|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **«Московский государственный технический университет  имени Н.Э. Баумана»**  **(МГТУ им. Н.Э. Баумана)** |

**Отчет по лабораторной работе**

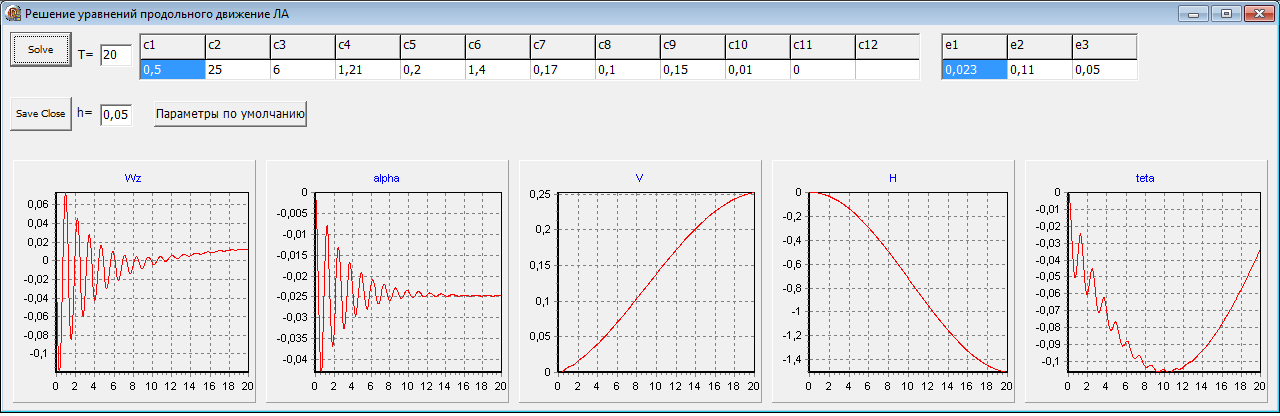
“Моделирование продольного движения ЛА”

Студент Рабаданов Г.Р.

Группа ИУ2-109

2011 г

Программа

****

Программа решает систему уравнений продольного движения самолета по заданным коэффициентам. Используется линейная модель движения в вариациях. Строятся графики изменения величин Wz, угла атаки, скорости, высоты и угла тангажа. Орган управления – руль высоты – находится в постоянном положении.

# Объявление классов

*type*

*//Хранилище параметров движения*

***TParametresOfFly = packed record***

*a: array[1..5,0..5] of double;*

*c: array[1..11] of double;*

*e: array[1..3] of double;*

*h: double;*

*tt: double;*

*end;*

*TFormBase = class;*

*TBorder = class;*

*TController = class;*

*TStorage = class;*

*//Абстрактный класс фармы*

***TFormBase = class abstract (TForm)***

*//Набор значений Ci*

*StringGrid1: TStringGrid;*

*//Набор значнеий Ei*

*StringGrid2: TStringGrid;*

*Label1: TLabel;*

*//Время интегрировния*

*Edit1: TEdit;*

*//Сохранить и выйти*

*Button1: TButton;*

*//Решалка с-мы ду и построение графиков*

*Button2: TButton;*

*Chart1: TChart;*

*Series1: TLineSeries;*

*Chart2: TChart;*

*Chart3: TChart;*

*Chart4: TChart;*

*Chart5: TChart;*

*Series2: TLineSeries;*

*Series3: TLineSeries;*

*Series4: TLineSeries;*

*Series5: TLineSeries;*

*Label2: TLabel;*

*//Шаг интегрирования*

*Edit2: TEdit;*

*//Загрузить параметры движения по умолчанию*

*Button3: TButton;*

*procedure Button1Click(Sender: TObject);*

*procedure Button2Click(Sender: TObject);*

*procedure FormActivate(Sender: TObject);*

*procedure Button3Click(Sender: TObject);*

*end;*

*//Класс-граница*

***TBorder = class (TFormBase)***

*Private*

*\_Controller: TController;*

*//Метод, обновляющий параметры движения на форме*

*procedure UpdateParam;*

*public*

*//Метод, закрывающий и сохраняющий параметры движения из формы*

*procedure SaveCloseButton;*

*//Метод, решающий уравнения и отрисовывающий графики*

*procedure SolveButton;*

*//Метод, связывающий границу с контроллером*

*procedure Connect(\_Controller\_: TController);*

*//Метод, отрисовывающий графику*

*procedure PlotGraph(f: array of double; series: TLineSeries);*

*//Метод, выравнивающий элементы на форме(задает начальные положения)*

*procedure FormActivate;*

*//Метод, загружающий деолтне параметры движения*

*procedure LoadDefaultParametres;*

*end;*

***TController=class***

*private*

*//указатель на класс-границу*

*\_Border: TBorder;*

*//указатель на класс-хранилище*

*\_Storage: TStorage;*

*//возвращает значение функции угловой скорости для данного момента времени*

*function w\_t(w,alpha,V,H,teta: double):double;*

*//возвращает значение функции угла атаки для данного момента времени*

*function alpha\_t(w,alpha,V,H,teta: double):double;*

*//возвращает значение функции скорости для данного момента времени*

*function V\_t(w,alpha,V,H,teta: double):double;*

*//возвращает значение функции высоты для данного момента времени*

*function H\_t(w,alpha,V,H,teta: double):double;*

*//возвращает значение функции угла тангажа для данного момента времени*

*function teta\_t(w,alpha,V,H,teta: double):double;*

*public*

*//Значения соответствующих величин от времени*

*w: array of double;*

*alpha: array of double;*

*V: array of double;*

*H: array of double;*

*teta: array of double;*

*//Метод, соединяющий с границей и хранилищем*

*procedure Connect(\_Border\_: TBorder; \_Storage\_: TStorage);*

*//Метод, выполняющий операции при закрытии формы*

*procedure Exit;*

*//Метод, решающий с-му ду*

*procedure Solve;*

*//Метод, загружающий параметры движения из хранилища*

*function GetParametresOfFly: TParametresOfFly;*

*//Метод, устанавливающий параметры движения*

*procedure SetParametresOfFly(p: TParametresOfFly);*

*//Метод, загружающий параметры по умолчанию*

*procedure LoadDefaultParametres;*

*//Метод, инициализующий загрузку параметров из файла*

*procedure LoadParamFromFile;*

*end;*

*//Класс-хранилище*

***TStorage=class***

*private*

*//Указатель на класс-контроллер*

*\_Controller: TController;*

*//Метод, считающий коэффицинты движения линейной системы по параметрам режима полета*

*procedure UpdateA;*

*public*

*//Хранилище параметров полета*

*param: TParametresOfFly;*

*//Конструктор =))*

*constructor Create;*

*//Метод, соединяющий с контроллером*

*procedure Connect(\_Controller\_: TController);*

*//Метод, записывающий текущие параметра движения в файл*

*procedure SaveParamToFile;*

*//Метод, устанавливающий рабочие(текущие) параметры движения*

*procedure SetParametresOfFly(p: TParametresOfFly);*

*//Метод, получающий текущие параметры движения*

*function GetParametresOfFly: TParametresOfFly;*

*//Метод, устанавливающий параметры по умолчанию как текущие*

*procedure SetDefaultParametres;*

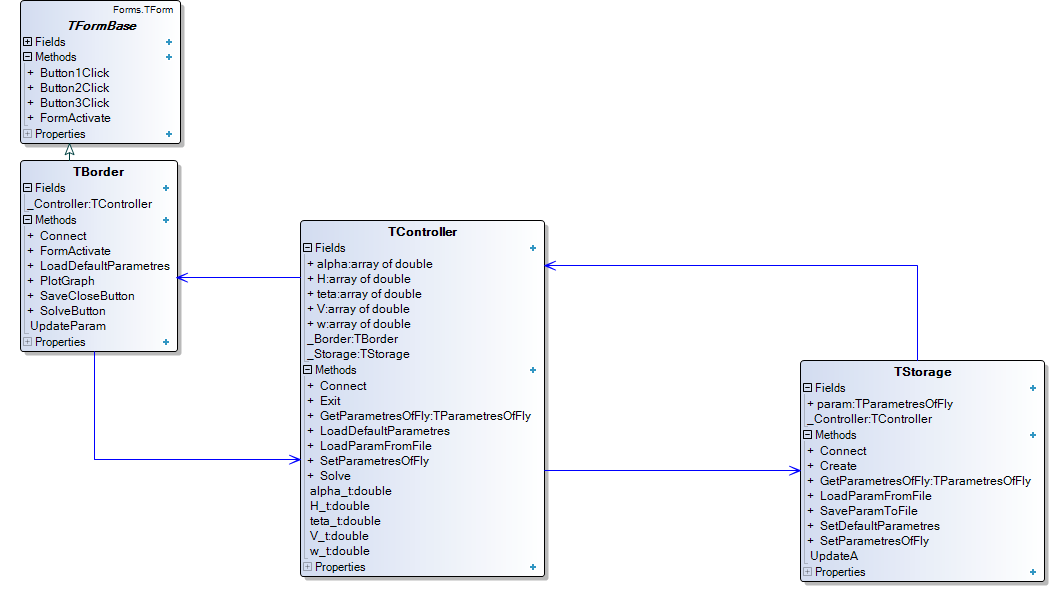
*//Метод, загружащий параметры из файла и устанавливающий их по умолчанию*

*procedure LoadParamFromFile;*

*end;*

**Диаграмма классов**

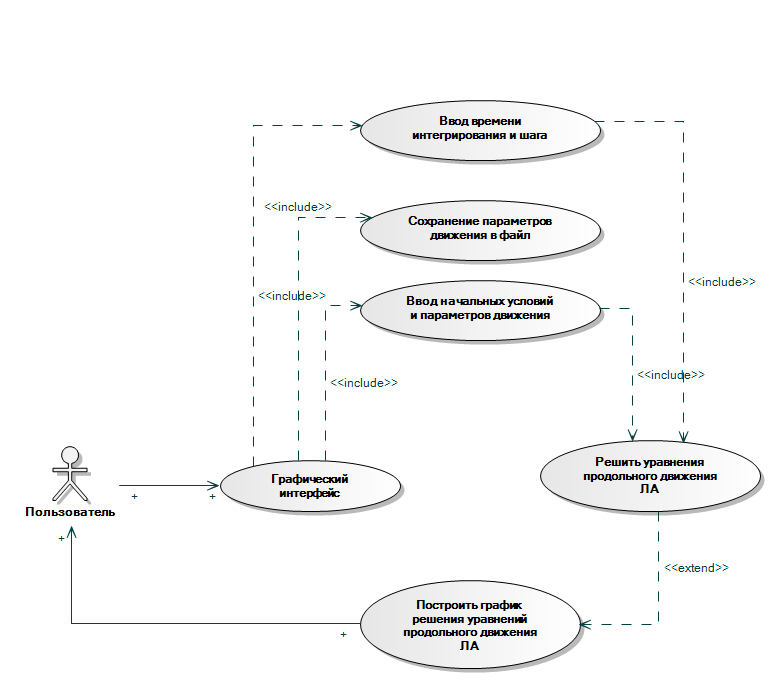
Диаграмма классов (Class diagram)— статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.



**Диаграмма вариантов использования**

Диаграмма вариантов использования (Use case diagram)— диаграмма, на которой отражены отношения, существующие между [акторами](http://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D0%BE%D1%80_(UML)) и вариантами использования.

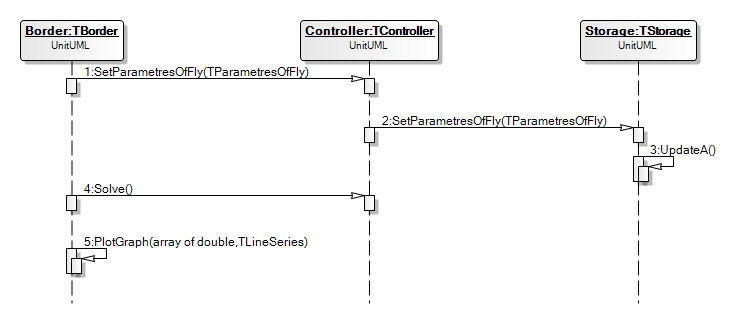
Основная задача— представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.



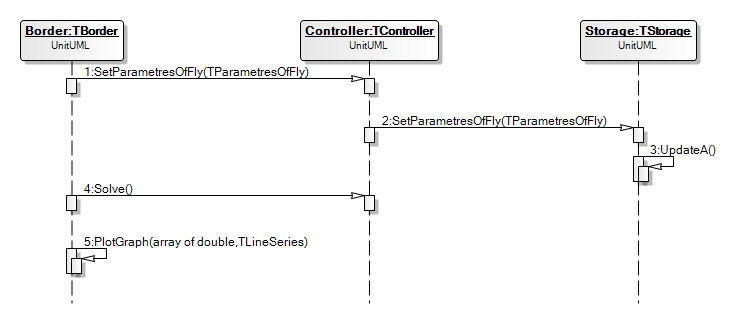
**Диаграмма взаимодействия**

Диаграмма последовательности *(*sequence diagram)—диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

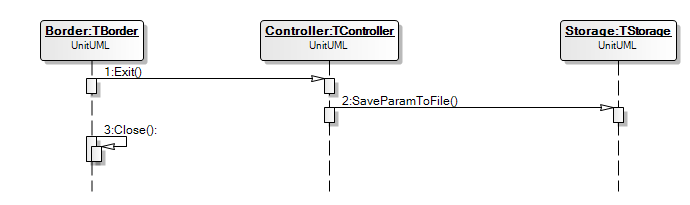
*1)Загрузка программы*



*2) Решение системы уравнений и построение графика*

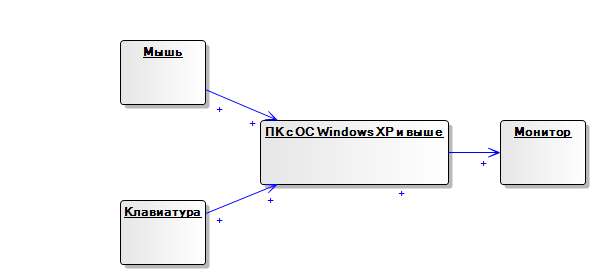


*3)Выход из программы и сохранение параметров*



**Диаграмма развертывания**

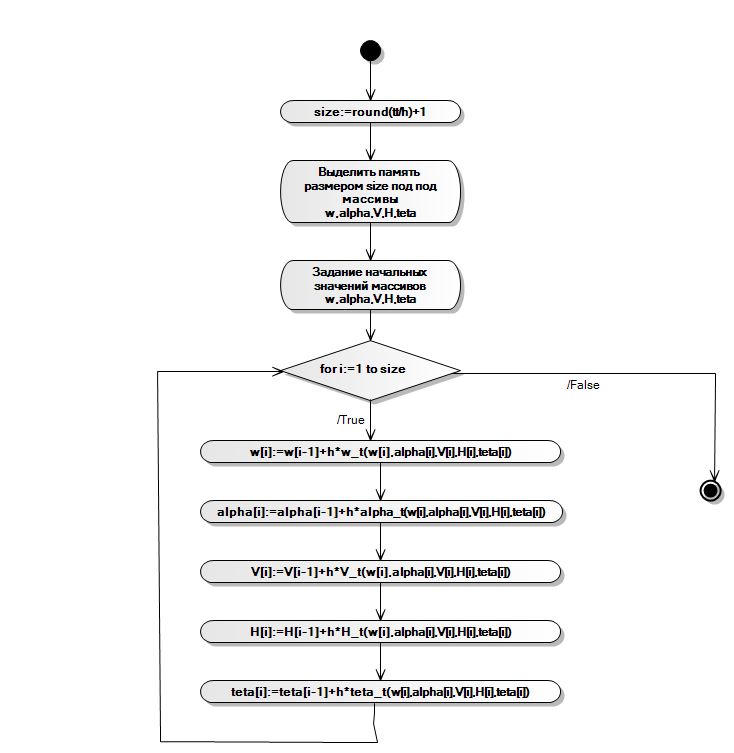
Диаграмма развёртывания (Deployment diagram)— служит для моделирования работающих узлов(аппаратных средств, node).



**Диаграмма деятельности**

Диаграмма деятельности (Activity diagram)— диаграмма, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью (activity) понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов— вложенных видов деятельности и отдельных действий (action), соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

*Алгоритм решения системы ДУ*



**Диаграмма компонентов**

Диаграмма компонентов (Component diagram)— статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты ит.п.

