# Ne-T\_fit.py マニュアル

## 1. プログラムの目的

Ne-T\_fit.py は、半導体のキャリア統計（フェルミ準位 Ef、電子濃度 n=Ne、正孔濃度 p、ドナー/アクセプターのイオン化率）を、フェルミ・ディラック積分（オーダー 1/2）を用いて計算し、さらに実験データ Ne(T) に対してパラメータフィッティングを行うプログラムです。

主な用途は以下です。

* **計算（sim）**：与えた物理パラメータ（Eg, ED, EA, ND, NA, me, mh）から Ne(T) を計算しプロットする
* **読み込み表示（read）**：入力データ（Ne(T)）を表示・プロットする
* **フィッティング（fit）**：観測データ Ne(T) から、ED, ND, EA, NA, Eg を最適化して推定する
* **誤差解析（fit 内）**：ヤコビアンによる線形化近似で共分散・相関・固有値/固有ベクトルを出力し、さらに誤差伝播で Ne(T) の推定誤差帯（薄色）をプロットする
* **固定パラメータ提案（fit 内）**：大きい誤差・強相関・不確か方向への寄与を基準に、固定（または外部拘束）候補を推定する

## 2. 原理

### 2.1 半導体キャリア統計（基本式）

温度 T における伝導帯端 Ec、価電子帯端 Ev のもとで、電子・正孔密度はフェルミ・ディラック積分（オーダー 1/2）で計算します。

有効状態密度（cm^-3）は、

$$
N\_c(T) = 2\left(\frac{m\_e^\\* m\_0 kT}{2\pi\hbar^2}\right)^{3/2}\times 10^{-6},
\qquad
N\_v(T) = 2\left(\frac{m\_h^\\* m\_0 kT}{2\pi\hbar^2}\right)^{3/2}\times 10^{-6}
$$

ここで は電子質量、$m\_e^\\*, m\_h^\\*$ は有効質量比です。

フェルミ準位 Ef に対して、電子濃度 Ne と正孔濃度 Nh は

を用います（ は eV 単位に変換して使用）。

本プログラムでは、積分の安定性のため

* を expit で評価
* を数値積分 quad で計算（0〜500 を既定）

としています。

### 2.2 不純物のイオン化（ドナー ND+, アクセプター NA-）

ドナー準位を とし、ドナーの中性占有率は

よってイオン化ドナーは

アクセプター準位を とし、アクセプターのイオン化率（占有率）は

よってイオン化アクセプターは

ここでも expit を用いて数値的安定性を確保します。

### 2.3 電荷中性条件と Ef の決定

フェルミ準位 Ef は電荷中性条件

を満たす解として求めます。  
本プログラムでは brentq による 1 次元根探索を使用し、解が見つからない場合は探索区間を自動拡張します。

### 2.4 フィッティング（mode=fit）

観測データ に対し、モデル の **log10 残差**を定義します。

目的関数（最小化）は

です。最適化には scipy.optimize.minimize を用い、既定は Nelder–Mead です。

また ND, NA は桁が大きい量のため、最適化では内部的に

* ND, NA を **log10 空間**で最適化
* ED, EA, Eg は線形空間で最適化

とします。

### 2.5 共分散行列・相関行列（線形化近似）

最適解 近傍で残差ベクトルを線形化し、ヤコビアン

を数値微分で求めます。  
このとき、内部パラメータの共分散は

（: フィットに使った点数、: 自由パラメータ数）で近似します。

相関係数行列は

で計算します。

### 2.6 固有値・固有ベクトル解析

* Cov の固有分解：不確かさ（分散）が大きい方向の線形結合を抽出
* の固有分解：小さい固有値の方向は「データが決めにくい方向」

ここで

* Cov の大きい固有値 → 不確かさが大きい主方向
* の小さい固有値 → 曲率が小さい＝目的関数があまり変わらない＝決まらない方向

となります（Cov は に比例するため）。

### 2.7 誤差伝播による Ne(T) 推定誤差帯

モデル出力

の微分ベクトル を数値微分で求め、

より を得ます。  
これを用いて

を計算し、プロットでは薄色帯で表示します（--band\_sigma で倍率指定）。

### 2.8 固定すべきパラメータの推定（ヒューリスティック）

以下の要素をスコア化して候補を提案します。

* 相対誤差（stderr / |estimate|）が大きい
* 相関 |corr| が閾値以上（例 0.95）
* Cov の「最大不確か方向」の固有ベクトルに強く寄与

これは数値的な「決まりにくさ」の指標であり、最終判断は物理的妥当性・独立測定の有無で行ってください。

## 3. 使い方 (usage)

### 3.1 共通

* 入力ファイル：CSV または Excel
* 温度列・Ne 列は 0 開始の列番号で指定
* Excel の場合は --sheet でシート名または番号（例 “0”）を指定可能

主なモード：

* --mode read : データを表示・プロット
* --mode sim : 与えたパラメータで Ne(T) を計算・プロット（入力があれば重ね描き）
* --mode fit : データに対してパラメータフィット＋誤差解析＋誤差帯プロット

### 3.2 主なオプション

I/O： - --input FILE : 入力データ（CSV/Excel） - --sheet SHEET : Excel シート名/番号 - --temp\_col N : 温度列（0開始） - --ne\_col N : Ne列（0開始）

物理パラメータ（初期値/固定値）： - --Eg VALUE : バンドギャップ Eg [eV] - --me VALUE : 電子有効質量比 - --mh VALUE : 正孔有効質量比 - --ND VALUE : ドナー濃度 ND [cm^-3] - --NA VALUE : アクセプター濃度 NA [cm^-3] - --ED VALUE : ドナー準位深さ ED [eV] - --EA VALUE : アクセプター準位深さ EA [eV]

フィット設定： - --method METHOD : 最適化法（既定 nelder-mead） - --fix LIST : 固定パラメータ（例 EA,NA,Eg）。対象は ED,ND,EA,NA,Eg - --Tfitmin VALUE : フィットに使う最低温度[K]（既定 -1e100） - --Tfitmax VALUE : フィットに使う最高温度[K]（既定 +1e100）

誤差帯・ヤコビアン設定： - --band\_sigma VALUE : 誤差帯幅（既定 1.0＝±1σ） - --jac\_relstep VALUE : 数値微分 相対ステップ（既定 1e-6） - --jac\_absstep VALUE : 数値微分 絶対ステップ（既定 1e-12）

出力： - --save FILE : fit結果を保存する json（既定 fit\_params.json） - --load FILE : 初期値を json から読込 - --diag\_save FILE : 共分散/相関/固有値などの診断 json（既定 fit\_diagnostics\_opt.json） - --suggest\_save FILE : 固定候補提案 json（既定 fit\_fix\_suggestions.json）

## 4. 使用例

### 4.1 データ表示（read）

例：Excel の 1列目が温度、4列目が Ne のとき（0開始で temp\_col=0, ne\_col=3）

python Ne-T\_fit.py –input cIGZO-1e16.xlsx –ne\_col 3 –mode read –temp\_col 0

### 4.2 計算（sim）

例：入力データの温度点でモデル計算し、データに重ね描き

python Ne-T\_fit.py –input cIGZO-1e16.xlsx –ne\_col 3 –mode sim –Eg 3.5 –ND 1e17 –NA 0 –me 0.32 –ED 0.1

入力が無い場合は Tmin〜Tmax の等間隔点で計算します。

python Ne-T\_fit.py –mode sim –Tmin 100 –Tmax 600 –nT 80 –Eg 3.5 –ND 1e17 –NA 0 –me 0.32 –ED 0.1

### 4.3 フィッティング（fit）

例：Eg, EA, NA を固定して n型側のみフィット（ED と ND を推定）

python Ne-T\_fit.py –input cIGZO-1e16.xlsx –ne\_col 3 –mode fit –Eg 3.5 –EA 0.22 –NA 0 –me 0.32 –ED 0.1 –fix Eg,EA,NA

例：特定の温度範囲だけでフィット（T=200〜500 K）

python Ne-T\_fit.py –input cIGZO-1e16.xlsx –ne\_col 3 –mode fit –Eg 3.5 –NA 0 –me 0.32 –ED 0.1 –Tfitmin 200 –Tfitmax 500 –fix Eg,EA,NA

例：誤差帯（±1σ）を安定化したい場合（誤差帯計算側の有限差分を大きめに）

python Ne-T\_fit.py –input cIGZO-1e16.xlsx –ne\_col 3 –mode fit –Eg 3.5 –NA 0 –me 0.32 –ED 0.1 –fix Eg,EA,NA –band\_sigma 1.0 –jac\_relstep 1e-4 –jac\_absstep 1e-8

### 4.4 出力ファイルの確認

* fit結果：fit\_params.json（--save で変更可）
* 診断（共分散・相関・固有値等）：fit\_diagnostics\_opt.json（--diag\_save）
* 固定候補提案：fit\_fix\_suggestions.json（--suggest\_save）

これらは後処理（表作成、報告資料、再フィットの初期値）に利用できます。

## 5. 重要な注意（数値安定性）

* F\_{1/2} 積分の高速化のため、既定では eta を 1e-3 に量子化してキャッシュします。
* しかし、数値ヤコビアン計算ではこの量子化により残差が段階的になり、J≈0 となることがあります。
* 本プログラムは、ヤコビアン計算時のみ FD キャッシュ量子化を無効化（FD\_CACHE\_DISABLE=True）して回避します。
* さらに誤差帯（prediction band）は、brentq や quad の数値ノイズの影響を受けやすいので、必要に応じて --jac\_relstep 1e-4 --jac\_absstep 1e-8 など大きめの差分を推奨します。