Chapter 1

はじめに

次の文章はどちらが正しいだろう ？

- ユーザは “ブラックボックス” としてのソフトウェアを望んでいる。すなわち，ユーザはアルゴリズムの詳細を理解しなくとも一般的な問題に対して全幅の信頼を持って利用できるソフトウェアを求めている。
- ユーザは特定の応用に対してデータ構造が調整できることを望んでいる。たとえ，そのソフトウェアが一般的な解法として提供されたものほど信頼性がないとしても。

ユーザの集団が異なれば，これはどちらも正しいと言えるだろう。

伝統的には，ユーザはブラックボックスとしてのソフトウェアを要求し，それらは LAPACK, LINPACK, NAG および IMSL などの数学ライブラリの形として提供されてきた。ごく最近，高性能計算のグループはユーザの問題に応じてあつらえたソフトウェアを書かなければならないことを認識し出した。その理由として，既存のソフトウェアライブラリは機能的に不適切であったり，ある特定の問題に対して不自然あるいは不便なデータ構造をしていたり，きわめて汎用的なソフトウェアは興味のある特別のケースに適用したときあまりにも性能を犠牲にしていることをあげている。

われわれは両方のユーザグループのニーズに応えることができるのだろうか ?われわれはきっとできると信じている。それゆえ，本書で 『templates』の使い方を紹介することにしよう。template とは，伝統的なソフトウェアライブラリにお決まりの実行形式のオブジェクトコードやソースコードのことではなく，もっと一般的なアルゴリズムの記述を意味している。templates はキーとなるアルゴリズムの一般的記述であるにもかかわらず，ユーザの要求に応じた仕立て直しを可能にしている。たとえば，問題に応じた特別なデータ構造用に，または問題を実行する特別なコンピューティングシステム用にプログラムを仕立て直すことができる。

本書では大規模で疎な線形連立 1 次方程式を解くために反復法を使うことに焦点を合わせている。

そのような問題を解くためには多くの解法が存在している。うまくやるには問題に対する最も効果的な解法を見つけることである。不幸にして，ある種の問題には良い解法が別の種類の問題にはあまり良くないかもしれない。実際のところ，まったく動かないかもしれない。

そのため，われわれは templates を提供する以外に，効果的な手法をどのように選ぶのか，それをどうやって記述するのかについても提案している。また，特定の種類の行列に対してその手法をどのように適用するかについても提案をしている。われわれは「反復法 (Iterative methods)」に限定している。反復法は精度が十分になるまで繰り返して近似解を改良する手法である。これらの解法は，行列–ベクトル積 $\boldsymbol{y} = A\boldsymbol{x}$ ($\boldsymbol{z} = A^T\boldsymbol{y}$ が必要かもしれないが) を通じてのみ方程式の係数行列 A を参照する。すなわち，ユーザは与えられた $\boldsymbol{x}$ について，$\boldsymbol{y}$ または $\boldsymbol{z}$ を計算するためのサブルーチンだけは用意する必要がある。そこで，行列 A の疎の特徴やその他の特別な構造が完全に記述できるだろう。

本書を読んだあと，応用プログラムの開発者は並列計算機で高速に動くプログラムを得るために templates が使えると，われわれは確信している。専門家でなくても，特定の問題を解くための手法をどのように選択して実現するかがわかるだろう。専門家にとっても，これまでのように特定の計算機に合わせて大型の応用プログラムを調整するための膨大な投資をすることなしに，自分たちのコードを組み直したり修正したりできるようになることだろう。最後に，われわれはすべてのユーザが，使用したアルゴリズムをより深く理解することを期待している。教育が数学ソフトウェアの伝統的な目標のひとつになっていない現状では，われわれのアプローチはこうした価値あるサービスの提供に大いに役立つと確信している。

1.1 なぜ templates を使うのか ?

templates には 3 つの重要な利点がある。まず第 1 に，templates は一般的であり，かつ再利用できる。そのため，いろいろな機種に簡単に持っていける。この特徴はいろいろな並列アーキテクチャが与えられている場合に重要になる。

第 2 に，templates は 2 つの異なった集団にその専門性を発揮させる。数値解析の専門家は，特定の数値計算技術の深い知識を反映させて templates を新しく作り出す。そして計算科学者は一般的な templates の記述に「付加価値」としての性能を与えて，それを特定の状況や応用の必要性に応じて仕立て直す。

そして第 3 に，templates は言語固有の話ではない。むしろ Algol 風な構造で表され

て，FORTRAN (可能な限り，基本線形代数サブプログラムBLAS を用いて) やC のような目的言語にただちに変換可能なものである。このような親しみやすい体裁をとることによって，ユーザがアルゴリズムを信頼してくれると確信しているし，ユーザが数値計算の技術や並列プログラミングについてより良い理解を得ることも期待している。

おのおのの template には以下の点のいくつか，またはすべてが提供されている。

- 反復計算の流れに関する数学的な記述
- 収束や停止規準についての検討
- 特定の形の行列 (たとえば，帯行列) に手法を適用する際の提案
- チューニングに関するアドバイス (例として，どの前処理が適用できて，どれが適用できないかなど)
- 並列向きにする際の秘訣と，
- どのようなときにその手法を使うか，なぜ使うかについてのヒント

templates おのおのについて，以下のものが電子メールを通じて入手できる。

- 密行列に対する MATLAB のプログラム
- BLAS をコールしている FORTRAN-77 プログラム

詳細は付録 A を参照のこと[1]。

1.2 どんな解法が含まれているか

多くの反復法が開発されていて，それらすべてを網羅することは不可能である。われわれは反復法の歴史的な経緯を説明しているか，あるいは大規模で疎な線形方程式の解法に対して最新の成果を表しているかのいずれかの理由で以下の解法を選び出した。取り上げた解法は，

1. Jacobi 法
2. ガウス–ザイデル法

[1] BLAS のブロック化についての議論は [66, 67, 68, 141] と LAPACK のルーチン [3] を参照のこと。また付録 B も参照せよ。

3. 逐次的過剰緩和法 (SOR 法)
4. 対称逐次的過剰緩和法 (SSOR 法)
5. 共役勾配法 (CG 法)
6. 最小残差法 (MINRES 法) と対称 LQ 法 (SYMMLQ 法)
7. 正規方程式に対する共役勾配法 (CGNE 法と CGNR 法)
8. 一般化最小残差法 (GMRES 法)
9. 双共役勾配法 (BiCG 法)
10. 擬似最小残差法 (QMR 法)
11. 2 乗共役勾配法 (CGS 法)
12. 安定化双共役勾配法 (Bi-CGSTAB 法)
13. チェビシェフ反復法

である。

各解法に対して，われわれは解法の歴史に対する議論と多くの文献参照を含んだ一般的な記述をしている。また，その解法を選択するための数学的条件も与えている。

われわれは「料理の本」を書くことを意図していないし，“numerical recipes” という言葉は慎重に避けている。なぜなら，これらの言葉はわれわれのアルゴリズムが方程式に関する知識なしに盲目的に使用できるということを意味してしまうからである。反復法での最新の成果はこのようなことを許していない。すなわち，線形方程式についての知識はこれらのアルゴリズムの収束を保証するために必要だし，また一般的に，知識が増えれば増えるほど，アルゴリズムをチューニングできるようになる。そこで，われわれはアルゴリズムの概要を，解法を選択し特定の高性能計算機上に実装するための指針とともに示すことにした。合わせて，前処理行列の使い方やデータの適切な格納方法の問題についても検討を加えている。