DEVELOPMENT OF SCRIPT INTERPRETER FOR BEAM COMMISSIONING AT J-PARC LINAC

Hiroshi Ikeda[1A)], Hiroyuki Sako[B)]
[A)] Visible Information Center, Inc.
440 Muramatsu, Tokai, Naka, Ibaraki 319-1112, Japan
[B)] Japan Atomic Energy Agency
2-4 Shirakata-Shirane, Tokai, Naka, Ibaraki 319-1195, Japan

Abstract

We have developed the script interpreter for commissioning of J-PARC and we name the environment using the interpreter JCE, Java Commissioning Environment. In this report we describe briefs about the structure of JCE and some technical information to implement new commands prepared for commissioning.

J-PARCリニアックのビームコミッショニング用 スクリプトインタプリタの開発

１．はじめに

コミッショニングでは、様々な機器の計測データとシミュレーションとの比較から、機器パラメータの較正と制御アルゴリズムの修正を繰り返し行う。このため、この用途に適した特別なソフトウェア開発環境が求められる。

KEKではこの目的のためSAD[1]と呼ばれる開発環境が構築されている。また、J-PARCと同規模の陽子加速器Spallation Neutron Sourceにて軌道計算・制御用ソフトウェアフレームワークXAL[2]が開発されている。我々は、これらの技術をJ-PARCのコミッショニングに適用できるように新規にインタプリタを開発してきており、これを用いた開発環境をJCEと命名している。

本論文では、JCEの構造及びLINACのコミッショニングで使用されたコマンドの技術的情報について記述する。

２．JCEの構造

JCEは、Mathematicaに似た構文を持つスクリプトでプログラムを記述する。これは、数式を宣言的に定義することができる。また、JCEインタプリタは、このような数式の定義から自動的に値を推論する強力な評価エンジンを持つ。

スクリプトは式と呼ばれる単位で記述される。

2.1 式

式は原子式と複合式で表わされる。

原子式は、文字列、数値、シンボルなどで表わされる。文字列は“abc”のようにクォーテーションで囲み、シンボルはアルファベットと数字の組み合わせで表現される。

複合式は式を頭部に１つ、引数に任意個保持した式として再帰的に定義される。これは、カギ括弧([])を用いて、(頭部)[(引数1), (引数2), …] で表わされる。例えば、f[123, a, “abc”] が挙げられる。なお、特定の複合式には略記が容易されている。例えば、式 x=1 は Set[x, 1] の略記である。

2.2 評価

これらの式を副作用のある変形を行うことで、スクリプトの実行とする。この操作を評価と呼びこれを行うモジュールを評価エンジンと呼ぶものとする。

評価は、ボトムアップで再帰的に行われる。この時に行われる操作は、大別してリライティングとコマンドの２種類があり、いずれも特定のシンボルに関連付けられて定義される。リライティングとコマンドをどのように組み合わせて実行を制御するかが、評価エンジンの重要な機能の１つである。

2.3 リライティング

リライティングは、数式の宣言的定義を基に数式を推論する。また、リライティングは、一般的なプログラミング言語における変数-値の関係（代入などの概念）を表す役目も担っている。

例えば、式 x=1 は、式 x から 1 への変形規則の定義を行うSetコマンド(=)として実行され、ここで、x が評価されると 1 へ書き換えられる。

数式の宣言的定義には、特別な原子式としてパターンと呼ばれるものが使われる。これは、シンボルにアンダースコア(_)を語尾に付けたものであり、任意の式を表現する。例えば、式 f[x_] := x+1 を

[1] E-mail: ikeda@vic.co.jp

SetDelayedコマンド(:=)にて実行して書き換え規則を定義した場合、f[2]を評価すると2+1に置き換えられる。

リライティングは、スクリプトの実行中にその変形規則の定義が変化する。一方、コマンドは、あらかじめ決められた静的な変形規則である。

2.4 コマンド

コマンドは、JCEに実装レベルで予め組み込まれた操作である。リライティングは基本的に式の変形を行うだけだが、コマンドは、任意の操作を行うことが可能である。リライティングの変形規則の定義も、Set(:=)等のコマンドによって行われる。

例えば、式 x=1 は、xから1へのリライティングの変形規則の定義を追加するコマンドを実行する。また、1+2 は Plus[1,2]の略記になっており、Plusコマンドは加算を実行する。引数が数値であるため、これは数値3を変形結果とする。

コマンドは、その実行中に式の評価を行うことができ、コマンドと評価が再帰的に織り込まれる。これにより、例えば、Forコマンドによるループコマンドは、内部の式を評価することをコマンドで制御する。また、パラメータの最適化を行うコマンドでは、最適化の対象となる式は、そのコマンド内部で何度も評価されることになる。

JCEにどのようにコマンドが用意されているかが、JCEそのものの能力を決める。また、コマンドの追加の容易さは、JCEの拡張の容易さとなる。一部のコマンドは、評価エンジンやリライティング、式の構築など、中核となるシステムの部分に直接的に結びつき、分離することはできない。しかし、その他のコマンドは、中核と分離して実装される。

3．コマンドの実装

JCEのコマンドは、幾つかに分類されて実装されている。表 1にこれを示す。

表 1　コマンドの分類

種類	内容
core	評価エンジンを構築するにあたり、それを組み込む必要があるコマンド（評価エンジンと切り離せないコマンド）
primitives	基本コマンド
math	基本コマンドのうち特に数学関数
jjframes	GUIを扱うためのコマンド。基本的なUIコンポーネント、KBFrame、プロットなど
ca	JCAを用いたチャネルアクセス
wavearchive	波形記録用の書式を用いたチャネルアクセスおよびファイルとのアクセス
xaloptics	XALの設定ファイルを利用した各種シミュレーション
jparcdb	J-PARC用に開発されたDBへのアクセス

JCEは、SADの基本的なコマンド及び数学関数に関するコマンドはほぼ使用可能となっている。基本的なコマンドとは、代入、四則演算などの数学演算、リスト（配列のようなもの）に対する要素の追加・削除・抽出・要素への関数への適用等、条件式やループ等のフロー制御、ファイルの入出力等、通常のプログラミング言語なら備わっている基本的な動作を命令するためのコマンド群である。数学関数とは、対数や三角関数・逆三角関数、ベッセル関数や複素演算、行列、乱数、近似解やパラメータの最小化・最適化である。UIコンポーネントについても、通常のコンポーネントの他、XYグラフなどのプロットなどが利用できる。

以下に、J-PARCのコミッショニングで用いられたコマンドを実装するにあたり技術的な情報について幾つか記述する。

3.1 GUI

JCEの前身であるSADでは、GUIはTcl/Tkを用いている。一方、JCEではその実装言語であるJavaの標準的なGUIフレームワークとしてSwingを用いている。

両者のUIスレッドの所有権の差異を、以下に述べる独自にイベントループ(2.2)を構築することで吸収している。また、UIコンポーネントの粒度の差異を埋めるため、複数のSwingのコンポーネントを組み合わせて1つのコンポーネントを構築し、これを「オブジェクト化」(2.3)してJCEからアクセス可能にしている。

3.2 イベントループ

処理の簡単化及び効率性のため、JCEの評価はシングルスレッドで行うものとしている（以下、このスレッドをメインスレッドと呼ぶ）。このため、外部システムからのイベントは、メインスレッドに委譲する必要がある。これを実現するため、イベントキュー及びイベントを受け取るためのイベントループの仕組みを構築した。

メインスレッドは外部システムからのイベントを受け取るためには、イベントループに入る必要がある。

イベントキューはスレッドセーフであるため、イベントの供給源を選ばない。このため、イベントを発生させる外部システムはすべて同一のイベントキューを用いるものとすることで、イベントループ中は、これら全てのイベントの発生を検知することができる。

3.3 オブジェクト化

JCEが状態を持つ外部システムと連動する場合、JCE側でそのリソースの管理をする必要がある。JCEでは、これは特定のシンボルと結び付けて管理し、またリライティングを用いて比較的容易にアクセスできるようにする仕組みを取り入れている。

また、JCEの実装言語であるJavaでは、JavaBeansと呼ばれる標準的なコンポーネントの考え方があり、

これを利用して外部システムのラッピングクラスの形式を統一し、コーディングを簡単化している。ただし、Beanの考え方を直接ラッパークラスに適用すると、アクセス可能な範囲に混乱を招くため、Beanとしてアクセス可能なオブジェクトを生成するメソッドを用意するものしている。

3.4 パラメータの最適化

関数の極小値を求めるためのPowellとBrentの手法を組み合わせたロジックを基本に、目標値とのχ^2の極小値を求めることで、パラメータの最適化を行う。

Powellは、多次元から１次元への関数に対し極小値を求めるため、降下するための方向集合として単に単位ベクトルを用いる代わりに互いに共役なベクトルを用いる方法である。これにより、2次導関数が方向によって大きく異なる関数に対しても計算ステップが極端に増加することがなくなる。BrentはPowellの計算中に用いられ、１次元から１次元への関数における極小値を求める。これは、放物線補完と黄金比分割を組み合わせたロジックを持つ。

また、多次元から多次元への関数に対する目標値への最適解を求めるため、特異値分解と数値微分によるニュートン法を組み合わせたロジックも実装済みである（特異値分解によって得られるχ^2-Fitの解空間に対する解を求める）。

3.5 シミュレーション

JCEに用意されている加速器のシミュレーションは、XALの他、SAD, MAD, Trace3Dが用意されている。ここで、SADはJCEの前身となったSADに組み込まれているロジックを用いる。XALは、入力として加速器の機器情報を記述した設定ファイルと、ビーム（計算ロジック）の情報を記述した設定ファイルの２つを用いる。これらのデータは、XALの計算前にJCEに読み込む。これらのデータはコマンド経由で編集することができ、これにより、先のパラメータの最適化コマンドと組み合わせることで、最適な設定値を導き出す。

SAD, MAD, Trace3Dを計算する際の入力データは、XALの計算と同等の条件とするためXALの入力データから自動生成される。なお、コミッショニングではXALを用いている。

3.6 EPICSの制御

EPICSはリアルタイムに機器を制御するために広く使われているライブラリで、加速器機器の制御にも使われている。Javaを用いたEPICSによる通信には、JCAというライブラリを用いる。また、XALではJCAをさらにカプセル化し、特にモニタリングに関して有用な仕組みを用意している。JCEでは、XAL経由で機器にアクセスしている。

４．コミッショニング

以下の機能を持つコマンドを組み合わせ、コミッショニング用スクリプトを作成した。

- GUI
  - 基本UIコンポーネント
  - 複合的なコンポーネント(KBFrame)
  - XYグラフ、波形グラフ、トレンドグラフ
  - オプティクスプロット（加速器機器の表示）
- パラメータの最適化とXALによるシミュレーションの組み合わせ
- EPICSの制御
- 波形記録用書式（ウェブフォーム）を用いたチャネルアクセスとファイルアクセス

実際に作成されたコミッショニング用スクリプトの種類を以下に挙げる。例として、図1にQ磁石設定を行うスクリプトの抜粋を示す。

- Q磁石設定
- ビーム位置モニタ
- ビームロスモニタ
- ビーム電流モニタ
- ワイヤスキャナー（ビーム幅測定用）
- FCTによるビームエネルギー測定
- 横方向ビームマッチングアプリ
- モニターデジタイザー設定アプリ

参考文献

[1] URL: http://acc-physics.kek.jp/SAD/sad.html
[2] URL: http://www.sns.gov/APGroup/appProg/xal/xal.htm

```
path='/jk/dev/operation_app/jkLinac/accom/OM/setup/';

configs={"181MeV 5mA", "181MeV 20mA", "181MeV 25mA", "181MeV 27.5mA", "181MeV 30mA", "3MeV 5mA", "20MeV 5mA", "37MeV 5mA", "50MeV 5mA",
"S01 RFscan",
"S02 RFscan",

(中略)

tf = TabFrame[w,
!************ control ***********
  Add->{
    Tab[TabName->"Control",
      Add->{KBFComponentFrame[f,
        Add->{
          ScrollPane[Width->850,Height->600,
            Add->{KBFComponentFrame[
              Add-> { KBFGroup[Text->"Magnet Settings"]},
(後略)
```

図 1　Q磁石設定用スクリプト（抜粋）