第1章　MATLAB導入の前に

1.1　はじめに

本章では，まず最初に，組み込みソフトウェアの開発過程において，MATLABが果たす役割について紹介します．次にMATLABと組み込みシステムとの関連について解説します．

また，MATLAB本体とMATLABのプロダクト・ファミリの概要を示します．

1.2　二つのスキット

組み込みプログラムの開発において，MATLABが果たす役割を説明するために，ここで，二つのスキットを示します．このスキットは，私が頭の中で創作したものなので，実在するものとの関連は一切ないことを，あらかじめ断っておきます．

スキットA

A社とB社はプレス加工機を設計，製造する会社です．良い意味の競合によって，両社ともに，順調に業績を伸ばしています(**図1-1**)．

さて，この国では，振動に関する新しい法律が施行されることになりました．国内において機械装置が発生する振動のパワー・スペクトラムの500 Hz以上の成分が総パワーの20％を超えると，その工場に対して課徴金を課すというものです．

A，B両社が自社の加工機に対して振動の計測を行ったところ，通常の加工時においては，新しい法律の基準値を超えることはないが，板厚40ミリを超える大型のものを加工する際に，振動の振幅値が基準を超える恐れがあることがわかりました．

対策として，これまでの制御装置に新しい機能を付け加える必要があります．

必要な新機能は，プレス加工機に加速度センサを取り付け，センサからの出力波形を2重積分して振幅を算出し，その振幅の自己相関関数を計算し，それを高速フーリエ変換してパワー・スペクトラムを計算し，スペクトラムの500 Hzを超える部分の割合を集計して，その数値が基準値を超えそうになったときプレスの送り速度を低速モードにシフトするというものです．

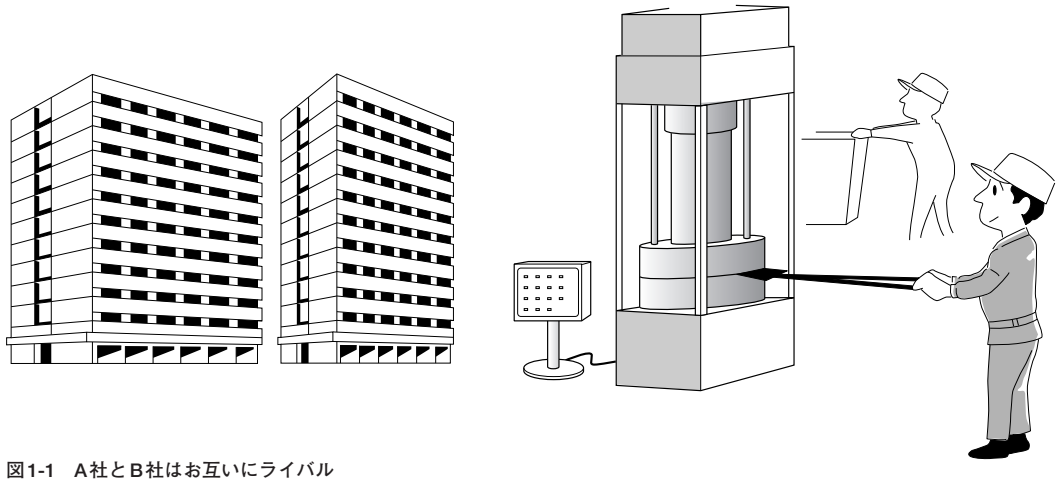

図1-1　A社とB社はお互いにライバル

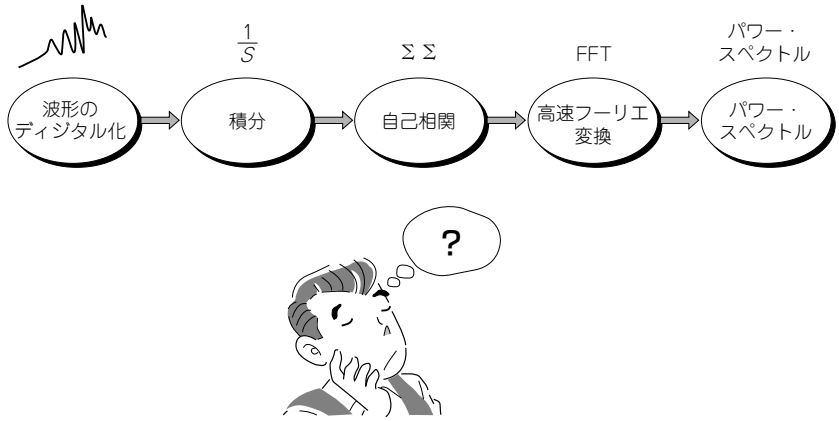

図1-2　複雑な計算式

A社には，大学の応用数理工学科を卒業した若手エンジニアがいました．a君です．新しい法律に適合させるために追加する機能の設計は，当然のようにa君の担当になりました．

a君は，大学時代に勉強した教科書を開いて，まず，数値積分のアルゴリズムを調べながら機能の処理方法を考えました(**図1-2**)．教科書には，いくつかの数値積分のアルゴリズムが述べられています．どれが良いかわからないので，実際に加速度センサのデータを使用して，コンピュータを使って計算したところ，この場合シンプソンの数値積分法が良いことがわかり，これを採用することにしました．

続いて，自己相関関数の計算法を勉強して，それから高速フーリエ変換のアルゴリズムを理解し，フローチャートを書いて，C言語のプログラムを作りました．デバッグによってバグを除去して，プログラムを完成しました．

A社の制御コンピュータは，RISC型のCPUを採用しているので，浮動小数点の演算ができません．

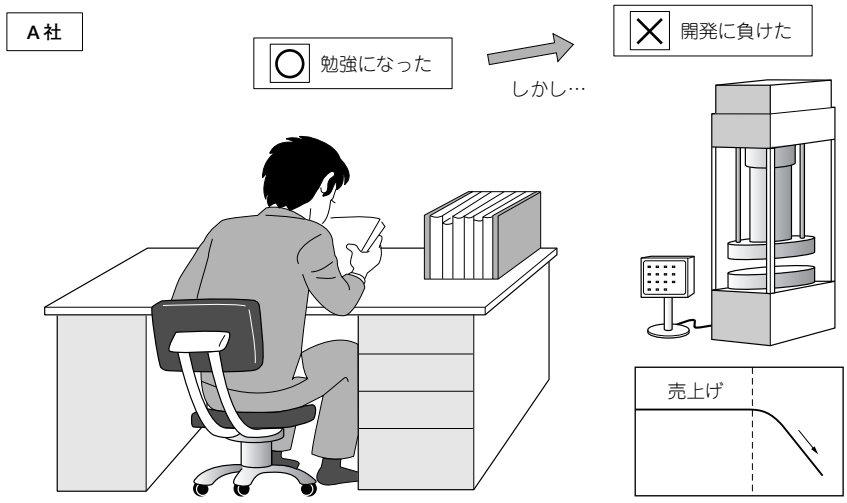

図1-3　勉強になったが，開発には負けた

プログラムを整数型の演算に書き直して，実機に実装し，テストを繰り返しました．サンプル周波数の調整と高速フーリエ変換を行う際のウィンドウの幅の調整も繰り返し実行しました．

新プレス機の制御装置の設計は，a君にとってとても良い勉強になりました．複雑なアルゴリズムを実装することによって数値計算の意味を体得することができ，また，実装するためのテクニックを身につけることができました．

しかし，A社の新プレス機の開発は，新しい法律の施行から約6ヵ月間遅れたため，その間国内の工場へ出荷することができず，A社のプレス機のシェアは大きく落ち込みました(**図1-3**)．

さて，それに対してB社は，社内に数学を得意とするエンジニアはいなかったのですが，大学の情報工学科を卒業して，社内ネットワークのメンテナンスをしている社員がいました．b君です．

残念なことに，b君は数学が得意ではなかったので，努力はしたのですが，高速フーリエ変換などの計算式をよく理解できませんでした．

しかし，研究室に在籍したときに，教授がMATLABを使って組み込みプログラムの開発ができる，と言っていたことを記憶していたので，新プレス機の制御プログラムを開発するのにMATLABを使ってみることにしました．

パソコン(PC)にMATLABをインストールして，Simulinkを立ち上げ，Signal Processing Blocksetのライブラリから数値積分，自己相関，FFTのブロックをドラッグして制御のシミュレーション・モデルを組みます．データ・ソースとして，実測データを記録したファイルを使い，結果はスコープ・ブロックを使ってグラフ表示します．

Simulinkの実行開始ボタンをクリックするとシミュレーションが始まり，時々刻々の状態がスコープに表示されます．ここで制御系のゲインをチューニングして，シミュレーションにおける最適構造を求めます．

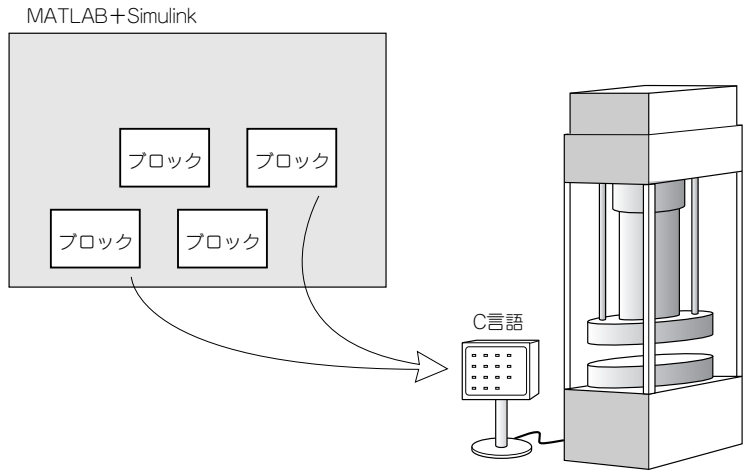

図1-4　ブロックを集めてビルド

Real-Time Workshopによってブロック図をビルドすると，制御系のアルゴリズムはC言語のプログラムに変換されます(**図1-4**)．

このCプログラムを検討して，必要なところを加筆，訂正し，ターゲット機のコンパイラにかけ，最終的な実行ファイルを作ります．

このプログラムを制御用のコンピュータ実機にロードして，実機テストをします．結果は，新しい法律をクリアしていました．

B社は新プレス機を新しい法律が施行される前に新聞発表し，そして順調に出荷を始めました．A社が対応に手間取っている間に，B社のプレス機は市場の大半を独占し，B社の利益は倍増しました．

b君は，数学の学力はいまいちだったかもしれませんが，その功績が認められ，B社の将来を担う技術者と考えられるまでになりました．

スキットB

C社は，乗用車を製造する巨大企業です．世界市場において，社運をかけて商戦を展開しています．食うか食われるかの戦いです．

乗用車設計のポイントは軽量化です．自動車が軽ければ加速性能は上がり，かつガソリン消費量は低減します．1ガロン当りの走行距離は伸び，環境に対する悪影響は減少します．一石二鳥です．

しかし一方で，乗用車ボディの鋼板を薄くすると，衝突時の安全性は低下します．人が乗る輸送機械である以上，安全性を無視してボディの重量を削ることはできません．

C社の若手エンジニア，c君は，新たな提案をしました．

自動車の構造に関与しない制御部に使用する鋳鉄のコントロール・ロッドを，細くしなやかなワイヤに置き換えるという提案です(**図1-5**)．

C社の技術部門は，蜂の巣を突いたような状態になりました．

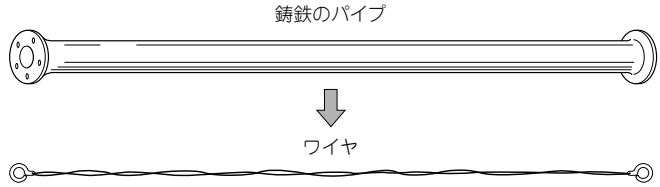

図1-5　鋳鉄のパイプをワイヤで置き換える

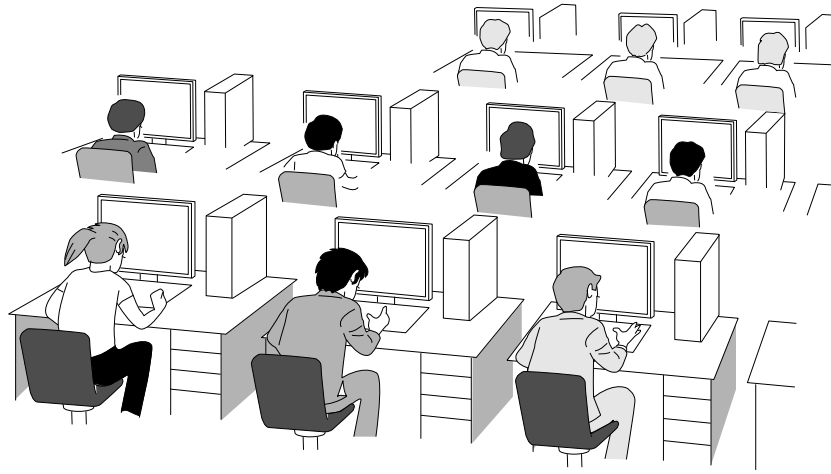

図1-6　まず，シミュレーションで確かめよう

これまでの乗用車の歴史において，コントロール用の鋳鉄のロッドをワイヤで置き換えるという提案をした人はいません．

そんなことをしたら，重量は減るかもしれないが乗用車の安全は吹き飛んでしまう，事故時にワイヤが切れてエンジンが暴走したらどうするのだ，事故による訴訟が多発する可能性がある，……いろいろな意見が出されました．

C社の技術担当の重役は考えました．保身の立場から言えば，提案を退ければよい，この提案はあまりにも危険が多すぎる．しかし，もし成功して特許が成立すれば，ひょっとして世界のトップに躍り出る可能性もある，そういう新芽を摘み取りたくない．

考えに考えた末に，重役は決断しました．極秘のプロジェクト・チームを結成し，MATLABのネットワーク・ライセンスを大量に購入し，コンピュータ・シミュレーションによってあらゆる状況をコンピュータ上に構築して徹底的に検討を行う，ただし，絶対に実機の試作を行ってはならないというのです(**図1-6**)．

MATLAB，Simulink，SimDriveline，SimMechanicsなどを使って，コンピュータ内に乗用車モデルを構築し，力学に基づいたシミュレーション実験を行い，あらゆる状況を検討するのです．仮想的な実験なので，実害はありません．

こうして，C社の社運を賭けたプロジェクト部隊が発足しました．このプロジェクトは，現在進行中です．

■ 1.3　MATLABの誕生

1980年代に，米国東海岸マサチューセッツ州にあるThe MathWorks社は，MATLABの発売を開始しました．MATはマトリックス(matrix)から，LABはラボラトリ(laboratory)から切り取り，両者を貼り付けてMATLABという名前が生まれました．

この名前が示すとおり，MATLABは線形代数学の諸問題を処理するためのラボラトリ(計算環境)としてスタートしました．

当時，私は大阪大学工学部の教授で，研究室の学生は，計算処理はDEC社のUNIXワークステーション，グラフィックスはSGI社のIndigoを使って処理を行うという状況でした．PCは購入しましたが，研究活動において使用することはなく，複雑な計算は大学の計算センタのIBM7090や360などを使って処理しました．プログラムはFORTRANとパンチ・カードを使っていました(**図1-7**)．

MATLABは，マトリックスをカラム・ワイズ(縦切り)に処理します．こういうところを見ると，「FORTRANの処理も同じカラム・ワイズだったな」などと思い出して，過ぎ去った昔が懐かしくなります．

ところで，この20年間において，PCは大きく進歩しました．UNIXワークステーションのDEC社はすでに舞台から消え去り，科学計算の分野においてIBMの大型コンピュータを使う機会はありません．PCは，大学，企業の研究所，開発部門などにおける計算処理の主たる担い手となり，わずかにスーパー・コンピュータが生き残っている状況です．

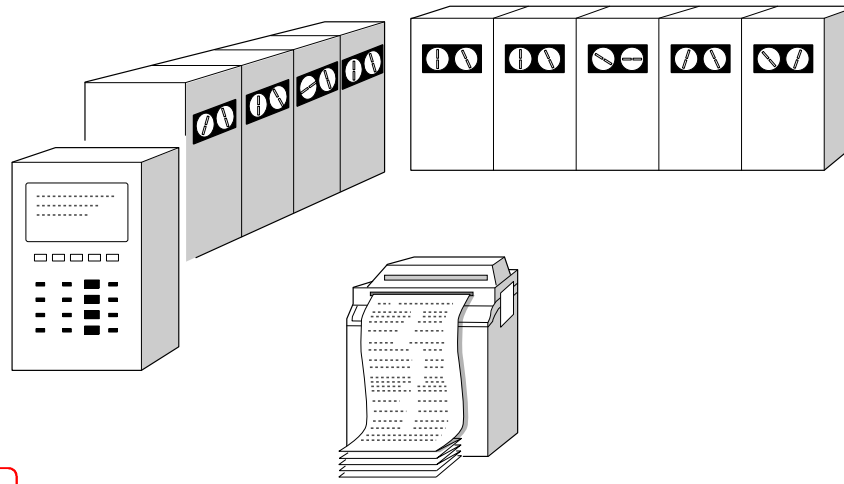

図1-7　パンチ・カードの時代に生まれたMATLAB

このような状況の変化に応じてMATLABも進化しました．MATLABの実行環境は，UNIXワークステーションからPCへ移行し，言語もFORTRANからCに変わりました(**図1-8**)．

もっと重要なことは，MATLABが進化したことです．マトリックスの解法に加えて，常微分方程式，偏微分方程式，さらに統計計算，データベース処理，社会モデルの構築，バイオメカニックスにおけるDNA合成などの数値解法が取り込まれ，MATLABは科学技術計算の広大な分野をカバーする巨大システムに成長しました．

The MathWorks社はこれらのソフトウェアを一つの大きなシステムとして一括販売するのではなく，プログラムを分野ごとに細かく分割して販売しています．これらの製品をプロダクト・ファミリと呼びます．

ユーザは，目的に応じて必要なプロダクト・ファミリ(コンポーネント)を購入し，目的に合致するシステムを構築し，それによって処理を行うことになります(**図1-9**)．

参考のために，The MathWorks社が現在販売しているプロダクト・ファミリのリストを，付録に記

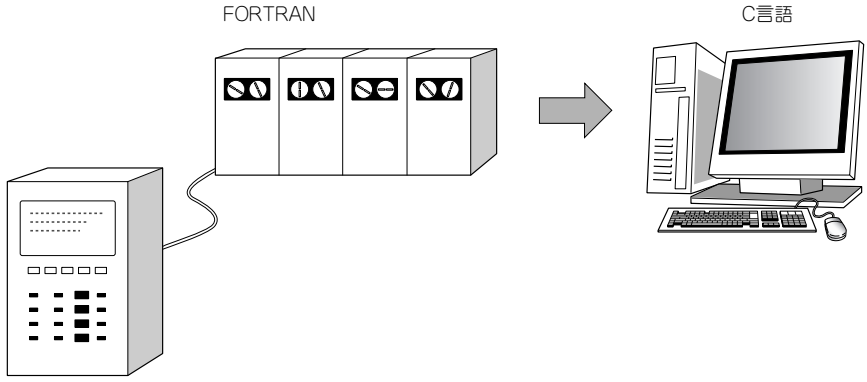

図1-8　PCの時代へ移行

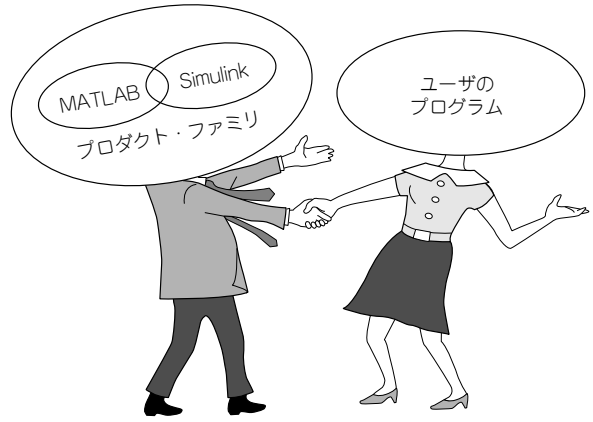

図1-9　MATLAB，Simulink，プロダクト・ファミリの関係

載しました．

MATLAB本体はオープン・ソースではありませんが，本体以外の一部のソースはWebで見ることができます．

MATLABに関するドキュメントはほとんどが英文ですが，総ページ数は20,000ページを超えます．MATLABを購入しなくても，これらの資料は閲覧やダウンロードが可能です．

必要ならば，ユーザのプログラムを組み込んでMATLABの関数と同等のレベルで実行することが可能です．しかも，その組み込み方法は公開されています．ここが，もっとも重要なポイントです．

■ 1.4　MATLABの構造

MATLABは，The MathWorks社が発売するプロダクト・ファミリの基本コンポーネントです．すなわち，コンテナの役割を果たし，プロダクト・ファミリは，すべてMATLABの上で動作します．MATLABがなければ，プロダクト・ファミリは動作しません．

ユーザは，MATLABを購入して，PCにインストールする必要があります．

Windowsのプログラムを開発する際に，例えば，Microsoft社のVisual Studioを使います．Visual Studioは，Windows OS上で動作し，プログラム開発を一括管理します．これをIDE（Integrated Development Environment：統合開発環境）といいます．

この用語を使うと，MATLABは，組み込みプログラム開発のIDE，統合開発環境と言えます．MATLABを立ち上げるということは，WindowsにおいてVisual Studioを立ち上げることと同等です（**図1-10**）．

MATLABの操作は，UNIXのシェル，DOSのコマンド・プロンプトと同じように，キーボードからコマンドを打ち込みます．この点が，Visual Studioのマウス操作とは異なります（**図1-11**）．

Visual StudioなどのIDEからMATLABへ移行すると，一瞬，WindowsからDOSへ逆戻りしたよう

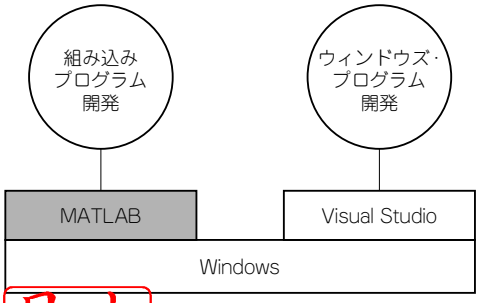

図1-10　MATLABとアプリケーション

図1-11　コマンド入力による操作

な感じを受けるかもしれません．しかし，コマンド入力は，使っているうちにすぐに慣れます．

MATLABは，現在およそ800の基本関数をもっています．MATLABスタート時の関数は，主として，マトリックス演算に関するものでしたが，そのあとに，常微分方程式と偏微分方程式の解法に関する関数が仲間入りし，同時に，計算結果のグラフ表示の関数が加えられて，ほぼ，現在の形ができあがりました(**図1-12**)．

もし，MATLABがスタート時のマトリックス演算だけのツールとして留まっていたら，今日のようなMATLABの展開はなかったかもしれません．

実際に，その程度の計算であれば，お金を出してMATLABを購入しなくても，インターネットからフリーのソフトウェアをダウンロードして，費用をかけずに計算処理することができます．

The MathWorks社は，精力的にアプリケーションの範囲を拡張してきました．例えば，いまここに制御系のゲインを決定する問題があったとします．もちろん，この制御系の動特性を常微分方程式で記述して，MATLABを使って解くことはできます．しかし，制御系の設計において，技術者は通常ラプラス変換を用いて動特性を伝達関数に変換し，その伝達関数を使って設計を行います．

The MathWorks 社は，ラプラス変換の微分演算子を使って，信号を処理する関数群を開発し，その関数群を直接MATLABに組み込むのではなく，本体から切り離して別売のソフトウェアとして販売しています．これらをツール・ボックス(Toolbox)といいます(**図1-13**)．制御系の設計に使用するツール・ボックスは，制御システム・ツール・ボックス(Control System Toolbox)と呼ばれています．

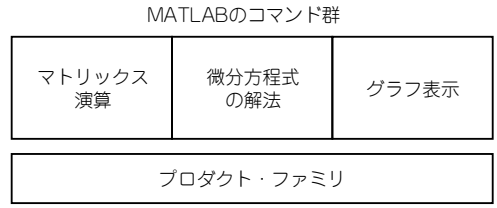

図1-12　MATLABコマンドのカテゴリ

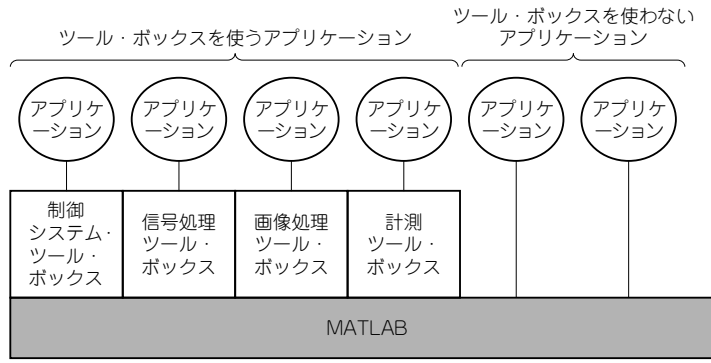

図1-13　MATLABとツール・ボックス

このほかに，フィルタ設計のツール・ボックス(Filter Design Toolbox)，システム同定のツール・ボックス(System Identification Toolbox)などがあります．詳しくは，付録を参考にしてください．

機器の設計において，概念を具象化する手段として，図面を用います．例えば，制御系の設計ではブロック線図を使用します．ブロック線図は，制御分野におけるUMLチャートの役割を果たします．

当然，MATLABに対して，**制御系をブロック線図で記述し処理したい**という要求が発生します．

この要求に答えるために，Simulinkが開発されました．Simulinkを使うと，対象のシステムをブロック図によって記述し，しかもマウスのクリックでシミュレーションを実行することができます．

Simulinkは，形態から言えば，**図1-14**に示すように，MATLABに対する一つのアドインですが，仕事の内容という面から表現すると，**図1-15**に示すように，MATLABと同格のパートナ的なソフトウェアと言えます．

Simulinkでのブロック図の制作は，Microsoft社のVisioなどと同じような要領で，あらかじめ用意しておいたテンプレートをエディタの画面にドラッグし，結線して，ブロック図を作成することができます．このテンプレートをブロック(block)と呼びます．

ここで，どのような機能のブロックがSimulinkに用意されているかが問題になります．ここでも，The MathWorks社は，MATLABと同じ原則を採用しています．例えば，信号処理に適したブロック群を開発して，これらのブロックをひとまとめにして信号処理ブロック・セット(Signal Processing Blockset)，航空機の設計に必要なブロックを開発して，航空宇宙ブロック・セット(Aerospace Blockset)として販売しています．

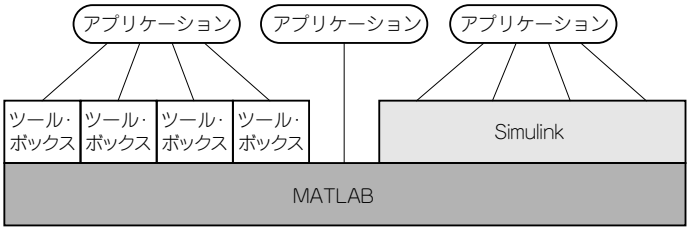

図1-14　SimulinkはMATLABのアドイン

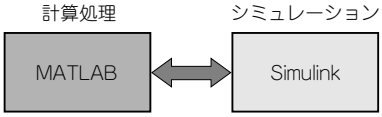

図1-15　MATLABとSimulinkはパートナ

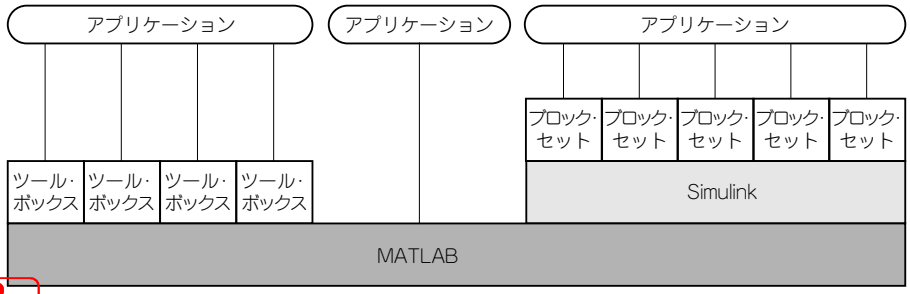

図1-16　MATLABの体系図

The MathWorks社はいろいろなブロック・セットを用意していますが，当然，エンジニアリングの全分野において発生する問題のすべてに対応することはできません．それは，およそ不可能なことです．

この対策として，C言語やC++，FORTRAN，Adaなどを使ってユーザが独自のブロックを構築し，それをSimulinkに組み込むことが可能となっています．

このブロックをカスタム・ブロック(custom blocks)といいます．

MATLABの構造は，最終的に**図1-16**となります．これがMATLABの横断的な構造です．

■ 1.5　組み込みプログラムの開発

組み込みプログラムを開発するために，MATLABのどのプロダクト・ファミリを入手すればよいか，ということについて説明します．

まず，組み込みプログラムを開発する過程を大きく二つに分けます．

第1の過程は，モデルを組み上げてシミュレーションを行うまでの過程です．これをモデリング過程(modeling process)と呼びます．

第2の過程は，シミュレーションの結果を組み込み系にダウンロードして，実機において検証を行う過程です．これを実装過程(implementing process)と呼びます．

組み込みプログラミングにおいて，この二つの過程を繰り返すことによって開発を進めます．これを基本サイクルと呼びます(**図1-17**)．

まず，モデリング過程について解説します．

基本のプロダクト・ファミリとしてMATLABが必要です．前節で述べたとおり，MATLABは絶対に必要です．

次に，ブロックを組み合わせて対象をモデル化するために，Simulinkが必要です．これがないとモデル図が描けません．

組み込み系においてシーケンス動作が必要な場合，このシーケンス操作を状態遷移図として表現するために，Stateflowが必要です．ここでシーケンス動作というのは，フローチャートでいえば分岐動作のこと，プログラムでいえばIF文のことです．

この三つのプロダクト・ファミリがあれば，とりあえず，組み込み系のシミュレーションを行うことができます(**図1-18**)．

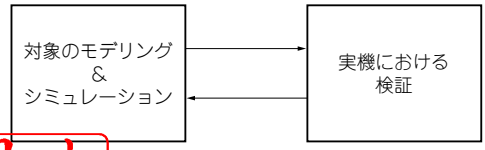

図1-17　組み込みプログラム開発のサイクル

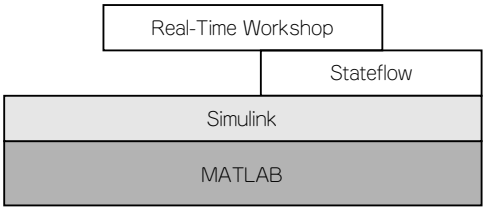

図1-18　組み込み開発の基本プロダクト

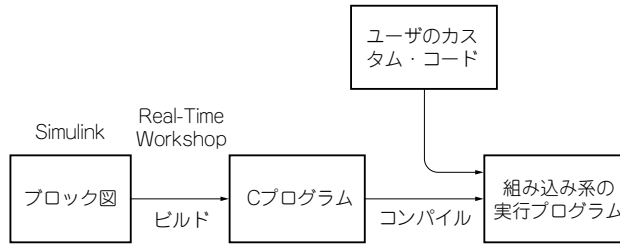

図1-19　最小必要プロダクト・ファミリ

もちろん，前述のように，もし制御系のブロック線図を描くのであればSimulink Control Design，信号処理をするのであればSignal Processing Blocksetというプロダクト・ファミリがあれば便利だということになります．

次に，実装過程に入ります．

まず最初に，作成したブロック図をC言語のプログラムに変換するソフトウェアが必要です．これには，Real-Time Workshopというプロダクト・ファミリを用います．Real-Time Workshopは，シミュレーションにより検証済みのブロック図をC言語に変換します．この過程をビルドといいます(**図1-19**)．

細かいことを言えば，C++，Ada，Fortranにも変換することができますが，本書ではCに変換する場合についてのみ述べます．

以上が，MATLABによる組み込みプログラム開発の前半部分です．

もし，費用を最小限に抑えて組み込みプログラムを自動生成し，どのようなCプログラムができるか検討したい，というのであれば，

MATLAB
Simulink
Real-Time Workshop

以上の3本だけで，とりあえず開発をスタートすることが可能です．

状態遷移図を組み込むことが必要ならば，オプションとして，

```
Stateflow
Stateflow Coder
```

があればよいでしょう．Stateflowのブロック図をCのプログラムにビルドするのであれば，Stateflow Coderが必要です．

また，組み込みプログラムをコンパクトに作りたいのであれば，

```
Real-Time Workshop Embedded Coder
```

があると便利です．

もちろんこれは，試験的にプログラムを自動生成することができるというだけのことであって，これだけで実務において効率よい開発ができる，ということを言っているわけではありません．この点は，とくに注意してください．

■ 1.6　実装過程の問題

組み込みプログラムの開発において，Cのプログラムができればそれで終わり，というものではありません．

むしろ問題はそこから始まる，といったほうが正しいと思います．

よく知られているように，組み込みターゲットにプログラムを実装する際に，プログラムの不変部はROMへ，可変部はRAMへ振り分けます．多くの場合，メモリ容量は厳しく制限されるので，プログラムの冗長な部分を削り取り，プログラム全体をスリムにする必要があります．

ターゲットに組み込むOSによって，割り込み処理などが変化するので，アプリケーション・プログラムの形式は大きく変化します．

DOSのように単機能のOSの場合は，プログラム内に割り込みルーチンの初期化やタイマ・スタートなどの手順を書き込む必要があります．

RTOSの場合は，そのOSが要求する形式にプログラムをカスタマイズする必要があります．

ターゲットのコンピュータがA-Dコンバータや D-A コンバータなどのアナログ変換回路を使用すれば，そのためのドライバ・ルーチンが必要です．

実装のターゲットにおいて，無限の選択枝がある以上，それらのすべてを満足するプログラムを自動生成することなどできるわけがありません．ここに，いろいろな意味で手作業が必要になります．

The MathWorks 社は，市販されているシングル・ボード・コンピュータやDSPボードに対して，直接ダウンロード実装できる支援プログラムを製作し販売しています．上流と下流を一本道で結ぶために努力しているわけです．

しかし私が調べたところでは，それらの直接ダウンロード実装可能なシステムは，現在，市場にでているボードのごく一部であって，ほとんどのものは直接には実装できない状態になっています．どこかで手作業が必要になります．

とくに，The MathWorks社の開発部隊は米国東海岸のマサチューセッツ州に集結しているので，米国製のボードがおもな対象になっていて，わが国のボードは一部を除いてほとんどのものが対象から外れています．

この状況は，組み込みシステム開発においてはあたりまえのことであって，なげいたり悲観したりすることではありません．

MATLABに対して，必要ならばユーザのプログラムを組み込んでMATLABの関数と同等に実行することは可能であり，しかもその組み込み方法は公開されています．

また，スクリプトを書けば，Real-Time Workshopのビルド過程を制御し，希望する形式でCプログ

ラムを生成することができます．

ここまでできれば，自由自在です．しかし，自由とは選択肢が多いということであり，「難解」の別の姿ともいえます．

MATLABによって組み込みプログラムを自動生成して，ターゲットを動かすまでの技術を身につけることは，容易なことではありません．しかし，困難があるからこそ，それに挑戦する価値があるのであり，それを体得したときに得られる利益も大きいのです．

第2章　MATLABの基本フレームワーク

2.1　はじめに

本章では，組み込みプログラムの開発において，中心的な役割を果たすMATLABの基本フレームワークについて解説します．

このMATLABの基本フレームワークはSimulinkに受け継がれ，そして，組み込みプログラム自身に継承されます．要するに，ルーツから勉強を始めるということです．

これまでにMATLABを使った経験があり，MATLABの基本フレームワークは十分に身についているという方は，本章をスキップして第3章へ進んでください．

2.2　MATLABのスタート

それでは，MATLABを操作するところから始めます．

ここでは，MATLAB 7.0.1（R 14）SP1，Windows版を使います．

本書において取り扱う機能は，MATLABの基本的な機能に限るので，無理にバージョンを合わせる必要はありません．皆さんの手元に異なるバージョンのMATLABがある場合は，それを使ってください．

ただし，バージョンが異なると，ユーザ・インターフェース（ダイアログなど）の画面が異なる場合があるので，その点は注意してください．

もし，皆さんの手元にMATLABがなければ，30日間有効の試用版を使ってみることができます．入手方法は，サイバネットシステムのWebサイト（http://www.cybernet.co.jp/matlab/product/beta.shtml）を参照してください．

私が調べた範囲内では，MATLABの日本語化作業は現在進行形の状態です．ユーザ・インターフェースの日本語化とドキュメントの日本語化が並行して進められています．

しかし，それら作業の足並みは，必ずしもそろっているとは言えません．そのために，ダイアログなどのユーザ・インターフェースは英語表示なのに，それに対応するドキュメントの説明は日本語，あるいは逆にユーザ・インターフェースは日本語表示で，それに対応するドキュメントの説明は英語とい