Scale growth and isotope diffusion - version 7

In [1]:
```
version  = 7
```

This code was written to solve a fully one-dimensionsal version of the Mishin and Borchardt (MB) equations. Starting with version 7 it is becoming a 2D version, to include diffusion into the grains from the grain boundaries.

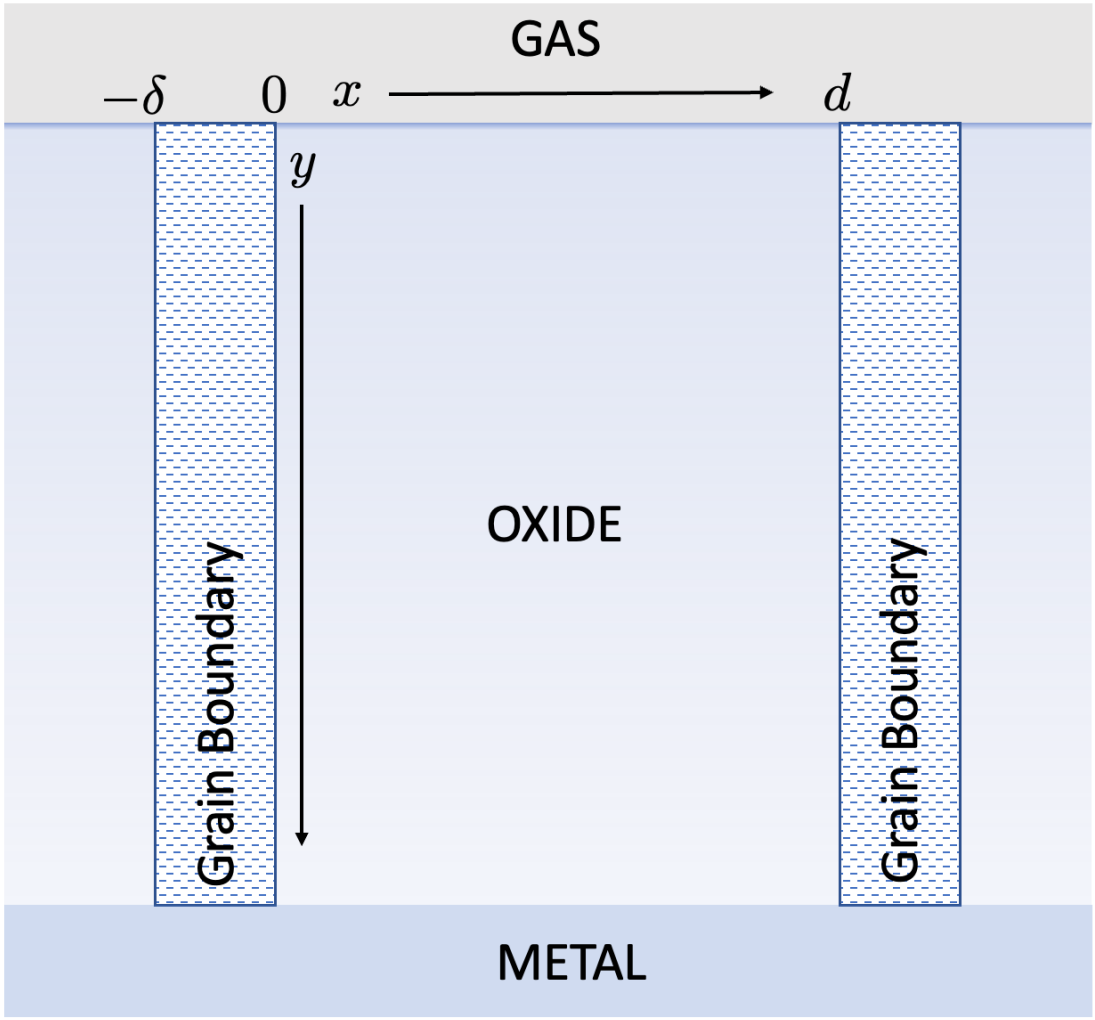
Their variable c' is the excess concentration of an isotope (we assume ^{18}O) on the grain boundary (GB) perpendicular to the interfaces. In this code c' is called c_GB.

Changes to version 6 (tbc):\ Replace te simple bleeding model by proper finite-difference diffusion into the bulk (x-direction)\ Update plotting function to save 6 figures to a page. \ Update documentation cell.

Changes to version 5:\ Introduce some bleeding of the isotope into the bulk (x-direction)\ Update plotting function to save numbered figures to pdfs.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import sys
```

Diagram.png



Simplest approach first. The 1-D equation with only oxygen transport, and only on GB, looks like (MB-I,17}

$$\frac{\partial c'}{\partial t} = \frac{\partial}{\partial y}\left(D'\frac{\partial c'}{\partial y}\right) - V'\frac{\partial c'}{\partial y}$$

where

$$V' = \frac{J'_{\mathrm{O}}}{c_{\mathrm{O}}} = \frac{1}{kTL}\int_{\mu_L}^{\mu_0} D'_{\mathrm{O}}\mathrm{d}\mu$$

and

$$D' = f'D'_{\mathrm{O}}$$

and

$$f' = (1 - c'_{\mathrm{O}}/c'_{\mathrm{O}p})$$

Maintaining the fiction that GB transport is by vacancy hopping, f' is the fractional occupation of oxygen sites by vacancies, thus $c_{\mathrm{O}p}$ is the density of oxygen in the perfectly stoichiometric boundary.

If we assume zero divergence of the flux, $D'_{\mathrm{O}} = constant$ and $\mu = kT\ln c'_{\mathrm{O}}$ we can write the flux above as

$$J'_{\mathrm{O}} = D'_{\mathrm{O}}(c'_{\mathrm{O}}(0) - c'_{\mathrm{O}}(L))/L.$$

In terms of the fractional occupancy of oxygen sites by vacancies (effectively zero at the scale surface) this can be expressed as

$$J'_{\mathrm{O}} = D'_{\mathrm{O}}\frac{3f'(L)}{\Omega L}.$$

NOTE: This is a bad notation because $f'(L)$ is constant, the value at the scale-metal boundary, even as L increases with t.

Boundary conditions:

$$c'(0,t) = 1.0$$

$$\frac{\partial c'(y=1,t)}{\partial y} = 0$$

$$f'(0) = 0$$

$$f'(L) = parameter$$

where $f'(L)$ will be set to a small oxygen vacancy site occupancy at the scale-metal interface. We can assume there are practically no vacancies on the outside of the scale compared to the oxide-metal interface.

In order to scale the length with time, we need the velocities of the interfaces. With only oxygen diffusion considered, the velocity V' given above is all we need. MB defined the velocity of the oxide-metal interfac (MB-I-6) as:

$$V_2 = [\nu j'_{\mathrm{O}} + (1-\nu) j_{\mathrm{O}}]/c_{\mathrm{O}},$$

where ν is the fraction of the width in the x-direction occupied by grain boundary, and the two fluxes refer to grain boundary and bulk respectively. c_{O} stands for the concentration of oxygen atoms per unit volume. I have used a different notation here, and start by ignoring the bulk transport altogether, so in the notation of the code $j'_{\mathrm{O}} \rightarrow$ j_O_GB. A similar equation applies to the gas-scale interface with O replaced by Al. Our rest frame of reference is the alumina lattice, on which the majority of ions at any instant are vibrating but not contributing to the fluxes. We are simulating on a grid (a numpy array) of fixed nodes separated by dy, so we need to discover how many timesteps, delta_n_t, each of size dt, are needed for the scale to advance by dy, at which point we have to extend by one node the segment of nodes that define the scale within the fixed array, occupied by alumina, by one element. The velocity of the interface will be

$$\mathrm{V_2} = \frac{\mathrm{dl}}{\mathrm{delta\_n\_t\,dt}}$$

so

$$\mathrm{delta\_n\_t} = \frac{\mathrm{d}l}{\mathrm{d}t}\frac{c_{\mathrm{O}}}{\nu j'_{\mathrm{O}}} = \frac{\mathrm{d}l}{\mathrm{d}t}\frac{3}{\nu j'_{\mathrm{O}}\Omega} = \frac{\mathrm{d}l}{\mathrm{d}t}\frac{L}{\nu D'_{\mathrm{O}} f'(L)}$$

where Ω is the volume of an Al_2O_3 molecular unit in the scale, neglecting the small concentration of defects. Within this simple model we can can also express V_2 as

$$V_2 = \frac{\nu D'_{\mathrm{O}} f'(L)}{L(t)}$$

The reference position for velocities here is the location of the oxygen lattice, which is the location of the original slab of oxide scale. So V_2 is positive, because the metal substrate is moving to the right! In a similar way, with respect to the oxygen lattice, the position of the scale gas interface is moving to the left, so its velocity V_1 is negative, given by:

$$V_1 = -\frac{\nu D'_{\mathrm{Al}} f'(\mathrm{Al})}{L(t)},$$

where $f'(\mathrm{Al})$ will be set to the Al vacancy concentration at the scale-gas boundary, and like $f'(L)$ it is a constant, independent of L. \ The parabolic growth law follows from the rate of change of L:

$$\frac{\mathrm{d}L}{\mathrm{d}t} = V_2 - V_1 = \frac{\nu(D'_{\mathrm{O}} f'(L) + D'_{\mathrm{Al}} f'(\mathrm{Al}))}{L(t)}.$$

So if we start the clock at time $t = 0$ when the scale thickness is L_0 we have

$$L(t) = \sqrt{L_0^2 + 2\nu t[D'_{\mathrm{O}} f'(L) + D'_{\mathrm{Al}} f'(\mathrm{Al})]}$$

INTERPRETATION OF EXPERIMENTS

If we can measure separately the thickness of new scale formed at the metal and at the gas interfaces, call it l_2 and l_1 respectively, where

$$L(t) = L_0 + l_1(t) + l_2(t),$$

it is easy to show that this simple model makes the prediction:

$$\frac{l_2(t)}{l_1(t)} = \frac{D'_{\mathrm{O}} f'(L)}{D'_{\mathrm{Al}} f'(\mathrm{Al})}.$$

Thus the ratio of thicknesses of the new scale is independent of time. These equations might enable us to extract the products $\nu(D'_{\mathrm{O}} f'(L)$ and $D'_{\mathrm{Al}} f'(\mathrm{Al}))$ from experimental data, but the factors ν and the "vacancy concentrations" $f'(L)$ and $f'(\mathrm{Al})$ would remain undetermined. Perhaps by inserting reasonable values of these parameters the unreasonableness of the simple model could be established, and alternative transport mechanisms considered.

DIMENSIONLESS UNITS

It is very convenient, for generality of interpretation as well as for coding, to work with dimensionless variables, such that:

Unit of length will be initial thickness of the scale, $L(0)$ referred to in the code as length_0

Unit of time will be $L(0)^2/D'_{\mathrm{O}}$ referred to in the code as length_0^2/mobility_Ox_GB

In these dimensionless units, V_2 and delta_n_t are simply

$$V_2 = \nu f'(L(0))/L$$

referred to in the code as $\mathrm{v\_2} = \mathrm{nu*f\_Ox\_GB}$ initially, and this decreases by a faxtor $L(0)/L(t)$ as the thickness of the scale increases. We expect the scale to grab a node from the metal after a number of time given roughly by

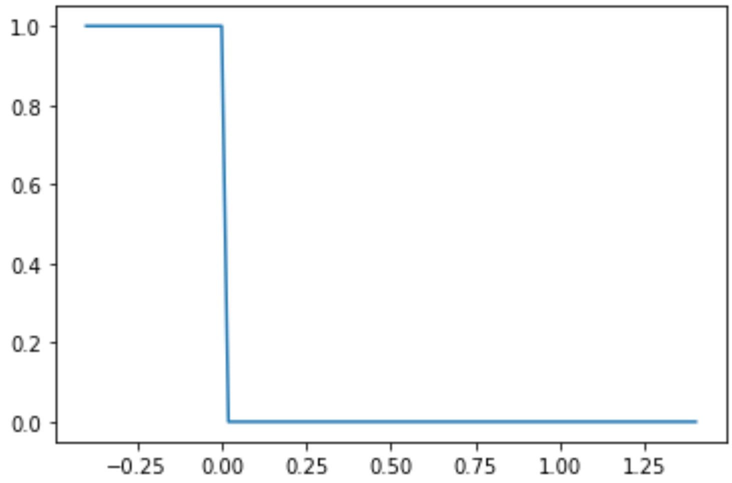
$$\mathrm{delta\_n\_t} = dl/V_2$$

In [3]:
```
"""
Initialize global variables

"""
# Initial scale thickness, the length scale for dimensionless equations,
# so not necessarily referenced in code.
length_0 = 1.0
# Oxygen diffusion coeffient $D_{\text{O}}'$ on the grain boundary as defi
# Should always be 1.0 for dimensionless equations, so not necessarily re
mobility_O_GB = 1.0
# Aluminium diffusion coeffient $D_{\text{Al}}'$ on the grain boundary as
```

```
# Aluminium diffusion coefficient $D_{\text{Al}}$ on the grain boundary as
# represented here as its ratio to $D_{\text{O}}'$.
mobility_Al_GB = 1.0
# Initialize the number of nodes of each type on the grid along the y-axi
# Number of nodes for growth into gas phase
ngridy_gas_0 = 20
# Number of nodes covering initial scale
ngridy_scale_0 = 51
# Number of nodes for growth into metal phase
ngridy_metal_0 = 20
# Fixed total provision of nodes for  gas-scale-metal. This will not be c
ngridy_max = ngridy_scale_0 + ngridy_gas_0 + ngridy_metal_0
# Ratio nu of thickness of grain boundary delta to (delta + d)
nu = 0.01
# f' as defined by MB, or the oxygen vacancy concentration at the oxide-m
f_Ox_GB = 0.0001
# We define here the notionally similar quantity for Al, which would be t
# concentration at the gas-scale interface. This is only used to scale th
# gas-scale interface
f_Al_GB = 0.0001
# Initial velocity of scale-metal interface (> 0)
# Note - velocities are measured with respect to the fixed lattice of the
# So we imagine that as the scale grows it pushes the interfaces back in
v_2_0 = nu*f_Ox_GB
# Initial velocity of gas-scale interface (< 0)
v_1_0 = - nu*f_Al_GB*mobility_Al_GB
# Distances along the y-axis for the fixed array of nodes to cover the th
# The origin (y = 0) is chosen to be the first node of the initial scale,
# and y=1 is the initial position of the last node of the scale.
# These distances are set up here and should not be changed by the subseq
gridy_max = (np.linspace(0,ngridy_max-1,ngridy_max)-ngridy_gas_0)/(ngridy
# Increment of distances, a constant
dy = gridy_max[1]-gridy_max[0]
# Initial excess oxygen isotope.
# The gas is treated as having an isotope fraction of 1.
c_GB_0 = np.zeros_like(gridy_max)
c_GB_0[0:ngridy_gas_0+1] = 1.0
np.set_printoptions(precision=6)
print('gridy_max =\n', np.array2string(gridy_max))
```

In [4]:
```
ngridy_gas = ngridy_gas_0
ngridy_scale = ngridy_scale_0
ngridy_metal = ngridy_metal_0
c_GB = np.copy(c_GB_0)
c_bulk = np.copy(c_GB_0)
```

In [5]:
```
fig = plt.figure()
"""
The initial fraction of isotope

"""
ax = fig.add_subplot()
line_c_GB_0 = ax.plot(gridy_max, c_GB_0)
plt.show()
```



In [6]:

```
"""
DEFINE TRACER  DIFFUSION COEFFICIENT AND ITS DERIVATIVE

"""
# Oxygen tracer diffusion coefficient d_GB and its linear y-derivative dd
d_GB = np.zeros_like(gridy_max)
dd_GB = np.zeros_like(gridy_max)
for ny in range(ngridy_gas+1,ngridy_gas+ngridy_scale):
    d_GB[ny] = vac_Ox[ny]
    dd_GB[ny] = (vac_Ox[ny+1]-vac_Ox[ny-1])/(2.0*dy)
# Correct the last value of this gradient, else it would be negative and
dd_GB[ngridy_gas+ngridy_scale-1] = dd_GB[ngridy_gas+ngridy_scale-2]
print('\nTracer diffusion coefficient:\n',d_GB)
print('\ny-derivative of tracer diffusion coefficient:\n',dd_GB)
```

```
Tracer diffusion coefficient:
 [0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00
 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00
 0.0e+00 0.0e+00 0.0e+00 2.0e-06 4.0e-06 6.0e-06 8.0e-06 1.0e-05 1.2e-05
 1.4e-05 1.6e-05 1.8e-05 2.0e-05 2.2e-05 2.4e-05 2.6e-05 2.8e-05 3.0e-05
 3.2e-05 3.4e-05 3.6e-05 3.8e-05 4.0e-05 4.2e-05 4.4e-05 4.6e-05 4.8e-05
 5.0e-05 5.2e-05 5.4e-05 5.6e-05 5.8e-05 6.0e-05 6.2e-05 6.4e-05 6.6e-05
 6.8e-05 7.0e-05 7.2e-05 7.4e-05 7.6e-05 7.8e-05 8.0e-05 8.2e-05 8.4e-05
 8.6e-05 8.8e-05 9.0e-05 9.2e-05 9.4e-05 9.6e-05 9.8e-05 1.0e-04 0.0e+00
 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00
 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00 0.0e+00
 0.0e+00]

y-derivative of tracer diffusion coefficient:
 [0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00
 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00
 0.e+00 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04
 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04
 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04
 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04
 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04 1.e-04
 1.e-04 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00
 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00 0.e+00
 0.e+00]
```

In [7]:
```
def update_ranges(direction, ngridy_gas, ngridy_scale, ngridy_metal,c_GB,
    """
    Increment the numbers of nodes in each region, ngridy_gas, ngridy_sca
    according to the direction of scale growth.
    The overall number of nodes remains fixed. Possible parameter values:
    direction = +1 : The scale grows by 1 node while the metal retreats b
    direction = -1: The scale grows by 1 node while the gas retreats by 1
    Then update the other parameters, c_GB, vac_Ox, d_GB, dd_GB, because
    of the scale.

    """
    print('\n*** ENTERING update_ranges ***\n')
    print('Previous values of ngridy_gas, ngridy_scale, ngridy_metal',ngr
    if direction != 1 and direction != -1:
        print('Argument of update_ranges = ', direction)
        sys.exit('Argument of update_ranges must be -1 or +1')
    if direction == 1:
        c_GB[ngridy_gas+ngridy_scale] = c_GB[ngridy_gas+ngridy_scale-1]
        ngridy_metal = ngridy_metal-1
        if ngridy_metal == 0:
            raise ValueError('RUN OUT OF METAL NODES! Increase ngridy_met
        print('One node to be added to scale grid and taken from metal gr
    if direction == -1:
        ngridy_gas = ngridy_gas-1
        if ngridy_gas == 0:
            raise ValueError('RUN OUT OF GAS NODES! Increase ngridy_gas_0
        print('One node to be added to scale grid and taken from gas grid
    ngridy_scale = ngridy_scale+1
    for ny in range(ngridy_gas + 1,ngridy_gas + ngridy_scale):
        vac_Ox[ny] = f_Ox_GB*(gridy_max[ny]-gridy_max[ngridy_gas])*((ngri
    for ny in range(ngridy_gas + 1,ngridy_gas + ngridy_scale):
        d_GB[ny] = vac_Ox[ny]
        dd_GB[ny] = (vac_Ox[ny+1]-vac_Ox[ny-1])/(2.0*dy)
# Correct the last value of this gradient, else it would be negative and
    dd_GB[ngridy_gas+ngridy_scale-1] = dd_GB[ngridy_gas+ngridy_scale-2]
#   print('vac_Ox :\n',vac_Ox,'\n')
#   print('c_GB :\n',c_GB,'\n')
    print('New values of ngridy_gas, ngridy_scale, ngridy_metal = ',ngrid
#   print('\nTracer diffusion coefficient:\n',d_GB)
#   print('\ny-derivative of tracer diffusion coefficient:\n',dd_GB,'\n')
    if direction==1:
        print('\n*** LEAVING update_ranges, scale node added in metal dir
    if direction==-1:
        print('\n*** LEAVING update_ranges, scale node added in gas direc
    return ngridy_gas, ngridy_scale, ngridy_metal, c_GB, vac_Ox, d_GB, dd
```

In [8]:
```
def tracer_diffusion(timesteps,dt, ngridy_gas, ngridy_scale, ngridy_metal
    """
    Test simple tracer diffusion, ideal solution, vacancy mechanism. Dime
    All locally varying quantities are defined on a fixed array, gridy_ma
    timesteps    : maximum number of timesteps to iterate
    dt           : the length of a timestep
    ngridy_gas   : number of nodes in the gas (on the y axis)
    ngridy_scale : number of nodes in the scale
    ngridy_metal : number of nodes in the metal
    vac_Ox[]     : concentration per site of oxygen vacancies
    c-GB[]       : local grain-boundary ratio of isotope to total oxygen
    d_GB[]       : local oxygen tracer diffusion coefficient
    dd_GB[]      : y-derivative of d_GB[]
    bleed        : parameter to describe rate of bleeding of c_GB into bu

    """
    print('\n*** STARTING tracer_diffusion ***\n')
# Dimensionless oxygen current J_\text{O}_GB, depends only on time-depend
# Note that j_O_GB = f_Ox_GB/length
# length should start with value 1, but will be incremented in the time l
#
    print('timesteps = ',timesteps,'  dt = ',dt,'\ndy =',dy,'\n')
    print('ngridy_gas =',ngridy_gas)
    print('ngridy_scale =',ngridy_scale)
    print('ngridy_gas =',ngridy_metal)
# The inflation factor must be <1 for convergence of the Euler method.
    print('Inflation factor = ', 2*dt*f_Ox_GB/(dy*dy))
    print('A near-optimal timestep would be ', dy*dy/(2.01*f_Ox_GB))
#    c_GB_temp = np.copy(c_GB)
# length should start with value 1, but will be incremented in the time l
    length = length_0
# The distance moved by the scale-metal interface since the last update o
# When delta_length reaches dl it's time to redefine the next gridpoint a
    delta_length_1 = 0.
    delta_length_2 = 0.
# Counter for iterations of the time loop between the single node transfe
    t_count_1 = 0
# Counter for iterations of the time loop between the single node transfe
    t_count_2 = 0
# Initial velocity of gas-scale interface ( < 0 )
    v_1 = v_1_0
# Initial velocity of scale-metal interface ( > 0 )
    v_2 = v_2_0
    if v_2*dt > dy:
         print('Error, length_increment will exceed grid spacing! v_1 =',
    length_increment_1 = v_1*dt
    length_increment_2 = v_2*dt
# t_count_1 counts the number of timesteps executed before extension of s
    t_count_1 = 0
# t_count_2 counts the number of timesteps executed before extension of s
    t_count_2 = 0
# Counts how many times a node is added to the scale
    n_nodes_added = 0
    """
    The main time loop starts here

    """
    for t in range(1,timesteps+1):
        if delta_length_2 < dy-length_increment_2:
            for ny in range(ngridy_gas_0+1,ngridy_gas+ngridy_scale):
                c_GB[ny] = ( c_GB[ny] + (dd_GB[ny]- f_Ox_GB)*dt*((c_GB[ny
                             d_GB[ny]*dt*(c_GB[ny-1]+c_GB[ny+1]-2*c_GB[ny
                temp = bleed*(c_GB[ny]-c_bulk[ny])
```

```
                temp = bleed*(c_GB[ny]-c_bulk[ny])
                c_GB[ny] = c_GB[ny]-temp*dt
                c_bulk[ny] = c_bulk[ny] + temp*nu*dt
# Apply zero gradient boundary condition at th = e scale metal interface
            c_GB[ngridy_gas+ngridy_scale-1] = c_GB[ngridy_gas+ngridy_scal
# Increment the notional thickness of the scale
# This is increment of scale thickness due to the velocity of the scale-m
            length_increment_2 = v_2*dt
            length = length+length_increment_2
            delta_length_2 = delta_length_2+length_increment_2
            v_2 = v_2_0/length
            t_count_2 = t_count_2 + 1
# loop back unless it's time to grab another node of scale from the metal
        else:
            print('\n* Seems like time to add a scale node from metal *\n
            print('\nt_count_2 =',t_count_2 )
#           print('\nc_GB :\n',c_GB,'\n')
            print('delta_length_2 =', delta_length_2,  't =', t , 't_coun
            delta_length_2 = 0.0
            ngridy_gas, ngridy_scale, ngridy_metal, c_GB, vac_Ox, d_GB, d
                update_ranges(1, ngridy_gas, ngridy_scale, ngridy_metal,
            n_nodes_added = n_nodes_added + 1
#           print('c_GB[ngridy_gas-1]',c_GB[ngridy_gas-1])
#           print('c_GB[ngridy_gas]',c_GB[ngridy_gas])
#           print('c_GB[ngridy_gas+ngridy_scale]',c_GB[ngridy_gas+ngridy_
#           print('c_GB[ngridy_gas+ngridy_scale-1]',c_GB[ngridy_gas+ngrid
#           print('c_GB[ngridy_gas+ngridy_scale-2]',c_GB[ngridy_gas+ngrid
#           print('\nlength = ',length)
            print('\nExpected number of timesteps before incrementing sca
            print(' based on initial increment only = ',int(dy/(v_2*dt)),
            print(' Number of nodes added to scale :',n_nodes_added)
            t_count_2=0
            graphs = plot_concentrations('1.{}'.format(n_nodes_added), 3,
                title='t = {},  bleed = {}'.format(t,bleed), c_1=c_GB, c_
                label_2='bulk isotope frac')
        if delta_length_1 < dy-length_increment_1:
            length_increment_1 = - v_1*dt
            length = length+length_increment_1
            delta_length_1 = delta_length_1+length_increment_1
            v_1 = v_1_0/length
            t_count_1 = t_count_1 + 1
        else:
            print('\n* Seems like time to add a scale node from gas *\n')
            print('\nt_count_1 =',t_count_1 )
#           print('\nc_GB :\n',c_GB,'\n')
            print('delta_length_1 =', delta_length_1,  't =', t , 't_coun
            delta_length_1 = 0.0
            ngridy_gas, ngridy_scale, ngridy_metal, c_GB, vac_Ox, d_GB, d
                update_ranges(-1, ngridy_gas, ngridy_scale, ngridy_metal,
            n_nodes_added = n_nodes_added + 1
#           print('c_GB[ngridy_gas-1]',c_GB[ngridy_gas-1])
#           print('c_GB[ngridy_gas]',c_GB[ngridy_gas])
#           print('c_GB[ngridy_gas+ngridy_scale]',c_GB[ngridy_gas+ngridy_
#           print('c_GB[ngridy_gas+ngridy_scale-1]',c_GB[ngridy_gas+ngrid
#           print('c_GB[ngridy_gas+ngridy_scale-2]',c_GB[ngridy_gas+ngrid
#           print('\nlength = ',length)
            print('\nExpected number of timesteps before adding node to s
            print(' based on initial increment only = ',-int(dy/(v_1*dt))
            print(' Number of nodes added to scale :',n_nodes_added)
            t_count_1=0
            graphs = plot_concentrations('1.{}'.format(n_nodes_added), 3,
                title='t = {},  bleed = {}'.format(t,bleed), c_1=c_GB, c_
                label_2='bulk isotope frac')
    print('\n*** LEAVING tracer diffusion ***\n')
```

```
    return t, t_count_1, t_count_2, ngridy_gas, ngridy_scale, ngridy_meta
```

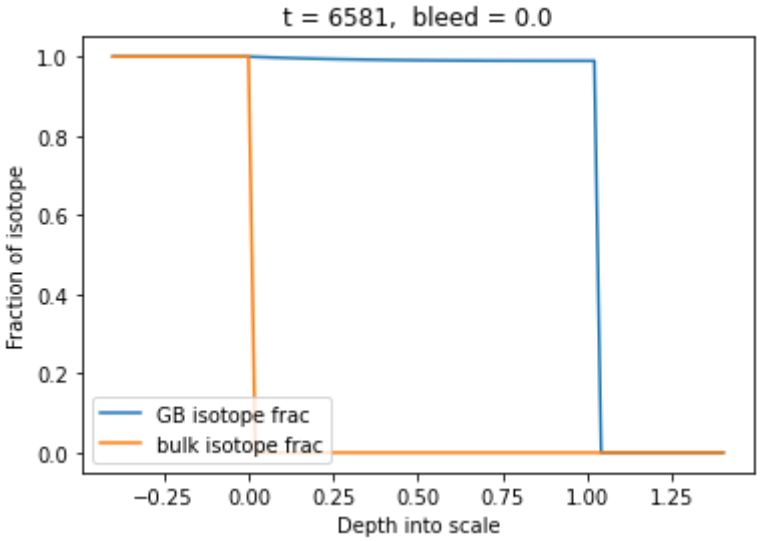
In [9]:

```
def diffusion_test(timesteps,dt, ngridy_gas, ngridy_scale, ngridy_metal,
    """
    Test simple Fickian diffusion with unit oxygen diffusion coefficient

    """
# This function also evaluates for test purposes the velocity of the scal
    print('\n*** STARTING Fick Law diffusion_test ***\n')
    print('timesteps = ',timesteps,'  dt = ',dt,'\n')
# The inflation factor must be <1 for convergence of this Euler method.
    print('Inflation factor = ', 2*dt/(dy*dy))
    print('A near-optimal timestep would be ', dy*dy/(2.01))
# length should start with value 1, but will be incremented in the time l
    length = length_0
# velocity of the scale-metal interface
    v_2_0 = nu*f_Ox_GB/length
    v_2 = v_2_0
# The increment of scale thickness due to the velocity of the scale-metal
# updated in the time loop.
# The distance moved by the scale-metal interface since the last update o
# When delta_length reaches dl it's time to redefine the next gridpoint a
    delta_length = 0.
    t_count=0
#
    length_increment = v_2*dt
# The main TIME LOOP starts here
    for t in range(1,timesteps+1):
        if delta_length < dy-length_increment:
            for ny in range(ngridy_gas+1,ngridy_gas+ngridy_scale):
                c_GB2[ny] = c_GB2[ny] + dt*(c_GB2[ny-1]+c_GB2[ny+1]-2*c_G
# Apply zero gradient boundary condition at the scale-metal interface
            c_GB2[ngridy_gas+ngridy_scale-1] = c_GB2[ngridy_gas+ngridy_sc
# Increment the notional thickness of the scale
# This is increment of scale thickness due to the velocity of the scale-m
            length_increment = v_2*dt
            length = length+length_increment
            delta_length = delta_length+length_increment
            v_2 = v_2_0/length
            t_count = t_count+1
# loop back unless it's time to grab another node of scale from the metal
        else:
            print('\n* Seems like time to add a scale node *\n')
            print('\nt_count =',t_count )
#           print('\nc_GB :\n',c_GB,'\n')
            print('delta_length =', delta_length,  't =', t , 't_count ='
            delta_length_2 = 0.0
            ngridy_gas, ngridy_scale, ngridy_metal, c_GB2, vac_Ox, d_GB,
                update_ranges(1, ngridy_gas, ngridy_scale, ngridy_metal,
            print('ngridy_gas, ngridy_scale, ngridy_metal = ',ngridy_gas,
            print('vac_Ox =\n',vac_Ox)
            print('\nc_GB2[ngridy_gas-1]',c_GB2[ngridy_gas-1])
            print('c_GB2[ngridy_gas]',c_GB2[ngridy_gas])
            print('c_GB2[ngridy_gas+ngridy_scale]',c_GB2[ngridy_gas+ngrid
            print('c_GB2[ngridy_gas+ngridy_scale-1]',c_GB2[ngridy_gas+ngr
            print('\nlength = ',length)
            print('\nExpected number of timesteps before adding node to s
            print(' based on initial increment only = ',int(dy/(v_2*dt)),
            t_count=0
    print('\n*** LEAVING diffusion_test ***\n')
    return t, t, ngridy_gas, ngridy_scale, ngridy_metal, c_GB2
```

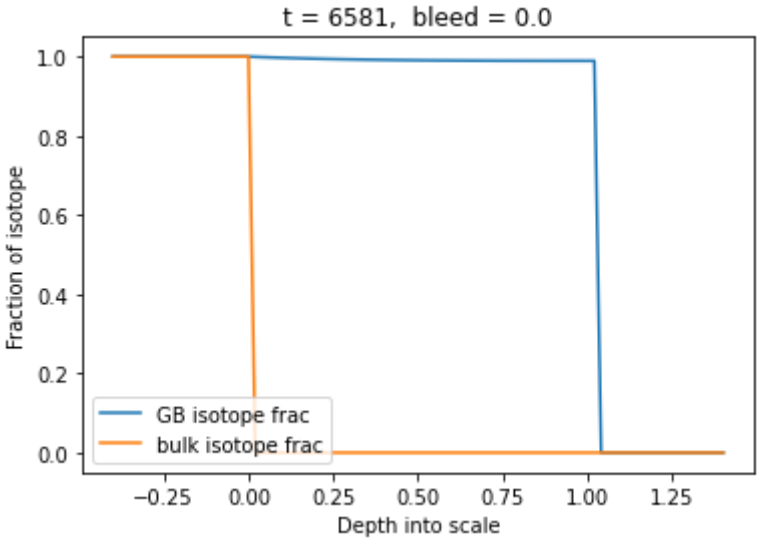
In [10]:
```
def plot_concentrations(fig_number,first_or_second, title = 'Scale growth
    """
    If first_or_second = 1, plot first graph of c_1
    If first_or_second = 2,  plot second graph of c_2
    If first_or_second = anything else, plot both graphs

    """
    fig1 = plt.figure(fig_number)
    ax = fig1.add_subplot()
    if first_or_second == 1:
        line_c_1 = ax.plot(gridy_max, c_1,label=label_1)
    elif first_or_second == 2:
        line_c_2 = ax.plot(gridy_max, c_2,label=label_2)
    else:
        line_c_1 = ax.plot(gridy_max, c_1,label=label_1)
        line_c_2 = ax.plot(gridy_max, c_2,label=label_2)
    plt.legend(loc="lower left")
    plt.xlabel('Depth into scale', fontdict=None, labelpad=None        )
    plt.ylabel('Fraction of isotope', fontdict=None, labelpad=None    )
    plt.title(title)
    plt.savefig('Figure_{}.pdf'.format(fig_number))
    return plt.show()
```

In [11]:
```
"""
MAIN CODE FOR EXECUTION

"""
t,  t_count_1, t_count_2, ngridy_gas, ngridy_scale, ngridy_metal, c_GB, c
tracer_diffusion(50000, 3.1, ngridy_gas, ngridy_scale, ngridy_metal, vac_
fig_1 = plot_concentrations(101,3, c_1=c_GB, c_2=c_bulk, label_1='GB isot
fig_2 = plot_concentrations(102,1, c_1=c_GB, c_2=c_bulk, label_1='GB isot
print('c_GB =\n',c_GB)
#
#
# c_GB2 is only used here for test purposes with Fick's Law diffusion.
# c_GB2 = np.copy(c_GB_0)
# t, t_count_2,  ngridy_gas, ngridy_scale, ngridy_metal, c_GB2 = \
# diffusion_test(5000,0.00019, ngridy_gas, ngridy_scale, ngridy_metal, va
# print('\nTotal timesteps taken = ',t, '\nSince final growth step, t_cou
# print('Total change in nodes per phase:')
# print ('gas:',ngridy_gas-ngridy_gas_0, ' scale:',ngridy_scale-ngridy_sc
#       ' metal:',ngridy_metal-ngridy_metal_0,'\n')
# print(' C_GB2 =', c_GB2)
# graphs = plot_concentrations(1,c_1=c_GB,c_2=c_GB2, label_1='isotope fra
```

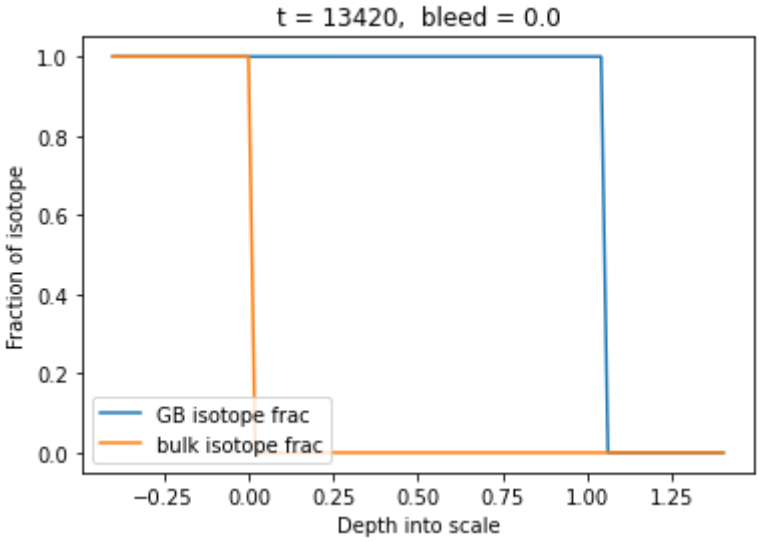
```
*** STARTING tracer_diffusion ***

timesteps =  50000   dt =  3.1
dy = 0.020000000000000018

ngridy_gas = 20
ngridy_scale = 51
ngridy_gas = 20
Inflation factor =  1.5499999999999972
A near-optimal timestep would be  1.990049751243785

* Seems like time to add a scale node from metal *


t_count_2 = 6580
delta_length_2 = 0.019998194389471285 t = 6581 t_count_2 = 6580


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 20 51 20
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  20 52 19

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  6709  Actual number =  6580

 Number of nodes added to scale : 1
```



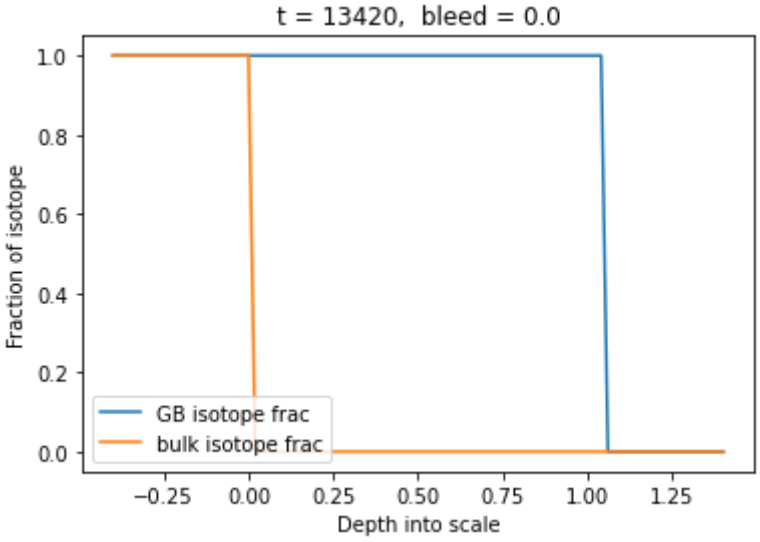
```
* Seems like time to add a scale node from gas *


t_count_1 = 6580
delta_length_1 = 0.01999813478815728 t = 6581 t_count_1 = 6580


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 20 52 19
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  19 53 19

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  6709  Actual number =  6580

 Number of nodes added to scale : 2
```



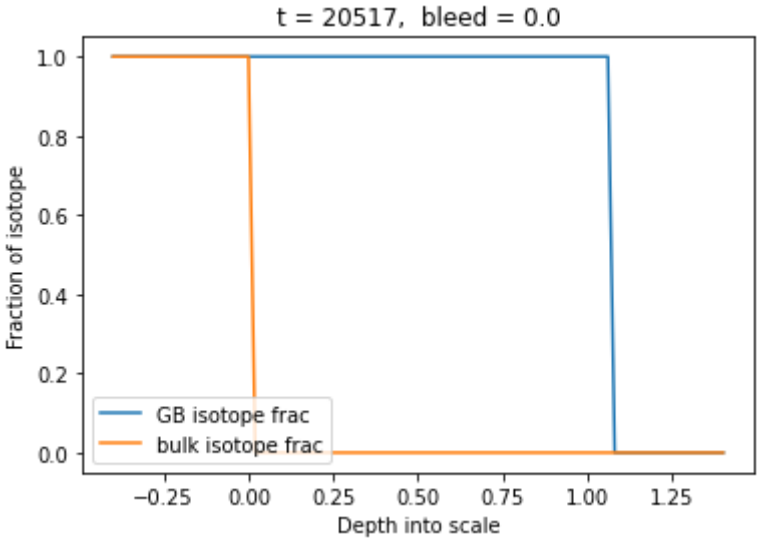

```
* Seems like time to add a scale node from metal *


t_count_2 = 6838
delta_length_2 = 0.019998139768959634 t = 13420 t_count_2 = 6838


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 19 53 19
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  19 54 18

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  6967  Actual number =  6838

 Number of nodes added to scale : 3
```



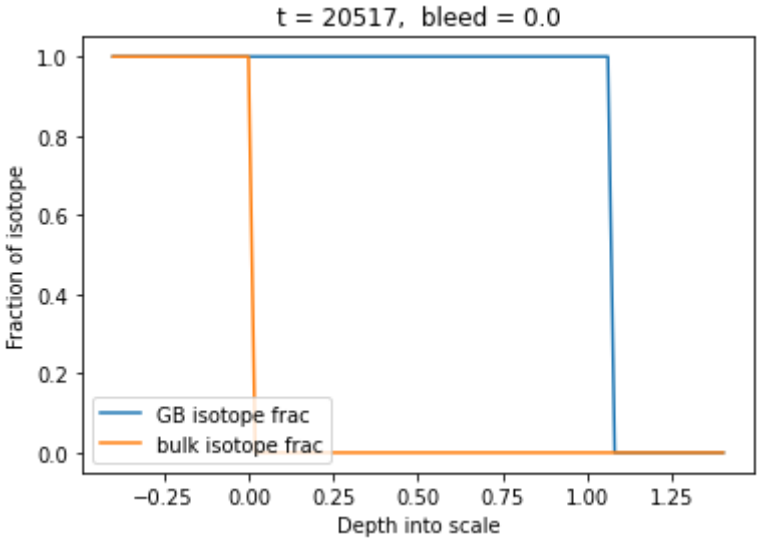

```
* Seems like time to add a scale node from gas *


t_count_1 = 6838
delta_length_1 = 0.019998084573478615 t = 13420 t_count_1 = 6838


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 19 54 18
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  18 55 18

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  6967  Actual number =  6838

 Number of nodes added to scale : 4
```



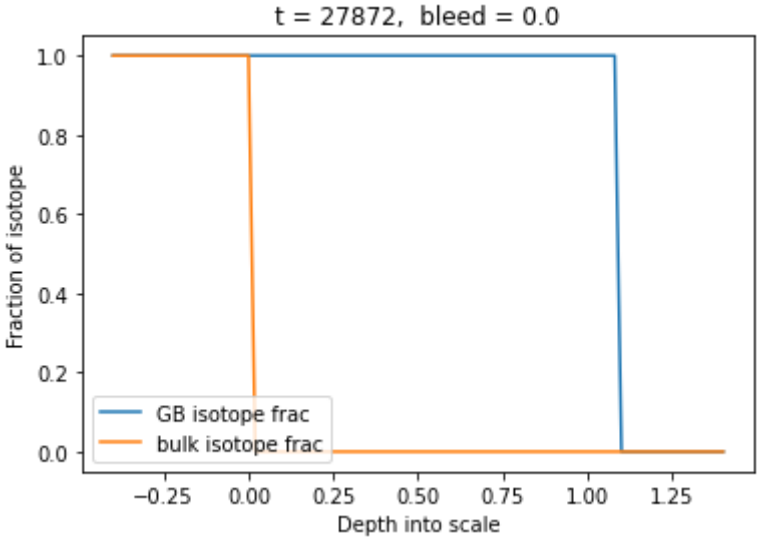
```
* Seems like time to add a scale node from metal *


t_count_2 = 7096
delta_length_2 = 0.019998091215162533 t = 20517 t_count_2 = 7096


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 18 55 18
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  18 56 17

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  7225  Actual number =  7096

 Number of nodes added to scale : 5
```



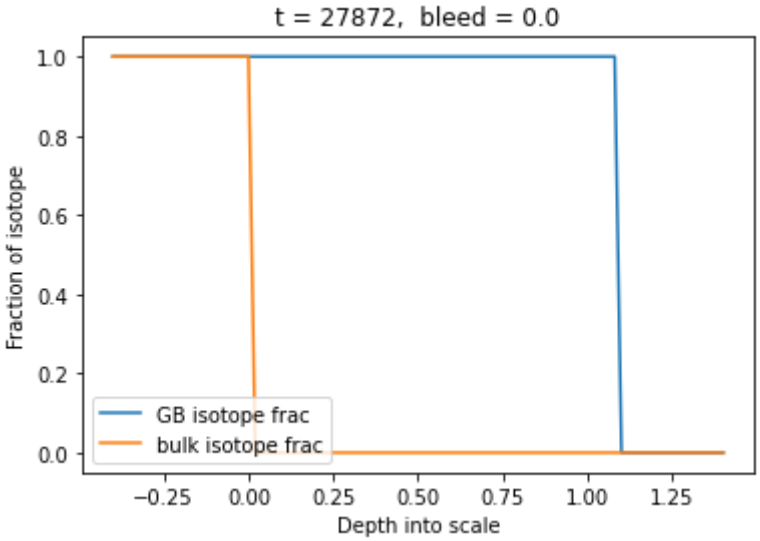

```
* Seems like time to add a scale node from gas *


t_count_1 = 7096
delta_length_1 = 0.019998039962043843 t = 20517 t_count_1 = 7096


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 18 56 17
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  17 57 17

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  7225  Actual number =  7096

 Number of nodes added to scale : 6
```



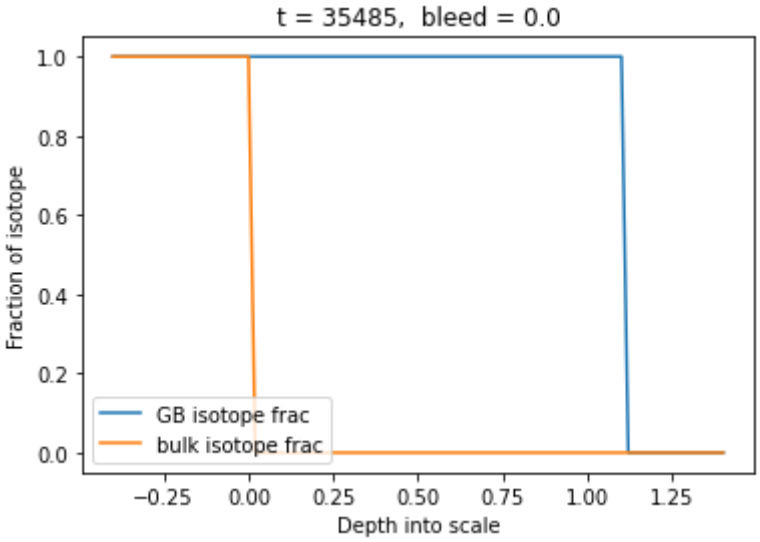

```
* Seems like time to add a scale node from metal *


t_count_2 = 7354
delta_length_2 = 0.01999804787689511 t = 27872 t_count_2 = 7354


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 17 57 17
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  17 58 16

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  7483  Actual number =  7354

 Number of nodes added to scale : 7
```



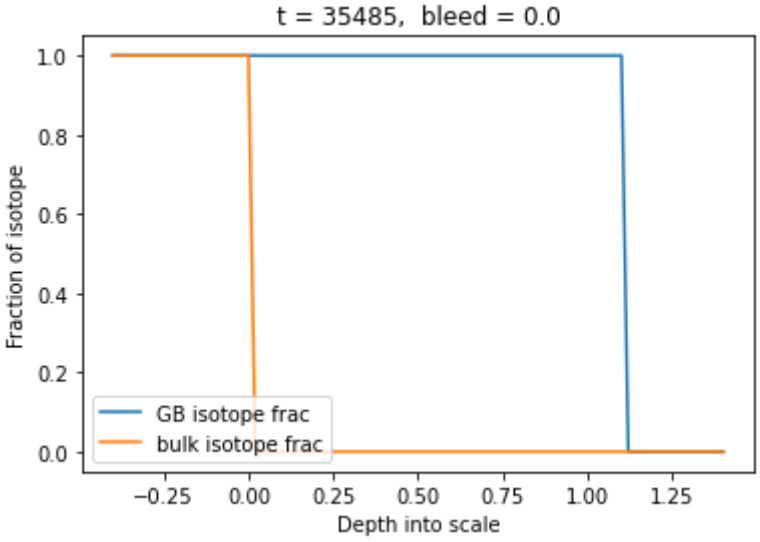
```
* Seems like time to add a scale node from gas *


t_count_1 = 7354
delta_length_1 = 0.019998000158312504 t = 27872 t_count_1 = 7354


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 17 58 16
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  16 59 16

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  7483  Actual number =  7354

 Number of nodes added to scale : 8
```



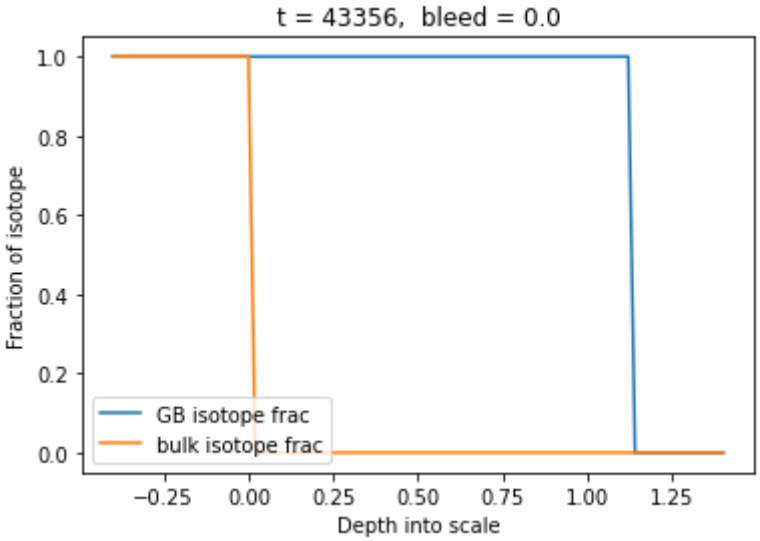

```
* Seems like time to add a scale node from metal *


t_count_2 = 7612
delta_length_2 = 0.019998009039803543 t = 35485 t_count_2 = 7612


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 16 59 16
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  16 60 15

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  7741  Actual number =  7612

 Number of nodes added to scale : 9
```



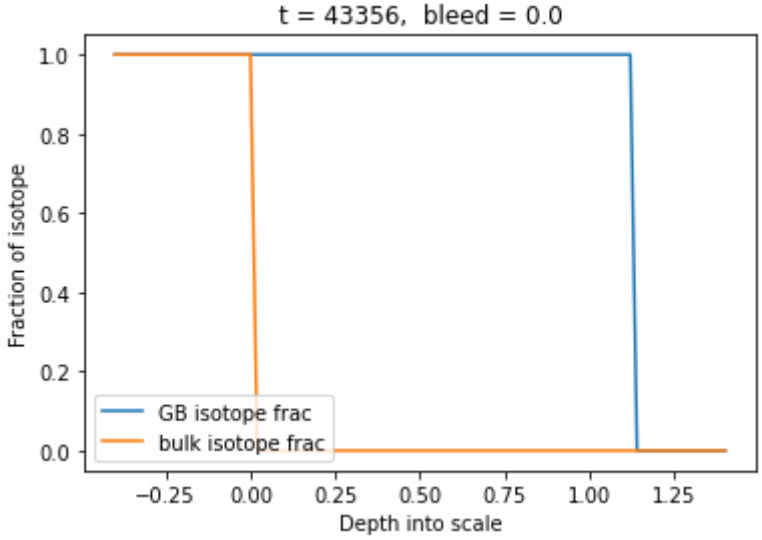

```
* Seems like time to add a scale node from gas *


t_count_1 = 7612
delta_length_1 = 0.01999796450230822 t = 35485 t_count_1 = 7612


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 16 60 15
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  15 61 15

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  7741  Actual number =  7612

 Number of nodes added to scale : 10
```



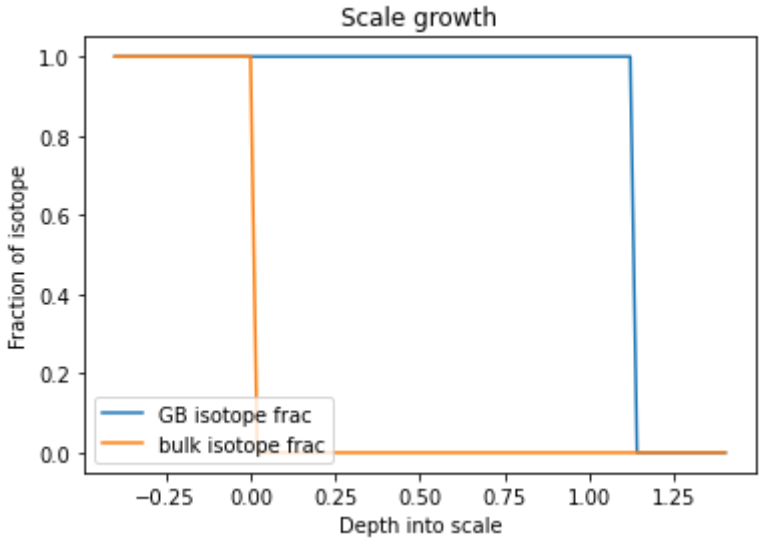
```
* Seems like time to add a scale node from metal *


t_count_2 = 7870
delta_length_2 = 0.01999797410715881 t = 43356 t_count_2 = 7870


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 15 61 15
One node to be added to scale grid and taken from metal grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  15 62 14

*** LEAVING update_ranges, scale node added in metal direction ***


Expected number of timesteps before incrementing scale grid
 based on initial increment only =  7999  Actual number =  7870

 Number of nodes added to scale : 11
```

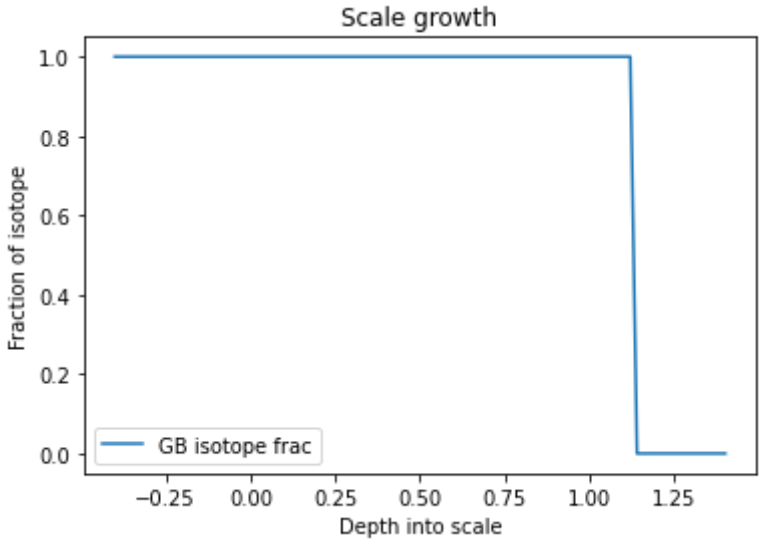



```
* Seems like time to add a scale node from gas *


t_count_1 = 7870
delta_length_1 = 0.019997932442909722 t = 43356 t_count_1 = 7870


*** ENTERING update_ranges ***

Previous values of ngridy_gas, ngridy_scale, ngridy_metal 15 62 14
One node to be added to scale grid and taken from gas grid
New values of ngridy_gas, ngridy_scale, ngridy_metal =  14 63 14

*** LEAVING update_ranges, scale node added in gas direction ***


Expected number of timesteps before adding node to scale
 based on initial increment only =  7999  Actual number =  7870

 Number of nodes added to scale : 12
```




```
*** LEAVING tracer_diffusion ***
```






```
c_GB =
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [ ]: