

GetFEM++ ユーザドキュメントの地域化 L10N 日本語翻訳作業

@tkoyama010^{1†}

¹GetFEM++ Japanese Team

Localization(L10N) of GetFEM++ User Document Translation for Japanese

@tkoyama010^{*†}

^{*}GetFEM++ Japanese Team

Abstract

The GetFEM++ project focuses on the development of a generic and efficient C++ library for finite element methods elementary computations. The document of this library are generated by Sphinx and are available for modification and reuse under the terms of the GNU Free Documentation License . So, we translated this document to Japanese by using Transifex and distributed it in TechBookFest5.

Keywords: GetFEM++, L10N, Sphinx, Transifex, translate-shell

1. はじめに

GetFEM++ は汎用的で効率的な有限要素法の C++ ライブラリです。有限要素法を用いて線形および非線形偏微分方程式を解くためのフレームワークを提供することを目指しています。GetFEM++ ライブラリには多くの文書があり、その多くはさまざまな著者によって寄稿されています。GetFEM++ の文書化に使用されるマークアップは docutils プロジェクトによって開発された reStructuredText で、Sphinx というツールセットを使用しています。また、ドキュメントは GNU Free Documentation License により改変および頒布が許可されています。

そこで、Sphinx の国際化機能を使用して GetFEM++ のドキュメント翻訳作業を翻訳プラットフォーム Transifex で行いました。さらに、技術同人誌即売会である技術書典 5 で翻訳したドキュメントの頒布を行いました。本報告では以上の作業内容について報告をします。なお作業環境は Ubuntu18.10 です。

2. GetFEM++ プロジェクト

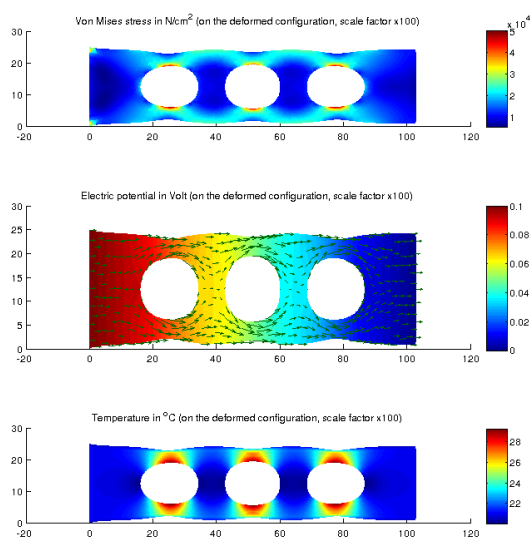
GetFEM++ の英語の Wikipedia を翻訳し、日本語の Wikipedia を作成しました。以下に内容を引用します。

GetFEM++ は Python, Matlab そして Scilab のインタフェースを使用可能な C++ の汎用ライブラリである。このライブラリの目的は有限要素法を用いて線形および非線形偏微分方程式を解くためのフレームワークを提供することである。有限要素近似や数値積分法の選択の柔軟性が特徴の一つである。「GetFEM++ - Wikipedia」(Oct.13,2018 12:01 UTC) より引用

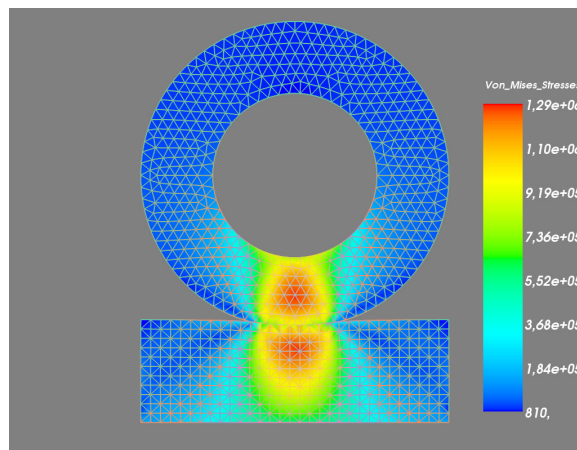
ライセンスや受賞歴についてはページを参照してください。GetFEM++ のチュートリアルには図 1 に示す例が掲載されています。GetFEM++ はソースのホスティングサービスとして Savannah を使用しています。GNU Savannah について Wikipedia の説明を引用します。

GNU Savannah は、フリーソフトウェアプロジェクトのための協働型ソフトウェア開発管理システムを提供するフリーソフトウェア財団のプロジェクトです。現在は、CVS、GNU arch、Subversion、Git、Mercurial、GNU Bazaar、メーリングリスト、ウェブホスティング、ファイルホスティング、バグ管理サービスといった機能を提供しています。Savannah は、SourceForge.net と同じソフトウェアをベースと

[†] E-mail address of corresponding author: tkoyama010@gmail.com



(a) 熱弾性的および電気結合の例



(b) 車輪の接触の例

Fig. 1: GetFEM++ を使用した解析例

したホスティングサービスシステムである Savane を使っています。

「[GNU Savannah - Wikipedia](#)」(Oct.13,2018 11:31 UTC) より引用

このように、[Savannah](#) は GNU プロジェクトやフリーソフトのホスティングサービスとして広く使われています。このホスティングサービスでは非開発ユーザの場合、以下のようにして、リポジトリをクローンしコンパイルとインストールを行います。

```
01 user@linux ~$ git clone https://git.savannah.nongnu.org/git/getfem.git
```

標準的な GNU ツールを使用しているので、GetFEM++ ライブラリのインストールは幾分標準的です。異なるプラットフォーム上のインストールの詳細については、[ダウンロードとインストールページ](#)を参照してください。また、ソースからコンパイルする必要のない場合には以下のように aptitude(もしくは apt) コマンドによりインストールすることも可能です。

```
01 user@linux ~$ sudo aptitude install python-getfem++ libgetfem++-dev
```

これにより [GetFEM++](#) の [Python](#) インターフェースとライブラリがインストールされます。

一方、Savannah のプロジェクトの開発に参加するには図 2 の "New User" からユーザー登録が必要です。開発者としてプロジェクトに参加するには、管理者にメッセージを送信し参加を承認される必要があります。2018 年 10 月現在の [GetFEM++](#) の管理者は [Konstantinos Poullos](#) 氏と [Yves Renard](#) 氏です。開発者として権限が与えられたら、[How to contribute / Git repository on Savannah](#) を参考にプロジェクトに commit をします。開発者として、リポジトリをクローンする場合は以下のようにします。

```
01 developer@linux ~$ git clone ssh://savannah-login@git.sv.gnu.org:/srv/git/getfem.git
```

ここで、savannah-login は Savannah に登録したユーザ名です。ただし、何の設定もせずこれを実行すると認証エラーが発生します。このコマンドを実行する前に Savannah に認証鍵を登録しておく必要があります。まずは、ホームディレクトリで ssh-keygen コマンドを実行して認証鍵を作成してください。

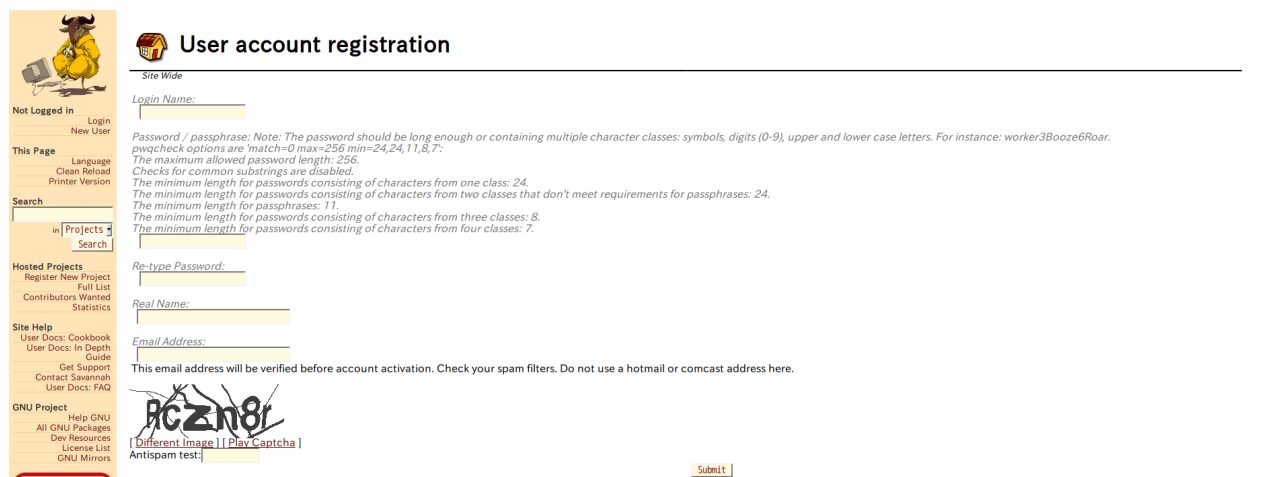


Fig. 2: Savannah の新規ユーザ作成画面

```

01 developer@linux ~$ cd /home/developer
02 developer@linux ~$ ssh-keygen
03 Generating public/private rsa key pair.
04 Enter file in which to save the key (/home/developer/.ssh/id_rsa):
05 Created directory '/home/developer/.ssh'.
06 Enter passphrase (empty for no passphrase):
07 Enter same passphrase again:
08 Your identification has been saved in /home/developer/.ssh/id_rsa.
09 Your public key has been saved in /home/developer/.ssh/id_rsa.pub.
10 The key fingerprint is:
11 SHA256:NVKmqK8YvqZiSakm3N05+zLkr1PARjZnm1DIvqKVWks developer@linux
12 以下略
13 developer@linux ~$

```

これにより/home/developer/.ssh/id_rsa.pub に RSA 公開鍵のファイルが作成されます。これを、Savannah に図 3 のように作成されたファイルの中身を登録すると登録した認証鍵を持つパソコンで Git の操作が可能になります。

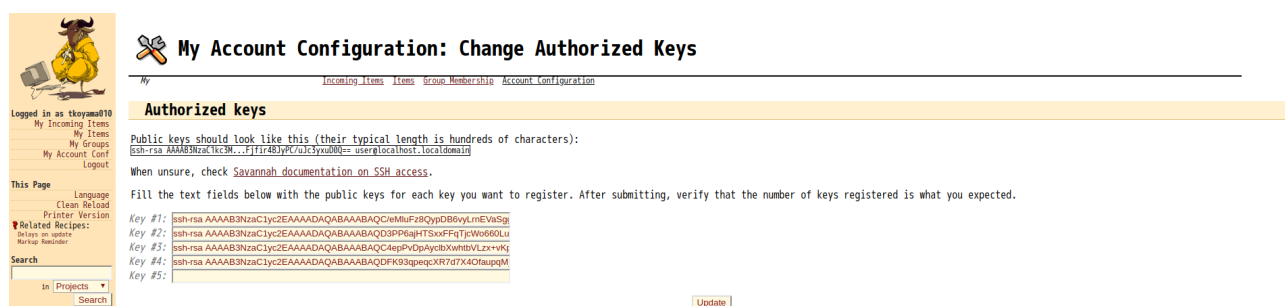


Fig. 3: Savannah の承認鍵の設定

GetFEM++ プロジェクトでソースの変更を行う際には Git で以下のようにブランチを作成する規約となっています。

```

01 developer@linux ~$ cd /path/to/getfem
02 developer@linux ~$ git branch devel-name-subject

```

```
03 developer@linux ~$ git checkout devel-name-subject
```

ここで、name は開発者の名前、subject は開発内容です。^{*1} 修正後 commit をしてブランチに push します。

```
01 developer@linux ~$ cd /path/to/getfem
02 developer@linux ~$ git commit -m"開発内容のメッセージ。もちろん英語!"
03 developer@linux ~$ git push origin devel-name-subject
```

修正後メーリングリスト getfem-commits@nongnu.org に連絡をして master ブランチへのマージを管理者にリクエストしてください。誤って master ブランチに push しないように十分注意してください!!! 管理者から注意され revert されます。

3. GetFEM++ のコンパイルとテスト方法

GetFEM++ のコンパイルとテスト方法について説明します。まずは、getfem のリポジトリ直下にある autogen.sh を実行します。実行には libtoolize と automake が必要になります。

```
01 devlname@linux ~$ sudo aptitude install libtool automake
```

また、コンパイルに使用するライブラリとして以下のパッケージをインストールします。

- libqhull-dev - calculate convex hulls and related structures (development files)
- libmumps-dev - Direct linear systems solver - parallel development files
- libmumps-ptscotch-dev - Direct linear systems solver - PTScotch-version development files
- libmumps-scotch-dev - Direct linear systems solver - Scotch-version development files
- libmumps-seq-dev - Direct linear systems solver - non-parallel development files
- liblapack-dev - Library of linear algebra routines 3 - static version
- libopenblas-dev - Optimized BLAS (linear algebra) library (development files) ^{*2}

さらに、Python のためのヘッダーファイルとスタテックライブラリ Scipy をインストールします。Python2 の場合

- python-dev - header files and a static library for Python (default)
- python-scipy - scientific tools for Python

Python3 の場合

- python3-dev - header files and a static library for Python (default)
- python3-scipy - scientific tools for Python

をそれぞれインストールします。インストール後 ./configure をプロジェクトフォルダで実行して、Makefile を作成します。Python2 の場合、

```
01 developer@linux ~$ ./configure --with-pic
```

Python3 の場合、

```
01 developer@linux ~$ ./configure --with-pic --enable-python3
```

^{*1} プロジェクトへの貢献方法はプログラムの新機能のパッチ作成だけではありません。ドキュメントを読んでいる際に気づいた Typo(打ち間違い)の修正も大切な貢献です。著者は Typo の修正のために `fixfixmisspell` という branch を作成したところ、Typo 修正用の公式 branch として採用されました。詳細は、<http://getfem.org/project/contribute.html> の "Specific branch for doc improvements and typo-fixes" を参照してください。自分の小さな提案が採用されることも OSS への貢献の面白さです。

^{*2} 第 14 回オープン CAE 勉強会@関東（構造など）にて openblas が通常の blas より高速であるとの指摘を @michioga 氏よりいただきました。

を実行します。Makefile 作成後以下のコマンドでコンパイルとテストの実行をします。

```
01 developer@linux ~$ make && make check
```

`automake` により整備されたテスト項目が実行され結果がログファイル `tests/test-suite.log` に出力されます。テストに失敗した場合は `getfem-users@nongnu.org` に `tests/test-suite.log` を添付し報告をしてください (もちろんメッセージは英語です!!!)。

4. GetFEM++ の Sphinx ドキュメント日本語化プロジェクト

4.1. sphinx-intl による翻訳ワークフロー

GetFEM++ の Sphinx ドキュメントをコンパイルするには `doc/sphinx` で以下のコマンドを実行します。

```
01 translator@linux ~$ make html (html を作成する場合)
02 translator@linux ~$ make pdf (pdf を作成する場合)
```

これにより `doc/sphinx/build` 以下の `html` ディレクトリと `pdf` ディレクトリにドキュメントが作成されます。コンパイルされた Sphinx ドキュメントを国際化のページを元に翻訳します。ワークフローを図 4 に示します。一連の作業は `sphinx-intl` を使用して行います。`sphinx-intl` は Sphinx での翻訳フローを便利にするツールです。

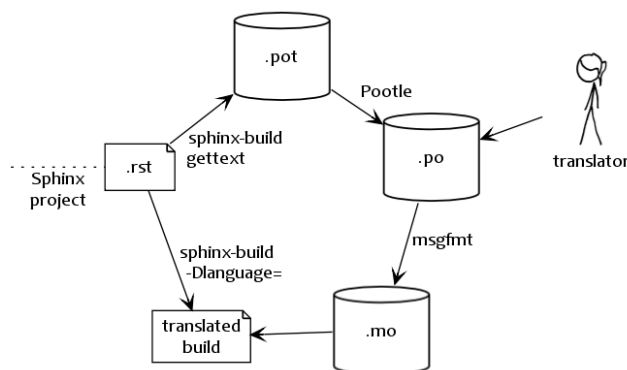


Fig. 4: Sphinx による翻訳のビジュアルなワークフロー (「国際化」より引用)

`sphinx-intl` のインストールは `pip` を使用して以下のように行います。

```
01 translator@linux ~$ sudo aptitude install python-pip
02 translator@linux ~$ sudo pip install sphinx-intl
```

`aptitude` で `sphinx-intl` をインストールをすることも可能ですが、Python2 のバージョンを確実にインストールするためにこの方法をとります。次に、`doc/sphinx/Makefile.am` と `doc/sphinx/source/conf.py` に以下の変更を適用します (「Sphinx と transifex を活用した翻訳手順」(Oct.14,2018 05:45 UTC) を参照)。

```
diff --git a/doc/sphinx/Makefile.am b/doc/sphinx/Makefile.am
index adb8259e..2cab2a69 100644
--- a/doc/sphinx/Makefile.am
+++ b/doc/sphinx/Makefile.am
@@ -21,8 +21,10 @@
# You can set these variables from the command line.
PYTHON          = python
SPHINXROOT      = http://svn.python.org/projects
-SPHINXOPTS     =
+LANGUAGE       = en
```

```

+SPHINXOPTS      = -D language=$(LANGUAGE)
SPHINXBUILD      = $(PYTHON) tools/sphinx-build.py
+SPHINXINTL      = sphinx-intl
PAPER            =
SOURCES          = $(srcdir)/source
DISTVERSION      = @VERSION@
@@ -33,8 +35,9 @@ PAPEROPT_a4      = -D latex_paper_size=a4
PAPEROPT_letter  = -D latex_paper_size=letter
ALLSPHINXOPTS    = -b $(BUILDER) -d build/doctrees $(PAPEROPT_$(PAPER)) \
                  $(SPHINXOPTS) $(SOURCES) build/$(BUILDER)
+I18NSPHINXOPTS  = $(SPHINXOPTS) ./source

-.PHONY: help checkout update images build view html htmlview htmlhelp latex pdf
        linkcheck clean upload
+.PHONY: help checkout update images build view html htmlview htmlhelp latex pdf
        linkcheck clean upload gettext

help:
@@ -85,6 +88,7 @@ images:
        -cd $(srcdir)/source/scilab/images/; make png

build: $(srcdir)/source/matlab/cmdref.rst $(srcdir)/source/python/cmdref.rst $(
        srcdir)/source/scilab/cmdref.rst checkout images
+
+    $(SPHINXINTL) build
        echo # rm -fr build/$(BUILDER)/_images
        echo # rm -fr build/$(BUILDER)/*.png
        mkdir -p build/$(BUILDER) build/doctrees
@@ -110,7 +114,7 @@ htmlhelp: build
latex: BUILDER = latex
latex: build
        @echo "Build finished; the LaTeX files are in build/latex."
-        @echo "Run \`make all-pdf' or \`make all-ps' in that directory to" \
+        @echo "Run \`make all-pdf' in that directory to" \
        "run these through (pdf)latex."

pdf: BUILDER = latex
@@ -122,6 +126,11 @@ linkcheck: BUILDER = linkcheck
linkcheck: build
        @echo "Link check complete; look for any errors in the above output " \
        "or in build/$(BUILDER)/output.txt."

+gettext:
+    $(SPHINXBUILD) -b gettext $(I18NSPHINXOPTS) locale
+    $(SPHINXINTL) update -p locale -l $(LANGUAGE)
+    @echo
+    @echo "Build finished. The message catalogs are in locale."

clean:

```

```
-rm -rf build/
```

```
diff --git a/doc/sphinx/source/conf.py b/doc/sphinx/source/conf.py
index 14377f9d..fcaf2b7c 100644
--- a/doc/sphinx/source/conf.py
+++ b/doc/sphinx/source/conf.py
@@ -55,7 +55,7 @@ extensions = ['sphinx.ext.pngmath', 'sphinx.ext.autodoc',
#source_suffix = '.rst'

# The encoding of source files.
-#source_encoding = 'utf-8'
+source_encoding = 'utf-8'

# The master toctree document.
#master_doc = 'contents'
@@ -75,6 +75,9 @@ extensions = ['sphinx.ext.pngmath', 'sphinx.ext.autodoc',
# relative to the source directory
#locale_dirs = []

+gettext_compact = False
+locale_dirs = ['locale/']
+
# Add any paths that contain templates here, relative to this directory.
templates_path = ['.templates']
```

つぎに、以下のコマンドを doc/sphinx 以下で実行し locale ディレクトリ以下を初期設定あるいは更新をします。

```
01 translator@linux ~$ make gettext
```

このコマンドを実行すると pot ファイルの作成と、./locale/ja/LC_MESSAGES/以下に po のファイルが作成もしくは更新されます。図 4に示すように翻訳者は po ファイルの中身を編集します。po ファイルの中身は次のようになっています。

```
#: src/name.c:36
msgid "My name is %s.\n"
msgstr ""
```

翻訳者はこのファイルを編集し以下のようにします。(「[gettext - Wikipedia](#)」(Oct.13,2018 17:31 UTC) より引用)

```
#: src/name.c:36
msgid "My name is %s.\n"
msgstr "私の名前は %s です。 \n"
```

翻訳が完了したら、make html もしくは make latex コマンドを実行します。LANGUAGE の en の部分は日本語の翻訳を適用する際には ja に変更してください。翻訳を適用する場合は make pdf は使用できないため注意してください。make latex を実行した後 doc/sphinx/build/latex 以下で make all-pdf-ja を実行してください。

4.2. Transifex の使用方法

前節で作成した*.po ファイルを Transifex を使用して管理します。Transifex のサービスについて Wikipedia の内容を引用します。

[Transifex](#) は Web ベースの翻訳プラットフォームであり、ローカライズ管理システムとしても知られている。ソフトウェア、文書、Web サイトなど頻繁に更新されるコンテンツと技術的なプロジェクトを対象とし、開発者が使用するツールと統合することにより翻訳ワークフローを自動化している。有料機能とオープンソースプロジェクト用の無料アカウント機能の両方を SaaS として提供している。サイト内では翻訳者が共同作業できるようにするために翻訳対象のファイルホスティングや、フィード、掲示板投稿、翻訳提案や投票などのソーシャル・ネットワーキング・サービスとしての機能が利用できる。サイトは Django と Python で構築されている。「[Transifex - Wikipedia](#)」(Oct.14,2018 06:28 UTC) より引用

本プロジェクトで新しく GetFEM++5.3 の翻訳プロジェクトのために [図 5](#) のページを作成しました。翻訳の提案などはこちらのページで行うことができます。[Transifex](#) での翻訳結果を反映するためにはクライアントコマンド



Fig. 5: [GetFEM++5.3 翻訳プロジェクト](#)

tx を使用します。tx コマンドは以下のようにインストールします。

```
01 translator@linux ~$ sudo aptitude install transifex-client
```

インストール後 doc/sphinx 以下で以下のコマンドを実行してください。

```
01 translator@linux ~$ tx pull -l ja
```

このコマンドは設定ファイル doc/sphinx/.tx/config を参照して source/locale 以下に*.po ファイルをダウンロードするコマンドです。^{*3} API キーを聞かれた際には [Transifex](#) のアカウントのページで API キーを発行してください。po ファイルをダウンロード後、前節で紹介した手順で翻訳されたドキュメントを作成します。なお本プロジェクトではトラブル防止のため、翻訳の修正を提案する場合は必ず [Transifex](#) の Web ページから行ってください。tx push コマンドは使用しないでください。また、tx push をした後に make gettext を行うと po ファイルが更新されダウンロードした翻訳が消えることがあるため注意してください。

以上の手順をまとめたものが [GetFEM++](#) の公式ドキュメント [Contributing to document translation](#) として Merge されました。今後はこちらの文書で翻訳方法の情報更新を行います。

5. 技術書典 5 サークル参加

翻訳した文書を頒布するため、技術書の即売会である [技術書典 5](#) に参加しました。出典ブースの様子を [図 6](#) に示します。日時は 2018/10/08 (月) 11:00~17:00 で場所は 池袋サンシャインシティ 2F 展示ホール D (文化会館ビル 2F) でした。50 部を印刷し、頒布数は 38 部 (内 4 部は挨拶用・2 部はオープン CAE 勉強会参加者様) でした。ブースには以下の方ような方がいらっしゃいました。

- 昔商用 FEM コードを使われていて今は使われていない方
- 商用 FEM コードを使用されている方
- これから FEM を勉強される学生の方
- OpenFOAM を使用されていて、構造にもオープンソースがあることをはじめて知った方

^{*3} doc/sphinx/.tx/config はバージョンアップの際に Transifex のメンテナーが更新を行います。doc/sphinx で以下のコマンドを実行します。

```
01 admin@linux ~$ sphinx-intl update-txconfig-resources --pot-dir build/locale --transifex-project-name="getfem-<version number>"
```




Fig. 6: サークル GetFEM++@技術書典5のブース

- ポスターの絵がきっかけで CAE に興味を持たれた方

一般購入者の方で構造でオープンソースを使用されている方はおらず、ポスターのような解析がオープンソースでもできることに驚かれもしました。オープンソースの有限要素法解析ソフトへの認知度は低く、今後も認知のためのドキュメント頒布が必要と考えています。次回の出典では図 7 に示す 3 冊のドキュメントの配布を予定しています。

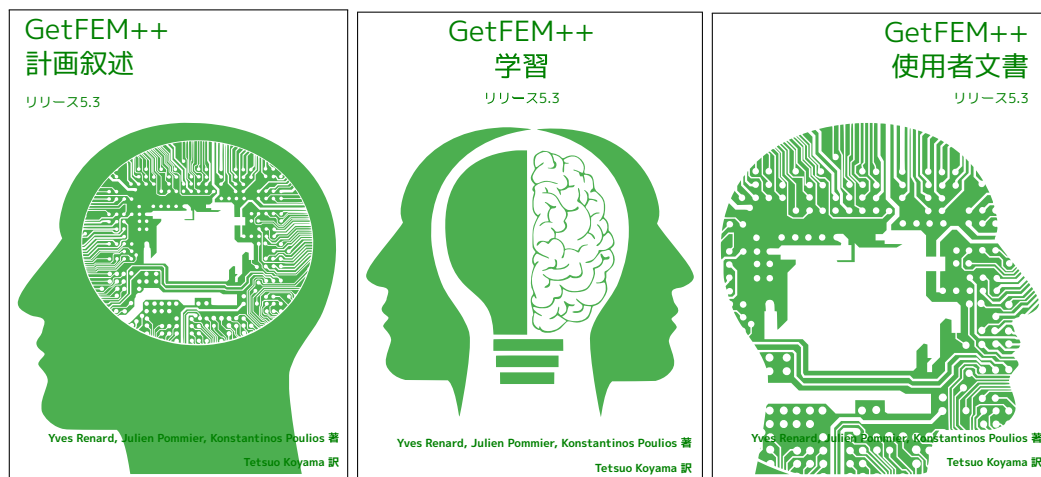


Fig. 7: 次回配布予定のドキュメント

6. まとめ

以上のまとめを以下に示します。

- [GetFEM++ の Wikipedia の日本語版](#)を作成することにより GetFEM++ プロジェクトの目的を紹介した。
- [GetFEM++](#) のソースのホスティングサービス [Savannah](#) でプロジェクトに貢献する方法を紹介した。
- [GetFEM++ の Sphinx ドキュメント日本語化プロジェクト](#)の内容について紹介した。
- [技術書典5](#)への参加報告をおこなった。

今回翻訳した文書の html は [GitHub Pages](#) として公開されている。