Создание интерактивных графических моделей в CAS MAXIMA при использовании ОС GNU Linux.

Предисловие.

Не так давно я посмотрел фильм: [TED] Конрад Вольфрам. Как учить детей настоящей математике с помощью компьютеров в котором автор использовал некие программы для улучшения процесса обучения детей, представляющие собой математические модели управляемые с помощью интерфейсных графических элементов, таких как кнопки, шкалы, меню. Я слышал что подобные интерфейсные элементы встречаются в других математических пакетах и естественно знал, что в максиме, ничего подобного нет, не считая конечно wxMaxima, которая является небольшой графической надстройкой над максимой, но и она не позволяет встраивать подобные элементы в расчетный документ, тем самым не позволяет создать математическую модель управление расчетом в которой можно производить с помощью графических интерфейсных элементов. Поэтому вопрос о создании интерактивных интерфейсов в максиме долгое время оставался ддя меня открытым и не видя очевидных путей его решения я решил поизучать, как работает графическая система отображения в Максиме.

Предпосылки.

В максиме имеются два основных способа отображения графической информации это функции plot и draw, простейший график рисуется вызовом функции plot2d(x^2,[x,-9,9]);. Оба этих класса функций используют другой математический пакет gnuplot, как раз и специализирующийся на построении и выводе графиков. Поэтому я занялся изучением возможностей данного пакета и в процессе изучения документации наткнулся на один очень интересный параметр, который сыграл ключевую роль в успешной реализации задуманного. При установке типа терминала в gnuplot, для типа терминала X11 допускается параметр window, описан он плохо, и для чего необходим можно было лишь только догадываться, поэтому я провел эксперимент, по изучению возможности использования этого параметра, было бы очень приятно если бы gnuplot мог выводить графики в любое окно которое мы ему укажем в этом параметре, для эксперимента я написал простейшую программу на питоне в которой создавал фрейм заданного размера и пару кнопок, идентификатор этого окна, а вернее фрейма очень легко узнать командой:

```
xwininfo -tree
  Root window id: 0xbd (the root window) (has no name)
  Parent window id: 0x12559b9 (has no name)
     1 child:
     0x3c0000e (has no name): ()  640x509+0+0  +2+80
        3 children:
        0x3c00012 (has no name): ()  67x29+109+480  +111+560
        0x3c00011 (has no name): ()  109x29+0+480  +2+560
        0x3c00010 (has no name): ()  640x480+0+0  +2+80
```

Теперь запустим gnuplot и дадим там несколько команд:

```
gnuplot> set terminal x11 window "3c00010"
Terminal type set to 'x11'
Options are 'XID 0x3C00010 nopersist'
gnuplot> plot(sin(x));
```

вуаля!!! гнуплот рисует график функции в указанном окне, при этом все кнопки приложения работают . Вот так вот и выяснилось, что этот параметр является ключевым в возможности интеграции gnuplota в различные графические приложения в системе Xwindow.

Реализация.

Идея построения интерактивной графической параметрической модели родилась

после этого эксперимента довольно быстро, суть ее состояла в том что, мы добавим в максиму функцию позволяющую строить интерактивные графики организованные по принципу Модель-Вид-Контроллер, в которой maxima будет отвечать за модель, gnuplot отвечать за вид, а управлять всем этим и реализовывать контроллер будет программа на питоне(замечу, выбор питона это чисто мое предпочтение, контроллер можно организовать на любом, приятном для вас языке). Здесь я приведу окончательный и полный текст программы на максиме и опишу что делает каждая ее строчка, но для начала опишу что нужно от пользователя что бы мы получили интерактивную графическую модель, во первых ее надо определить на языке максима, а так же указать параметры, их начальные значения и диапазон значений(не обязательно, но я не проверял что будет если это не сделать :-)), а так же указать где находиться на диске вид-контроллер для данной модели.

Создаем модель в максиме:
к примеру это будет график функции n*sin(x) и график функции sin(b*x)
допустим в нашей модели n*sin(x), n должен меняться от 0.1 до 10.0 с шагом 0.2, поскольку я намереваюсь управлять этим значением при помощи шкалы значений целых чисел, то в формулу надо внести изменения (a/10.)*sin(x), а диапазон передаваемый в контроллер будет от 1 до 100, с начальным значением 1 и шагом изменения 2(для демонстрации).
параметр b начальное значение 1 диапазон от 1 до 12, и зададим параметр c меняющий диапазон расчета графиков от 2 до 5, с начальным значением 2

```
g1:explicit((a/10.)*sin(x),x,-c*%pi,c*%pi);
g2:explicit(sin(b*x),x,-c*%pi,c*%pi);
model:'draw2d(color=red,g1, color=blue,g2);
param_set:[[a,1,"1:100:2"],[b,1,"1:12"],[c,2,"2:5"]];
name_model:sconcat(maxima_tempdir,"/work/maxima/gnuplot/mvc_3scale.py");
```

model — это наша параметризованная модель, param_set — это набор параметров с заданием начальных значений и диапазона возможных значений, name_model — это имя контроллера который будет вызываться для отображения нашей модели вместо стандартной программы отображения графиков gnuplot. Все модель задана осталось только запустить ее в обработку, для обработки я выбрал имя функции idraw, в смысле интерактивное рисование.

```
idraw(model,param_set,name_model);
```

Итак функция idraw:

```
/* i -означает интерактивный */
idraw(model,param_set,name_model) := block(
   [old_plot_format,t1,pipe_view_name,pipe_view,plt_fmt,cmd,ll],
   pipe_view_inter_n:sconcat(maxima_tempdir, "/maxima.pipe_interact"),
   pipe_view_param_n:sconcat(maxima_tempdir, "/maxima.pipe_param"),
      /*подготовим канал для взаимодествия с процессом интерактивного представления
модели вид-контроллер и канал для передачи параметров от модели в контроллер*/
   if not probe_file(pipe_view_inter_n) then system(sconcat("mkfifo ", pipe_view_inter_n)),
   if not probe_file(pipe_view_param_n) then system(sconcat("mkfifo ", pipe_view_param_n)),
      /*запомним предыдущий способ отрисовки графиков и установим свой*/
   old_plot:gnuplot_command,
   gnuplot_command:name_model,
   gnuplot_close(), /*надо обязательно закрыть предыдущую сессию gnuplot иначе максима не
вызовет новую модель*/
      /*выполним отрисовку интерактивной модели с установками по умолчанию*/
   t1:map(lambda([x],x[1]=x[2]),param_set),
```

```
   ev(model,nouns,t1),
      /*через канал передачи параметров максимы и интерактивного вида-контроллера
настроим контроллер*/
      /*с обратной стороны канала уже должен работать процесс считывающий данные из
него, костыль с |cat >file приделан т. к. максима почему то не может открыть на запись файл
fifo*/
   pipe_view:openw(sconcat("| cat >",pipe_view_param_n)),
      /*в цикле передадим имеющиеся параметры*/
   map(lambda([x],printf(pipe_view,"~a:~d:~a~%", x[1],x[2],x[3]),0),param_set),
   printf(pipe_view,"end~%"),
   close(pipe_view),
      /*теперь открываем на чтение канал интерактивной передачи параметров*/
   kill(pipe_view),
   pipe_view:openr(pipe_view_inter_n),
     /*запускаем цикл чтения данных из канала управления моделью*/
   cmd:1,
   while cmd=1 do block(
     if stringp(ll : readline(pipe_view)) then block (
        print("read from pipe_view: ", ll), /*для отладки*/
                  /*вид приходящей строки (cmd:1,t1:[a = 1,b = 1,c = 2])*/
        eval_string(ll),  /*устанавливаем переменные t1,cmd*/
                  /*получая новые значения параметров, пересчитываем модель и
перерисовываем view*/
        if cmd=1 then
           ev(model,nouns,t1 )
     )
     else
        cmd:0
   ),
   close(pipe_view),
   gnuplot_close(),
   gnuplot_command:old_plot
);
```

view в gnuplot не нуждается в какой либо программе, это полностью управляемый элемент через stdin в который может писать команды и maxima и контроллер.

Что же должен представлять собой контроллер? У нас это программа на питоне, которая с одной стороны должна принимать команды от максимы и передавать их в gnuplot, с другой стороны она должна передавать установленные в контроллере новые значения параметров обратно в максиму. Вкратце опишу что в ней происходит:
0) Поскольку функция idraw заменила название основной утилиты для отрисовки графиков, то вместо gnuplot будет вызвана наша программа, итак создается main_controller_process
1) create_Tk_interface() Создаем графический интерфейс, как то фрейм в который будет выводить gnuplot построенные на основе данных из максимы, элементы управления — Scale, которые затем будут инициализироваться принятыми из максимы параметрами, и значения которых в последствии будут передаваться обратно в максму, и пара кнопок(пересчитать и выход).
2)create_gnuplot_pipe() создаем процесс gnuplot и получаем два канала к нему для ввода и вывода данных.
3)change_terminal() поскольку интерфес Тк мы построили, гнуплот запустили, можем дать команду в гнуплот об установке типа терминала X11 и конкретного окна для вывода

графиков.
4)make_process_reader() Запускаем дочерний питон процесс, который будет читать данные из stdin потока и передавать их в процесс gnuplot, таким образом обеспечиваем передачу данных из максимы в гнуплот.
5)open_pipe_param(read) открываем канал для чтения значений параметров из максимы
5.1) do_read_param() читаем построчно данные из максимы, разбираем ввод и формируем хеш массив param
6) do_set_widget_param() используя ранее полученные значения параметров сохраненные в хеш массиве param
7) open_pipe_interact(write) открываем на запись канал для интерактивной передачи команд в максиму, где сейчас должна работать функция idraw, читающая и интерпретирующая эти команды.
8)do_main_Tkloop() запускаем цикл обработки сообщений Tk интерфейса

В процессе работы Tk интерфейса могут происходить следующие события:
8.1)set_param() Пользователь установил новое значение параметра, это новое значение просто сохраняется в хеш массиве param
8.2) recalc() Пользователь нажал на кнопку пересчитать, при этом все текущие значения сохраненные в массиве param оформляются в командную строку которую способна интерпретировать максима и через канал передаются туда, в максиме, эти данные интерпретируются моделью, после чего сформированные команды для gnuplot передаются через процесс-посредник в gnuplot, таким образом гнуплот выводит новое графическое представление модели, в соответствии с установленными значениями параметров.
8.3) quit() пользователь нажал на кнопку выйти, или другим образом завершил работу контроллера, при этом в максиму передается команда, завершающая цикл чтения канала интерактивной передачи значений параметров, сигналом kill завершается работа процесса посредника child_reader_process, затем закрывается и сеанс gnuplot.
На этом программа работы контроллера и завершается.
Для лучшего понимания процесса взаимодействия всех участников данной системы, я нарисовал диаграмму взаимодействия процессов рис 1. Данной модели соответствует работа программ mvc_2edit_1scale.py и mvc_3scale.py, их я здесь приводить не буду, т. к. с ними вы сможете ознакомиться изучив прилагаемы к этому документу файлы.

Собираем все вместе: определяем модель в каком нибудь файле, например model2.mac содержащий:

```
work_dir:sconcat(maxima_tempdir,"/work/maxima/gnuplot/");
/*загружаем функцию интерактивного рисования*/
load(sconcat(work_dir,"idraw"));
/*описание модели*/
g1:implicit(a=x^2+y^2+z^2,x,-c,c,y,-c,c,z,-c,c);
g2:explicit(sin(b*x)*sin(2*y),x,-2*c,2*c,y,-2*c,2*c);
model:'draw3d(surface_hide=true,color=red,g1, color=blue,g2);
param_set:[[a,1,"1:6:1"],[b,1,"1:12"],[c,1,"1:15"]];
name_model:sconcat(work_dir,"mvc_3scale.py");
```

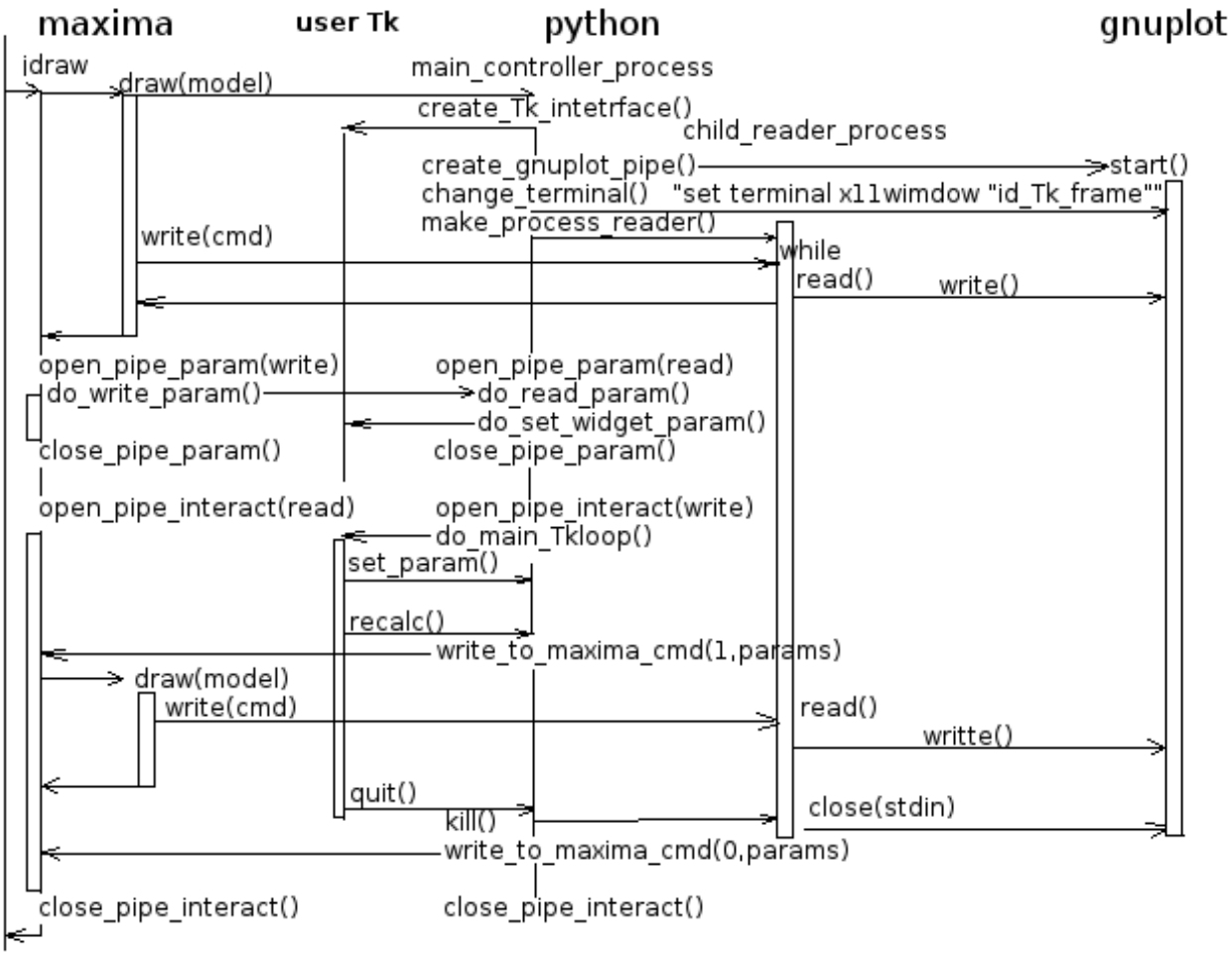
корректируем файл контроллера или создаем новый, меняя состав виджетов и названия параметров, соответствующих нашей модели.
Загружаем максиму, и даем команду:

```
load(sconcat(maxima_tempdir,"/maxima/gnuplot/model3.mac"));
```

ну и теперь выполняем отрисовку модели, и управление моделью через полученный интерфейс.

```
idraw(model,param_set,name_model);
```

рисунок 1



В результате мы должны получить нечто подобное рис 2.
Пример использующий контроллер mvc_2edit_1scale.py допускает изменение параметров с помощью полей ввода целых чисел и чисел с плавающей запятой(вернее точкой). Но для того что бы он заработал, пришлось слегка изменить формат передачи параметров и начать указывать в них какого типа значение будет принимать параметр.

Пример создадим файл model2_1.mac

```
work_dir:sconcat(maxima_tempdir,"/work/maxima/gnuplot/");
load(sconcat(work_dir,"idraw"));

g1:explicit(a*sin(x),x,-c*%pi,c*%pi);
g2:explicit(sin(b*x),x,-c*%pi,c*%pi);
call_d:'draw2d(color=red,g1, color=blue,g2);
model:call_d;
param_set:[[a,55.0,"f:0.1:100.0"], [b,17,"i:1:100"], [c,2,"1:10"]];
name_model:sconcat(work_dir,"mvc_2edit_1scale.py");
```

в определении параметров добавили столбец определяющий тип параметра иint(i) или float(f).

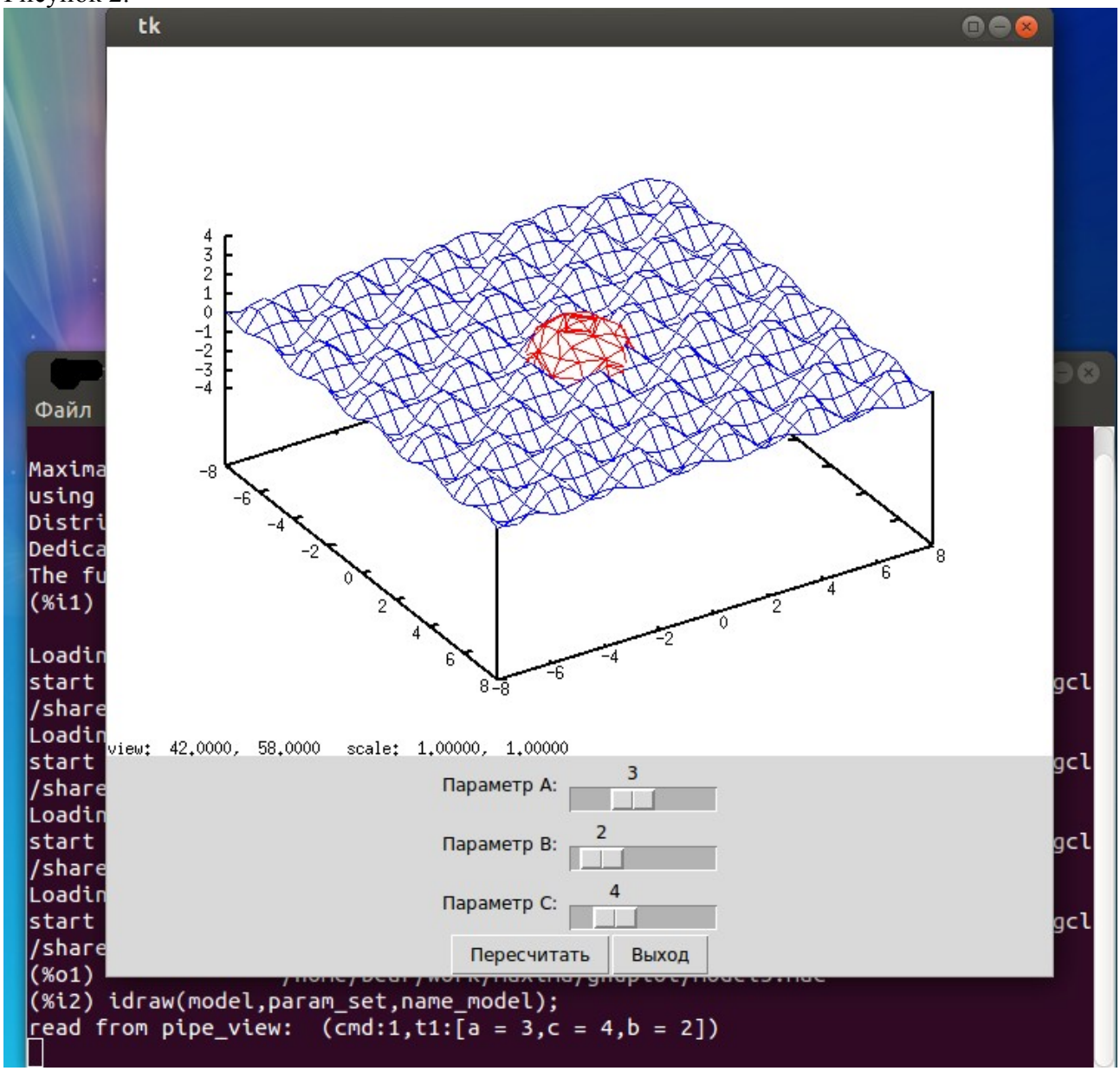
```
load(sconcat(maxima_tempdir,"/work/maxima/gnuplot/model2_1.mac"));
```

и запустим интерпретацию нашей модели модели

```
idraw(model,param_set,name_model);
```

должны получить нечто подобное рис.3

Рисунок 2.

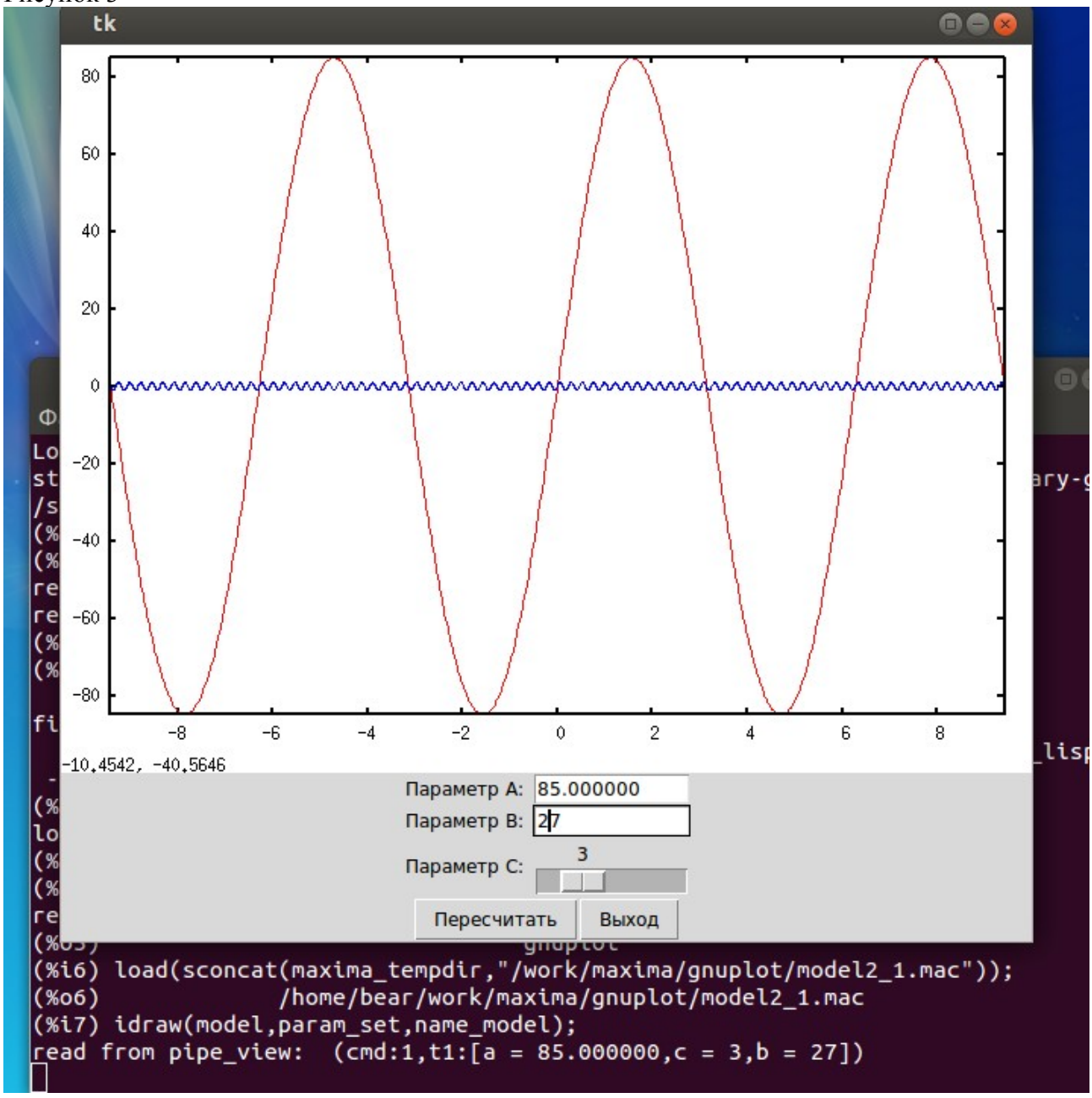


Ну и в завершении я представлю программу mvc_any_param.py которая идеально подойдет для людей не желающих программировать для каждой модели еще и контроллер, она способна в минимальном представлении отрисовать любое передаваемое ей количество параметров любого типа. Для того что бы она работала, необходимо опять изменить формат передаваемых параметров, и для каждого параметра указывать какого типа виджет мы желаем использовать для его редактирования. Добавим в начало каждой строки параметров, там где мы ранее задавали диапазон, еще одно поле со значениями s — Scale или e — Entry. Вот изучаем пример, файл model4.mac:

```
work_dir:sconcat(maxima_tempdir,"/work/maxima/gnuplot/");
load(sconcat(work_dir,"idraw"));

g1:explicit((a/10.)*sin(x),x,-c*%pi,c*%pi);
g2:explicit(sin(b*x),x,-c*%pi,c*%pi);
model:'draw2d(color=red,g1, color=blue,g2);

param_set:[[a,55.0,"e:f:10.0:1000.0"], [b,17,"e:i:1:100"], [c,2,"s:1:10"]];
```

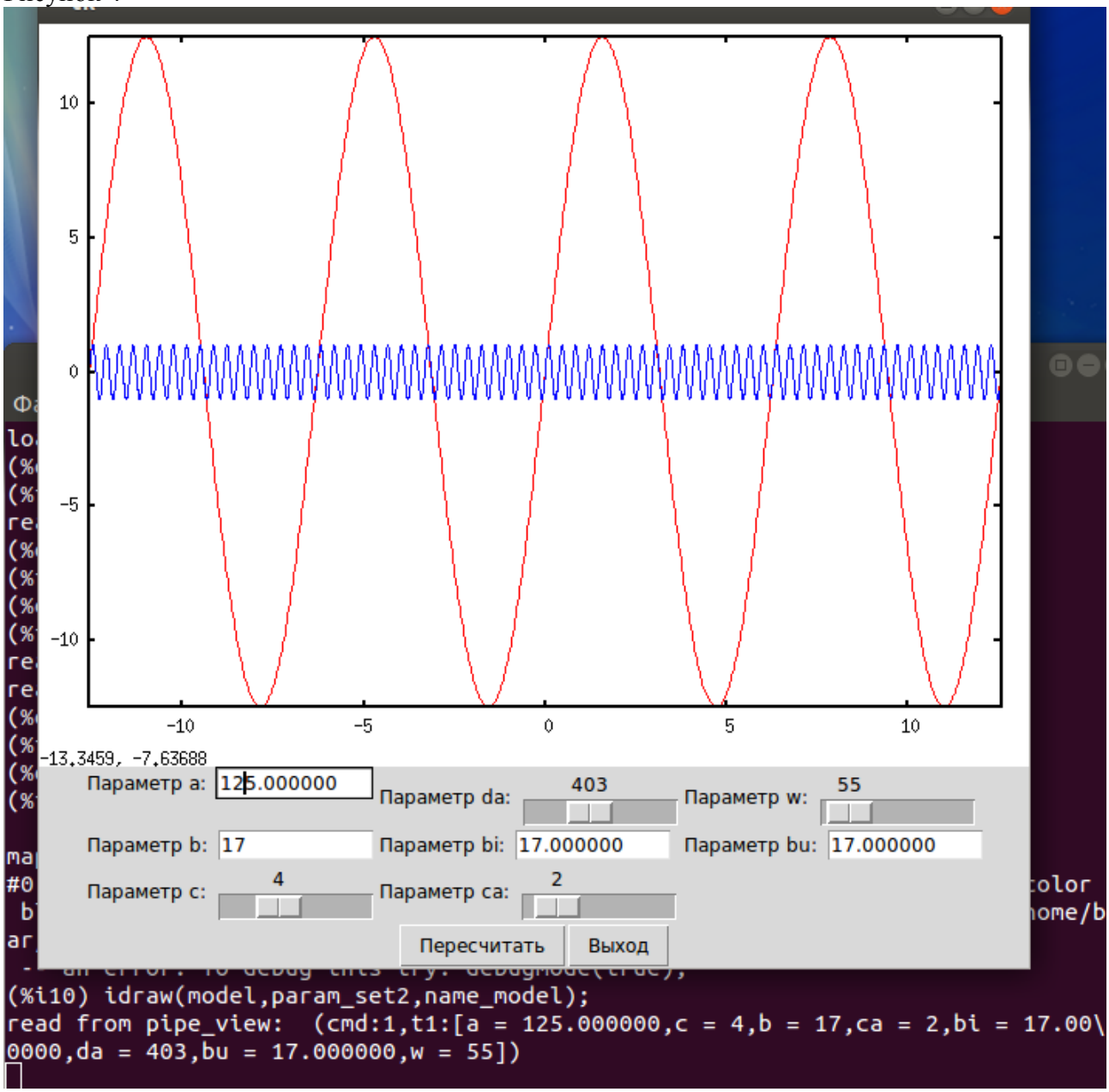
```
param_set1:[[a,55.0,"e:f:10.0:1000.0"], [b,17,"e:i:1:100"], [c,2,"s:1:10"],
        [da,55,"s:10:1000"], [bi,17,"e:f:1:100"], [ca,2,"s:0:20"]];
param_set2:[[a,55.0,"e:f:10.0:1000.0"], [b,17,"e:i:1:100"], [c,2,"s:1:10"],
        [da,55,"s:10:1000"], [bi,17,"e:f:1:100"], [ca,2,"s:0:20"],
        [w,55,"s:10:1000"], [bu,17,"e:f:1:100"]];

name_model:sconcat(work_dir,"mvc_any_param.py");
```

Рисунок 3



Загрузив нашу модель командой:
load(sconcat(maxima_tempdir,"/work/maxima/gnuplot/model4.mac"));
и запустим интерпретацию нашей модели модели
idraw(model,param_set2,name_model);
Мы должны получить что то подобное рисунку 4
Я не стал изменять саму модель, просто продемонстрировал простоту добавления параметров.

Рисунок 4



Заключение.

Ну вот собственно говоря и все. В принципе в программе можно еще можно что делать и совершенствовать. Но как говориться: лучшее враг хорошего, к нему можно бесконечно приближаться, а работу надо рано или поздно заканчивать. Удачи вам, удосужившимся ознакомиться с моим трудом, надеюсь знакомство с максимой, линуксом и питоном обогатят вашу жизнь, сделают шире ваши возможности, как это произошло со мной.

Разработал Гагин Михаил ака NuINu с благодарностью к разработчикам Maxima, Gnuplot, Python, Linux и всему сообществу OpenSource.