tkfermi\_integral.py ライブラリのマニュアルを以下に示します。

# tkfermi\_integral ライブラリ マニュアル

## 1) このライブラリの主な機能や目的

tkfermi\_integral.py は、フェルミ・ディラック積分 (Fermi-Dirac integral) の高速かつロバストな計算を提供するPythonライブラリです。半導体物理学や統計物理学における電子・正孔密度、占有確率などの計算に特化しています。

主な機能は以下の通りです。

* **フェルミ・ディラック分布関数の計算**: 電子の占有確率や正孔の確率を数値安定的に計算します。対数スケールでの計算もサポートします。
* **フェルミ・ディラック積分の計算**: 任意の次数 のフェルミ・ディラック積分 を計算します。特に、 と については専用のラッパー関数が用意されています。
* **高速かつ高精度な計算**:
  + 極限領域（縮退極限、非縮退極限）では解析的な近似式を使用します。
  + 中間領域では scipy.integrate.quad を用いた数値積分を使用します。
  + functools.lru\_cache を活用し、一度計算された結果をキャッシュすることで、同じ引数での再計算を避けて性能を向上させています。
* **キャリア密度計算の支援**: 半導体における伝導帯の電子密度および価電子帯の正孔密度を、状態密度とフェルミ準位から計算するためのヘルパー関数を提供します。
* **数値安定性**: 指数関数のオーバーフロー・アンダーフローを防ぐためのクリッピングや、scipy.special.expit, log\_expit の利用により、広範囲の入力値に対して安定した計算を保証します。

## 2) このライブラリを他のプログラムからimportする方法

このライブラリは、Pythonの標準的な方法でインポートできます。tkfermi\_integral.py ファイルがPythonのパス上にあるか、現在の作業ディレクトリにあることを前提とします。

**モジュール全体をインポートする場合:**

import tkfermi\_integral  
  
# フェルミ・ディラック積分 F\_{1/2}(eta) を計算する例  
eta\_val = 5.0  
result = tkfermi\_integral.FermiIntegral\_half(eta\_val)  
print(f"F\_1/2({eta\_val}) = {result}")

**特定の関数や変数をインポートする場合:**

from tkfermi\_integral import FermiIntegral\_fast, electron\_density, xlim\_exp  
  
# フェルミ・ディラック積分 F\_r(eta) を計算する例  
eta\_val = 2.0  
r\_val = 1.5  
result = FermiIntegral\_fast(eta\_val, r\_val)  
print(f"F\_{r\_val}({eta\_val}) = {result}")  
  
# 電子密度を計算する例  
Nc\_val = 2.5e19  
eta\_c\_val = 3.0  
n\_density = electron\_density(Nc\_val, eta\_c\_val)  
print(f"Electron density = {n\_density}")  
  
# xlim\_exp を参照する例  
print(f"xlim\_exp value = {xlim\_exp}")

## 3) 必要な非標準ライブラリとインストール方法

このライブラリは、以下の非標準ライブラリに依存しています。

* **numpy**: 数値計算（特に配列操作）に使用されます。
* **scipy**: 数値積分 (scipy.integrate.quad) や特殊関数 (scipy.special.expit, scipy.special.log\_expit) に使用されます。

これらのライブラリは pip を使用してインストールできます。

pip install numpy scipy

## 4) importできる変数と関数

### インポートできる変数

* xlim\_exp: float
  + 指数関数を含む計算 (exp(x)) において、x の値が極端に大きくならないようにクリッピングする境界値です。これにより、数値オーバーフローやアンダーフローを防ぎ、数値安定性を向上させます。デフォルト値は 40.0 です。

### インポートできる関数

* fermi\_dirac\_e\_npvector(x)
  + **動作**: Fermi-Dirac分布関数 を計算します。numpy 配列を効率的に処理し、数値安定性を確保するために xlim\_exp でクリッピングを行います。eta や g は含まれず、単純な形の分布関数を計算します。
  + **引数**:
    - x (numpy.ndarray): エネルギーレベル の配列。
  + **戻り値**:
    - numpy.ndarray: x に対応するFermi-Dirac分布関数の値の配列。
* fermi\_dirac\_e\_if(x, eta, g=1.0)
  + **動作**: Fermi-Dirac分布関数 を計算します。単一のスカラ値に対して動作し、if 文を用いたクリッピングで数値安定性を確保します。
  + **引数**:
    - x (float): エネルギーレベル 。
    - eta (float): フェルミ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子（スピンの縮退度など）。
  + **戻り値**:
    - float: Fermi-Dirac分布関数の値。
* fermi\_dirac\_e(x, eta, g=1.0)
  + **動作**: Fermi-Dirac分布関数 を計算します。scipy.special.expit (シグモイド関数) を使用して数値安定性を高めています。numpy 配列とスカラのどちらにも対応します。
  + **引数**:
    - x (float または numpy.ndarray): エネルギーレベル 。
    - eta (float または numpy.ndarray): フェルミ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float または numpy.ndarray: Fermi-Dirac分布関数の値。
* log\_fermi\_dirac\_e(x, eta, g=1.0)
  + **動作**: Fermi-Dirac分布関数 の対数 を数値安定的に計算します。scipy.special.log\_expit を使用します。
  + **引数**:
    - x (float または numpy.ndarray): エネルギーレベル 。
    - eta (float または numpy.ndarray): フェルミ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float または numpy.ndarray: Fermi-Dirac分布関数の対数値。
* fermi\_dirac\_h(x, eta, g=1.0)
  + **動作**: 正孔 (vacancy/hole) の確率 を計算します。fermi\_dirac\_e と同様に expit を使用して数値安定性を確保します。
  + **引数**:
    - x (float または numpy.ndarray): エネルギーレベル 。
    - eta (float または numpy.ndarray): フェルミ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float または numpy.ndarray: 正孔の確率。
* log\_fermi\_dirac\_h(x, eta, g=1.0)
  + **動作**: 正孔の確率 の対数 を数値安定的に計算します。
  + **引数**:
    - x (float または numpy.ndarray): エネルギーレベル 。
    - eta (float または numpy.ndarray): フェルミ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float または numpy.ndarray: 正孔の確率の対数値。
* ionized\_acceptor\_frac(eta, xA, g=1.0)
  + **動作**: アクセプタがイオン化している割合 (ionized acceptor fraction) を計算します。Fermi-Dirac分布関数 fermi\_dirac\_e と同じ定義を使用します。
  + **引数**:
    - eta (float): フェルミ準位 。
    - xA (float): アクセプタ準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float: イオン化アクセプタの割合。
* ionized\_donor\_frac(eta, xD, g=1.0)
  + **動作**: ドナーがイオン化している割合 (ionized donor fraction) を計算します。Fermi-Dirac分布関数 fermi\_dirac\_h (正孔の確率) と同じ定義を使用します。
  + **引数**:
    - eta (float): フェルミ準位 。
    - xD (float): ドナー準位 。
    - g (float, オプション, デフォルトは1.0): 分布関数の前因子。
  + **戻り値**:
    - float: イオン化ドナーの割合。
* FermiIntegral\_fast(eta, r, \*, epsabs=1.0e-10, epsrel=1.0e-8, limit=50)
  + **動作**: 一般的な次数 のフェルミ・ディラック積分 を計算します。縮退極限、非縮退極限の解析的近似と、中間領域での数値積分 (scipy.integrate.quad) を組み合わせて、高速かつ高精度な結果を提供します。内部で lru\_cache を使用し、同じ引数での再計算を避けます。
  + **引数**:
    - eta (float): フェルミ準位 。
    - r (float): 積分次数。
    - epsabs (float, キーワード引数, オプション): scipy.integrate.quad に渡される絶対誤差許容値。
    - epsrel (float, キーワード引数, オプション): scipy.integrate.quad に渡される相対誤差許容値。
    - limit (int, キーワード引数, オプション): scipy.integrate.quad に渡される最大積分区間数。
  + **戻り値**:
    - float: フェルミ・ディラック積分の値 。
* FermiIntegral\_half(eta, \*, epsabs=1.0e-10, epsrel=1.0e-8, limit=50)
  + **動作**: 次数 のフェルミ・ディラック積分 を計算します。FermiIntegral\_fast のラッパー関数です。
  + **引数**:
    - eta (float): フェルミ準位 。
    - epsabs, epsrel, limit は FermiIntegral\_fast と同様。
  + **戻り値**:
    - float: フェルミ・ディラック積分 の値。
* FermiIntegral\_3half(eta, \*, epsabs=1.0e-10, epsrel=1.0e-8, limit=50)
  + **動作**: 次数 のフェルミ・ディラック積分 を計算します。FermiIntegral\_fast のラッパー関数です。
  + **引数**:
    - eta (float): フェルミ準位 。
    - epsabs, epsrel, limit は FermiIntegral\_fast と同様。
  + **戻り値**:
    - float: フェルミ・ディラック積分 の値。
* electron\_density(Nc, eta\_c, \*, epsabs=1.0e-10, epsrel=1.0e-8, limit=50)
  + **動作**: 伝導帯の電子密度 を計算します。標準的な放物線バンドモデルを使用し、次の式に基づきます: 。
  + **引数**:
    - Nc (float): 伝導帯の実効状態密度。単位は戻り値に引き継がれます (例: または )。
    - eta\_c (float または numpy.ndarray): 無次元化された電子の化学ポテンシャル 。
    - epsabs, epsrel, limit は FermiIntegral\_half に渡されます。
  + **戻り値**:
    - float または numpy.ndarray: 電子密度 。Nc と同じ単位。
* hole\_density(Nv, eta\_v, \*, epsabs=1.0e-10, epsrel=1.0e-8, limit=50)
  + **動作**: 価電子帯の正孔密度 を計算します。標準的な放物線バンドモデルを使用し、次の式に基づきます: 。ここで、 です。
  + **引数**:
    - Nv (float): 価電子帯の実効状態密度。単位は戻り値に引き継がれます (例: または )。
    - eta\_v (float または numpy.ndarray): 無次元化された正孔の化学ポテンシャル 。
    - epsabs, epsrel, limit は FermiIntegral\_half に渡されます。
  + **戻り値**:
    - float または numpy.ndarray: 正孔密度 。Nv と同じ単位。

## 5) main scriptとして実行したときの動作

tkfermi\_integral.py を直接実行 (python tkfermi\_integral.py) した場合、自己テストが実行されます。このテストは、ライブラリの基本的な機能と使い方を示すデモンストレーションとして機能します。

具体的な動作は以下の通りです。

1. **定数の定義**: 伝導帯と価電子帯の実効状態密度 Nc と Nv を 1.0e19 に設定し、eta\_list として を定義します。
2. **の計算**: eta\_list の各値に対して、次数 のフェルミ・ディラック積分 (FermiIntegral\_half) を計算し、結果を整形して表示します。
3. **、電子密度 、正孔密度 の計算**: eta\_list の各値に対して、次数 のフェルミ・ディラック積分 (FermiIntegral\_3half)、電子密度 (electron\_density)、および正孔密度 (hole\_density) を計算し、それぞれの結果を整形して表示します。
4. **キャッシュ情報の表示**: \_FermiIntegral\_cached 関数が保持しているキャッシュの使用状況 (cache\_info()) を表示します。これにより、ライブラリのキャッシュ機能がどのように機能しているかを確認できます。

この出力は、異なるフェルミ準位 () の条件下で、フェルミ・ディラック積分とそれから導かれるキャリア密度がどのように変化するかを示すとともに、ライブラリが効率的に動作していることを示唆します。