H2O')
fp
qNv

qNvminusPrevost=qNv-Prevost
zEmission
%end of loop for different values of nabsorb and for the differnt options
end

% graphical output

switch option
    case {1}
       xtitle1 =...
       ['   Stack in vacuum. Or no interaction with O2 and N2 '];
       xtitle3 =...
       [' "backrad" is not backradiation of heat'];
    case {2}
       xtitle1 =...
       ['   Reversed solution for given temperatures.'];
       xtitle3 =...
       [' "backrad" is not backradiation of heat'];


   case {3}
       xtitle1 = ...
       ['Results for diurnal variation of sea surface'];
       if sea==0
       xtitle1 = ...
       ['Results for diurnal variation of land surface '];
       end
    case {4}
       xtitle1 = ...
       ['  Comparison FEM with Schwarzschild'];
    case {5}
       xtitle1 = ...
       ['Sensitivity analysis'];

end

    xtitle2a = [windowfactor, ' Planck = ',num2str(Planckcase),' nods = ',...
        num2str(nods), '  m = ', num2str(m),'  height = ',...
        num2str(height/1000),' km    LR = ', num2str(1000*LR) ' K/km'];
    xtitle2b= [ 'Equilibrium Point:    ftot =  ',num2str(maxftot), ' TsK =  ',
num2str(TsK),...
                '   epssurf =  ',num2str(epssurf)];


% graphics and further analysis for option 1

if option==1 % begin option ==1

   TLR(nods)=TLR(nods-1);
   tempv(nabsorb+1,:)= TLR; %add environmental lapse rate temperature for
comparison

   for ifig=1:2
```

```
   if ifig==1
   figure
   xtitle = ['fig 1.1a ',xtitle1];

   plot(tempv,znod/1000,'LineWidth',2)

      for jf=1:nabsorb
  switch jf
       case {1}
         str1=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,'ELR Temp' )
       case {2}
           str2=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2, 'ELR Temp')
       case {3}
            str3=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,'ELR Temp' )
       case {4}
         str4=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4,'ELR Temp')
       case {5}
          str5=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4,str5,'ELR Temp')
       case {6}
         str6=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4,str5,str6,'ELR Temp')
   end
   end

   title({xtitle;xtitle2a;xtitle2b;'Temperature of IR-active trace gas for
various ftot,compared to ELR T'});
   xlabel('Temperature  K')
   ylabel('Height in km')

   else

  if ifig==2

      figure

   plot(thetav,znod/1000,'LineWidth',2)

    for jf=1:nabsorb
  switch jf
       case {1}
         str1=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1)
       case {2}
           str2=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2)
       case {3}
            str3=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3 )
       case {4}
         str4=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4)
       case {5}
          str5=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4,str5)
       case {6}
         str6=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2,str3,str4,str5,str6)
```

```
   end
   end



   xtitle = ['fig 1.1b ',xtitle1];

   title({xtitle;xtitle2a;xtitle2b;'Theta of IR-active trace gas for various
ftot'});
   xlabel('Theta W/m^2')
   ylabel('Height in km')


  end
   end
   end

   %title({xtitle;xtitle2a;xtitle2b;'Temperature of IR-active trace gas for
various ftot,compared to ELR T'});

   %xlabel('Temperature  K','LineWidth',2)
   ylabel('Height in km')


   figure % figure 1.2

    xtitle = ['fig 1.2 ',xtitle1];


   plot(fp,OLR,fp,qsurf,fp,qwindow,fp,backrad,':.',fp,qtoav,':.','LineWidth',2)
   legend('OLR=qsurf','qsurf=OLR','qwindow','"backrad"','qtoa = 240')
   title({xtitle;xtitle2a;xtitle2b;xtitle3;'Fluxes due to radiation only, no
interaction with air, OLR=qsurf'})
   xlabel('ftot','LineWidth',2)
   ylabel('W/m^2','LineWidth',2)


end    %end loop output option == 1

if option ==2 % begin loop output option ==2

  figure %
  disp('Input for global heat budget')
  OLRftot=OLR(nabsorb)
  qsurfftot=qsurf(nabsorb)
  qwindowftot=qwindow(nabsorb)
  qabsorbftot=qabsorb(nabsorb)
  backradftot=backrad(nabsorb)

  xtitle1 = ['Main result of stack model'];
  xtitle3 =...
       [' "backrad" is not backradiation of heat'];
    xtitle = ['fig 2.1 ',xtitle1];


 plot(fp,OLR,fp,qsurf,fp,qwindow,fp,qabsorb,fp,backrad,':.',...
                                                fp,qtoav,':.','LineWidth',2)
  legend('OLR','qsurf','qwindow','qabsorb','"backrad"','qtoa = 240')
  xlabel('ftot')
  ylabel('W/m^2')
  title({xtitle;xtitle2a;xtitle2b;xtitle3;'OLR, qsurf, qwindow and qabsorp as
function of ftot'})
```

```
  figure
  xtitle = ['fig 2.1a ',xtitle1];
  plot(fp,OLR,fp,zEmission,'LineWidth',2)
  legend('OLR','zEmission')
  xlabel('ftot')
  ylabel('W/m^2 for OLR and meters for zEmission')
  title({xtitle;xtitle2a;xtitle2b;'OLR and average emission height of OLR'})


  figure % figure 2.2   option 2

   xtitle = ['fig 2.2 ',xtitle1];


  plot(qv,znod/1000,'LineWidth',2)   %distribution of heat deposits due to the
imposed LR
  title({xtitle;xtitle2a;xtitle2b;...
      ' Heat deposit by other than LW radiation, for different ftot'})
  xlabel('W/m^3')
  ylabel('Height in km')
  for jf=1:nabsorb
  switch jf
       case {1}
         str1=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1)
       case {2}
           str2=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];;
         legend(str1,str2)
       case {3}
            str3=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
         legend(str1,str2,str3)
       case {4}
         str4=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
         legend(str1,str2,str3,str4)
       case {5}
          str5=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
         legend(str1,str2,str3,str4,str5)
       case {6}
         str6=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
         legend(str1,str2,str3,str4,str5,str6)
   end
   end







  figure % figure 2.3   option 2
   xtitle = ['fig 2.3 ',xtitle1];


  plot(qoutv,znod/1000, qintov,znod/1000,'LineWidth',2)
  title({xtitle;xtitle2a;xtitle2b;' Distribution of absorbed and emitted heat
for different ftot'})
  xlabel('W/m^3')
  ylabel('Height in km')


  for jf=1:nabsorb
  switch jf
       case {1}
         str1=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
```

```
        legend(str1)
      case {2}
         str2=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2)
      case {3}
          str3=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3)
      case {4}
        str4=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4)
      case {5}
         str5=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4,str5)
      case {6}
        str6=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4,str5,str6)
   end
  end
  legendold=legend;

  figure % figure 2.4 option 2
   xtitle = ['fig 2.4 ',xtitle1];


  plot(qzv,znod/1000,qtoaz,znod/1000,':','LineWidth',2)
  title({xtitle;xtitle2a;xtitle2b;' LW radiation due to qsurf, but mainly
convection for high ftot'})
  xlabel('W/m^2')
  ylabel('Height in km')
   for jf=1:nabsorb
  switch jf
      case {1}
        str1=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf))];
        legend(str1,'qtoa=240')
      case {2}
         str2=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,'qtoa=240')
      case {3}
          str3=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,'qtoa=240')
      case {4}
        str4=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4,'qtoa=240')
      case {5}
         str5=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4,str5,'qtoa=240')
      case {6}
        str6=  ['ftot=',num2str(fp(jf)), ' OLR= ',num2str(OLR(jf)) ];
        legend(str1,str2,str3,str4,str5,str6,'qtoa=240')
   end
   end
  deltaT=5;
      q=K*theta;
      q0=q;
      OLR0=-q(nods);
      qsurf0=q(1);
             %Ferenc Miskolczi publication values . see ref [1]
      qatmSW0=60;
      qsurfSW0=OLR0-qatmSW0;
      theta0=theta;
      for j=1:3          %different temperature distributions
        mixlair=25;

        for k=1:5
```

```
          if j==2                %j=2      mixlair input
             for i=1:nods
              expz=exp(-znod(i)/mixlair);
              TLR(i)=TLR0(i)+(k-3)*deltaT*expz; %for k=1 TLR(1)=288-2*16,5
                                                %which is 33 degrees below
                                                %288K
              theta(i)=sigma*TLR(i)^4;
             end
          elseif j==1            %j=1 only surface temperature changes
              TLR=TLR0;
              theta=theta0;
          elseif j==3            %j=3  translated ELR distribution from TsK0 +/-
deltaT
              TLR=TLR0+(k-3)*deltaT;
              for i=1:nods
                 theta(i)=sigma*TLR(i)^4;
             end
          end
          TLR(1)=TLR0(1)+(k-3)*deltaT;
          theta(1)=sigma*TLR(1)^4;
          theta(nods)=zeroK;
          qwindowv(k)=(1-ftot)*epssurf*theta(1);
          TLR1v(k)=TLR(1);
          q=K*theta;


          qsum=sum(q);
          qsum0=sum(q0);
          OLRv(k)=-q(nods);
          qsurfv(k)=q(1);
          qatmSWv(k)=-qatmSW0*q(nods)/OLR0;
          qconv(k)=OLRv(k)-qsurfv(k)-qatmSWv(k);
       end
       OLRm(j,:)=OLRv;
       qsurfm(j,:)=qsurfv;
       qatmSWm(j,:)=qatmSWv;
       qwindowm(j,:)=qwindowv;
       if j==3
       B=zeros(2);
       a=zeros(2,1);
       rhs=zeros(2,1);
       for k=1:5
           Tk=TLR1v(k);
           B(1,1)=B(1,1)+1;
           B(1,2)=B(1,2)+ Tk;
           B(2,2)=B(2,2)+Tk^2;
           rhs(1)=rhs(1)+qconv(k);
           rhs(2)=rhs(2)+qconv(k)*Tk;

       end
       TLR1v;
       B(2,1)=B(1,2);
       a=inv(B)*rhs;
       end
      end

      theta=theta0;

      xtitle=['Steady state solutions for various atmospheric and surface
temperatures'];
       figure  %figure 2.5
       xtitle1=['fig 2.5 ',xtitle];
       xtitle2=['Atmospheric T according to lapse rate LR =',num2str(1000*LR),'
```

```
down to the surface'];
       xtitle3= ['But surface temperature different as given in x-axis'];

       plot(TLR1v',OLRm(1,:),TLR1v',qsurfm(1,:)+qatmSWm(1,:)-
qwindowm(1,:),TLR1v',...
            qsurfm(1,:),':.',TLR1v',qwindowm(1,:),':.',TLR1v',...
            qsurfm(1,:)-qwindowm(1,:),'Linewidth',2)
       title({xtitle1;xtitle2;xtitle3});
      legend('OLR1','qatmLW1+qatmSW1','qsurf1','qwindow','qatmLW1')

      ylabel('Fluxes W/m^2')
      xlabel('Surface Temperature K')

       figure %figure 2.6
         plot(TLR1v',OLRm(2,:),TLR1v',qsurfm(2,:)+qatmSWm(2,:)-
qwindowm(2,:),TLR1v',...
           qsurfm(2,:),':.',TLR1v',qwindowm(2,:),':.',TLR1v',...
           qsurfm(2,:)-qwindowm(2,:),'Linewidth',2)
      legend('OLR2','qatmLW2+qatmSW2','qsurf2','qwindow', 'qatmLW2')
      xtitle1=['fig 2.6 ',xtitle];
      xtitle2=['Atmospheric temperature according to lapse rate LR
=',num2str(1000*LR),...
               ' , near surface...'];
       xtitle3= ['TLR=TLR0+LR*z+(TsK-288)exp(-z/mixlair),mixlair=
',num2str(mixlair)];

        title({xtitle1;xtitle2;xtitle3})
        ylabel('Fluxes W/m^2')
      xlabel('Surface Temperature K')

        figure

       plot(TLR1v',OLRm(3,:),TLR1v',qsurfm(3,:)+qatmSWm(3,:),TLR1v',...
           qsurfm(3,:),TLR1v',qwindowm(3,:),'Linewidth',2)
       xtitle1=['fig 2.7 ',xtitle];
       xtitle2=['Atmospheric temperature according to lapse rate LR
=',num2str(1000*LR)];
       xtitle3= ['Starting from surface temperature (x-axis)'];
       title({xtitle1;xtitle2;xtitle3})

      legend('OLR3','qsurf3+qatmSW3','qsurf3','qwindow')
      ylabel('Fluxes W/m^2')
      xlabel('Surface Temperature K')


      figure


plot(TLR1v',OLRm(1,:),TLR1v',qsurfm(1,:)+qatmSWm(1,:),TLR1v',qsurfm(1,:),...
            TLR1v',qwindowm(1,:),'Linewidth',2)
        hold


plot(TLR1v',OLRm(2,:),':.',TLR1v',qsurfm(2,:)+qatmSWm(2,:),':.',TLR1v',...
           qsurfm(2,:),':.','Linewidth',2)


plot(TLR1v',OLRm(3,:),'--',TLR1v',qsurfm(3,:)+qatmSWm(3,:),'--',TLR1v',...
           qsurfm(3,:),'--','Linewidth',2)
        xtitle1=['fig 2.8 ',xtitle];
        xtitle2=['case 1: atmospheric T does not change only surface T'];
        xtitle3=['case 2: from surface T exponential decay towards lapse rate
T'];
        xtitle4=['case 3: atmospheric T defined by lapse rate from new surface
```

```
T'];
          title({xtitle1;xtitle2;xtitle3;xtitle4})
        legend('OLR1','qsurf1+qatmSW1','qsurf1','qwindow',...
                'OLR2','qsurf2+qatmSW2','qsurf2',...
                'OLR3','qsurf3+qatmSW3','qsurf3');


      ylabel('Fluxes W/m^2')
      xlabel('Surface Temperature K')
      i=0
      while znod(i+1)< mixlair
          i=i+1;
          zplot(i)=znod(i);
          TLRp(i)=TLR0(i);
          N2plot=i;

      end
      TLR1p=TLRp;
      TLR1p(1)=TLR1p(1)+deltaT;
      TLRm1p=TLRp;
      TLRm1p(1)=TLRm1p(1)-deltaT;
      TLR2p=TLRp;
      TLRm2p=TLRp;
      for i=1:N2plot
      TLR2p(i)=TLR2p(i)+deltaT*exp(-znod(i)/mixlair);
      TLRm2p(i)=TLRm2p(i)-deltaT*exp(-znod(i)/mixlair);
      end
      TLR3p=TLRp +deltaT;
      TLRm3p=TLRp -deltaT;
      figure
      plot(TLR1p,zplot,TLR2p,zplot,TLR3p,zplot,'linewidth',2)
      legend ('case 1','case 2', 'case 3')
      xtitle1=['fig 2.9 ',xtitle];
        xtitle2=['case 1: atmospheric lapse rate, atmospheric T does not change
only surface T'];
        xtitle3=['case 2: exp(-z/relax) decay towards lapse rate mixlair= ',...
            num2str(mixlair)];
        xtitle4=['case 3: atmospheric T defined by lapse rate starting'...
                 'from new surface T'];
         title({xtitle1;xtitle2;xtitle3;xtitle4})

      hold
      plot(TLRm1p,zplot,TLRm2p,zplot,TLRm3p,zplot,'linewidth',2)
      xlabel('Temperature  K')
      ylabel ('Height  meters')

end %end loop output option 2

if option==3   %begin loop output option 3 diurnal transient
theta0=theta;
daysec=24*3600;
Tmin=1000.0;
Tmax=-1000.0;
OLRmax=0.0;
OLRmin=1500.0;
iplot=0;
iplotd=0;
time=0;
oclock=0;
Tav=0;
OLRav=0;

q = K*theta;
OLR=-q(nods);
```

```
OLR0=OLR;
T1p=TE(1);
qwindow=(1-ftot)*epssurf*Planck(Planckcase)*T1p^Planckcase;  %global and annual
mean through window
if qwindow < 0
    qwindow=0;
end

qwindow0=qwindow;                          %memorize steady-state value
qatmLW0=q(1)-qwindow0 ;    %global and annual mean LW atmospheric absorption
from
                          %surface outgoing LW radiation
qatmLW=qatmLW0;
qsurfLW=q(1);
qsurfLW0=qsurfLW;
q(1)=0;
q(nods)=0;
qsum=sum(q);
qsum0=qsum;
                    %given by K&T type of publications, see ref [1]
qatmSW0=60;
qsurfSW0=OLR0-qatmSW0;
qatmSW=qatmSW0;
qconv0=qsum0-qatmSW0;
qconv=qconv0;

qconvreal=qconv;         % we suppose a delay in qconv due to enertia
                         % sailplaners know that thermals begin by 10
                         % o'clock, we give a time consyant of 2
                         % hours.(input)
qdistr=q/qsum;      %distribution of incoming SW and of qconv,
                     %probably better to take fdH2O for SW

qsunav=0;
qtot=0;
qconvav=0;
qconvfactor=1;

TLRmaxd=zeros(nods,1);
TLRmind=zeros(nods,1);
rhs=zeros(Ntot,1);
TLR0=TLR;

 qtotatmp=qatmSW+qatmLW+qconv;

while time<timetot

    iplot=iplot+1;
    iplotd=iplotd+1;
    timep(iplot)=time;
    oclockp(iplotd)=oclock;
    OLRp(iplot)=OLR;
    TE1p(iplot)=TE(1);
    qsun=pi*OLR0*cos(omega*time+pi);
    if qsun<0
        qsun=0;
    end
     %qsun=OLR0;  %test steady state

     qsurfSW = qsun/OLR0*qsurfSW0;           %incoming SW sunlight at surface
     qsurfSWp(iplot)=qsurfSW;                %prepare plotting
     qatmSW = qsun/OLR0*(OLR0-qsurfSW0);     % SW sun light absorbed in
atmosphere
     qatmSWp(iplot)=qatmSW;                  %prepare plotting
```

```
     qsunp(iplot)=qsun;


    time=time+deltat;
    oclock=oclock+deltat;

    qwindow=(1-ftot)*epssurf*Planck(Planckcase)*TE(1)^Planckcase;  %SS value 52
    if qwindow < 0
       qwindow=0;
    end

    q = K*theta;
    qsurfLW=q(1);
    q(1)=0;
    q(nods)=0;          % q is now mechanisms other than LW radiation
    qsum=sum(q);
    qdistr =q/qsum;
    qatmLW=qsurfLW-qwindow;             % LW surface radiation absorped by
atmosphere.
                                        %SS value 168
    qatmSW=qsun/OLR0*(OLR0-qsurfSW0);   %SW sun light absorbed by atmosphere
                                        %SS value 72

    switch convhyp
    case {1}
       qconv=qconv0;
    case {2}
       dTsK=TE(1)-TsK0;
       if dTsK>=0
          qconv=qconv0*(2-exp(-dTsK/20))*qconvfactor;
       else
           qconv=qconv0*exp(dTsK/20)*qconvfactor;
       end
    case {3}
       dTsK=TE(1)-TsK0;
       if dTsK>=0
          qconv=qconv0*(2-exp(-dTsK))*qconvfactor;
       else
           qconv=qconv0*exp(dTsK)*qconvfactor;
       end
    case {4}
       qconv=qsun*qconv0/OLR0*qconvfactor;
    end




    qtotatm=qatmSW+qatmLW+qconvreal;
    qnetatm=qtotatm-qtotatmp;
    qtotatmp=qtotatm;
    rhs=zeros(Ntot,1);
    if sea==1
    if TX(nods)<TX0(Ntot)
        TX=TX0;
   %rhs = rhs-20*KWC*(TX-TX0);
    end
    end

    rhs = rhs-KE*(TX-TX0);

    for i=1:nods
      rhs(i)=rhs(i)+qdistr(nods-i+1)*qnetatm;
    end
```

```
    qsurf1=qsurfSW;                          %part is going to be be absorbed
                                             %below sea surface
    qsurf2=qconvreal+qatmLW+qwindow;

    if sea ==1
        for i=1:Npen
            rhs(nods+i-1)=rhs(nods+i-1)+fdepth(i)*qsurf1;
        end
        rhs(nods)=rhs(nods)-qsurf2;
    else                                  %for land no penetration only conduction
        rhs(nods)=rhs(nods)+qsurf1-qsurf2;
    end


   dTX = invMED*rhs*deltat;
   TX=TX+dTX;
   TX(1)=0;

   if tauconv>0                                        %space-time finite element
       ratio=deltat/tauconv;
       dqconvreal=ratio/(1+2/3*ratio)*(-qconvreal+qconv);
       qconvreal=qconvreal+dqconvreal;
    else
        qconvreal=qconv;
   end

   smooth=1;
   if smooth>0
       TXnods=TX(nods);
     for i=1:Ntot          % smooth temperatures
        TXi=TX(i);
        if TXnods>TX0(nods)    %afternoon
            if TXi<TX0(i)
               TX(i)=TX0(i);
            end
        elseif TXnods<TX0(nods)    %morning
            if TXi>TX0(i)
                TX(i)=TX0(i);
            end
        end
     end
   end                        %end smooth

   %define TLR and TE from TX

    for i=1:nods
        TLR(i)=TX(nods-i+1);
    end
    for i=1:Nze                             %extract TE from TX
        TE(i)=TX(nods+i-1);
    end

    for i=1:nods
        theta(i)=Planck(Planckcase)*TLR(i)^Planckcase;        %define new system
parameters
    end

    theta(nods)=zeroK;
    OLR=qsurfLW +qatmSW +qconvreal;
    if OLR>OLRmax
        OLRmax=OLR;
    elseif OLR<OLRmin
        OLRmin=OLR;
    end
```

```
    if     TE(1)>Tmax
           Tmax=TE(1);
           Tmaxd=TE;
           TLRmaxd=TLR;
           TXmaxd=TX;
    elseif TE(1)<Tmin
            Tmin=TE(1);
            Tmind=TE;
            TLRmind=TLR;
             TXmind=TX;
    end

    TEdayp(iplotd)=TE(1);
    qconvdayp(iplotd)=qconv;
    qconvrealdayp(iplotd)=qconvreal;
    qwindowdayp(iplotd)=qwindow;
    qsurfLWdayp(iplotd)=qsurfLW;
    qatmLWdayp(iplotd)=qwindow+qatmLW;
    qatmSWdayp(iplotd)=qwindow+qatmLW+qatmSW;
    OLRdayp(iplotd)=qwindow+qatmLW+qatmSW+qconvreal;
    if oclock>daysec;
        if timetot-time>daysec
            Tmin=1000;
            Tmax=0;
            OLRmim=1500;
            OLRmax=0;
        end
        oclock=oclock-daysec;
        iplotd=0;
        qconvav1=qconvav/time;
        qconvfactor=qconvfactor*qconv0/qconvav1;
    end
    qsunav=qsunav+qsun*deltat;
    OLRav=OLRav+OLR*deltat;
    Tav=Tav+TE(1)*deltat;
    qtot=qtot+qsun*deltat;
    qconvav=qconvav+qconvreal*deltat;
end

deltaT1=Tmin-TLR0(1);
deltaT3=Tmax-TLR0(1);
% Tmax and Tmin plots for zx<5*mixlair meter
j=0;
for i=Ntot:-1:1
    if zx(i)<1.5*mixlair
        j=j+1;
        linezeroplot(j)=0;
        zxplot(j)=zx(i);
        TXmaxplot(j)=TXmaxd(i);
        TXminplot(j)=TXmind(i);
        TX0plot(j)=TX0(i);
    end
end

Nmaxplot=j;

qsunav=qsunav/time;
qav=qtot/time;
Tav=Tav/time;
OLRav=OLRav/time;
timeday=timep/3600/24;
```

```
%graphics and further analysis for option 3

figure
xtitle = ['fig 3.1 ',xtitle1];
xtitle3=['OLRmax = ',num2str(OLRmax),' OLRmin = ',num2str(OLRmin),...
         ' Tsmax = ', num2str(Tmax), ' Tsmin = ', num2str(Tmin)];
switch convhyp
    case{1}
     xtitle2=['constant convection value ,' num2str(qconv0),' W/m^2'];

    case{2}
     xtitle2=['convection proportional 2-exp(-dTsK/20) and exp(dTsk/20)for
dTsK<0   , time  delay ',...
               num2str(tauconvh),' hr'];

    case{3}
     xtitle2=[' convection proportional to 2-exp(-dTsK) and dTsK<0 to exp(dTsK),
time delay ',...
               num2str(tauconvh),' hr'];
    case{4}
     xtitle2=[' convection proportional to qsun with average global and annual
mean, time delay ',...
               num2str(tauconvh),' hr'];


     end
plot(timeday,OLRp,timeday,TE1p,timeday,qsunp,timeday,...
     qsurfSWp,timeday,qatmSWp,'Linewidth',2)
xtitle4 = ['Diurnal variation of OLR, qsun and surface temperature'];
title({xtitle;xtitle2;xtitle3;xtitle4})
legend('OLR','Ts','qsuntot','qsunSurf','qsunAtm')
if timetoth>25
xlabel('days')
else
xlabel('oclock, hours')
end
ylabel('fluxes   W/m^2     temperature K')


figure
xtitle = ['fig 3.2 ',xtitle1];

oclockp=oclockp/3600;
plot(oclockp,OLRdayp,oclockp,TEdayp,oclockp,qsurfLWdayp,...
    oclockp,qwindowdayp,oclockp,qconvrealdayp,oclockp,qconvdayp,'linewidth',2)
if sea == 1
xtitle4=['Diurnal variation OLR and Ts for sea surface'];
else
xtitle4=['Diurnal variation OLR and  Ts for land surface'];
end
title({xtitle;xtitle2a;xtitle2b;xtitle3;xtitle4});
legend('OLR','Ts','qsurfLW+qwindow','qwindow','qconvreal','qconv')
xlabel('oclock hours')
ylabel('fluxes   W/m^2     temperature K');

figure
xtitle = ['fig 3.3 ',xtitle1];
plot(oclockp,TEdayp,'linewidth',2)
if sea ==1
xtitle4=['Diurnal variation surface temperature for sea surface'];
else
xtitle4=['Diurnal variation surface temperature for land surface'];
end
```

```
title({xtitle;xtitle2a;xtitle2b;xtitle3;xtitle4});
legend('Ts')
xlabel('oclock hours')
ylabel('     temperature K');


figure
xtitle = ['fig 3.4 ',xtitle1];
 plot(oclockp,OLRdayp,'linewidth',2)
if sea ==1
xtitle4=['Diurnal variation OLR for sea surface'];
else
xtitle4=['Diurnal variation OLR for land surface'];
end
title({xtitle;xtitle2a;xtitle2b;xtitle3;xtitle4});
legend('OLR')
xlabel('oclock  hours')
ylabel('   W/m^2 ');

figure
xtitle = ['fig 3.5 ',xtitle1];

plot(Tmaxd,ze,Tmind,ze,'linewidth',2)

if sea ==1
    xtitle4=['Max and minimum temperatures below sea surface'];
else
    xtitle4 =['Max and minimum temperatures below land surface'];

end

title({xtitle;xtitle2a;xtitle2b;xtitle3;xtitle4});

legend('Tmax','Tmin');
ylabel('depth meters');
xlabel(' temperature K');

figure
xtitle = ['fig 3.6 ',xtitle1];

if sea ==1
    xtitle4=['Max and minimum temperatures for sea surface'];
else
    xtitle4 =['Max and minimum temperatures for land surface'];

end


xtitle3=['OLRmax = ',num2str(OLRmax),' OLRmin = ',num2str(OLRmin),...
         ' Tsmax = ', num2str(Tmax), ' Tsmin = ', num2str(Tmin)];
plot(TXmaxplot',zxplot,TXminplot',zxplot,TX0plot',zxplot,'linewidth',2)
title({xtitle;xtitle2a;xtitle2b;xtitle3;xtitle4})
legend('after noon','morning inversion','mean lapse rate')
xlabel('Air (z>0) and sub-surface (z<0) temperature  K')
ylabel(' z in meters')


for i=1:40

    dTsK =-10 +i/2;
    dTsKv(i)=dTsK;
    if dTsK<0
        qconvv(i,3)=109*exp(dTsK);
        qconvv(i,2)=109*exp(dTsK/20);
```

```
        qconvv(i,1)=109;
    else
        qconvv(i,3)=109*(2-exp(-dTsK));
        qconvv(i,2)=109*(2-exp(-dTsK/20));
        qconvv(i,1)=109;
    end
end
figure
xtitle = ['fig 3.7 ',xtitle1];

plot(dTsKv,qconvv,'linewidth',2)
title({xtitle;'Examples of convection options'})
legend('constant convection','hypothesis 2: exp(dTsK/10)','hypothesis 3
:exp(dTsK/2)')
xlabel('dTsK in degrees')
ylabel( 'W/m^2')
end    %end loop if  option ==3


%graphics and further analysis for options 2 and 5









if option==4     %begin output option==4   Schwarzschild
  figure  % figure 4.1
  xtitle = ['fig 4.1 ',xtitle1];
  OLR=OLR;
  OLRS0=OLRS0;
  backrad=backrad;
  backradS0=backradS0;
  qwindow=qwindow;
  qtoav=qtoav;

plot(fp,OLR,fp,OLRS0,'--',fp,backrad,':.',fp,backradS0,'--',fp,qsurf,fp,qsurfS0,
...
      fp,qwindow',':.',fp,qtoav',':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) with original
Schwarzschild solutions'})
  legend('OLR option 2','OLRS0',...
         '"backrad" option2','backradS0','qsurf
option2','qsurfS0','qwindow','qtoa=240')
  xlabel('ftot')
  ylabel('W/m^2')

  figure  % figure 4.2
  xtitle = ['fig 4.2 ',xtitle1];

  plot(fp,OLR,fp,OLRS0,'--',fp,OLRS0MBR,'--',...
       fp,backrad,':.',fp,backradS0,'--',...
       fp,qwindow,':.',fp,qtoav,':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) with  Schwarzschild
modified with backrad'})
  legend('OLR option 2','OLRS0','OLRS0MBR',...
         '"backrad" option2','backradS0','qwindow','qtoa=240')
  xlabel('ftot')
  ylabel('W/m^2')
```

```
  figure  % figure 4.3
  xtitle = ['fig 4.3 ',xtitle1];

  plot(fp,OLR,fp,OLRS0,'--',fp,OLRS0MW,'--',...
       fp,backrad,':.',fp,backradS0,'--',...
       fp,qwindow,':.',fp,qtoav,':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) with  Schwarzschild
modified with window'})
  legend('OLR option 2','OLRS0','OLRS0MW',...
         '"backrad" option2','backradS0','qwindow','qtoa=240')
  ylabel('W/m^2')
  xlabel('ftot')



  figure  % figure 4.4
  xtitle = ['fig 4.4 ',xtitle1];

  plot(fp,OLR,fp,OLRS0,'--',fp,OLRS0M,'--',...
       fp,backrad,':.',fp,backradS0,'--',...
       fp,qwindow,':.',fp,qtoav,':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) with Schwarzschild
modified backrad and window'})
  legend('OLR option 2','OLRS0','OLRS0M',...
         '"backrad" option2','backradS0','qwindow','qtoa=240')
  xlabel('ftot')
  ylabel('W/m^2')


  figure  % figure 4.5
  xtitle = ['fig 4.5 ',xtitle1];

 plot(fp,OLR,fp,OLRS0,'--',fp, OLRS1,'--',fp,OLRS2,'--',...
              fp,OLRS0M,'--',fp, OLRS1M,'--',fp,OLRS2M,'--',...
       fp,backrad,':.',fp,backradS0,'--',fp,backradS1,'--',...
       fp,backradS2,'--',fp,qwindow,':.',fp,qtoav,':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) with various
modified Schwarzschild'})
  legend('OLR option 2','OLRS0','OLRS1','OLRS2',...
                        'OLRS0M','OLRS1M','OLRS2M',...

'"backrad"option2','backradS0','backradS1','backradS2','qwindow','qtoa=240')
  xlabel('ftot')
  ylabel('W/m^2')

  figure  % figure 4.6
  xtitle = ['fig 4.6 ',xtitle1];


plot(fp,OLR,fp,qsurf,fp,qwindow,fp,qabsorb,fp,backrad,':.',fp,backradSvf,'--',fp
, OLRSvf,'--',...
       fp,qsurfSvf,'--', fp,qabsorbSvf,'--', fp,qtoav,':.','LineWidth',2)

  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM (option 2) and FEM with
Schwarzschild viewfactors(*Svf)'})
  legend('OLR option 2','qsurf option 2','qwindow option 2','qabsorb option
2',...
         '"backrad"option2','backradSvf','OLRSvf','qsurfSvf',...
         'qabsorbSvf','qtoa = 240')
  xlabel('ftot')
```

```
  ylabel('W/m^2')

  figure   % figure 4.7
  xtitle = ['fig 4.7 ',xtitle1];

  surf(viewfactorF)
  title({xtitle;xtitle2a;xtitle2b;' Finite Element ViewfactorF'})
  xlabel('node number j')
  ylabel('node number i')
  zlabel('viewfactorF(i,j)')
  colormap(jet)
  colorbar

  figure   % figure 4.8
  xtitle = ['fig 4.7 ',xtitle1];

  surf(viewfactorS)
  title({xtitle;xtitle2a;xtitle2b;' Schwarzschild ViewfactorS for watervapor
distribution'})
  xlabel('node number j')
  ylabel('node number i')
  zlabel('viewfactorS(i,j)')
  colormap(jet)
  colorbar




 figure   % figure 4.9
  xtitle = ['fig 4.9 ',xtitle1];
  plot(znod/1000,windowS,znod/1000,windowF,'LineWidth',2)
  title({xtitle;xtitle2a;xtitle2b;'FEM windowF and Schwarzschild windowS for
watervapor distribution'})
  xlabel('height km')
  ylabel('windowF or windowS')
  legend('Schwarzschild', 'FEM')

  figure   % figure 4.10
  xtitle = ['fig 4.10 ',xtitle1];
  for i=1:nods
      vi(i)=i;
  end

  plot(vi,windowS,vi,windowF,'LineWidth',2)
  title({xtitle;xtitle2a;xtitle2b;'Comparison FEM windowF and Schwarzschild
windowS'})
  xlabel('node number')
  ylabel('windowF or windowS')
  legend('Schwarzschild', 'FEM')

end  %end loop output option ==4



%graphics and further analysis for options  5

if option ==5           %sensitivity analysis for option ==5

disp('Sensitivity analysis')
disp(' ')
   nods
```

```
   height
   toa
 for i=1:nods-1
   % theta(i)=Planck(Planckcase)*TLR(i)^Planckcase;       %sigma*T^4
    dthetadTi=Planckcase*Planck(Planckcase)*TLR(i)^(Planckcase-1); %4*sigma*T^3
    zi=znod(i);
    dthetadT(i)=dthetadTi;
    dthetadELR(i)=dthetadTi*zi;
end

  theta(nods)=zeroK;
  dthetadT(nods)=0;
  dthetadELR(nods)=0;
  Prevost=epssurf*theta(1)    %OLR for f=0. no H2O and no CO2
  epssurf
  OLRCO2 = -knodsco2'*theta;       %OLR for f=ftotco2
  OLRH2O = -knodsh2o'*theta;       %OLR for only H2O
  forcingco2=epssurf*theta(1)-OLRCO2
  forcingh2o=epssurf*theta(1)-OLRH2O


   OLR=epssurf*theta(1)-forcingh2o-forcingco2
   epsvector =zeros(nods,1);
   epsvector(1)=epssurf;
   dOLRdTs = -dthetadT*(knodsco2+epsvector+knodsh2o); %   %variation of OLR with
surface temperature W/m^2/K
   disp(['Change in OLR due to dTs: dOLR/dTs = ',num2str(dOLRdTs)])
   dOLRdELR = -dthetadELR*(knodsco2+epsvector+knodsh2o);

  dTs = forcingco2/dOLRdTs;
  dTs = round(dTs*1000)/1000;
  ftotco2
  ftoth2o
  maxftot
  disp('ELR SENSITITY FOR DOUBLING OF CO2')
  dOLRdELRpkm=dOLRdELR/1000;
  dELR = forcingco2/dOLRdELR;
  dELRpkm=dELR*1000;
  ELRpkm=ELR*1000;
  ELRpkmnew=ELRpkm+dELRpkm;
  dTtoa=dELR*toa;
  disp(['dOLRco2 = ', num2str(-forcingco2), ' dOLR/dELR =
',num2str(dOLRdELR/1000)])
  disp(['ELR K/km = ',num2str(ELRpkm), ' dELR K/km = ',num2str(dELRpkm),...
        ' dT at TOA = ',num2str(dTtoa)])
  disp(' ')

  disp(['For option ',num2str(option),' with height =', num2str(height)])
  disp('SURFACE TEMPERATURE SENSITIVITY FOR DOUBLING OF CO2')
  disp(['dOLRco2 = ',num2str(-forcingco2),'   dOLR/dTs = ',num2str(dOLRdTs),...
        '  dTs = ', num2str(dTs)])
  disp(' ')
  disp('Hit any key to continue with the plots')

  pause

  xtitle1 = ['  Temperature distribution in atmososphere'];
    figure
    xtitle = ['fig 5.2 ',xtitle1];

    plot(TLR,znod/1000,TLR+dTLR',znod/1000,'--','LineWidth',2)
    legend('TLR','TLR+1 degree K')
    title({xtitle;' Atmosphere up to 30 km'})
    xlabel('Temperature, K')
```

```
    ylabel('Height in km')


   xtitle1 = ['  Normalized Temperature, water vapor for m = ', num2str(m),' and
CO2 distribution'];
    TLR=TLR/TsK;

    figure
    xtitle = ['fig 5.3 ',xtitle1];

    plot(TLR,znod/1000,co2,znod/1000,h2o,znod/1000,'LineWidth',2)
    legend('TLR/TsK', 'fdCO2','fdH2O')
    title({xtitle;' Atmosphere up to 30 km'})
    xlabel('Normalized Distribution')
    ylabel('Height in km')
    theta=TLR.^Planckcase;
    co2theta=co2.*theta;
    h2otheta=h2o.*theta;

    figure
    xtitle = ['fig 5.4 ',xtitle1];

    plot(theta,znod/1000,co2theta,znod/1000,h2otheta,znod/1000,'LineWidth',2)
    switch Planckcase
        case {1}
           legend('theta=(TLR/TsK)', 'fdCO2*(TLR/TsK)','fdH2O*(TLR/TsK)')
           title({xtitle;'Multiplied by theta=(TLR/TsK)   Atmosphere up to 30
km'})
        case {2}
       legend('theta=(TLR/TsK)^2', 'fdCO2*(TLR/TsK)^2','fdH2O*(TLR/TsK)^2')
       title({xtitle;'Multiplied by theta=(TLR/TsK)^2   Atmosphere up to 30
km'})
        case {3}
       legend('theta=(TLR/TsK)^3', 'fdCO2*(TLR/TsK)^3','fdH2O*(TLR/TsK)^3')
       title({xtitle;'Multiplied by theta=(TLR/TsK)^3   Atmosphere up to 30
km'})
        case {4}
       legend('theta=(TLR/TsK)^4', 'fdCO2*(TLR/TsK)^4','fdH2O*(TLR/TsK)^4')
       title({xtitle;'Multiplied by theta=(TLR/TsK)^4   Atmosphere up to 30
km'})
    end
    xlabel('Normalized Distribution')
    ylabel('Height in km')


    sh2o=zeros(nods2,1);
    sco2=zeros(nods2,1);
    sTLR=zeros(nods2,1);
    sznod=zeros(nods2,1);
    stheta=zeros(nods2,1);
    for i=1:nods2
        sh2o(i)=h2o(i);
        sco2(i)=co2(i);
        sTLR(i)=TLR(i);
        stheta(i)=theta(i);
        sznod(i)=znod(i);
    end

    figure
    xtitle = ['fig 5.5 ',xtitle1];

    plot(sTLR,sznod/1000,sco2,sznod/1000,sh2o,sznod/1000,'LineWidth',2)
    legend('TLR/TsK', 'fdCO2','fdH2O')
    title({xtitle;' Atmosphere up to 10 km'})
```

```
    xlabel('Normalized Distribution')
    ylabel('Height in km')

    sh2otheta=sh2o.*stheta;
    sco2theta=sco2.*stheta;
    figure
    xtitle = ['fig 5.6 ',xtitle1];

plot(stheta,sznod/1000,sco2theta,sznod/1000,sh2otheta,sznod/1000,'LineWidth',2)
    legend('theta=(TLR/TsK)^4', 'fdCO2*(TLR/TsK)^4','fdH2O*(TLR/TsK)^4')
    title({xtitle;'Multiplied by theta=(TLR/TsK)^4   Atmosphere up to 10 km'})

    xlabel('Normalized Distribution')
    ylabel('Height in km')


  end  %end loop if option==5
  disp(' ')
 end  %end option>0
end
```