Matlab-like Environment for Accelerator Modeling and Simulation*

Hiroshi Nishimura, Lawrence Berkeley National Laboratory,
University of California, Berkeley, CA 94720 USA

Abstract

An interactive GUI and a flexible programmability are equally important for accelerator modeling and simulation studies. However, it is not a trivial task to provide both of them. We discuss the use of Matlab and O-Matrix that is one of the Matlab-like environments on Windows, to obtain a reasonable solution by presenting examples at the ALS.

1 INTERACTIVE GUI AND PROGRAMMABILITY

1.1 Needs for Programmability

A modern GUI does not always provide flexible programmability for users. This becomes apparent in the case of an accelerator control system that has adopted a modern interactive GUI environment. If the specifications for such a system can cover all the needs of its future users, the software developers would be able to pre-program the required functionality with the support of a GUI. However, it is not realistic to assume such a scenario. Users will find missing functions at the last moment. Therefore, it is essential for an interactive GUI environment to provide some kind of programmability or extendibility to users.

In case of accelerator modeling and simulation studies, a GUI environment is not so popular as for a control system but it rises the same issue. As flexible programmability often becomes more important than a modern GUI, the GUI must cooperate with it.

Since user's algorithms are described by using a kind of language, a modeling program must parse and execute them. There are basically three ways to solve it:

1. Embed a language processing capability in it.
2. Export its application program interface (API).
3. Adopt an existing programmable environment.

The first case is usually realized by providing its own scripting language that is usually a small interpreter. The second case is common for a large scale environment like a machine control system. On the other hand, the third is suitable for a small system like a modeling program.

1.2 Experience at ALS

In the ALS[1] design phase, we took the first case to create a simulation program with full programmability. We wrote Tracy[2] by using the Pascal S compiler[3] itself as a framework to access physics library functions but without any graphics capabilities.

During the ALS commissioning phase, we wrote TracyV[4] to simulate the existing ALS storage ring in a modern GUI environment but without programmability. we do not feel too much inconvenience as it is to operate a well-defined virtual machine. At the same time, a C++ class library Goemon[5] was developed and used where a new logic is required.

When the commissioning is completed, Matlab[6] was adopted and has been in heavy use for the ALS machine controls and studies[7]. As its C-like interpreter is suitable for matrix calculations, it carries out orbit control quite effectively. Stimulated by the successful use of Matlab, we started using it for accelerator modeling and simulation studies as described in this paper.

2 MATLAB AS A MODELING ENVIRONMENT

2.1 Matlab as an Environment

Matlab is widely used on various platforms mainly for numerical calculations. It is an environment with a C-like interpreter that is suitable for matrix calculations with a rich set of library routines that cover mathematics, statistics, optimization and graphics. It also supports external routines that are written in C/C++ by users. Therefore, a user can use Matlab as an environment to manipulate the user's routines.

2.2 Matlab for Accelerator Modeling and Simulation

Accelerator optics and orbit calculations are mostly done by matrix and vector manipulations, therefore Matlab fits the needs effectively with various library functions. In addition to it, Matlab makes a lattice definition simple and flexible by providing the list data type, which releases us from writing a lattice parser and allows us to handle the lattice configuration dynamically.

* This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Material Sciences Division, U. S. Department of Energy, under Contract No. DE-AC03-76SF00098.

TracyML[8] is our first effort of writing an accelerator modeling code using the standard 4x5 matrix formalism entirely in Matlab. Lattice configurations, optics and orbit calculations, and various kinds of fitting and optimization calculations can be flexibly mixed in Matlab programs. Here is an example of defining a beam line in TracyML:

```
D1=drift('D1 ', 0.2);
D2=drift('D2 ', 0.1);
QF=quad ('QF ', 0.3, 2.12);
QD=quad ('QD ', 0.1,-2.25);
B =bend ('BEND',0.86,3.0,1.5,1.5,0.0);
LINE1 =[ D1 QF D2 ];
LINE2 =[ D2 QD D1 ];
CELL  =[ LINE1 B LINE2];
```

As this example shows, the Matlab list structure is appropriate to describe the lattice configuration. A version 5 of Matlab supports user-defined data types and some of the object-oriented features. TracyML uses the Matlab record structure to simplify the support of beam line elements such as magnets. It has been found to be convenient for fitting and optimizing calculations. On the other hand, as it is entirely written in Matlab, the execution speed is not adequate for numerical-intensive tasks like particle tracking. Fig. 1 shows an example of TracyML.

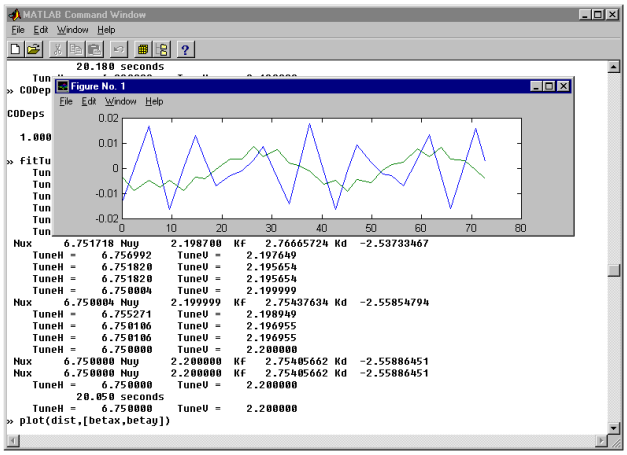

Fig.1 TracyML

2.3 Matlab External Calls

When the execution speed becomes a bottleneck in Matlab, we must start using external library calls that are supported by dynamic link libraries (DLL) on Windows. In case of accelerator modeling and simulation, there are two possibilities for the location that holds the virtual machine:

1. Keep the virtual machine in the Matlab layer and use external calls at a low level.
2. Keep the virtual machine in the external DLL layer and use external calls to manipulate it.

As our objective is to speed up the particle tracking speed, we must take the case 2, which leads to the use of a C++ class library Goemon as its external library.

The problem is that Matlab on Windows requires a separate DLL for each individual external call and each DLL carries its own memory space that is inadequate to share the virtual machine object. Possible solutions are, for example:

1. Use one generic external call and dispatch it to multiple Matlab calls.
2. Use one common DLL to interface DLLs for external calls to an external library.
3. Use an other Matlab-like environment that does not carry this kind of problem.

We are in a process of using case 2. At the same time, we found another environment (case 3), O-Matrix[9], and created TracyM[10] to access Goemon library from its Matlab-like environment, as described in the next chapter.

In case of the accelerator controls, we do not have to share the variables in the DLL layer because:

1. The operational context is held in the Matlab layer.
2. The real machine is common to all the external calls.

Therefore, Matlab does not have this kind of problem.

3 O-MATRIX AS A MODELING ENVIRONMENT

3.1 O-Matrix as an Environment

O-Matrix is a Matlab-like environment available on Windows that allows one DLL to provide multiple external calls. We can effectively wrap multiple C/C++ library routines for use from the environment. Therefore, if we are to hold the context in an external layer for faster execution speed, O-Matrix becomes more suitable than Matlab. On the other hand, one of the demerits of using O-Matrix will be the loss of the data structures that are available in Matlab 5, which leads us to do more in a DLL layer.

3.2 TracyM

TracyM is an attempt to obtain both modern GUI environment and programmability for accelerator design, simulation and modeling studies. It is built on top of O-Matrix by using the C++ class library Goemon in a DLL that serves as a physics library. The merits of TracyM over Tracy ML are:

1. Faster execution speed as all the CPU-intensive calculations are done in the DLL.
2. Better C++ Compatibility as it wraps the Goemon C++ class library.

In addition to these merits, it becomes easier to create external GUI tools as described in the next chapter. Here is an example of TracyM calculating the ALS booster ring properties.

```
clear
include TracyM.mat # link to DLL
initTracy
### Define Lattice ####
SYM=marker("")
LBF=drift("LBF",0.496875)
LBD=drift("LBD",0.546875)
LB =drift("LB ",1.05)
#** Bending **
BU =bend("BU",0.525,7.5,7.5,0.0, 0.0)
BD =bend("BD",0.525,7.5,0.0,7.5, 0.0)
BB =[BU,BD]
#** Quadrupole **
F2 =quad("F2",0.15, 2.7682214)
D2 =quad("D2",0.10,-2.5401249)
#** Construct One Superperiod **
DBF =[D2, LBD , BB , LBF , F2]
FBD =[F2, LBF , BB , LBD , D2]
FLD =[F2, LBF , LB,  LBD , D2]
DLF =[D2, LBD , LB,  LBF , F2]
BL1 =[SYM,DBF,FBD,DBF,FLD,DLF,FBD,DBF,FBD]
#** Create a Ring **
ring(BL1,4)

setQforTune(F2,D2)
getTwiss(1)

PrintTwiss
# plot Twiss functions
ginit
x=getVec(0)   # S
y=getVec(10)  # BetaH
gplot(x,y)
y=getVec(14)  # BetaV
gplot(x,y)

calcIntegral # synchrotron integrals

for n=1 to 30 begin
   E = n*50*1.0e6
   emt=calcEmittance(E)
   taux=DampingTimeH(emt)
   Emit0=Emittance(emt)
   print n*50, taux, Emit0
end

terminateTracy
```

Fig.2 is the result of this program.

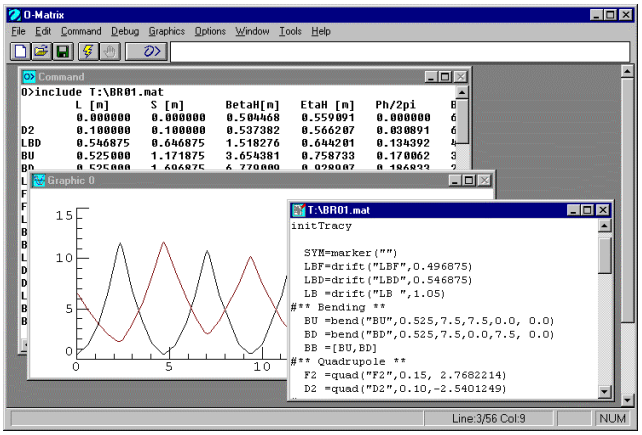

Fig.2 TracyM, Example (1)

Another example is of the dynamic aperture calculation of the ALS storage ring shown in Fig.3. Here is a routine to calculate the dynamic aperture.

```
for j=-20 to 20 begin
  for i=-30 to 30 begin
    x=i/1000.0
    y=j/1000.0
    n=quickTurns(512,0.10,x,y)
    print i,j,n
    if n>=512 then begin
      gplot(i,j,"square")
    end
  end
end
```

By using the external routine (quickTurns) we can keep the native speed where numerical calculation becomes intensive.

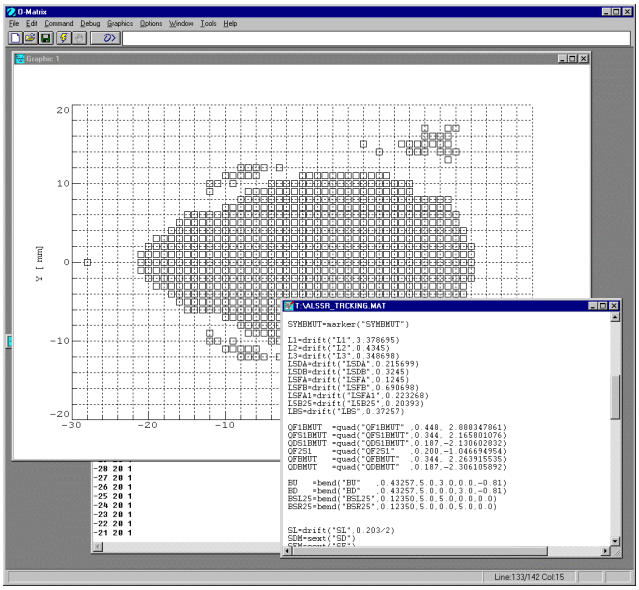

Fig.3 TracyM, Example (2)

4 MATLAB-LIKE ENVIRONMENT AND STANDARD GUI

4.1 A Matlab-LIKE ENVIRONMENT AND GUI

We have shown the examples of TracyML and TracyM with some graphics displays, but none of them can claim that it has a GUI. It is true that Matlab and O-Matrix do support creation of a GUI with buttons and menus, but we feel more comfortable to have dedicated GUI applications that are natively compiled for interactive use. In the case of accelerator modeling and control, it is possible to prepare a standard set of GUI for common use, such as displaying Twiss functions and closed orbits. For these items, users should not be forced to create event-driven routines every time they need them.

4.2 TracyM and its External Tools

From such an aspect, TracyM is accompanied by several external programs that provide standard displays with

GUI. Fig.4 shows two of such programs, Table View and Graph View, displaying the current status of the virtual machine of TracyM by communicating with TracyM through Win32 shared memory.

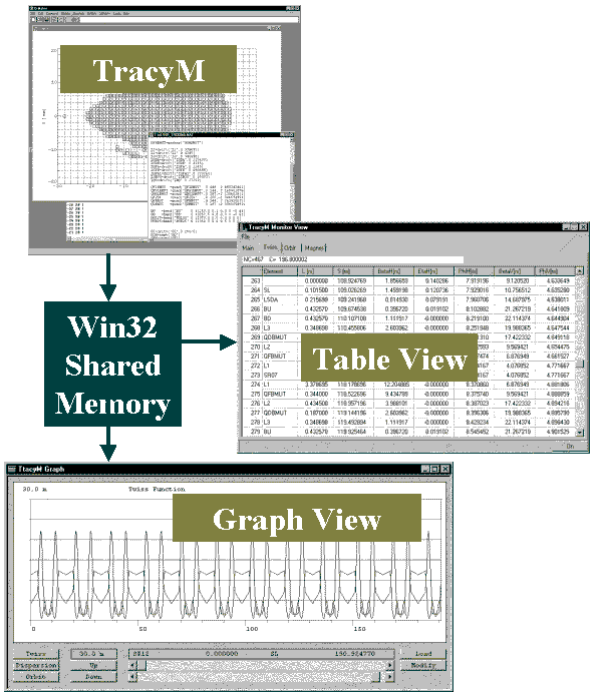

Fig.4 TracyM and External Tools

Another external tool is Command View that manipulates a standard set of knobs of the virtual machine using GUI. In this case, TracyM becomes a server for Command Panel and communicating with it through a Win32 pipe. Although this scheme is still in a development phase, we do see some possibility of creating a GUI application without sacrificing a flexible programmability. In this scheme, the role of TracyM is to create a virtual machine for use from external GUI tools.

5 CONCLUSION

A Matlab-like environment such as O-Matrix can serve as a programmable environment for C/C++ libraries that are specialized in various fields. Matlab 5 with user-definable data structure works efficiently for accelerator control, modeling and simulations. Lattice definition becomes simple and dynamic by using the list structure. When the execution speed is required, O-Matrix becomes effective because of its external calling scheme. If we can identify a standard set of displays, separate programs dedicated to them may become advantageous.

REFERENCES

[1] "1-2 GeV Synchrotron Radiation Source, Conceptual Design Report," LBL PUB-5172 Rev. LBL,1986.
A. Jackson, IEEE 93PAC, 93CH3279-7,1432, 1993.

[2] H. Nishimura, "TRACY, A Tool for Accelerator Design and Analysis", EPAC 88,803,1989.
J.Bengtsson, E.Forest and H.Nishimura, "Tracy Users Manual", unpublished.

[3] R.E.Berry, "Programming Language Translation", Ellis Horwood Ltd., England, 1981.

[4] H. Nishimura, " Accelerator Modeling and Control Using Delphi on Windows NT", IWCSMSA96, KEK Proceedings 97-19, 174.

[5] H. Nishimura, " Taking an Object-Oriented View of Accelerators", IEEE 95PAC, 95CB35843,2162,1996.
H. Nishimura, LSAP-153, LBL Internal Report, 1993.

[6] The Mathworks, Inc. Natic, MA., U.S.

[7] G. J. Portmann,"ALS Storage Ring Setup & Control Using MATLAB", LBNL ALS/LSAP-248,1990.
G. J. Portmann,"Recipe for ALS Storage Ring Operation", LBNL ALS/LSAP-249,1990.

[8] H. Nishimura,"Introduction to TracyML", LBNL ALS/LSAP-246,1998.

[9] Harmonic Software, Inc. Seattle, WA., U.S.

[10] H. Nishimura, "Introduction to TracyM, Part 1", LBNL ALS/LSAP-258,1998.