MercurySmelting_ThermoModel

October 13, 2022

1 Modeling explosive volcanism on Mercury

Here we employ the thermodanymic magmatic gas model of Iacovino (2015), in python beta as "TAVERNmag". Experimental results detailed in the main text as well as solubility limits are used as constraints to determine which magmatic volatile compositions could feasibly have produced the observed pyroclastic deposits on Mercury.

First we import the necessary libraries, including the model of Iacovino (2015) in python beta "TAVERNmag". We also define a status_bar, which we can use to display the progress of long computations.

[ ]:
```
import sys

# setting path
sys.path.append('..')

%pip install plotly
%pip install scikit-image
%pip install kaleido

import tavernmag as tv
import plotly.figure_factory as ff
import plotly.express as px
import numpy as np
import sys
import pickle
import copy

# Define a status bar for long calls
def status_bar(percent, btext=None, barLen=20):
    """
    Prints an updating status bar to the terminal or jupyter notebook.

    Parameters
    ----------
    percent: float
        Percent value of progress from 0 to 1
```

```
    btext: string
        Any extra text to display next to status bar

    barLen: int
        Length of bar to print
    """
    sys.stdout.write("\r")
    sys.stdout.write("[{:<{}}] {:.0f}%".format("=" * int(barLen * percent),
                                               barLen, percent * 100))
    if btext is not None:
        sys.stdout.write(" " + str(btext))
    if percent == 1.0:
        sys.stdout.write("\n")
    sys.stdout.flush()
```

```
Requirement already satisfied: plotly in /opt/anaconda3/lib/python3.9/site-
packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in
/opt/anaconda3/lib/python3.9/site-packages (from plotly) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
ERROR: Could not find a version that satisfies the requirement pickle (from
versions: none)
ERROR: No matching distribution found for pickle
Note: you may need to restart the kernel to use updated packages.
```

1.1 Define necessary constants

[ ]:
```
# Fluid species considered
fluid_species_names = ['CO', 'CO2', 'H2', 'H2O', 'H2S', 'O2', 'S2', 'SO2']

# Molecular Weights
molecularWeight = tv.core.oxideMass

# Specific Heat Rations (C_P/C_V) at 1600 K and 1 atm
specificHeatRatios = {'CO': 1.306,
                      'CO2': 1.166,
                      'H2': 1.39,
                      'H2O': 1.21,
                      'H2S': 1.19,
                      'O2': 1,
                      'S2': 1.29,
                      'SO2': 1.18}

# Other constants
g = 3.7 # m/s2, gravitational acceleration due to gravity on Mercury
R = 8.3145 # J/mol-K, gas constant
T = 1600 # degrees K, temperature
```

```
# Define deposit radius in meters
deposit_radius = 110800
```

1.2 Generate bulk gas compositions

Here we generate all potential bulk gas compositions for a C-O-H-S fluid in terms of H_{tot}, C_{tot}, and S_{tot} in steps of 5 mol% such that they evenly cover the entire H-C-S binary. These are then translated into H_2O_{tot}, CO_{2tot}, and S_{tot} for input into TAVERNmag. These will be speciated properly according to the system fO_2, pressure, and temperature in the next step.

1.2.1 Generic gas compositions

Here we create a generic list of all H_{tot}, C_{tot}, and S_{tot} values ranging from 1 to 100 in steps of 5. We are creating compositions in this space since that is how we will plot them up later.

[ ]:
```
# define a function to generate an evenly spaced list
def sums(length, total_sum, step):
    """Generate a list of tuples where each tuple has length
    number of values and a sum of total_sum. Step is the step
    size."""
    if length == 1:
        yield (total_sum,)
    else:
        for value in range(total_sum + 1):
            for permutation in sums(length - 1, total_sum - value, step):
                remainder = value % step
                if remainder == 0:
                    yield (value,) + permutation

# create a dict with keys Htot, Ctot, and Stot, each ranging from 1-100 in␣
 ↪steps of 5
bulk_comp_list_simp = []
for l in list(sums(3,100,5)):
    bulk_comp_list_simp.append({'Htot': l[0], 'Ctot': l[1], 'Stot': l[2]})

# create new dict to be converted to moles and wt%
bulk_comp_list_moles = []
bulk_comp_list_wtper = []
for l in list(sums(3,100,5)):
    bulk_comp_list_moles.append({'Htot': l[0], 'Ctot': l[1], 'Stot': l[2]})
    bulk_comp_list_wtper.append({'Htot': l[0], 'Ctot': l[1], 'Stot': l[2]})
```

1.2.2 Convert gas compositions to H_2O_{tot}, $CO_{2,tot}$, S_{tot} in moles

We need to convert our generic compositions from H_{tot}, C_{tot}, and S_{tot} to H_2O_{tot}, $CO_{2,tot}$, and S_{tot} (e.g., where all H species are given as H_2O) since these are the inputs required by TAVERNmag. This is done separately so that the H-C-S values for each set of gas compositions cover the entire

ternary space evenly. The below compositions will be speciated properly in the next step, when we apply the TAVERNmag model.

Below we compute the gas compositions in terms of mol%.

[ ]:
```
# convert to H2Otot, CO2tot, Stot
# in terms of moles
for comp in bulk_comp_list_moles:
    comp['Htot'] /= 0.6666
    comp['Ctot'] *=3

    # normalize
    comp_sum = comp['Htot']+comp['Ctot']+comp['Stot']
    comp['Htot'] *= 100/comp_sum
    comp['Ctot'] *= 100/comp_sum
    comp['Stot'] *= 100/comp_sum

    # rename
    comp['H2O'] = comp.pop('Htot')
    comp['CO2'] = comp.pop('Ctot')
    comp['S'] = comp.pop('Stot')
```

1.2.3 Convert gas compositions to H2Otot, CO2tot, Stot in wt%

Here we compute the gas compositions as above, but in terms of wt%.

[ ]:
```
# in terms of wtpercent
for comp in bulk_comp_list_wtper:
    comp['Htot'] *= (molecularWeight['H2O']/molecularWeight['H2'])
    comp['Ctot'] *= (molecularWeight['CO2']/molecularWeight['C'])

    # normalize
    comp_sum = comp['Htot']+comp['Ctot']+comp['Stot']
    comp['Htot'] *= 100/comp_sum
    comp['Ctot'] *= 100/comp_sum
    comp['Stot'] *= 100/comp_sum

    # rename
    comp['H2O'] = comp.pop('Htot')
    comp['CO2'] = comp.pop('Ctot')
    comp['S'] = comp.pop('Stot')
```

Optionally, we can print the three dicts that we made to check their values.

[ ]:
```
for i in range(len(bulk_comp_list_simp)):
    print(bulk_comp_list_simp[i])
    print(bulk_comp_list_wtper[i])
    #print(bulk_comp_list_moles[i])
```

```
    print('\n')
```

```
{'Htot': 0, 'Ctot': 0, 'Stot': 100}
{'H2O': 0.0, 'CO2': 0.0, 'S': 100.0}


{'Htot': 0, 'Ctot': 5, 'Stot': 95}
{'H2O': 0.0, 'CO2': 16.167468672544654, 'S': 83.83253132745534}


{'Htot': 0, 'Ctot': 10, 'Stot': 90}
{'H2O': 0.0, 'CO2': 28.933712804795068, 'S': 71.06628719520494}


{'Htot': 0, 'Ctot': 15, 'Stot': 85}
{'H2O': 0.0, 'CO2': 39.26987712213637, 'S': 60.730122877863636}


{'Htot': 0, 'Ctot': 20, 'Stot': 80}
{'H2O': 0.0, 'CO2': 47.809518015747486, 'S': 52.19048198425252}


{'Htot': 0, 'Ctot': 25, 'Stot': 75}
{'H2O': 0.0, 'CO2': 54.98356489897193, 'S': 45.016435101028094}


{'Htot': 0, 'Ctot': 30, 'Stot': 70}
{'H2O': 0.0, 'CO2': 61.09532916652977, 'S': 38.90467083347022}


{'Htot': 0, 'Ctot': 35, 'Stot': 65}
{'H2O': 0.0, 'CO2': 66.36448962331845, 'S': 33.63551037668154}


{'Htot': 0, 'Ctot': 40, 'Stot': 60}
{'H2O': 0.0, 'CO2': 70.95405881883498, 'S': 29.04594118116501}


{'Htot': 0, 'Ctot': 45, 'Stot': 55}
{'H2O': 0.0, 'CO2': 74.98754751208664, 'S': 25.012452487913375}


{'Htot': 0, 'Ctot': 50, 'Stot': 50}
{'H2O': 0.0, 'CO2': 78.56024648031888, 'S': 21.43975351968112}


{'Htot': 0, 'Ctot': 55, 'Stot': 45}
```

{'H2O': 0.0, 'CO2': 81.74685071739358, 'S': 18.253149282606415}

{'Htot': 0, 'Ctot': 60, 'Stot': 40}
{'H2O': 0.0, 'CO2': 84.60673854896528, 'S': 15.393261451034723}

{'Htot': 0, 'Ctot': 65, 'Stot': 35}
{'H2O': 0.0, 'CO2': 87.18770615702503, 'S': 12.81229384297496}

{'Htot': 0, 'Ctot': 70, 'Stot': 30}
{'H2O': 0.0, 'CO2': 89.52866024357304, 'S': 10.471339756426943}

{'Htot': 0, 'Ctot': 75, 'Stot': 25}
{'H2O': 0.0, 'CO2': 91.66159286923765, 'S': 8.338407130762347}

{'Htot': 0, 'Ctot': 80, 'Stot': 20}
{'H2O': 0.0, 'CO2': 93.61305222474577, 'S': 6.386947775254227}

{'Htot': 0, 'Ctot': 85, 'Stot': 15}
{'H2O': 0.0, 'CO2': 95.40525331416481, 'S': 4.5947466858351955}

{'Htot': 0, 'Ctot': 90, 'Stot': 10}
{'H2O': 0.0, 'CO2': 97.05692737111208, 'S': 2.9430726288879194}

{'Htot': 0, 'Ctot': 95, 'Stot': 5}
{'H2O': 0.0, 'CO2': 98.58397900402582, 'S': 1.416020995974184}

{'Htot': 0, 'Ctot': 100, 'Stot': 0}
{'H2O': 0.0, 'CO2': 100.00000000000001, 'S': 0.0}

{'Htot': 5, 'Ctot': 0, 'Stot': 95}
{'H2O': 31.98742875406169, 'CO2': 0.0, 'S': 68.01257124593832}

{'Htot': 5, 'Ctot': 5, 'Stot': 90}
{'H2O': 29.20241982585787, 'CO2': 11.974521021173484, 'S': 58.82305915296865}

{'Htot': 5, 'Ctot': 10, 'Stot': 85}

{'H2O': 26.863527146451347, 'CO2': 22.030905139800097, 'S': 51.10556771374857}

{'Htot': 5, 'Ctot': 15, 'Stot': 80}
{'H2O': 24.871507728255388, 'CO2': 30.595860948813392, 'S': 44.53263132293121}

{'Htot': 5, 'Ctot': 20, 'Stot': 75}
{'H2O': 23.154523222573022, 'CO2': 37.978267104898066, 'S': 38.8672096725289}

{'Htot': 5, 'Ctot': 25, 'Stot': 70}
{'H2O': 21.659291478710706, 'CO2': 44.40721739194535, 'S': 33.933491129343935}

{'Htot': 5, 'Ctot': 30, 'Stot': 65}
{'H2O': 20.345458533408284, 'CO2': 50.05621911057927, 'S': 29.59832235601245}

{'Htot': 5, 'Ctot': 35, 'Stot': 60}
{'H2O': 19.18190179575637, 'CO2': 55.0590880325568, 'S': 25.759010171686825}

{'Htot': 5, 'Ctot': 40, 'Stot': 55}
{'H2O': 18.144233134771248, 'CO2': 59.52068421206821, 'S': 22.335082653160544}

{'Htot': 5, 'Ctot': 45, 'Stot': 50}
{'H2O': 17.213070816469745, 'CO2': 63.52434202504954, 'S': 19.262587158480713}

{'Htot': 5, 'Ctot': 50, 'Stot': 45}
{'H2O': 16.372817571603466, 'CO2': 67.13712403155097, 'S': 16.490058396845587}

{'Htot': 5, 'Ctot': 55, 'Stot': 40}
{'H2O': 15.610779896690898, 'CO2': 70.41360765957715, 'S': 13.975612443731935}

{'Htot': 5, 'Ctot': 60, 'Stot': 35}
{'H2O': 14.916522358771818, 'CO2': 73.3986614587698, 'S': 11.68481618245839}

{'Htot': 5, 'Ctot': 65, 'Stot': 30}
{'H2O': 14.281386859779719, 'CO2': 76.12951208799184, 'S': 9.589101052228447}

{'Htot': 5, 'Ctot': 70, 'Stot': 25}

{'H2O': 13.698129701446785, 'CO2': 78.63730480365543, 'S': 7.664565494897785}

{'Htot': 5, 'Ctot': 75, 'Stot': 20}
{'H2O': 13.16064409503694, 'CO2': 80.9482965665019, 'S': 5.891059338461164}

{'Htot': 5, 'Ctot': 80, 'Stot': 15}
{'H2O': 12.663745533282382, 'CO2': 83.08477886387044, 'S': 4.251475602847174}

{'Htot': 5, 'Ctot': 85, 'Stot': 10}
{'H2O': 12.20300401483766, 'CO2': 85.06579908325696, 'S': 2.7311969019053812}

{'Htot': 5, 'Ctot': 90, 'Stot': 5}
{'H2O': 11.774611607787854, 'CO2': 86.90772994084938, 'S': 1.3176584513627707}

{'Htot': 5, 'Ctot': 95, 'Stot': 0}
{'H2O': 11.375276962570291, 'CO2': 88.6247230374297, 'S': 0.0}

{'Htot': 10, 'Ctot': 0, 'Stot': 90}
{'H2O': 49.82162117315192, 'CO2': 0.0, 'S': 50.17837882684809}

{'Htot': 10, 'Ctot': 5, 'Stot': 85}
{'H2O': 46.377166378712985, 'CO2': 9.508533145815163, 'S': 44.114300475471865}

{'Htot': 10, 'Ctot': 10, 'Stot': 80}
{'H2O': 43.3781834391241, 'CO2': 17.787326274659222, 'S': 38.83449028621668}

{'Htot': 10, 'Ctot': 15, 'Stot': 75}
{'H2O': 40.743501839194835, 'CO2': 25.060453319915588, 'S': 34.196044840889584}

{'Htot': 10, 'Ctot': 20, 'Stot': 70}
{'H2O': 38.410542252320575, 'CO2': 31.5006666144711, 'S': 30.08879113320832}

{'Htot': 10, 'Ctot': 25, 'Stot': 65}
{'H2O': 36.330282179464184, 'CO2': 37.243294413391844, 'S': 26.42642340714397}

{'Htot': 10, 'Ctot': 30, 'Stot': 60}

{'H2O': 34.463773463568664, 'CO2': 42.395854412349934, 'S': 23.140372124081406}

{'Htot': 10, 'Ctot': 35, 'Stot': 55}
{'H2O': 32.7796808462432, 'CO2': 47.04484864456198, 'S': 20.175470509194817}

{'Htot': 10, 'Ctot': 40, 'Stot': 50}
{'H2O': 31.252508581415626, 'CO2': 51.260658973362645, 'S': 17.48683244522173}

{'Htot': 10, 'Ctot': 45, 'Stot': 45}
{'H2O': 29.861300703581385, 'CO2': 55.101135045356145, 'S': 15.037564251062465}

{'Htot': 10, 'Ctot': 50, 'Stot': 40}
{'H2O': 28.588673647374126, 'CO2': 58.61426477658229, 'S': 12.797061576043571}

{'Htot': 10, 'Ctot': 55, 'Stot': 35}
{'H2O': 27.42008612236168, 'CO2': 61.84018988766418, 'S': 10.739723989974133}

{'Htot': 10, 'Ctot': 60, 'Stot': 30}
{'H2O': 26.34328102286909, 'CO2': 64.81274653048577, 'S': 8.843972446645143}

{'Htot': 10, 'Ctot': 65, 'Stot': 25}
{'H2O': 25.34785386830854, 'CO2': 67.56065662315552, 'S': 7.091489508535945}

{'Htot': 10, 'Ctot': 70, 'Stot': 20}
{'H2O': 24.42491552438276, 'CO2': 70.1084589194523, 'S': 5.466625556164935}

{'Htot': 10, 'Ctot': 75, 'Stot': 15}
{'H2O': 23.56682601941144, 'CO2': 72.47724381779645, 'S': 3.955930162792128}

{'Htot': 10, 'Ctot': 80, 'Stot': 10}
{'H2O': 22.766982565310016, 'CO2': 74.68523853645713, 'S': 2.5477788982328606}

{'Htot': 10, 'Ctot': 85, 'Stot': 5}
{'H2O': 22.019649328229782, 'CO2': 76.74827703741798, 'S': 1.232073634352241}

{'Htot': 10, 'Ctot': 90, 'Stot': 0}

{'H2O': 21.31981966062939, 'CO2': 78.68018033937062, 'S': 0.0}

{'Htot': 15, 'Ctot': 0, 'Stot': 85}
{'H2O': 61.194334046672786, 'CO2': 0.0, 'S': 38.80566595332722}

{'Htot': 15, 'Ctot': 5, 'Stot': 80}
{'H2O': 57.68609240333661, 'CO2': 7.884772680167414, 'S': 34.42913491649598}

{'Htot': 15, 'Ctot': 10, 'Stot': 75}
{'H2O': 54.55829185671982, 'CO2': 14.914504040269604, 'S': 30.527204103010586}

{'Htot': 15, 'Ctot': 15, 'Stot': 70}
{'H2O': 51.752231484423625, 'CO2': 21.221124396483535, 'S': 27.026644119092833}

{'Htot': 15, 'Ctot': 20, 'Stot': 65}
{'H2O': 49.22069612660109, 'CO2': 26.910749815003847, 'S': 23.868554058395063}

{'Htot': 15, 'Ctot': 25, 'Stot': 60}
{'H2O': 46.925278180693965, 'CO2': 32.069701378056166, 'S': 21.005020441249854}

{'Htot': 15, 'Ctot': 30, 'Stot': 55}
{'H2O': 44.83441539742884, 'CO2': 36.76891523716489, 'S': 18.39666936540627}

{'Htot': 15, 'Ctot': 35, 'Stot': 50}
{'H2O': 42.92193088272865, 'CO2': 41.0672239586848, 'S': 16.01084515858654}

{'Htot': 15, 'Ctot': 40, 'Stot': 45}
{'H2O': 41.16593148365511, 'CO2': 45.01383240126602, 'S': 13.820236115078881}

{'Htot': 15, 'Ctot': 45, 'Stot': 40}
{'H2O': 39.54796595813377, 'CO2': 48.65020972929742, 'S': 11.801824312568812}

{'Htot': 15, 'Ctot': 50, 'Stot': 35}
{'H2O': 38.05237415846607, 'CO2': 52.011552122887004, 'S': 9.936073718646927}

{'Htot': 15, 'Ctot': 55, 'Stot': 30}

{'H2O': 36.66577850598584, 'CO2': 55.12792568851125, 'S': 8.206295805502908}

{'Htot': 15, 'Ctot': 60, 'Stot': 25}
{'H2O': 35.37668273814729, 'CO2': 58.02516827488768, 'S': 6.598148986965022}

{'Htot': 15, 'Ctot': 65, 'Stot': 20}
{'H2O': 34.175152424309225, 'CO2': 60.72560751371226, 'S': 5.099240061978528}

{'Htot': 15, 'Ctot': 70, 'Stot': 15}
{'H2O': 33.05255844850494, 'CO2': 63.248637342103656, 'S': 3.6988042093913935}

{'Htot': 15, 'Ctot': 75, 'Stot': 10}
{'H2O': 32.00136944117809, 'CO2': 65.61118451225276, 'S': 2.3874460465691563}

{'Htot': 15, 'Ctot': 80, 'Stot': 5}
{'H2O': 31.01498259851398, 'CO2': 67.82808882495527, 'S': 1.1569285765307462}

{'Htot': 15, 'Ctot': 85, 'Stot': 0}
{'H2O': 30.087584854411723, 'CO2': 69.91241514558828, 'S': 0.0}

{'Htot': 20, 'Ctot': 0, 'Stot': 80}
{'H2O': 69.07856896353388, 'CO2': 0.0, 'S': 30.921431036466124}

{'Htot': 20, 'Ctot': 5, 'Stot': 75}
{'H2O': 65.69597015837265, 'CO2': 6.734696819296114, 'S': 27.569333022331232}

{'Htot': 20, 'Ctot': 10, 'Stot': 70}
{'H2O': 62.62918118705795, 'CO2': 12.840621618610628, 'S': 24.53019719433144}

{'Htot': 20, 'Ctot': 15, 'Stot': 65}
{'H2O': 59.83594710835128, 'CO2': 18.40190326889448, 'S': 21.762149622754237}

{'Htot': 20, 'Ctot': 20, 'Stot': 60}
{'H2O': 57.28122934203553, 'CO2': 23.488302988764865, 'S': 19.230467669199605}

{'Htot': 20, 'Ctot': 25, 'Stot': 55}

{'H2O': 54.93572822307047, 'CO2': 28.158155914109503, 'S': 16.90611586282002}

{'Htot': 20, 'Ctot': 30, 'Stot': 50}
{'H2O': 52.77475426018232, 'CO2': 32.46061840308058, 'S': 14.764627336737101}

{'Htot': 20, 'Ctot': 35, 'Stot': 45}
{'H2O': 50.777355715470556, 'CO2': 36.43740501029272, 'S': 12.785239274236716}

{'Htot': 20, 'Ctot': 40, 'Stot': 40}
{'H2O': 48.92563707348704, 'CO2': 40.12414540332119, 'S': 10.95021752319177}

{'Htot': 20, 'Ctot': 45, 'Stot': 35}
{'H2O': 47.20422138556198, 'CO2': 43.55145482602488, 'S': 9.244323788413137}

{'Htot': 20, 'Ctot': 50, 'Stot': 30}
{'H2O': 45.59982225868318, 'CO2': 46.74578626142462, 'S': 7.654391479892198}

{'Htot': 20, 'Ctot': 55, 'Stot': 25}
{'H2O': 44.10090025987333, 'CO2': 49.73011452475722, 'S': 6.1689852153694575}

{'Htot': 20, 'Ctot': 60, 'Stot': 20}
{'H2O': 42.697384930346736, 'CO2': 52.52448972857199, 'S': 4.7781253410812665}

{'Htot': 20, 'Ctot': 65, 'Stot': 15}
{'H2O': 41.38044824400023, 'CO2': 55.146488323031235, 'S': 3.4730634329685452}

{'Htot': 20, 'Ctot': 70, 'Stot': 10}
{'H2O': 40.14231873550591, 'CO2': 57.61158316372306, 'S': 2.2460981007710217}

{'Htot': 20, 'Ctot': 75, 'Stot': 5}
{'H2O': 38.97612802744304, 'CO2': 59.93344907353774, 'S': 1.0904228990192224}

{'Htot': 20, 'Ctot': 80, 'Stot': 0}
{'H2O': 37.8757833538119, 'CO2': 62.1242166461881, 'S': 0.0}

{'Htot': 25, 'Ctot': 0, 'Stot': 75}

{'H2O': 74.86597681087147, 'CO2': 0.0, 'S': 25.13402318912854}

{'Htot': 25, 'Ctot': 5, 'Stot': 70}
{'H2O': 71.66664005889874, 'CO2': 5.877414906159725, 'S': 22.455945034941525}

{'Htot': 25, 'Ctot': 10, 'Stot': 65}
{'H2O': 68.7295389961296, 'CO2': 11.273083729259612, 'S': 19.997377274610766}

{'Htot': 25, 'Ctot': 15, 'Stot': 60}
{'H2O': 66.02370147385373, 'CO2': 16.243904564942042, 'S': 17.732393961204213}

{'Htot': 25, 'Ctot': 20, 'Stot': 55}
{'H2O': 63.522848010003585, 'CO2': 20.83815473804149, 'S': 15.63899725195492}

{'Htot': 25, 'Ctot': 25, 'Stot': 50}
{'H2O': 61.20453547774612, 'CO2': 25.09706390908306, 'S': 13.698400613170822}

{'Htot': 25, 'Ctot': 30, 'Stot': 45}
{'H2O': 59.0494817019552, 'CO2': 29.056054839812496, 'S': 11.894463458232309}

{'Htot': 25, 'Ctot': 35, 'Stot': 40}
{'H2O': 57.041027890945195, 'CO2': 32.745730945300565, 'S': 10.21324116375425}

{'Htot': 25, 'Ctot': 40, 'Stot': 35}
{'H2O': 55.164707165571656, 'CO2': 36.19266893744752, 'S': 8.642623896980819}

{'Htot': 25, 'Ctot': 45, 'Stot': 30}
{'H2O': 53.40789553377235, 'CO2': 39.420060010202, 'S': 7.172044456025649}

{'Htot': 25, 'Ctot': 50, 'Stot': 25}
{'H2O': 51.75952749851909, 'CO2': 42.44823228849093, 'S': 5.79224021298998}

{'Htot': 25, 'Ctot': 55, 'Stot': 20}
{'H2O': 50.209862753446004, 'CO2': 45.29507942536289, 'S': 4.495057821191103}

{'Htot': 25, 'Ctot': 60, 'Stot': 15}

{'H2O': 48.75029357054722, 'CO2': 47.97641444483252, 'S': 3.2732919846202564}

{'Htot': 25, 'Ctot': 65, 'Stot': 10}
{'H2O': 47.37318483359615, 'CO2': 50.50626361215395, 'S': 2.120551554249899}

{'Htot': 25, 'Ctot': 70, 'Stot': 5}
{'H2O': 46.07174043934345, 'CO2': 52.89711186456199, 'S': 1.0311476960945478}

{'Htot': 25, 'Ctot': 75, 'Stot': 0}
{'H2O': 44.83989113140989, 'CO2': 55.16010886859012, 'S': 0.0}

{'Htot': 30, 'Ctot': 0, 'Stot': 70}
{'H2O': 79.29486333025221, 'CO2': 0.0, 'S': 20.705136669747787}

{'Htot': 30, 'Ctot': 5, 'Stot': 65}
{'H2O': 76.28890067235382, 'CO2': 5.213740563458485, 'S': 18.4973587641877}

{'Htot': 30, 'Ctot': 10, 'Stot': 60}
{'H2O': 73.5025181071748, 'CO2': 10.046626882659863, 'S': 16.45085501016535}

{'Htot': 30, 'Ctot': 15, 'Stot': 55}
{'H2O': 70.91250354680486, 'CO2': 14.538919539013047, 'S': 14.54857691418211}

{'Htot': 30, 'Ctot': 20, 'Stot': 50}
{'H2O': 68.49880524738096, 'CO2': 18.72539760406507, 'S': 12.775797148553982}

{'Htot': 30, 'Ctot': 25, 'Stot': 45}
{'H2O': 66.24401165961415, 'CO2': 22.636260821170037, 'S': 11.119727519215829}

{'Htot': 30, 'Ctot': 30, 'Stot': 40}
{'H2O': 64.1329307368463, 'CO2': 26.29785928143607, 'S': 9.569209981717616}

{'Htot': 30, 'Ctot': 35, 'Stot': 35}
{'H2O': 62.15224719908979, 'CO2': 29.73328788765685, 'S': 8.114464913253356}

{'Htot': 30, 'Ctot': 40, 'Stot': 30}

{'H2O': 60.29024140478397, 'CO2': 32.96287396174145, 'S': 6.746884633474576}

{'Htot': 30, 'Ctot': 45, 'Stot': 25}
{'H2O': 58.53655728622702, 'CO2': 36.00457975285922, 'S': 5.45886296091376}

{'Htot': 30, 'Ctot': 50, 'Stot': 20}
{'H2O': 56.88200964111973, 'CO2': 38.87433668373927, 'S': 4.243653675140989}

{'Htot': 30, 'Ctot': 55, 'Stot': 15}
{'H2O': 55.31842320740916, 'CO2': 41.5863244699115, 'S': 3.0952523226793462}

{'Htot': 30, 'Ctot': 60, 'Stot': 10}
{'H2O': 53.838497569374375, 'CO2': 44.153205435531554, 'S': 2.008296995094065}

{'Htot': 30, 'Ctot': 65, 'Stot': 5}
{'H2O': 52.43569318358138, 'CO2': 46.58632219750753, 'S': 0.977984618911093}

{'Htot': 30, 'Ctot': 70, 'Stot': 0}
{'H2O': 51.10413477046285, 'CO2': 48.89586522953716, 'S': 0.0}

{'Htot': 35, 'Ctot': 0, 'Stot': 65}
{'H2O': 82.79332689921412, 'CO2': 0.0, 'S': 17.20667310078588}

{'Htot': 35, 'Ctot': 5, 'Stot': 60}
{'H2O': 79.97319035842465, 'CO2': 4.684742149001649, 'S': 15.342067492573719}

{'Htot': 35, 'Ctot': 10, 'Stot': 55}
{'H2O': 77.33884632711047, 'CO2': 9.060850305457611, 'S': 13.600303367431914}

{'Htot': 35, 'Ctot': 15, 'Stot': 50}
{'H2O': 74.87252010498499, 'CO2': 13.157851368656065, 'S': 11.969628526358964}

{'Htot': 35, 'Ctot': 20, 'Stot': 45}
{'H2O': 72.55863425331769, 'CO2': 17.00162220049305, 'S': 10.439743546189257}

{'Htot': 35, 'Ctot': 25, 'Stot': 40}

{'H2O': 70.38347924857061, 'CO2': 20.614936727632188, 'S': 9.001584023797207}

{'Htot': 35, 'Ctot': 30, 'Stot': 35}
{'H2O': 68.33494165053197, 'CO2': 24.017917501981344, 'S': 7.647140847486675}

{'Htot': 35, 'Ctot': 35, 'Stot': 30}
{'H2O': 66.40227839848775, 'CO2': 27.228410634396703, 'S': 6.3693109671155606}

{'Htot': 35, 'Ctot': 40, 'Stot': 25}
{'H2O': 64.57592835438565, 'CO2': 30.262298854589343, 'S': 5.1617727910249975}

{'Htot': 35, 'Ctot': 45, 'Stot': 20}
{'H2O': 62.84735411377633, 'CO2': 33.13376429093805, 'S': 4.018881595285614}

{'Htot': 35, 'Ctot': 50, 'Stot': 15}
{'H2O': 61.20890856091388, 'CO2': 35.855510145906564, 'S': 2.93558129317955}

{'Htot': 35, 'Ctot': 55, 'Stot': 10}
{'H2O': 59.65372176714483, 'CO2': 38.43894857768584, 'S': 1.9073296551693266}

{'Htot': 35, 'Ctot': 60, 'Stot': 5}
{'H2O': 58.17560470408235, 'CO2': 40.89436064952433, 'S': 0.9300346463933231}

{'Htot': 35, 'Ctot': 65, 'Stot': 0}
{'H2O': 56.76896692559317, 'CO2': 43.231033074406845, 'S': 0.0}

{'Htot': 40, 'Ctot': 0, 'Stot': 60}
{'H2O': 85.62669328390133, 'CO2': 0.0, 'S': 14.373306716098673}

{'Htot': 40, 'Ctot': 5, 'Stot': 55}
{'H2O': 82.97871563962653, 'CO2': 4.253202220321032, 'S': 12.768082140052442}

{'Htot': 40, 'Ctot': 10, 'Stot': 50}
{'H2O': 80.48960090655497, 'CO2': 8.251237601104286, 'S': 11.259161492340745}

{'Htot': 40, 'Ctot': 15, 'Stot': 45}

{'H2O': 78.14546919522098, 'CO2': 12.016400128411481, 'S': 9.838130676367538}

{'Htot': 40, 'Ctot': 20, 'Stot': 40}
{'H2O': 75.93401177747292, 'CO2': 15.568460176821542, 'S': 8.497528045705545}

{'Htot': 40, 'Ctot': 25, 'Stot': 35}
{'H2O': 73.84427489127448, 'CO2': 18.925011763834696, 'S': 7.230713344890824}

{'Htot': 40, 'Ctot': 30, 'Stot': 30}
{'H2O': 71.86647828808815, 'CO2': 22.101764000471974, 'S': 6.031757711439871}

{'Htot': 40, 'Ctot': 35, 'Stot': 25}
{'H2O': 69.99186217908975, 'CO2': 25.112786927451847, 'S': 4.895350893458401}

{'Htot': 40, 'Ctot': 40, 'Stot': 20}
{'H2O': 68.21255752660956, 'CO2': 27.970719854083065, 'S': 3.816722619307379}

{'Htot': 40, 'Ctot': 45, 'Stot': 15}
{'H2O': 66.52147562948089, 'CO2': 30.686948707133737, 'S': 2.791575663385376}

{'Htot': 40, 'Ctot': 50, 'Stot': 10}
{'H2O': 64.91221373504081, 'CO2': 33.27175763756754, 'S': 1.8160286273916495}

{'Htot': 40, 'Ctot': 55, 'Stot': 5}
{'H2O': 63.37897402791341, 'CO2': 35.734459141388065, 'S': 0.8865668306985388}

{'Htot': 40, 'Ctot': 60, 'Stot': 0}
{'H2O': 61.91649383487549, 'CO2': 38.08350616512452, 'S': 0.0}

{'Htot': 45, 'Ctot': 0, 'Stot': 55}
{'H2O': 87.96816250793496, 'CO2': 0.0, 'S': 12.031837492065042}

{'Htot': 45, 'Ctot': 5, 'Stot': 50}
{'H2O': 85.47723049966524, 'CO2': 3.894459962768834, 'S': 10.628309537565924}

{'Htot': 45, 'Ctot': 10, 'Stot': 45}

{'H2O': 83.1234818736305, 'CO2': 7.574439888387882, 'S': 9.30207823798163}

{'Htot': 45, 'Ctot': 15, 'Stot': 40}
{'H2O': 80.89588765574896, 'CO2': 11.057183080909263, 'S': 8.046929263341779}

{'Htot': 45, 'Ctot': 20, 'Stot': 35}
{'H2O': 78.78457026123681, 'CO2': 14.358132699095183, 'S': 6.857297039668016}

{'Htot': 45, 'Ctot': 25, 'Stot': 30}
{'H2O': 76.78065706465351, 'CO2': 17.491160693054283, 'S': 5.728182242292219}

{'Htot': 45, 'Ctot': 30, 'Stot': 25}
{'H2O': 74.87615576262661, 'CO2': 20.46876266900582, 'S': 4.655081568367572}

{'Htot': 45, 'Ctot': 35, 'Stot': 20}
{'H2O': 73.06384783695462, 'CO2': 23.302224454918193, 'S': 3.633927708127191}

{'Htot': 45, 'Ctot': 40, 'Stot': 15}
{'H2O': 71.33719712384078, 'CO2': 26.001765048392297, 'S': 2.6610378277669167}

{'Htot': 45, 'Ctot': 45, 'Stot': 10}
{'H2O': 69.69027104802431, 'CO2': 28.57665976354855, 'S': 1.7330691884271383}

{'Htot': 45, 'Ctot': 50, 'Stot': 5}
{'H2O': 68.11767252127721, 'CO2': 31.035346704658597, 'S': 0.8469807740641742}

{'Htot': 45, 'Ctot': 55, 'Stot': 0}
{'H2O': 66.6144808579103, 'CO2': 33.385519142089706, 'S': 0.0}

{'Htot': 50, 'Ctot': 0, 'Stot': 50}
{'H2O': 89.93559982027857, 'CO2': 0.0, 'S': 10.064400179721432}

{'Htot': 50, 'Ctot': 5, 'Stot': 45}
{'H2O': 87.58704906653811, 'CO2': 3.5915275736879915, 'S': 8.821423359773897}

{'Htot': 50, 'Ctot': 10, 'Stot': 40}

{'H2O': 85.35803542959584, 'CO2': 7.000252689146729, 'S': 7.641711881257407}

{'Htot': 50, 'Ctot': 15, 'Stot': 35}
{'H2O': 83.23965905828565, 'CO2': 10.239785467773038, 'S': 6.520555473941311}

{'Htot': 50, 'Ctot': 20, 'Stot': 30}
{'H2O': 81.2238821959891, 'CO2': 13.32241767055163, 'S': 5.453700133459252}

{'Htot': 50, 'Ctot': 25, 'Stot': 25}
{'H2O': 79.30342726395297, 'CO2': 16.25927855441703, 'S': 4.4372941816300076}

{'Htot': 50, 'Ctot': 30, 'Stot': 20}
{'H2O': 77.47168906894011, 'CO2': 19.060469128939804, 'S': 3.4678418021200836}

{'Htot': 50, 'Ctot': 35, 'Stot': 15}
{'H2O': 75.72265890308056, 'CO2': 21.73517822683434, 'S': 2.5421628700851024}

{'Htot': 50, 'Ctot': 40, 'Stot': 10}
{'H2O': 74.05085869803517, 'CO2': 24.29178319888141, 'S': 1.6573581030834184}

{'Htot': 50, 'Ctot': 45, 'Stot': 5}
{'H2O': 72.45128371318766, 'CO2': 26.737937558165186, 'S': 0.8107787286471625}

{'Htot': 50, 'Ctot': 50, 'Stot': 0}
{'H2O': 70.91935249474272, 'CO2': 29.080647505257264, 'S': 0.0}

{'Htot': 55, 'Ctot': 0, 'Stot': 45}
{'H2O': 91.61199950071425, 'CO2': 0.0, 'S': 8.388000499285743}

{'Htot': 55, 'Ctot': 5, 'Stot': 40}
{'H2O': 89.39232980353273, 'CO2': 3.332321395864853, 'S': 7.275348800602407}

{'Htot': 55, 'Ctot': 10, 'Stot': 35}
{'H2O': 87.27767653857933, 'CO2': 6.506984873312612, 'S': 6.215338588108049}

{'Htot': 55, 'Ctot': 15, 'Stot': 30}

{'H2O': 85.26075916833098, 'CO2': 9.534920477922759, 'S': 5.204320353746279}

{'Htot': 55, 'Ctot': 20, 'Stot': 25}
{'H2O': 83.33495494391452, 'CO2': 12.426070737539513, 'S': 4.238974318545951}

{'Htot': 55, 'Ctot': 25, 'Stot': 20}
{'H2O': 81.4942262578944, 'CO2': 15.189499725282959, 'S': 3.31627401682265}

{'Htot': 55, 'Ctot': 30, 'Stot': 15}
{'H2O': 79.73305741691779, 'CO2': 17.833487980818852, 'S': 2.4334546022633603}

{'Htot': 55, 'Ctot': 35, 'Stot': 10}
{'H2O': 78.04639943935715, 'CO2': 20.365615383949255, 'S': 1.587985176693604}

{'Htot': 55, 'Ctot': 40, 'Stot': 5}
{'H2O': 76.42962171425097, 'CO2': 22.792833727547873, 'S': 0.7775445582011453}

{'Htot': 55, 'Ctot': 45, 'Stot': 0}
{'H2O': 74.8784695467849, 'CO2': 25.121530453215097, 'S': 0.0}

{'Htot': 60, 'Ctot': 0, 'Stot': 40}
{'H2O': 93.05749263908261, 'CO2': 0.0, 'S': 6.942507360917403}

{'Htot': 60, 'Ctot': 5, 'Stot': 35}
{'H2O': 90.95457118754665, 'CO2': 3.108011342758368, 'S': 5.937417469694968}

{'Htot': 60, 'Ctot': 10, 'Stot': 30}
{'H2O': 88.9445933969561, 'CO2': 6.078656664429867, 'S': 4.976749938614029}

{'Htot': 60, 'Ctot': 15, 'Stot': 25}
{'H2O': 87.02153068879373, 'CO2': 8.920845898573084, 'S': 4.05762341263318}

{'Htot': 60, 'Ctot': 20, 'Stot': 20}
{'H2O': 85.1798648238656, 'CO2': 11.642734723087633, 'S': 3.1774004530467765}

{'Htot': 60, 'Ctot': 25, 'Stot': 15}

{'H2O': 83.41453501907436, 'CO2': 14.25180271894321, 'S': 2.33366226198243}

{'Htot': 60, 'Ctot': 30, 'Stot': 10}
{'H2O': 81.72089150661922, 'CO2': 16.754922007328933, 'S': 1.5241864860518488}

{'Htot': 60, 'Ctot': 35, 'Stot': 5}
{'H2O': 80.09465463819697, 'CO2': 19.158417691571714, 'S': 0.7469276702313122}

{'Htot': 60, 'Ctot': 40, 'Stot': 0}
{'H2O': 78.53187877686103, 'CO2': 21.468121223138965, 'S': 0.0}

{'Htot': 65, 'Ctot': 0, 'Stot': 35}
{'H2O': 94.31671277893898, 'CO2': 0.0, 'S': 5.683287221061025}

{'Htot': 65, 'Ctot': 5, 'Stot': 30}
{'H2O': 92.31975732260322, 'CO2': 2.9119949155990916, 'S': 4.768247761797683}

{'Htot': 65, 'Ctot': 10, 'Stot': 25}
{'H2O': 90.40561111167692, 'CO2': 5.7032359601832745, 'S': 3.891152928139812}

{'Htot': 65, 'Ctot': 15, 'Stot': 20}
{'H2O': 88.56922790492244, 'CO2': 8.381081649827676, 'S': 3.049690445249875}

{'Htot': 65, 'Ctot': 20, 'Stot': 15}
{'H2O': 86.8059633098582, 'CO2': 10.952304517815234, 'S': 2.2417321723265755}

{'Htot': 65, 'Ctot': 25, 'Stot': 10}
{'H2O': 85.11153556289058, 'CO2': 13.423148305724984, 'S': 1.4653161313844223}

{'Htot': 65, 'Ctot': 30, 'Stot': 5}
{'H2O': 83.48199081486591, 'CO2': 15.799378584639767, 'S': 0.7186306004943301}

{'Htot': 65, 'Ctot': 35, 'Stot': 0}
{'H2O': 81.91367232956003, 'CO2': 18.08632767043997, 'S': 0.0}

{'Htot': 70, 'Ctot': 0, 'Stot': 30}

{'H2O': 95.4234864134753, 'CO2': 0.0, 'S': 4.57651358652471}

{'Htot': 70, 'Ctot': 5, 'Stot': 25}
{'H2O': 93.52296000761883, 'CO2': 2.7392364283797352, 'S': 3.7378035640014184}

{'Htot': 70, 'Ctot': 10, 'Stot': 20}
{'H2O': 91.69665990038538, 'CO2': 5.371490191059409, 'S': 2.931849908555203}

{'Htot': 70, 'Ctot': 15, 'Stot': 15}
{'H2O': 89.94032094602639, 'CO2': 7.902908659930678, 'S': 2.156770394042931}

{'Htot': 70, 'Ctot': 20, 'Stot': 10}
{'H2O': 88.2499986323403, 'CO2': 10.33917707645161, 'S': 1.4108242912081046}

{'Htot': 70, 'Ctot': 25, 'Stot': 5}
{'H2O': 86.62203950683015, 'CO2': 12.685561176545733, 'S': 0.692399316624121}

{'Htot': 70, 'Ctot': 30, 'Stot': 0}
{'H2O': 85.05305481688677, 'CO2': 14.94694518311322, 'S': 0.0}

{'Htot': 75, 'Ctot': 0, 'Stot': 25}
{'H2O': 96.40391716166319, 'CO2': 0.0, 'S': 3.5960828383368115}

{'Htot': 75, 'Ctot': 5, 'Stot': 20}
{'H2O': 94.59139439932994, 'CO2': 2.5858282690530734, 'S': 2.8227773316169733}

{'Htot': 75, 'Ctot': 10, 'Stot': 15}
{'H2O': 92.8457695772299, 'CO2': 5.0762168622072785, 'S': 2.078013560562814}

{'Htot': 75, 'Ctot': 15, 'Stot': 10}
{'H2O': 91.16340612742805, 'CO2': 7.4763538744008455, 'S': 1.360239998171116}

{'Htot': 75, 'Ctot': 20, 'Stot': 5}
{'H2O': 89.5409263691071, 'CO2': 9.79105806031239, 'S': 0.6680155705805159}

{'Htot': 75, 'Ctot': 25, 'Stot': 0}

{'H2O': 87.97518887370752, 'CO2': 12.024811126292468, 'S': 0.0}

{'Htot': 80, 'Ctot': 0, 'Stot': 20}
{'H2O': 97.27847075976024, 'CO2': 0.0, 'S': 2.7215292402397533}

{'Htot': 80, 'Ctot': 5, 'Stot': 15}
{'H2O': 95.54650235329595, 'CO2': 2.448691768849702, 'S': 2.0048058778543365}

{'Htot': 80, 'Ctot': 10, 'Stot': 10}
{'H2O': 93.87512785160072, 'CO2': 4.811714656386864, 'S': 1.3131574920123998}

{'Htot': 80, 'Ctot': 15, 'Stot': 5}
{'H2O': 92.26122205453652, 'CO2': 7.093487133425086, 'S': 0.6452908120383903}

{'Htot': 80, 'Ctot': 20, 'Stot': 0}
{'H2O': 90.70187104397023, 'CO2': 9.298128956029773, 'S': 0.0}

{'Htot': 85, 'Ctot': 0, 'Stot': 15}
{'H2O': 98.06341917945072, 'CO2': 0.0, 'S': 1.9365808205492743}

{'Htot': 85, 'Ctot': 5, 'Stot': 10}
{'H2O': 96.40540622588556, 'CO2': 2.3253684621921025, 'S': 1.2692253119223216}

{'Htot': 85, 'Ctot': 10, 'Stot': 5}
{'H2O': 94.80252699624295, 'CO2': 4.573411700514905, 'S': 0.6240613032421537}

{'Htot': 85, 'Ctot': 15, 'Stot': 0}
{'H2O': 93.2520764324601, 'CO2': 6.747923567539894, 'S': 0.0}

{'Htot': 90, 'Ctot': 0, 'Stot': 10}
{'H2O': 98.77186249246121, 'CO2': 0.0, 'S': 1.2281375075387904}

{'Htot': 90, 'Ctot': 5, 'Stot': 5}
{'H2O': 97.18194443376548, 'CO2': 2.21387139878618, 'S': 0.6041841674483338}

{'Htot': 90, 'Ctot': 10, 'Stot': 0}

```
{'H2O': 95.64240091843646, 'CO2': 4.357599081563537, 'S': 0.0}
```

```
{'Htot': 95, 'Ctot': 0, 'Stot': 5}
{'H2O': 99.41446583077017, 'CO2': 0.0, 'S': 0.5855341692298309}
```

```
{'Htot': 95, 'Ctot': 5, 'Stot': 0}
{'H2O': 97.8874227256249, 'CO2': 2.11257727437509, 'S': 0.0}
```

```
{'Htot': 100, 'Ctot': 0, 'Stot': 0}
{'H2O': 100.0, 'CO2': 0.0, 'S': 0.0}
```

1.3 Use TAVERNmag to speciate each bulk gas composition at 1 bar, 1400 °C, and fO_2 of IW-4.5

Below we have used pickle files to save having to recompute these lists every time the notebook is run. A pickle file is simply a way to save some variable to permanent memory such that it can be reimported for use later. The below code checks to see if there is a pickle file saved. If so, it simply loads the values computed previously. If not, it computes the values. The pickle files have the extension ".p" and are saved in the same location as this notebook file. Deleting them and rerunning this notebook will force the code to calculate these values anew.

1.3.1 Some notes on model optimization and computation time

As the python implementation of Iacovino (2015; TAVERNmag) is still in beta, the choice of precisely which minimization functions to use to solve various functions within the code is not yet finalized. In particular, the calculation of fugacities requires the minimization of a set of two equations with two unknowns (eqn. 9 in Iacovino, 2015) to solve for fH_2 and fS_2. In general, the choice of minimization function does not significantly change the calculated fugacities, but it can effect model instabilities. For example, the use of scipy.optimize's least_squares minimization routine results in model instabilities on the H_{tot}-S_{tot} binary at approximately H_{tot} = 70 mol% and S_{tot} = 30 mol% and extending also toward the C vertex of the teranary. Instead here we employ the GEKKO optimization library, which results in no instabilities but is extremely computationally intensive. Where scipy.optimize may take <1 minute to complete the two calls below, GEKKO may take 20 minutes.

GEKKO is currently the default behavior, but passing `opt='leastsq'` to the `tv.calculate_speciation()` function will result in the use of the scipy.optimize minimization routine.

1.3.2 Speciate molar gas compositions

We continue below to compute molar and wt% gas compositions separately. This is so that when we recast these into H_{tot}, C_{tot}, and S_{tot} space, the speciated compositions and other calculated values corresponding to each composition covers the entire H-C-S ternary evenly.

[ ]:
```
# in terms of moles
try:
    speciated_gas_list_moles_wtfrac = pickle.
 ↪load(open("speciated_gas_list_moles_wtfrac_new.p", "rb"))
except:
    speciated_gas_list_moles = []
    iterno = 0
    for gascomp in bulk_comp_list_moles:
        iterno += 1
        # create a tv.MagmaticFluid() object
        fluid_obj = tv.MagmaticFluid(gascomp, units='molpercent',␣
 ↪default_units='wtpercent')
        speciated_gas_list_moles.append(tv.
 ↪calculate_speciation(sample=fluid_obj, pressure=1, temperature=1400,
                                                                  ␣
 ↪fO2_buffer="IW", fO2_delta=-4.5).result)
        percent = iterno/len(bulk_comp_list_moles)
        status_bar(percent)

    # convert from wt% to wt frac
    speciated_gas_list_moles_wtfrac = []
    for gascomp in speciated_gas_list_moles:
        speciated_gas_list_moles_wtfrac.append({k: v/100 for k, v in gascomp.
 ↪items()})

    # Save a pickle file of this calculation, since it takes soooo long...
    pickle.dump(speciated_gas_list_moles_wtfrac,␣
 ↪open("speciated_gas_list_moles_wtfrac_new.p", "wb"))
```

1.3.3 Speciate wt% gas compositions

[ ]:
```
# in terms of wtpercent
try:
    speciated_gas_list_wtper_wtfrac = pickle.
 ↪load(open("speciated_gas_list_wtper_wtfrac_new.p", "rb"))
except:
    speciated_gas_list_wtper = []
    iterno = 0
    for gascomp in bulk_comp_list_wtper:
        iterno += 1
        #create a tv.MagmaticFluid() object
        fluid_obj = tv.MagmaticFluid(gascomp, units='wtpercent',␣
 ↪default_units='wtpercent')
        speciated_gas_list_wtper.append(tv.
 ↪calculate_speciation(sample=fluid_obj, pressure=1, temperature=1400,
```

```
↪fO2_buffer="IW", fO2_delta=-4.5).result)
        percent = iterno/len(bulk_comp_list_wtper)
        status_bar(percent)

    # convert from wt% to wt frac
    speciated_gas_list_wtper_wtfrac = []
    for gascomp in speciated_gas_list_wtper:
        speciated_gas_list_wtper_wtfrac.append({k: v/100 for k, v in gascomp.
↪items()})

    # Save a pickle file of this calculation, since it takes soooo long...
    pickle.dump(speciated_gas_list_wtper_wtfrac,␣
↪open("speciated_gas_list_wtper_wtfrac_new.p", "wb"))
```

1.4 Calculate sum total of released volatiles for set of pyroclastic deposit radii

Using equation 5 from the main text to solve for n_{Tot} the volatile weight fraction of a mixed gas:

$n_{Tot} = \frac{gX}{20RT} \cdot \sum W_i \left(\frac{m_i(\gamma_i - 1)}{\gamma_i} \right)$

Where g is the gravitational acceleration on Mercury, X is the maximum distance that particles can be ejected on an airless body in meters (radius of the pyroclastic deposit), R is the gas constant (8.3145 J-mol^{-1}-K^{-1}), T is the temperature in K, m_i is the molecular weight of gas i, γ_i is the specific heat ratio of gas i, and W_i is the weight fraction of gas i in the total gas.

Constants used here are defined above in cell "Define necessary constants". The value of X can be varied to calculate ternary diagrams for any given pyroclastic deposit radius.

[ ]:
```
X = deposit_radius #deposit radius in meters as defined by user above
```

1.4.1 Calculate n_{Tot} for molar compositions (in wt%)

- Input: Speciated gases generated from molar compositions, in weight fraction
- Output: n_{Tot} values for each composition in wt%

[ ]:
```
# using moles composition space
nTot_list_from_moles = []
for mixed_gas in speciated_gas_list_moles_wtfrac:
    nsum =␣
↪sum([mixed_gas[species]*(molecularWeight[species]*(specificHeatRatios[species]-1)/
↪specificHeatRatios[species]) for species in fluid_species_names])
    nTot_list_from_moles.append((g*X)/(20*R*T) * nsum)
```

1.4.2 Calculate n_{tot} for wt% compositions (in wt%)

- Input: Speciated gases generated from wt% compositions, in weight fraction
- Output: n_{Tot} values for each composition in wt%

```
[ ]: # using wtpercent composition space
nTot_list_from_wtper = []
for mixed_gas in speciated_gas_list_wtper_wtfrac:
    nsum =␣
 ↪sum([mixed_gas[species]*(molecularWeight[species]*(specificHeatRatios[species]-1)/
 ↪specificHeatRatios[species]) for species in fluid_species_names])
    nTot_list_from_wtper.append((g*X)/(20*R*T) * nsum)
```

1.4.3 Convert molar n_{tot} values from wt% to mol%

Next, in order to plot n_{Tot} values both in terms of wt% and mol%, we must convert the output for our molar composition from wt% into mol%. To do this, we need to consider the bulk silicate melt composition. Here we use the composition of S-free Sil-3 from the main text.

```
[ ]: # for moles composition space
# Recalculate nTot values as moles using S-free Sil-3 as bulk composition of␣
 ↪melt

silicate_composition_wtpercent = {'SiO2': 54.05,
                                  'TiO2': 1.26,
                                  'Al2O3': 12.60,
                                  'Cr2O3': 0.74,
                                  'FeO': 3.40,
                                  'MnO': 0.65,
                                  'MgO': 14.25,
                                  'CaO': 5.23,
                                  'K2O': 0.25,
                                  'Na2O': 6.81}
nTot_list_as_moles = []
for i, mixed_gas in enumerate(speciated_gas_list_moles_wtfrac):
    bulk_silicate_w_volatiles = silicate_composition_wtpercent
    for species in fluid_species_names:
        bulk_silicate_w_volatiles[species] =␣
 ↪nTot_list_from_moles[i]*speciated_gas_list_moles_wtfrac[i][species]
    mol_prop_ox = {k: v/molecularWeight[k] for k, v in␣
 ↪bulk_silicate_w_volatiles.items()}
    bulk_silicate_w_volatiles_moles = {k: v/sum(mol_prop_ox.values()) for k, v␣
 ↪in mol_prop_ox.items()}
    nTot_as_moles = 100*sum(v for k, v in bulk_silicate_w_volatiles_moles.
 ↪items() if k in fluid_species_names)
    nTot_list_as_moles.append(nTot_as_moles)
```

1.4.4 Plot results (as gas)

Here we generate Figures 5a and 5b in the main text. These show the amount of gas in mol% or wt%, respectively, required to produce pyroclastic deposits of a given radius (as defined by variable X above) at 1 bar, 1400 °C and an fO_2 of IW-4.5. Note that in these plots, we consider all gas

species.

```
[ ]: # molpercent composition space
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

fig = ff.create_ternary_contour(np.array([Htot, Ctot, Stot], dtype=float), np.
 ↪array(nTot_list_as_moles, dtype=float),
                                pole_labels=['$H_{tot}$', '$C_{tot}$',␣
 ↪'$S_{tot}$'],
                                interp_mode='cartesian',
                                ncontours=20,
                                colorscale='Viridis',
                                showscale=True,
                                showmarkers=False,
                                title='As gas, in mol%')
fig.show()
fig.write_image("as_gas_molper.svg")

# wtpercent composition space
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

fig = ff.create_ternary_contour(np.array([Htot, Ctot, Stot], dtype=float), np.
 ↪array(nTot_list_from_wtper, dtype=float),
                                pole_labels=['$H_{tot}$', '$C_{tot}$',␣
 ↪'$S_{tot}$'],
                                interp_mode='cartesian',
                                ncontours=20,
                                colorscale='Viridis',
                                showscale=True,
                                showmarkers=False,
                                title='As gas, in wt%')
fig.show()
fig.write_image("as_gas_wtper.svg")
```

1.5 Calculate dissolved C, H, S in magma (no solubility limits, no smelting)

Next, we can calculate the concentration of dissolved C, H, and S in a magma that would generate the gas compositions plotted in the previous section. We first do this calculation without regard to solubility limits.

First, we convert our speciated gases back into C_{tot}, H_{tot}, and S_{tot} and then sum these. We will do this in wt% composition space.

1.5.1 Calculate dissolved volatiles for wt% compositions (in wt%)

[ ]:
```
#using wtpercent composition space
C_dissolved = []
H_dissolved = []
S_dissolved = []
tot_dissolved = []
for i in range(len(nTot_list_from_wtper)):
    C_from_CO =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['CO']*molecularWeight['C']/
 ↪molecularWeight['CO']
    C_from_CO2 =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['CO2']*molecularWeight['C']/
 ↪molecularWeight['CO2']
    H_from_H2 =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['H2']*molecularWeight['H']/
 ↪molecularWeight['H2']
    H_from_H2O =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['H2O']*molecularWeight['H']/
 ↪molecularWeight['H2O']
    H_from_H2S =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['H2S']*molecularWeight['H']/
 ↪molecularWeight['H2S']
    S_from_S2 =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['S2']*molecularWeight['S']/
 ↪molecularWeight['S2']
    S_from_SO2 =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['SO2']*molecularWeight['S']/
 ↪molecularWeight['SO2']
    S_from_H2S =␣
 ↪nTot_list_from_wtper[i]*speciated_gas_list_wtper_wtfrac[i]['H2S']*molecularWeight['S']/
 ↪molecularWeight['H2S']
    C_dissolved.append(C_from_CO + C_from_CO2)
    H_dissolved.append(H_from_H2 + H_from_H2O + H_from_H2S)
    S_dissolved.append(S_from_S2 + S_from_SO2 + S_from_H2S)
    tot_dissolved.append(C_from_CO + C_from_CO2 + H_from_H2 + H_from_H2O +␣
 ↪H_from_H2S + S_from_S2 + S_from_SO2 + S_from_H2S)
```

1.5.2 Convert dissolved values into mol% space (in mol%)

Next, we can convert these dissolved volatile concentrations from wt% to mol%. To do this, we need to consider the bulk silicate melt composition. Here we use the composition of S-free Sil-3 from the main text.

[ ]:
```
# Recalculate tot_dissolved values as moles using S-free Sil-3 as bulk␣
 ↪composition of melt
silicate_composition_wtpercent_diss = {'SiO2': 54.05,
```

```
                                    'TiO2': 1.26,
                                    'Al2O3': 12.60,
                                    'Cr2O3': 0.74,
                                    'FeO': 3.40,
                                    'MnO': 0.65,
                                    'MgO': 14.25,
                                    'CaO': 5.23,
                                    'K2O': 0.25,
                                    'Na2O': 6.81}

tot_diss_as_moles = []
for i in range(len(tot_dissolved)):
    bulk_with_volatiles_diss = silicate_composition_wtpercent_diss
    bulk_with_volatiles_diss['C'] = C_dissolved[i]
    bulk_with_volatiles_diss['H'] = H_dissolved[i]
    bulk_with_volatiles_diss['S'] = S_dissolved[i]

    #convert to mol%
    mol_prop_ox = {k: v/molecularWeight[k] for k, v in bulk_with_volatiles_diss.
 ↪items()}
    bulk_with_volatiles_diss_moles = {k: 100*v/sum(mol_prop_ox.values()) for k,␣
 ↪v in mol_prop_ox.items()}
    tot_diss_as_moles.append(bulk_with_volatiles_diss_moles['C'] +␣
 ↪bulk_with_volatiles_diss_moles['H'] +
                             bulk_with_volatiles_diss_moles['S'])
```

1.5.3 Plot results (dissolved volatiles, no solubility limits, no smelting)

Here we generate Figures 5c and 5d in the main text. These demonstrate the corresponding amount of dissolved H, C, and/or S in mol% or wt%, respectively, required to produce the same deposits modeled above assuming volatiles sourced only from the magma (not constrained by solubility limits). Note that in these plots, we consider dissolved H, C, and S only whereas the previous plots considered all gas species. For example, here all C species are computed in as the concentration of C (as is typically done when reporting dissolved volatile contents, since particular complexing in the melt may not be known). In the previous plots, CO and CO_2 are both considered in the gas.

[ ]:
```
# molpercent composition space
# bulk_comp_list
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

# use nTot_list a function

fig = ff.create_ternary_contour(np.array([Htot, Ctot, Stot], dtype=float), np.
 ↪array(tot_diss_as_moles, dtype=float),
```

```
                                pole_labels=['$H_{diss}$', '$C_{diss}$',␣
 ↪'$S_{diss}$'],
                                interp_mode='cartesian',
                                ncontours=20,
                                colorscale='YlOrRd',
                                showscale=True,
                                showmarkers=False,
                                title='Dissolved volatiles, in mol% (without␣
 ↪smelting)')
fig.show()
fig.write_image("diss_molper_nosmelt.svg")

# wtpercent composition space
# bulk_comp_list
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

# use nTot_list a function

fig = ff.create_ternary_contour(np.array([Htot, Ctot, Stot], dtype=float), np.
 ↪array(tot_dissolved, dtype=float),
                                pole_labels=['$H_{diss}$', '$C_{diss}$',␣
 ↪'$S_{diss}$'],
                                interp_mode='cartesian',
                                ncontours=20,
                                colorscale='YlOrRd',
                                showscale=True,
                                showmarkers=False,
                                title='Dissolved volatiles, in wt% (without␣
 ↪smelting)')
fig.show()
fig.write_image("diss_wtper_nosmelt.svg")
```

1.6 Calculate dissolved C, H, S in magma *in addition to smelting* (no solubility limits)

The above ternaries illustrate the dissolved volatile concentrations corresponding to the gas releases needed to generate pyroclastic deposits of a given radius. That assumes no solubility limits on the magma. It also assumes that all of the gas released originated from dissolved volatiles in a melt and does not consider the additional volatiles that may be sourced via smelting.

Our experiments indicate a maximum of 5 wt% FeO smelted in the melt, which would require the smelting of <1 wt% graphite and result in ~2 wt% pure CO gas or an equivalent deposit radius of ~21,000 m. Given smelting of 5 wt% FeO in the melt, we can calculate the mass of *additional* volatiles needed to produce deposits larger than 21,000 m.

1.6.1 First, calculate n_{tot} for volatiles in addition to smelting

[ ]:
```
# using wtpercent composition space, only volatiles needed in addition to␣
 ↪smelting
nTot_list_from_wtper_minus_smelting = []
for mixed_gas in speciated_gas_list_wtper_wtfrac:
    nsum =␣
 ↪sum([mixed_gas[species]*(molecularWeight[species]*(specificHeatRatios[species]-1)/
 ↪specificHeatRatios[species]) for species in fluid_species_names])
    X_wSmelt = X-21000
    if X_wSmelt <0:
        X_wSmelt = 0
    nTot_list_from_wtper_minus_smelting.append((g*(X_wSmelt))/(20*R*T) * nsum)
```

1.6.2 Second, calculate dissolved H, C, and S in magma required in addition to smelting

[ ]:
```
#using wtpercent compsition space, in addition to volatiles from smelting
C_dissolved_wSmelt = []
H_dissolved_wSmelt = []
S_dissolved_wSmelt = []
tot_dissolved_wSmelt = []
for i in range(len(nTot_list_from_wtper_minus_smelting)):
    C_from_CO =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['CO']*molecularWe
 ↪molecularWeight['CO']
    C_from_CO2 =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['CO2']*molecularW
 ↪molecularWeight['CO2']
    H_from_H2 =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['H2']*molecularWe
 ↪molecularWeight['H2']
    H_from_H2O =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['H2O']*molecularW
 ↪molecularWeight['H2O']
    H_from_H2S =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['H2S']*molecularW
 ↪molecularWeight['H2S']
    S_from_S2 =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['S2']*molecularWe
 ↪molecularWeight['S2']
    S_from_SO2 =␣
 ↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['SO2']*molecularW
 ↪molecularWeight['SO2']
```

```
    S_from_H2S =␣
↪nTot_list_from_wtper_minus_smelting[i]*speciated_gas_list_wtper_wtfrac[i]['H2S']*molecularW
↪molecularWeight['H2S']
    C_dissolved_wSmelt.append(C_from_CO + C_from_CO2)
    H_dissolved_wSmelt.append(H_from_H2 + H_from_H2O + H_from_H2S)
    S_dissolved_wSmelt.append(S_from_S2 + S_from_SO2 + S_from_H2S)
    tot_dissolved_wSmelt.append(C_from_CO + C_from_CO2 + H_from_H2 + H_from_H2O␣
↪+ H_from_H2S + S_from_S2 + S_from_SO2 + S_from_H2S)
```

1.6.3 Finally, plot ternary results for dissolved volatiles (without solubility limits) in addition to smelting

Here we plot Figure 5e from the main text. This figure demonstrates the amount of dissolved H, C, and/or S in wt% required to produce deposits greater than 21,000 m in radius in addition to ~2 wt% CO produced via the smelting of 5 wt% FeO.

[ ]:
```
# wtpercent composition space in addition to smelting
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

fig = ff.create_ternary_contour(np.array([Htot, Ctot, Stot], dtype=float), np.
↪array(tot_dissolved_wSmelt, dtype=float),
                                pole_labels=['$H_{diss}$', '$C_{diss}$',␣
↪'$S_{diss}$'],
                                interp_mode='cartesian',
                                ncontours=20,
                                colorscale='Cividis',
                                showscale=True,
                                showmarkers=False,
                                title='Dissolved volatiles, in wt% (with␣
↪smelting of 5 wt% FeO)')
fig.show()
fig.write_image("diss_wtper_smelt.svg")
```

1.7 Calculate plausible dissolved C, H, S in magma (with solubility limits)

Taken alone, the above calculations do not consider volatile solubility. That is, we set no a priori limit to the amount of H, C, or S that a mercurian magma could hold. Within this set of all possible gas compositions, we can use solubility constraints to narrow down the range of plausible dissolved volatile contents necessary to drive the largest pyroclastic deposits on Mercury. Here we assume maximum dissolved volatile concentrations in reduced mercurian magmas of ~1000 ppm total C (Li et al., 2017), ~1–5 wt% total S (Namur et al., 2016), and ~3000 ppm total H (Hirschmann et al., 2012). H and C maxima are taken based on the highest H and C reported in solubility experiments performed on basalts at reducing conditions (Li et al., 2017; Hirschmann et al., 2012). The S solubility is extremely dependent upon temperature. The Sil-3 composition used in our experiments (representing *Borealis Planitia*) at 1340 °C and 2.5 GPa can contain up to ~1 wt% S

(modeled with Namur et al., 2016). More refractory compositions like that of the High Magnesium Region require higher temperatures to be molten and therefor have higher modeled S solubilities, up to around 5 wt% S. Here, we take the upper bound of 5 wt% as an estimate of the maximum S carrying capacity for any mercurian melt.

Solubility limits used here: - C = 1000 ppm (Li et al., 2017) - H = 3000 ppm (Hirschmann et al., 2017) - S = 5 wt% (Namur et al., 2016)

1.7.1 Plot all plausible dissolved volatile concentrations with or withour smelting

Only deposits with radii 60 km produce any plausible dissolved volatile concentrations. Change the value of X above to see how these plots change with deposit radius.

[ ]:
```
# mol composition space, without smelting
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

plausible_pts_H = []
plausible_pts_C = []
plausible_pts_S = []
for i in range(len(Htot)):
    H_diss = tot_diss_as_moles[i] * Htot[i]/100
    C_diss = tot_diss_as_moles[i] * Ctot[i]/100
    S_diss = tot_diss_as_moles[i] * Stot[i]/100
    if H_diss <= 14.25 and C_diss <= 0.4 and S_diss <= 7.48:
        plausible_pts_H.append(Htot[i])
        plausible_pts_C.append(Ctot[i])
        plausible_pts_S.append(Stot[i])

fig = px.scatter_ternary(tot_dissolved, a=plausible_pts_H, b=plausible_pts_C,␣
 ↪c=plausible_pts_S)
print("Mol% no smelting")
fig.show()
fig.write_image("diss_mol_no_smelting_plausible.svg")

# wtpercent composition space, without smelting
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

plausible_pts_H = []
plausible_pts_C = []
plausible_pts_S = []
for i in range(len(Htot)):
    H_diss = tot_dissolved[i] * Htot[i]/100
    C_diss = tot_dissolved[i] * Ctot[i]/100
    S_diss = tot_dissolved[i] * Stot[i]/100
```

```
    if H_diss <= 0.3 and C_diss <= 0.1 and S_diss <= 5.0:
        plausible_pts_H.append(Htot[i])
        plausible_pts_C.append(Ctot[i])
        plausible_pts_S.append(Stot[i])
plausible_pts_H

fig = px.scatter_ternary(tot_dissolved, a=plausible_pts_H, b=plausible_pts_C,␣
 ↪c=plausible_pts_S)
print("Wt% no smelting")
fig.show()
fig.write_image("diss_no_smelting_plausible.svg")

# wtpercent composition space, with smelting
Htot = [x['Htot'] for x in bulk_comp_list_simp]
Ctot = [x['Ctot'] for x in bulk_comp_list_simp]
Stot = [x['Stot'] for x in bulk_comp_list_simp]

plausible_pts_H = []
plausible_pts_C = []
plausible_pts_S = []
for i in range(len(Htot)):
    H_diss = tot_dissolved_wSmelt[i] * Htot[i]/100
    C_diss = tot_dissolved_wSmelt[i] * Ctot[i]/100
    S_diss = tot_dissolved_wSmelt[i] * Stot[i]/100
    if H_diss <= 0.3 and C_diss <= 0.1 and S_diss <= 5.0:
        plausible_pts_H.append(Htot[i])
        plausible_pts_C.append(Ctot[i])
        plausible_pts_S.append(Stot[i])

fig = px.scatter_ternary(tot_dissolved_wSmelt, a=plausible_pts_H,␣
 ↪b=plausible_pts_C, c=plausible_pts_S)
print("wt% with smelting")
fig.show()
fig.write_image("diss_with_smelting_plausible.svg")
```

```
Mol% no smelting
```

```
Wt% no smelting
```

```
wt% with smelting
```

[ ]: