Vol.2018-HPC-165 No.18
2018/7/31

運用効率化を目指した設備シミュレーションのための ソフトウェア基盤の設計

松葉 浩也[1,a)]　松田 元彦[1,b)]　河合 直聡[1,c)]

概要：様々な産業設備の省エネルギー運用が重要視される中、シミュレーションによって設備の効率的な運用パラメータを決定するニーズが高まっている。動力、冷却系、電力系など様々な要素を含む設備をシミュレーションで再現するためには、設備の構成要素が個別にシミュレーションできるだけでは不十分で、構成要素間の連携が必須である。本稿では簡易な記述にて、シミュレーションモデルやそれらの接続関係を表現可能な新しい言語の設計を議論する。

Design of Simulation Platform for Efficient Facility Operations

Hiroya Matsuba[1,a)]　Motohiko Matsuda[1,b)]　Masatoshi Kawai[1,c)]

Abstract: This paper discusses the design of new software platform that enables us to easily describe simulation models of industrial facilities, such as power plants or cooling systems. These models are intended to be used for searching for efficient operation parameters of the facilities. Because the industrial facilities usually consist of various kinds of components, creating simulation models for such facilities requires the combined simulation models of each component. This paper presents initial design of a new computer language that enables simple descriptions of simulation models and their combination.

1. はじめに

水処理プラント、発電設備、石油化学工場など、我々の生活は多くの産業設備によって支えられている。これらの産業設備は、様々な物理、化学現象を利用して原材料から目的物を生成し、その過程でエネルギーを消費し、副産物を生成する。産業設備が効率的に運用されるべきであることは多くの人が同意するところではあり、その場合「効率的」とは具体的には、一定の目的を果たす際に消費する原材料やエネルギー、あるいは副産物の生成量を最小化することを意味することが多い。

産業設備の運用効率は、設計時点で決定する設備の機能や性能だけでなく、運用時に決定するパラメータに依る部分も大きい。例えば、ある産業設備の中に常にセ氏 50 度以下に冷却しておくことが求められる機械がある場合を想定する。通常は少し余裕をもたせ、例えば定常状態で 40 度となるよう冷却設備の出力を調整する。この冷却設備の出力が運用パラメータであり、40 度を目指した出力に設定するか、50 度を目指した出力に設定するかによって全体的な運用効率は変化する。

産業設備では、多くの運用パラメータを決定する必要があるが、熟練者の経験を頼りに安全第一で設定される場合があり、そこに改善の余地がある。例えば先述の例では、過去の実績を基に、安定して稼働している 40 度を選択しているものの、それより高い温度を不適切としている根拠は明らかでない場合がある。あるいは、気温など、日々変化する環境に併せて運転員が調整を行うようなパラメータに関して、妥当性の評価を行うためのデータが不足している場合もある。このように、過剰な安全率を見込んだり、パラメータ選択の根拠を検討するデータの不足などにより、設備の運用効率が最適化されていないケースは珍しくない。このような状況を解消して、全体的な運用効率を改善

[1] 理化学研究所計算科学研究センター
RIKEN Center for Computational Science
[a)] hiroya.matsuba@riken.jp
[b)] m-matsuda@riken.jp
[c)] masatoshi.kawai@riken.jp

することは多くの設備運用者にとって重要な課題である。

未経験の運用パラメータを試行し、効果や安全性を検証する手段として、計算機によるシミュレーションが挙げられる。産業設備全体の運転状況を正確に再現できるシミュレータがあれば、一部の機械の出力を下げる影響がどのように広がるかなど、運用パラメータ変更の結果を実際の設備を使わずに検証することが可能となり、効率的な運用パラメータの探索に有効なツールとなる。

産業設備のシミュレーションを行う際の最大の問題は、設備のシミュレーションモデルの構築が困難な点にある。特に大型の産業設備は、多くの機械から構成されており、高度な科学的現象を活用するものもある。それらの構成機器をモデル化し、シミュレーション可能な状態に仕上げるためには機械工学と計算機双方に深い知識が必要である。さらに、個々の構成要素がモデル化できたとしても、それを接続し、設備全体の挙動を再現するためにはさらなるソフトウェア開発が必要である。自動車業界など、シミュレーション活用が進んでいる業界もあるが、その多くは一度のシミュレーション結果が数万台から数百万台の製品につながる業界である。このような場合はシミュレーション作成に必要な相対的コストが小さいため、技術力の高いシミュレーションエンジニアの雇用や養成が可能である。しかし産業設備については、まったく同じものは世界にひとつしか存在しない場合が多く、個々の設備のシミュレータ作成に多くの投資は不可能である。このような状況にある産業設備のシミュレーションを普及させるためには、現状よりも低コストで多種多様な機械から構成される産業設備をモデル化するための、モデル作成支援ソフトウェアを検討する必要がある。

本稿では産業設備を対象としたシミュレーションモデルの構築支援に必要なソフトウェアの設計について、特に新しいモデル記述言語の設計を中心に議論する。提案ソフトウェアでは特に次の二点に注力している。一点目は多様な構成要素の迅速なモデル記述である。前述のように産業設備は多くの機械から構成される。それぞれの構成要素は、1 次元に簡略化して動作だけを記述すれば良い場合もあれば、内部構造や周辺環境を含めた 3 次元の精緻なシミュレーションが必要なものも存在する。これらモデル化に関しては、手法こそ研究が進んでいるものの、その統一的な記述方法は存在せず、現状では例えば Modelica [6] と OpenFOAM [8] のような、まったく使い勝手の異なるものを複数学習する必要がある。本研究ではこの状況を改善し、モデル化手法が異なる対象であっても、類似性のある記述でモデル化が可能なモデル記述言語を検討する。二点目はモデル化された構成要素の相互接続である。産業設備は構成要素が接続されて成立しているため、シミュレーションにおいても各モデルの相互接続が容易に表現できる必要がある。これは各モデルを連成させてシミュレーションすることに他ならないが、こちらについても一点目と同様、手法の研究は進んでいるものの汎用的な記述方法は未整備である。提案ソフトウェアでは、その一部を構成するモデル記述言語の機能の一部として、モデル間の接続の効率的な表現を可能とすることを目指す。

提案ソフトウェアは、汎用プログラミング言語の特徴を有した DSL(Domain Specific Language) と位置付けている。汎用プログラミング言語は汎用であるがために目的を果たすまでの工数が多くなり、また、開発者によって大きく異なる設計を許してしまう側面がある。一方 DSL は、プログラム作成者が自身が必要とする機能を明確に認識できており、将来の拡張の予定もない場合は高生産性言語としての価値は高いが、徐々に機能、精度、性能などを向上させながら手探りで開発を進める場合、DSL が自身のプログラムの成長を制限するリスクがあるため採用が難しい。提案ソフトウェアは無限の拡張性を残しつつ、シミュレーションプログラム作成において高い生産性を実現することを目指す。

2. 必要機能の検討

本章では提案ソフトウェアに必要な機能を利用者の立場から検討する。本ソフトウェアの利用者としては、シミュレーション対象の設備の全体的な構成および個々の構成要素となる機械について、背景となる物理現象の基礎も含めた一定の理解を有する技術者または技術者のグループを想定する。計算機に対する習熟度としては、情報処理や並列計算の専門家ではないものの、並列化されていないスクリプト言語等を扱う技術力を有することは仮定する。具体的には対象設備を運用する企業の計画立案を専門とする技術者、運用改善を手がける技術コンサルタント等が該当する。

以下、要件を順に検討する。

2.1 モデル記述の容易性

産業設備を構成する多くの機器は、流体や熱の移動、電気的現象などの自然現象を利用していることが多いため、これらの現象を効率よく記述できることが重要である。自然現象を微分方程式で定式化し、数値的に解くことは盛んに行われているが、特に並列計算機業界では 2018 年現在でも C 言語等の低級言語での実装が標準的である。このようなプログラムには、モデルそのものではなく、離散化された微分方程式を数値的に解く手順のみが記述されるため、非専門家が作成するのは難しく、可読性も再利用性も低い。利用者の生産性のためには、微分方程式をできるだけ元の数式に近い形で記述できる機能が必要である。

微分方程式の種類や解法に依らない、できる限り統一的な記述も求められる。同じ対象でも複数のモデル化手法が考えられる場合があり、試行錯誤の中でモデル化手法を切り替える必要に迫られた際に、利用者の迅速な対応を可能

とするためである。例えば、計算機室のような常に冷却が必要な空間をモデル化することを考える。この空間は、熱容量を持つ物体に正負の熱源が接しているものとしてモデル化することも可能であり、また、３次元熱流体シミュレーションにより空間内の温度分布まで精緻に再現することも可能である。この選択はシミュレーションの目的や要求精度によって異なるが、当初は簡易なモデルを選択し、精度が十分でないことが判明した場合にのみ、精緻なモデルに変更するような試行錯誤は当然に発生する。このような場合に迅速な切り替えを可能にするためには、微分方程式の種類や解法にできるだけ依存しない記法の提供が必要である。

さらに制御や人為的イベントを記述するインタフェースも求められる。産業設備のシミュレーションにおいては、定常運転状態を再現したいだけでなく、異常発生時の挙動や人為的な操作を行った結果を調べたい場合も多い。このような非連続的な制御やイベントは、自然現象とは独立した形で記述できる必要がある。

2.2 モデル間接続の容易性

様々な機械から成る産業設備のモデル化においては、構成要素となる個々の機械をモデル化し、それを接続する形で全体がモデル化できることが必須である。全体を階層構造なくモデル化することは、実装とデバッグの両面から非効率的であるためである。また、構成要素となる機械自体についても、より細分化された部品の集合体として表現できることが重要である。

モデル間接続は連成シミュレーションと同義であり、それ自体は新しいものではないが、簡易かつ汎用的な記述でそれを実現するためには新たなソフトウェアが必要である。再び上記の空間冷却の例を考える。この例では、部屋を熱容量のある物体と考える場合でも、流体に満たされた空間と考える場合でも、対象自体に違いがあるわけではないため、他の要素（空調機など）とのインタフェースはモデル化方法に依存すべきではない。仮に、全体が１次元でモデル化されている中で計算機室だけを３次元シミュレーションに切り替えた場合、３次元の流体シミュレーションの方が多くの情報を出力するため、その出力を取捨選択、あるいは統計処理などし、熱容量のある単一の物体に近似した場合と同種のデータを生成する接続ソフトウェアが必要である。この接続ソフトウェアを簡易な記述で作成する方法が求められる。

3. 設計

本章では、提案ソフトウェアの具体的な構成を検討する。前章までに利用者の立場から議論した要件を満たすことは当然の前提とし、さらに、計算機科学、計算機工学の知見を加えた言語処理系、実行系を設計することを目指す。

3.1 利用者の定義

提案ソフトウェアのユーザーとして「エンドユーザー」と「シミュレーション作成者」を想定する。エンドユーザーとは、自身が管理する設備のシミュレーションを構築し、実行する立場のユーザーで、正確なシミュレーション結果を迅速に得ることに興味がある利用者である（前章は主にこの立場のユーザー視点で必要機能を検討した）。一方、シミュレーション作成者とはシミュレーションモデルの作成の専門家であり、軽量で正確なシミュレーションを行うための理論を開発、実装する立場である。

3.2 設計目標

提案ソフトウェアは、エンドユーザーとシミュレーション作成者が異なることを前提とする。そのため、以下の点に留意して設計する。

可読性

シミュレーション作成者が実装したモデルがエンドユーザーのニーズに合うか否かを、エンドユーザーが容易に判断できる必要がある。一般的にはこれはドキュメントを整備することで実現されているが、ドキュメントとプログラムを常に同期させて更新するのは容易でない。言語仕様を適切に設計し、シミュレーション作成者の意図がドキュメントなしでも伝わりやすくすることも重要である。

強制性

同じ内容の表現に複数の記述方法があると、エンドユーザーは混乱する。同じシミュレーションを実現するコードが異なる表現で乱立することを防ぐために、記述言語の表現力は適切に制限し、ある特定の概念を書く方法を、高々１個に強制することが望ましい。

再利用性

異なるシミュレーション作成者が作成したモデルを拡張した別のモデルを容易に開発できるよう、記述言語はオブジェクト指向の概念を採り入れ、既存の実装を再利用しながらの段階的な拡張を可能とすべきである。

3.3 モデル化対象の検討

シミュレーションには様々なモデル化手法があり単一の表現方法でそのすべてに対応するのは現実的でないため、まずは対象となるシミュレーションの分類を試みる。分類方法には様々な可能性があるが、ここではまず、正確な空間情報の必要性で分類する。

構造解析や流体シミュレーションなど、現在、スーパーコンピュータで盛んに実行されているシミュレーションは、対象となる空間の情報を使用する。多くの場合、空間情報はメッシュとして与えられる。また、ビルからの避難シミュレーションのように、エージェントモデルで表現されるシミュレーションも、エージェントが移動可能な領域として空間の情報（格子やグラフ構造）を使用する場合が

ある。

一方、空間の情報を用いないシミュレーションも存在する。例えば電気回路のシミュレーションはほとんどの場合、配線長を考慮する必要はなく、部品の接続関係のみが意味を持つ。また、本稿でもすでに述べたよう、空間内での変化を考慮せずにモデル化しても十分な精度が得られる場合、時間変化のみに注目したモデルで十分である。このようなシミュレーションはしばしば「1D シミュレーション」などと呼ばれる。

本稿では前者の空間情報を用いるシミュレーションに注力して議論する。空間的広がりを持つシミュレーションは、近年の計算機の大規模化によって可能になった部分が多く、ソフトウェア環境は未成熟である。OpenFOAM のような利便性と生産性の両立を目指すフレームワークも存在するが、基本的には利用者は特定用途を想定した商用シミュレータを使用するか、完全にプログラムを自作するかの両極端である。提案ソフトウェアはこの中間的存在を目指しており、対応可能なシミュレーションの種類には極力制限を設けず、かつ簡便な記述と容易な利用を実現する。

後者の 1D シミュレーションについてはすでに Modelica[6] と呼ばれる記述言語が存在しており、商用およびフリーの処理系が普及しているので、本稿では深く議論しない。Modelica は最初の提案が 1997 年である。すでに 20 年以上が経過しており、技術的にも成熟しつつあるため、この分野については既存の規格、実装を用いるのが合理的である。

3.4 エンドユーザーインタフェースの検討

空間情報を用いるシミュレーションは次の 4 要素から成り立つと考えられ、これはモデル化手法には大きく依存しない。

基本モデル　支配方程式など、系全体にわたって有効な基本的原理。

空間情報　シミュレーション対象の空間を定義するデータ。メッシュやグラフ構造など。

対象固有の条件　境界条件など、シミュレーション対象に固有の条件、あるいはシミュレーションの実行ごとに与える条件。

制御　設備に故障を発生させる、あるいは急激な条件の変化を与えるなど、人為的に条件を大きく変更するための指示。

提案ソフトウェアは、これら 4 点の情報をエンドユーザーから受け取ることを基本とする。この基本構造を固定することは、先に必要機能として議論した、モデル化手法に依らない統一的なインタフェースの提供を実現する重要な要素である。

上記の 4 要素を検討すると、最初に挙げた「基本モデル」はシミュレーション対象に依らない汎用性がある部分であり、それ以下の「空間情報」「対象固有の条件」「制御」はシミュレーション対象固有である。つまり、前者はシミュレーション作成者によって提供され、エンドユーザーはそれを選択する。後者はエンドユーザーから直接提供される。また、両者は依存関係にあり、基本モデルの構成方法により、エンドユーザーが提供すべき空間情報の種類や性質、与えるべき条件、受取可能な制御情報は異なる。

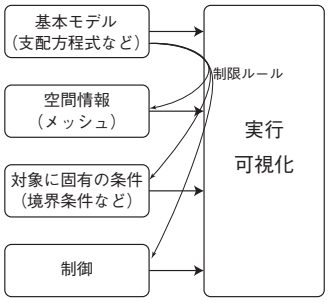

図 1　シミュレーションの抽象的構造

以上の議論を踏まえ、**図 1** に提案ソフトウェアの基本的な入力の関係を示す。エンドユーザーにより上記の 4 要素が実行系に入力され、シミュレーションは実行される。ただし、シミュレーション作成者が記述する基本モデルにより、空間情報、対象固有の条件、および制御には制限が設けられる。モデルごとに独自フォーマットの入力ファイルが乱立するのを避けるため、基本モデルの記述では、実行系がサポートする入力の中から受取可能なものを選択することのみが可能であり、モデル固有のファイルフォーマットを定義したり、モデル独自の入力を定義することはできない仕様とする。新たなモデル作成の際に、新たな入力が必要となった際には、汎用性に注意しながら、実行系側に新たなファイルフォーマットを規定する。

3.5 基本モデル記述言語の検討

提案ソフトウェアは特定のモデル化方法に依存するものではないが、シミュレーション全体を支配するルールの記述方法を検討するにあたり、本稿では最初のステップとして偏微分方程式で記述されるモデルを例に検討する。

図 2 に 2 次元流体シミュレーションの記述例を示す。本図のコードは設計中のものであり、本稿執筆時点で処理系は未実装である。したがって、実現可能性に関しては一定の留意はしているものの、未確定な部分は残されている。

コードの最上部はモデルの機能と性質を定義している。キーワード `class` に続く部分はモデルの名称であり、続いて `declaration` セクションに支配方程式である偏微分方程式が記述される。この偏微分方程式を計算機が理解し、解くことができれば最善であるが、現実的には離散化手法を計算機が自動的に発見するのは困難であるため、この部分の記述はシミュレーション実行には利用されない（後に説明するように離散化式は別途モデル作成者によって与え

```
class 2D-CFD
    declaration:
       ∂u/∂t + (u·∇)u = -∇p + (1/Re)∇²u
       ∇·u = 0
    time: t
    space: 2d-orthogonal-grid(x, y)
    condition: Re, dt, max(t),
               initial(u), boundary(u)
    control: /* nothing */
relation
    fgen pdcc(f)[s] = (f(s:+1) - f(s:-1))/(2*ds)
    fgen pdce(f)[s] = (f(s:0) - f(s:-1))/ds
    fgen p2dcc(f)[s] =
            (f(s:+1) - 2*f(s:0) + f(s:-1))/(2*ds^2)

    function k = pdcc(u_x)[x]^2 + pdcc(u_y)[y]^2 +
                 2*pdcc(u_x)[x]*pdcc(u_y)[y]
    function l = pdce(pdcc(u_x)[x] + pdcc(u_y)[y])[t]
    equation pss(p) {
         p2dcc(p)[x] + p2dcc(p)[y] = - k - l
    }

    fgen kk(v, f)[s] = v(s:0)*
          (-f(s:+2)+8*(f(s:+1)-f(s:-1))+f(s:-2))/
          (12*ds) +   |v(s:0)|*
          (f(s:+2)-4*f(s:+1)+6*f(s:0)-4*f(s:-1)+f(s:-2))/
          (4*ds)
    fgen ddt(v)[r, s] = - pdcc(p)[r] +
           (1/Re)*(p2dcc(v)[r] + p2dcc(v)[s]) -
           (kk(u_r, v)[r] + kk(u_s, v)[s])

    forward tstep_ux(u_x)[t] { ddt(u_x)[x, y] }
    forward tstep_uy(u_y)[t] { ddt(u_y)[x, y] }
end
```

図 2 コード例

らる)。この declaration セクションは、現時点では変数の宣言と、エンドユーザーの直感的な理解のための見出しの役割のみを果たす。一方で、このレベルの抽象的な記述は、可視化、チューニング、解法選択などの自動化に有用な情報を与える可能性があるため、今後の検討にてより多くの役割が与えられる可能性がある。

続く time セクションはこのモデルが時間発展することを意味しており、モデル記述の中で変数 t が時間を表すことを意味する。

セクション、space, condition, および control がそれぞれ上で議論した「空間情報」「対象固有の条件」「制御」に対する要求を記述している。この例では空間については２次元の直交格子で定義される空間を求めており、モデル記述内の変数 x および y がその空間内の座標であることを意味している。対象固有の条件については、定数として Re（レイノルズ数)、dt（時間間隔)、および変数 t の最大値（シミュレーション終了までのステップ数）をエンドユーザーから受け取ることを求めている。制御に関してはこの例では利用しないものとしている。

続く relation セクションには declaration セクションに記述した偏微分方程式の離散化式が記述され、ここでの記述が実際のシミュレーション実行に利用される。この例では、既知の速度場を用いてポアソン方程式を解き圧力場を求め、それを基に次のタイムステップの速度場を陽的に求める手順が記述されている。この手順の記述は宣言的であり、テキスト中の記述の前後は処理の順序とは無関係である。ただし、全体の求解順序が自動的に決定できるよう、依存関係が解決でき、循環しないよう記述するのはシミュレーション作成者の責任としている。これは一見難しく見えるが、概念は表計算ソフトと同じであり、多くのユーザーになじみのある仕組みだと考えられる。

言語仕様は検討中であり、シンタックス、セマンティックスともに現時点では正確には定義されていないが、以下のような機能を提供する予定である。

関数から変数への暗黙変換

declaration セクションに関数として登場した記号は、relation セクションでは time および space で定義した空間にわたり離散的に定義された配列と解釈される。また、declaration セクションにあるベクトル関数 u は、u_x と記述することで x 軸成分のみを抽出したスカラー関数になる。

関数ジェネレーターの記述

数学の演算子に相当する関数を生成する記号を定義する機能を持つ。例えば本セクションの最上部にある

```
fgen pdcc(f)[s] = (f(s:+1) - f(s:-1))/(2*ds)
```

は中心差分で計算した偏導関数を離散化した配列（偏微分係数の配列）を生成する。この新しく生成される配列の各要素は、配列 f の同じ位置から s 軸方向に 1 進んだ値と 1 戻った値を加算し、2*ds で除することを定義している。

中間関数の記述

function 記述子にて declaration セクションには登場しない中間関数を定義できる。これは系全体で定義された新たな配列と考えることができる。ただし、これは記述を補助するのみであり、実際にこの関数を示す配列をシミュレーション実行時に確保する必要があるか否かは処理系によって自動判定される。図の例では関数 k,l ともに一時変数であり、後に代入されて消滅するため実際にメモリは確保されない。

連立方程式の求解

記述子 equation にて連立方程式の解を求めることを要求する。equation pss(p) のうち、pss はこの処理に付ける

名前であり、それに続き p について解くことを意味している。

陽解法での前進

記述子 forward は陽解法にて前進することを要求する。

```
forward n(p)[t] { s }
```

は、変数（関数）p の t 軸方向について、各位置での偏微分係数が s であることを意味する。

3.6 境界条件入力および他モデルとの接続方法の検討

提案ソフトウェアにおけるシミュレーション同士の接続は、主に弱連成を対象とする。シミュレーション同士の接続には強連成と弱連成があり、強連成はモデル構築時に複数の現象等を併せて定式化し単一の問題（多くは連立方程式）に帰着させる方法、弱連成は一方のシミュレーション結果を交換して他方の入力とする方法である。計算精度や計算機プログラムの単純さでは強連成のメリットが大きいが、弱連成の方が汎用性が高い。様々な構成要素が登場し得る設備シミュレーションの観点からは、まずは汎用性を目指すことが妥当と判断している。

弱連成の場合、一方のシミュレーションの結果を他方に入力することになるので、弱連成の簡易な記述方式を検討することと、初期値や境界値を与える方法を検討することには多くの共通性がある。そこで、ここでは初期値や境界値を与える方法と、シミュレーション結果を効率的に収集する方法をそれぞれ検討する。前者のみを利用すれば通常のシミュレーションの初期値、境界値の指定、後者と併せて利用すれば弱連成のサポート機構となる。また、可視化システム等への入力として後者のみを利用する可能性も考えられる。

```
initial
  u = (0.01, 0.01)
  constant_grad(p, x)
boundary
  p@B1_x = 100, p@B2_x = 200
end
```

図 3 初期・境界条件の設定例

図 3 に初期値、境界値の入力方法の例を示す。モデル記述同様に検討段階であるが、重要な点は入力する空間データに名前付けを行う点である。例では B1 または B2 と名付けられた境界面の x 方向の値を境界条件として設定している。通常、メッシュ等の空間データはその形状を表現する数値が羅列されるのみであるが、本来、メッシュ生成元となる CAD などのデータには、変や面に名前を与えていることが多い（たとえ作成者が与えていなくても内部的に名前が割り当てられる）。提案ソフトウェアでは、メッシュなどの空間データ作成時にこの名前データを引き継ぐようにし、その名前を元に境界条件、初期条件を記述することとする。これにより可読性が高く、形状変更にも容易に対応でき、誤りの少ない条件設定が可能になることが期待される。

シミュレーション結果の取得についてもアイディアは同じであり、空間情報に含まれる名前に対応する場所の数値を抽出する記述を可能とする。ただし、３次元シミュレーションを１次元モデルに接続する場合など、複数の値を合計したり平均したりする必要があるため、その処理も同時に記述できるよう、言語を設計中である。

4. 実例への適用

提案ソフトウェアは、実例として著者らが所属する理化学研究所計算科学研究センターにて運用中の京コンピュータの冷却系に適用しながら開発を進めている。京コンピュータの冷却系は水冷と空冷のハイブリッドシステムであり、冷却水の温度が重要な運用パラメータとなっている [11], [12]。

この冷却系をシミュレーションするにあたり、パイプを流れる冷却水や熱交換器など、多くの構成要素は機械自体を精緻にシミュレーションで再現する必要はなく、全体的な熱の出入りを１次元でモデル化すれば十分であると考えられる。一方、風の影響を受ける冷却塔周辺や計算機の一部部品を空冷している計算機室内部は、３次元の熱流体シミュレーションを実施して、各機器が置かれた状況を正確に再現する必要があると見込んでいる。

このようなシミュレーションを構築するためには、全体的な系を Modelica を用いて１次元でモデリングし、それと流体シミュレーションをそれに接続する。この流体シミュレーションの構築と Modelica との接続部分において、提案ソフトウェアが活躍することを目指している。

5. 関連研究

簡易なモデル記述を目指した言語として、本稿ですでに紹介している Modelica[1], [6] がある。Modelica には二つの特徴がある。微分代数方程式を記述するのみでシミュレーションが構築できること、および、微分代数方程式で記述された部品を組み合わせが容易に記述できる点である。Modelica は１次元モデリングツールとして広く利用されており、実用性も高い。

Modelica は時間微分のみを含む微分代数方程式を対象とするが、Modelica を偏微分方程式に拡張する研究も行われている [4], [9], [10]。これらの手法は、偏微分方程式を空間的に離散化する部分を手動で行い時間微分のみを残した連立微分方程式を Modelica に解かせる手法、あるいは Modelica 発祥のシミュレーション間接続方式である FMI [5] を用いて微分代数方程式によるモデルと偏微分方程式によるモデルを組み合わせるものである。前者は偏微

分方程式の解法の一部のみに対応することになり、解くことのできる問題を制限する。また、偏微分方程式の解法に関わる最新の研究成果を取り入れるのが難しい。後者は偏微分方程式部分に関しては、従来通りモデル作成者による低レベル言語でのシミュレーション実装を要求することになり、簡易なモデル記述を目指す本稿の目標とは相容れない。Modelica は 1 次元モデリングを目的とした言語であるため、空間の情報を受け取る概念がない。これは Modelica の長所であり、空間情報を与える必要がないことが 1 次元モデリングの利点のひとつである。この Modelica を、空間方向の微分を含んだ偏微分方程式に拡張しようとすること自体、根本的な設計方針に矛盾を含むものであり、我々はこのような無理な拡張よりも自然な形で空間情報を受け取るインタフェースを設計する方が合理的と考えている。

Diffpack[2], [3] は偏微分方程式によるモデル記述のための C++言語を用いたフレームワークである。偏微分方程式でモデル化される対象の簡易な記述という点に限って見れば本稿の提案と共通する目的を持つ。ただし、提案ソフトウェアに関しては、本稿では最初の例として偏微分方程式に限った例を議論したものの、設計上は、空間、法則、条件、制御の 4 入力を基に状態変化を再現するモデル全体を対象としており、モデル化対象と抽象化レベルが Diffpack とは異なる。また Diffpack は C++を用いたフレームワークであるため、C++言語の性質に強く影響を受ける。特に、C++は手続き型言語であるため、ある準備された基本的なクラスを継承して自らのモデルを実装する際には、親クラスの構造や処理を理解する必要があり学習コストが高い。本稿で提案した言語は宣言的な記述から成り立っており、プログラム中の各式は変数と変数の関係を表現するのみである。これは実行時にシミュレータが持つ状態に依存せずに読むことができる点で可読性が高く、ツールに習熟するまでの学習コスト低減に貢献する。

Formura[7] は偏微分方程式の離散化式を数式に近い直観的な言語で記述することを可能とする言語処理系である。本処理系はこの離散化式を受け取り、実行可能なシミュレーションコードを生成する。特に、京コンピュータのような超大規模計算機の性能を発揮できる高速なコードを生成することを特徴としている。一方、利用者の要求が Formura が扱うことのできる範囲を超えた場合、利用者は新たにプログラムを書き直すか Formura 処理系自体を改造する必要があり、いずれもコストが高い。本稿で議論した提案ソフトウェアはエンドユーザーとシミュレーション作成者を分離し、後者にも汎用性の高い簡易な記述方法を提供する点で、特定用途特化型の DSL を補完するものである。

6. 結論

本稿では産業設備のシミュレーションを目的として、シミュレーションモデルを低コストで入力するためのソフトウェアを設計した。産業設備が様々な構成要素を持ち、必要なモデル化手法を限定することが難しい現実に即し、「空間」「法則」「条件」「制御」の 4 種類の入力を基に時間発展を追う仕組み全体をターゲットとする抽象化を行った点が特徴である。本稿では特に偏微分方程式によるモデル化に注目し、モデル記述言語およびモデル間のデータ交換の表現方法を検討した。本提案は初期段階であり、今後、より多くの現実的な問題に即した言語使用の拡張と、処理系の実装を進めていく予定である。

参考文献

[1] Elmqvist, H.: A Structured Model Language for Large Continuous Systems, PhD Thesis, Department of Automatic Control, Lund University, Sweden (1978).
[2] inuTech: Diffpack web page, `http://diffpack.de/`. [Online; accessed 07-July-2018].
[3] Langtangen, H. P.: *Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, Springer Berlin Heidelberg (2003).
[4] Li, Z., Zhang, H. and Zheng, L.: Description of PDE Models in Modelica, *2008 International Symposium on Computer Science and Computational Technology*, Vol. 1, pp. 809–813 (online), DOI: 10.1109/ISCSCT.2008.347 (2008).
[5] Modelica Association: Function Mockup Interface, `https://fmi-standard.org/`. [Online; accessed 03-July-2018].
[6] Modelica Association: Modelica and the Modelica Association, `https://www.modelica.org/`. [Online; accessed 06-June-2018].
[7] Muranushi, T., Nishizawa, S., Tomita, H., Nitadori, K., Iwasawa, M., Maruyama, Y., Yashiro, H., Nakamura, Y., Hotta, H., Makino, J., Hosono, N. and Inoue, H.: Automatic Generation of Efficient Codes from Mathematical Descriptions of Stencil Computation, *Proceedings of the 5th International Workshop on Functional High-Performance Computing*, FHPC 2016, pp. 17–22 (2016).
[8] OpenCFD Ltd.: OpenFOAM: Official home of The Open Source Computational Fluid Dynamics (CFD) Toolbox, `https://www.openfoam.com/`. [Online; accessed 03-July-2018].
[9] Saldamli, L.: PDEModelica – A High-Level Language for Modeling with Partial Differential Equations, PhD Thesis, Department of Computer and Information Science, Linköpings universitet (2002).
[10] Stavåker, K., Fritzson, P., Stavaker, K., Fritzson, P., Song, C., Wlotzka, M., Heuveline, V., Song, C., Wlotzka, M. and Heuveline, V.: Pde Modeling with Modelica via Fmi Import of Hiflow 3 C + + Components with Parallel Multi-core Simulations (2014).
[11] 塚本俊之，草野義博，平井慶太，　魏杰：「京」にみる冷却技術，伝熱: 特集スーパーコンピュータと伝熱，Vol. 52, No. 220, pp. 21–26 (2013).
[12] 関口芳弘：スーパーコンピュータ「京」を支える建築・設備技術，富士通，Vol. 63, No. 3, pp. 247–253 (2012).