МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Кафедра
«Управление и информатика в технических системах»

ПАРАМЕТРИЧЕСКИЙ СИНТЕЗ САУ С ПОМОЩЬЮ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ

Рекомендовано редакционно-издательским советом университета в качестве методических указаний

для студентов специальности
“УПРАВЛЕНИЕ И ИНФОРМАТИКА В ТЕХНИЧЕСКИХ СИСТЕМАХ”

МОСКВА – 2010

УДК: 681.3.001.2:621.396.6

М17

Монахов О. И., Сафронов А. И., Ковалев М. В.. Рындина Е. Ю. Параметрический синтез САУ с помощью пакетов прикладных программ. Методические указания к курсовому проектированию по дисциплине «Автоматизация проектирования систем и средств управления». – М.: МИИТ. 2010. – 138 с.

Данные методические указания предназначены для изучения основ проектирования в рамках курса «Автоматизация проектирования систем и средств управления», а также могут быть использованы при выполнении лабораторных работ и в дипломном проектировании. Методические указания составлены в виде описания последовательности действий пользователя при работе с пакетами MBTY, MATLAB, LABVIEW с подробными комментариями. Изучать принципы работы пакетов рекомендуется в процессе выполнения заданий, приведенных в конце каждого раздела.

Предисловие

Курсовая работа – это, в принципе, самостоятельная работа студента, при выполнении которой возникает много вопросов, ответы на которые можно получить, в частности, на консультациях с преподавателем. Чаще всего это вопросы, связанные с выполнением тех или иных разделов, алгоритмами работы программ, оформлением пояснительной записки и т.п. Однако конкретные вопросы, связанные с методикой работы на ПЭВМ с использованием сложных программных продуктов, заданием численных значений параметров алгоритмов, последовательностью действий студента при формировании тех или иных графических образов и т.п. иногда остаются не очень понятными.

Данный материал, предназначенный для студентов 5-го курса специальности «Управление и Информатика в Технических Системах», озабоченных выполнением курсового проекта по дисциплине «Автоматизация проектирования систем и средств управления». В этих методических указаниях подробно описываются действия студента при решении конкретной задачи (устойчивость, качество, оптимизация параметров САУ) при использовании таких программных продуктов как LabView, MathLab, МВТУ. Даются рекомендации по разработке индивидуальной программы для решения подобных задач с использованием языка Delphi.

Нижеприведённые разделы написаны:

1. MathLab – Сафронов А.И.
2. МВТУ – Ковалёв М.В.
3. LabView – Рындина Е.Ю.
4. Delphi – Сафронов А.И., Рындина Е.Ю.

Цель курсового проектирования

Целью курсового проекта является параметрический синтез корректирующего устройства, обеспечивающего устойчивость заданной следящей системы автоматического управления (САУ) и выполнение поставленных требований к динамике САУ [1].

Введение

На рис.1 изображена структурная схема исследуемой в данном курсовом проекте следящей САУ.

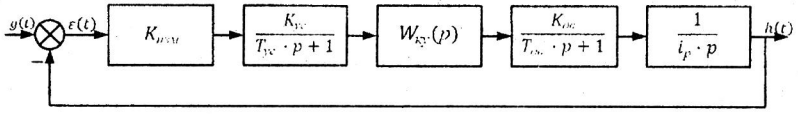


Рисунок 1 – Структурная схема следящей САУ

Рассматриваемая САУ состоит из следующих звеньев:

- Сельсинная пара (СД-СП), работающая в трансформаторном режиме и имеющая статический коэффициент усиления $K_{изм}$.
- Усилитель с передаточной функцией:

$$W_{ус}(p)=\frac{K_{ус}}{T_{ус}\cdot p+1} \qquad (1)$$

где $K_{ус}$ – коэффициент усиления усилителя;

$T_{ус}$ – постоянная времени усилителя.

- Синтезируемое корректирующее устройство с передаточной функцией $W_{ку}(p)$.
- Двигатель с передаточной функцией:

$$W_{дв}(p)=\frac{K_{дв}}{(T_{дв}\cdot p+1)\cdot p} \qquad (2)$$

где $K_{дв}$ – коэффициент усиления усилителя;

$T_{дв}$ – постоянная времени усилителя.

- Редуктор с передаточной функцией:

$$W_p(p) = \frac{1}{i_p} \tag{3}$$

где i_p – передаточное число редуктора.

На вход системы поступает воздействие типа единичная ступень $g(t) = 1(t)$. На выходе получаем $h(t)$ – переходную функцию системы. Следящая система отрабатывает рассогласование между входом и выходом $\varepsilon(t)$.

Под синтезом корректирующего устройства САУ понимается определение структуры и параметров корректирующего устройства в соответствии с заданным критерием качества функционирования. В данном курсовом проекте предъявляются требования по выполнению заданного времени регулирования переходного процесса t_p и перерегулирования σ.

В приложении 1 табл.1. представлены исходные данные для выполнения курсовой работы.

Коэффициент усиления всей системы (общий коэффициент усиления), который будет обеспечивать заданную кинетическую ошибку ε при входном воздействии типа $\omega t \cdot 1(t)$, вычисляется по формуле:

$$K_{общ} = \frac{\omega}{\varepsilon} \tag{4}$$

где ω – скорость изменения входного воздействия;

ε – заданная кинетическая ошибка.

Коэффициент предварительного безынерционного усилителя рассчитывается по формуле:

$$K_{пр.ус} = \frac{K_{общ}}{\left(\frac{K_{изм} \cdot K_{ус} \cdot K_{ов}}{i_p} \right)} \tag{5}$$

Замечание 1

Перед решением задачи параметрического синтеза средствами рассматриваемых далее пакетов прикладных программ *МВТУ*, *Matlab*,

Labview необходимо произвести расчеты «вручную». Для этого надлежит построить логарифмические амплитудно-частотные характеристики (ЛАЧХ) нескорректированной системы $L_{нск}(\omega)$ и желаемой системы $L_{ж}(\omega)$ [2]. При последовательной коррекции ЛАЧХ искомого корректирующего устройства $L_{ку}(\omega)$ находится как разница между ЛАЧХ желаемой и ЛАЧХ нескорректированной системы, т.е. $L_{ку}(\omega) = L_{ж}(\omega) - L_{нск}(\omega)$. При параллельной коррекции для нахождения ЛАЧХ корректирующего устройства $L_{ку}(\omega)$ необходимо из ЛАЧХ нескорректированной системы вычесть ЛАЧХ желаемой системы и ЛАЧХ звена, охватываемого обратной связью с КУ, т.е. $L_{ку}(\omega) = L_{нск}(\omega) - L_{ж}(\omega) - L_{охв}(\omega)$. По точкам излома полученной $L_{ку}(\omega)$ определяется структура и значения параметров корректирующего устройства.

Полученную при «ручном» расчете структуру и параметры корректирующего устройства следует рассматривать как исходные, и использовать в качестве начальной информации при решении поставленной задачи проектирования с помощью программных средств *MBTY*, *Matlab*, *Labview*.

4. Рекомендации по Pascal и Delphi

4.1. Ввод исходных данных

Прежде, чем вести разговор о методике задания исходных данных, нужно определиться с выбором языка программирования. Этот момент является принципиальным, поскольку дальнейшая обработка исходных данных в среде объектно-ориентированных языков (ООЯП) имеет существенные отличия по сравнению с обработкой данных в среде алгоритмических языков программирования (АЯП). Однако, не будем упускать из рассмотрения некоторый общий случай, который будет удобен лишь программисту-разработчику, но отнюдь не рядовому пользователю. Чтобы вышеописанные различия не являлись для Вас пустыми словами, рассмотрим их на конкретных примерах, где в качестве АЯП будем использовать Pascal, а в качестве ООЯП - Delphi. Если у Вас ранее была практика работы в среде Pascal, то переход в среду Delphi не составит особых трудностей, поскольку, синтаксически (по написанию операторов) эти два языка программирования схожи.

Договоримся, что далее в тексте методических указаний при описании синтаксических форматов операторов конструкция {текст} будет означать не более чем комментарий. Соответственно, при программировании, ввод символов «{» и «}» не требуется.

4.1.1. Непосредственное присвоение значений переменным и константам на основе исходных данных:

Как было сказано ранее, - этот метод удобен лишь для программиста-разработчика, то есть, имеет смысл использовать его на этапе отладки программы. Начнём с рассмотрения присвоения значений константам. Под константы в обоих языках отводится определённый раздел объявления, начинающийся со служебного слова «const» (от английского constant - постоянная). Далее присваиваются значения по следующей структуре:

{имя переменной} = {значение};

На рис. 4.1. приведены константы и присвоенные им значения для варианта №30

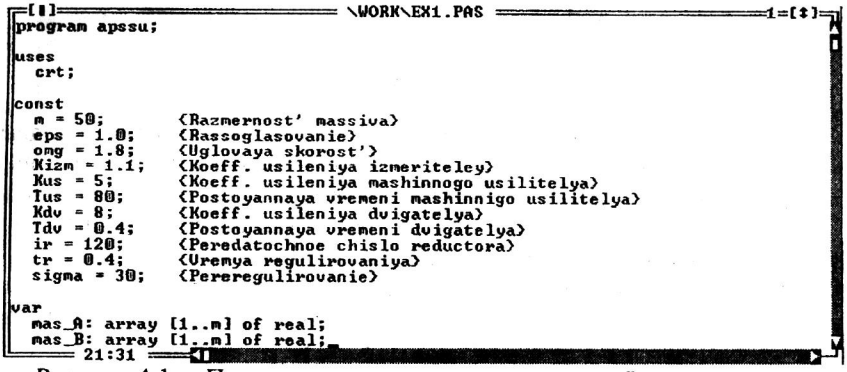


Рисунок 4.1. – Пример ввода исходных данных путём присвоения значений константам

Попутно рассмотрим присвоение значений переменным в теле программы. Структура присвоения стандартная. В АЯП Pascal реализуется при помощи оператора присвоения «:=»:

{имя переменной}:={значение};

На рис. 4.2. приведён фрагмент с переменными и присвоенными им значениями в соответствии с исходными данными, заданными по варианту №30.

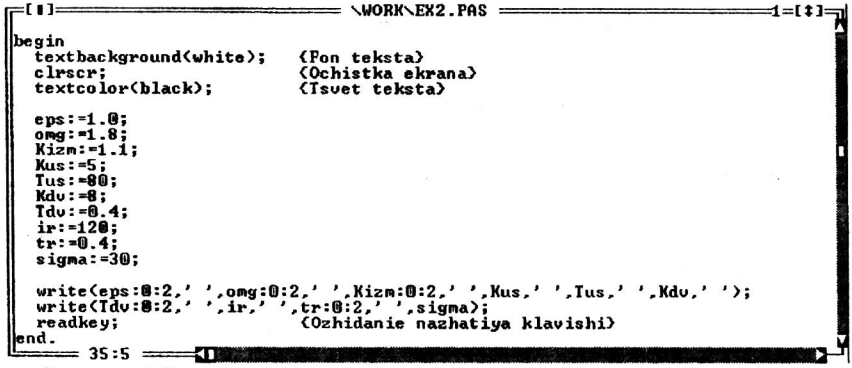


Рисунок 4.2. – Пример ввода исходных данных путём присвоения значений переменным

Замечание 1.1: в случае непосредственного присвоения значений переменным не забудьте объявить переменные в соответствующем разделе, начинающемся со служебного слова «var» (от английского variable - переменная). Пример такого объявления приведён на рис. 4.3.

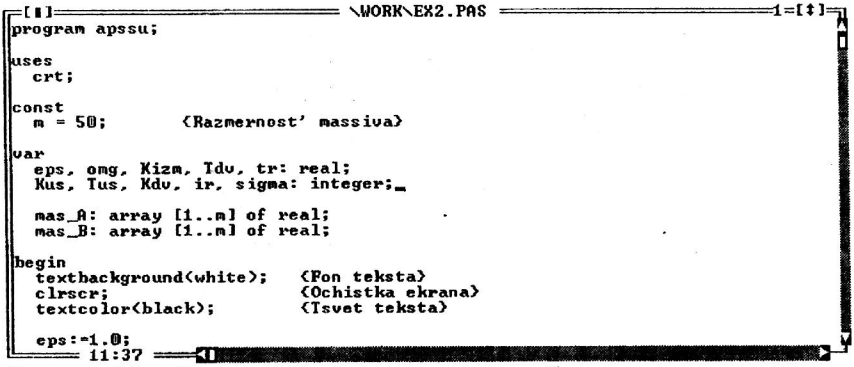
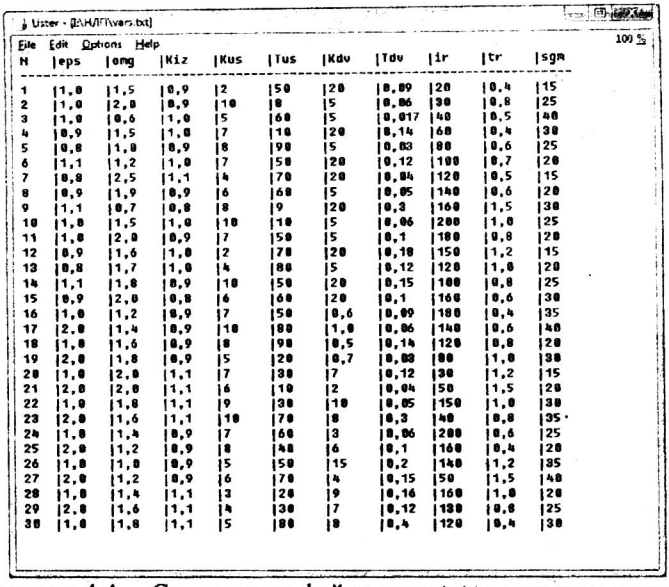


Рисунок 4.3. – Пример оформления разделов объявления переменных и констант

4.1.2. Ввод исходных данных из файла:

Пусть для выполнения курсового проекта преподавателем предложена методика ввода информации из выданного на носителе (дискета/компакт-диск/dvd/usb-flash накопитель) текстового файла. В выданном файле содержатся данные, которые обязательно разделены между собой некоторыми символами-разделителями, будь то пробел или несколько пробелов для более конкретного обособления. Однако, пробел не всегда удобен при программировании, как разделяющий символ, поскольку он может использоваться в иных целях, и это приведёт к ошибке считывания. Избежать появления подобных ошибок можно посредством введения редко используемых символов, таких как «|», «@», «~» и других. Авторами, в качестве примера, выбран символ «|», так как он при вертикальной разметке файла помогает наглядно обособить столбцы таблицы исходных данных. Содержимое файла с таблицей исходных данных изображено на рис. 4.4.



User - [D:\НИЛ\vars.txt]

File Edit Options Help

N	eps	omg	Kiz	Kus	Tus	Kdv	Tdv	ir	tr	sgm
1	1,0	1,5	0,9	2	50	20	0,09	20	0,4	15
2	1,0	2,0	0,9	10	0	5	0,06	30	0,8	25
3	1,0	0,6	1,0	5	60	5	0,017	40	0,5	40
4	0,9	1,5	1,0	7	10	20	0,14	60	0,4	30
5	0,8	1,0	0,9	8	90	5	0,03	80	0,6	25
6	1,1	1,2	1,0	7	50	20	0,12	100	0,7	20
7	0,8	2,5	1,1	4	70	20	0,04	120	0,5	15
8	0,9	1,9	0,9	6	60	5	0,05	140	0,6	20
9	1,1	0,7	0,8	8	9	20	0,3	160	1,5	30
10	1,0	1,5	1,0	10	10	5	0,06	200	1,0	25
11	1,0	2,0	0,9	7	50	5	0,1	180	0,8	20
12	0,9	1,6	1,0	2	70	20	0,10	150	1,2	15
13	0,8	1,7	1,0	4	80	5	0,12	120	1,0	20
14	1,1	1,0	0,9	10	50	20	0,15	100	0,8	25
15	0,9	2,0	0,8	6	60	20	0,1	160	0,6	30
16	1,0	1,2	0,9	7	50	0,6	0,09	180	0,4	35
17	2,0	1,4	0,9	10	80	1,0	0,06	140	0,6	40
18	1,0	1,6	0,9	8	90	0,5	0,14	120	0,8	20
19	2,0	1,8	0,9	5	20	0,7	0,08	80	1,0	30
20	1,0	2,0	1,1	7	30	7	0,12	30	1,2	15
21	2,0	2,0	1,1	6	10	2	0,04	50	1,5	20
22	1,0	1,0	1,1	9	30	10	0,05	150	1,0	30
23	2,0	1,6	1,1	10	70	8	0,3	40	0,8	35
24	1,0	1,4	0,9	7	60	3	0,06	200	0,6	25
25	2,0	1,2	0,9	8	40	6	0,1	160	0,4	20
26	1,0	1,0	0,9	5	50	15	0,2	140	1,2	35
27	2,0	1,2	0,9	6	70	4	0,15	50	1,5	40
28	1,0	1,4	1,1	3	20	9	0,16	160	1,0	20
29	2,0	1,6	1,1	4	30	7	0,12	130	0,8	25
30	1,0	1,8	1,1	5	80	8	0,4	120	0,4	30

Рисунок 4.4. – Содержание файла «vars.txt» исходных данных

Файлы, порой, имеют свойство теряться, искажаться (изменяется имя, содержимое) или же просто присутствовать не в той папке. На все эти случаи необходимо предусмотреть проверки. Прежде всего, нужно предусмотреть проверку наличия файла в папке. Без неё не имеет смысла осуществлять прочие проверки, которые нужны будут лишь в том случае, когда нет доступа к файлу для его редактирования. Если же файл для редактирования открыт, то предусматривать проверки в программе совсем необязательно, достаточно лишь вручную отредактировать все искажённые данные, если таковые имеются.

Далее приведён фрагмент кода программы, составленной авторами данных методических указаний на ООЯП Delphi, касающийся считывания исходных данных из файла.

Пример 1.1:

```
var
  frmMain: TfrmMain;
  ft: TextFile;
  {Выбор метода задания исходных данных}
procedure TfrmMain.rgVariantClick(Sender: TObject);
begin
  case rgVariant.ItemIndex of
    0: {Случай ручного ввода исходных данных}
      begin
        {01 Доступны элементы ввода epsilon}
        lbeps.Enabled:=True;
        eeps.Enabled:=True;
        lbRazm1.Enabled:=True;
        {02 Доступны элементы ввода omega}
        lbomg.Enabled:=True;
        eomg.Enabled:=True;
        lbRazm2.Enabled:=True;
        {03 Доступны элементы ввода Kizm}
        lbkizm.Enabled:=True;
        ekizm.Enabled:=True;
        lbRazm3.Enabled:=True;
        {04 Доступны элементы ввода Kus}
        lbkus.Enabled:=True;
        ekus.Enabled:=True;
        lbRazm4.Enabled:=True;
        {05 Доступны элементы ввода Tus}
        lbtus.Enabled:=True;
        etus.Enabled:=True;
        lbRazm5.Enabled:=True;
        {06 Доступны элементы ввода Kdv}
        lbkdv.Enabled:=True;
        ekdv.Enabled:=True;
        lbRazm6.Enabled:=True;
        {07 Доступны элементы ввода Tdv}
        lbtdv.Enabled:=True;
        etdv.Enabled:=True;
        lbRazm7.Enabled:=True;
        {08 Доступны элементы ввода ip}
        lbi.Enabled:=True;
        ei.Enabled:=True;
        lbRazm8.Enabled:=True;
        {09 Доступны элементы ввода tp}
        lbtreg.Enabled:=True;
        etreg.Enabled:=True;
        lbRazm9.Enabled:=True;
        {10 Доступны элементы ввода sigma}
        lbsigm.Enabled:=True;
        esigm.Enabled:=True;
        lbRazm10.Enabled:=True;
        {Названия кнопок}
        btnApply.Caption:='Не используется';
```

```
    btnEntry.Caption:='1. Проверить ввод';
    btnPereschot.Caption:='2. Пересчёт единиц';
    btnOK.Caption:='3. Готово';
    {Состояние кнопок}
    eNumVar.Enabled:=False;
    lbNumVar.Enabled:=False;
    btnApply.Enabled:=False;
    btnEntry.Enabled:=True;
  end;
1: {Случай ввода исходных данных из БД}
  begin
    {01 Недоступны элементы ввода epsilon}
    lbeps.Enabled:=False;
    eeps.Enabled:=False;
    lbRazm1.Enabled:=False;
    {02 Недоступны элементы ввода omega}
    lbomg.Enabled:=False;
    eomg.Enabled:=False;
    lbRazm2.Enabled:=False;
    {03 Недоступны элементы ввода Kizm}
    lbkizm.Enabled:=False;
    ekizm.Enabled:=False;
    lbRazm3.Enabled:=False;
    {04 Недоступны элементы ввода Kus}
    lbkus.Enabled:=False;
    ekus.Enabled:=False;
    lbRazm4.Enabled:=False;
    {05 Недоступны элементы ввода Tus}
    lbtus.Enabled:=False;
    etus.Enabled:=False;
    lbRazm5.Enabled:=False;
    {06 Недоступны элементы ввода Kdv}
    lbkdv.Enabled:=False;
    ekdv.Enabled:=False;
    lbRazm6.Enabled:=False;
    {07 Недоступны элементы ввода Tdv}
    lbtdv.Enabled:=False;
    etdv.Enabled:=False;
    lbRazm7.Enabled:=False;
    {08 Недоступны элементы ввода ip}
    lbi.Enabled:=False;
    ei.Enabled:=False;
    lbRazm8.Enabled:=False;
    {09 Недоступны элементы ввода tp}
    lbtreg.Enabled:=False;
    etreg.Enabled:=False;
    lbRazm9.Enabled:=False;
    {10 Недоступны элементы ввода sigma}
    lbsigm.Enabled:=False;
    esigm.Enabled:=False;
    lbRazm10.Enabled:=False;
    {Названия кнопок}
    btnApply.Caption:='1. Принять';
```

```
        btnEntry.Caption:='Не используется';
        btnPereschot.Caption:='3. Пересчёт единиц';
        btnOK.Caption:='4. Готово';
        {Состояние кнопок}
        eNumVar.Enabled:=True;
        lbNumVar.Enabled:=True;
        btnApply.Enabled:=True;
        btnEntry.Enabled:=False;
      end;
  end;
end;
  {Выбор метода задания исходных данных}
  {Проверка на наличие существующих вариантов}
procedure TfrmMain.btnApplyClick(Sender: TObject);
var
  error,i,boardcount: integer;
  stroka, st: string;
  flgboard, flgExist, flgSim: boolean;
begin
  case rgVariant.ItemIndex of
    1: {Считывание данных из базы данных}
      begin
        flgSim:=False;
        st:=eNumVar.Text;
        for i:=1 to length(st) do
          begin
            {В таблице символов ASCII позиции 48 - 57
             соответствуют цифрам "0" - "9"}
            if ((ord(st[i]) < 48) or
               (ord(st[i]) > 57)) then
              begin
                flgSim:=True;
                break;
              end;
          end;
        if not flgSim then
          if (StrToInt(eNumVar.Text) > 30) or
             (StrToInt(eNumVar.Text) < 1) then
            MessageDlg('Введите вариант в диапазоне 1 -
                        30', mtWarning, [mbYes],0)
          else
            begin
              tbVariant.Open;
              tbVariant.RecNo:=StrToInt(eNumVar.Text);
              eeps.Text:=tbVariant.
                        FieldByName('Epsilon').AsString;
              eomg.Text:=tbVariant.
                        FieldByName('Omega').AsString;
              ekizm.Text:=tbVariant.
                        FieldByName('Kizm').AsString;
              ekus.Text:=tbVariant.
                        FieldByName('Kus').AsString;
              etus.Text:=tbVariant.
```

```
                    FieldByName('Tus').AsString;
          ekdv.Text:=tbVariant.
                    FieldByName('Kdv').AsString;
          etdv.Text:=tbVariant.
                    FieldByName('Tdv').AsString;
          ei.Text:=tbVariant.
                    FieldByName('Ir').AsString;
          etreg.Text:=tbVariant.
                    FieldByName('Treg').AsString;
          esigm.Text:=tbVariant.
                    FieldByName('Sigma').AsString;
          tbVariant.Close;
        end
    else
      MessageDlg('Введённые символы не относятся к'
                 + ' разрешённым: "0" - "9"',
                 mtError,[mbOK],0);
  end;
2: {Считывание данных из файла}
  begin
    flgExist:=True;
    flgSim:=False;
    AssignFile(ft, 'vars.txt');
    {$IOChecks off}
    Reset(ft);
    error:=IOResult;
    {$IOChecks on}
    if not (error = 0) then
      begin
        MessageDlg('Ошибка! Файл не найден',
                   mtError, [mbOK],0);
        rgVariant.ItemIndex:=0;
      end
    else
      begin
        st:=eNumVar.Text;
        for i:=1 to length(st) do
          begin
            {В таблице символов ASCII позиции 48 -
             57 соответствуют цифрам "0" - "9"}
            if ((ord(st[i]) < 48) or
                (ord(st[i]) > 57)) then
              begin
                flgSim:=True;
                break;
              end;
          end;
        if not flgSim then
          if StrToInt(eNumVar.Text) < 1 then
            MessageDlg('Проверьте ввод. Номер
                       варианта должен быть' +
                      ' натуральным числом',
                       mtWarning, [mbOK],0)
```

```
else
  begin
    for i:=0 to StrToInt(eNumVar.Text) do
      begin
        readln(ft);
        if Eof(ft) then
          begin
            MessageDlg('Вашего варианта в
                       списке нет.' +
                       ' Задайте его
                       вручную',
                       mtWarning,
                       [mbOK],0);
            rgVariant.ItemIndex:=0;
            flgExist:=False;
            break;
          end;
      end;
    if flgExist then
      begin
        readln(ft,stroka);
        flgboard:=false;
        boardcount:=0;
        for i:=1 to length(stroka) do
          begin
            if (stroka[i]='|') then
              begin
                boardcount:=boardcount+1;
                flgboard:=true;
              end;
            if flgboard then
              case boardcount of
                1:
                  begin
                    if not (stroka[i+1]=
                            ' ') then
                      eeps.Text:=eeps.
                          Text+stroka[i+1]
                    else
                      flgboard:=false;
                  end;
                2:
                  begin
                    if not (stroka[i+1]=
                            ' ') then
                      eomg.Text:=eomg.
                          Text+stroka[i+1]
                    else
                      flgboard:=false;
                  end;
                3:
                  begin
                    if not (stroka[i+1]=
```

```
                ' ') then
          ekizm.Text:=ekizm.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    4:
      begin
        if not (stroka[i+1]=
                ' ') then
          ekus.Text:=ekus.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    5:
      begin
        if not (stroka[i+1]=
                ' ') then
          etus.Text:=etus.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    6:
      begin
        if not (stroka[i+1]=
                ' ') then
          ekdv.Text:=ekdv.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    7:
      begin
        if not (stroka[i+1]=
                ' ') then
          etdv.Text:=etdv.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    8:
      begin
        if not (stroka[i+1]=
                ' ') then
          ei.Text:=ei.
              Text+stroka[i+1]
        else
          flgboard:=false;
      end;
    9:
      begin
```

```
                                    if not (stroka[i+1]=
                                            ' ') then
                                      etreg.Text:=etreg.
                                          Text+stroka[i+1]
                                    else
                                      flgboard:=false;
                                  end;
                                10:
                                  begin
                                    if not (stroka[i+1]=
                                            ' ') then
                                      esigm.Text:=esigm.
                                          Text+stroka[i+1]
                                    else
                                      flgboard:=false;
                                  end;
                              end;
                          end;
                        MessageDlg('Считано: '+stroka,
                                mtInformation, [mbOK],0);
                    end;
                end
            else
              MessageDlg('Введённые символы не относятся
                          к' +' разрешённым: "0" - "9"',
                          mtError,[mbOK],0);
            CloseFile(ft);
          end;
      end;
  end;
  eNumVar.Enabled:=False;
  btnApply.Enabled:=False;
  btnPereschot.Enabled:=True;
end;
  {Проверка на наличие существующих вариантов}
  {Проверка ввода данных при пользовательском вводе}
procedure TfrmMain.btnEntryClick(Sender: TObject);
var
  i,j: integer;                          {Параметры цикла}
  coma: integer;                         {Счётчик запятых}
  stroka: array [1..10] of string;       {Строковый массив}
  flgSim, flgEmpty, flgComa: boolean;    {Флаги ошибок}
begin
  flgSim:=False;    {Флаг ошибочных символов снят}
  flgEmpty:=False;  {Флаг пустых полей ввода снят}
  flgComa:=False;{Флаг наличия более одной запятой снят}
  {Заполнение строкового массива содержимым полей ввода}
  stroka[1]:=eeps.Text;
  stroka[2]:=eomg.Text;
  stroka[3]:=ekizm.Text;
  stroka[4]:=ekus.Text;
  stroka[5]:=etus.Text;
  stroka[6]:=ekdv.Text;
```

```
stroka[7]:=etdv.Text;
stroka[8]:=ei.Text;
stroka[9]:=etreg.Text;
stroka[10]:=esigm.Text;
{Проверка на наличие пустых полей ввода}
for i:=1 to 10 do
  if stroka[i]='' then
    begin
      flgEmpty:=True;
      break;
    end;
{Проверка на наличие ошибочных символов}
if not flgEmpty then
  for i:=1 to 10 do
    begin
      coma:=0; {сброс счётчика запятых}
      for j:=1 to length(stroka[i]) do
        begin
          {В таблице символов ASCII позиции 48 - 57
           соответствуют цифрам "0" - "9", а позиция
           44 есть ","}
          if ((ord(stroka[i][j]) < 48)
             or (ord(stroka[i][j]) > 57)) and
             (not (ord(stroka[i][j]) = 44)) then
            begin
              flgSim:=True;
              break;
            end;
          if (ord(stroka[i][j]) = 44) then
            begin
              coma:=coma+1;
              if coma > 1 then
                begin
                  flgComa:=True;
                  break;
                end;
            end;
        end;
      if flgSim then
        break;
    end;
{Сообщения об ошибках по соответствующим этим ошибкам
 флагам}
if flgEmpty then
  MessageDlg('Заполнены не все поля',
             mtError,[mbOK],0);
if flgComa then
  MessageDlg('В поле допускается не более одной
             запятой', mtError,[mbOK],0);
if flgSim then
  MessageDlg('Введённые символы не относятся к' +
             ' разрешённым: "0" - "9" и ","',
             mtError,[mbOK],0);
```

```
    {При безошибочном вводе разрешить следующий шаг
     выполнения программы}
    if (not flgEmpty) and (not flgSim) and
       (not flgComa) then
      begin
        btnEntry.Enabled:=False;
        btnPereschot.Enabled:=True;
      end;
  end;
  {Проверка ввода данных при пользовательском вводе}
```

4.1.3. Ввод исходных данных посредством пользовательского ввода АЯП:

Рассматриваемый вариант позволяет любому, без исключения, пользователю программы задать необходимые ему расчётные значения непосредственно в процессе выполнения программы. Запрограммировать такой ход можно при помощи оператора «read» или «readln». В первом случае значения будут считываться из одной строки, если разделять их пробелом, а по нажатию Enter курсор будет переводиться на следующую строчку. Во втором случае после ввода значения будет производиться перевод курсора на следующую строку. Синтаксический формат оператора следующий:

read({имя переменной, которой присваивается значение});

или

readln({имя переменной, которой присваивается значение});

Согласно правилам вежливости необходимо перед каждым оператором «read» или «readln» записывать оператор «write» или «writeln», служащих для вывода на экран/дисплей/монитор побуждающего сообщения, объясняющего пользователю, - значение какой именно переменной он в данный момент вводит. В первом случае побуждающее сообщение выводится без последующего перевода курсора на следующую строку, во втором случае перевод курсора на новую строку производится. Синтаксический формат оператора следующий:

write('{Побуждающее сообщение}');

или

writeln('{Побуждающее сообщение}');

На рис. 4.5. представлен фрагмент программы, на конкретном примере демонстрирующий вышеописанные рекомендации. При этом не стоит забывать о сделанном в пункте 4.1.2 замечании.

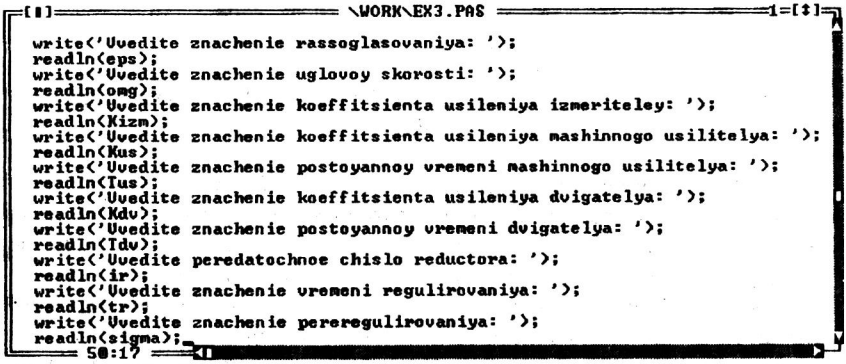


Рисунок 4.5. - Пример ввода исходных данных путём запроса на ввод значений с клавиатуры

Замечание 1.2: если Вами предусмотрен пользовательский ввод значений переменных, то не забудьте объявить переменные в разделе объявления переменных, начинающимся со служебного слова «var» (от английского variable - переменная). Пример такого объявления приведён на рис 4.3. Предпочтительнее задавать вещественные типы (real или extended) для переменных, вводимых пользователем с клавиатуры.

4.1.4. Ввод исходных данных посредством пользовательского ввода ООЯП:

В ООЯП для ввода информации пользователем могут быть использованы различные методы, как прямой, предназначенный для ввода отдельного значения для отдельной переменной с выводом диалогового окна посредством оператора «Input», так и косвенный, - через преобразование текстовой информации, заданной в полях ввода. Второй метод представляется авторам наиболее удобным в использовании и простым в программировании, поскольку не

требуется введение дополнительных инструкций типа «try». Поля ввода являются не только удобным средством ввода исходных данных, но и своеобразной ячейкой памяти для вывода промежуточных значений при установке параметра .Enabled (доступно) равным «False». Хотя, на самом же деле, для корректного предоставления пользователю промежуточной информации призвана к использованию функция «MessageDlg» с использованием дополнительного диалогового окна.

Теперь имеет смысл сказать несколько слов о том, почему авторами изысканы обходные методы без использования операторов и функций с диалоговыми окнами. Задачей в данном случае является получение к конкретному моменту времени не одного, а нескольких значений, которые будут успешно и единовременно отображены в ячейках памяти. С использованием специализированных операторов значение отдельно взятой переменной будет выводиться в своё собственное диалоговое окно с запросом подтверждения. Что при использовании таких операторов для многих переменных усложняет работу, а при отладке становится раздражающим фактором, который в некоторых случаях может приводить к агрессии и к физическим воздействиям на техническое оборудование программистом-разработчиком.

Итак, работать договорились с элементами «TextEdit» в которые задаётся текстовая информация типа «string» (строка). Для использования её в расчётных целях необходимо провести преобразование. В Delphi предусмотрены специальные функции перевода числа, представленного в виде символов-цифр в целое число «StrToInt» или в вещественное число «StrToFloat». Но при этом на тот или иной элемент «TextEdit» необходимо наложить ограничение на ввод тех или иных символов. Если требуется получить целое число, то нужно допустить ввод только тех символов, что соответствуют цифрам «0» - «9». Если же требуется получить вещественное число, то следует допустить помимо символов-цифр символ запятой, как разделяющего символа целой и вещественной части, проще говоря, плавающей точки. Но при этом следует исключить случай ошибочного ввода более, чем одной запятой. Данные рекомендации отражены в последнем фрагменте примера 1.1.

4.2. Перемножение многочленов:

Студентам, для выполнения курсового проекта, предлагается использовать программу перемножения многочленов «PEREMNOG.BAS», составленную Сеславиным А.И. на языке Basic.

Применение программы заключается не в непосредственном её использовании, а в изучении алгоритма, который должен быть внедрён в общую структуру программы, составляемой на языке Pascal или Delphi. Такой метод перевода программ с одного языка на другой получил название «трансляция». Таким образом, студентам предлагается транслировать программу «PEREMNOG.BAS» для дальнейшего её использования, а также для отработки навыков работы с различными языками программирования.

Пример 1.2:

```
const
  {Постоянная размерности массивов}
  m0=15;
  {Постоянные порядков многочленов исходной СУ}
  zv_a=0; {Звенья вида Tp+1 числителя}
  zv_b=3; {Звенья вида Tp+1 знаменателя}
var
  frmPeremnozhenie: TfrmPeremnozhenie;
  K, Kus, Kdv, Iinv, Tus, Tdv, Kobsh: real;
  mas_A, mas_A1, mas_B, mas_B1: array [1..m0] of real;
  Mult_A: array [1..m0] of real;       {Массив коэффициентов
                                        перемноженного многочлена
                                        числителя (от англ.
                                        Multiply)}
  Mult_B: array [1..m0] of real;       {Массив коэффициентов
                                        перемноженного многочлена
                                        знаменателя (от англ.
                                        Multiply)}
  stroka: string;
  flgcorr: boolean;

{Перемножение многочленов}
procedure TfrmPeremnozhenie.btnPeremnozhClick(Sender: TObject);
var
  Order_A: array [1..m0] of integer;     {Массив порядков
                                          перемножаемых
                                          многочленов числителя}
  Order_B: array [1..m0] of integer;     {Массив порядков
                                          Перемножаемых
                                          многочленов
                                          знаменателя}
  A: array [1..m0] of real;              {Массив коэффициентов
                                          многочлена числителя}
  B: array [1..m0] of real;              {Массив коэффициентов
                                          многочлена знаменателя}
  Temp1,Temp2: array [1..m0] of real;    {Рабочие массивы для
                                          числителя}
  Temp3,Temp4: array [1..m0] of real;    {Рабочие массивы для
                                          знаменателя}
```

```
  num_poly_a: integer;                    {Число многочленов
                                           числителя}
  num_poly_b: integer;                    {Число многочленов
                                           знаменателя}
  i,j,p,l,il,jl,pl,ll: integer;          {Переменные цикла}
  mem1,mem2,mem3,mem4: integer;
  Order_Sum_A,Order_Sum_B: integer;
  Main_Order_A,Main_Order_B: integer;
  flgUnmult: boolean;
  flgStill: boolean;
  T1,T2,T3,tau1,tau2,tau3: real;
begin
  {Вычисление общего коэффициента усиления системы}
  Kobsh:=1;
  for i:=0 to m0 do
    Kobsh:=Kobsh*mas_A[i];
  if flgcorr and (rgcorr.ItemIndex = 1) then
    Kobsh:=Kobsh/mas_A[3];
  {Здесь mas_A[3] - Кус}
  ekobsh.Text:=FloatToStrF(Kobsh, ffFixed, 5, 5);
  {Вычисление общего коэффициента усиления системы}
  T1:=StrToFloat(et2.Text);
  T2:=StrToFloat(et4.Text);
  T3:=StrToFloat(et6.Text);
  tau1:=StrToFloat(et1.Text);
  tau2:=StrToFloat(et3.Text);
  tau3:=StrToFloat(et5.Text);
  Kus:=StrToFloat(ekus.Text);
  Tus:=StrToFloat(et5.Text);
  if flgcorr and (rgcorr.ItemIndex = 1) then
    case StrToInt(ecorr.Text) of
      1: {Одно корректирующее звено}
        begin
          mas_A1[1]:=Kus;
          mas_A1[2]:=tau1*Kus;
          for i:=3 to m0 do
            mas_A1[i]:=0;
          for i:=7 to m0 do
            mas_B[i]:=0;
        end;
      2: {Два корректирующих звена}
        begin
          mas_A1[1]:=Kus;
          mas_A1[2]:=tau2*Kus;
          mas_A1[3]:=1;
          mas_A1[4]:=T1;
          for i:=5 to m0 do
            mas_A1[i]:=0;
          mas_B[5]:=1+Kus;
          mas_B[6]:=Tus+T1+(tau2+tau1)*Kus;
          mas_B[7]:=T1*Tus+tau1*tau2*Kus;
          for i:=8 to m0 do
            mas_B[i]:=0;
```

```
      end;
    3:  {Три корректирующих звена}
      begin
        mas_A1[1]:=Kus;
        mas_A1[2]:=tau3*Kus;
        mas_A1[3]:=1;
        mas_A1[4]:=T1;
        mas_A1[5]:=1;
        mas_A1[6]:=T2;
        for i:=7 to m0 do
          mas_A1[i]:=0;
        mas_B[5]:=1+Kus;
        mas_B[6]:=Tus+T1+T2+(tau3+tau2+tau1)*Kus;
        mas_B[7]:=(T1+T2)*Tus+T1*T2+
                  ((tau1+tau2)*tau3+tau1*tau2)*Kus;
        mas_B[8]:=T1*T2*Tus+tau1*tau2*tau3*Kus;
        for i:=9 to m0 do
          mas_B[i]:=0;
      end;
  end;
{Перемножение многочленов числителя и знаменателя}
{--------------Очищение используемых массивов}
flgUnmult:=False;
flgStill:=False;
for i:=1 to m0 do
  begin
    A[i]:=0;
    B[i]:=0;
  end;
for i:=1 to m0 do
  begin
    Temp1[i]:=0;
    Temp2[i]:=0;
    Temp3[i]:=0;
    Temp4[i]:=0;
    Order_A[i]:=0;
    Order_B[i]:=0;
  end;
for i:=1 to m0 do
  begin
    Mult_A[i]:=0;
    Mult_B[i]:=0;
  end;
{--------------Очищение используемых массивов}
{--------------Ввод информации по многочленам}
num_poly_a:=zv_a+StrToInt(ecorr.Text); {Число перемножаемых
                    многочленов числителя с учётом коррекции}
if not (flgcorr and (rgcorr.ItemIndex = 1)) then
  num_poly_b:=zv_b+StrToInt(ecorr.Text)  {Число перемножаемых
                  многочленов знаменателя с учётом коррекции}
else
  num_poly_b:=zv_b;
if num_poly_a = 0 then
```

```
  begin
    flgUnmult:=True;
    num_poly_a:=2;
  end;
if num_poly_a = 1 then
  begin
    flgStill:=True;
    num_poly_a:=2;
  end;
for i:=1 to num_poly_a do
  Order_A[i]:=1;  {Принимаем, что все рассматриваемые
                                        многочлены 1-го порядка}
if not (flgcorr and (rgcorr.ItemIndex = 1)) then
  for i:=1 to num_poly_b do
    Order_B[i]:=1  {Принимаем, что все рассматриваемые
                                        многочлены 1-го порядка}
else
  begin
    for i:=1 to num_poly_b do
      Order_B[i]:=1; {Принимаем, что все (кроме последнего)
                                        многочлены 1-го порядка}
    Order_B[num_poly_b]:=StrToInt(ecorr.Text);
  end;
{--------------Ввод информации по многочленам}
{--------------Подсчёт порядка перемноженного массива}
Main_Order_A:=0;
Main_Order_B:=0;
for i:=1 to num_poly_a do
  Main_Order_A:=Main_Order_A+Order_A[i];
for i:=1 to num_poly_b do
    Main_Order_B:=Main_Order_B+Order_B[i];
for i:=1 to Main_Order_A+num_poly_a do
  A[i]:=mas_A1[i];
for i:=1 to Main_Order_B+num_poly_b do
  B[i]:=mas_B[i];
{--------------Подсчёт порядка перемноженного массива}
{Подготовка первых двух многочленов для перемножения}
for i:=1 to Order_A[1]+1 do
  Temp1[i]:=A[i];
for i:=1 to Order_B[1]+1 do
  Temp3[i]:=B[i];
for i:=1 to Order_A[2]+1 do
  Temp2[i]:=A[Order_A[1]+1+i];
for i:=1 to Order_B[2]+1 do
  Temp4[i]:=B[Order_B[1]+1+i];
mem1:=Order_A[1];
mem2:=Order_A[2];
mem3:=Order_B[1];
mem4:=Order_B[2];
{Подготовка первых двух многочленов для перемножения}
{--------------Перемножение первых двух многочленов числителя}
for j:=1 to mem1+mem2+1 do
  Mult_A[j]:=0;
```

```
for j:=1 to mem3+mem4+1 do
  Mult_B[j]:=0;
for j:=1 to mem1+mem2+1 do
  begin
    i:=j;
    if i > (mem2+1) then
      i:=mem2+1;
    l:=j-mem1;
    if l < 1 then
      l:=1;
    for p:=l to i do
      Mult_A[j]:=Mult_A[j]+Temp2[j-p+1]*Temp1[p];
  end;
{Перемножение первых двух многочленов знаменателя}
for j:=1 to mem3+mem4+1 do
  begin
    i:=j;
    if i > (mem4+1) then
      i:=mem4+1;
    l:=j-mem3;
    if l < 1 then
      l:=1;
    for p:=l to i do
      Mult_B[j]:=Mult_B[j]+Temp4[j-p+1]*Temp3[p];
  end;
{Перемножение первых двух многочленов}
{Подготовка ответа при num_poly=2}
for i:=1 to Order_A[1]+Order_A[2]+1 do
  Temp1[i]:=Mult_A[i];
Order_Sum_A:=Order_A[1]+Order_A[2];
for i:=1 to Order_B[1]+Order_B[2]+1 do
  Temp3[i]:=Mult_B[i];
Order_Sum_B:=Order_B[1]+Order_B[2];
{--------------Подготовка ответа при num_poly=2}
if num_poly_a > 2 then
  begin
    {Подготовка многочленов для перемножения при k>2}
    for i:=3 to num_poly_a do
      begin
        Order_Sum_A:=Order_Sum_A+Order_A[i];
        for j:=1 to Order_A[i]+1 do
          Temp2[j]:=A[Order_Sum_A-Order_A[i]+i-1+j];
        mem2:=Order_Sum_A-Order_A[i];
        mem1:=Order_A[i];
    {Подготовка многочленов для перемножения при k>2}
        {--------------Перемножение двух многочленов}
        for i1:=1 to mem1+mem2+1 do
          Mult_A[i1]:=0;
        for j:=1 to mem1+mem2+1 do
          begin
            i1:=j;
            if i1 > (mem2+1) then
              i1:=mem2+1;
```

```
            l:=j-mem1;
            if l < 1 then
              l:=1;
            for p:=1 to i1 do
              Mult_A[j]:=Mult_A[j]+Temp2[j-p+1]*Temp1[p];
          end;
        {--------------Перемножение двух многочленов}
        {--------------Подготовка ответа при k>2}
        for l:=1 to Order_Sum_A+1 do
          Temp1[l]:=Mult_A[l];
        {--------------Подготовка ответа при k>2}
      end;
  end;
if num_poly_b > 2 then
  begin
    {Подготовка многочленов для перемножения при k>2}
    for i:=3 to num_poly_b do
      begin
        Order_Sum_B:=Order_Sum_B+Order_B[i];
        for j:=1 to Order_B[i]+1 do
          Temp4[j]:=B[Order_Sum_B-Order_B[i]+i-1+j];
        mem4:=Order_Sum_B-Order_B[i];
        mem3:=Order_B[i];
    {Подготовка многочленов для перемножения при k>2}

        {Перемножение двух многочленов}
        for i1:=1 to mem3+mem4+1 do
          Mult_B[i1]:=0;
        for j:=1 to mem3+mem4+1 do
          begin
            i1:=j;
            if i1 > (mem4+1) then
              i1:=mem4+1;
            l:=j-mem3;
            if l < 1 then
              l:=1;
            for p:=1 to i1 do
              Mult_B[j]:=Mult_B[j]+Temp4[j-p+1]*Temp3[p];
          end;
        {Перемножение двух многочленов}
        {Подготовка ответа при k>2}
        for l:=1 to Order_Sum_B+1 do
          Temp3[l]:=Mult_B[l];
        {Подготовка ответа при k>2}
      end;
  end;
{Выдача ответа}
if flgUnmult then
  begin
    Mult_A[i]:=mas_A1[1];
    for i:=2 to m0 do
      Mult_A[i]:=0;
    num_poly_a:=0;
```

```
  eOrd1.Text:=IntToStr(num_poly_a);
end;
 flgStill then
begin
  Mult_A[1]:=mas_A1[1];
  Mult_A[2]:=mas_A1[2];
  for i:=3 to m0 do
    Mult_A[i]:=0;
  num_poly_a:=1;
  eOrd1.Text:=IntToStr(num_poly_a);
end;
)r i:=(Order_Sum_A+1)+1 to m0 do
Mult_A[i]:=0;
)r i:=(Order_Sum_B+1)+1 to m0 do
Mult_B[i]:=0;
a1.Text:=FloatToStrF(Mult_A[1], ffFixed, 7, 7);
a2.Text:=FloatToStrF(Mult_A[2], ffFixed, 7, 7);
a3.Text:=FloatToStrF(Mult_A[3], ffFixed, 7, 7);
a4.Text:=FloatToStrF(Mult_A[4], ffFixed, 7, 7);
1.Text:=FloatToStrF(Mult_B[1], ffFixed, 7, 7);
2.Text:=FloatToStrF(Mult_B[2], ffFixed, 7, 7);
3.Text:=FloatToStrF(Mult_B[3], ffFixed, 7, 7);
4.Text:=FloatToStrF(Mult_B[4], ffFixed, 7, 7);
5.Text:=FloatToStrF(Mult_B[5], ffFixed, 7, 7);
6.Text:=FloatToStrF(Mult_B[6], ffFixed, 7, 7);
7.Text:=FloatToStrF(Mult_B[7], ffFixed, 7, 7);
f not (flgUnmult or flgStill) then
 eOrd1.Text:=IntToStr(Order_Sum_A);
Ord.Text:=IntToStr(Order_Sum_B);
--------------Выдача ответа}
;
```

4.3. Расчёт устойчивости:

Для расчёта устойчивости системы автоматического авления (САУ) студентам предлагаются к использованию два ебраических и два частотных критерия устойчивости.

Хочется заметить, что частотные критерии устойчивости жны для программирования, поскольку являются аналитическими. кие аналитические методы расчёта сложны для программирования, кольку требуют символических преобразований (параметрического чёта). Символические преобразования, в свою очередь должны ить пакетный характер, то есть должны быть учтены все возможные ианты преобразований, из которых должны быть составлены лиотеки. На основе этих библиотек и построена работа того или го пакета программ. В задачу студентов, в рамках курсового екта, не входит составление программного пакета, посему вся

аналитическая часть максимально сводится к численному расч
который годен лишь для узкого круга частных случаев.

4.3.1. Алгебраический критерий устойчивости Рауса

Теоретическое описание метода:

Формулировка: для устойчивости линейной стационар
системы необходимо и достаточно, чтобы коэффициенты пер
столбца таблицы Рауса были положительными. Если это
выполняется, то система неустойчива.

Метод Рауса является представителем семей
алгебраических критериев устойчивости. К достоинствам ме
относятся простая реализация на ЭВМ, а также простота анализа
систем небольшого (до третьего) порядка. К недостаткам ме
можно отнести ненаглядность метода, - по нему сложно суди
степени устойчивости, о её запасах.

Предложен метод в 1877 году английским математиком Э.
Раусом. Целесообразно использовать метод при анализе устойчив
систем высокого порядка (выше третьего).

Работает данный метод с коэффициен
характеристического уравнения ЗАМКНУТОЙ системы
передаточной функцией $W(p)=\frac{A(p)}{B(p)}$, для кот
характеристическое уравнение записывается в виде $B(p)=0$.

В развёрнутом виде $B(p)$ записывается как:

$$B(p)=a_0\cdot p^n+a_1\cdot p^{n-1}+\ldots+a_n$$

Далее метод сводится к составлению таблицы вида:

Таблица 4.1.

Коэффициенты r_i	Номер строки	Номер столбца		
		1	2	3
	1	$c_{1,1} = a_0$	$c_{2,1} = a_2$	$c_{3,1} = a_4$
	2	$c_{1,2} = a_1$	$c_{2,2} = a_3$	$c_{3,2} = a_5$
$r_3 = \frac{a_0}{a_1}$	3	$c_{1,3} = c_{2,1}$ $-r_3 \cdot c_{2,2}$	$c_{2,3} = c_{3,1}$ $-r_3 \cdot c_{3,2}$	$c_{3,3} = c_{4,1}$ $-r_3 \cdot c_{4,2}$
$r_3 = \frac{a_1}{a_{1,3}}$	4	$c_{1,4} = c_{2,2}$ $-r_4 \cdot c_{2,3}$	$c_{2,4} = c_{3,2}$ $-r_4 \cdot c_{3,3}$	$c_{3,4} = c_{4,2}$ $-r_4 \cdot c_{4,3}$
$r_5 = \frac{c_{1,3}}{c_{1,4}}$	5	$c_{1,5} = c_{2,3}$ $-r_5 \cdot c_{2,4}$	$c_{2,5} = c_{3,3}$ $-r_5 \cdot c_{3,4}$	$c_{3,5} = c_{4,3}$ $-r_5 \cdot c_{4,4}$
…	…	…	…	…
$r_i = \frac{c_{1,i-2}}{c_{1,i-1}}$	i	$c_{1,i} = c_{2,i-2}$ $-r_i \cdot c_{2,i-1}$	$c_{2,i} = c_{3,i-2}$ $-r_i \cdot c_{3,i-1}$	$c_{3,i} = c_{4,i-2}$ $-r_i \cdot c_{4,i-1}$
…	…	…	…	…

4.3.2. Алгебраический критерий устойчивости Гурвица

Теоретическое описание метода:

Критерий сформулирован и доказан в 1895 году немецким математиком А. Гурвицем. Критерий устойчивости Гурвица позволяет получать аналитические выражения для исследования влияния какого-либо параметра (параметров) на устойчивость системы.

Формулировка: система устойчива по критерию Гурвица, если в процессе составления матрицы при положительности коэффициентов характеристического уравнения $a_0, a_1, ..., a_n$ все n определителей Гурвица $\Delta_0, \Delta_1, ..., \Delta_n$, составленные по определённой схеме, положительны. Если хотя бы один из определителей Гурвица отрицательный, то система неустойчива.

Матрица по которой вычисляются определители Гурвица составляется следующим образом:

1. На главной диагонали записываются все коэффициенты с a_1 до a_n;

2. В каждом столбце выше диагональных элементов записываются коэффициенты с последовательно возрастающими индексами, а ниже – с последовательно убывающими индексами;

3. На место коэффициентов с индексами больше n или меньше нуля проставляются нули.

$$\Delta = \begin{vmatrix} a_1 & a_3 & a_5 & a_7 & a_9 & \dots & 0 \\ a_0 & a_2 & a_4 & a_6 & a_8 & \dots & 0 \\ 0 & a_1 & a_3 & a_5 & a_7 & \dots & 0 \\ 0 & a_0 & a_2 & a_4 & a_6 & \dots & 0 \\ 0 & 0 & a_1 & a_3 & a_5 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_n \end{vmatrix}.$$

4.3.3. Частотный критерий устойчивости Найквиста

Теоретическое описание метода:

Устойчивость ЗАМКНУТОЙ системы гарантирована в том случае, когда все корни характеристического уравнения РАЗОМКНУТОЙ системы лежат в левой полуплоскости комплексной плоскости корней (действительные - вдоль отрицательной части действительной оси, комплексные – попарно симметричные относительно отрицательной части действительной оси), и при этом годограф системы не охватывает точку Найквиста (-1; 0j). Охваты считаются левее точки Найквиста и классифицируются, как показано на рис. 4.6.