

# 技術書邦訳の手引き

英語マニュアルを日本語の技術書へ再構成するために

粉体 粒子

2026 年 5 月 10 日

## 概要

本書は、英文で書かれた技術マニュアルを、単なる訳文でなく、日本語の技術書として利用できる形へ再構成するための実務手引きである。技術書の邦訳では、英語を日本語に置き換えるだけでは足りない。本書では、原文の論理を保ちながら日本語の技術書として読みやすく整えるためのテクニックと、訳語、コード、数式、図表、組版で失敗しやすい要所を解説する。したがって、訳語の統一、章立ての調整、コードと本文の対応、組版、PDF・EPUB の検査までを一つの作業系として扱う。

## 目次

1	<b>はじめに</b>	1
1.1	本書の目的	1
2	<b>技術書邦訳の基本原則</b>	1
2.1	翻訳ではなく再構成である	1
2.2	読者の作業から逆算して書く	1
2.3	理由を書く	2
2.4	訳語を固定する	2
3	<b>全体の作業順序</b>	2
3.1	基本フロー	2
3.2	章単位で進める	3
4	<b>原文調査</b>	3
4.1	文書構造の把握	3
4.2	翻訳対象と非翻訳対象	3
5	<b>文体と訳文の作り方</b>	4
5.1	基本は「である」調	4
5.2	英語の構造を直訳しない	4
5.3	助動詞を訳し分ける	4
6	<b>訳語表と表記規約</b>	5
6.1	訳語表の作成	5

6.2	表記規約を先に決める . . . . .	5
7	コード, API, 数式の扱い . . . . .	5
7.1	コードは翻訳しない . . . . .	5
7.2	長いコード行 . . . . .	6
7.3	数式まわりの訳し方 . . . . .	6
8	図表, 索引, 表紙 . . . . .	6
8.1	図表 . . . . .	6
8.2	索引と用語表 . . . . .	6
8.3	表紙 . . . . .	6
9	LuaLaTeX と組版 . . . . .	7
9.1	LuaLaTeX を使う理由 . . . . .	7
9.2	最小の前付け . . . . .	7
9.3	PDF の確認 . . . . .	7
10	レビュー . . . . .	7
10.1	翻訳レビュー . . . . .	7
10.2	組版レビュー . . . . .	8
11	最終チェックリスト . . . . .	8
12	結言 . . . . .	9

## 1 はじめに

### 1.1 本書の目的

本書の目的は、英文技術マニュアルを邦訳するときの判断基準と作業順序を明確にすることである。対象は、ソフトウェアの利用説明書、API リファレンス、解析手順書、チュートリアル、研究・教育用の技術文書である。これらの文書では、読者が本文を読んだ後に実際の作業を再現できることが最も重要である。

技術書邦訳において避けるべきなのは、原文の単語を逐語的に置き換えただけの文書である。見かけ上は日本語になっていても、操作手順、条件、例外、数式中の記号、コード中の識別子との対応が崩れていれば、読者は動けない。邦訳とは、原文の情報を日本語の読者が運用できる形へ移植する作業に他ならない。

## 2 技術書邦訳の基本原則

### 2.1 翻訳ではなく再構成である

技術書の邦訳は、原文の構造をそのまま写す作業ではない。原文の章立てを尊重しつつ、日本語の読者が迷わない順序へ再構成する必要がある。とりわけ英語では一文に収まっている条件、理由、例外、操作が、日本語では複数文に分けた方が明瞭になることが多い。

ただし、再構成とは内容を勝手に増減することではない。原文の条件、必須事項、推奨、制約、例外は必ず残

す。削ってよいのは英語特有の冗長な構文であり、削ってはならないのは技術的判断に関わる情報である。

## 2.2 読者の作業から逆算して書く

邦訳文は、読むためだけでなく使うための文書である。したがって、次の問いから逆算して文章を整える。

1. 読者はこの節を読んだ後、何を実行するのか。
2. どのファイル、コマンド、画面、設定値を確認するのか。
3. 正常終了を何で判断するのか。
4. 失敗した場合、どの条件を疑うのか。

この四点が曖昧な段落は、たとえ日本語として流暢でも技術書としては弱い。必要に応じて、段落の末尾に確認項目を加えるとよい。

## 2.3 理由を書く

手順だけを並べると、読者は条件が少し変わっただけで応用できなくなる。したがって、「なぜその操作が必要か」を本文中に書く。たとえば、「絶対パスを指定する」とだけ書くのではなく、「バッチ実行時にはカレントディレクトリが変わることがあるため、絶対パスを指定する」と書く。これにより、読者は環境が変わったときにも判断できる。

## 2.4 訳語を固定する

同一概念に複数の訳語を当てると、読者は別概念だと誤解する。とくに技術書では、`body`、`interaction`、`contact`、`engine`、`scene` のような語がソフトウェア固有の概念を表すことが多い。翻訳開始前に訳語表を作り、章をまたいでも同じ判断で訳す。

# 3 全体の作業順序

## 3.1 基本フロー

技術書邦訳は、おおむね次の順序で進める。

1. 原文の構造を調査する。
2. 翻訳対象と非翻訳対象を分ける。
3. 訳語表と表記規約を作る。
4. 章単位で邦訳する。
5. コード、API 名、数式、図表、リンクを保護しながら本文を整える。
6. 用語の揺れ、英文残り、参照切れを検査する。
7. PDF と EPUB を生成する。
8. マージン、フォント、図表、索引、表紙を確認する。
9. 提出用ファイルだけを整理する。

この順序は一方方向ではない。PDF を組んで初めて長いコード行のはみ出しが見つかることがある。EPUB を開いて初めて画像の縮尺や目次の不備が分かることもある。したがって、翻訳、組版、検査、修正を反復する前

提で計画を立てる。

## 3.2 章単位で進める

大部のマニュアルを一括で訳すと、用語の揺れや参照切れを見落としやすい。そこで、章単位で次を確認しながら進める。

- 章の目的が冒頭で分かるか。
- 操作手順が実行順に並んでいるか。
- API 名やファイル名を訳していないか。
- 数式の変数説明が原文と対応しているか。
- 図表のキャプションが本文と矛盾していないか。

章ごとに品質をそろえてから全体を通読すると、後戻りが少ない。

## 4 原文調査

### 4.1 文書構造の把握

最初に、原文がどのような仕組みで作られているかを調べる。Sphinx, Markdown, LaTeX, HTML, 独自 XML など、形式によって修正すべき場所と触ってはならない場所が異なる。

確認すべき項目は次の通りである。

- 目次と章立て。
- 内部リンクと外部リンク。
- API リファレンスの自動生成部分。
- 図表と画像ファイルの置き場所。
- 数式の記法。
- コードブロックの言語。
- 索引, 用語集, 参考文献。
- ライセンス表記, 奥付, 謝辞。

自動生成された API リファレンスは、本文解説とは扱いが異なる。クラス名, 関数名, 属性名を翻訳してはならない。説明文だけを邦訳し、コードとして参照される名前は原文どおり残す。

### 4.2 翻訳対象と非翻訳対象

次の要素は、原則として翻訳しない。

- クラス名, 関数名, メソッド名, 属性名。
- コマンド名, オプション名, 環境変数名。
- ファイル名, ディレクトリー名, URL。
- コードブロックの実行部分。
- 参考文献の書誌情報。

- 公式名称，製品名，プロジェクト名。

ただし，本文中で概念として導入する場合は日本語を併記する。たとえば `Body` はコード上のクラス名としては `Body` のまま残し，本文では「ボディ (Body)」のように導入する。この一手間により，読者はコード名と訳語の対応を失わずに済む。

## 5 文体と訳文の作り方

### 5.1 基本は「である」調

技術書の本文は，原則として「である」調で統一する。説明が簡潔になり，章をまたいでも文体が安定するためである。操作手順では「実行する」「確認する」「保存しておく」のような簡潔な文を用いる。必要な箇所ですべて「してください」を使ってもよいが，本文全体で「ですます調」と「である調」が無秩序に混ざる状態は避ける。

### 5.2 英語の構造を直訳しない

英語の技術文は，主語，動詞，目的語，修飾句を順に積み重ねる。日本語で同じ順序を保つと，長く読みにくい文になりやすい。意味単位で分け，読者にとって自然な主語へ置き換える。

避けたい訳	改善例
互いに接触している粒子は力を及ぼす。	接触している粒子対ごとに接触力を計算する。
ユーザーは形状を定義する。	まず，解析対象の形状を設定する。
これは接触力を計算するために使われる。	この接触モデルは，粒子間の法線力と接線力を計算するために用いられる。

英語の `it`，`this`，`they` は，日本語では具体名に戻す。指示語をそのまま「これ」「それ」と訳すと，何を指すのか曖昧になる。

### 5.3 助動詞を訳し分ける

英語の助動詞は，技術上の強さが異なる。一律に訳してはならない。

英語	訳し方
<code>can</code>	機能として可能である。通常は「できる」と訳す。
<code>may</code>	可能性がある，または任意でもよい。
<code>should</code>	推奨，標準的手順，望ましい設定を表す。
<code>must</code>	条件を満たさないと成立しない必須事項である。
<code>required</code>	必要である。
<code>optional</code>	任意である。

`should` をすべて「すべきである」と訳すと，日本語では過度に命令的になる場合がある。実務的には「通常は」「推奨される」「設定するのが望ましい」を使い分ける。

## 6 訳語表と表記規約

### 6.1 訳語表の作成

翻訳開始前に訳語表を作る。訳語表は固定的な辞書ではなく、文脈に応じた訳し分けの規則も含める。

英語	基本訳	注意
particle	粒子	DEM などの粒子法では標準的な訳語である。
body	ボディ	ソフトウェア内部のオブジェクトを指す場合は「物体」より安全である。
interaction	相互作用	接触に限定される場合は「接触相互作用」とする。
contact	接触	interaction と混同しない。
scene	シーン	解析対象全体または状態全体を指す。
engine	エンジン	処理モジュールであることを初出で補う。
time step	時間ステップ	dt などの記号と対応させる。
stress	応力	変数説明では省略しない。
strain	ひずみ	表記を統一する。
overlap	オーバーラップ	必要に応じて「重なり量」を併記する。

### 6.2 表記規約を先に決める

長い技術書では、記号の意味が一定である方が読者の負担が小さい。次のような規約を最初に決めておく。

- File > Open: プルダウンメニュー。
- ReuseInstance=true: 設定ファイル中の文字列。
- [実行] : ボタン。
- hoge.ins: ファイル名。
- Ctrl + E: キーボード操作。

この規約を作らずに作業を始めると、章ごとに表記が揺れる。後から直すのは意外に手間がかかるので、早い段階で決めておく方がよい。

## 7 コード、API、数式の扱い

### 7.1 コードは翻訳しない

コード中の識別子は翻訳しない。読者は本文とコードを往復しながら読むため、本文の説明とコードの名前が対応していなければならない。

```
0.bodies.append(sphere((0,0,0), radius=1))
```

このコードは、本文では「0.bodies.append() はシーンにボディを追加する」と説明する。メソッド名を日本語に置き換えたり、コードに見える形で訳語を混ぜたりしてはならない。

## 7.2 長いコード行

印刷用 PDF では、長いコード行がマージンを越えやすい。辞書、関数呼び出し、長い URL は、コピー可能性を保ちながら複数行に分ける。

```
plot.plots = {
    'i': (('t', 'xr:'),),
    't': (('z_sph', 'r:'), None, ('v_sph', 'g--'))
}
```

行分割によって意味が変わらないことを確認する。Python, C++, シェルコマンドでは、改行位置によって構文が変わる場合があるため注意する。

## 7.3 数式まわりの訳し方

数式の前後では、変数、添字、単位、条件を原文と対応させる。変数の説明を省略すると、読者は式を実装や図と結び付けられない。

ここで、 $F_n$  は法線方向の接触力、 $k_n$  は法線ばね定数、 $u_n$  は法線方向のオーバーラップである。

数式自体を意識してはならない。式の意味を日本語で補うことと、式を別物に変えることは違う。

## 8 図表、索引、表紙

### 8.1 図表

図表は、原文のキャプションを訳すだけでは不十分である。読者が図を見て何を確認すべきかを短く補う。図が小さすぎると印刷 PDF では意味がなく、逆に大きすぎると余白不足やページ数増加につながる。図表は本文中の説明と対応していなければならない。

### 8.2 索引と用語表

技術書では、索引は読者が再確認するための道具である。API 索引だけでなく、英語用語索引、日本語用語索引、専門用語対照表を用意するとよい。

- クラス索引: クラス名、関数名、モジュール名など。
- 英語用語索引: 原文の専門用語を探すための索引。
- 日本語用語索引: 訳語から本文へ戻るための索引。
- 専門用語対照表: 英語、日本語、訳し分けの注記をまとめる表。

### 8.3 表紙

印刷用表紙では、表表紙、背表紙、裏表紙、裁ち落としを 1 枚の PDF として作る。背幅はページ数と用紙種別によって変わる。本文のページ数が変わったら、表紙も必ず再生成する。EPUB 用表紙は、印刷用の表表紙部分と一致させると、読者が同じ本として認識しやすい。

## 9 LuaLaTeX と組版

### 9.1 LuaLaTeX を使う理由

日本語を含む技術書では、LuaLaTeX を使うとフォント、数式、日本語組版、PDF 生成を一貫して扱いやすい。Sphinx などから LaTeX を生成する場合でも、最終的な PDF は LuaLaTeX で複数回コンパイルし、目次、索引、相互参照を解決する。

```
lualatex -interaction=nonstopmode -halt-on-error manual.tex
lualatex -interaction=nonstopmode -halt-on-error manual.tex
```

### 9.2 最小の前付け

新規文書を作る場合は、次の前付けを出発点にするとよい。装飾を凝り過ぎず、まず本文が安定して読めることを優先する。

```
\documentclass[a4paper,10pt]{ltjsarticle}
\usepackage[margin=24mm]{geometry}
\usepackage[haranoaji]{luatexja-preset}
\usepackage{enumitem}
\usepackage{fvextra}
\usepackage{xurl}
\usepackage[colorlinks=true,
  linkcolor=blue!45!black,
  urlcolor=blue!45!black,
  citecolor=blue!45!black]{hyperref}
\setlength{\parindent}{1em}
\setlength{\emergencystretch}{3em}
```

### 9.3 PDF の確認

PDF では、見た目だけでなく機械的な条件も確認する。

- ページサイズが提出先の指定と一致しているか。
- 長い URL、コード行、表、数式が余白からはみ出していないか。
- フォントがすべて埋め込まれているか。
- 目次、索引、内部リンクが機能しているか。

判型は `pdftinfo`、フォントは `pdf fonts` で確認できる。さらに `pdftoppm` でページ画像を生成し、ページとして読めるかを目視する。

## 10 レビュー

### 10.1 翻訳レビュー

翻訳レビューでは、原文と邦訳を一文ずつ対応させる。ただし、語順や文の分割は日本語として自然になるように変えてよい。重要なのは、技術的意味を変えないことである。

確認する観点は次の通りである。

- 原文の条件，例外，推奨，必須が残っているか。
- because, therefore, however などの論理関係が保たれているか。
- 指示語の対象が明確か。
- 訳語が章をまたいで統一されているか。
- コードと本文の対応が崩れていないか。
- 数式の記号説明が抜けていないか。

## 10.2 組版レビュー

組版レビューでは，文章としての正しさだけでなく，読めるページになっているかを見る。PDF を実際に画像として確認し，次を調べる。

- 文字が枠や余白からはみ出していない。
- 図が小さすぎない。
- 表がページ幅を越えていない。
- リンク色が読みやすい。
- 索引が貧弱すぎない。
- 章見出し，ページ番号，装飾が不自然でない。

## 11 最終チェックリスト

提出前に，少なくとも次を確認する。

1. 文書の対象読者と目的が冒頭で明示されている。
2. 章立てが原文の論理構造と対応している。
3. 説明文として残った英文がない。
4. API 名，クラス名，関数名，コマンド名は原文どおりである。
5. 訳語表に従って用語が統一されている。
6. 数式，変数，単位，添字が原文と対応している。
7. コードブロックが壊れていない。
8. 図表，キャプション，本文が対応している。
9. PDF の判型が提出先の指定と一致している。
10. 全ページが安全マージン内に収まっている。
11. フォントがすべて埋め込まれている。
12. EPUB の表紙，目次，リンク，画像が機能している。
13. 印刷用表紙の背幅が本文ページ数と一致している。
14. ライセンス表記，奥付，謝辞が必要に応じて入っている。

## 12 結言

技術書邦訳の要点は、原文の表層を日本語へ移すことではなく、原文が提供していた作業系を日本語読者の環境で再現できるようにすることである。用語、コード、数式、図表、索引、表紙、配布ファイルは互いに独立していない。どこか一つの対応が崩れれば、読者は同じ操作や理解にたどり着けなくなる。

したがって、邦訳作業では、文章、組版、検査を分けて考えつつ、最後には一つの出版物として統合する必要がある。この姿勢を守れば、翻訳文は単なる補助資料でなく、日本語の技術書として長く利用できるものになる。