BQS EoS Module

MUSES collaboration meeting
May 16th-17th 2023

Developed by: *Hitansh Shah*, ***Johannes Jahan*** (MUSES module)
P. Parotto, J. Karthein (initial core code)
C. Ratti

Overview

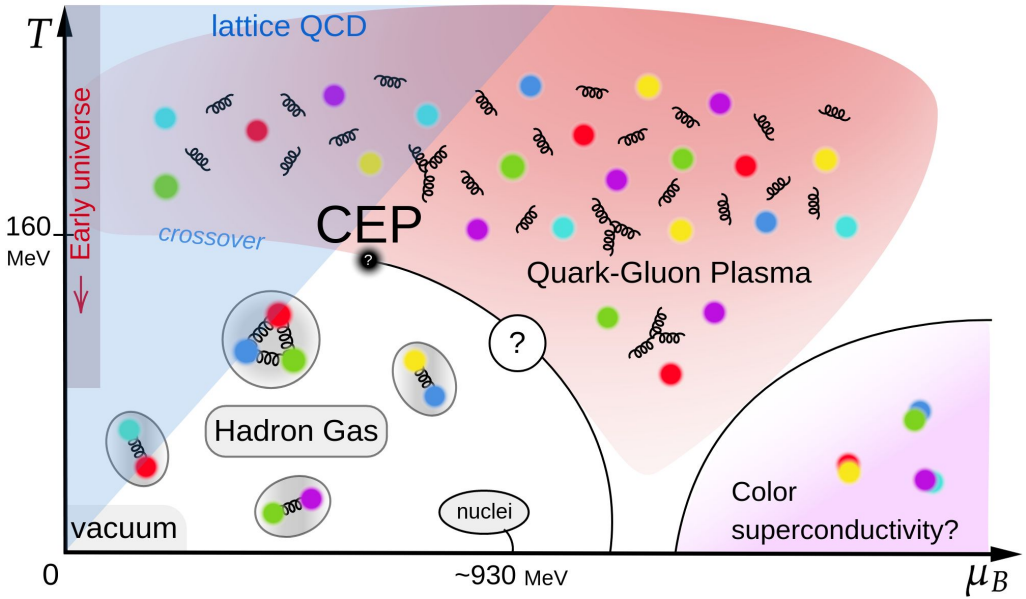
- Physics context
- Module Status
- Inputs format
- Outputs format
- Module core code

Physics content

Based on a code written by P. Parotto & J. Karthein *(after the work presented in Phys.Rev.C 100 (2019) 6, 064910).*

Calculation of pressure via a Taylor expansion towards the different $\mu_{B,Q,S}$ directions:

$$\frac{P(T,\mu_B,\mu_Q,\mu_S)}{T} = \sum_{i,j,k} \frac{1}{i!j!k!} \chi^{BQS}_{ijk} \left(\frac{\mu_B}{T}\right)^i \left(\frac{\mu_Q}{T}\right)^j \left(\frac{\mu_S}{T}\right)^k$$



Physics content

Based on a code written by P. Parotto & J. Karthein *(after the work presented in Phys.Rev.C 100 (2019) 6, 064910).*

Calculation of pressure via a Taylor expansion towards the different $\mu_{B,Q,S}$ directions:

$$\frac{P(T,\mu_B,\mu_Q,\mu_S)}{T} = \sum_{i,j,k} \frac{1}{i!j!k!} \chi_{ijk}^{BQS} \left(\frac{\mu_B}{T}\right)^i \left(\frac{\mu_Q}{T}\right)^j \left(\frac{\mu_S}{T}\right)^k$$

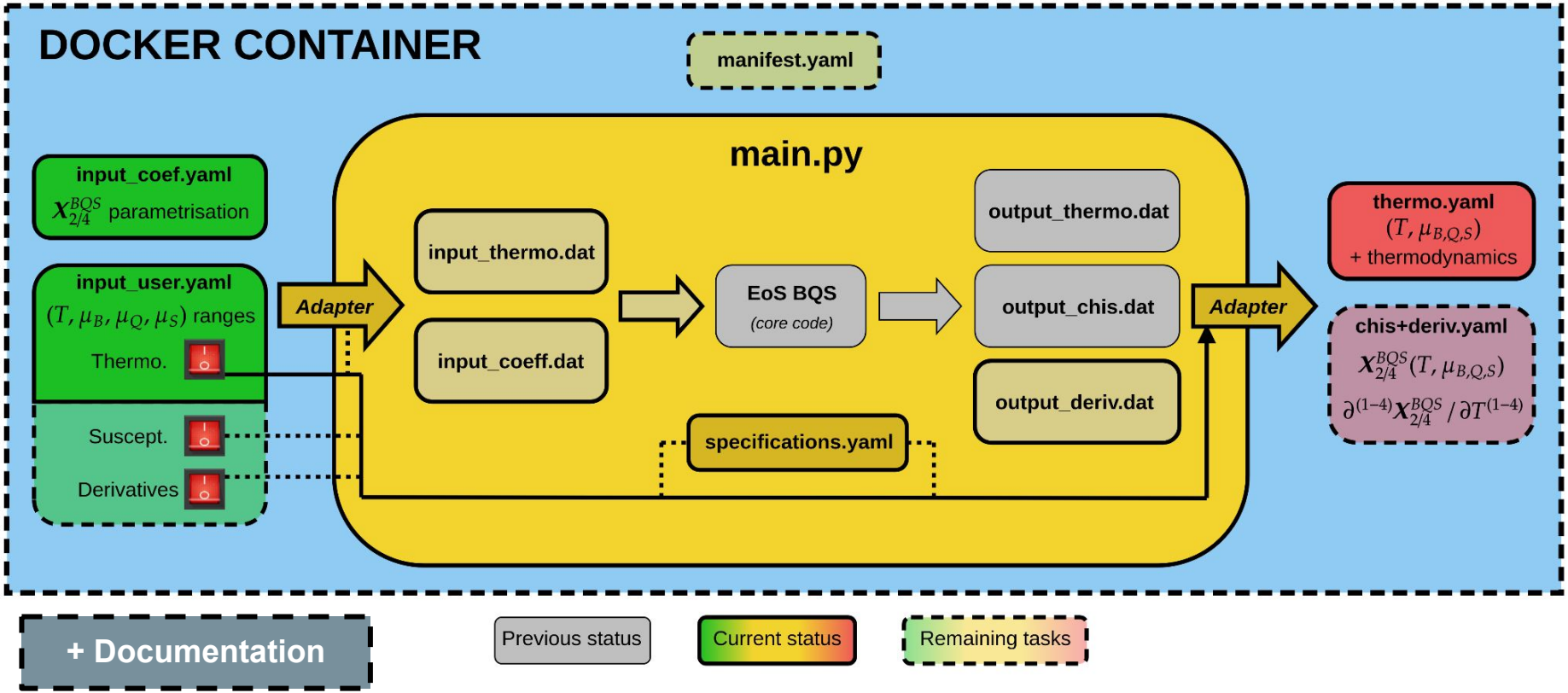
with parametrised susceptibilities:

- $\chi_2^B(T)) = e^{-h_1/t' - h_2/t'} \times f_3\left(1 + \tanh(f_4 t' + f_5)\right) \qquad \text{where } t' = T/200\,\text{MeV}$

- $\chi_{ijk}^{BQS}(T) = \frac{\sum_{n=0}^{9} a_n^{ijk}/t^n}{\sum_{n=0}^{9} b_n^{ijk}/t^n} + c_0^{ijk} \qquad \text{where } t = T/154\,\text{MeV}$

Applicable within the ranges $T \in [30, 800]$ MeV & $\mu_i \in [0, 450]$ MeV

Module status



Inputs format

BQS_EoS_input_user.yaml

$[T^{min}, T^{max}]$; δT (MeV)

def: [30, 800] ; 5 (MeV)

$[\mu^{min}_{B/Q/S}, \mu^{max}_{B/Q/S}]$; $\delta\mu_{B/Q/S}$ (MeV)

def: [0, 450] ; 5 (MeV)

```
get_P:
  description: Indicates if pressure values should be stored in the output
  default: True
  value:
```

+ ε/T^4 (**def:** False)
  s/T^3 (**def:** False)
  c_s^2 (**def:** False)
  n_B/T^3 (**def:** False)
  n_Q/T^3 (**def:** False)
  n_S/T^3 (**def:** False)



$$\left(\chi^{BQS}_{ijk}(T, \mu_{B,Q,S}) \right.$$
$$\frac{\partial \chi^{BQS}_{ijk}}{\partial T}(T, \mu_{B,Q,S})$$
$$\left. \frac{\partial^2 \chi^{BQS}_{ijk}}{\partial T^2}(T, \mu_{B,Q,S}) \right)$$

BQS_EoS_input_coef.yaml

$h_1,\ h_2,\ f_3,\ f_4,\ f_5,\ a^{ijk}_{0-9},\ b^{ijk}_{0-9},\ c^{ijk}_0$ to parametrise χ^{BQS}_{ijk}

Outputs format

BQS_EoS_thermo.yaml

item:

- T (MeV)
  μ_B (MeV)
  μ_Q (MeV)
  μ_S (MeV)
  P (MeV/fm^3)
  n_B/T^3 (MeV^3)
  n_Q/T^3 (MeV^3)
  n_S/T^3 (MeV^3)
  ε/T^4 (MeV^4)
  s/T^3 (MeV^3)
  c_s^2

BQS_EoS_deriv.yaml

item:

- T (MeV)
  μ_B (MeV)
  μ_Q (MeV)
  μ_S (MeV)
  χ_{ijk}^{BQS} $(i+j+k=0,2,4)$
  $\partial^n \chi_{ijk}^{BQS}/\partial T^n$ $(n=1,2,3?,4?)$

Module core code

New potential features:

- higher order derivatives $\frac{\partial^{3/4}\chi_{ijk}^{BQS}}{\partial T^{3/4}}(T,\mu_{B,Q,S})$ **(?)**

Optimisation:

Currently, code takes **~30min** to calculate the **whole 4D phase diagram** *(112M points @ δ=5 MeV)*

~30s for full 3D (1.25M points) / ~4.5s for full 2D (14k points)

➢ Adding **switchers** to the code itself to **avoid unnecessary variable calculation** might reduce this time

➢ **Parallelisation** of the **code** **(?)**

…. and what about the **Python converters**?

Takes ~1min to convert data for a full 2D phase diagram

➢ **Continuous dump** mandatory *(to avoid temporary memory space saturation)*
+ **parallelisation** of the converter *(doable with YAML dumper..?)*

Summary

Current status:

- ❖ Code works and run properly for the full 4D phase diagram
  $$T \in [30, 800]\ \mathrm{MeV}\ \ \&\ \ \mu_i \in [0, 450]\ \mathrm{MeV}$$
- ❖ Wrapping architecture (YAML in / out) complete

Left to do:

- ❑ Containerise the “vanilla” version of the code *(provides only thermodynamics)*
- ❑ Integrate the module into calculation engine
- ❑ Add susceptibilities and derivatives in the YAML output
- ❑ Discuss addition of additional derivatives
- ❑ Optimise the code *(switchers + parallelisation of core code / Python converters)*
- ❑ Complete the documentation