

```

% этот та же модель изотопного состава осадков, но не её параметры
% выбираются каждый раз случайным образом из заданного диапазона.
% к этому скрипту обращается скрипт "SIM_Monte_Carlo".

% стираем результаты расчетов предыдущего цикла:
clear k
clear k1
clear k2
clear k3
clear alfa17_e
clear alfa17_ef
clear alfa17_kin
clear alfa18_e
clear alfa18_ef
clear alfa18_kin
clear alfa_kin17
clear alfa_kin18
clear alfa_kinD
clear alfaD_e
clear alfaD_ef
clear alfaD_kin
clear aux17
clear aux17_2
clear aux18
clear aux18_2
clear auxD
clear auxD_2
clear delta18_s
clear deltaD_s
clear delta17_s
clear deltaT_total
clear deltaT_lat
clear dL_rel
clear dyv_rel
clear dyw
clear k17_modif
clear k18_modif
clear kD_modif
clear L
clear lat_grad_T
clear ps
clear Si
clear R17_m
clear R17_s
clear R18_m
clear R18_s
clear RD_m
clear RD_s
clear ys
clear yw
clear z
clear z1
clear idx

```

```

clear idx1
clear graddD_T
clear graddxs_dD
clear grad170excess_dD
clear fitdD_T
clear fitdxs_dD
clear fit170excess_dD

% по предельным значениям диапазонов случайным образом считаем значения
% входных параметров.
% предельные значения диапазонов заданы в скрипте SIM_Monte_Carlo

% условия в источнике:
% т-ра в источнике влаги
teta_s = (rand - 0.5)*(teta_s_max-teta_s_min)+(teta_s_min+teta_s_max)/2;
if Humid_quest == 1
    % влажность считается случайным образом в заданных пределах:
    H_s0 = (rand - 0.5)*(H_s0_max-H_s0_min)+(H_s0_min+H_s0_max)/2;
elseif Humid_quest == 2
    H_s0 = teta_s*gradHs_T+freemem;    % влажность считается по температуре в источнике
else
    disp('нужно ввести 1 или 2. Попробуй еще раз');
    return;
end
% кин. к-т для кислорода 18:
k_180 = (rand - 0.5)*(k_180_max-k_180_min)+(k_180_min+k_180_max)/2;
% во сколько раз кин. к-т для дейтерия больше, чем к-т для кислорода 18:
kD_vs_k18 = (rand - 0.5)*(kD_vs_k18_max-kD_vs_k18_min)+(kD_vs_k18_min+kD_vs_k18_max)/2;
k_D = k_180 * kD_vs_k18;                % кин. к-т для дейтерия
% во сколько раз кин. к-т для кислорода 17 больше, чем к-т для кислорода 18
k17_vs_k18 = (rand - 0.5)*(k17_vs_k18_max-k17_vs_k18_min)+...
    (k17_vs_k18_min+k17_vs_k18_max)/2;
k_170 = k_180 * k17_vs_k18;            % кин. к-т для кислорода 17
delta18_m = (rand - 0.5)*(delta18_m_max-delta18_m_min)...
    +(delta18_m_min+delta18_m_max)/2; % изот. состав мор. воды по кислороду 18
deltaD_m = (rand - 0.5)*(deltaD_m_max-deltaD_m_min)...
    +(deltaD_m_min+deltaD_m_max)/2;   % изот. состав мор. воды по дейтерию
% если надо, чтобы dxs не менялся, тогда deltaD_m = 8*delta18_m;
delta17_m = (rand - 0.5)*(delta17_m_max-delta17_m_min)...
    +(delta17_m_min+delta17_m_max)/2; % изот. состав мор. воды по кислороду 17
% если надо, чтобы 170-xs не менялся, тогда
% delta17_m = ((delta18_m/1000+1)^0.528 -1)*1000;

Lambda180 = (rand - 0.5)*(Lambda180_max-Lambda180_min)...
    +(Lambda180_min+Lambda180_max)/2; % параметр циркуляции
% во сколько раз лямбда для дейтерия больше, чем для кислорода 18
LambdaD_vs_180 = (rand - 0.5)*(LambdaD_vs_180_max-...
    LambdaD_vs_180_min)+(LambdaD_vs_180_min+LambdaD_vs_180_max)/2;
LambdaD = Lambda180 * LambdaD_vs_180;    % а вот и лямбда для дейтерия
% во сколько раз лямбда для кислорода 17 больше, чем для кислорода 18
Lambda170_vs_180 = (rand - 0.5)*(Lambda170_vs_180_max...
    -Lambda170_vs_180_min)+(Lambda170_vs_180_min+Lambda170_vs_180_max)/2;
Lambda170 = Lambda180 * Lambda170_vs_180; % а вот и лямбда для кислорода 17

```

```

% степень для расчета равновесного к-та фракционирования для кислорода 17
n = (rand - 0.5)*(n_max-n_min)+(n_min+n_max)/2;
% параметры траектории:
gam = (rand - 0.5)*(gam_max-gam_min)+(gam_min+gam_max)/2; % кривизна траектории
% вертикальный градиент температуры
beta_E = (rand - 0.5)*(beta_E_max-beta_E_min)+(beta_E_min+beta_E_max)/2;
% температура конденсации в конце, *C
T_d = (rand - 0.5)*(T_d_max-T_d_min)+(T_d_min+T_d_max)/2;
% cloud physics:
L0 = (rand - 0.5)*(L0_max-L0_min)+(L0_min+L0_max)/2; % влагонасыщение облаков
Nu = (rand - 0.5)*(Nu_max-Nu_min)+(Nu_min+Nu_max)/2; % доля испаряющейся в облаке влаги
% параметр перенасыщения ледяных облаков влагой
sigma0 = (rand - 0.5)*(sigma0_max-sigma0_min)+(sigma0_min+sigma0_max)/2;
% температура перехода от смешанных к ледяным облакам
T_i = (rand - 0.5)*(T_i_max - T_i_min)+(T_i_min + T_i_max)/2;
% температура перехода от жидких к смешанным облакам
T_w = (rand - 0.5)*(T_w_max - T_w_min)+(T_w_max + T_w_min)/2;
% к-ты диффузии молекул воды в воздухе
Dif_180 = (rand - 0.5)*(Dif_180_max-Dif_180_min)+(Dif_180_min+Dif_180_max)/2;
Dif_D = (rand - 0.5)*(Dif_D_max-Dif_D_min)+(Dif_D_min+Dif_D_max)/2;
Dif_170 = (rand - 0.5)*(Dif_170_max-Dif_170_min)+(Dif_170_min+Dif_170_max)/2;

% ПРОВЕРКА ВХОДНЫХ ПАРАМЕТРОВ

if T_i > T_w
    disp('Проверь параметры модели. T_i должно быть ниже, чем T_w');
    return;
end

if H_s0 > 1
    disp('влажность не должна превышать 1');
    return;
end

% БЛОК 1. ИЗОТОПНЫЙ СОСТАВ ВОДЯНОГО ПАРА В ИСТОЧНИКЕ

% считаем к-ты k* (см. предпоследнюю формулу на стр. 26 у Salamatin et al., 2004)
k18_modif = k_180 + Lambda180*(1-k_180);
kD_modif = k_D + LambdaD*(1-k_D);
k17_modif = k_170 + Lambda170*(1-k_170);

alfa18_e = exp(A18_v_l/((teta_s+273.15)^2)-B18_v_l/(teta_s+273.15)-C18_v_l); % равновесные к-ты
alfaD_e = exp(AD_v_l/((teta_s+273.15)^2)-BD_v_l/(teta_s+273.15)+CD_v_l); % равновесные к-ты
alfa17_e = alfa18_e^n;
alfa18_kin = (1-k18_modif*H_s0)/(1-k18_modif); % кинетические коэффициенты фракционирования
alfaD_kin = (1-kD_modif*H_s0)/(1-kD_modif);
alfa17_kin = (1-k17_modif*H_s0)/(1-k17_modif);
alfa18_ef = alfa18_e*alfa18_kin; % эффективные коэффициенты фракционирования
alfaD_ef = alfaD_e*alfaD_kin;
alfa17_ef = alfa17_e*alfa17_kin;
R18_m = (delta18_m+1000)/1000*2005; % изотопный состав воды в R
RD_m = (deltaD_m+1000)/1000*312;
R17_m = (delta17_m+1000)/1000*380;

```

```

R18_s = R18_m/alfa18_ef; % изотопный состав первичного пара в R
RD_s = RD_m/alfaD_ef;
R17_s = R17_m/alfa17_ef;
delta18_s = (R18_s/2005-1)*1000; % изотопный состав первичного пара в промилле
lnd180_s = log(1+delta18_s/1000)*1000;
deltaD_s = (RD_s/312-1)*1000;
lndD_s = log(1+deltaD_s/1000)*1000;
delta17_s = (R17_s/380-1)*1000;
dxs_s = deltaD_s-8*delta18_s;
dln_s = (lndD_s)-(-0.0285*((lnd180_s)^2) + 8.47*(lnd180_s)); % dln по Uemura
O17xs_s = (log(delta17_s/1000+1) - 0.528*log(delta18_s/1000+1)) * 1000000;

% БЛОК 2. РАСПРЕДЕЛЕНИЕ ТЕМПЕРАТУРЫ, ДАВЛЕНИЯ И К-ТОВ ФРАКЦИОНИРОВАНИЯ ВДОЛЬ ТРАЕКТОРИИ

inc = S/300; % разбиваем нашу траекторию на 300 отрезков
path = 0:inc:S;
path=path';

path(:,2) = E_d*((1-exp(-gam*path(:,1)))/(1-exp(-gam*S))); % считаем высоту воздушной массы, м

deltaT_total = teta_s - T_d; % считаем распределение температуры вдоль траектории, *C
beta_E = beta_E/1000;
deltaT_lat = deltaT_total - beta_E*E_d;
lat_grad_T = deltaT_lat/S;
path(:,3) = teta_s - path(:,2)*beta_E - lat_grad_T*path(:,1);

path(1,5) = p_sl*1000000; % задаем начальное значение давления, Па
path(1,4) = path(1,5)/(Rg*((path(1,3)+273.15))); % считаем начальное значение мольной плотности
for i = 2:301
    dp = path(i-1,4)*-1*grav*Ma; % приращение давления на 1 м, Па
    path(i,5) = path(i-1,5)+dp*(path(i,2)-path(i-1,2)); % давление на след. шаге расчета, Па
    path(i,4) = path(i,5)/(Rg*((path(i,3)+273.15))); % мольная плотность, моль/м3
end

path(:,6) = 610.6*exp(A_w*path(:,3)./(B_w + path(:,3))); % давление насыщения пара над водой
path(:,7) = 610.6*exp(A_i*path(:,3)./(B_i + path(:,3))); % и надо льдом; NB!: для T > 0 знач

path(:,8) = exp(A18_v_l./((path(:,3)+273.15).^2)-B18_v_l./(path(:,3)+273.15)-C18_v_l); % равно
path(:,9) = exp(AD_v_l./((path(:,3)+273.15).^2)-BD_v_l./(path(:,3)+273.15)+CD_v_l); % % рав
path(:,10) = exp(AD_v_scl./((path(:,3)+273.15).^2 - CD_v_scl); % для переохлажденной жидкости, т
path(:,11) = exp(A18_v_i./((path(:,3)+273.15).^2)-B18_v_i./(path(:,3)+273.15)+C18_v_i); % пар-л
path(:,12) = exp(AD_v_i./((path(:,3)+273.15).^2)-BD_v_i./(path(:,3)+273.15)+CD_v_i); % пар-л

% БЛОК 3. ОТ ПОВЕРХНОСТИ МОРЯ ДО НАСЫЩЕНИЯ

ps = path(1,6)*H_s0; % давление водяного пара в начале траектории, Па
ys = ps/path(1,5); % мольная концентрация пара в начале траектории
for i = 1:301
    if ys < path(i,6) / path(i,5)
        path(i,14) = ys; % пока не достигнуто насыщение, мольная конц-я пара равна начальной
        k = i;
    else
        break
    end
end

```

```

end
end

if ys < path(301,6) / path(301,5) % если влага так и не достигла насыщения
    path(301,30) = -1000; % приписываем изотопному составу осадков условную величину
    path(301,31) = -1000;
    path(301,32) = -1000;
    path(301,33) = -1000;
    path(301,34) = -1000;
    path(301,35) = -1000;
    return % и заканчиваем расчет
end

path(k,18) = R18_s;
path(k,21) = (path(k,18)/2005-1)*1000;
path(k,19) = RD_s;
path(k,22) = (path(k,19)/312-1)*1000;
path(k,20) = R17_s;
path(k,23) = (path(k,20)/380-1)*1000;

% БЛОК 4. ОТ НАСЫЩЕНИЯ ДО ПЕРВОЙ ПОРЦИИ ЖИДКОЙ ВЛАГИ

if L0 > 0 % если L0 задан равным 0, эту секцию просто пропускаем
    path(k+1,14) = path(k+1,6)/path(k+1,5); % концентрация пара по-прежнему равна насыщенной
    yw = path(k,14) - path(k+1,14); % первая порция влаги (снижение кол-ва водяного пара)
    L = yw/path(k+1,14); % относительное содержание влаги в воздухе (yw/yv) в первый момент
    path(k+1,15) = yw;
    path(k+1,16) = L;
    path(k+1,18) = R18_s; % изотопный состав (180) водяного пара
    path(k+1,21) = (path(k+1,18)/2005 - 1) * 1000; % d180_v
    path(k+1,20) = R17_s; % то же для 170
    path(k+1,23) = (path(k+1,20)/380 - 1) * 1000; % d170_v
    path(k+1,19) = RD_s; % то же для дейтерия
    path(k+1,22) = (path(k+1,19)/312 - 1) * 1000; % dD_v
    k1 = k + 1;
    for i = k+2:301
        if L < L0 % пока L не достигло L0
            path(i,14) = path(i,6) / path(i,5); % количество пара всегда равно насыщенному
            dyw = path(i-1,14)-path(i,14); % лишний пар уходит во влагу
            yw = yw + dyw; % влаги становится больше
            path(i,15) = yw; % записываем yw в 15-ю колонку
            L = yw / path(i,14); % L также подрастает
            path(i,16) = L; % записываем L в 16-ю колонку
            k1 = i;
            path(i,18) = R18_s; % теперь считаем изотопный состав для 180 (R18_v)
            path(i,21) = (path(i,18)/2005-1)*1000; % d180 в паре (d180_v)
            path(i,20) = R17_s; % для кислорода 17
            path(i,23) = (path(i,20)/380-1)*1000; % d170 в паре
            path(i,19) = RD_s; % для дейтерия
            path(i,22) = (path(i,19)/312-1)*1000; % dD в паре
        else
            break % если L превысит L0 - останавливаем расчет
        end
    end
end % конец расчета на 301-й строке

```

```
else
```

```
    k1 = k+1;
```

```
end
```

```
if k1 == 301
```

```
    % если модель просчитала траекторию до конца, а L так и не достигло  
    % т.е. образование осадков так и не началось
```

```
    path(301,30) = -1000;
```

```
    % приписываем изотопному составу осадков условную величину
```

```
    path(301,31) = -1000;
```

```
    path(301,32) = -1000;
```

```
    path(301,33) = -1000;
```

```
    path(301,34) = -1000;
```

```
    path(301,35) = -1000;
```

```
    return
```

```
    % и заканчиваем расчет
```

```
end
```

```
% БЛОК 5. ЖИДКИЕ ОСАДКИ
```

```
if path(i,3) > T_w
```

```
for i = k1:301
```

```
    if path(i,3) > T_w
```

```
        % если температура воздуха выше перехода от жидких к смешанным осадкам
```

```
        path(i,14) = path(i,6)/path(i,5); % концентрация пара по-прежнему равна насыщенной
```

```
        path(i,16) = L0; % L = L0
```

```
        dyv_rel = (path(i,14) - path(i-1,14))/path(i-1,14); % относительное изменение кол-ва
```

```
        path(i,24) = (dyv_rel * (path(i,8) - 1) / (1 + path(i,8)*L0))...
```

```
        * (1+path(i-1,21)/1000) * 1000; % приращение d180 в паре (отрицательное) - формула
```

```
        path(i,21) = path(i-1,21) + path(i,24); % это приращение прибавляется к d180_v
```

```
        path(i,30) = path(i,8) * (1 + L0) / (1 + path(i,8)*L0) * ...
```

```
        (path(i,21)+1000) - 1000; % d180 в осадках (нижняя часть формулы (4) у Саламатина
```

```
        path(i,26) = (dyv_rel * (path(i,8)^n - 1) / (1 + (path(i,8)^n)*L0)) ...
```

```
        * (1+path(i-1,23)/1000) * 1000; % приращение d170 в паре
```

```
        path(i,23) = path(i-1,23) + path(i,26); % это приращение прибавляется к d170_v
```

```
        path(i,32) = (path(i,8)^n) * (1 + L0) / (1 + (path(i,8)^n)*L0) ...
```

```
        * (path(i,23)+1000) - 1000; % d170 в осадках
```

```
    if path(i,3) > 0
```

```
        path(i,25) = (dyv_rel * (path(i,9) - 1) / (1 + path(i,9)*L0))...
```

```
        * (1+path(i-1,22)/1000) * 1000; % приращение dD в паре
```

```
        path(i,22) = path(i-1,22) + path(i,25); % это приращение прибавляется к dD_v
```

```
        path(i,31) = path(i,9)*(1 + L0) / (1 + path(i,9)*L0) * ...
```

```
        (path(i,22)+1000) - 1000; % dD в осадках
```

```
    else % для переохлажденной жидкости
```

```
        path(i,25) = (dyv_rel * (path(i,10) - 1) / (1 + path(i,10)*L0))...
```

```
        * (1+path(i-1,22)/1000) * 1000; % приращение dD в паре
```

```
        path(i,22) = path(i-1,22) + path(i,25); % это приращение прибавляется к dD_v
```

```
        path(i,31) = path(i,10)*(1 + L0) / (1 + path(i,10)*L0) *...
```

```
        (path(i,22)+1000) - 1000; % dD в осадках
```

```
    end
```

```
    path(k1:301,33) = path(k1:301,31) - 8 * path(k1:301,30); % dxs в осадках
```

```
    % dln по Uemura:
```

```
    path(k1:301,42) = log(1+path(k1:301,31)/1000)*1000;
```

```
    path(k1:301,43) = log(1+path(k1:301,30)/1000)*1000;
```

```

path(k1:301,34) = path(k1:301,42)-(-0.0285*((path(k1:301,43)).^2)...
    + 8.47*path(k1:301,43)); % dln Uemura
path(k1:301,35) = (log(path(k1:301,32)/1000+1) - ...
    0.528*log(path(k1:301,30)/1000+1)) * 1000000; % 170-excess в осадках
k2 = i;
else
    break % когда температура достигает T_w - заканчиваем расчет и переходим к следующему
end
end

else
    k2 = k1; % если т-ра ниже, чем T_w, то модель пропускает жидкие осадки, и сразу переходит к следующему
end

% БЛОК 6. СМЕШАННЫЕ ОСАДКИ

if k2 < 301

if path(i,3) > T_i

for i = k2+1:301
    if path(i,3) > T_i % если температура воздуха выше перехода к ледяным облакам
        L = (L0/(T_w-T_i)) * (path(i,3)-T_i); % считает эффективную L (формула 8 у Саламатина)
        path(i,16) = L;
        path(i,17) = sigma0 + ((1-sigma0)/(T_w-T_i)) * (path(i,3) - T_i); % считаем эффективную
        path(i,37) = path(i,17) * path(i,6) + (1 - path(i,17)) * path(i,7); % apparent saturation
        path(i,14) = path(i,37) / path(i,5); % концентрация пара равна насыщенной
        dyv_rel = (path(i,14) - path(i-1,14))/path(i-1,14); % относительное изменение кол-ва
        Si = 1 + path(i,17) * (path(i,6) / path(i,7) - 1); % supersaturation ratio Si (формула 10)
        path(i,38) = Si; % пишем Si в нашу табличку
        alfa_kin18 = 1 / (1 / (path(i,11)*Si) + Dif_180 * (1 - 1 / Si)); % кинетический к-т для кислорода
        alfa_kinD = 1 / (1 / (path(i,12)*Si) + Dif_D * (1 - 1 / Si)); % кинетический к-т для дейтерия
        alfa_kin17 = (path(i,11))^n * (Si/(1+path(i,11)*(Si-1)*Dif_180))^0.518; % кинетический к-т для кислорода
        dL_rel = (path(i,16) - path(i-1,16)) / (1 + path(i,16)); % = dL/(1+L)
        aux18 = Nu * path(i,8) + (1 - Nu) * alfa_kin18; % вспомогательная ф-я (Nu*alfa_w+(1-Nu)*alfa_kin18)
        auxD = Nu * path(i,10) + (1 - Nu) * alfa_kinD; % то же для дейтерия
        aux17 = Nu * (path(i,8)^n) + (1 - Nu) * alfa_kin17; % то же для кислорода 17
        aux18_2 = 1 + L*path(i,8); % еще один вспомогательный для кислорода 18
        auxD_2 = 1 + L*path(i,10); % он же для дейтерия
        aux17_2 = 1 + L*(path(i,8)^n); % он же для кислорода 17
        path(i,24) = (((alfa_kin18 + L*aux18)/aux18_2-1)*dyv_rel + ...
            dL_rel*((1+L)*aux18/aux18_2-1)) * (path(i-1,21)/1000+1) * 1000; % приращение d180_v
        path(i,21) = path(i-1,21) + path(i,24); % прибавляем эту добавку к d180_v
        path(i,25) = (((alfa_kinD + L*auxD)/auxD_2-1)*dyv_rel + ...
            dL_rel*((1+L)*auxD/auxD_2-1)) * (path(i-1,22)/1000+1) * 1000; % приращение dD_v
        path(i,22) = path(i-1,22) + path(i,25); % прибавляем эту добавку к dD_v
        path(i,26) = (((alfa_kin17 + L*aux17)/aux17_2-1)*dyv_rel + ...
            dL_rel*((1+L)*aux17/aux17_2-1)) * (path(i-1,23)/1000+1) * 1000; % приращение d170_v
        path(i,23) = path(i-1,23) + path(i,26); % прибавляем эту добавку к d170_v
        % !!!!! ниже ПРИБЛИЗИТЕЛЬНЫЙ пересчет изотопного состава осадков, НЕ как у Саламатина
        path(i,30) = ((path(i,8)*(1+L)/(1+path(i,8)*L))*...
            (path(i,3)-T_i)/(T_w-T_i) + alfa_kin18*(T_w-path(i,3)))/...
    end
end

```



```

        (T_w-T_i))*(path(i,21)+1000) - 1000; % d180 в осадках
    path(i,31) = ((path(i,10)*(1+L)/(1+path(i,10)*L))*...
        (path(i,3)-T_i)/(T_w-T_i) + alfa_kinD*(T_w-path(i,3))/...
        (T_w-T_i))*(path(i,22)+1000) - 1000; % dD в осадках
    path(i,32) = (((path(i,8)^n)*(1+L)/(1+(path(i,8)^n)*L))*...
        (path(i,3)-T_i)/(T_w-T_i) + alfa_kin17*(T_w-path(i,3))/...
        (T_w-T_i))*(path(i,23)+1000) - 1000; % d170 в осадках
    k3 = i;
    path(k2:301,33) = path(k2:301,31) - 8 * path(k2:301,30); % dxs в осадках
    % dln по Uemura:
    path(k2:301,42) = log(1+path(k2:301,31)/1000)*1000;
    path(k2:301,43) = log(1+path(k2:301,30)/1000)*1000;
    path(k2:301,34) = path(k2:301,42)-(-0.0285*((path(k2:301,43)).^2)...
        + 8.47*path(k2:301,43)); % dln Uemura
    path(k2:301,35) = (log(path(k2:301,32)/1000+1) - ...
        0.528*log(path(k2:301,30)/1000+1)) * 1000000; % 170-excess в осадках
else
    break % когда температура достигает T_i, переходим к расчету ледяных осадков
end

end

else
    k3 = k2; % если температура ниже T_i, то этот блок пропускаем и переходим к ледяным осадкам
end

else % если k2 = 301, пропускаем этот блок
    k3 = k2;
end

% БЛОК 7. ЛЕДЯНЫЕ ОСАДКИ

if k3 < 301

for i = k3+1:301
    path(i,37) = sigma0 * path(i,6) + (1 - sigma0) * path(i,7); % apparent saturation vapor pressure
    path(i,14) = path(i,37) / path(i,5); % концентрация пара равна насыщенной
    dyv_rel = (path(i,14) - path(i-1,14))/path(i-1,14); % относительное изменение кол-ва в
    Si = 1 + sigma0 * (path(i,6) / path(i,7) - 1); % supersaturation ratio Si (формула 5)
    path(i,38) = Si; % пишем Si в нашу табличку
    alfa_kin18 = 1 / (1 / (path(i,11)*Si) + Dif_180 * (1 - 1 / Si)); % кинетический к-т фракции
    alfa_kinD = 1 / (1 / (path(i,12)*Si) + Dif_D * (1 - 1 / Si)); % кинетический к-т для d
    alfa_kin17 = (path(i,11))^n * (Si/(1+path(i,11)*(Si-1)*Dif_180))^0.518; % кинетический к-т для d170
    path(i,24) = dyv_rel * (alfa_kin18 - 1) * (path(i-1,21)/1000+1) * 1000; % приращение d180_v
    path(i,21) = path(i-1,21) + path(i,24); % приращение добавляется к d180_v
    path(i,25) = dyv_rel * (alfa_kinD - 1) * (path(i-1,22)/1000+1) * 1000; % приращение dD_v
    path(i,22) = path(i-1,22) + path(i,25); % приращение добавляется к dD_v
    path(i,26) = dyv_rel * (alfa_kin17 - 1) * (path(i-1,23)/1000+1) * 1000; % приращение d170_v
    path(i,23) = path(i-1,23) + path(i,26); % приращение добавляется к d170_v
    path(i,30) = alfa_kin18 * (path(i,21)+1000) - 1000; % d180 в осадках
    path(i,31) = alfa_kinD * (path(i,22)+1000) - 1000; % dD в осадках
    path(i,32) = alfa_kin17 * (path(i,23)+1000) - 1000; % d170 в осадках
end

```



```

end

else
end

path(k3:301,33) = path(k3:301,31) - 8 * path(k3:301,30); % dxs в осадках
% dln по Uemura:
path(k3:301,42) = log(1+path(k3:301,31)/1000)*1000;
path(k3:301,43) = log(1+path(k3:301,30)/1000)*1000;
path(k3:301,34) = path(k3:301,42)-(-0.0285*((path(k3:301,43)).^2) +...
    8.47*path(k3:301,43)); % dln Uemura
path(k3:301,35) = (log(path(k3:301,32)/1000+1) - ...
    0.528*log(path(k3:301,30)/1000+1)) * 1000000; % 170-excess в осадках

% БЛОК 8. ПРОСТРАНСТВЕННЫЕ ГРАДИЕНТЫ ИЗОТОПНОГО СОСТАВА

% логический массив: 0 - т-ра выше T_i, 1 - т-ра ниже T_i:
z = path(1:301,3) < T_i;
% номера строк таблицы path, для которых т-ра ниже T_i:
idx = find(z);
% для т-ры ниже T_i считаем linear fit d180 vs teta_s:
fitd180_T = polyfit(path(idx,3),path(idx,30),1);
gradd180_T = fitd180_T(1,1); % берем slope этого fit
% логический массив, смотрим в каких строчках d180 в осадках < -40 промилле
z1 = path(1:301,30) < -40;
idx1 = find(z1); % номера строк таблицы path, для которых d180 ниже -40
% для d180 ниже -40 считаем linear fit dxs vs d180
fitdxs_d180 = polyfit(path(idx1,30),path(idx1,33),1);
graddxs_d180 = fitdxs_d180(1,1); % берем slope этого fit
% для d180 ниже -40 считаем linear fit 170-excess vs d180
fit170excess_d180 = polyfit(path(idx1,30),path(idx1,35),1);
grad170excess_d180 = fit170excess_d180(1,1); % берем slope этого fit

```