BurnMan

CIG webinar
Oct 8th 2015

Sanne Cottaar, Timo Heister, Bob Myhill, Ian Rose, Cayman Unterborn

https://geodynamics.org/cig/software/burnman/
Introduction

- What is BurnMan?
  - Python library + examples
  - open source
  - modular
  - easy scripts (no user interface)
  - thermoelastic toolkit

- GOAL: Self-consistently compute the mineral properties for the Earth (and other planets)
History of BurnMan

- Initial question at CIDER 2012 workshop: ‘What is the Mg/Si ratio of the lower mantle?’ or ‘Do the upper and lower mantle have the same major element composition?’
- Started diving into the forward model - from a composition to seismic velocities.
- Realized that:
  - It is often hard to reproduce the results of a paper.
  - Many people do the same problem with various different codes and excel sheets.
- Started to produce an open source code to improve reproducibility.
Toolbox overview

Composition

Equation of state

Thermoelastic parameters

Seismic Velocities
Detailed Overview

- **Geotherm (4.2)**
  Define range of pressures and temperatures (e.g. compute the adiabat)

- **Minerals (4.1)**
  Define composition from mineral database or user supplied minerals

- **Equation of state (3.1)**
  Choose or supply a thermo-elastic model.

- **Averaging scheme (3.2)**
  Choose predefined schemes (Voigt/Reuss/Hill/Hashin-Shtrikman) or user defined.

- **Observed seismic velocities (4.3)**
  Supplied 1D seismic profiles (e.g. PREM/AK135) or user-supplied seismic model

- **planetary conditions and composition**

- **elastic properties**

- **multi-phase seismic velocities**

- **visual comparison or misfit**
Equations of state

\[ \rho = f(P, T, X) \]

Isothermal EoS:
- Birch-Murnaghan (2nd and 3rd order)
- Modified Tait (Holland & Powell, 2011)

Full thermal EoS:
- Birch-Murnaghan with a Mie-Gruneisen-Debye thermal part
- Stixrude and Lithgow-Bertelloni variant of the same
- Holland and Powell (2001) extension of the Modified Tait

examples/example_compare_all_methods.py
Equations of state: the bigger picture

Inputs

$P, T, \chi$

Outputs

$\mu, \gamma, G, H, F, S$

$\rho, \alpha, K_T, K_S, C_p, C_v$
Geotherms

$T = f(r \text{ or } P)$

examples/example_geotherms.py
Mineral libraries

Parameter libraries are (amongst others) included for mantle mineral from Stixrude and Lithgow-Bertelloni (2011) and Holland & Powell (2011, 2013).

Exact parameters depend on the equation of state.

Easy to add your own minerals.

Note: everything in BurnMan is in SI units.
Solid solutions

Different models to compute the state of a solid solution:

Ideal solution

(A)symmetric regular solution (Holland & Powell, 2003)

Subregular solution (Helffrich and Wood, 1993)
We have some combination of minerals/solid solutions (a rock) and would like to know its:

- density
- bulk modulus
- shear modulus
- seismic wavespeeds
Averaging schemes

Voigt bound

Reuss bound

Hashin-Shtrikman bound

http://petrowiki.spe.org/
Averaging schemes

- Example: different ways to mix a rock made out of 40% periclase and 60% Mg-perovskite

- Does the choice of averaging scheme matter? Sometimes!
Seismic models

Currently 1D radial models only. Models can be queried for seismic moduli when density is included.

PREM Dziewonski & Anderson (1981)
STW105 Kustowski, Ekström and Dziewonski (2008)
AK135 Kennett, Engdahl & Buland (1995)
IASP91 Kennett & Engdahl (1991)

Note: models are in SI units in BurnMan, but here converted during plotting.
User defined modules

- equations of state
- mineral and solid solution libraries
- solid solution models
- geotherms
- seismic models
- averaging schemes
- layers within planet

And contribute them to BurnMan for others to use!
Part 2: Some more complete examples
Example script

This example illustrates defining a composition and geotherm and computing seismic properties for the lowermost mantle and compare to PREM.
import numpy as np
import matplotlib.pyplot as plt

import burnman
from burnman import minerals

# For the preset minerals from the SLB_2011, the equation of state formulation
# from Stixrude and Lithgow-Bertolloni (2005) will be used.
rock = burnman.Composite([0.8, 0.2], [minerals.SLB_2011.mg_perovskite(),
                            minerals.SLB_2011.periclase()])

# Here we create and load the PREM seismic velocity model, which will be
# used for comparison with the seismic velocities of the "rock" composite
seismic_model = burnman.seismic.PREM()

# We create an array of 20 depths at which we want to evaluate PREM, and then
# query the seismic model for the pressure, density, P wave speed, S wave
# speed, and bulk sound velocity at those depths
depths = np.linspace(750e3, 2800e3, 20)
pressure, seis_rho, seis_vp, seis_vs, seis_vphi = seismic_model.evaluate_all_at(depths)

# Here we use the Brown & Shankland (1981) geotherm for mapping temperature to pressure
temperature = burnman.geotherm.brown_shankland(pressure)

# Here is the step which does the heavy lifting.
density, vp, vs, vphi, K, G = burnman.velocities_from_rock(rock, pressure, temperature)
Reproducibility

One of the goals of BurnMan is to be able to reproduce studies, and use parameters published by previous studies in a self-consistent way.

This example illustrates the importance of using the same equation-of-state as was used when fitting the data. This is especially important when extrapolating to high pressures.
Mg/Si ratio in the lower mantle

Goal: Finding the ratio of bridgmanite (perovskite) to ferropericlase in the lower mantle by fitting PREM.

import numpy as np
import matplotlib.pyplot as plt

import burnman
from burnman import minerals

# Define reference model and depth to evaluate
seismic_model = burnman.seismic.PREM()
number_of_points = 20
depths = np.linspace(700e3, 2800e3, number_of_points)
seis_p, seis_rho, seis_vp, seis_vs, seis_vphi, seis_K, seis_G = 
    seismic_model.evaluate(['pressure', 'density', 'v_p', 'v_s', 'v_phi', 'K', 'G'], depths)

# Define geotherm
temperature = burnman.geotherm.brown_shankland(seis_p)

# Define solid solutions
perovskite = minerals.SLB_2011.mg_fe_perovskite()
perovskite.set_composition([0.94, 0.06, 0.1])  # Set molar_fraction of mg_perovskite, fe_perovskite and al_perovskite
ferropericline = minerals.SLB_2011.ferropericline()
ferropericline.set_composition([0.8, 0.2])  # Set molar_fraction of MgO and FeO

def material_error(amount_perovskite):
    # Define composite using the values
    rock = burnman.Composite([amount_perovskite, 1.0 - amount_perovskite],
        [perovskite, ferropericline])

    # Compute velocities
    mat_rho, mat_vp, mat_vs, mat_vphi, mat_K, mat_G = 
        burnman.velocities_from_rock(rock, seis_p, temperature, burnman.averaging_schemes.VoigtReussHill())

    print "Calculations are done for:"
    rock.debug_print()

    # Calculate errors
    [vs_err, vphi_err, rho_err, K_err, G_err] = 
        burnman.compare_l2(depths, [mat_vs, mat_vphi, mat_rho, mat_K, mat_G], [seis_vs, seis_vphi, seis_rho, seis_K, seis_G])

    return vs_err, vphi_err, rho_err, K_err, G_err

# Run through fractions of perovskite
xx = np.linspace(0.0, 1.0, 40)
errs = np.array([material_error(x) for x in xx])
Part 3: Research Projects
Application to Research:
Predicting velocity and density contrasts at 660 km

P-to-s conversions or receiver functions

Akimotoite at the bottom of a cold slab

wd = wadselyite
mj = majorite
aki = akimotoite
rw = ringwoodite
pc = periclase
brm = bridgmanite or perovskite
Application to Research: Predicting velocity and density contrasts at 660 km

Scenario 1
ringwoodite -> perovskite

Scenario 2
ringw. -> akimotoite -> perov.

Shear wave velocities around 660 km (km/s)

Resulting synthetic receiver functions

#Example computing impedance contrasts at 660 km

# Defining a rock using a predefined solid solution from the mineral library database.

pv=minerals.SLB_2011.mg_fe_perovskite(molar_fractions=[0.9,0.1,0.])
aki=minerals.SLB_2011.akimotoite(molar_fractions=[0.9,0.1,0.])
ring=minerals.SLB_2011.mg_fe_ringwoodite(molar_fractions=[0.9,0.1])
pc=minerals.SLB_2011.ferropericlase(molar_fractions=[0.9,0.1])

#Define Composites
rock = burnman.Composite([0.8, 0.2], phases=[pv, pc])
rock2 = burnman.Composite([0.8, 0.2], phases=[aki, pc])
rock3 = burnman.Composite([0.6, 0.4], phases=[ring, aki])

#seismic model for comparison:
# pick from .prem() .slow() .fast() (see burnman/seismic.py)

number_of_points = 20 #set on how many depth slices the computations should be done
# we will do our computation and comparison at the following depth values:
depths = np.linspace(500e3, 800e3, number_of_points)
Planetary reference models

\[ \rho \left( P, T \right) \]

\[ g \left( \rho \right) \rightarrow P \left( \rho, g \right) \]
Example: internal structure of Mercury

rock=?
Reference model for mantle convection simulation
Application to Research: How Interior Composition Affects Exoplanet Observables

\[
\frac{dm(r)}{dr} = 4\pi r^2 \rho(r)
\]

\[
\frac{dP(r)}{dr} = -\frac{Gm(r)\rho(r)}{r^2}
\]

\[
P(r) = f(\rho(r), T(r))
\]
Application to Research: How Interior Composition Affects Exoplanet Observables
The Future

- release at the end of the year:
  - Clean up the hierarchy of different materials for clarity and consistency
  - Performance improvements
  - Python 3 compatibility

- long term:
  - Gibbs free energy minimization and phase equilibria
  - coupling with ASPECT
  - 3D seismic variations
Thank you!

This collaboration is initiated at and funded through CIDER (www.deep-earth.org).

BurnMan is hosted by Computational Infrastructure for Geodynamics (CIG).

Thanks to many for discussions and input!

Anyone is welcome to get involved! If you have an idea, contact us!

Section 4: Live Tutorial

- tutorial demonstration
- you can find them in burnman under tutorial/step_* .py