Webnucleo Technical Report: libstatmech

Tianhong Yu, Bradley S. Meyer

April 19, 2012

This technical report describes some details of calculations in the libstatmech module.

1 Introduction

libstatmech is a library of C codes for computing the thermodynamics of fermion and boson gases. It is built on top of libxml, the GNOME C xml toolkit. Users can create fermions and bosons and calculate their thermodynamic quantities numerically with either default integrands (fully relativistic, non-interacting particles) or user-supplied ones or with user-defined functions. Users can also define their own thermodynamics quantities as functions or integrands. A well-documented API allows users to incorporate libstatmech into their own codes, and examples in the libstatmech distribution demonstrate the API.

2 Fermions and Bosons

Fermions and bosons are stored as Libstatmech__Fermion and Libstatmech__Boson structures, respectively in libstatmech. To create a fermion or boson, the user must supply the particle's name, its rest mass in MeV, its multiplicity (usually $2J + 1$, where J is the particle's spin), and its charge (in units of the proton's charge). When a fermion or boson is created, it automatically has attached to it four thermodynamic quantities, namely, the number density, the pressure, the energy density, and the entropy density. These quantities are computed in cgs units. Once a fermion or boson is created, a user may compute one of the four thermodynamic quantities by the API routine Libstatmech__Fermion__computeQuantity() or Libstatmech__Boson__computeQuantity(). These routines will compute the various quantities by numerical integration of the default integrands, which are those for fully relativistic, non-interacting particles (see §3 for the definition of these integrals). The user may also invert the number density integrand to compute the chemical potential. The four standard quantities are identified by the strings "number density", "pressure", "energy density", and "entropy density", which may also be set by the defined parameters S_NUMBER_DENSITY, S_PRESSURE, S_ENERGY_DENSITY, and S_ENTROPY_DENSITY.

3 Default Integrands

This section presents the default integrands used in libstatmech. They are for non-interacting, fully relativistic fermions and bosons. The key parameters are m, the particle rest mass, g, the particle multiplicity, T, the Temperature, $\hbar$, Planck's constant divided by 2π, c, the speed of light in vacuum, and k, Boltzmann's constant. We use GSL defined values of these constants (see the GSL documentation) in all cases. We choose the cgs system of units. The other parameters are α and γ. These are defined as follows:

$$\alpha = \frac{\mu - mc^2}{kT} \equiv \frac{\mu'}{kT},$$

where μ is the full chemical potential and μ' is the chemical potential less the particle's rest mass energy, and

$$\gamma = \frac{mc^2}{kT}.$$

3.1 Fermions

In deriving our integrands, we considered fermions and anti-fermions and assumed them to be in annihilation equilibrium with the photon field. Thus, for example, we assumed the reaction $e^+ + e^- \rightleftharpoons \gamma + \gamma$, where the γ represents a photon. Because the photon has zero chemical potential, the equilibrium implies

$$\mu_{e^+} = -\mu_{e^-}.$$

We may then write in terms of chemical potentials less the rest mass:

$$\mu'_{e^+} = -\mu'_{e^-} - 2mc^2,$$

where m is the electron rest mass. On division by kT, this then becomes

$$\alpha_{e^+} = -\alpha_{e^-} - 2\gamma.$$

The resulting integrands are the following:

Number Density:

$$n_{e^+e^-} = \frac{(mc^2)^3 g}{2\pi^2(\hbar c)^3\gamma^3} \int_0^\infty (x+\gamma)\sqrt{x^2 + 2\gamma x} \left[\frac{1}{1 + \exp(x - \alpha)} - \frac{1}{1 + \exp(x + 2\gamma + \alpha)} \right] dx$$

We note that in the special case that the fermion rest mass is zero, the default integrand for the number density may be integrated directly. The result is

$$n_{e^+e^-} = \frac{(kT)^3 g}{2\pi^2(\hbar c)^3} \left(\frac{\pi^3}{3}\alpha + \frac{\alpha^3}{3} \right)$$

As of version 0.5, libstatmech uses this result in default calculations of the number density and chemical potential when the fermion rest mass is zero.

Pressure:

$$P_{e^+e^-} = \frac{(mc^2)^4 g}{2\pi^2(\hbar c)^3\gamma^4}\int_0^\infty (x+\gamma)\sqrt{x^2+2\gamma x}\,\{\ln(1+\exp[\alpha-x])+\ln(1+\exp[-x-2\gamma-\alpha])\}\;dx$$

Energy Density:

$$\epsilon_{e^+e^-} = \frac{(mc^2)^4 g}{2\pi^2(\hbar c)^3\gamma^4}\int_0^\infty (x+\gamma)^2\sqrt{x^2+2\gamma x}\left[\frac{1}{1+\exp(x-\alpha)}+\frac{1}{1+\exp(x+2\gamma+\alpha)}\right]dx$$

Entropy Density:

$$s_{e^+e^-} = \frac{k(mc^2)^3 g}{2\pi^2(\hbar c)^3\gamma^3}\int_0^\infty (x+\gamma)\sqrt{x^2+2\gamma x}$$

$$\left[\frac{x-\alpha}{1+\exp(x-\alpha)}+\ln[1+\exp(\alpha-x)]+\ln[1+\exp(-x-2\gamma-\alpha)]+\frac{x+2\gamma+\alpha}{1+\exp(x+2\gamma+\alpha)}\right]dx$$

3.2 Bosons

In the absence of evidence for a conserved boson number, we neglected the anti-bosons. The resulting integrands are the following:

Number Density:

$$n = \frac{(kT)^3 g}{2\pi^2(\hbar c)^3}\int_0^\infty \frac{(x+\gamma)\sqrt{x^2+2\gamma x}}{\exp(x-\alpha)-1}\,dx$$

Pressure:

$$P = -\frac{(kT)^4 g}{2\pi^2(\hbar c)^3}\int_0^\infty (x+\gamma)\sqrt{x^2+2\gamma x}\ln[1-\exp(\alpha-x)]\,dx$$

Energy Density:

$$\epsilon = \frac{(kT)^4 g}{2\pi^2(\hbar c)^3}\int_0^\infty \frac{(x+\gamma)^2\sqrt{x^2+2\gamma x}}{\exp(x-\alpha)-1}\,dx$$

Entropy Density:

$$s = -\frac{(kT)^3 g}{2\pi^2(\hbar c)^3}\int_0^\infty (x+\gamma)\sqrt{x^2+2\gamma x}\left\{\ln[1-\exp(\alpha-x)]+\frac{\alpha-x}{\exp(x-\alpha)-1}\right\}dx$$

4 User-Supplied Functions and Integrands for Thermodynamic Quantities

If a user wishes to use a function or an intergrand other than the default version to compute a thermodynamic quantity, he or she must supply those. To supply a function for a thermodynamic quantity of a fermion, the user writes a routine with the prototype

```
double
fermion_function(
  Libstatmech__Fermion *p_fermion,
  double d_T,
  double d_mukT,
  void *p_user_data
);
```

Here, **p_fermion** is a pointer to a libstatmech fermion structure, **d_T** is the temperature in K, **d_mukT** is the μ'/kT, the chemical potential (less the rest mass, that is, $\mu' = \mu - mc^2$, where μ is the full chemical potential) divided by kT, where k is Boltzmann's constant, and **p_user_data** is a pointer to a user-defined structure carrying extra data into the routine. The user's routine need not be named **fermion_function**. Analogously, a user-supplied function for a boson thermodynamic quantity has the prototype

```
double
boson_function(
  Libstatmech__Boson *p_boson,
  double d_T,
  double d_mukT,
  void *p_user_data
);
```

For both the fermion and boson function, the user's routine must return the thermodynamic quantity for the input temperature, μ/kT, and other user data.

A user may also supply an integrand for a thermodynamic quantity. Here the respective prototypes are

```
double
fermion_integrand(
  Libstatmech__Fermion *p_fermion,
  double d_x,
  double d_T,
  double d_mukT,
  void *p_user_data
);
```

and

```
double
boson_integrand(
  Libstatmech__Boson *p_boson,
  double d_x,
  double d_T,
  double d_mukT,
  void *p_user_data
);
```

The input parameters are the same as for the functions. The additional parameter **d_x** is the integration variable.

Once the user has written an appropriate function and integrand, he or she updates them for the fermion or boson. For example, to update the pressure for a fermion **p_fermion** with function **my_pressure_function** and **my_pressure_integrand**, the user calls

```
Libstatmech__Fermion__updateQuantity(
  p_fermion,
  S_PRESSURE,
  (Libstatmech__Fermion__Function) my_pressure_function,
  (Libstatmech__Fermion__Integrand) my_pressure_integrand
);
```

Now when the user calls Libstatmech__Fermion__computeQuantity for **p_fermion** and for the pressure, libstatmech will compute the pressure by first evaluating my_pressure_function for the input temperature, chemical potential, and other data and will add to it the result of the numerical integration of the integrand my_pressure_integrand. If either the function or integrand is set to NULL, it will not be used in the calculation of the quantity. If the integrand is set to DEFAULT_INTEGRAND, the integrand will be reset to the default for that quantity.

The user may also define his or her own quantity (other than one of the four standard ones). To do so, the user writes the appropriate function and integrand and then sets the quantity for the fermion or boson with the updateQuantity() routine but with the string giving the quantity name set appropriately. The quantity is then computed by calling computeQuantity with that quantity name. For example, suppose we have a fermion pointed to by **p_fermion**. Now we write a Libstatmech__Fermion__Function my_enthalpy_function and a Libstatmech__Fermion__Integrand my_enthalpy_integrand that follow the appropriate prototypes. To compute the enthalpy at a temperature of 10^3 K and $\mu'/kT = -23$ with no extra data, we set the quantity enthalpy and then compute it:

```
Libstatmech__Fermion__updateQuantity(
  p_fermion,
  "enthalpy",
  (Libstatmech__Fermion__Function) my_enthalpy_function,
  (Libstatmech__Fermion__Integrand) my_enthalpy_integrand
);

fprintf(
  stdout,
  "The enthalpy density is %g (ergs/cc)\n",
  Libstatmech__Fermion__computeQuantity(
    p_fermion,
    "enthalpy",
    1000.,
    -23.,
    NULL
  );
```

Examples in the libstatmech distribution further demonstrate how to supply user-defined functions and integrands and how to apply them to thermodynamic quantities.

5 Numerical Calculation of the Integrals

The integration variable for quantity integrands is, in effect, the magnitude of the particle momentum converted to energy in units of kT; thus, the full range of integration is from zero to ∞. Users may reset the integral lower limit with the updateIntegralLowerLimit API routines. The integrals are computed with Gnu Scientific Library (GSL) routines. It is efficient, when the $\mu'/kT > 0$, to integrate from the lower limit to μ'/kT and then from μ'/kT to ∞. The former integral is done with the GSL routine **gsl_integration_qags** while the latter is done with **gsl_integration_qagiu**. When $\mu'/kT \leq 0$, the full integration is done with **gsl_integration_qagiu**.

Numerical integration of a quantity continues until the approximation to the integral satisfies the absolute tolerance ϵ_{abs} and relative tolerance ϵ_{rel}. By default these are both 10^{-8} but the user may update them with the updateQuantityIntegralAccuracy() API routines. The default values provide an excellent compromise between accuracy and speed, and most users should probably not need to change them. Users who need the routines to be fast (up to ~ 3 times faster than the default calculations) and who can sacrifice some accuracy may wish to increase the tolerance numbers (tolerances of 10^{-2} lead to results that are still typically accurate up to about four decimal places).

6 Inversion of the Number Density Integral

It is often the case that one knows the temperature and number density for a gas of fermions or bosons and seeks the chemical potential. To do this, one must invert the number density integral; that is, given T and n, the temperature and number density, one must find the μ'/kT such that the integral of the number density agrees with n. libstatmech does this by finding the root of the function

$$f = n_0 - n(T, \mu'/kT, X), \tag{1}$$

where n_0 is the given number density, n is the result of the integration [or, more generally, the function plus integral that gives the number density as computed from computeQuantity()], and X represents other parameters to the number density. The root is μ'/kT. To find this root, libstatmech uses the Brent solver in GSL. Iteration proceeds until the root achieves a relative tolerance of 10^{-12}. This tolerance is set by the parameter D_EPS_ROOT in Libstatmech.h. Our experience is that the chosen value works well.

7 Temperature Derivatives of Thermodynamic Quantities

Users may compute temperature derivatives of thermodynamic quantities with the computeTemperatureDerivatives() API routines. The temperature derivatives are computed with the GSL routine **gsl_deriv_central**. The step size for the differentiation is a fraction 0.001 of the temperature at which the derivative is desired. This fraction may be changed by setting D_STEP_FRACTION in Libstatmech.h to a different value.