

ПРИЛОЖЕНИЕ

Здесь приводится текст кода всех основных и значимых модулей программы STE в следующем порядке:

- STEtraject – главный модуль;
- STEaddTrj – добавления траекторий;
- STEoptions – модуль основных параметров;
- OurPlotting – модуль построения графиков;
- Constants – модуль определения постоянных величин;
- Solver – модуль расчета траекторий частиц;
- AdvOptToOurPlot – дополнительные опции к модулю графики;
- Limit – установка ограничений на вывод траекторий по перечной координате или скорости;
- ModelDechan – выбор модели деканализирования, по расстоянию наибольшего сближения или по потерям энергии, непосредственно расчет по моделям деканализирования реализован модуле OurPlotting и STEtraject;
- SpecialLoadFails – специальная загрузка больших по объему данных файлов;
- NormalDistribution – модуль установки Гауссового распределения;
- Plotting – модуль расчета потерь энергии.

unit STEtraject;

interface

uses

Constants, Solver, Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, StdCtrls, ComCtrls, ClipBrd, STEaddTrj, ExtDlgs, AboutProgram, Buttons, Gauges, Spin, ShellCtrls, Menus, Math, ImgList, ToolWin, NormalDistribution, DB, DBClient, DBXpress, FMTBcd, SqlExpr, ZAbstractRODataset, ZAbstractDataset, ZDataset, ZConnection, ZAbstractTable, XPMan, TeeProcs, TeeDraw3D;

type

TForm1 = **class**(TForm)
 Timer1: TTimer;
 LabeledEdit1: TLabeledEdit;
 LabeledEdit2: TLabeledEdit;
 SavePictureDialog1: TSavePictureDialog;
 Label1: TLabel;
 LEMultX: TLabeledEdit;
 LEMultY: TLabeledEdit;
 LECenX: TLabeledEdit;
 LECenY: TLabeledEdit;
 CheckBox1: TCheckBox;
 Label2: TLabel;
 LEGraphT0: TLabeledEdit;
 LEGraphTLast: TLabeledEdit;
 LEEpsilon: TLabeledEdit;
 LEDeltaTheta: TLabeledEdit;
 LECalcPeriodFrom: TLabeledEdit;
 LECalcPeriodTo: TLabeledEdit;
 LEortL: TLabeledEdit;
 LEortK: TLabeledEdit;
 Edit1: TEdit;
 Label5: TLabel;
 Gauge1: TGauge;
 Edit2: TEdit;
 Label7: TLabel;
 Label6: TLabel;
 SaveDialog1: TSaveDialog;
 OpenDialog1: TOpenDialog;
 LEEnergy: TLabeledEdit;
 MainMenu1: TMainMenu;

N1: TMenuItem;
N2: TMenuItem;
N3: TMenuItem;
N4: TMenuItem;
N5: TMenuItem;
dat1: TMenuItem;
UxUxtdat1: TMenuItem;
N8: TMenuItem;
NCopyToClipboard: TMenuItem;
NSaveToFile: TMenuItem;
SolveTrBtn: TBitBtn;
ComboPlosk: TComboBox;
LEAEM: TLabeledEdit;
ComboParticles: TComboBox;
LEnx: TLabeledEdit;
H1: TMenuItem;
Vmolier: TMenuItem;
Vmolierz: TMenuItem;
LEzsmall: TLabeledEdit;
BitBtn1: TBitBtn;
N15: TMenuItem;
YesGame: TMenuItem;
NoGame: TMenuItem;
Optionsgame: TMenuItem;
Y1: TMenuItem;
Label3: TLabel;
N9: TMenuItem;
N18: TMenuItem;
N19: TMenuItem;
N20: TMenuItem;
N21: TMenuItem;
LabeledEdit4: TLabeledEdit;
Label9: TLabel;
ContinueCalculation: TCheckBox;
LEZ1: TLabeledEdit;
CoolBar1: TCoolBar;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
ToolBar2: TToolBar;
AddTrBtn: TToolButton;
Panel1: TPanel;
LabeledEdit3: TLabeledEdit;
ToolButton9: TToolButton;

ComboCrystal: TComboBox;
 LEDeltaThetaInEps: TLabeledEdit;
 Label4: TLabel;
 Label8: TLabel;
 Vx1: TMenuItem;
 TrAction1: TLabel;
 SQLConnection1: TSQLConnection;
 SQLQueryMSDB1: TSQLQuery;
 SQLQueryMSDB2: TSQLQuery;
 SQLQueryMSdl0: TSQLQuery;
 SQLQueryMSdl1: TSQLQuery;
 SQLQueryMSdl2: TSQLQuery;
 SQLQueryMSdl3: TSQLQuery;
 SQLQueryMS: TSQLQuery;
 SQLDataSetMSdl: TSQLDataSet;
 SQLDataSetMSdb: TSQLDataSet;
 ZQueryMS: TZQuery;
 ZQueryMSdl0: TZQuery;
 ZQueryMSdl1: TZQuery;
 ZQueryMSdl2: TZQuery;
 ZQueryMSdl3: TZQuery;
 ZQueryMSDB1: TZQuery;
 ZQueryMSDB2: TZQuery;
 ZTableMSdl: TZTable;
 ZTableMSdb: TZTable;
 Timer2: TTimer;
 XPManifest1: TXPManifest;
 ZQueryMSothers: TZQuery;
 SaveFileProperties: TMenuItem;
 LoadFileProperties: TMenuItem;
 OpenFileDialog2: TOpenDialog;
 SaveDialog2: TSaveDialog;
 ToolButton11: TToolButton;
 Button3: TButton;
 GroupBox1: TGroupBox;
 GroupBox2: TGroupBox;
 Label10: TLabel;
 Label11: TLabel;
 Label12: TLabel;
 GroupBox3: TGroupBox;
 TrAction: TStatusBar;
 LEMySQL: TLabeledEdit;
 Panel2: TPanel;

```

ZQuery1: TZQuery;
ZQueryMSdb: TZQuery;
ZQueryMSdbid: TIntegerField;
ZQueryMSdbdtrajectory: TIntegerField;
ZQueryMSdbduzel: TIntegerField;
ZQueryMSdbdv: TFloatField;
ZQueryMSdbdx: TFloatField;
ZQueryMSdbde: TFloatField;
ZQueryMSdbdf: TFloatField;
ZQueryMSdbIdCount: TIntegerField;
ZQueryCountid: TZQuery;
ZQueryCountidcountid: TIntegerField;
ToolButton3: TToolButton;
ToolButton4: TToolButton;
Draw3D1: TDraw3D;
Panel3: TPanel;
N6: TMenuItem;
Sep5: TToolButton;
AbFilesButton: TToolButton;
NSpecLoad: TMenuItem;
procedure OnCreate(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure SolveTrBtnClick;
procedure AddTrBtnClick(Sender: TObject);
procedure SaveTrDatBtnClick(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure LEEpsilonChange(Sender: TObject);
procedure AboutBtnClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure LEEnergyChange(Sender: TObject);
procedure CChoicePlane(Sender: TObject);
procedure CChoiceParticles(Sender: TObject);
procedure ChangeFlying(Sender: TObject);
procedure VmolierClick(Sender: TObject);
procedure VmolierzClick(Sender: TObject);
procedure FastAddTrajectory(Sender: TObject);
procedure YesGameClick(Sender: TObject);
procedure NoGameClick(Sender: TObject);
procedure OptionsgameClick(Sender: TObject);
procedure ChangeOptions;
procedure FormShow(Sender: TObject);
procedure NPlottingClick(Sender: TObject);
procedure SaveAllTrajectory(b: byte; w: string);

```

```

procedure LoadAllTrajectory(Sender: TObject);
procedure ToolButton7Click(Sender: TObject);
function CreateDirForTrj:string;
function CreateInscription:string;
procedure SaveTrj(x:integer; w:string);
procedure BitBtn2Click(Sender: TObject);
procedure SaveMenuTrj(Sender: TObject);
procedure CChoiceCrystal(Sender: TObject);
procedure LabeledEdit1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
procedure Vx1Click(Sender: TObject);
procedure UpdateInitialConditions(a: integer);
procedure InitialConditions;
procedure SaveAllTrajectoryMSToDB(NameDBtoinsert:string;SaveDialogActive:boolean;
    N_Save:boolean;N_point:integer;N_point_Amount:integer;Layer_Save:boolean;Layer_ReSave:boo
    lean);
procedure SaveAllTrajectoryMS-
    ToDBdl(Namedltoinsert:string;SaveDialogActive:boolean;table_type:string);
procedure InsertDa-
    taMS1(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer;
        DataToFieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double);
procedure InsertDa-
    taMS1vxe(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer;
        DataToFieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double);
procedure InsertDa-
    taMS2(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer;
        DataTo-
        FieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double;DataToFieldMS5:double);
procedure InsertDa-
    taMS2vxe(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer;
        DataTo-
        FieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double;DataToFieldMS5:double);
procedure InsertDataMS0dl(NamedltoInsert:string;DataToFieldMS0:string);
procedure InsertDataMS1dl(NamedltoInsert:string;DataToFieldMS1:integer);
procedure InsertDataMS2dl(NamedltoInsert:string;DataToFieldMS2:integer);
procedure InsertDataMS3dl(NamedltoInsert:string;DataToFieldMS3:double);
procedure CreateMSdl(NameDB:string);
procedure CreateMSdb(NameDB:string);
procedure CreateMSDataBase(NameDB:string);
function LoadTrajectoryMSdl0:string;
function LoadTrajectoryMSdl1:integer;
function LoadTrajectoryMSdl2:boolean;

```

```

function LoadTrajectoryMSdl3:double;
function FloatForAccess(N:Variant):String;
function DateForMySQL(D:Variant):string;
function VarToSQL(AValue:Variant):string;
procedure Timer2Timer(Sender: TObject);
procedure Solving_Energy_Loss(ModelDech: integer; Dechennaling_ with podsloi, Normaliza-
tion: boolean; SEL_Nsloev, SEL_Nm, SEL_Nmax: integer;
    SEL_Lev_dE, SEL_Prav_dE, Normalization_Value: double; SEL_FileName:string);
procedure STE_Refreshment;
procedure LoadFilePropertiesClick(Sender: TObject);
procedure SaveFilePropertiesClick(Sender: TObject);
procedure Panel2Db1Click(Sender: TObject);
procedure LEMySQLChange(Sender: TObject);
procedure DatasetNewFields;
procedure DestroyAllData(Sender: TObject);
procedure ShowActiveConnectionToMySQL;
procedure Button6Click(Sender: TObject);
procedure CreateOurConnection(strDb,strDataDir:string);
procedure DestroyOurConnection;
procedure AbFilesButtonClick(Sender: TObject);
procedure AbFilesShow(DirTab,Tab1,Tab2:string);
procedure SaveFileInitIni(SFileNameIni: string);
procedure LoadFileInitIni(LFileNameIni: string);
function SaveBoolIni(BoolValue: boolean):string;
function LoadBoolIni(sBoolValue: string):boolean;
function SaveIntIni(IntValue: integer):string;
function LoadIntIni(sIntValue: string):integer;
procedure Load_STE_File(SpLoadTr: boolean; tmpOpenFileName: string);
procedure ReStartProgram(tmpLoad: boolean; FullDataPath: string; TopPosition,LeftPosition:
integer);
procedure LoadParametr;
function DeleteVremDir(PathVr: string): boolean;
function CreateVremDir(PathVr: string): boolean;
procedure FormActivate(Sender: TObject);
procedure NSpecLoadClick(Sender: TObject);
procedure CompatibilityMyFiles(FullDirMyTable: string; ArrMyTables: TStrings);

private
    { Private declarations }
    FDataset: TZQuery;
    FConnection: TZConnection;
    CDSInitialization: TClientDataSet;

```

```

FileLoadPath: string;
SaveDialogFile: TSaveDialog;
a_tau, b_tau: double;
R_tau, cnt_SpLoad: integer;

```

public

```

Procedure SetCoeff;
{ Public declarations }
property Dataset: TZQuery read FDataset write FDataset;
property Connection: TZConnection read FConnection write FConnection;
end;

```

```

dblptr=^double;
blnptr=^boolean;

```

```

ProgressControl=object
  ind:array of dblptr;
  indcs:array of blnptr;
  ThreadCount:integer;
  interval:integer;
  Current:integer;
constructor Init(thrcnt:integer);
destructor Done;
  Procedure ActivateProgressControl;
  Procedure DeActivateProgressControl;
  Procedure ProcessRefresh;
end;

```

```

TAllTraject=object
  Traject:array of OneTrajectory;
  CntOfTraject:integer;
  Progress:ProgressControl;
constructor Init(calculus:integer;xlayer,vlayer,stsloi:array of
double;amount:integer;ser:PSeriaObj;Stdepth,depth,lRange,rRange:double;NodeCnt:integer);
destructor Done;
procedure SolveAll;
end;

```

var

```

Form1: TForm1;
Trajects: TAllTraject;
MultX,MultY,GraphT0,GraphTLast: double;
CenX,CenY: integer;

```



```

inscription,CurrentDirTrj: string;
calc: integer;
CntGlobal: integer;
OuterLoad: boolean;
FirstLoad: boolean;
Save_DBdl,Create_DB: byte;
DirProgram: string;
LEalpha,LEbeta: string;
LoadMyDataDone, SolveMyDataDone: boolean;
N_step_uzel: integer;

```

implementation

uses DialogToSave, STEoptions, ThreePointBending, OurPlotting,
 AdvOptToOurPlot, AboutFiles, IniFiles, ShellApi, ModelDechan,
 SpecialLoadFails, Plotting;

```
{ $R *.dfm }
```

var

```

Big_J, Cnt_Originally, d_int: integer;
LastLayer, Insert_Data_MS_sloi: boolean;
LastLayer_Value: double;
SEL_XY_was_first: boolean;
Number_automatic_step, Inversely_Number_automatic_step: integer;
TDirMySQL, TDataDirMySQL: string;
VisibleMySQLDir: boolean;
CrystalSolving, Z1Solving: string;
Edit_from_FormPlotting: array[1..9] of string;
Edit_from_BendingForm: array[1..12] of string;
Edit_from_optiongamev: array[1..3] of string;
VXTr_from_AddTrj: array[1..5] of string;
Form11_Checked: array[1..4] of boolean;
FOptions_Checked: array[1..2] of boolean;
Edit_from_PlotF: array[1..2] of string;

```

Procedure TForm1.SetCoeff;

var

```

i,code:integer;
r,rr:double;
s,sr:string;

```

begin

```
s:=LEGraphT0.Text;
```

```

Val(s,r,code);
GraphT0:=r;
s:=LabeledEdit4.Text;
Val(s,rr,code);
GraphT0:=GraphT0-rr;
s:=LEGraphTLast.Text;
Val(s,r,code);
GraphTLast:=r-rr;
s:=LEMultX.Text;
Val(s,r,code);
MultX:=r;
s:=LEMultY.Text;
Val(s,r,code);
MultY:=r;
s:=LECenX.Text;
Val(s,i,code);
CenX:=i;
s:=LECenY.Text;
Val(s,i,code);
CenY:=i;
s:=LEortk.Text;
Val(s,i,code);
k:=i;
s:=LEortl.Text;
Val(s,i,code);
l:=i;
InitialConditions;
end;

procedure TAllTraject.SolveAll;
var
  i:integer;
begin
  Progress.ActivateProgressControl;
  for i:=0 to (CntOfTraject-1) do
    with Traject[i] do
      begin
        Progress.Current:=i;
        while not Solved do
          begin
            Solve;
            Progress.ProcessRefresh;
            Application.ProcessMessages;

```

```

    end;
end;
Progress.DeActivateProgressControl;
end;

constructor TAllTraject.Init(calculus:integer;xlayer,vlayer,stsloi:array of
double;amount:integer;ser:PSeriaObj;Stdepth,depth,lRange,rRange:double;NodeCnt:integer);
var
    i,j,k:integer;
    x,v:double;
begin
    k:=-1;
    CntOfTraject:=ser^.AllTrCnt;
    SetLength(Traject,CntOfTraject);
    Progress.Init(CntOfTraject);
if calculus=0 then
    begin
        for i:=0 to (ser^.SeriaCnt-1) do
            begin
                if (ser^.Seria[i].count > 1) then
                    for j:=1 to ser^.Seria[i].count do
                        begin
                            inc(k);
                            x:=ser^.Seria[i].x1st+(j-1)*(ser^.Seria[i].x2nd-ser^.Seria[i].x1st)/(ser^.Seria[i].count-1);
                            if VGauss=false then
                                v:=ser^.Seria[i].v1st+(j-1)*(ser^.Seria[i].v2nd-ser^.Seria[i].v1st)/(ser^.Seria[i].count-1)
                            else
                                begin
                                    v:=0;
                                end;
                            Traject[k].Init(Stdepth,depth,lRange,rRange,NodeCnt,v,x,stsloi[k]);
                            Progress.ind[k]:= @Traject[k].PartDone;
                            Progress.indcs[k]:= @Traject[k].CanSolve;
                        end
                    else
                        begin
                            inc(k);
                            Tra-
                                ject[k].Init(Stdepth,depth,lRange,rRange,NodeCnt,ser^.Seria[i].v1st,ser^.Seria[i].x1st,stsloi[k]);
                            Progress.ind[k]:= @Traject[k].PartDone;
                            Progress.indcs[k]:= @Traject[k].CanSolve;
                        end;
                    end;
                end;
            end;
        end;
    end;

```

```

end
else if calculus=1 then
  for j:=0 to amount-1 do
    begin
      inc(k);
      Traject[k].Init(Stdepth,depth,lRange,rRange,NodeCnt,vlayer[k],xlayer[k],stsloi[k]);
      Progress.ind[k]:= @Traject[k].PartDone;
      Progress.indcs[k]:= @Traject[k].CanSolve;
    end
  end;

destructor TAllTraject.Done;
var
  k:integer;
begin
  for k:=0 to (CntOfTraject-1) do
    Traject[k].Done;
  Finalize(Traject);
  Progress.Done;
end;

Procedure ProgressControl.ProcessRefresh;
var
  i:integer;
  d:double;
begin
  d:=0;
  for i:=0 to (ThreadCount-1) do
    if (ind[i] <> nil) then
      d:=d+ind[i]^;
  d:=d/ThreadCount;
  Form1.Gauge1.Progress:=round(d*100);
  indcs[Current]^:=true;
end;

constructor ProgressControl.Init(thrcnt:integer);
var
  i:integer;
begin
  ThreadCount:=thrcnt;
  interval:=100;
  SetLength(ind,ThreadCount);
  SetLength(indcs,ThreadCount);

```

```

for i:=0 to ThreadCount-1 do
    begin
        ind[i]:=nil;
        indcs[i]:=nil;
    end;
    DeActivateProgressControl;
end;

destructor ProgressControl.Done;
begin
    Finalize(ind);
    Finalize(indcs);
    ThreadCount:=0;
end;

Procedure ProgressControl.ActivateProgressControl;
begin
    Form1.Timer1.Interval:=interval;
    Form1.Timer1.Enabled:=true;
end;

Procedure ProgressControl.DeActivateProgressControl;
begin
    Form1.Timer1.Enabled:=false;
end;

procedure TForm1.OnCreate(Sender: TObject);
var
    s:string;
    i,code:integer;
    r:double;
    SetEd: TextFile;
    Text: string;
begin
    DecimalSeparator := '.';
    Form1.Left := 151;
    Form1.Top := 99;
    Form1.Caption := 'STE. Моделирование траекторий частиц в кристаллах';
    OutPathForDataBack := './data';
    OutPathForData := ExtractFilePath(Application.ExeName) + 'data';

    VisibleMySqlDir:=True;

```

```

TDirMySql:=LEMySql.Text;
getdir(0, DirProgram);
l:=1;k:=1;tipV:=0;
UpdateInitialConditions(0);
try
  LoadFileInitIni('initial' + '.ini');
except
  MessageDlg('Файл загрузки начальных установок не доступен, будут применены типовые
установки',
mtWarning, [mbOk], 0);
  ComboCrystal.Text:='Ge';
  ComboPlosk.Text:='110';
  LEZ1.Text:='1';
  ComboParticles.Text:='Другое';
  LEAEM.Text:='1.00728';
  LEnx.Text:='40';
  LEzsmall.Text:='15';
  LabeledEdit1.Text:='3';
  LabeledEdit2.Text:='3000';
  LabeledEdit3.Text:='1';
  LabeledEdit4.Text:='0';
  LEEnergy.Text:='450E+9';
  LEEpsilon.Text:='9.32353637872253E-0006';
  LEDeltaTheta.Text:='2';
  LECalcPeriodFrom.Text:='-1E+10';
  LECalcPeriodTo.Text:='1E+10';
  LEortk.Text:='1';
  LEortl.Text:='1';
  Edit1.Text:='0';
  Edit2.Text:='0';
  LEMultX.Text:='1206.520';
  LEMultY.Text:='701.598';
  LECenX.Text:='0';
  LECenY.Text:='1239';
  porog_max_eV2:=0.0025;
  porog_min_eV2:=0.0001;
  shag_A:=5900000000;
  min_shag_A:=650;
  gamev:=1;
  tipV:=0;
  OverBarrierOff:=0;
  BorOff:=1;
  SrKvFlEp:=1;

```

```

    VMolierOff:=0;
    EnergyLossVariant4Off:=false;
end;
    SavePictureDialog1.DefaultExt:='*.wmf';
    SetCoeff;
end;

procedure TForm1.ChangeOptions;
var
s:string;
begin
    if gamev=1 then
        begin
            Form1.YesGame.Checked:=True;
            Form1.NoGame.Checked:=False;
            FormPlotting.BitBtn13.Enabled:=False;
            FormPlotting.BitBtn3.Enabled:=True;
            FormPlotting.BitBtn4.Enabled:=False;
            FormOptions.CheckBox18.Checked := True;
        end
        else if gamev=0 then
            begin
                Form1.NoGame.Checked:=True;
                Form1.YesGame.Checked:=False;
                FormPlotting.BitBtn13.Enabled:=True;
                FormPlotting.BitBtn3.Enabled:=False;
                FormPlotting.BitBtn4.Enabled:=False;
                FormOptions.CheckBox18.Checked := False;
            end;
        if tipV=0 then
            begin
                Vmolier.Checked:=True;
                Vmolierz.Checked:=False;
                LEzsmall.Visible:=False;
            end
            else if tipV=1 then
                begin
                    Vmolierz.Checked:=True;
                    Vmolier.Checked:=False;
                    LEzsmall.Visible:=True;
                end;
        if OverBarrierOff=1 then FormOptions.CheckBox1.Checked:=True
        else if OverBarrierOff=0 then

```

```

begin
  FormOptions.CheckBox1.Checked:=False;
  FormOptions.CheckBox1Click(FormOptions.CheckBox1);
end;

if BorOff=1 then FormOptions.CheckBox2.Checked:=True
else if BorOff=0 then
  begin
    FormOptions.CheckBox2.Checked:=False;
    FormOptions.CheckBox2Click(FormOptions.CheckBox2);
  end;

if SrKvFIEp=1 then FormOptions.CheckBox3.Checked:=True
else if SrKvFIEp=0 then
  begin
    FormOptions.CheckBox3.Checked:=False;
    FormOptions.CheckBox3Click(FormOptions.CheckBox3);
  end;

if VMoilerOff=1 then FormOptions.CheckBox4.Checked:=True
else if VMoilerOff=0 then
  begin
    FormOptions.CheckBox4.Checked:=False;
    FormOptions.CheckBox4Click(FormOptions.CheckBox4);
  end;

if EnergyLossVariant4Off=true then FormOptions.CheckBox5.Checked:=True
else if EnergyLossVariant4Off=false then
  begin
    FormOptions.CheckBox5.Checked:=False;
    FormOptions.CheckBox5Click(FormOptions.CheckBox5);
  end;

FormOptions.LabeledEdit1.Text:=FloatToStr(porog_max_eV2);
FormOptions.LabeledEdit2.Text:=FloatToStr(porog_min_eV2);
FormOptions.LabeledEdit3.Text:=FloatToStr(shag_A);
FormOptions.LabeledEdit4.Text:=FloatToStr(min_shag_A);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  with Trajects.Progress do
    indcs[Current]^:=false;
end;

function TForm1.CreateVremDir(PathVr: string): boolean;
var
  CreateVr: boolean;

```



```

begin
  CreateVr := DirectoryExists(PathVr);

  if CreateVr then Result := True;

  if CreateVr = False then
    begin
      if not CreateDir(PathVr) then
        begin
          raise Exception.Create('Невозможно создать ' + PathVr);
          Exit;
        end
      else Result := True;
    end;
end;

function TForm1.DeleteVremDir(PathVr: string): boolean;
begin
  if RemoveDir(PathVr) then Result := True;
end;

procedure TForm1.SolveTrBtnClick;
var
  amnt,cnt,code,ass,i:integer;
  stdep,dep,dopdep,left,right:double;
  s,sdir,sdirs:string;
  sttime,entime:TDateTime;
  xla,vla,start_sloi:array of double;
  automatic,j:integer;
  p_r:array of integer;
  found:boolean;
  Temp_x:array of double;
  tay3x1,tay3x2,tay3x3,tay3x4:integer;
  SaveF: TextFile;
  FileName_Solving_Exit, w, wFileName: string;
  left_dE, right_de: double;
  PathVrem: string;
  CreateVrem: boolean;
begin
  if (Auto_Save) and (TrDataFormat=4) then
    begin
      PathVrem := OutPathForData + '\vrem';
      CreateVrem := CreateVremDir(PathVrem);

```

```

if not CreateVrem then
  begin
    MessageDlg('Нет доступа для создания временной папки.', mtInformation, [mbOk], 0);
    Exit;
  end;
try
  CreateOurConnection('vrem', OutPathForDataBack);
  Connection.Connected := True;
except
  MessageDlg('Ошибка при подключении к временной папке', mtError, [mbOk], 0);
  Exit;
end;
try
  w:=FormOptions.LEDBName.Text;
  w:=ExtractFileName(w);
  CreateMSDataBase(w);
  wFileName:=w;
except
  MessageDlg('Неверное значение имени файла куда предполагается сохранение
данных', mtError, [mbOk], 0);
  exit;
end;
end;
N_Point_Save_Value_Str:=FormOptions.LabeledEdit6.Text;
Auto_Solving_**with**_step:=FormOptions.LabeledEdit5.Text;
Insert_Data_MS_sloi:=False;
LastLayer:=False;
Inversely_Number_automatic_step:=0;
stime:=Now;
InitialConditions;
Curv:=StrToFloat(Edit1.Text)*d_small*1E+10/(Sqr(epsilon)*1E+10);
s:=Edit2.Text;
Val(s,kt,code);
s:=LabeledEdit2.Text;
Val(s,cnt,code);
CntGlobal:=cnt;
s:=LECalcPeriodFrom.Text;
Val(s,left,code);
s:=LECalcPeriodTo.Text;
Val(s,right,code);
s:=LEortL.Text;
Val(s,l,code);
s:=LEortK.Text;

```

```

Val(s,k,code);
s:=FormOptions.LabeledEdit1.Text;
Val(s,porog_max_eV2,code);
s:=FormOptions.LabeledEdit2.Text;
Val(s,porog_min_eV2,code);
s:=FormOptions.LabeledEdit3.Text;
Val(s,shag_A,code);
s:=FormOptions.LabeledEdit4.Text;
Val(s,min_shag_A,code);
if VGauss=True then
  begin
    Randomize;
    SetLength(p_r,Allseria.AllTrCnt);
    p_r[0]:=RandomRange(0,Allseria.AllTrCnt);
    for i:=1 to Allseria.AllTrCnt-1 do
      repeat
        j:=0;
        found:=false;
        p_r[i]:=RandomRange(0,Allseria.AllTrCnt);
        repeat
          if p_r[i]=p_r[j] then
            if j=i then j:=j+1
            else found:=True else j:=j+1;
          until (j>Allseria.AllTrCnt-1) or (found=True) or (j>i);
        until found=False;
      end;
    FileName_Solving_Exit:='Exit.txt';
    try
      if Auto_Solving_Exit then
        begin
          AssignFile(SaveF, FileName_Solving_Exit);
          Rewrite(SaveF);
          CloseFile(SaveF);
          if sublayer_on_formoptions then sublay-
er_on_formoptions_Value:=StrToInt(FormOptions.Edit14.Text);
          if normalization_on_formoptions then normaliza-
tion_on_formoptions_Value:=StrToFloat(FormOptions.Edit15.Text);
          left_dE:=0;
          right_dE:=StrToFloat(FormOptions.Edit13.Text);
        end;
      except
        MessageDlg('Ошибка. Возможно открыт файл данных потерь энергии. Закройте файл. Пе-
резапустите программу.', mtError, [mbOk], 0);

```

```

Exit;
end;
if (Auto_Solving=False) and (ContinueCalculation.Checked=False) and
(Three_point_bending=False) then
begin
calc:=0;
stdep:=0;
SetLength(start_sloi,Allseria.AllTrCnt);
for i:=0 to Allseria.AllTrCnt-1 do start_sloi[i]:=stdep;
s:=LabeledEdit1.Text;
Val(s,dep,code);
LastLayer_Value:=dep;
Cnt_Originally:=cnt;
step_on_tau:=dep;
CrystalSolving:=ComboCrystal.Text;
Z1Solving:=LEZ1.Text;
Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
if VGauss=True then
begin
SetLength(Temp_x,Allseria.AllTrCnt);
for i:=0 to Allseria.AllTrCnt-1 do
Temp_x[i]:=Trajects.Traject[i].x_arr[0];
for i:=0 to Allseria.AllTrCnt-1 do
Trajects.Traject[i].x_arr[0]:=Temp_x[p_r[i]];
Finalize(Temp_x);
for i:=0 to Allseria.AllTrCnt-1 do
Trajects.Traject[i].v_arr[0]:=v_ND[i];
end;
Trajects.SolveAll;
LastLayer:=True;
FormPlotting.TntBitBtn2Click(Self);
inscription:=CreateInscription;
ContinueCalculation.Enabled:=True;
AddTrBtn.Enabled:=False;
BitBtn1.Enabled:=False;
if Auto_Solving_Exit then
Solving_Energy_Loss(ModelDechanneling, sublayer_on_formoptions, normaliza-
tion_on_formoptions,
sublayer_on_formoptions_Value, Trajects.CntOfTraject, Trajects.Traject[0].Nmax,
left_dE, right_dE, normalization_on_formoptions_Value, FileName_Solving_Exit);
if Auto_Save then Form1.SaveAllTrajectory(2, wFileName)
else OKBottomDlg1.ShowModal
end

```

```

else if (FormOptions.CheckBox6.Checked=False) and (ContinueCalculation.Checked=True) then
  begin
    calc:=1;
    SetLength(vla,Trajects.CntOfTraject);
    SetLength(xla,Trajects.CntOfTraject);
    SetLength(start_sloi,Trajects.CntOfTraject);
    amnt:=Trajects.CntOfTraject;
    stdep:=Trajects.Traject[0].t_arr[Trajects.Traject[0].Nmax];
    s:=FloatToStr(RoundTo(stdep,-2));
    LabeledEdit4.Text:=s;
    s:=LabeledEdit1.Text;
    Val(s,dep,code);
    s:=LabeledEdit3.Text;
    Val(s,dopdep,code);
    dep:=dep+dopdep;
    s:=FloatToStr(RoundTo(dep,-2));
    LabeledEdit1.Text:=s;
    for i:=0 to Trajects.CntOfTraject-1 do
      begin
        vla[i]:=Trajects.Traject[i].v_arr[Trajects.Traject[0].Nmax];
        xla[i]:=Trajects.Traject[i].x_arr[Trajects.Traject[0].Nmax];
        start_sloi[i]:=Trajects.Traject[i].sloi;
      end;
    Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
    Trajects.SolveAll;
    FormPlotting.TntBitBtn2Click(Self);
    inscription:=CreateInscription;
    AddTrBtn.Enabled:=False;
    BitBtn1.Enabled:=False;
    if Auto_Save then Form1.SaveAllTrajectory(2, wFileName)
      else OKBottomDlg1.ShowModal
    end
  else if (Auto_Solving) and (Three_point_bending = False) then
    begin
      calc:=0;
      stdep:=0;
      SetLength(start_sloi,Allseria.AllTrCnt);
      for i:=0 to Allseria.AllTrCnt-1 do start_sloi[i]:=stdep;
      s:=FormOptions.LabeledEdit5.Text;
      Val(s,dopdep,code);
      step_on_tau:=dopdep;
      s:=LabeledEdit1.Text;
      Val(s,dep,code);

```

```

LastLayer_Value:=Round(dep/step_on_tau);
Cnt_Originally:=cnt;
cnt:=Round(cnt*dopdep/dep);
CntGlobal:=cnt;
automatic:=Round(dep/dopdep);
Number_automatic_step:=automatic;
dep:=dopdep;
s:=FloatToStr(RoundTo(dep,-2));
LabeledEdit1.Text:=s;
LabeledEdit2.Text:=IntToStr(CntGlobal);
CrystalSolving:=ComboCrystal.Text;
Z1Solving:=LEZ1.Text;
Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
if VGauss=True then
    begin
        SetLength(Temp_x,Allseria.AllTrCnt);
        for i:=0 to Allseria.AllTrCnt-1 do
            Temp_x[i]:=Trajects.Traject[i].x_arr[0];
        for i:=0 to Allseria.AllTrCnt-1 do
            Trajects.Traject[i].x_arr[0]:=Temp_x[p_r[i]];
        Finalize(Temp_x);
        for i:=0 to Allseria.AllTrCnt-1 do
            Trajects.Traject[i].v_arr[0]:=v_ND[i];
        end;
        TrAction.Panels[1].Text:='Расчет слоя № 0';
        Trajects.SolveAll;
        FormPlotting.TntBitBtn2Click(Self);
        inscription:=CreateInscription;
        if Auto_Solving_Exit then
            Solving_Energy_Loss(ModelDechanneling, sublayer_on_formoptions, normaliza-
tion_on_formoptions,
            sublayer_on_formoptions_Value, Trajects.CntOfTraject, Trajects.Traject[0].Nmax,
            left_dE, right_dE, normalization_on_formoptions_Value, FileName_Solving_Exit);
        if Auto_Save then
            begin
                sdirs:=GetCurrentDir;
                SaveAllTrajectory(2, wFileName);
            end;
        for j:=1 to automatic-1 do
            begin
                if j=(automatic-1) then LastLayer:=True;
                calc:=1;
                SetLength(vla,Trajects.CntOfTraject);

```

```

SetLength(xla,Trajects.CntOfTraject);
SetLength(start_sloi,Trajects.CntOfTraject);
amnt:=Trajects.CntOfTraject;
stdep:=Trajects.Traject[0].t_arr[Trajects.Traject[0].Nmax];
s:=FloatToStr(RoundTo(stdep,-2));
LabeledEdit4.Text:=s;
dep:=dep+dopdep;
s:=FloatToStr(RoundTo(dep,-2));
LabeledEdit1.Text:=s;
for i:=0 to Trajects.CntOfTraject-1 do
    begin
        vla[i]:=Trajects.Traject[i].v_arr[Trajects.Traject[0].Nmax];
        xla[i]:=Trajects.Traject[i].x_arr[Trajects.Traject[0].Nmax];
        start_sloi[i]:=Trajects.Traject[i].sloi;
    end;
Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
TrAction.Panels[1].Text:='Расчет слоя № ' + IntToStr(j);
Trajects.SolveAll;
FormPlotting.TntBitBtn2Click(Self);
inscription:=CreateInscription;
if Auto_Solving_Exit then
    Solving_Energy_Loss(ModelDechanneling, sublayer_on_formoptions, normaliza-
tion_on_formoptions,
        sublayer_on_formoptions_Value, Trajects.CntOfTraject, Trajects.Traject[0].Nmax,
        left_dE, right_dE, normalization_on_formoptions_Value, FileName_Solving_Exit);
    if Auto_Save then SaveAllTrajectory(2, wFileName);
end;
if Auto_Save then SetCurrentDir(sdirc);
end
else if Three_point_bending then
    begin
        Curv:=StrToFloat(BendingForm.Edit2.Text)*d_small*1E+10/(Sqr(epsilon)*1E+10);
        s:=BendingForm.Edit1.Text;
        Val(s,kt,code);
        calc:=0;
        stdep:=0;
        SetLength(start_sloi,Allseria.AllTrCnt);
        for i:=0 to Allseria.AllTrCnt-1 do start_sloi[i]:=stdep;
        s:=FormOptions.LabeledEdit5.Text;
        Val(s,dopdep,code);
        step_on_tau:=dopdep;
        s:=LabeledEdit1.Text;
        Val(s,dep,code);
    end

```

```

JK:=Round(dep);
LastLayer_Value:=Round(dep/step_on_tau);
Cnt_Originally:=cnt;
cnt:=Round(cnt*dopdep/dep);
CntGlobal:=cnt;
automatic:=Round(dep/dopdep);
Number_automatic_step:=automatic;
dep:=dopdep;
s:=FloatToStr(RoundTo(dep,-2));
LabeledEdit1.Text:=s;
LabeledEdit2.Text:=IntToStr(CntGlobal);
CrystalSolving:=ComboCrystal.Text;
Z1Solving:=LEZ1.Text;
Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
if VGauss=True then
    begin
        SetLength(Temp_x,Allseria.AllTrCnt);
        for i:=0 to Allseria.AllTrCnt-1 do
            Temp_x[i]:=Trajects.Traject[i].x_arr[0];
        for i:=0 to Allseria.AllTrCnt-1 do
            Trajects.Traject[i].x_arr[0]:=Temp_x[p_r[i]];
        Finalize(Temp_x);
        for i:=0 to Allseria.AllTrCnt-1 do
            Trajects.Traject[i].v_arr[0]:=v_ND[i];
        end;
        TrAction.Panels[1].Text:='Расчет слоя № 0';
        Trajects.SolveAll;
        FormPlotting.TntBitBtn2Click(Self);
        inscription:=CreateInscription;
        if Auto_Solving_Exit then
            Solving_Energy_Loss(ModelDechanneling, sublayer_on_formoptions, normaliza-
tion_on_formoptions,
            sublayer_on_formoptions_Value, Trajects.CntOfTraject, Trajects.Traject[0].Nmax,
            left_dE, right_dE, normalization_on_formoptions_Value, FileName_Solving_Exit);
        if Auto_Save then
            begin
                sdirs:=GetCurrentDir;
                SaveAllTrajectory(2, wFileName);
            end;
        tay3x1:=StrToInt(BendingForm.Edit9.Text);
        tay3x2:=StrToInt(BendingForm.Edit10.Text);
        tay3x3:=StrToInt(BendingForm.Edit11.Text);
        tay3x4:=StrToInt(BendingForm.Edit12.Text);

```



```

for j:=1 to automatic-1 do
  begin
    if j=(automatic-1) then LastLayer:=True;
    if (j=tay3x1) then
      begin
        Curv:=StrToFloat(BendingForm.Edit4.Text)*d_small*1E+10/(Sqr(epsilon)*1E+10);
        s:=BendingForm.Edit3.Text;
        Val(s,kt,code);
      end;
    if (j=tay3x2) then
      begin
        Curv:=StrToFloat(BendingForm.Edit6.Text)*d_small*1E+10/(Sqr(epsilon)*1E+10);
        s:=BendingForm.Edit5.Text;
        Val(s,kt,code);
      end;
    if (j=tay3x3) then
      begin
        Curv:=StrToFloat(BendingForm.Edit8.Text)*d_small*1E+10/(Sqr(epsilon)*1E+10);
        s:=BendingForm.Edit7.Text;
        Val(s,kt,code);
      end;
    calc:=1;
    SetLength(vla,Trajects.CntOfTraject);
    SetLength(xla,Trajects.CntOfTraject);
    SetLength(start_sloi,Trajects.CntOfTraject);
    amnt:=Trajects.CntOfTraject;
    stdep:=Trajects.Traject[0].t_arr[Trajects.Traject[0].Nmax];
    s:=FloatToStr(RoundTo(stdep,-2));
    LabeledEdit4.Text:=s;
    dep:=dep+dopdep;
    s:=FloatToStr(RoundTo(dep,-2));
    LabeledEdit1.Text:=s;
    for i:=0 to Trajects.CntOfTraject-1 do
      begin
        vla[i]:=Trajects.Traject[i].v_arr[Trajects.Traject[0].Nmax];
        xla[i]:=Trajects.Traject[i].x_arr[Trajects.Traject[0].Nmax];
        start_sloi[i]:=Trajects.Traject[i].sloi;
      end;
    Trajects.Init(calc,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
    TrAction.Panels[1].Text:='Расчет слоя № ' + IntToStr(j);
    Trajects.SolveAll;
    FormPlotting.TntBitBtn2Click(Self);
    inscription:=CreateInscription;

```

```

        if Auto_Solving_Exit then
            Solving_Energy_Loss(ModelDechanneling, sublayer_on_formoptions, normaliza-
tion_on_formoptions,
                sublayer_on_formoptions_Value, Trajects.CntOfTraject, Trajects.Traject[0].Nmax,
                left_dE, right_dE, normalization_on_formoptions_Value, FileName_Solving_Exit);
            if Auto_Save then SaveAllTrajectory(2, wFileName);
        end;
        if Auto_Save then SetCurrentDir(sdirs);
    end;
    if Trajects.Traject[Trajects.CntOfTraject-1].Solved then
        begin
            SolveMyDataDone := True;
            TrAction.Panels[1].Text := 'Расчет завершен...';
        end;
        LEGraphT0.Text := LabeledEdit4.Text;
        LEGraphTLast.Text := LabeledEdit1.Text;
        SetCoeff;
        Form1.Caption := inscription;
        entime := Now;
        if DateToStr(sttime) <> DateToStr(entime) then
            Label9.Caption := DateTimeToStr(sttime) + ' - ' + DateTimeToStr(entime) else
            Label9.Caption := TimeToStr(sttime) + ' - ' + TimeToStr(entime);

        if (Auto_Save) and (TrDataFormat=4) then
            begin
                DeleteVremDir(PathVrem);
                DestroyOurConnection;
            end;

    end;

    procedure TForm1.AddTrBtnClick(Sender: TObject);
    var SetTrEd: TextFile;
        Text: String;
    begin
        Form2.Visible:=true;
    end;

    procedure TForm1.SaveTrDatBtnClick(Sender: TObject);
    var
        i:integer;
        s:string;
    begin

```

```

for i:=0 to (Trajects.CntOfTraject-1) do
  begin
    str(i,s);
    Trajects.Traject[i].SvTable2Dat(s+'_xv.dat');
  end;
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then
    begin
      LEMultX.Enabled:=false;
      LEMultY.Enabled:=false;
      LECenX.Enabled:=false;
      LECenY.Enabled:=false;
    end
  else
    begin
      LEMultX.Enabled:=true;
      LEMultY.Enabled:=true;
      LECenX.Enabled:=true;
      LECenY.Enabled:=true;
    end;
  end;

procedure TForm1.LEEpsilonChange(Sender: TObject);
var
  s:string;
  r:double;
  code:integer;
begin
  s:=LEDeltaTheta.Text;
  Val(s,r,code);
  if code=0 then
    begin
      DeltaTheta:=r*Pi/(epsilon*180);
      LEDeltaThetaInEps.Text:=FloatToStr(RoundTo(DeltaTheta,-4));
    end;
  end;

procedure TForm1.AboutBtnClick(Sender: TObject);
begin
  FormAbout.Visible:=true;

```

end;

procedure TForm1.FormClose(Sender: TObject; **var** Action: TCloseAction);

var SetEd: TextFile; s:string;

begin

SQLConnection1.Close;

try

SaveFileInitIni('initial' + '.ini');

except

MessageDlg('Сохранение настроек невозможно!',mtError,
[mbOk], 0);

end;

end;

procedure TForm1.UpdateInitialConditions(a: integer);

begin

if a=0 **then**

begin

FirstLoad:=True;

CChoiceCrystal(Self);

CChoiceParticles(Self);

CChoicePlane(Self);

FirstLoad:=False;

end;

end;

procedure TForm1.InitialConditions;

begin

UpdateInitialConditions(0);

LEEnergyChange(Self);

end;

procedure TForm1.LEEnergyChange(Sender: TObject);

var

s,sr:string;

r:double;

code:integer;

begin

s:=LEEnergy.Text;

Val(s,r,code);

if code=1 **then**

begin

```

    MessageDlg('E не может быть нулевой, по умолчанию принимаем 1 эВ', mtInformation,
[mbOk],0);
    r:=1;
    LEEnergy.Text:='1';
end;
E:=r*1;
s:=LEnx.Text;
Val(s,r,code);
nxRange:=Round(r);
s:=LEZ1.Text;
Val(s,r,code);
Z1:=Round(r);
if Z1<0 then
    begin
        Z1:=Abs(Z1);
        Z1_Znak:=-1;
    end
    else Z1_Znak:=1;
s:=LEAEM.Text;
Val(s,r,code);
M1:=r;
m0c2:=r*931500000;
if tipV=1 then
    begin
        s:=LEzsmall.Text;
        Val(s,r,code);
        z_small:=Round(r);
        if Z1<=z_small then
            begin
                Z1:=z_small+1;
                Str(Z1,s);
                LEZ1.Text:=s;
                ChangeFlying(Self);
            end;
        end;
    end;
InitConstants;
r:=Sqrt(Abs(Vmax)/(E+m0c2/(1+m0c2/E)));
Str(r,sr);
LEEpsilon.Text:=sr;
Val(sr,r,code);
epsilon:=r;
s:=LEDeltaTheta.Text;
Val(s,r,code);

```

```

DeltaTheta:=r*Pi/(epsilon*180);
LEDeltaThetaInEps.Text:=FloatToStr(RoundTo(DeltaTheta,-4));
InitConstants;
PhiEpsilon:=Phi_kl/epsilon;
end;

```

```

procedure TForm1.CChoicePlane;

```

```

begin

```

```

  if ComboPlosk.Text='100' then

```

```

    begin

```

```

      ax:=d_small;
      ay:=d_small;
      az:=d_small;
      x_stj[1]:=0; x_stj[2]:=0; x_stj[3]:=1/2; x_stj[4]:=1/2;
      x_stj[5]:=1/4; x_stj[6]:=1/4; x_stj[7]:=3/4; x_stj[8]:=3/4;
      y_stj[1]:=0; y_stj[2]:=1/2; y_stj[3]:=0; y_stj[4]:=1/2;
      y_stj[5]:=1/4; y_stj[6]:=3/4; y_stj[7]:=1/4; y_stj[8]:=3/4;
      z_stj[1]:=0; z_stj[2]:=1/2; z_stj[3]:=1/2; z_stj[4]:=0;
      z_stj[5]:=1/4; z_stj[6]:=3/4; z_stj[7]:=3/4; z_stj[8]:=1/4;
      tip_ploskost:='100';

```

```

    end

```

```

  else if ComboPlosk.Text='110' then

```

```

    begin

```

```

      ax:=d_small/sqrt(2);
      ay:=d_small/sqrt(2);
      az:=d_small;
      x_stj[1]:=0; x_stj[2]:=0.3536; x_stj[3]:=0.3536; x_stj[4]:=0.7071;
      x_stj[5]:=0.3536; x_stj[6]:=0.7071; x_stj[7]:=0.7071; x_stj[8]:=1.0607;
      y_stj[1]:=0; y_stj[2]:=0.3536; y_stj[3]:=-0.3536; y_stj[4]:=0;
      y_stj[5]:=0; y_stj[6]:=0.3536; y_stj[7]:=-0.3536; y_stj[8]:=0;
      z_stj[1]:=0; z_stj[2]:=1/2; z_stj[3]:=1/2; z_stj[4]:=0;
      z_stj[5]:=1/4; z_stj[6]:=3/4; z_stj[7]:=3/4; z_stj[8]:=1/4;
      tip_ploskost:='110';

```

```

    end

```

```

  else if ComboPlosk.Text='111' then

```

```

    begin

```

```

      ax:=d_small/sqrt(3);
      ay:=d_small/sqrt(2);
      az:=d_small/sqrt(1.5);
      x_stj[1]:=0; x_stj[2]:=0.5774; x_stj[3]:=0.5774; x_stj[4]:=0.5774;
      x_stj[5]:=0.433; x_stj[6]:=1.0104; x_stj[7]:=1.0104; x_stj[8]:=1.0104;
      y_stj[1]:=0; y_stj[2]:=0.3536; y_stj[3]:=-0.3536; y_stj[4]:=0;

```

```

    y_stj[5]:=0; y_stj[6]:=0.3536; y_stj[7]:=-0.3536; y_stj[8]:=0;
    z_stj[1]:=0; z_stj[2]:=0.2041; z_stj[3]:=0.2041; z_stj[4]:=-0.4082;
    z_stj[5]:=0; z_stj[6]:=0.2041; z_stj[7]:=0.2041; z_stj[8]:=-0.4082;
    tip_ploskost:='111';
end;
if FirstLoad=False then LEEnergyChange(Self);
end;

procedure TForm1.CChoiceParticles;
begin
    if ComboParticles.Text='Протон' then
        begin
            LEZ1.Text:='1';
            LEAEM.Text:='1.00728';
        end
    else if ComboParticles.Text='Pb' then
        begin
            LEZ1.Text:='82';
            LEAEM.Text:='207.2';
        end
    else if ComboParticles.Text='Дпырое' then
        begin
            LEZ1.Text:=LEZ1.Text;
            LEAEM.Text:=LEAEM.Text;
        end;
    if FirstLoad=False then LEEnergyChange(Self);
end;

procedure TForm1.CChoiceCrystal(Sender: TObject);
begin
    if ComboCrystal.Text='C' then Crystal:='C'
    else if ComboCrystal.Text='Si' then Crystal:='Si'
    else if ComboCrystal.Text='Ge' then Crystal:='Ge';
    InitCrystalConstants;
    if FirstLoad=False then LEEnergyChange(Self);
end;

procedure TForm1.ChangeFlying(Sender: TObject);
begin
    ComboParticles.Text:='Дпырое';
end;

procedure TForm1.VmolierClick(Sender: TObject);

```

```

begin
    tipV:=0;
    ChangeOptions;
end;

procedure TForm1.VmolierzClick(Sender: TObject);
begin
    tipV:=1;
    ChangeOptions;
end;

procedure TForm1.FastAddTrajectory(Sender: TObject);
begin
    Form1.AddTrBtnClick(Self);
    STEaddTrj.AllSeria.Init(10,@Form2);
    Form2.BitBtn1Click(Self);
end;

procedure TForm1.YesGameClick(Sender: TObject);
begin
    gamev := 1;
    ChangeOptions;
end;

procedure TForm1.NoGameClick(Sender: TObject);
begin
    gamev := 0;
    ChangeOptions;
end;

procedure TForm1.OptionsgameClick(Sender: TObject);
begin
    FormOptions.Visible := True;
end;

procedure TForm1.ReStartProgram(tmpLoad: boolean; FullDataPath: string; TopPosi-
tion,LeftPosition: integer);
var
    FExe : string;
procedure Exec(const FileName, Parameters, Directory: string);
var
    Operation: string;

```



```

begin
    Operation := 'open';
    ShellExecute(Windows.GetForegroundWindow, PChar(Operation), PChar(FileName),
PChar(Parameters), PChar(Directory),
    SW_SHOWNORMAL);
end;
begin
    FExe:=ParamStr(0);
    if tmpLoad then
        Exec(FExe, '/l' + ' ' + FullDataPath + ' ' + IntToStr(TopPosition) + ' ' + IntToStr(LeftPosition),
ExtractFilePath(FExe))
    else
        Exec(FExe, '/r' + ' ' + IntToStr(TopPosition) + ' ' + IntToStr(LeftPosition), ExtractFile-
Path(FExe));
        SaveFileInitIni('initial' + '.ini');
        Application.Terminate;
    end;

procedure TForm1.LoadParams;
var
    STE_Params_Load,
    STE_Params_ReStart: Boolean;
begin
    STE_Params_Load := (ParamCount > 0) and (CompareText(ParamStr(1), '/l') = 0);
    STE_Params_ReStart := (ParamCount > 0) and (CompareText(ParamStr(1), '/r') = 0);
    if STE_Params_Load then
        begin
            Form1.Top := StrToInt(ParamStr(3));
            Form1.Left := StrToInt(ParamStr(4));
            FileLoadPath := ParamStr(2);
            LoadWithParameters := True;
        end;
    if STE_Params_ReStart then
        begin
            Form1.Top := StrToInt(ParamStr(2));
            Form1.Left := StrToInt(ParamStr(3));
        end;
    end;

procedure TForm1.FormShow(Sender: TObject);
begin
    ChangeOptions;
    FormOptions.RadioButton123Click(Self);

```

```

STE_Refreshment;
LoadParametrs;
end;

procedure TForm1.STE_Refreshment;
begin
Form2.LEalpha.Text:=LEalpha;
Form2.LEbeta.Text:=LEbeta;
if Vgauss then Form2.CheckBox2.Checked:=True;
with Form2 do
  begin
    LabeledEdit1.Text:=VXTr_from_AddTrj[1];
    LabeledEdit2.Text:=VXTr_from_AddTrj[2];
    LabeledEdit3.Text:=VXTr_from_AddTrj[3];
    LabeledEdit4.Text:=VXTr_from_AddTrj[4];
    LabeledEdit5.Text:=VXTr_from_AddTrj[5];
  end;
with BendingForm do
  begin
    Edit1.Text:=Edit_from_BendingForm[1];
    Edit2.Text:=Edit_from_BendingForm[2];
    Edit3.Text:=Edit_from_BendingForm[3];
    Edit4.Text:=Edit_from_BendingForm[4];
    Edit5.Text:=Edit_from_BendingForm[5];
    Edit6.Text:=Edit_from_BendingForm[6];
    Edit7.Text:=Edit_from_BendingForm[7];
    Edit8.Text:=Edit_from_BendingForm[8];
    Edit9.Text:=Edit_from_BendingForm[9];
    Edit10.Text:=Edit_from_BendingForm[10];
    Edit11.Text:=Edit_from_BendingForm[11];
    Edit12.Text:=Edit_from_BendingForm[12];
  end;
with FormPlotting do
  begin
    LabeledEdit1.Text:=Edit_from_FormPlotting[1];
    LabeledEdit3.Text:=Edit_from_FormPlotting[2];
    LabeledEdit4.Text:=Edit_from_FormPlotting[3];
    LabeledEdit5.Text:=Edit_from_FormPlotting[4];
    LabeledEdit8.Text:=Edit_from_FormPlotting[5];
    LabeledEdit9.Text:=Edit_from_FormPlotting[6];
    LabeledEdit10.Text:=Edit_from_FormPlotting[7];
    LabeledEdit11.Text:=Edit_from_FormPlotting[8];
    Edit11.Text:=Edit_from_FormPlotting[9];

```

```

end;
with AdvOptF do
  begin
    LabeledEdit1.Text:=Edit_from_FormPlotting[1];
    LabeledEdit3.Text:=Edit_from_FormPlotting[2];
    LabeledEdit4.Text:=Edit_from_FormPlotting[3];
    LabeledEdit5.Text:=Edit_from_FormPlotting[4];
    LabeledEdit8.Text:=Edit_from_FormPlotting[5];
    LabeledEdit9.Text:=Edit_from_FormPlotting[6];
    LabeledEdit10.Text:=Edit_from_FormPlotting[7];
    LabeledEdit11.Text:=Edit_from_FormPlotting[8];
    Edit11.Text:=Edit_from_FormPlotting[9];
  end;
with FormOptions do
  begin
    Edit13.Text := Edit_from_optiongamev[1];
    Edit14.Text := Edit_from_optiongamev[2];
    Edit15.Text := Edit_from_optiongamev[3];
    LabeledEdit5.Text := Auto_Solving_ with step;
    LabeledEdit6.Text := N_Point_Save_Value_Str;
    LEDBName.Text := DataBase_Name;
    if Last_Layer_ReSave then CheckBox10.Checked := True;
    if Last_Layer_Save then CheckBox9.Checked := True;
    if N_Point_Save then CheckBox11.Checked := True;
    if Auto_Save then CheckBox7.Checked := True;
    if Auto_Solving then CheckBox6.Checked := True;
    if Auto_Solving_Exit then CheckBox12.Checked := True;
    if Three_point_bending then CheckBox8.Checked := True;
    if sublayer_on_formoptions then CheckBox13.Checked := True;
    if normalization_on_formoptions then CheckBox14.Checked := True;
    if Potential_Axil_Plane then CheckBox17.Checked := True;
  end;
with ModelDechF do
  begin
    ChDechEn.Checked := FOptions_Checked[1];
    ChDechDistance.Checked := FOptions_Checked[2];
    if AllDataAveraging then RadioButton1.Checked := True
      else RadioButton2.Checked := True;
  end;
with PlotF do
  begin
    CheckBox11.Checked := Form11_Checked[1];
    CheckBox1.Checked := Form11_Checked[2];

```

```

    CheckBox2.Checked := Form11_Checked[3];
    ChEffect.Checked := Form11_Checked[4];
    PlotF.EdEffectFrom.Text := Edit_from_PlotF[1];
    PlotF.EdEffectTo.Text := Edit_from_PlotF[2];
end;
end;

procedure TForm1.NPlottingClick(Sender: TObject);
var
    i:integer;
    Text,Sx,Sv:string;
begin
    PlotF.Show;
end;

procedure TForm1.SaveAllTrajectory(b: byte; w: string);
var
    tmpFileName, tmpDirFile: string;
    tmpd, i: integer;
    CreateVrem: boolean;
    tmpCurrentDir, PathVrem: string;
    CanSaveFile: boolean;
begin

    if (b=1) or (b=3) or (b=4) then
        begin
            tmpCurrentDir := GetCurrentDir;
            SaveDialogFile := TSaveDialog.Create(Self);
            if b=3 then SaveDialogFile.FileName := ComboCrystal.Text + tip_ploskost;
            if SaveDialogFile.Execute then
                try
                    tmpFileName := ExtractFileName(SaveDialogFile.FileName);
                    tmpDirFile := ExtractFilePath(SaveDialogFile.FileName);
                    CanSaveFile := True;
                except
                    MessageDlg('Ошибка. Возможно ошибка в имени файла.', mtInformation, [mbOk], 0);
                    Exit;
                end;
            SaveDialogFile.Free;
            if not SetCurrentDir(tmpCurrentDir) then
                begin
                    CanSaveFile := False;
                    MessageDlg('Ошибка. Возможно неверная директория.', mtInformation, [mbOk], 0);

```

```

end;
if CanSaveFile then
begin
  PathVrem := tmpDirFile + 'vrem';
  CreateVrem := CreateVremDir(PathVrem);
  if not CreateVrem then
  begin
    MessageDlg('Нет доступа для создания временной папки.', mtInformation, [mbOk], 0);
    Exit;
  end;
  tmpd:=Length(tmpDirFile);
  if tmpDirFile[tmpd] = '\' then Delete(tmpDirFile, tmpd, 1);
  tmpd:=tmpd-1;
  for i:=0 to tmpd do
    if tmpDirFile[tmpd-i]='\' then
      tmpDirFile[tmpd-i]:='/';
  try
    CreateOurConnection('vrem', tmpDirFile);
    Connection.Connected := True;
  except
    MessageDlg('Ошибка при подключении к временной папке.', mtError, [mbOk], 0);
    Exit;
  end;
  try
    CreateMSDataBase(tmpFileName);
  except
    MessageDlg('Введено ошибочное имя файла.', mtError, [mbOk], 0);
    Exit;
  end;
  SaveTrj(1, tmpFileName);
  DeleteVremDir(PathVrem);
  DestroyOurConnection;
end;
end;
if (b=2) then
  if TrDataFormat=4 then
  begin
    SaveTrj(2, w);
  end
  else SaveTrj(2,LabeledEdit4.Text+'-'+LabeledEdit1.Text+'.trj');
if (b=5) then SaveTrj(2,
  ' Si ('+tip_ploskost+') Z1='+LEZ1.Text+' Частиц '+FloatToStr(Allseria.AllTrCnt)+
  ' E='+LEEnergy.Text+

```

```

        'əB IIIar '+FloatToStr(RoundTo(Trajects.Traject[0].tau,-5))+
        ('+FloatToStr(RoundTo(Trajects.Traject[0].tau/epsilon*d_small,-2))+ ' A) '+
        LabeledEdit4.Text+'-'+LabeledEdit1.Text+' ray');
end;

```

```

procedure TForm1.SaveTrj(x:integer; w:string);
var
SaveAllTrj: TextFile;
s: string;
i, j: integer;
d: double;
begin
if (TrDataFormat=4) then
    begin
        try
            if ((x=1) or (x=2)) and (Create_DB=0) then
                begin

                    if (N_Point_Save) or (Auto_Solving=False) then
                        begin
                            s:=w + 'dl';
                            CreateMSdl(s);
                            s:=w + 'db';
                            CreateMSDB(s);
                        end;
                    if N_Point_Save then
                        begin
                            s:=w + '_particles_dl';
                            CreateMSdl(s);
                            for i:=0 to Trajects.CntOfTraject-1 do
                                begin
                                    s:=w + '_particles' + IntToStr(i) + '_db';
                                    CreateMSDB(s);
                                end;
                            end;
                        end;
                    if (N_Point_Save=False) and (Auto_Solving) then
                        begin
                            for i:=0 to Number_automatic_step-1 do
                                begin
                                    s:=w + '_Layer' + IntToStr(i) + '_dl';
                                    CreateMSdl(s);
                                    s:=w + '_Layer' + IntToStr(i) + '_db';
                                    CreateMSDB(s);
                                end;
                            end;
                        end;
                end;
            except
                end;
        end;
    end;

```

```

    end;
end;
if Last_Layer_Save then
begin
    s:=w + '_LastLayer_dl';
    CreateMSdl(s);
    s:=w + '_LastLayer_db';
    CreateMSDB(s);
    end;
inc(Create_DB);
end;
except
    MessageDlg('Ошибка создания БД или пустых таблиц',mtError,[mbOk], 0);
    exit;
end;
try
    if (x=1) or ((x=2) and (Save_DBdl=0)) then
    begin
        if N_Point_Save then
        begin
            N_Point_Save_Value_Int:=StrToInt(N_Point_Save_Value_Str);
            if (Auto_Solving=False) and (N_Point_Save_Value_Int=1) then
                d_int:=Trajects.Traject[0].Nmax
            else
                begin
                    d:=(1+Trajects.Traject[0].Nmax/N_Point_Save_Value_Int);
                    if d=Int(d) then d_int:=Trunc(d)-2 else d_int:=Trunc(d)-1;
                end;
            tau_As_Step_Saving:=Trajects.Traject[0].tau*N_Point_Save_Value_Int;

            if Auto_Solving then
                cnt_for_Saving_if_N_Point_True:=Round((d_int+1)*LastLayer_Value)
            else
                cnt_for_Saving_if_N_Point_True:=Round((d_int+1));
            cnt_for_Saving_if_N_Point_True_on_one_step_tau:=(d_int+1);
            SaveAllTrajectoryMSToDBdl(w+'dl', False, 'for_db');
            s:=w + '_particles_dl';
            SaveAllTrajectoryMSToDBdl(s, False, 'for_particles');
        end
        else if (N_Point_Save=False) and (Auto_Solving=False) then
        begin
            SaveAllTrajectoryMSToDBdl(w+'dl', False, 'ordinary_state');
        end;
    end;
end;

```

```

    end;
inc(Save_DBdl);
if ((x=2) and (Last_Layer_Save) and (LastLayer)) then
    begin
        cnt_for_Saving_if_N_Point_True:=CntGlobal;
        s:=w + '_LastLayer_dl';
        SaveAllTrajectoryMSToDBdl(s, False, 'for_LastLayer');
    end;
if (x=2) and (Last_Layer_Save) and (Last_Layer_ReSave) then
    begin
        cnt_for_Saving_if_N_Point_True:=CntGlobal;
        s:=w + '_LastLayer_dl';
        if (Save_DBdl>1) then
            begin
                ZQueryMSothers.SQL.Clear;
                ZQueryMSothers.SQL.Text:='delete from ' + s;
                ZQueryMSothers.ExecSQL;
            end;
        SaveAllTrajectoryMSToDBdl(s, False, 'for_LastLayer');
    end;
if ((x=2) and (N_Point_Save=False) and (Auto_Solving)) then
    begin
        i:=Inversely_Number_automatic_step;
        SaveAllTrajectoryMSToDBdl(w+'_Layer' + IntToStr(i) + '_dl', False, 'ordinary_state');
    end;
except
    MessageDlg('При сохранении дополнительных данных возникла ошибка',mtError,[mbOk],
0);
end;
try
    if (x=1) or (x=2) then
        begin
            if N_Point_Save then
                begin
                    SaveAllTrajectoryMSToDB(w, False, N_Point_Save, N_Point_Save_Value_Int, d_int,
Last_Layer_Save, Last_Layer_ReSave);
                end
            else if (N_Point_Save=False) and (Auto_Solving) then
                begin
                    i:=Inversely_Number_automatic_step;
                    SaveAllTrajectoryMSToDB(w + '_Layer' + IntToStr(i) + '_db', False, False, 1, 1, False,
False);
                    Inversely_Number_automatic_step:=Inversely_Number_automatic_step+1;

```



```

        end
        else SaveAllTrajectoryMSToDB(w+'db', False, False, 1, 1, False, False)
    end
    else SaveAllTrajectoryMSToDB(w+'db', False, False, 1, 1, False, False);
except
    MessageDlg('При сохранении основных данных возникла ошибка',mtError,[mbOk], 0);
end;
end;
end;

procedure
TForm1.SaveAllTrajectoryMSToDB(NameDBtoinsert:string;SaveDialogActive:boolean;
N_Save:boolean;N_point:integer;N_point_Amount:integer;Layer_Save:boolean;Layer_ReSave:boo
lean);
var
i,j,n,p: integer;
s: string;
begin
    DecimalSeparator:='.';
    TrAction.Panels[1].Text:='Сохранение данных...';
    Gauge1.Progress:=0;
    if gamev=0 then Gauge1.MaxValue:=Trajects.CntOfTraject*Trajects.Traject[0].NodeCount
    else if gamev=1 then
        Gauge1.MaxValue:=2*Trajects.CntOfTraject*Trajects.Traject[0].NodeCount;
    if gamev=0 then
        begin
            if N_Save then
                begin
                    for i:=0 to Trajects.CntOfTraject-1 do
                        begin
                            s:=NameDBtoInsert + '_particles' + IntToStr(i) + '_db';
                            if Insert_Data_MS_sloi=False then
                                begin
                                    InsertDataMS1vxe(s, i , -1, Trajects.Traject[i].sloi, 0, 0);
                                end;
                                    InsertDataMS1vxe(s, i, Big_J, Trajects.Traject[i].v_arr[0], Trajects.Traject[i].x_arr[0], del-
taE[i,0]);
                                    n:=0;
                                    j:=0;
                                    Repeat
                                        n:=n+N_point;
                                        InsertDataMS1vxe(s, i, j + 1 + Big_J, Trajects.Traject[i].v_arr[n], Tra-
jects.Traject[i].x_arr[n], deltaE[i, n]);

```

```

        j:=j+1;
    Until j=N_point_Amount;
end;
Insert_Data_MS_sloi:=True;
if Layer_Save then
begin
    s:=NameDBtoInsert + '_LastLayer_db';
    if Layer_ReSave then
    begin
        ZQueryMSothers.SQL.Clear;
        ZQueryMSothers.SQL.Text:='delete from ' + s;
        ZQueryMSothers.ExecSQL;
    end;
    if (Layer_ReSave) or (Layer_Save and LastLayer) then
    begin
        for i:=0 to Trajects.CntOfTraject-1 do
        begin
            InsertDataMS1(s, i, -1, Trajects.Traject[i].sloi, 0, 0);
            for j:=0 to Trajects.Traject[0].Nmax do
                InsertDataMS1(s, i, j, Trajects.Traject[i].v_arr[j], Trajects.Traject[i].x_arr[j], del-
taE[i,j]);
            end;
        end
    end;
end;
Big_J:=Big_J + N_point_Amount + 1;
if LastLayer then
begin
    for i:=0 to Trajects.CntOfTraject-1 do
    begin
        s:=NameDBtoInsert + '_particles' + IntToStr(i) + '_db';
        ZQueryMSothers.SQL.Clear;
        ZQueryMSothers.SQL.Text:='insert into ' + NameDBtoInsert + 'db (dtrajecto-
ry,duzel,dv,dx,de) select dtrajectory,duzel,dv,dx,de from ' + s;
        ZQueryMSothers.ExecSQL;
    end;
end;
end
else
begin
    for i:=0 to Trajects.CntOfTraject-1 do
    begin
        InsertDataMS1(NameDBtoInsert, i, -1, Trajects.Traject[i].sloi, 0, 0);
        for j:=0 to Trajects.Traject[0].Nmax do

```

```

        InsertDataMS1(NameDBtoInsert, i, j, Trajects.Traject[i].v_arr[j], Tra-
jects.Traject[i].x_arr[j], deltaE[i,j]);
    end;
end;
end
else
    if gamev=1 then
        begin
            if N_Save then
                begin
                    for i:=0 to Trajects.CntOfTraject-1 do
                        begin
                            s:=NameDBtoInsert + '_particles' + IntToStr(i) + '_db';
                            if Insert_Data_MS_sloi=False then
                                begin
                                    InsertDataMS2vxe(s, i, -1, Trajects.Traject[i].sloi, 0, 0, 0);
                                end;
                                InsertDataMS2vxe(s, i, Big_J, Trajects.Traject[i].v_arr[0], Trajects.Traject[i].x_arr[0],
                                deltaE[i,0], Trajects.Traject[i].f_arr[0]);
                                n:=0;
                                j:=0;
                                Repeat
                                    n:=n+N_point;
                                    InsertDataMS2vxe(s, i, j + 1 + Big_J, Trajects.Traject[i].v_arr[n], Tra-
jects.Traject[i].x_arr[n], deltaE[i, n], Trajects.Traject[i].f_arr[n]);
                                    j:=j+1;
                                Until j=N_point_Amount;
                                end;
                                Insert_Data_MS_sloi:=True;
                                if Layer_Save then
                                    begin
                                        s:=NameDBtoInsert + '_LastLayer_db';
                                        if Layer_ReSave then
                                            begin
                                                ZQueryMSothers.SQL.Clear;
                                                ZQueryMSothers.SQL.Text:='delete from ' + s;
                                                ZQueryMSothers.ExecSQL;
                                            end;
                                        if (Layer_ReSave) or (Layer_Save and LastLayer) then
                                            begin
                                                for i:=0 to Trajects.CntOfTraject-1 do
                                                    begin
                                                        InsertDataMS2(s, i, -1, Trajects.Traject[i].sloi, 0, 0, 0);

```

```

        for j:=0 to Trajects.Traject[0].Nmax do
            InsertDataMS2(s, i, j, Trajects.Traject[i].v_arr[j], Trajects.Traject[i].x_arr[j], deltaE[i,j], Trajects.Traject[i].f_arr[j]);
        end;
    end
end;
Big_J:=Big_J + N_point_Amount + 1;
if LastLayer then
begin
    for i:=0 to Trajects.CntOfTraject-1 do
        begin
            s:=NameDBtoInsert + '_particles' + IntToStr(i) + '_db';
            ZQueryMSothers.SQL.Clear;
            ZQueryMSothers.SQL.Text:='insert into ' + NameDBtoInsert + 'db (dtrajectory,duzel,dv,dx,de) select dtrajectory,duzel,dv,dx,de from ' + s;
            ZQueryMSothers.ExecSQL;
        end;
    end;
else
begin
    for i:=0 to Trajects.CntOfTraject-1 do
        begin
            InsertDataMS2(NameDBtoInsert, i, -1, Trajects.Traject[i].sloi, 0, 0, 0);
            for j:=0 to Trajects.Traject[0].Nmax do
                InsertDataMS2(NameDBtoInsert, i, j, Trajects.Traject[i].v_arr[j], Trajects.Traject[i].x_arr[j], deltaE[i,j], Trajects.Traject[i].f_arr[j]);
            end;
        end;
    end;
end;
Gauge1.Progress:=0;
TrAction.Panels[1].Text:="";
end;

```

procedure

```

TForm1.SaveAllTrajectoryMSToDBdl(NamedltoInsert:string;SaveDialogActive:boolean;table_type:string);

```

begin

```

TrAction.Panels[1].Text:='Сохранение настроек...';
Gauge1.Progress:=0;
Gauge1.MaxValue:=46;
InsertDataMS0dl(NamedltoInsert,Form2.LabeledEdit1.Text);
InsertDataMS0dl(NamedltoInsert,Form2.LabeledEdit2.Text);

```

```

InsertDataMS0dl(NamedltoInsert,Form2.LabeledEdit3.Text);
InsertDataMS0dl(NamedltoInsert,Form2.LabeledEdit4.Text);
if table_type='for_particles' then InsertDataMS0dl(NamedltoInsert,'1')
    else InsertDataMS0dl(NamedltoInsert,Form2.LabeledEdit5.Text);
InsertDataMS0dl(NamedltoInsert,ComboCrystal.Text);
InsertDataMS0dl(NamedltoInsert,ComboPlosk.Text);
InsertDataMS0dl(NamedltoInsert,LEZ1.Text);
InsertDataMS0dl(NamedltoInsert,ComboParticles.Text);
InsertDataMS0dl(NamedltoInsert,LEAEM.Text);
InsertDataMS0dl(NamedltoInsert,LEnx.Text);
InsertDataMS0dl(NamedltoInsert,LEzsmall.Text);
if (table_type='for_db') or (table_type='for_particles') then InsertDataMS0dl(NamedltoInsert,FloatToStr(LastLayer_Value*step_on_tau))
    else InsertDataMS0dl(NamedltoInsert,LabeledEdit1.Text);
if (table_type='for_db') or (table_type='for_particles') then InsertDataMS0dl(NamedltoInsert,IntToStr(Cnt_Originally))
    else InsertDataMS0dl(NamedltoInsert,LabeledEdit2.Text);
InsertDataMS0dl(NamedltoInsert,LabeledEdit3.Text);
InsertDataMS0dl(NamedltoInsert,LEEnergy.Text);
InsertDataMS0dl(NamedltoInsert,LEEpsilon.Text);
InsertDataMS0dl(NamedltoInsert,LEDeltaTheta.Text);
InsertDataMS0dl(NamedltoInsert,LECalcPeriodFrom.Text);
InsertDataMS0dl(NamedltoInsert,LECalcPeriodTo.Text);
InsertDataMS0dl(NamedltoInsert,LEortk.Text);
InsertDataMS0dl(NamedltoInsert,LEortl.Text);
InsertDataMS0dl(NamedltoInsert>Edit1.Text);
InsertDataMS0dl(NamedltoInsert>Edit2.Text);
InsertDataMS0dl(NamedltoInsert,LEMultX.Text);
InsertDataMS0dl(NamedltoInsert,LEMultY.Text);
InsertDataMS0dl(NamedltoInsert,LECenX.Text);
InsertDataMS0dl(NamedltoInsert,LECenY.Text);
InsertDataMS0dl(NamedltoInsert,inscription);
InsertDataMS0dl(NamedltoInsert,FormOptions.LabeledEdit1.Text);
InsertDataMS0dl(NamedltoInsert,FormOptions.LabeledEdit2.Text);
InsertDataMS0dl(NamedltoInsert,FormOptions.LabeledEdit3.Text);
InsertDataMS0dl(NamedltoInsert,FormOptions.LabeledEdit4.Text);
InsertDataMS0dl(NamedltoInsert,Form2.LEalpha.Text);
InsertDataMS0dl(NamedltoInsert,Form2.LEbeta.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm>Edit1.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm>Edit2.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm>Edit3.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm>Edit4.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm>Edit5.Text);

```

```

InsertDataMS0dl(NamedltoInsert,BendingForm.Edit6.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit7.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit8.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit9.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit10.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit11.Text);
InsertDataMS0dl(NamedltoInsert,BendingForm.Edit12.Text);
InsertDataMS0dl(NamedltoInsert,FormOptions.Edit13.Text);
InsertDataMS1dl(NamedltoInsert,CntGlobal);
InsertDataMS1dl(NamedltoInsert,tipV);
InsertDataMS1dl(NamedltoInsert,gamev);
InsertDataMS1dl(NamedltoInsert,OverBarrierOff);
InsertDataMS1dl(NamedltoInsert,BorOff);
InsertDataMS1dl(NamedltoInsert,SrKvFlEp);
InsertDataMS1dl(NamedltoInsert,VMoilerOff);
if (table_type='ordinary_state') then InsertDataMS1dl(NamedltoInsert,CntGlobal)
    else InsertDataMS1dl(NamedltoInsert,cnt_for_Saving_if_N_Point_True);
if (table_type='for_LastLayer') then InsertDa-
taMS1dl(NamedltoInsert,cnt_for_Saving_if_N_Point_True)
    else if (table_type='ordinary_state') then InsertDataMS1dl(NamedltoInsert,CntGlobal)
        else InsertDataMS1dl(NamedltoInsert,cnt_for_Saving_if_N_Point_True_on_one_step_tau);
InsertDataMS2dl(NamedltoInsert,ord(EnergyLossVariant4Off));
InsertDataMS2dl(NamedltoInsert,ord(VGauss));
InsertDataMS3dl(NamedltoInsert,Trajects.Traject[0].t_arr[0]);
InsertDataMS3dl(NamedltoInsert,Trajects.Traject[0].tau);
if (((table_type='for_LastLayer') or (table_type='ordinary_state')) then InsertDa-
taMS3dl(NamedltoInsert,Trajects.Traject[0].tau)
    else InsertDataMS3dl(NamedltoInsert,tau_As_Step_Saving);
InsertDataMS3dl(NamedltoInsert,step_on_tau);
InsertDataMS0dl(NamedltoInsert, Auto_Solving_with_Step);
InsertDataMS0dl(NamedltoInsert, N_Point_Save_Value_Str);
InsertDataMS0dl(NamedltoInsert, FormOptions.LEDBName.Text);
InsertDataMS2dl(NamedltoInsert, ord(Auto_Solving));
InsertDataMS2dl(NamedltoInsert, ord(Three_Point_Bending));
InsertDataMS2dl(NamedltoInsert, ord>Last_Layer_ReSave));
InsertDataMS2dl(NamedltoInsert, ord>Last_Layer_Save));
InsertDataMS2dl(NamedltoInsert, ord(N_Point_Save));
InsertDataMS2dl(NamedltoInsert, ord(Auto_Save));
InsertDataMS2dl(NamedltoInsert, ord(Auto_Solving_Exit));
InsertDataMS2dl(NamedltoInsert, ord(sublayer_on_formoptions));
InsertDataMS2dl(NamedltoInsert, ord(normalization_on_formoptions));
InsertDataMS1dl(NamedltoInsert, ModelDechanneling);
InsertDataMS2dl(NamedltoInsert, ord(Potential_Axil_Plane));

```

```

InsertDataMS2dl(NamedltoInsert, ord(AllDataAveraging));
Gauge1.Progress:=0;
TrAction.Panels[1].Text:=";
end;

```

procedure

```

TForm1.InsertDataMS1(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer
;
DataToFieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double);
begin
  try
    ZQueryMSDB1.SQL.Clear;
    ZQueryMSDB1.SQL.Text:='insert into ' +NameDBtoInsert +
      '(dtrajectory,duzel,dv,dx,de) values ('+VarToSQL(DataToFieldMS0)+
      ','+VarToSQL(DataToFieldMS1)+','+FloatForAccess(DataToFieldMS2)+
      ','+FloatForAccess(DataToFieldMS3)+','+FloatForAccess(DataToFieldMS4)+')';
    ZQueryMSDB1.ExecSQL;
  except
  end;
  Gauge1.Progress:=Gauge1.Progress+1;
end;

```

procedure

```

TForm1.InsertDataMS1vxe(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer;
DataToFieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double);
begin
  try
    ZQueryMSDB1.SQL.Clear;
    ZQueryMSDB1.SQL.Text:='insert into ' +NameDBtoInsert +
      '(dtrajectory,duzel,dv,dx,de) values ('+VarToSQL(DataToFieldMS0)+
      ','+VarToSQL(DataToFieldMS1)+','+FloatForAccess(DataToFieldMS2)+
      ','+FloatForAccess(DataToFieldMS3)+','+FloatForAccess(DataToFieldMS4)+')';
    ZQueryMSDB1.ExecSQL;
  except
  end;
  Gauge1.Progress:=Gauge1.Progress+1;
end;

```

procedure

```

TForm1.InsertDataMS2(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:integer
;

```

```

DataTo-
FieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double;DataToFieldMS5:double);
begin
  try
    ZQueryMSDB2.SQL.Clear;
    ZQueryMSDB2.SQL.Text:='insert into ' +NameDBtoInsert +
      ' VALUES (NULL,'+VarToSQL(DataToFieldMS0)+
      ','+VarToSQL(DataToFieldMS1)+','+FloatForAccess(DataToFieldMS2)+
      ','+FloatForAccess(DataToFieldMS3)+','+FloatForAccess(DataToFieldMS4)+','+FloatForAccess(D
ataToFieldMS5)+)';
    ZQueryMSDB2.ExecSQL;
  except
  end;
  Gauge1.Progress:=Gauge1.Progress+1;
end;

```

procedure

```

TForm1.InsertDataMS2vxe(NameDBtoInsert:string;DataToFieldMS0:integer;DataToFieldMS1:int
eger;

```

```

DataTo-

```

```

FieldMS2:double;DataToFieldMS3:double;DataToFieldMS4:double;DataToFieldMS5:double);

```

```

begin
  try
    ZQueryMSDB2.SQL.Clear;
    ZQueryMSDB2.SQL.Text:='insert into ' +NameDBtoInsert +
      ' (dtrajectory,duzel,dv,dx,de) values ('+VarToSQL(DataToFieldMS0)+
      ','+VarToSQL(DataToFieldMS1)+','+FloatForAccess(DataToFieldMS2)+
      ','+FloatForAccess(DataToFieldMS3)+','+FloatForAccess(DataToFieldMS4)+)';
    ZQueryMSDB2.ExecSQL;
  except
  end;
  Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

procedure TForm1.InsertDataMS0dl(NamedltoInsert:string;DataToFieldMS0:string);

```

```

begin
  ZQueryMSdl0.SQL.Clear;
  ZQueryMSdl0.SQL.Text:='insert into ' +NamedltoInsert + ' (id,dstr) VALUES (NULL,' + Var-
ToSQL(DataToFieldMS0) + ')';
  ZQueryMSdl0.ExecSQL;
  Gauge1.Progress:=Gauge1.Progress+1;
end;

```



```

procedure TForm1.InsertDataMS1dl(NamedltoInsert:string;DataToFieldMS1:integer);
begin
    ZQueryMSdl1.SQL.Clear;
    ZQueryMSdl1.SQL.Text:='insert into ' +NamedltoInsert +' (id,dint) VALUES (NULL,' + Var-
ToSQL(DataToFieldMS1) + ')';
    ZQueryMSdl1.ExecSQL;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

procedure TForm1.InsertDataMS2dl(NamedltoInsert:string;DataToFieldMS2:integer);
begin
    ZQueryMSdl2.SQL.Clear;
    ZQueryMSdl2.SQL.Text:='insert into ' +NamedltoInsert +' (id,dbool) VALUES (NULL,' + Var-
ToSQL(DataToFieldMS2) + ')';
    ZQueryMSdl2.ExecSQL;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

procedure TForm1.InsertDataMS3dl(NamedltoInsert:string;DataToFieldMS3:double);
begin
    ZQueryMSdl3.SQL.Clear;
    ZQueryMSdl3.SQL.Text:='insert into ' +NamedltoInsert +' (id,dfl) VALUES (NULL,'
+FloatForAccess(DataToFieldMS3)+ ')';
    ZQueryMSdl3.ExecSQL;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

procedure TForm1.CreateMSDataBase(NameDB:string);
begin
    Connection.Connected:=False;
    Connection.Connected:=True;
    ZQueryMS.SQL.Text:='CREATE DATABASE '+NameDB;
    ZQueryMS.ExecSQL;
    ZQueryMS.SQL.Clear;
    Connection.Connected:=False;
    Connection.Database:=NameDB;
    Connection.Connected:=True;
end;

```

```

procedure TForm1.CreateMSdl(NameDB:string);
begin

```

```

    ZQueryMS.SQL.Text:='CREATE TABLE ' +NameDB +' (id INT UNSIGNED NOT NULL
    AUTO_INCREMENT, dstr VARCHAR(140), dint INT, dbool BOOL, dfl DOUBLE, PRIMARY
    KEY (id));
    ZQueryMS.ExecSQL;
    ZQueryMS.SQL.Clear;
end;

```

```

procedure TForm1.CreateMSdb(NameDB:string);
begin
    ZQueryMS.SQL.Text:='CREATE TABLE ' +NameDB +' (id INT UNSIGNED NOT NULL
    AUTO_INCREMENT, dtrajectory INT NOT NULL, duzel INT NOT NULL, dv DOUBLE NOT
    NULL, dx DOUBLE NOT NULL, de DOUBLE NOT NULL, df DOUBLE, PRIMARY KEY (id));
    ZQueryMS.ExecSQL;
    ZQueryMS.SQL.Clear;
end;

```

```

function TForm1.LoadTrajectoryMSdl0:string;
begin
    LoadTrajectoryMSdl0:=ZTableMSdl.Fields[1].AsString;
    ZTableMSdl.Next;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

function TForm1.LoadTrajectoryMSdl1:integer;
begin
    LoadTrajectoryMSdl1:=ZTableMSdl.Fields[2].AsInteger;
    ZTableMSdl.Next;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

function TForm1.LoadTrajectoryMSdl2:boolean;
var
    i:integer;
begin
    i:=ZTableMSdl.Fields[3].AsInteger;
    if i=0 then LoadTrajectoryMSdl2:=False
    else if i=1 then LoadTrajectoryMSdl2:=True;
    ZTableMSdl.Next;
    Gauge1.Progress:=Gauge1.Progress+1;
end;

```

```

function TForm1.LoadTrajectoryMSdl3:double;
begin

```

```

LoadTrajectoryMSdl3:=ZTableMSdl.Fields[4].AsFloat;
ZTableMSdl.Next;
end;

procedure TForm1.DatasetNewFields;
var
I:integer;
FieldDefs: TFieldDefs;
CalcField: TStringField;
begin
    Dataset.Fields.Clear;
    Dataset.Close;
    FieldDefs := Dataset.FieldDefs;
    FieldDefs.Update;
    if Dataset.FindField('Calculated') = nil then
        begin
            for I := 0 to FieldDefs.Count - 1 do
                FieldDefs[I].CreateField(Dataset).DataSet := Dataset;
                CalcField := TStringField.Create(nil);
                CalcField.Size := 10;
                CalcField.FieldName := 'Calculated';
                CalcField.FieldKind := fkCalculated;
                CalcField.Visible := True;
                CalcField.DataSet := Dataset;
            end;
            Dataset.Open;
        end;

procedure TForm1.LoadAllTrajectory(Sender: TObject);
begin
    if LoadWithParameters then Load_STE_File(False, FileLoadPath)
    else
        begin
            OpenFileDialog1.Filter := 'Files trajectories (*db.MYD)|*db.MYD';
            OpenFileDialog1.InitialDir:=TDataDirMySql;
            if OpenFileDialog1.Execute then ReStartProgram(True, OpenFileDialog1.FileName, Form1.Top,
Form1.Left);
        end;
    end;

procedure TForm1.Load_STE_File(SpLoadTr: boolean; tmpOpenFileName: string);
var

```

```

s, tmpDirServer, tmpTableName, tmpTableNamedb, tmpTableNamedl: string;
step_solving_on_tau: double;
VGauss_Tmp: boolean;
Min, Max, i, j, code, Nm, cnt, amnt, r, b, w, j_first_in_tau, m, m_first_in_tau,
QueryCount_Left, QueryCount_Right, Tr_i, Tr_b, Count_of_Load: integer;
dep, stdep, left, right, tau: double;
xla, vla, start_sloi: array of double;
Load_without_QueryCount: boolean;
FieldDefs: TFieldDefs;
CalcField: TStringField;
tmp_Array_0: array [1..3] of string;
tmp_Array_1: integer;
tmp_Array_2: array [1..11] of boolean;

```

begin

if SpLoadTr = False **then**

```

    DirAndNameMyDB(tmpOpenFileName);
    tmpDirServer := DirMyTableName;
    tmpTableName := MyTableName;
    tmpTableNamedb := MyTableNamedb;
    tmpTableNamedl := MyTableNamedl;
    CreateOurConnection(tmpTableName, tmpDirServer);
    Connection.Connected := True;
    ZTableMSdl.Close;
    ZTableMSdl.Connection := Connection;
    ZTableMSdl.TableName := tmpTableNamedl;
    ZTableMSdl.Open;
    TrAction.Panels[1].Text := 'Загрузка настроек...';
    Min := 0;
    Max := ZTableMSdl.RecordCount-2;
    Gauge1.MinValue := Min;
    Gauge1.MaxValue := Max;
    Gauge1.Progress := 0;
    l:=1; k:=1; tipV:=1;
    ZTableMSdl.First;
    Form2.LabeledEdit1.Text:=LoadTrajectoryMSdl0;
    Form2.LabeledEdit2.Text:=LoadTrajectoryMSdl0;
    Form2.LabeledEdit3.Text:=LoadTrajectoryMSdl0;
    Form2.LabeledEdit4.Text:=LoadTrajectoryMSdl0;
    Form2.LabeledEdit5.Text:=LoadTrajectoryMSdl0;
    ComboCrystal.Text:=LoadTrajectoryMSdl0;
    ComboPlosk.Text:=LoadTrajectoryMSdl0;
    LEZ1.Text:=LoadTrajectoryMSdl0;
    ComboParticles.Text:=LoadTrajectoryMSdl0;

```

LEAEM.Text:=LoadTrajectoryMSdl0;
 LEnx.Text:=LoadTrajectoryMSdl0;
 LEzsmall.Text:=LoadTrajectoryMSdl0;
 LabeledEdit1.Text:=LoadTrajectoryMSdl0;
 LabeledEdit2.Text:=LoadTrajectoryMSdl0;
 LabeledEdit3.Text:=LoadTrajectoryMSdl0;
 LEEnergy.Text:=LoadTrajectoryMSdl0;
 LEEpsilon.Text:=LoadTrajectoryMSdl0;
 LEDeltaTheta.Text:=LoadTrajectoryMSdl0;
 LECalcPeriodFrom.Text:=LoadTrajectoryMSdl0;
 LECalcPeriodTo.Text:=LoadTrajectoryMSdl0;
 LEortk.Text:=LoadTrajectoryMSdl0;
 LEortl.Text:=LoadTrajectoryMSdl0;
 Edit1.Text:=LoadTrajectoryMSdl0;
 Edit2.Text:=LoadTrajectoryMSdl0;
 LEMultX.Text:=LoadTrajectoryMSdl0;
 LEMultY.Text:=LoadTrajectoryMSdl0;
 LECenX.Text:=LoadTrajectoryMSdl0;
 LECenY.Text:=LoadTrajectoryMSdl0;
 inscription:=LoadTrajectoryMSdl0;
 Val(LoadTrajectoryMSdl0,porog_max_eV2,code);
 Val(LoadTrajectoryMSdl0,porog_min_eV2,code);
 Val(LoadTrajectoryMSdl0,shag_A,code);
 Val(LoadTrajectoryMSdl0,min_shag_A,code);
 Form2.LEalpha.Text:=LoadTrajectoryMSdl0;
 Form2.LEbeta.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit1.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit2.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit3.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit4.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit5.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit6.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit7.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit8.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit9.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit10.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit11.Text:=LoadTrajectoryMSdl0;
 BendingForm.Edit12.Text:=LoadTrajectoryMSdl0;
 FormOptions.Edit13.Text:=LoadTrajectoryMSdl0;
 CntGlobal:=LoadTrajectoryMSdl1;
 tipV:=LoadTrajectoryMSdl1;
 gamev:=LoadTrajectoryMSdl1;
 OverBarrierOff:=LoadTrajectoryMSdl1;

```

BorOff:=LoadTrajectoryMSdl1;
SrKvFlEp:=LoadTrajectoryMSdl1;
VMoilerOff:=LoadTrajectoryMSdl1;
cnt_for_Saving_if_N_Point_True:=LoadTrajectoryMSdl1;
cnt_for_Saving_if_N_Point_True_on_one_step_tau:=LoadTrajectoryMSdl1;
EnergyLossVariant4Off:=LoadTrajectoryMSdl2;
VGauss:=LoadTrajectoryMSdl2;
VGauss_Tmp:=VGauss;
VGauss:=False;
Val(Form2.LabeledEdit5.Text,Nm,code);
Val(Form1.LabeledEdit1.Text,dep,code);
cnt:=cnt_for_Saving_if_N_Point_True;
Val(Form1.LabeledEdit3.Text,stdep,code);
Val(Form1.LECalcPeriodFrom.Text,left,code);
Val(Form1.LECalcPeriodTo.Text,right,code);
SetLength(Emax,Nm);
if SpLoadTr then
    begin
        SetLength(deltaE,Nm,cnt_SpLoad);
    end
    else SetLength(deltaE,Nm,cnt);
SetLength(start_sloi,Nm);
for i:=0 to Nm-1 do start_sloi[i]:=0;
STEaddTrj.AllSeria.Init(10,@Form2);
Form2.BitBtn1Click(Self);
VGauss:=VGauss_Tmp;
if VGauss then Form2.CheckBox2.Checked:=True
    else Form2.CheckBox2.Checked:=False;
if SpLoadTr then
    Trajects.Init(0,xla,vla,start_sloi,amnt,@AllSeria,stdep,b_tau,left,right,cnt_SpLoad)
    else
        Trajects.Init(0,xla,vla,start_sloi,amnt,@AllSeria,stdep,dep,left,right,cnt);
Trajects.Traject[0].t_arr[0]:=LoadTrajectoryMSdl3;
Trajects.Traject[0].tau:=LoadTrajectoryMSdl3;
tau_As_Step_Saving:=LoadTrajectoryMSdl3;
step_on_tau:=LoadTrajectoryMSdl3;
try
    tmp_Array_0[1] := LoadTrajectoryMSdl0;
    tmp_Array_0[2] := LoadTrajectoryMSdl0;
    tmp_Array_0[3] := LoadTrajectoryMSdl0;
    tmp_Array_2[1] := LoadTrajectoryMSdl2;
    tmp_Array_2[2] := LoadTrajectoryMSdl2;
    tmp_Array_2[3] := LoadTrajectoryMSdl2;

```

```

tmp_Array_2[4] := LoadTrajectoryMSdl2;
tmp_Array_2[5] := LoadTrajectoryMSdl2;
tmp_Array_2[6] := LoadTrajectoryMSdl2;
tmp_Array_2[7] := LoadTrajectoryMSdl2;
tmp_Array_2[8] := LoadTrajectoryMSdl2;
tmp_Array_2[9] := LoadTrajectoryMSdl2;
tmp_Array_1 := LoadTrajectoryMSdl1;
tmp_Array_2[10] := LoadTrajectoryMSdl2;
tmp_Array_2[11] := LoadTrajectoryMSdl2;
if (tmp_Array_0[1]<>"") and (tmp_Array_0[2]<>"") then
  begin
    Auto_Solving_With_Step := tmp_Array_0[1];
    N_Point_Save_Value_Str := tmp_Array_0[2];
    DataBase_Name := tmp_Array_0[3];
    Auto_Solving := tmp_Array_2[1];
    Three_Point_Bending := tmp_Array_2[2];
    Last_Layer_ReSave := tmp_Array_2[3];
    Last_Layer_Save := tmp_Array_2[4];
    N_Point_Save := tmp_Array_2[5];
    Auto_Save := tmp_Array_2[6];
    Auto_Solving_Exit := tmp_Array_2[7];
    sublayer_on_formoptions := tmp_Array_2[8];
    normalization_on_formoptions := tmp_Array_2[9];
    ModelDechanneling := tmp_Array_1;
    Potential_Axil_Plane := tmp_Array_2[10];
    AllDataAveraging := tmp_Array_2[11];
    with FormOptions do
      begin
        LabeledEdit5.Text := Auto_Solving_With_step;
        LabeledEdit6.Text := N_Point_Save_Value_Str;
        LEDBName.Text := DataBase_Name;
        if Last_Layer_ReSave then CheckBox10.Checked := True
          else CheckBox10.Checked := False;
        if Last_Layer_Save then CheckBox9.Checked := True
          else CheckBox9.Checked := False;
        if N_Point_Save then CheckBox11.Checked := True
          else CheckBox11.Checked := False;
        if Auto_Save then CheckBox7.Checked := True
          else CheckBox7.Checked := False;
        if Auto_Solving then CheckBox6.Checked := True
          else CheckBox6.Checked := False;
        if Auto_Solving_Exit then CheckBox12.Checked := True
          else CheckBox12.Checked := False;
      end
  end

```

```

if Three_point_bending then CheckBox8.Checked := True
  else CheckBox8.Checked := False;
if sublayer_on_formoptions then CheckBox13.Checked := True
  else CheckBox13.Checked := False;
if normalization_on_formoptions then CheckBox14.Checked := True
  else CheckBox14.Checked := False;
if ModelDechanneling = 1 then ModelDechF.ChDechEn.Checked := True
  else ModelDechF.ChDechDistance.Checked := True;
if Potential_Axil_Plane then FormOptions.CheckBox17.Checked := True
  else FormOptions.CheckBox17.Checked := False;
if AllDataAveraging then ModelDechF.RadioButton1.Checked := True
  else ModelDechF.RadioButton2.Checked := True;
end;
end;
except
end;
ZTableMSdl.Close;
TrAction.Panels[1].Text:='Загрузка данных...';
Min := 0;
case SpLoadTr of
False:
begin
  Dataset := TZQuery.Create(Self);
  Dataset.Connection := Connection;
  Dataset.CachedUpdates := True;
  Dataset.SQL.Text := 'select count(id) from ' + tmpTableNameedb;
  DatasetNewFields;
  Max := Dataset.Fields[0].Value;
  Gauge1.MinValue:=Min;
  Gauge1.MaxValue:=Max - 1;
  Gauge1.Progress:=0;
  r := 1;
  Tr_i := 0;
  Tr_b := 10000;
  Count_of_Load := Round(Max/Tr_b);
  if Count_of_Load<=2 then Load_without_QueryCount:=True;
  if Load_without_QueryCount then
    begin
      QueryCount_Left := 1;
      QueryCount_Right := Max;
      Dataset.SQL.Text := 'select * from ' + tmpTableNameedb + ' where id between '
        + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
    end

```



```

else
begin
    QueryCount_Left := 1+Tr_i*Tr_b;
    if (Tr_i=Count_of_Load-1) then QueryCount_Right:= Max
    else QueryCount_Right := Tr_b+Tr_i*Tr_b;
    Dataset.SQL.Text := 'select * from ' + tmpTableName + ' where id between '
        + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
end;
DatasetNewFields;
if gamev=0 then
for i:=0 to Nm-1 do
begin
    Trajects.Traject[i].sloi:= Dataset.Fields[3].Value;
    if (r=QueryCount_Right) and (Load_without_QueryCount = False) and (r>Max)
    then
begin
    inc(Tr_i);
    QueryCount_Left := 1+Tr_i*Tr_b;
    if (Tr_i=Count_of_Load-1) then QueryCount_Right:= Max
    else QueryCount_Right := Tr_b+Tr_i*Tr_b;
    Dataset.SQL.Text := 'select * from ' + tmpTableName + ' where id between '
        + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
    DatasetNewFields;
end
    else
        Dataset.Next;
inc(r);
Gauge1.Progress:=Gauge1.Progress+1;
for j:=0 to cnt-1 do
begin
    Trajects.Traject[i].v_arr[j]:= Dataset.Fields[3].Value;
    Trajects.Traject[i].x_arr[j]:= Dataset.Fields[4].Value;
    deltaE[i,j]:= Dataset.Fields[5].Value;
    if Dataset.Fields[6].Value=NULL then
        Trajects.Traject[i].f_arr[j]:= 0
    else Trajects.Traject[i].f_arr[j]:= Dataset.Fields[6].Value;
    if (r=QueryCount_Right) and (Load_without_QueryCount = False) and (r>Max)
    then
begin
    inc(Tr_i);
    QueryCount_Left:= 1+Tr_i*Tr_b;
    if (Tr_i=Count_of_Load-1) then QueryCount_Right:= Max
    else QueryCount_Right:= Tr_b+Tr_i*Tr_b;

```

```

        Dataset.SQL.Text := 'select * from ' + tmpTableNameedb + ' where id between '
            + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
        DatasetNewFields;
    end
    else
        Dataset.Next;
    inc(r);
    Gauge1.Progress := Gauge1.Progress+1;
    Application.ProcessMessages;
end;
end
else if gamev=1 then
    for i:=0 to Nm-1 do
        begin
            Trajects.Traject[i].sloi:= Dataset.Fields[3].Value;
            if (r=QueryCount_Right) and (Load_without_QueryCount = False) and (r<>Max)
            then
                begin
                    inc(Tr_i);
                    QueryCount_Left:= 1+Tr_i*Tr_b;
                    if (Tr_i=Count_of_Load-1) then QueryCount_Right:= Max
                    else QueryCount_Right:= Tr_b+Tr_i*Tr_b;
                    Dataset.SQL.Text := 'select * from ' + tmpTableNameedb + ' where id between '
                        + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
                    DatasetNewFields;
                end
            else
                Dataset.Next;
            inc(r);
            Gauge1.Progress:=Gauge1.Progress+1;
            for j:=0 to cnt-1 do
                begin
                    Trajects.Traject[i].v_arr[j]:= Dataset.Fields[3].Value;
                    Trajects.Traject[i].x_arr[j]:= Dataset.Fields[4].Value;
                    deltaE[i,j]:= Dataset.Fields[5].Value;
                    if Dataset.Fields[6].Value=NULL then
                        Trajects.Traject[i].f_arr[j]:= 0
                    else Trajects.Traject[i].f_arr[j]:= Dataset.Fields[6].Value;
                    if (r=QueryCount_Right) and (Load_without_QueryCount = False) and (r<>Max)
                    then
                        begin
                            inc(Tr_i);
                            QueryCount_Left:= 1+Tr_i*Tr_b;

```

```

        if (Tr_i=Count_of_Load-1) then QueryCount_Right:= Max
        else QueryCount_Right:= Tr_b+Tr_i*Tr_b;
        Dataset.SQL.Text := 'select * from ' + tmpTableName + ' where id between '
        + IntToStr(QueryCount_Left) + ' and ' + IntToStr(QueryCount_Right);
        Dataset.NewFields;
    end
    else
        Dataset.Next;
    inc(r);
    Gauge1.Progress:=Gauge1.Progress+1;
    Application.ProcessMessages;
end;
end;
Dataset.Destroy;
end;
True:
begin
    r := FSpLoad.ResultS.Count;
    Max := cnt_SpLoad*Nm*r;
    Gauge1.MinValue:=Min;
    Gauge1.MaxValue:=Max - 1;
    Gauge1.Progress:=0;
    Dataset := TZQuery.Create(Self);
    Dataset.Connection := Connection;
    Dataset.CachedUpdates := True;
    w:=0; j:=0; j_first_in_tau:=0; m_first_in_tau:=0;
    for b := 0 to r - 1 do
        begin
            Dataset.SQL.Text := 'select * from ' + FSpLoad.ResultS[b] + 'db';
            Dataset.NewFields;
            if gamev = 0 then
                for i:=0 to Nm-1 do
                    begin
                        Trajects.Traject[i].sloi:= Dataset.Fields[3].Value;
                        Dataset.Next;
                        j := j_first_in_tau;
                        m := m_first_in_tau;
                        while j <= cnt - 1 do
                            begin
                                Trajects.Traject[i].v_arr[m]:= Dataset.Fields[3].Value;
                                Trajects.Traject[i].x_arr[m]:= Dataset.Fields[4].Value;
                                deltaE[i,m]:= Dataset.Fields[5].Value;
                                if Dataset.Fields[6].Value=NULL then

```

```

        Trajects.Traject[i].f_arr[m]:= 0
        else Trajects.Traject[i].f_arr[m]:= Dataset.Fields[6].Value;
    m := m + 1;
    j := j + N_step_uzel;
    if j <= cnt - 1 then Dataset.MoveBy(N_step_uzel);
    Gauge1.Progress := Gauge1.Progress+1;
    Application.ProcessMessages;
end;
end
else if gamev = 1 then
    for i := 0 to Nm - 1 do
        begin
            Dataset.Locate('duzel;dtrajectory', VarArrayOf(['-1',i]),[]);
            Trajects.Traject[i].sloi:= Dataset.Fields[3].Value;
            Dataset.Next;
            j := j_first_in_tau;
            m := m_first_in_tau;
            if j <> 0 then Dataset.MoveBy(j);
            while j <= cnt - 1 do
                begin
                    Trajects.Traject[i].v_arr[m]:= Dataset.Fields[3].Value;
                    Trajects.Traject[i].x_arr[m]:= Dataset.Fields[4].Value;
                    deltaE[i,m]:= Dataset.Fields[5].Value;
                    if Dataset.Fields[6].Value=Null then
                        Trajects.Traject[i].f_arr[m]:= 0
                        else Trajects.Traject[i].f_arr[m]:= Dataset.Fields[6].Value;
                    m := m + 1;
                    j := j + N_step_uzel;
                    if j <= cnt - 1 then Dataset.MoveBy(N_step_uzel);
                    Gauge1.Progress:=Gauge1.Progress+1;
                    Application.ProcessMessages;
                end;
            end;
            w := cnt - 1 - j;
            j_first_in_tau := (-1)* w;
            m_first_in_tau := m;
        end;
        Dataset.Destroy;
    end;
end;

if SpLoadTr then
    begin

```

```

tau := tau_As_Step_Saving * N_step_uzel;
for i:=0 to Nm-1 do
  begin
    Trajects.Traject[i].t_arr[0]:=Trajects.Traject[0].t_arr[0];
    step_solving_on_tau:=step_on_tau;
    for j:=1 to cnt_SpLoad-1 do
      begin
        Trajects.Traject[i].t_arr[j]:=Trajects.Traject[i].t_arr[j-1]+tau;
        Application.ProcessMessages;
      end;
    end;
    for i:=0 to Nm-1 do Emax[i] := E - deltaE[i, cnt_SpLoad - 1];
    glub := cnt_SpLoad-1;
  end
else
  begin
    tau := tau_As_Step_Saving;
    for i:=0 to Nm-1 do
      begin
        Trajects.Traject[i].t_arr[0]:=Trajects.Traject[0].t_arr[0];
        step_solving_on_tau:=step_on_tau;
        for j:=1 to cnt-1 do
          begin
            if
j=Round((cnt_for_Saving_if_N_Point_True_on_one_step_tau)*step_solving_on_tau/step_on_tau)
then
              begin
                Trajects.Traject[i].t_arr[j]:=step_solving_on_tau;
                step_solving_on_tau:=step_solving_on_tau+step_on_tau;
              end
            else Trajects.Traject[i].t_arr[j]:=Trajects.Traject[i].t_arr[j-1]+tau;
            Application.ProcessMessages;
          end;
        end;
        tau := Trajects.Traject[0].tau;
        for i:=0 to Nm-1 do Emax[i] := E - deltaE[i, cnt - 1];
        glub := cnt-1;
      end;
    LabeledEdit4.Text:=FloatToStr(RoundTo(Trajects.Traject[0].t_arr[0],-2));
    SetCoeff;
    ChangeOptions;
    if (TypeMyFile = 1) then
      begin

```

```

    i := Pos('Траекторий:', inscription);
    insert('1 из ', inscription, i+12);
end;
if (TypeMyFile = 2) then
    begin
        i := Length(inscription);
        insert(' Последний слой', inscription, i+2);
    end;
if (TypeMyFile = 3) then
    begin
        i := Length(inscription);
        if SpLoadTr then
            insert(' ' + IntToStr(r) + ' слоя ', inscription, i+2)
        else
            begin
                s := tmpTableNamedb;
                Delete(s, (Length(tmpTableName) - 6), 7);
                Delete(s, 1, Pos('_layer', s) + 5);
                Delete(s, Pos('_db', s), Pos('_db', s) + 3);
                insert(' Слой № ' + s, inscription, i+2);
            end;
        end;
        Form1.Caption := inscription;
        ContinueCalculation.Enabled:=True;
        Gauge1.MaxValue:=100;
        Gauge1.Progress := Gauge1.MaxValue;
        LEGraphT0.Text:=LabeledEdit4.Text;
        LEGraphTLast.Text:=LabeledEdit1.Text;
        OuterLoad:=True;
        LoadMyDataDone := True;
        TrAction.Panels[1].Text:='Загрузка успешно завершена...';
        Sep5.Visible := True;
        AbFilesButton.Visible := True;
        AbFilesShow(ExtractFilePath(tmpOpenFileName), tmpTableNamedb, tmpTableNamedl);
        Connection.Connected := False;
        DestroyOurConnection;
        LoadWithParameters := False;
    end;

procedure TForm1.CompatibilityMyFiles(FullDirMyTable: string; ArrMyTables: TStrings);
var
    tmp_sCompat, sCompat, s, tmpDirServer, tmpTableName, tmpTableNamedb, tmpTableNamedl:
    string;

```

```

a, i, j, ij, tmp_iCompat, iCompat, float_to_int, R_tau: integer;
tmp_fCompat, fCompat: double;
float_to_str: string;
begin
DirAndNameMyDB(FullDirMyTable);
tmpDirServer := DirMyTableName;
tmpTableName := MyTableName;
tmpTableNamedb := MyTableNamedb;
tmpTableNamedl := MyTableNamedl;
CreateOurConnection(tmpTableName, tmpDirServer);
Connection.Connected := True;
Dataset := TZQuery.Create(Self);
Dataset.Connection := Connection;
Dataset.CachedUpdates := True;
Dataset.SQL.Text := 'select * from ' + tmpTableNamedl + ' where id in (13,57,60)';
Dataset.NewFields;
sCompat := Dataset.Fields[1].Value; Dataset.Next;
iCompat := Dataset.Fields[2].Value; Dataset.Next;
fCompat := Dataset.Fields[4].Value;
a_tau := fCompat;
a := ArrMyTables.Count;
for i := 1 to a - 1 do
    begin
        tmpTableNamedl := ArrMyTables.Strings[i] + 'dl';
        Dataset.SQL.Text := 'select * from ' + tmpTableNamedl + ' where id in (13,57,60)';
        Dataset.NewFields;
        tmp_sCompat := Dataset.Fields[1].Value; Dataset.Next;
        tmp_iCompat := Dataset.Fields[2].Value; Dataset.Next;
        tmp_fCompat := Dataset.Fields[4].Value;
        if (Round(tmp_fCompat) = StrToInt(sCompat)) and (tmp_iCompat = iCompat) then
            begin
                sCompat := tmp_sCompat;
                iCompat := tmp_iCompat;
                fCompat := tmp_fCompat;
            end
        else
            begin
                FSpLoad.Label6.Caption := 'Совместимость: файлы не совместимы';
                Exit;
            end;
        end;
    end;
b_tau := StrToInt(tmp_sCompat);
R_tau := tmp_iCompat;

```

```

FSpLoad.Label6.Caption := 'Совместимость: Все в порядке. Файлы совместимы.';
Dataset.SQL.Text := 'select * from ' + tmpTableName + ' where id in (5,56,57,63)';
Dataset.NewFields;
tmp_sCompat := Dataset.Fields[1].Value; Dataset.Next;
tmp_iCompat := Dataset.Fields[2].Value; Dataset.Next;
iCompat := Dataset.Fields[2].Value; Dataset.Next;
tmp_fCompat := Dataset.Fields[4].Value;
FSpLoad.LabStepOnTau.Caption := 'Шаг по тау: ' + FloatToStr(tmp_fCompat);
FSpLoad.LabPointOnStep.Caption := 'Количество сохраненных точек на шаг по тау: ' + IntToStr(iCompat);
FSpLoad.LabCountfor1Tr.Caption := 'Всего точек для 1 траектории: ' + IntToStr(tmp_iCompat);
FSpLoad.LabCountTr.Caption := 'Траекторий: ' + tmp_sCompat;
cnt_SpLoad := Trunc( ((b_tau-a_tau-1)*(Round(R_tau/tmp_fCompat)-1)+
Round(R_tau/tmp_fCompat))/N_step_uzel ) + 1;
FSpLoad.Edit1.Text := IntToStr(cnt_SpLoad);
Dataset.Destroy;
Connection.Connected := False;
DestroyOurConnection;
FSpLoad.ButRunLoad.Enabled := True;
end;

```

```

function TForm1.CreateDirForTrj:string;

```

```

var

```

```

sdir:string;

```

```

Present: TDateTime;

```

```

Hour, Min, Sec, MSec: Word;

```

```

begin

```

```

Present:=Now;

```

```

DecodeTime(Present, Hour, Min, Sec, MSec);

```

```

sdir:=DateToStr(Date)+' '+IntToStr(Hour)+'ч.'+IntToStr(Min)+'м Si ('+tip_ploskost+')

```

```

Z1:='LEZ1.Text+' Траекторий '+FloatToStr(Allseria.AllTrCnt)+

```

```

' Глубина '+LabeledEdit1.Text+'

```

```

('+FloatToStr(RoundTo(Trajects.Traject[0].t_arr[Trajects.Traject[0].Nmax]/epsilon*d_small,-2))+
A) '+'

```

```

' E='+LEEnergy.Text+ ' эВ Узлов '+LabeledEdit2.Text+

```

```

' Шаг '+FloatToStr(RoundTo(Trajects.Traject[0].tau,-5))+

```

```

('+FloatToStr(RoundTo(Trajects.Traject[0].tau/epsilon*d_small,-2))+ A)';

```

```

if not DirectoryExists(sdir) then

```

```

    if not CreateDir(sdir) then

```

```

        begin

```

```

            ShowMessage('Ошибка создания папки'+sdir);

```

```

            sdir:="";

```

```

        end;

```



```
CreateDirForTrj:=sdir;  
end;
```

```
function TForm1.CreateInscription:string;  
var  
  inscr:string;  
begin  
  inscr:=CrystalSolving + ' ('+tip_ploskost+') Z1=' + Z1Solving + ' Траекторий:  
' + FloatToStr(Allseria.AllTrCnt) +  
  ' E=' + LEEnergy.Text + ' эВ';  
  CreateInscription:=inscr;  
end;
```

```
procedure TForm1.ToolButton7Click(Sender: TObject);  
var  
  sdir:string;  
begin  
  sdir:=CreateDirForTrj  
end;
```

```
procedure TForm1.BitBtn2Click(Sender: TObject);  
begin  
  OKBottomDlg1.ShowModal  
end;
```

```
procedure TForm1.SaveMenuTrj(Sender: TObject);  
begin  
  SaveAllTrajectory(1, "");  
end;
```

```
procedure TForm1.LabeledEdit1MouseMove(Sender: TObject; Shift: TShiftState;  
  X, Y: Integer);  
var  
  code:integer;  
  r:double;  
begin  
  Val(LabeledEdit1.Text,r,code);  
  if code=0 then  
    begin  
      LabeledEdit1.Hint:=FloatToStr(RoundTo(r/epsilon*d_small,-2))+ ' A ';  
      LabeledEdit1.ShowHint:=True;  
    end  
  end
```

```

    else LabeledEdit1.ShowHint:=False;
end;

procedure TForm1.Vx1Click(Sender: TObject);
var
    i,k: integer;
    SaveF: TextFile;
begin
    SaveDialog1.DefaultExt:= 'txt';
    SaveDialog1.Filter := 'Текстовые файлы (*.txt)|*.txt';
    DecimalSeparator:='.';
    if SaveDialog1.Execute then
        begin
            AssignFile(SaveF,SaveDialog1.FileName);
            Rewrite(SaveF);
            k:=Trajects.CntOfTraject;
            Writeln(SaveF,k);
            for i:=0 to k-1 do Writeln(SaveF,FloatToStr(Trajects.Traject[i].v_arr[0]));
            CloseFile(SaveF);
        end;
    DecimalSeparator:='.';
end;

function TForm1.FloatForAccess(N:Variant):String;
var s:string;
    i:integer;
begin
    if VarIsNull(N) then begin
        Result:='NULL';
        exit;
    end;
    s:=FloatToStr(N);
    For i:=1 to Length(S) do begin
        if s[i]=' ' then s[i]:='.';
    end;
    Result:=s;
end;

function TForm1.DateForMySQL(D:Variant):string;
begin
    if VarIsNull(D) then begin
        Result:='NULL';
        exit;

```

```

end;
Result:="" + FormatDateTime('yyyy"."mm"."dd',D) + "";
end;

```

```

function TForm1.VarToSQL(AValue:Variant):string;
begin
  case VarType(AValue) of
    varString,varUnknown,varStrArg:Result:="" + VarToStr(AValue) + "";
    varSmallint,varInteger,varInt64,varWord,varLongWord:Result:=VarAsType(AValue,varInteger);
    varEmpty,varNull:Result:='NULL';
    varDouble,varCurrency:Result:=FloatForAccess(AValue);
    varDate:Result:=DateForMySQL(AValue);
  else
    Result:='NULL';
  end;
end;

```

```

procedure TForm1.SaveFileInitIni(SFileNameIni: string);
begin
with TIniFile.Create(ExtractFilePath(ParamStr(0)) + SFileNameIni) do
  try
    WriteString('InitStr', 'CCrystal', ComboCrystal.Text);
    WriteString('InitStr', 'CPlosk', ComboPlosk.Text);
    WriteString('InitStr', 'Z1', LEZ1.Text);
    WriteString('InitStr', 'CParticles', ComboParticles.Text);
    WriteString('InitStr', 'AEM', LEAEM.Text);
    WriteString('InitStr', 'nx', LEnx.Text);
    WriteString('InitStr', 'z', LEzsmall.Text);
    WriteString('InitStr', 'F1_LE1', LabeledEdit1.Text);
    WriteString('InitStr', 'F1_LE2', LabeledEdit2.Text);
    WriteString('InitStr', 'F1_LE3', LabeledEdit3.Text);
    WriteString('InitStr', 'F1_LE4', LabeledEdit4.Text);
    WriteString('InitStr', 'Energy', LEEnergy.Text);
    WriteString('InitStr', 'Epsilon', LEEpsilon.Text);
    WriteString('InitStr', 'DeltaTheta', LEDeltaTheta.Text);
    WriteString('InitStr', 'CPeriodFrom', LECalcPeriodFrom.Text);
    WriteString('InitStr', 'CalcPeriodTo', LECalcPeriodTo.Text);
    WriteString('InitStr', 'k', LEortk.Text);
    WriteString('InitStr', 'l', LEortl.Text);
    WriteString('InitStr', 'F1_E1', Edit1.Text);
    WriteString('InitStr', 'F1_E2', Edit2.Text);
    WriteString('InitStr', 'MX', LEMultX.Text);
    WriteString('InitStr', 'MY', LEMultY.Text);
  finally
    Free;
  end;

```

```

WriteString('InitStr', 'CX', LECenX.Text);
WriteString('InitStr', 'CY', LECenY.Text);
WriteString('InitStr', 'alpha', Form2.LEalpha.Text);
WriteString('InitStr', 'beta', Form2.LEbeta.Text);
WriteString('InitStr', 'F2_LE1', Form2.LabeledEdit1.Text);
WriteString('InitStr', 'F2_LE2', Form2.LabeledEdit2.Text);
WriteString('InitStr', 'F2_LE3', Form2.LabeledEdit3.Text);
WriteString('InitStr', 'F2_LE4', Form2.LabeledEdit4.Text);
WriteString('InitStr', 'F2_LE5', Form2.LabeledEdit5.Text);
WriteString('InitStr', 'BF_E1', BendingForm.Edit1.Text);
WriteString('InitStr', 'BF_E2', BendingForm.Edit2.Text);
WriteString('InitStr', 'BF_E3', BendingForm.Edit3.Text);
WriteString('InitStr', 'BF_E4', BendingForm.Edit4.Text);
WriteString('InitStr', 'BF_E5', BendingForm.Edit5.Text);
WriteString('InitStr', 'BF_E6', BendingForm.Edit6.Text);
WriteString('InitStr', 'BF_E7', BendingForm.Edit7.Text);
WriteString('InitStr', 'BF_E8', BendingForm.Edit8.Text);
WriteString('InitStr', 'BF_E9', BendingForm.Edit9.Text);
WriteString('InitStr', 'BF_E10', BendingForm.Edit10.Text);
WriteString('InitStr', 'BF_E11', BendingForm.Edit11.Text);
WriteString('InitStr', 'BF_E12', BendingForm.Edit12.Text);
WriteString('InitStr', 'FP_LE1', AdvOptF.LabeledEdit1.Text);
WriteString('InitStr', 'FP_LE3', AdvOptF.LabeledEdit3.Text);
WriteString('InitStr', 'FP_LE4', AdvOptF.LabeledEdit4.Text);
WriteString('InitStr', 'FP_LE5', AdvOptF.LabeledEdit5.Text);
WriteString('InitStr', 'FP_LE8', AdvOptF.LabeledEdit8.Text);
WriteString('InitStr', 'FP_LE9', AdvOptF.LabeledEdit9.Text);
WriteString('InitStr', 'FP_LE10', AdvOptF.LabeledEdit10.Text);
WriteString('InitStr', 'FP_LE11', AdvOptF.LabeledEdit11.Text);
WriteString('InitStr', 'FP_E11', AdvOptF.Edit11.Text);
WriteString('InitStr', 'FO_E13', FormOptions.Edit13.Text);
WriteString('InitStr', 'FO_E14', FormOptions.Edit14.Text);
WriteString('InitStr', 'FO_E15', FormOptions.Edit15.Text);
WriteString('InitStr', 'FO_LE1', FormOptions.LabeledEdit1.Text);
WriteString('InitStr', 'FO_LE2', FormOptions.LabeledEdit2.Text);
WriteString('InitStr', 'FO_LE3', FormOptions.LabeledEdit3.Text);
WriteString('InitStr', 'FO_LE4', FormOptions.LabeledEdit4.Text);
WriteString('InitStr', 'FO_LE5', FormOptions.LabeledEdit5.Text);
WriteString('InitStr', 'FO_LE6', FormOptions.LabeledEdit6.Text);
WriteString('InitStr', 'FO_DBName', FormOptions.LEDBName.Text);
WriteString('InitStr', 'DataDirMySQL', TDataDirMySQL);
WriteString('InitStr', 'EdEffect1', PlotF.EdEffectFrom.Text);
WriteString('InitStr', 'EdEffect2', PlotF.EdEffectTo.Text);

```

```

WriteString('InitStr', 'PathSpL', PathForSpecLoad);
WriteString('InitInt', 'gamev', SaveIntIni(gamev));
WriteString('InitInt', 'tipV', SaveIntIni(tipV));
WriteString('InitInt', 'OverBarrierOff', SaveIntIni(OverBarrierOff));
WriteString('InitInt', 'BorOff', SaveIntIni(BorOff));
WriteString('InitInt', 'SrKvFlEp', SaveIntIni(SrKvFlEp));
WriteString('InitInt', 'VMoilerOff', SaveIntIni(VMoilerOff));
WriteString('InitBool', 'EnergyLoss', SaveBoolIni(EnergyLossVariant4Off));
WriteString('InitBool', 'FO_C7', SaveBoolIni(FormOptions.CheckBox7.Checked));
WriteString('InitBool', 'FO_C11', SaveBoolIni(FormOptions.CheckBox11.Checked));
WriteString('InitBool', 'FO_C9', SaveBoolIni(FormOptions.CheckBox9.Checked));
WriteString('InitBool', 'FO_C10', SaveBoolIni(FormOptions.CheckBox10.Checked));
WriteString('InitBool', 'FO_C6', SaveBoolIni(FormOptions.CheckBox6.Checked));
WriteString('InitBool', 'VGauss', SaveBoolIni(VGauss));
WriteString('InitBool', 'ASolvingExit', SaveBoolIni(Auto_Solving_Exit));
WriteString('InitBool', 'ThPointBend', SaveBoolIni(Three_point_bending));
WriteString('InitBool', 'SubOnForOpt', SaveBoolIni(sublayer_on_formoptions));
WriteString('InitBool', 'NormOnForOpt', SaveBoolIni(normalization_on_formoptions));
WriteString('InitBool', 'F11_C11', SaveBoolIni(PlotF.CheckBox11.Checked));
WriteString('InitBool', 'F11_C1', SaveBoolIni(PlotF.CheckBox1.Checked));
WriteString('InitBool', 'F11_C2', SaveBoolIni(PlotF.CheckBox2.Checked));
WriteString('InitBool', 'PF_Effect', SaveBoolIni(PlotF.ChEffect.Checked));
WriteString('InitBool', 'MD_ChDecEn', SaveBoolIni(ModelDechF.ChDechEn.Checked));
WriteString('InitBool', 'MD_ChDecDist', SaveBoolIni(ModelDechF.ChDechDistance.Checked));
WriteString('InitBool', 'PotentialAxPl', SaveBoolIni(Potential_Axil_Plane));
WriteString('InitBool', 'AllDataAverag', SaveBoolIni(AllDataAveraging));
finally
  Free;
end;

```

end;

```

procedure TForm1.LoadFileInitIni(LFileNameIni: string);
var
  code: integer;
begin
  with TIniFile.Create(ExtractFilePath(ParamStr(0)) + LFileNameIni) do
    try
      ComboCrystal.Text := ReadString('InitStr', 'CCrystal', '');
      ComboPlosk.Text := ReadString('InitStr', 'CPlosk', '');
      LEZ1.Text := ReadString('InitStr', 'Z1', '');
      ComboParticles.Text := ReadString('InitStr', 'CParticles', '');
      LEAEM.Text := ReadString('InitStr', 'AEM', '');

```

```

LEnx.Text := ReadString('InitStr', 'nx', "");
LEzsmall.Text := ReadString('InitStr', 'z', "");
LabeledEdit1.Text := ReadString('InitStr', 'F1_LE1', "");
LabeledEdit2.Text := ReadString('InitStr', 'F1_LE2', "");
LabeledEdit3.Text := ReadString('InitStr', 'F1_LE3', "");
LabeledEdit4.Text := ReadString('InitStr', 'F1_LE4', "");
LEEnergy.Text := ReadString('InitStr', 'Energy', "");
LEEpsilon.Text := ReadString('InitStr', 'Epsilon', "");
LEDeltaTheta.Text := ReadString('InitStr', 'DeltaTheta', "");
LECalcPeriodFrom.Text := ReadString('InitStr', 'CPeriodFrom', "");
LECalcPeriodTo.Text := ReadString('InitStr', 'CalcPeriodTo', "");
LEortk.Text := ReadString('InitStr', 'k', "");
LEortl.Text := ReadString('InitStr', 'l', "");
Edit1.Text := ReadString('InitStr', 'F1_E1', "");
Edit2.Text := ReadString('InitStr', 'F1_E2', "");
LEMultX.Text := ReadString('InitStr', 'MX', "");
LEMultY.Text := ReadString('InitStr', 'MY', "");
LECenX.Text := ReadString('InitStr', 'CX', "");
LECenY.Text := ReadString('InitStr', 'CY', "");
LEalpha := ReadString('InitStr', 'alpha', "");
LEbeta := ReadString('InitStr', 'beta', "");
VXTr_from_AddTrj[1] := ReadString('InitStr', 'F2_LE1', "");
VXTr_from_AddTrj[2] := ReadString('InitStr', 'F2_LE2', "");
VXTr_from_AddTrj[3] := ReadString('InitStr', 'F2_LE3', "");
VXTr_from_AddTrj[4] := ReadString('InitStr', 'F2_LE4', "");
VXTr_from_AddTrj[5] := ReadString('InitStr', 'F2_LE5', "");
Edit_from_BendingForm[1] := ReadString('InitStr', 'BF_E1', "");
Edit_from_BendingForm[2] := ReadString('InitStr', 'BF_E2', "");
Edit_from_BendingForm[3] := ReadString('InitStr', 'BF_E3', "");
Edit_from_BendingForm[4] := ReadString('InitStr', 'BF_E4', "");
Edit_from_BendingForm[5] := ReadString('InitStr', 'BF_E5', "");
Edit_from_BendingForm[6] := ReadString('InitStr', 'BF_E6', "");
Edit_from_BendingForm[7] := ReadString('InitStr', 'BF_E7', "");
Edit_from_BendingForm[8] := ReadString('InitStr', 'BF_E8', "");
Edit_from_BendingForm[9] := ReadString('InitStr', 'BF_E9', "");
Edit_from_BendingForm[10] := ReadString('InitStr', 'BF_E10', "");
Edit_from_BendingForm[11] := ReadString('InitStr', 'BF_E11', "");
Edit_from_BendingForm[12] := ReadString('InitStr', 'BF_E12', "");
Edit_from_FormPlotting[1] := ReadString('InitStr', 'FP_LE1', "");
Edit_from_FormPlotting[2] := ReadString('InitStr', 'FP_LE3', "");
Edit_from_FormPlotting[3] := ReadString('InitStr', 'FP_LE4', "");
Edit_from_FormPlotting[4] := ReadString('InitStr', 'FP_LE5', "");
Edit_from_FormPlotting[5] := ReadString('InitStr', 'FP_LE8', "");

```

```

Edit_from_FormPlotting[6] := ReadString('InitStr', 'FP_LE9', "");
Edit_from_FormPlotting[7] := ReadString('InitStr', 'FP_LE10', "");
Edit_from_FormPlotting[8] := ReadString('InitStr', 'FP_LE11', "");
Edit_from_FormPlotting[9] := ReadString('InitStr', 'FP_E11', "");
Edit_from_optiongamev[1] := ReadString('InitStr', 'FO_E13', "");
Edit_from_optiongamev[2] := ReadString('InitStr', 'FO_E14', "");
Edit_from_optiongamev[3] := ReadString('InitStr', 'FO_E15', "");
Val(ReadString('InitStr', 'FO_LE1', ""), porog_max_eV2, code);
Val(ReadString('InitStr', 'FO_LE2', ""), porog_min_eV2, code);
porog_min_eV2 := 0.0001;
Val(ReadString('InitStr', 'FO_LE3', ""), shag_A, code);
Val(ReadString('InitStr', 'FO_LE4', ""), min_shag_A, code);
Auto_Solving_with_step := ReadString('InitStr', 'FO_LE5', "");
N_Point_Save_Value_Str := ReadString('InitStr', 'FO_LE6', "");
Database_Name := ReadString('InitStr', 'FO_DBName', "");
TDataDirMySQL := ReadString('InitStr', 'DataDirMySQL', "");
Edit_from_PlotF[1] := ReadString('InitStr', 'EdEffect1', "");
Edit_from_PlotF[2] := ReadString('InitStr', 'EdEffect2', "");
PathForSpecLoad := ReadString('InitStr', 'PathSpL', "");
gamev := LoadIntIni(ReadString('InitInt', 'gamev', ""));
tipV := LoadIntIni(ReadString('InitInt', 'tipV', ""));
OverBarrierOff := LoadIntIni(ReadString('InitInt', 'OverBarrierOff', ""));
BorOff := LoadIntIni(ReadString('InitInt', 'BorOff', ""));
SrKvFlEp := LoadIntIni(ReadString('InitInt', 'SrKvFlEp', ""));
VMoilerOff := LoadIntIni(ReadString('InitInt', 'VMoilerOff', ""));
EnergyLossVariant4Off := LoadBoolIni(ReadString('InitBool', 'EnergyLoss', ""));
Auto_Save := LoadBoolIni(ReadString('InitBool', 'FO_C7', ""));
N_point_Save := LoadBoolIni(ReadString('InitBool', 'FO_C11', ""));
Last_Layer_Save := LoadBoolIni(ReadString('InitBool', 'FO_C9', ""));
Last_Layer_ReSave := LoadBoolIni(ReadString('InitBool', 'FO_C10', ""));
Auto_Solving := LoadBoolIni(ReadString('InitBool', 'FO_C6', ""));
VGauss := LoadBoolIni(ReadString('InitBool', 'VGauss', ""));
Auto_Solving_Exit := LoadBoolIni(ReadString('InitBool', 'ASolvingExit', ""));
Three_point_bending := LoadBoolIni(ReadString('InitBool', 'ThPointBend', ""));
sublayer_on_formoptions := LoadBoolIni(ReadString('InitBool', 'SubOnForOpt', ""));
normalization_on_formoptions := LoadBoolIni(ReadString('InitBool', 'NormOnForOpt', ""));
Form11_Checked[1] := LoadBoolIni(ReadString('InitBool', 'F11_C11', ""));
Form11_Checked[2] := LoadBoolIni(ReadString('InitBool', 'F11_C1', ""));
Form11_Checked[3] := LoadBoolIni(ReadString('InitBool', 'F11_C2', ""));
Form11_Checked[4] := LoadBoolIni(ReadString('InitBool', 'PF_Effect', ""));
FOptions_Checked[1] := LoadBoolIni(ReadString('InitBool', 'MD_ChDecEn', ""));
FOptions_Checked[2] := LoadBoolIni(ReadString('InitBool', 'MD_ChDecDist', ""));
Potential_Axil_Plane := LoadBoolIni(ReadString('InitBool', 'PotentialAxPl', ""));

```

```

    AllDataAveraging := LoadBoolIni(ReaDString('InitBool', 'AllDataAverag', ''));
finally
    Free;
end;
TrAction.Panels[1].Text := 'Загрузка начальных данных выполнена...';
Timer2.Enabled := True;
end;

function TForm1.SaveBoolIni(BoolValue: boolean): string;
var
    iBoolValue: integer;
    sBoolValue: string;
begin
    Result := BoolToStr(BoolValue, True);
end;

function TForm1.LoadBoolIni(sBoolValue: string):boolean;
begin
    try
        if sBoolValue='False' then LoadBoolIni := False
        else if sBoolValue='True' then LoadBoolIni := True
        else LoadBoolIni := False;
    except
    end;
end;

function TForm1.SaveIntIni(IntValue: integer):string;
begin
    Result := IntToStr(IntValue);
end;

function TForm1.LoadIntIni(sIntValue: string):integer;
begin
    if sIntValue<>" then
        Result := StrToInt(sIntValue)
    else Result := 0;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
    Gauge1.Progress:=0;
    TrAction.Panels[1].Text:='...';
    Timer2.Enabled:=False;

```



```

if LoadWithParameters then LoadAllTrajectory(Self);
end;

procedure TForm1.Solving_Energy_Loss(ModelDech: integer; Dechennaling_ with podsloi,
Normalization: boolean; SEL_Nsloev, SEL_Nm, SEL_Nmax: integer;
SEL_Lev_dE, SEL_Prav_dE, Normalization_Value: double; SEL_FileName: string);
var
SEL_Podsloi: integer;
SEL_Nch_na_No: array of double;
SEL_Podsloi_t, SEL_dE_na_dL, SEL_norm_koef: double;
SEL_X, SEL_Y: array of double;
di, i, ia, ij, bv: integer;
SaveF: TextFile;
str: string;
dbl: double;
SEL_Uxx, tmpEp: double;
begin
try
case ModelDech of
1:
begin
if Dechennaling_ with podsloi then
begin
if ((SEL_Nmax+1) mod SEL_Nsloev) <> 0 then
begin
ShowMessage('Число подслоев не кратно числу узлов');
Exit;
end;
end;
SEL_Podsloi:=Round((SEL_Nmax+1)/SEL_Nsloev)-1;
SEL_Podsloi_t:=Trajects.Traject[0].tau*SEL_Podsloi;
SetLength(SEL_Nch_na_No, SEL_Nsloev);
di:=0;
bv:=0;
if AllDataAveraging = False then begin
for i:=0 to SEL_Nm-1 do
begin
while di <= SEL_Nsloev-1 do
begin
SEL_dE_na_dL:=(deltaE[i, (di+1)*SEL_Podsloi+bv]-deltaE[i,
(di)*SEL_Podsloi+bv])/(SEL_Podsloi_t/epsilon*d_small);
if ((SEL_dE_na_dL)>SEL_Lev_dE) and ((SEL_dE_na_dL)<SEL_Prav_dE) then
SEL_Nch_na_No[di]:=SEL_Nch_na_No[di]+1/SEL_Nm;
di := di + 1;

```

```

    bv := bv + 1;
  end;
  di := 0;
  bv := 0;
end;
else
  begin
    for i:=0 to SEL_Nm-1 do
      begin
        bv:=0;
        for di:=0 to SEL_Nsloev-1 do
          begin
            while bv<>(SEL_Podsloi+1)*(di+1) do
              begin
                SEL_dE_na_dL := deltaE[i,bv]/(Trajects.Traject[i].t_arr[bv]/epsilon*d_small);
                if (SEL_dE_na_dL > SEL_Lev_dE) and (SEL_dE_na_dL < SEL_Prav_dE) then
SEL_Nch_na_No[di]:=SEL_Nch_na_No[di]+1/(SEL_Nm*(SEL_Podsloi+1));
                  inc(bv);
                end;
              end;
            end;
          end;
        end;
      end;
    else
      begin
        SetLength(SEL_Nch_na_No,SEL_Nmax+1);
        for i:=0 to SEL_Nm-1 do
          for di:=0 to SEL_Nmax do
            if Trajects.Traject[0].t_arr[di]=0 then
              begin
                SEL_Nch_na_No[di]:=1;
              end
            else
              begin
                SEL_dE_na_dL:=deltaE[i,di]/(Trajects.Traject[i].t_arr[di]/epsilon*d_small);
                if ((SEL_dE_na_dL)>SEL_Lev_dE) and ((SEL_dE_na_dL)<SEL_Prav_dE) then
SEL_Nch_na_No[di]:=SEL_Nch_na_No[di]+1/SEL_Nm;
                  end;
                end;
              end;
            end;
          end;
        end;
      if Dechennaling_with_podsloi then
        begin
          if (Auto_Solving) and (SEL_XY_was_first=False) then

```

```

begin
  SetLength(SEL_X, 2+SEL_Nsloev);
  SetLength(SEL_Y, 2+SEL_Nsloev);
  ia:=2+SEL_Nsloev;
  ij:=0;
end
else if (Auto_Solving) and (SEL_XY_was_first) then
  begin
    SetLength(SEL_X, 1+SEL_Nsloev);
    SetLength(SEL_Y, 1+SEL_Nsloev);
    ia:=1+SEL_Nsloev;
    ij:=1;
  end
  else
    begin
      SetLength(SEL_X, 3+SEL_Nsloev);
      SetLength(SEL_Y, 3+SEL_Nsloev);
      ia:=3+SEL_Nsloev;
      ij:=0;
    end;
  end;
if Dechennaling_with_podsloi=False then
  begin
    if (Auto_Solving) and (SEL_XY_was_first=False) then
      begin
        SetLength(SEL_X, 2+SEL_Nmax);
        SetLength(SEL_Y, 2+SEL_Nmax);
        ia:=2+SEL_Nmax;
        ij:=0;
      end
    end
    else if (Auto_Solving) and (SEL_XY_was_first) then
      begin
        SetLength(SEL_X, 1+SEL_Nmax);
        SetLength(SEL_Y, 1+SEL_Nmax);
        ia:=1+SEL_Nmax;
        ij:=1;
      end
    else
      begin
        SetLength(SEL_X, 2+SEL_Nmax);
        SetLength(SEL_Y, 2+SEL_Nmax);
        ia:=2+SEL_Nmax;
        ij:=0;

```

```

        end;
    end;
    if (Auto_Solving=False) or ((Auto_Solving) and (SEL_XY_was_first=False)) then
        begin
            SEL_X[0]:=0;
            SEL_Y[0]:=0;
        end;
    if Normalization then SEL_norm_koef:=Normalization_Value else SEL_norm_koef:=1;
    if Dechennaling_with_podsloi then
        begin
            SEL_X[1-ij]:=Trajects.Traject[0].t_arr[0]/epsilon*d_small;
            SEL_Y[1-ij]:=(1-SEL_Nch_na_No[0])/SEL_norm_koef;
            bv:=0;
            for di:=0 to SEL_Nsloev-1 do
                begin
                    SEL_X[2+di-ij]:=Trajects.Traject[0].t_arr[(di+1)*SEL_Podsloi+bv]/epsilon*d_small;
                    SEL_Y[2+di-ij]:=(1-SEL_Nch_na_No[di])/SEL_norm_koef;
                    bv:=bv+1;
                end;
            if (Auto_Solving=False) then
                begin
                    SEL_X[2+SEL_Nsloev]:=Trajects.Traject[0].t_arr[SEL_Nsloev*SEL_Podsloi+bv-
1]/epsilon*d_small;
                    SEL_Y[2+SEL_Nsloev]:=0;
                end;
            end
        else
            for di:=0 to SEL_Nmax do
                begin
                    SEL_X[1+di-ij]:=Trajects.Traject[0].t_arr[di]/epsilon*d_small;
                    SEL_Y[1+di-ij]:=(1-SEL_Nch_na_No[di])/SEL_norm_koef;
                end;
            end;
        DecimalSeparator:='.';
        AssignFile(SaveF, SEL_FileName);
        Append(SaveF);
        for i:=0 to ia-1 do
            begin
                str:=FloatToStr(SEL_X[i]) + ' ' + FloatToStr(SEL_Y[i]);
                Writeln(SaveF, str);
            end;
        CloseFile(SaveF);
        DecimalSeparator:='.';
        SEL_XY_was_first:=True;

```

```

end;
2:
begin
  if Dechennaling_with_podsloi then
    begin
      if ((SEL_Nmax+1) mod SEL_Nsloev) <> 0 then
        begin
          ShowMessage('Число подслоев не кратно числу узлов');
          Exit;
        end;
        SEL_Podsloi:=Round((SEL_Nmax+1)/SEL_Nsloev)-1;
        SEL_Podsloi_t:=Trajects.Traject[0].tau*SEL_Podsloi;
        SetLength(SEL_Nch_na_No, SEL_Nsloev);
        di:=0;
        bv:=0;
        if AllDataAveraging = False then begin
          for i:=0 to SEL_Nm-1 do
            begin
              while di <= SEL_Nsloev-1 do
                begin
                  SEL_Uxx := ( U_xx(Trajects.Traject[i].x_arr[(di+1)*SEL_Podsloi+bv]) -
                    U_xx(Trajects.Traject[i].x_arr[(di)*SEL_Podsloi+bv])
                )/(SEL_Podsloi_t/epsilon*d_small);
                  if (SEL_Uxx > 0) then SEL_Nch_na_No[di] := SEL_Nch_na_No[di] + 1/SEL_Nm;
                  di := di + 1;
                  bv := bv + 1;
                end;
                di := 0;
                bv := 0;
              end;
            end;
          else
            begin
              for i:=0 to SEL_Nm-1 do
                begin
                  bv:=0;
                  for di:=0 to SEL_Nsloev-1 do
                    begin
                      while bv<>(SEL_Podsloi+1)*(di+1) do
                        begin
                          tmpEp := Vmax*Sqr(Trajects.Traject[i].v_arr[bv])/2+U(Trajects.Traject[i].v_arr[bv]);
                          if tmpEp < Vmax then
                            begin

```

```

        SEL_Uxx := U_xx(Trajects.Traject[i].x_arr[bv]);
        if SEL_Uxx > 0 then SEL_Nch_na_No[di] :=
SEL_Nch_na_No[di]+1/(SEL_Nm*(SEL_Podsloi+1));
        end;
        inc(bv);
        end;
    end;
end;
end
else
begin
    SetLength(SEL_Nch_na_No,SEL_Nmax+1);
    tmpEp := 0;
    for i:=0 to SEL_Nm-1 do
        begin
            di := 0;
            while di < SEL_Nmax do
                begin
                    tmpEp := Vmax*Sqr(Trajects.Traject[i].v_arr[di])/2+U(Trajects.Traject[i].v_arr[di]);
                    if tmpEp < Vmax then
                        begin
                            SEL_Uxx := U_xx(Trajects.Traject[i].x_arr[di]);
                            if SEL_Uxx > 0 then SEL_Nch_na_No[di] := SEL_Nch_na_No[di]+1/SEL_Nm;
                        end;
                        di := di + 1;
                    end;
                end;
            end;
        end;
    if Dechennaling_with_podsloi then
        begin
            if (Auto_Solving) and (SEL_XY_was_first=False) then
                begin
                    SetLength(SEL_X, 2+SEL_Nsloev);
                    SetLength(SEL_Y, 2+SEL_Nsloev);
                    ia:=2+SEL_Nsloev;
                    ij:=0;
                end
            else if (Auto_Solving) and (SEL_XY_was_first) then
                begin
                    SetLength(SEL_X, 1+SEL_Nsloev);
                    SetLength(SEL_Y, 1+SEL_Nsloev);
                    ia:=1+SEL_Nsloev;

```

```

    ij:=1;
  end
  else
    begin
      SetLength(SEL_X, 3+SEL_Nsloev);
      SetLength(SEL_Y, 3+SEL_Nsloev);
      ia:=3+SEL_Nsloev;
      ij:=0;
    end;
  end;
if Dechennaling_with_podsloi=False then
  begin
    if (Auto_Solving) and (SEL_XY_was_first=False) then
      begin
        SetLength(SEL_X, 2+SEL_Nmax);
        SetLength(SEL_Y, 2+SEL_Nmax);
        ia:=2+SEL_Nmax;
        ij:=0;
      end
      else if (Auto_Solving) and (SEL_XY_was_first) then
        begin
          SetLength(SEL_X, 1+SEL_Nmax);
          SetLength(SEL_Y, 1+SEL_Nmax);
          ia:=1+SEL_Nmax;
          ij:=1;
        end
        else
          begin
            SetLength(SEL_X, 2+SEL_Nmax);
            SetLength(SEL_Y, 2+SEL_Nmax);
            ia:=2+SEL_Nmax;
            ij:=0;
          end;
        end;
  end;
if (Auto_Solving=False) or ((Auto_Solving) and (SEL_XY_was_first=False)) then
  begin
    SEL_X[0]:=0;
    SEL_Y[0]:=0;
  end;
if Normalization then SEL_norm_koef:=Normalization_Value else SEL_norm_koef:=1;
if Dechennaling_with_podsloi then
  begin
    SEL_X[1-ij]:=Trajects.Traject[0].t_arr[0]/epsilon*d_small;

```

```

SEL_Y[1-ij]:=(1-SEL_Nch_na_No[0])/SEL_norm_koef;
bv:=0;
for di:=0 to SEL_Nsloev-1 do
  begin
    SEL_X[2+di-ij]:=Trajects.Traject[0].t_arr[(di+1)*SEL_Podsloi+bv]/epsilon*d_small;
    SEL_Y[2+di-ij]:=(1-SEL_Nch_na_No[di])/SEL_norm_koef;
    bv:=bv+1;
  end;
if (Auto_Solving=False) then
  begin
    SEL_X[2+SEL_Nsloev]:=Trajects.Traject[0].t_arr[SEL_Nsloev*SEL_Podsloi+bv-1]/epsilon*d_small;
    SEL_Y[2+SEL_Nsloev]:=0;
  end;
end
else
  for di:=0 to SEL_Nmax do
    begin
      SEL_X[1+di-ij]:=Trajects.Traject[0].t_arr[di]/epsilon*d_small;
      SEL_Y[1+di-ij]:=(1-SEL_Nch_na_No[di])/SEL_norm_koef;
    end;
    DecimalSeparator:='.';
    AssignFile(SaveF, SEL_FileName);
    Append(SaveF);
    for i:=0 to ia-1 do
      begin
        str:=FloatToStr(SEL_X[i]) + ' ' + FloatToStr(SEL_Y[i]);
        Writeln(SaveF, str);
      end;
    end;
    CloseFile(SaveF);
    DecimalSeparator:='.';
    SEL_XY_was_first:=True;
  end;
end;
except
  MessageDlg('Ошибка при расчете выхода',mtError,[mbOk], 0);
end;
end;

procedure TForm1.LoadFilePropertiesClick(Sender: TObject);
var
  LFileName: string;
begin

```



```

OpenDialog2.Filter := 'Files of programm"s properties (*.ini)|*.ini';
OpenDialog2.InitialDir:=DirProgram;
if OpenDialog2.Execute then
    try
        LFileName:=OpenDialog2.FileName;
        LoadFileInitIni(LFileName);
        ChangeOptions;
        STE_Refreshment;
    except
        MessageDlg('При загрузке возникла ошибка',mtError,[mbOk], 0);
        Exit;
    end;
end;

procedure TForm1.SaveFilePropertiesClick(Sender: TObject);
var
    SFileName: string;
begin
    SaveDialog2.Filter := 'Files of programm"s properties (*.ini)|*.ini';
    SaveDialog2.InitialDir:=DirProgram;
    if SaveDialog2.Execute then
        try
            SFileName:=SaveDialog2.FileName;
            SaveFileInitIni(SFileName);
        except
            MessageDlg('При сохранении возникла ошибка',mtError,[mbOk], 0);
        end;
    end;

procedure TForm1.Panel2DbClick(Sender: TObject);
begin
if VisibleMySQLDir then
    begin
        LMySQL.Visible:= True;
        VisibleMySQLDir:= False;
    end
    else
        begin
            LMySQL.Visible:= False;
            VisibleMySQLDir:= True;
        end;
    end;

```

```

procedure TForm1.LEMySQLChange(Sender: TObject);
begin
    TDirMySql:= LEMySql.Text;
end;

procedure TForm1.DestroyAllData(Sender: TObject);
begin
    ReStartProgram(False, "", Form1.Top, Form1.Left);
end;

procedure TForm1.ShowActiveConnectionToMySQL;
begin
    if Connection.Connected
    then TrAction.Panels[2].Text:='Server MySQL Active'
    else TrAction.Panels[2].Text:="";
end;

procedure TForm1.CreateOurConnection(strDb,strDataDir:string);
var
    tmpStringList: TStringList;
begin
    Connection := TZConnection.Create(Self);
    Connection.Protocol := 'mysqld-4.1';
    Connection.HostName := "";
    Connection.Database := strDb;
    Connection.User := "";
    Connection.Password := "";
    Connection.Port := 0;
    tmpStringList := TStringList.Create;
    try
        with tmpStringList do begin
            Add('compress=yes');
            Add('dbless=no');
            Add('userresult=no');
            Add('timeout=30');
            Add('ServerArgument1=--basedir=./');
            Add('ServerArgument2=--datadir=' + strDataDir);
            Add('ServerArgument3=--character-sets-dir=./share/charsets');
            Add('ServerArgument4=--language=./share/english');
            Add('ServerArgument5=--skip-innodb');
            Add('ServerArgument6=--key_buffer_size=32M');
        end;
        Connection.Properties.Assign(tmpStringList);

```

```

finally
    tmpStringList.free;
end;
ZQueryMS.Connection := Connection;
ZQueryMSDB1.Connection := Connection;
ZQueryMSDB2.Connection := Connection;
ZQueryMSdl0.Connection := Connection;
ZQueryMSdl1.Connection := Connection;
ZQueryMSdl2.Connection := Connection;
ZQueryMSdl3.Connection := Connection;
ZQueryMSothers.Connection := Connection;
end;

procedure TForm1.DestroyOurConnection;
begin
    Connection.Connected := False;
    Connection.Destroy;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
    PlotF.Show;
end;

procedure TForm1.AbFilesShow(DirTab,Tab1,Tab2:string);
begin
    with AbFilesF do
        begin
            Memo1.Clear;
            Memo1.Lines.Add('Общая информация');
            Memo1.Lines.Add("");
            Memo1.Lines.Add('Место расположения файла:');
            Memo1.Lines.Add(' ' + DirTab);
            Memo1.Lines.Add("");
            Memo1.Lines.Add('Загружен файл:');
            Memo1.Lines.Add(' ' + Tab1);
            Memo1.Lines.Add("");
            Memo1.Lines.Add('Файл настроек:');
            Memo1.Lines.Add(' ' + Tab2);
            Memo1.Lines.Add("");
            Memo1.Lines.Add('Параметры расчета');
            Memo1.Lines.Add("");
            Memo1.Lines.Add('Шаг по tau:');

```

```

Memo1.Lines.Add(' ' + FloatToStr(step_on_tau));
Memo1.Lines.Add("");
Memo1.Lines.Add('Количество сохраненных точек на шаг по тау:');
Memo1.Lines.Add(' ' + IntToStr(cnt_for_Saving_if_N_Point_True_on_one_step_tau));
Memo1.Lines.Add("");
Memo1.Lines.Add('Всего точек для 1 траектории:');
Memo1.Lines.Add(' ' + FloatToStr(cnt_for_Saving_if_N_Point_True));
end;
end;

procedure TForm1.AbFilesButtonClick(Sender: TObject);
begin
    AbFilesF.Show;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
    if StartDistributionOnVXSet then
    begin
        TrAction.Panels[1].Text := 'Начальное распределение задано...';
        Timer2.Enabled := True;
        StartDistributionOnVXSet := False;
    end;
end;

procedure TForm1.NSpecLoadClick(Sender: TObject);
begin
    FSpLoad.Show;
end;

end.

```

unit STEaddTrj;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, Constants, Buttons, NormalDistribution,
TeeProcs, TeeDraw3D;

type

TForm2 = **class**(TForm)
 LabeledEdit1: TLabeledEdit;
 StaticText1: TStaticText;
 LabeledEdit2: TLabeledEdit;
 StaticText2: TStaticText;
 LabeledEdit3: TLabeledEdit;
 LabeledEdit4: TLabeledEdit;
 LabeledEdit5: TLabeledEdit;
 BitBtn1: TBitBtn;
 BitBtn2: TBitBtn;
 CheckBox2: TCheckBox;
 LEalpha: TLabeledEdit;
 LEbeta: TLabeledEdit;
 Draw3D1: TDraw3D;
 procedure FormCreate(Sender: TObject);
 procedure BitBtn1Click(Sender: TObject);
 procedure BitBtn2Click(Sender: TObject);
 procedure CheckBox2Click(Sender: TObject);

private

 { Private declarations }

public

 { Public declarations }

end;

PForm2=^TForm2;

TSeriesProperties=record

 x1st,v1st:double;

 x2nd,v2nd:double;

 count:integer;

end;

PSeriesProperties=^TSeriesProperties;

```

TSerialObj=object
  Serial:array of TSerialProperties;
  SerialCnt:integer;
  MaxCnt,AllTrCnt:integer;
  Dlg:PForm2;
  constructor Init(cn:integer;d:PForm2);
  destructor Done;
  Procedure Add1Series(ser:PSerialProperties);
  function Execute:boolean;
end;
PSerialObj=^TSerialObj;

procedure InscriptionV(a:boolean);

var
  Form2: TForm2;
  SerialProperties:TSerialProperties;
  AllSerial:TSerialObj;
  VGauss:boolean;

implementation

{$R *.dfm}

function TSerialObj.Execute;
begin
  Execute:=true;
end;

procedure TSerialObj.Add1Series(ser:PserialProperties);
begin
  if (SerialCnt < MaxCnt) then
    begin
      Serial[SerialCnt].count:=ser^.count;
      Serial[SerialCnt].x1st:=ser^.x1st;
      Serial[SerialCnt].v1st:=ser^.v1st;
      Serial[SerialCnt].x2nd:=ser^.x2nd;
      Serial[SerialCnt].v2nd:=ser^.v2nd;
      AllTrCnt:=AllTrCnt+ser^.count;
      inc(SerialCnt);
    end;
  end;

```

```

constructor TSerialObj.Init(cn:integer;d:PForm2);
begin
    SerialCnt:=0;
    AllTrCnt:=0;
    Dlg:=d;
    MaxCnt:=cn;
    SetLength(Serial,cn);
end;

destructor TSerialObj.Done;
begin
    Finalize(Serial);
end;

procedure TForm2.BitBtn1Click(Sender: TObject);
var
    SetTrEd: TextFile;
    s: string;
    i, c1, c2, c3, c4, c5, c6, c7: integer;
    alpha_ND, beta_ND, sigma_ND, tmp_alpha, tmp_beta: double;
begin
    with SeriesProperties do
        begin
            s:=Form2.LabeledEdit1.Text;
            val(s,x1st,c1);
            s:=Form2.LabeledEdit2.Text;
            val(s,v1st,c2);
            s:=Form2.LabeledEdit3.Text;
            val(s,x2nd,c3);
            s:=Form2.LabeledEdit4.Text;
            val(s,v2nd,c4);
            s:=Form2.LabeledEdit5.Text;
            val(s,count,c5);
            ntr:=count;
            if VGauss=True then
                begin
                    a_ND:=v1st;
                    n_ND:=count-1;
                    sigma_ND:=v2nd;
                    s:=Form2.LEalpha.Text;
                    val(s,alpha_ND,c6);
                    s:=Form2.LEbeta.Text;
                    val(s,beta_ND,c7);

```

```

    tmp_alpha := alpha_ND*2;
    if tmp_alpha > 0 then tmp_alpha := tmp_alpha * (-1);
    tmp_beta := beta_ND*2;
end
else
    begin
        c6:=0;
        c7:=0;
    end;
end;
if (c1=0) and (c2=0)and (c3=0)and (c4=0)and (c5=0) and (c6=0) and (c7=0) then
begin
    if VGauss=True then
        begin
            NormalDistribution.ArrayNormalDistribution(sigma_ND, alpha_ND, beta_ND);
            for i:=0 to n_ND do
                if (v_ND[i] < tmp_alpha) or (v_ND[i] > tmp_beta) then
                    begin
                        MessageDlg('Проведен анализ полученных значений для нормального распределения.
Возможна ошибка. Следует ввести другие значения для нормального распределения.',
mtInformation,[mbOk], 0);
                        Exit;
                    end;
                end;
            if not LoadWithParameters then StartDistributionOnVXSet := True;
            AllSeria.Add1Series(@SeriesProperties);
            Form2.Visible:=false;
        end;
    end;

procedure TForm2.BitBtn2Click(Sender: TObject);
begin
    Form2.Visible:=false;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
    AllSeria.Init(10,@Form2);
end;

procedure TForm2.CheckBox2Click(Sender: TObject);
begin
    with Sender as TCheckBox do

```



```

begin
  Checked:=Checked;
  VGauss:=Checked;
  InscriptionV(Checked);
  Form2.LEalpha.Visible:=Checked;
  Form2.LEbeta.Visible:=Checked;
end;
end;

procedure InscriptionV(a:boolean);
begin
if a=True then
  begin
    Form2.LabeledEdit2.EditLabel.Caption:='v среднее';
    Form2.LabeledEdit4.EditLabel.Caption:='Ср.кв.отклонение';
  end;
if a=False then
  begin
    Form2.LabeledEdit2.EditLabel.Caption:='v=';
    Form2.LabeledEdit4.EditLabel.Caption:='v=';
  end;
end;

end.

```

unit STEoptions;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, Constants, Buttons, ComCtrls, OleCtrls,
TeeProcs, TeeDraw3D;

type

TFormOptions = **class**(TForm)
 GroupBox1: TGroupBox;
 LabeledEdit1: TLabeledEdit;
 LabeledEdit2: TLabeledEdit;
 Label1: TLabel;
 Label2: TLabel;
 LabeledEdit3: TLabeledEdit;
 Label3: TLabel;
 CheckBox1: TCheckBox;
 CheckBox2: TCheckBox;
 CheckBox3: TCheckBox;
 CheckBox4: TCheckBox;
 LabeledEdit4: TLabeledEdit;
 Label4: TLabel;
 GroupBox2: TGroupBox;
 CheckBox5: TCheckBox;
 CheckBox6: TCheckBox;
 LabeledEdit5: TLabeledEdit;
 CheckBox7: TCheckBox;
 CheckBox8: TCheckBox;
 Edit13: TEdit;
 Label11: TLabel;
 Label12: TLabel;
 RadioGroup1: TRadioGroup;
 RadioButton1: TRadioButton;
 RadioButton2: TRadioButton;
 RadioButton3: TRadioButton;
 RadioButton4: TRadioButton;
 LEDBName: TLabeledEdit;
 CheckBox9: TCheckBox;
 CheckBox11: TCheckBox;
 GroupBox3: TGroupBox;
 CheckBox10: TCheckBox;

```

LabeledEdit6: TLabeledEdit;
CheckBox12: TCheckBox;
CheckBox13: TCheckBox;
Edit14: TEdit;
CheckBox14: TCheckBox;
Edit15: TEdit;
CheckBox15: TCheckBox;
GroupBox4: TGroupBox;
LnucRad: TImage;
Label5: TLabel;
Label6: TLabel;
LnucBor: TImage;
E2var: TImage;
E4var: TImage;
GroupBox5: TGroupBox;
Label7: TLabel;
Draw3D1: TDraw3D;
ButValueLog: TButton;
Label8: TLabel;
ButSetDirAutoS: TButton;
SaveDialog1: TSaveDialog;
CheckBox17: TCheckBox;
CheckBox18: TCheckBox;
Button1: TButton;
BitBtn1: TBitBtn;
CheckBox16: TCheckBox;
Button2: TButton;
procedure FormCreate(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure CheckBox2Click(Sender: TObject);
procedure CheckBox3Click(Sender: TObject);
procedure CheckBox4Click(Sender: TObject);
procedure CheckBox5Click(Sender: TObject);
procedure SolveTrBtnClick(Sender: TObject);
procedure CheckBox8Click(Sender: TObject);
procedure RadioButton123Click(Sender: TObject);
procedure CheckBox11Click(Sender: TObject);
procedure CheckBox9Click(Sender: TObject);
procedure CheckBox10Click(Sender: TObject);
procedure CheckBox7Click(Sender: TObject);
procedure CheckBox6Click(Sender: TObject);
procedure CheckBox12Click(Sender: TObject);
procedure CheckBox13Click(Sender: TObject);

```

```

procedure CheckBox14Click(Sender: TObject);
procedure CheckBox15Click(Sender: TObject);
procedure DescribeSave;
procedure FormShow(Sender: TObject);
procedure LEDBNameChange(Sender: TObject);
procedure ButValueLogClick(Sender: TObject);
procedure ButSetDirAutoSClick(Sender: TObject);
function SelectDirPlus(hWnd: HWND; const Caption: string;
const Root: WideString): String;
procedure CheckBox17Click(Sender: TObject);
procedure CheckBox18Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure CheckBox16Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

```

var

```

FormOptions: TFormOptions;
myInitialDir: string;

```

implementation

uses STetraject, ThreePointBending, ValueLog, ActiveX, ShlObj,
AdvOptToOurPlot, ModelDechan, Math;

{ \$R *.dfm }

```

procedure TFormOptions.FormCreate(Sender: TObject);
begin
    FormOptions.Left:=90;
    FormOptions.Top:=60;
end;

```

```

procedure TFormOptions.CheckBox1Click(Sender: TObject);
begin
    if FormOptions.CheckBox1.Checked=True then OverBarrierOff:=1
    else OverBarrierOff:=0;

```

end;

procedure TFormOptions.CheckBox2Click(Sender: TObject);

begin

with Sender as TCheckBox **do**

begin

Checked:=Checked;

if Checked=True **then** BorOff:=1 **else** BorOff:=0;

if Checked=True **then**

begin

LnucRad.Visible:=True;

LnucBor.Visible:=False;

end

else

begin

LnucRad.Visible:=False;

LnucBor.Visible:=True;

end;

end;

end;

procedure TFormOptions.CheckBox3Click(Sender: TObject);

begin

if FormOptions.CheckBox3.Checked=True **then** SrKvFlEp:=1

else SrKvFlEp:=0;

end;

procedure TFormOptions.CheckBox4Click(Sender: TObject);

begin

if FormOptions.CheckBox4.Checked=True **then** VMoilerOff:=1

else VMoilerOff:=0;

end;

procedure TFormOptions.CheckBox5Click(Sender: TObject);

begin

with Sender as TCheckBox **do**

begin

Checked:=Checked;

if Checked=True **then** EnergyLossVariant4Off:=true

else EnergyLossVariant4Off:=false;

if Checked=True **then**

begin

E4var.Visible:=False;

```

        E2var.Visible:=True;
    end
    else
        begin
            E4var.Visible:=True;
            E2var.Visible:=False;
        end;
    end;

end;

procedure TFormOptions.SolveTrBtnClick(Sender: TObject);
begin
    FormOptions.Close;
    Form1.SolveTrBtnClick;
end;

procedure TFormOptions.CheckBox8Click(Sender: TObject);
begin
with Sender as TCheckBox do
    begin
        Checked:=Checked;
        CheckBox6.Checked:=Checked;
        Three_point_bending:=Checked;
    end;
end;

procedure TFormOptions.RadioButton123Click(Sender: TObject);
begin
if RadioButton1.Checked then TrDataFormat:=1;
if RadioButton2.Checked then TrDataFormat:=2;
if RadioButton3.Checked then TrDataFormat:=3;
if RadioButton4.Checked then TrDataFormat:=4;
end;

procedure TFormOptions.CheckBox11Click(Sender: TObject);
begin
with Sender as TCheckBox do
    begin
        Checked := Checked;
        CheckBox9.Enabled := Checked;
        if Checked=False then CheckBox9.Checked := False;
        N_Point_Save := Checked;
    
```

```
    end;  
    DescribeSave;  
end;
```

```
procedure TFormOptions.CheckBox9Click(Sender: TObject);  
begin  
    with Sender as TCheckBox do  
        begin  
            Checked:=Checked;  
            CheckBox10.Enabled:=Checked;  
            if Checked=False then CheckBox10.Checked:= False;  
            Last_Layer_Save:=Checked;  
        end;  
        DescribeSave;  
    end;  
end;
```

```
procedure TFormOptions.CheckBox10Click(Sender: TObject);  
begin  
    with Sender as TCheckBox do  
        begin  
            Last_Layer_ReSave:=Checked;  
        end;  
    end;  
end;
```

```
procedure TFormOptions.CheckBox7Click(Sender: TObject);  
begin  
    with Sender as TCheckBox do  
        begin  
            Checked:=Checked;  
            CheckBox11.Enabled:=Checked;  
            LabeledEdit6.Enabled:=Checked;  
            if Checked=False then CheckBox11.Checked:= False;  
            Auto_Save:=Checked;  
        end;  
        DescribeSave;  
    end;  
end;
```

```
procedure TFormOptions.CheckBox6Click(Sender: TObject);  
begin  
    with Sender as TCheckBox do  
        begin
```

```

    Auto_Solving:=Checked;
    if (not Checked) and (CheckBox8.Checked) then
        CheckBox8.Checked := Checked;
    end;
    DescribeSave;
end;

procedure TFormOptions.CheckBox12Click(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            Auto_Solving_Exit:=Checked;
        end;
        DescribeSave;
end;

procedure TFormOptions.CheckBox13Click(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            sublayer_on_formoptions:=Checked;
        end;
end;

procedure TFormOptions.CheckBox14Click(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            normalization_on_formoptions:=Checked;
        end;
end;

procedure TFormOptions.CheckBox15Click(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            Checked:=Checked;
            if Checked then
                begin
                    BendingForm.Show;
                end;
            if not Checked then
                begin

```



```

        BendingForm.Close;
    end;
end;
end;

procedure TFormOptions.DescribeSave;
var
s, s2, s3, tmpServer, tmpFullServer: string;
tmpa, i: integer;
begin
s3:=LEDBName.Text;
tmpFullServer := ParamStr(0);
tmpServer := ExtractFilePath(tmpFullServer);
tmpServer := tmpServer + 'data\...';
Label8.Caption := 'Папка для автосохранения, по умолчанию: ';
Label8.Caption := Label8.Caption + tmpServer;
s2:='папку с данными ' + '...\ ' + s3 + '\, файл(ы) данных ';
s:='Не выбран режим автосохранения траекторий';
if Auto_Save then
    begin
        s:='Сохранение, в ' + s2 + s3+ 'db (все траектории)';
    end;
if (Auto_Save) and (Auto_Solving) then
    begin
        s:='Сохранение, в ' + s2 + s3+ '_layer1_db (по слоям)';
    end;
if ((Auto_Save) and (Auto_Solving) and (N_Point_Save)) or ((Auto_Save) and (N_Point_Save))
then
    begin
        s:='Сохранение, в ' + s2 + s3+ '_particles1_db (по траекториям), ' + s3+ 'db (все траектории)'
        + ', сохраняется каждая ' + LabeledEdit6.Text + ' точка';
        if Last_Layer_Save then s:=s + ', ' + s3+ '_lastlayer_db (последний слой, сохраняются все
        точки)';
    end;
if Auto_Solving_Exit then
    begin
        s:=s + ', расчет и сохранение Выхода частиц с порогом ' + Edit13.Text + 'эВ/А в файл
        Exit.txt';
    end;
FormOptions.Label7.Caption:=s;
FormOptions.Label7.Width:=785;
FormOptions.Label7.Height:=48;
end;

```

```

procedure TFormOptions.FormShow(Sender: TObject);
begin
    DescribeSave;
end;

procedure TFormOptions.LEDBNameChange(Sender: TObject);
begin
    if Auto_Save then DescribeSave;
end;

procedure TFormOptions.ButValueLogClick(Sender: TObject);
begin
    FValueLog.Show;
end;

function BrowseCallbackProc(hwnd: HWND; uMsg: UINT; lParam: LPARAM; lpData:
LPARAM): integer; stdcall;
begin
    Result := 0;
    if uMsg = BFFM_INITIALIZED then
        SendMessage( hwnd, BFFM_SETSELECTION, 1,
            LongInt( PChar( myInitialDir ) ) );
end;

function TFormOptions.SelectDirPlus(hWnd: HWND; const Caption: string;
const Root: WideString): String;
var
    WindowList: Pointer;
    BrowseInfo : TBrowseInfo;
    Buffer: PChar;
    RootItemIDList, ItemIDList: PItemIDList;
    ShellMalloc: IMalloc;
    IDesktopFolder: IShellFolder;
    Eaten, Flags: LongWord;
    Cmd: Boolean;
begin
    FillChar( BrowseInfo, SizeOf( BrowseInfo ), 0 );
    if ( ShGetMalloc( ShellMalloc ) = S_OK ) and ( ShellMalloc <> nil ) then
        begin
            Buffer := ShellMalloc.Alloc( MAX_PATH );
            try
                RootItemIDList := nil;

```

```

if Root <> " then
begin
    SHGetDesktopFolder( IDesktopFolder );
    IDesktopFolder.ParseDisplayName( hWnd, nil,
    POleStr( Root ), Eaten, RootItemIDList, Flags );
end;
with BrowseInfo do
begin
    hwndOwner := hWnd;
    pidlRoot := RootItemIDList;
    pszDisplayName := Buffer;
    lpfn := @BrowseCallbackProc;
    lpszTitle := PChar( Caption );
    ulFlags := BIF_RETURNONLYFSDIRS or $0040 or BIF_EDITBOX or
BIF_STATUSTEXT;
end;
    WindowList := DisableTaskWindows( 0 );
try
        ItemIDList := ShBrowseForFolder( BrowseInfo );
    finally
        EnableTaskWindows( WindowList );
    end;
    Cmd := ItemIDList <> nil;
if Cmd then
begin
        ShGetPathFromIDList( ItemIDList, Buffer );
        ShellMalloc.Free( ItemIDList );
        Result:= Buffer;
    end;
    finally
        ShellMalloc.Free( Buffer );
    end;
end;
end;

```

```

procedure TFormOptions.ButSetDirAutoSClick(Sender: TObject);
var
    i, tmpd: integer;
    MyDir: string;
begin
    if OutPathForDataBack = './data' then
        myInitialDir := ExtractFilePath(Application.ExeName)

```

```

    else myInitialDir := OutPathForData;
MyDir := SelectDirPlus( Handle, 'Выберите каталог', " ");
if MyDir <> " then
begin
    OutPathForData := MyDir;
    tmpd:=Length(OutPathForData);
    if OutPathForData[tmpd] = '\' then Delete(OutPathForData,tmpd,1);
    Label8.Caption := 'Папка для автосохранения: ';
    Label8.Caption := Label8.Caption + OutPathForData + '\...!';
    OutPathForDataBack := OutPathForData;
    tmpd:=Length(OutPathForDataBack);
    tmpd:=tmpd-1;
    for i:=0 to tmpd do
        if OutPathForDataBack[tmpd-i]='\' then
            OutPathForDataBack[tmpd-i]:='/';
    end;
end;

procedure TFormOptions.CheckBox17Click(Sender: TObject);
begin
    with Sender as TCheckBox do
    begin
        Checked := Checked;
        Potential_Axil_Plane := Checked;
    end;
end;

procedure TFormOptions.CheckBox18Click(Sender: TObject);
begin
    with Sender as TCheckBox do
    begin
        Checked := Checked;
        if Checked then gameev := 1 else gameev := 0;
        Form1.ChangeOptions;
    end;
end;

procedure TFormOptions.Button1Click(Sender: TObject);
begin
    ModelDechF.Show;
end;

```

```

procedure TFormOptions.Button2Click(Sender: TObject);
var
  code, code2: integer;
  r, r2: double;
  tmpStrA: string;
begin
  Val(Form1.LabeledEdit1.Text, r, code);
  Val(Form1.LabeledEdit2.Text, r2, code2);
  if (code=0) and (code2=0) then
    begin
      tmpStrA := FloatToStr(RoundTo(r/epsilon*d_small/r2,-2))+ ' A.';
      MessageDlg('Одному узлу в Ангстремах соответствует значение ' + tmpStrA,
mtInformation,[mbOk], 0);
    end
    else MessageDlg('Не удастся вычислить значение одного узла в Ангстремах. Вероятно на-
чальные параметры на главной странице заданы некорректно.' , mtInformation,[mbOk], 0);
  end;

procedure TFormOptions.CheckBox16Click(Sender: TObject);
begin
  with Sender as TCheckBox do
    begin
      Checked := Checked;
      if Checked then ScatteringWithY := True else ScatteringWithY := False;
    end;
  end;

end.

```

unit OurPlotting;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Series, TeEngine, Chart, StdCtrls, ComCtrls, ExtCtrls, TeCanvas,
Buttons, TeeProcs, TeeDraw3D, Menus, ImgList, TeeFunci, TeeSurfa,
TeeComma, TeePoin3, TeeOpenGL, TeeEdit, Gauges, Math, TeeTools, Spin;

type

TPlotF = **class**(TForm)
 Panel2: TPanel;
 Panel3: TPanel;
 Draw3D1: TDraw3D;
 BPrevious: TSpeedButton;
 BNext: TSpeedButton;
 PageControl1: TPageControl;
 TabSheet1: TTabSheet;
 TreeView1: TTreeView;
 BaseSplitter1: TSplitter;
 Panel4: TPanel;
 Label4: TLabel;
 Label5: TLabel;
 Edit1: TEdit;
 UpDown1: TUpDown;
 Edit2: TEdit;
 UpDown2: TUpDown;
 Button3: TButton;
 Memo1: TMemo;
 Splitter1: TSplitter;
 PageExample: TPageControl;
 TabExample: TTabSheet;
 TabSource: TTabSheet;
 Memo3: TMemo;
 Images: TImageList;
 PopupMenu1: TPopupMenu;
 ShowAll1: TMenuItem;
 ImageList1: TImageList;
 ImageEL1: TImage;
 TeeCommander1: TTeeCommander;
 GLBox: TCheckBox;
 Label1: TLabel;

ChooseLang: TComboBox;
AdvOptBtn: TButton;
Gauge1: TGauge;
ImageEL2: TImage;
ImageEL3: TImage;
ImageEL4: TImage;
ImageEL5: TImage;
ImageEL6: TImage;
Button1: TButton;
LabWait: TLabel;
LimitShow: TButton;
TabSheet2: TTabSheet;
GroupBox1: TGroupBox;
XValuesMin: TLabeledEdit;
XValuesMax: TLabeledEdit;
GroupBox2: TGroupBox;
ZValuesMin: TLabeledEdit;
ZValuesMax: TLabeledEdit;
GroupBox4: TGroupBox;
Ymin: TLabeledEdit;
Ymax: TLabeledEdit;
CheckBox9: TCheckBox;
CheckBox10: TCheckBox;
IncrementOY: TCheckBox;
IncrementOX: TCheckBox;
CheckBox11: TCheckBox;
CheckBox1: TCheckBox;
CheckBox2: TCheckBox;
Edit12: TEdit;
Edit13: TEdit;
SpinEdit1: TSpinEdit;
GroupBox3: TGroupBox;
GroupBox6: TGroupBox;
GroupBox5: TGroupBox;
Xmin: TLabeledEdit;
Xmax: TLabeledEdit;
CheckBox12: TCheckBox;
GroupBox7: TGroupBox;
ChartEditor1: TChartEditor;
ChEffect: TCheckBox;
EdEffectFrom: TEdit;
EdEffectTo: TEdit;
Label2: TLabel;

```

Button2: TButton;
TabSheet3: TTabSheet;
LabMaxOX: TLabel;
LabMinOX: TLabel;
LabMaxOY: TLabel;
LabMinOY: TLabel;
LabAllPoint: TLabel;
Label3: TLabel;
ButThClassic: TButton;
ButThXP: TButton;
Chart1: TChart;
TeeOpenGL: TTeeOpenGL;
CheckXtSm: TCheckBox;
Series1: TLineSeries;
ButChangeChart: TButton;
procedure PopupMenu1Popup(Sender: TObject);
procedure ShowAll1Click(Sender: TObject);
procedure BNextClick(Sender: TObject);
procedure BPreviousClick(Sender: TObject);
function ImageIndex(Node: TTreeNode): Integer;
procedure TreeView1Change(Sender: TObject; Node: TTreeNode);
procedure TreeView1GetImageIndex(Sender: TObject; Node: TTreeNode);
procedure TreeView1GetSelectedIndex(Sender: TObject; Node: TTreeNode);
procedure PlotGraphics(Sender: TObject);
procedure VizibleChartWithMaxSpeed(MaxP: integer);
procedure tmpTeeFunctionCalculate(Sender: TCustomTeeFunction;
const x: Double; var y: Double);
procedure tmpTeeFunctionCalculateUx(Sender: TCustomTeeFunction;
const x: Double; var y: Double);
procedure tmpTeeFunctionCalculateUxx(Sender: TCustomTeeFunction;
const x: Double; var y: Double);
procedure tmpTeeFunctionCalculate2(Sender: TCustomTeeFunction;
const x: Double; var y: Double);
procedure ChooseLangChange(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure AdvOptBtnClick(Sender: TObject);
procedure CalculateEmax(Nm, Nmax: integer);
procedure PlotDescript;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure LimitFShow(Sender: TObject);
procedure SetLimit(LimV, LimX, LimColor, LimViz, LimHistog, LimEndPoint: boolean);
procedure OffLimit;
procedure tmpTeeFunctionCalculateRel(Sender: TCustomTeeFunction;

```



```

const x: Double; var y: Double);
  procedure tmpTeeFunctionCalculateRnucl(Sender: TCustomTeeFunction;
const x: Double; var y: Double);
  procedure SpinEdit1Change(Sender: TObject);
  procedure CheckBox10Click(Sender: TObject);
  procedure IncrementOYClick(Sender: TObject);
  procedure IncrementOXClick(Sender: TObject);
  procedure CheckBox1Click(Sender: TObject);
  procedure RadioButton1Click(Sender: TObject);
  procedure RadioButton2Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure ChartStat;
  procedure TabSheet3Show(Sender: TObject);
  procedure HistogramVX(NumHistog,tmpNm: integer; MainHistog: boolean;
tmpRazresh: double; ArrLimNm: array of byte; HistogColor: TColor);
  procedure ButThClassicClick(Sender: TObject);
  procedure ButThXPClick(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure ButChangeChartClick(Sender: TObject);
private
  { Private declarations }
  Emax:array of double;
  LeftChartValue, RightChartValue, LeftValueMainHistogram: double;
public
  { Public declarations }
  class procedure ChangeLanguage(ALanguage:Integer);
end;

```

var

PlotF: TPlotF;

implementation

uses STetraject, Constants, TeeConst, TeeProCo, TeeRussian, AdvOptToOurPlot,
Limit, TeeEditCha, STEoptions, ModelDechan, TeePDFCanvas, TeeJPEG, TeeThemes;

{ \$R *.dfm }

var

LimitOnSetNoFirst,

PlotHistogramMain, PlotHistogramSecond: boolean;

procedure TPlotF.PlotDescript;

begin

```
Memo1.Lines.Clear;  
ImageEL1.Visible:=False;  
ImageEL2.Visible:=False;  
ImageEL3.Visible:=False;  
ImageEL4.Visible:=False;  
ImageEL5.Visible:=False;  
ImageEL6.Visible:=False;  
AdvOptF.Button2.Enabled:=False;  
AdvOptF.Button3.Enabled:=False;  
Case TreeView1.Selected.StateIndex of  
1: begin  
    Memo1.Lines.Add('Непрерывный потенциал кристалла.');
```

end;

```
2: begin  
    Memo1.Lines.Add('3-х мерный потенциал кристалла.');
```

end;

```
3: begin  
    Memo1.Lines.Add('Коэффициент диффузии.');
```

end;

```
4: begin  
    end;
```

end;

```
5: begin  
    Memo1.Lines.Add('Траектории частиц  $x(t)$ .');
```

end;

```
6: begin  
    Memo1.Lines.Add('Траектории частиц  $v(t)$ .');
```

end;

```
7: begin  
    Memo1.Lines.Add('Траектории частиц  $v(x)$ .');
```

end;

```
8: begin  
    end;
```

end;

```
9: begin  
    Memo1.Lines.Add('Распределение в канале на заданной глубине,');
```

Memo1.Lines.Add('по поперечной энергии.');

```
    AdvOptF.Button2.Enabled:=True;  
    AdvOptF.Button3.Enabled:=True;  
    end;
```

```
10: begin  
    end;
```

end;

```
11: begin  
    Memo1.Lines.Add('Потери энергии от глубины.');
```

```

    ImageEL1.Visible:=True;
end;
12: begin
    Memo1.Lines.Add('Число частиц от полной энергии E. ');
    ImageEL2.Visible:=True;
end;
13: begin
    Memo1.Lines.Add('Число частиц N от потерянной энергии. ');
    Memo1.Lines.Add('Число частиц N нормировано на общее число частиц No. ');
    ImageEL3.Visible:=True;
    AdvOptF.Button2.Enabled:=True;
    AdvOptF.Button3.Enabled:=True;
end;
14: begin
    Memo1.Lines.Add('Число частиц N от потерянной энергии за пройденный слой. ');
    Memo1.Lines.Add('Число частиц N нормировано на общее число частиц No. ');
    ImageEL4.Visible:=True;
    AdvOptF.Button2.Enabled:=True;
    AdvOptF.Button3.Enabled:=True;
end;
15: begin
    Memo1.Lines.Add('Потери энергии за пройденный слой от пройденного слоя. ');
    ImageEL5.Visible:=True;
end;
16: begin
    Memo1.Lines.Add('Выход частиц. Доля деканализированных частиц по глубине. ');
    ImageEL6.Visible:=True;
end;
17: begin
    Memo1.Lines.Add('Электронная плотность. ');
end;
18: begin
    Memo1.Lines.Add('Ядерная плотность. ');
end;
19: begin
    Memo1.Lines.Add('Гистограмма распределения частиц по v на заданной глубине. ');
    AdvOptF.Button2.Enabled:=True;
    AdvOptF.Button3.Enabled:=True;
end;
20: begin
    Memo1.Lines.Add('Гистограмма распределения частиц по x на заданной глубине. ');
    AdvOptF.Button2.Enabled:=True;
    AdvOptF.Button3.Enabled:=True;

```

```

    end;
21: begin
    Memo1.Lines.Add('Производная от непрерывного потенциала.');
```

end;

```

22: begin
    Memo1.Lines.Add('Вторая производная от непрерывного потенциала.');
```

end;

```

23: begin
    Memo1.Lines.Add('Средний квадрат флуктуаций поперечной энергии.');
```

end;

end

end;

```

procedure TPlotF.PopupMenu1Popup(Sender: TObject);
begin
    ShowAll1.Caption:='Показать все ('+IntToStr(TreeView1.Items.Count)+' )';
end;
```

```

procedure TPlotF.ShowAll1Click(Sender: TObject);
var t : Integer;
begin
    With TreeView1 do
        for t:=0 to Items.Count-1 do
            begin
                Selected:=Items[t];
                Application.ProcessMessages;
            end;
```

end;

```

procedure TPlotF.BNextClick(Sender: TObject);
begin
    with TreeView1 do Items[Selected.AbsoluteIndex+1].Selected := true
end;
```

```

procedure TPlotF.BPreviousClick(Sender: TObject);
begin
    with TreeView1 do Items[Selected.AbsoluteIndex-1].Selected := true
end;
```

```

procedure TPlotF.TreeView1Change(Sender: TObject; Node: TTreeNode);
begin
    with Sender as TTreeView do
        begin
```

```

    BPrevious.Enabled := Assigned(Selected) and (Selected.AbsoluteIndex>0);
    BNext.Enabled := Assigned(Selected) and (Selected.AbsoluteIndex<Items.Count-1);
end;
    PlotDescript;
end;

```

```

function TPlotF.ImageIndex(Node:TTreeNode):Integer;
begin
    if Node.HasChildren then
        if Node.Expanded then result:= 1
            else result:= 0
        else
            if Node.TreeView=TreeView1 then
                result:= 2
            else
                result:= 3;
        end;
end;

```

```

procedure TPlotF.TreeView1GetImageIndex(Sender: TObject; Node: TTreeNode);
begin
    Node.ImageIndex:=ImageIndex(Node);
end;

```

```

procedure TPlotF.TreeView1GetSelectedIndex(Sender: TObject;
    Node: TTreeNode);
begin
    Node.SelectedIndex:=ImageIndex(Node);
end;

```

```

procedure TPlotF.VizibleChartWithMaxSpeed(MaxP:integer);
var
    MaxPoints:integer;
begin;
    with Chart1 do
        begin
            Title.Visible := False;
            Legend.Visible := False;
            LeftAxis.Logarithmic:=False;
            View3D := False;
            Title.Font.Name:='MS Sans Serif';
            Title.Font.Size:=8;
            Title.Font.Style:=[fsBold];
            LeftAxis.Title.Font.Name:='MS Sans Serif';

```

```

LeftAxis.Title.Font.Size:=10;
LeftAxis.Title.Font.Style := [];
LeftAxis.Increment := 0;
BottomAxis.Title.Font.Name:='MS Sans Serif';
BottomAxis.Title.Font.Size:=10;
BottomAxisAutomatic := True;
BottomAxis.Title.Font.Style := [];
BottomAxis.Increment := 0;
DepthAxis.Visible:=False;
DepthAxis.Title.Font.Name:='MS Sans Serif';
DepthAxis.Title.Font.Size:=10;
end;
MaxPoints:=MaxP;
Chart1.AutoRepaint := False;
Chart1.Axes.FastCalc:=True;
end;

procedure TPlotF.tmpTeeFunctionCalculate(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
begin
  y:=U(x);
end;

procedure TPlotF.tmpTeeFunctionCalculateUx(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
begin
  y:=U_x(x);
end;

procedure TPlotF.tmpTeeFunctionCalculateUxx(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
begin
  y:=U_xx(x);
end;

procedure TPlotF.tmpTeeFunctionCalculate2(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
var
  koef_OY: double;
begin
  koef_OY:= Sqr(Sqr(epsilon))*(1E+16)/Sqr(Vmax);
  y:=koef_OY*Diffuzia(x);
end;

```

```

procedure TPlotF.tmpTeeFunctionCalculateRel(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
begin
  y:=rho_el(x);
end;

procedure TPlotF.tmpTeeFunctionCalculateRnucl(Sender: TCustomTeeFunction;
  const x: Double; var y: Double);
begin
  y:=rho_nucl(x);
end;

procedure TPlotF.CalculateEmax(Nm,Nmax: integer);
var
  n:integer;
begin
  SetLength(Emax,Nm);
for n:=0 to Nm-1 do
    Emax[n]:=E-deltaE[n,Nmax];
end;

procedure TPlotF.HistogramVX(NumHistog,tmpNm: integer; MainHistog: boolean; tmpRazresh:
  double;
  ArrLimNm: array of byte; HistogColor: TColor);
var
  tmpLineSeries:TFastLineSeries;
  tmpVmin, tmpVmax, tmpV, razresh, tmpa, tmpb,
  tmpXmin, tmpXmax, tmpX, tmpLeftEf, tmpRightEf: double;
  tmpN_V: array of integer;
  tmpNInterval, tmpInterval, tmpEffect, i, Nm: integer;
  tmpHit: boolean;
begin
  case NumHistog of
  1:
    begin
      Chart1.LeftAxis.Title.Caption:='N';
      Chart1.BottomAxis.Title.Caption:='Поперечная скорость v';
      Nm := tmpNm;
      razresh := tmpRazresh;
      tmpVmin := Trajects.Traject[0].v_arr[glub];
      tmpVmax := Trajects.Traject[0].v_arr[glub];
      for i:=1 to Nm-1 do

```

```

begin
  if ArrLimNm[i] = 0 then
    begin
      tmpV := Trajects.Traject[i].v_arr[glub];
      if tmpV < tmpVmin then tmpVmin := tmpV
      else if tmpV > tmpVmax then tmpVmax := tmpV;
    end;
  end;
if MainHistog then
  LeftValueMainHistogram := tmpVmin
  else
    tmpVmin := tmpVmin -
      razresh*Frac( (tmpVmin - LeftValueMainHistogram)/razresh );
  tmpNInterval := Abs(Round(Int((tmpVmax - tmpVmin)/razresh))) + 1;
  SetLength(tmpN_V, tmpNInterval);
for i:=0 to Nm-1 do
  begin
    if ArrLimNm[i] = 0 then
      begin
        tmpa := tmpVmin;
        tmpb := tmpa + razresh;
        tmpV := Trajects.Traject[i].v_arr[glub];
        tmpInterval := 0;
        tmpHit := False;
        while not tmpHit do
          begin
            if (tmpV >= tmpa) and (tmpV < tmpb) then
              begin
                tmpN_V[tmpInterval] := tmpN_V[tmpInterval] + 1;
                tmpHit := True;
              end
            else
              begin
                tmpa := tmpa + razresh;
                tmpb := tmpb + razresh;
                inc(tmpInterval);
              end;
            end;
          end;
        end;
      end;
    end;
  end;
  tmpLineSeries := TFastLineSeries.Create(self);
  tmpa := tmpVmin;
  tmpLineSeries.AddXY(tmpa, 0, " ", clBlack);

```



```

for i:=0 to tmpNInterval - 1 do
  begin
    tmpLineSeries.AddXY(tmpa, tmpN_V[i], ", clBlack);
    tmpa := tmpa + razresh;
  end;
tmpLineSeries.AddXY(tmpa, tmpN_V[tmpNInterval - 1], ", clBlack);
tmpLineSeries.AddXY(tmpa, 0, ", clBlack);
Finalize(tmpN_V);
if ChEffect.Checked then
  begin
    tmpEffect:=0;
    tmpLeftEf := StrToFloat(PlotF.EdEffectFrom.Text);
    tmpRightEf := StrToFloat(PlotF.EdEffectTo.Text);
    for i:=0 to Nm - 1 do
      if ((Trajects.Traject[i].v_arr[glub]) > tmpLeftEf) and ((Trajects.Traject[i].v_arr[glub]) <
tmpRightEf) then
        tmpEffect := tmpEffect + 1;
        Chart1.Title.Text.Text:='На глубине ' +
        FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+ ' A'+
        ' (в ' +IntToStr(glub+1)+' точке), эффективность прохождения ' + Float-
        ToStr(RoundTo(tmpEffect/Nm*100, -2)) + '%';
      end
    else
      Chart1.Title.Text.Text:='На глубине ' +
      FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+ ' A'+
      ' (в ' +IntToStr(glub+1)+' точке)';
      tmpb := (tmpa - tmpVmin) * 0.15;
      Chart1.BottomAxis.SetMinMax(tmpVmin - tmpb, tmpa + tmpb);
      tmpLineSeries.Stairs := True;
      Chart1.AddSeries(tmpLineSeries);
      tmpLineSeries.Color := HistogColor;
    if MainHistog then
      begin
        LeftChartValue := Chart1.BottomAxis.Minimum;
        RightChartValue := Chart1.BottomAxis.Maximum;
      end
    else
      begin
        Chart1.BottomAxis.Minimum := LeftChartValue;
        Chart1.BottomAxis.Maximum := RightChartValue;
      end;
    end;
  end;
end;

```

end;

procedure TPlotF.ChartStat;

begin

if Chart1.SeriesCount > 0 **then**

begin

LabMaxOX.Caption := 'Max значение по оси OX ' + FloatToStr(Chart1.Series[0].MaxXValue);

LabMinOX.Caption := 'Min значение по оси OX ' + FloatToStr(Chart1.Series[0].MinXValue);

LabMaxOY.Caption := 'Max значение по оси OY ' + FloatToStr(Chart1.Series[0].MaxYValue);

LabMinOY.Caption := 'Min значение по оси OY ' + FloatToStr(Chart1.Series[0].MinYValue);

LabAllPoint.Caption := 'Кол-во точек ' + IntToStr(Chart1.Series[0].Count);

end;

end;

procedure TPlotF.PlotGraphics(Sender: TObject);

var

tmpPointSeries:TPointSeries;

tmpLineSeries:TFastLineSeries;

tmpSurfaceSeries:TSurfaceSeries;

tmpTeeFunction:TCustomTeeFunction;

i,j, Num3d, tmpSeriesCount: integer;

tmpStep3d,tmp3dx,tmp3dy,tmp3dz: double;

Nm,Nmax,Nsloev,Podsloi,Nsloev **for** _this_sloi,

Podsloi **for** _this_sloi,s,d,minEmax,maxEmax,

Nm_kus,v,Number_kus,m,norm_koef_b,Size_ar,

tmpNumPoints,tmpNumPointsZ: integer;

razresh,Podsloi_t,Xe,Ye,KoefX,Tmp,Eshag,

Emax_lev,Emax_prav,razresh2,lev2,prav2,

Lev_dE,Prav_dE,dE_na_dL,norm_koef,Nsrednee,

tmpXValuesMax,tmpXValuesMin,tmpZValuesMax,tmpZValuesMin,

tmpPeriod,tmpPeriodZ: double;

Nkusochkov: array of integer;

Glub_deltaE,Nch_na_No,tmp_t_arr: array of double;

tmp_y_arr: array of array of double;

ShowNumberOverBarrier: boolean;

NumberOverBarrier,NumberOverBarrierT: integer;

tmp_f_arr: array of array of double;

Ep: array of double;

Epop,Ep_min,Ep_max,Ep_shag: double;

SumF:extended;

tmpN_V, tmpN_X: array of integer;

tmpa, tmpb, tmpVmin, tmpVmax, tmpV,

```

tmpXmin, tmpXmax, tmpX, tmpLeftEf, tmpRightEf: double;
tmpNInterval, tmpInterval, tmpEffect: integer;
tmpHit: boolean;
tmp_Uxx, tmpEp: double;
ArrLimNmOnV: array of byte;
procedure MinMaxValues(NumSer:integer);
begin
  case NumSer of
1,3,17,18,21,22:
    begin
      tmpXValuesMin:=StrToFloat(XValuesMin.Text);
      tmpXValuesMax:=StrToFloat(XValuesMax.Text);
    end;
  2: begin
      tmpXValuesMin:=StrToFloat(XValuesMin.Text);
      tmpXValuesMax:=StrToFloat(XValuesMax.Text);
      tmpZValuesMin:=StrToFloat(ZValuesMin.Text);
      tmpZValuesMax:=StrToFloat(ZValuesMax.Text);
    end;
  end;
end;
procedure CommonInSeries(NumSer: integer);
begin
  LimitF.LimitStart.Enabled := False;
  LimitF.LimitEnd.Enabled := False;
  LimitF.CheckHistog.Enabled := False;
  PlotHistogramMain := False;
  case NumSer of
    2: begin
      Chart1.DepthAxis.Visible := True;
      Chart1.View3D := True;
    end;
    5,6,7: begin
      LimitF.LimitStart.Enabled := True;
      LimitF.LimitEnd.Enabled := True;
    end;
    9: begin
      Chart1.Title.Visible:=True;
    end;
    11: begin
      LimitF.LimitStart.Enabled := True;
      LimitF.LimitEnd.Enabled := True;
      razresh := StrToFloat(AdvOptF.LabeledEdit10.Text);

```

```

    end;
12: begin
    razresh := StrToFloat(AdvOptF.LabeledEdit10.Text);
    end;
13,14: begin
    razresh := StrToFloat(AdvOptF.LabeledEdit1.Text);
    Chart1.Title.Visible := True;
    end;
15: begin
    LimitF.LimitStart.Enabled := True;
    LimitF.LimitEnd.Enabled := True;
    razresh := StrToFloat(AdvOptF.LabeledEdit10.Text);
    end;
16: begin
    razresh := StrToFloat(AdvOptF.LabeledEdit10.Text);
    end;
19: begin
    razresh := StrToFloat(AdvOptF.EdResolution.Text);
    Chart1.Title.Visible := True;
    LimitF.LimitStart.Enabled := True;
    LimitF.LimitEnd.Enabled := True;
    LimitF.CheckHistog.Enabled := True;
    end;
20: begin
    razresh := StrToFloat(AdvOptF.EdResolution.Text);
    Chart1.Title.Visible := True;
    end;
end;
end;
procedure CommonInEnLossSeries;
begin
    try
        Gauge1.Progress:=0;
        Nm:=Trajects.CntOfTraject;
        Nmax:=Trajects.Traject[0].Nmax;
        if AdvOptF.CheckBox7.Checked=True then
            begin
                Nsloev:=StrToInt(AdvOptF.LabeledEdit9.Text);
                Nsloev_for_this_sloi:=Round(Nsloev/RoundTo(Trajects.Traject[0].Nmax/Nmax,-2));
                if LoadMyDataDone then
                    begin
                        if ((Nmax+1) mod Nsloev)<>0 then

```

```

        if (cnt_for_Saving_if_N_Point_True_on_one_step_tau mod
Round((Nmax+1)/Nsloev))<>0 then
            begin
                ShowMessage('Число точек в насчитанном слое (сохраненном) не кратно числу
задаваемых слоев');
                Exit;
            end;
            Podsloi:=Round((Trajects.Traject[0].Nmax+1)/Nsloev)-1;
            Podsloi_t := tau_As_Step_Saving*Podsloi
        end
    else
        begin
            if ((Nmax+1) mod Nsloev)<>0 then
                begin
                    ShowMessage('Число подслоев не кратно числу узлов');
                    Exit;
                end;
                Podsloi:=Round((Trajects.Traject[0].Nmax+1)/Nsloev)-1;
                Podsloi_t := Trajects.Traject[0].tau*Podsloi;
            end;
            Podsloi_for_this_sloi:=Round((Nmax+1)/Nsloev)-1;
            AdvOptF.Label16.Caption:='1 Precoat = ' + FloatToStr(Podsloi+1) + ' points = '
            + FloatToStr(RoundTo(Podsloi_t/epsilon*d_small,-2)) + ' A ';
            AdvOptF.LabeledEdit5.Text:=FloatToStr(Podsloi+1);
        end;
    except
        MessageDlg('Ошибка, возможно данные не загружены',mtError,[mbOk], 0);
    end;
end;

begin
    tmpSeriesCount:=Chart1.SeriesCount-1;
    for i:=0 to tmpSeriesCount do
        Chart1.Series[0].Free;
        TeeOpenGL.Active:=GLBox.Checked;
        Case TreeView1.Selected.StateIndex of
            1: begin
                Form1.InitialConditions;
                VizibleChartWithMaxSpeed(10000);
                MinMaxValues(1);
                tmpTeeFunction:=TCustomTeeFunction.Create(Self);
                tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculate;
                tmpTeeFunction.StartX:=tmpXValuesMin;

```

```

tmpPeriod:=1/1000;
tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);
tmpTeeFunction.Period:=tmpPeriod;
tmpTeeFunction.NumPoints:=tmpNumPoints;
tmpLineSeries:=TFastLineSeries.Create(self);
tmpLineSeries.SetFunction(tmpTeeFunction);
Chart1.AddSeries(tmpLineSeries);
Chart1.LeftAxis.Title.Caption:='U(x), эВ';
Chart1.BottomAxis.Title.Caption:='x/d';

```

end;

2: **begin**

```

Memo1.Text:='3-х мерный потенциал кристалла';
Form1.InitialConditions;
VizableChartWithMaxSpeed(10000);
CommonInSeries(2);
MinMaxValues(2);
tmpSurfaceSeries:=TSurfaceSeries.Create(Self);
Chart1.AddSeries(tmpSurfaceSeries);
tmpSurfaceSeries.IrregularGrid:=True;
tmpSurfaceSeries.FastBrush:=True;
tmpSurfaceSeries.UseColorRange:=False;
tmpSurfaceSeries.UsePalette:=True;
tmpSurfaceSeries.PaletteStyle:=psStrong;
tmpPeriod:=1/100;
tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);
tmpPeriodZ:=1/10000;
tmpNumPointsZ:=Round((tmpZValuesMax-tmpZValuesMin)/tmpPeriodZ);
tmpSurfaceSeries.NumXValues:=tmpNumPoints;
tmpSurfaceSeries.NumZValues:=tmpNumPointsZ;
tmpSurfaceSeries.TimesZOrder:=5;
for i:=0 to tmpNumPoints-1 do
  begin
    tmp3dx:=tmpPeriod*i;
    for j:=0 to tmpNumPointsZ-1 do
      begin
        tmp3dz:=tmpPeriodZ*j;
        tmp3dy:=Usp(tmp3dx,tmp3dz);
        tmpSurfaceSeries.AddXYZ(tmp3dx,tmp3dy,tmp3dz);
      end;
    end;
  end;
Chart1.LeftAxis.Title.Caption:='U(x,T), эВ';
Chart1.BottomAxis.Title.Caption:='x/d';
Chart1.DepthAxis.Title.Caption:='T';

```

```

end;
3: begin
    Form1.InitialConditions;
    VizableChartWithMaxSpeed(10000);
    CommonInSeries(3);
    MinMaxValues(3);
    Chart1.LeftAxis.Title.Caption:='D, мкрад2/МКМ';
    Chart1.BottomAxis.Title.Caption:='x/ax';
    tmpTeeFunction:=TCustomTeeFunction.Create(Self);
    tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculate2;
    tmpTeeFunction.StartX:=tmpXValuesMin;
    tmpPeriod:=1/1000;
    tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);
    tmpTeeFunction.Period:=tmpPeriod;
    tmpTeeFunction.NumPoints:=tmpNumPoints;
    tmpLineSeries:=TFastLineSeries.Create(self);
    tmpLineSeries.SetFunction(tmpTeeFunction);
    Chart1.LeftAxis.Logarithmic:=True;
    Chart1.AddSeries(tmpLineSeries);
end;
17,18: begin
    Form1.InitialConditions;
    VizableChartWithMaxSpeed(10000);
    if TreeView1.Selected.StateIndex=17 then
        begin
            CommonInSeries(17);
            MinMaxValues(17);
            Chart1.LeftAxis.Title.Caption:='rho_el(x), A-3';
            tmpTeeFunction:=TCustomTeeFunction.Create(Self);
            tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculateRel;
        end;
    if TreeView1.Selected.StateIndex=18 then
        begin
            CommonInSeries(18);
            MinMaxValues(18);
            Chart1.LeftAxis.Title.Caption:='rho_nucl(x), A-3';
            tmpTeeFunction:=TCustomTeeFunction.Create(Self);
            tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculateRnucl;
        end;
    Chart1.BottomAxis.Title.Caption:='x/ax';
    tmpTeeFunction.StartX:=tmpXValuesMin;
    tmpPeriod:=1/1000;
    tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);

```

```

tmpTeeFunction.Period:=tmpPeriod;
tmpTeeFunction.NumPoints:=tmpNumPoints;
tmpLineSeries:=TFastLineSeries.Create(self);
tmpLineSeries.SetFunction(tmpTeeFunction);
Chart1.AddSeries(tmpLineSeries);
end;
4: begin
end;
5: begin
VizableChartWithMaxSpeed(10000);
CommonInEnLossSeries;
CommonInSeries(5);
LabWait.Caption:='Подождите...';
Chart1.LeftAxis.Title.Caption:='x / ax';
if AdvOptF.CheckBoxAngstrom.Checked=False then
Chart1.BottomAxis.Title.Caption:='ray';
if AdvOptF.CheckBoxAngstrom.Checked=True then
Chart1.BottomAxis.Title.Caption:='глубина, Анг.';
if CheckXtSm.Checked=True then
Chart1.BottomAxis.Title.Caption:='г л у б и н а, с м';
Gauge1.MaxValue:=Nm;
if (AdvOptF.CheckBoxAngstrom.Checked) or (CheckXtSm.Checked) then
begin
if CheckXtSm.Checked then
norm_koef:=(d_small/epsilon)/1000000000
else norm_koef:=d_small/epsilon;
SetLength(tmp_t_arr,Nmax+1);
for i:=0 to Nmax do
tmp_t_arr[i]:=Trajects.Traject[0].t_arr[i]*norm_koef;
for i:=0 to Nm-1 do
begin
tmpLineSeries:=TFastLineSeries.Create(self);
with tmpLineSeries.XValues do
begin
Value:=TChartValues(tmp_t_arr);
Count:=Nmax+1;
Modified:=True;
end;
with tmpLineSeries.YValues do
begin
Value:=TChartValues(Trajects.Traject[i].x_arr);
Count:=Nmax+1;
Modified:=True;

```



```

        end;
        Chart1.AddSeries(tmpLineSeries);
        tmpLineSeries.Color := clBlack;
        Gauge1.Progress:=i+1;
    end;
end
else
    begin
        norm_koef:=1;
        for i:=0 to Nm-1 do
            begin
                tmpLineSeries:=TFastLineSeries.Create(self);
                with tmpLineSeries.XValues do
                    begin
                        Value:=TChartValues(Trajects.Traject[i].t_arr);
                        Count:=Nmax+1;
                        Modified:=True;
                    end;
                with tmpLineSeries.YValues do
                    begin
                        Value:=TChartValues(Trajects.Traject[i].x_arr);
                        Count:=Nmax+1;
                        Modified:=True;
                    end;
                Chart1.AddSeries(tmpLineSeries);
                tmpLineSeries.Color := clBlack;
                Gauge1.Progress:=i+1;
            end;
        end;
    end;
6: begin
    VizableChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(6);
    LabWait.Caption:='Подождите...';
    Chart1.LeftAxis.Title.Caption:='v(t)';
    if AdvOptF.CheckBoxAngstrom.Checked=False then
        Chart1.BottomAxis.Title.Caption:='ray';
    if AdvOptF.CheckBoxAngstrom.Checked=True then
        Chart1.BottomAxis.Title.Caption:='глубина, Анг.';
        Gauge1.MaxValue:=Nm;
        if AdvOptF.CheckBoxAngstrom.Checked then
            begin

```

```

norm_koef:=d_small/epsilon;
SetLength(tmp_t_arr,Nmax+1);
for i:=0 to Nmax do
  tmp_t_arr[i]:=Trajects.Traject[0].t_arr[i]*norm_koef;
for i:=0 to Nm-1 do
  begin
    tmpLineSeries:=TFastLineSeries.Create(self);
    with tmpLineSeries.XValues do
      begin
        Value:=TChartValues(tmp_t_arr);
        Count:=Nmax+1;
        Modified:=True;
      end;
    with tmpLineSeries.YValues do
      begin
        Value:=TChartValues(Trajects.Traject[i].v_arr);
        Count:=Nmax+1;
        Modified:=True;
      end;
    Chart1.AddSeries(tmpLineSeries);
    tmpLineSeries.Color := clBlack;
    Gauge1.Progress:=i+1;
  end;
end
else
  begin
    norm_koef:=1;
    for i:=0 to Nm-1 do
      begin
        tmpLineSeries:=TFastLineSeries.Create(self);
        with tmpLineSeries.XValues do
          begin
            Value:=TChartValues(Trajects.Traject[i].t_arr);
            Count:=Nmax+1;
            Modified:=True;
          end;
        with tmpLineSeries.YValues do
          begin
            Value:=TChartValues(Trajects.Traject[i].v_arr);
            Count:=Nmax+1;
            Modified:=True;
          end;
        Chart1.AddSeries(tmpLineSeries);

```

```

        tmpLineSeries.Color := clBlack;
        Gauge1.Progress:=i+1;
    end;
end;
end;
7: begin
    VizableChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(7);
    LabWait.Caption:='Подождите...';
    Chart1.LeftAxis.Title.Caption:='v(t)';
    Chart1.BottomAxis.Title.Caption:='x(t)';
    Gauge1.MaxValue:=Trajects.CntOfTraject;
    for i:=0 to Nm-1 do
        begin
            tmpPointSeries:=TPointSeries.Create(self);
            tmpPointSeries.Pointer.Style:=psCircle;
            tmpPointSeries.Pointer.HorizSize:=1;
            tmpPointSeries.Pointer.VertSize:=1;
            tmpPointSeries.Pointer.Pen.Visible:=False;
            with tmpPointSeries.XValues do
                begin
                    Value:=TChartValues(Trajects.Traject[i].x_arr);
                    Count:=Nmax+1;
                    Modified:=True;
                end;
            with tmpPointSeries.YValues do
                begin
                    Value:=TChartValues(Trajects.Traject[i].v_arr);
                    Count:=Nmax+1;
                    Modified:=True;
                end;
            Chart1.AddSeries(tmpPointSeries);
            tmpPointSeries.Color := clBlack;
            Gauge1.Progress:=i+1;
        end;
    end;
9: begin
    VizableChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(9);
    LabWait.Caption:='Подождите...';
    Chart1.LeftAxis.Title.Caption:='Плотность в канале';

```

```

Chart1.BottomAxis.Title.Caption:='Еп, эВ';
tmpLineSeries:=TFastLineSeries.Create(self);
if ShowNumberOverBarrier=false then
  begin
    NumberOverBarrier:=0;
    for i:=0 to Nm-1 do
      if Trajects.Traject[i].OverBarrier=true then inc(NumberOverBarrier);
      AdvOptF.Label12.Caption:='Надбарьерных '+IntToStr(NumberOverBarrier);
      ShowNumberOverBarrier:=True;
    end;
    SetLength(tmp_f_arr,Nm,Nmax+1);
    for i:=0 to Nm-1 do
      for j:=0 to Nmax do
        tmp_f_arr[i,j]:=Trajects.Traject[i].f_arr[j];
      SetLength(Ep,Nm);
      AdvOptF.Label14.Caption:='Центр (для 1й частицы) ~
'+FloatToStr(RoundTo((Vmax*Sqr(Trajects.Traject[0].v_arr[glub])/2+U(Trajects.Traject[0].x_arr[
glub])), -2))+' эВ';
      NumberOverBarrierT:=0;
      for i:=0 to Nm-1 do
        if (Trajects.Traject[i].OverBarrier=true) and (Trajects.Traject[i].CriticalPoint-1<=glub)
          then inc(NumberOverBarrierT);
      AdvOptF.Label13.Caption:='На этой глубине '+IntToStr(NumberOverBarrierT);
      Ep_min:=StrToFloat(AdvOptF.Edit8.Text);
      Ep_max:=StrToFloat(AdvOptF.Edit9.Text);
      Ep_shag:=0.01;
      Epop:=Ep_min;
      Gauge1.MaxValue:=Nm+Nm*Round((Ep_max-Ep_min)/Ep_shag);
      Gauge1.Progress:=0;
      for i:=0 to Nm-1 do
        begin
          j:=glub;
          while tmp_f_arr[i,j]<>0 do j:=j-1;
          Ep[i]:=Vmax*Sqr(Trajects.Traject[i].v_arr[j])/(2)+U(Trajects.Traject[i].x_arr[j]);
          Gauge1.Progress:=Gauge1.Progress+1;
        end;
      while Epop<=Ep_max do
        begin
          SumF:=0;
          for i:=0 to Nm-1 do
            if (AdvOptF.CheckBox4.Checked=False) or ((AdvOptF.CheckBox4.Checked=True)
and

```

```

        ((Trajects.Traject[i].OverBarrier=false) or ((Trajects.Traject[i].OverBarrier=True) and
        (Trajects.Traject[i].CriticalPoint-1>glub)))) then
            begin
                if tmp_f_arr[i,glub]<porog_min_eV2 then
                    tmp_f_arr[i,glub]:=tmp_f_arr[i,glub]+porog_min_eV2;
                    SumF:=SumF+exp(-Sqr(Epop-
                    Ep[i])/(2*tmp_f_arr[i,glub]))/Sqrt(2*Pi*tmp_f_arr[i,glub]));
                    Gauge1.Progress:=Gauge1.Progress+1;
                end
                else Gauge1.Progress:=Gauge1.Progress+1;
                SumF:=SumF/Nm;
                tmpLineSeries.AddXY(Epop,SumF,",clBlack);
                Epop:=Epop+Ep_shag;
            end;
            Chart1.Title.Text.Text:='На глубине '+
                FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
                A'+
                ' (в '+IntToStr(glub+1)+' точке)';
            Chart1.AddSeries(tmpLineSeries);
            tmpLineSeries.Color := clBlack;
        end;

```

```

11: begin
    VISIBLEChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(11);
    LabWait.Caption:='Подождите...';
    Chart1.LeftAxis.Title.Caption := 'дельта E';
    if AdvOptF.CheckBoxAngstrom.Checked=True then
        begin
            Chart1.BottomAxis.Title.Caption:='Глубина, Ангстрем';
            KoefX:=1/epsilon*d_small;
        end
        else
            begin
                Chart1.BottomAxis.Title.Caption:='Глубина, тау';
                KoefX:=1;
            end;
    Gauge1.MaxValue:=Nm*Nmax;
    for s:=0 to Nm-1 do
        begin
            tmpLineSeries:=TFastLineSeries.Create(self);
            for d:=0 to Nmax do

```

```

    begin
        Xe:=Trajects.Traject[s].t_arr[d]*KoefX;
        Ye:=deltaE[s,d];
        tmpLineSeries.AddXY(Xe,Ye,",clBlack);
        Gauge1.Progress:=Gauge1.Progress+1;
    end;
    Chart1.AddSeries(tmpLineSeries);
    tmpLineSeries.Color := clBlack;
end;
end;
12: begin
    VizableChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(12);
    CalculateEmax(Nm,Nmax);
    AdvOptF.RadioButton3.Checked:=True;
    minEmax:=0;
    maxEmax:=0;
    for i:=1 to Nm-1 do
        if Emax[i]<Emax[minEmax] then minEmax:=i;
        Tmp:=Emax[0];
        Emax[0]:=Emax[minEmax];
        Emax[minEmax]:=Tmp;
        for j:=1 to Nm-1 do
            if Emax[j]>Emax[maxEmax] then maxEmax:=j;
            Tmp:=Emax[Nm-1];
            Emax[Nm-1]:=Emax[maxEmax];
            Emax[maxEmax]:=Tmp;
            Eshag:=razresh;
            Nm_kus:=Round((Emax[Nm-1]-Emax[0])/Eshag);
            if ((Emax[Nm-1]-Emax[0])/Eshag)>Nm_kus then Nm_kus:=Nm_kus+2 else
Nm_kus:=Nm_kus+1;
            if AdvOptF.CheckBox5.Checked=True then
                if Nm_kus>Nm+1 then
                    begin
                        MessageDlg('Задано слишком маленькое разрешение, увеличиваем в соответствии с
кол-вом траекторий',mtInformation,[mbOk],0);
                        Eshag:=(Emax[Nm-1]-Emax[0])/Nm;
                        Nm_kus:=Round((Emax[Nm-1]-Emax[0])/Eshag);
                        AdvOptF.LabeledEdit10.Text:=FloatToStr(Eshag);
                        if ((Emax[Nm-1]-Emax[0])/Eshag+0.00000001)>=Nm_kus then Nm_kus:=Nm_kus+1
                    else Nm_kus:=Nm_kus;
                end;
            end;

```

```

SetLength(Nkusochkov,Nm_kus);
Gauge1.MaxValue:=Nm;
for v:=0 to Nm-1 do
  begin
    Number_kus:=0;
    Emax_lev:=Emax[0];
    Emax_prav:=Emax_lev+Eshag;
    if (Emax[v]>=Emax_lev) and (Emax[v]<Emax_prav) then inc(Nkusochkov[0]) else
      begin
        repeat
          Emax_lev:=Emax_lev+Eshag;
          Emax_prav:=Emax_lev+Eshag;
          inc(Number_kus);
        until ((Emax[v]>Emax_lev) and (Emax[v]<=Emax_prav+0.0001));
        inc(Nkusochkov[Number_kus]);
      end;
    Gauge1.Progress:=Gauge1.Progress+1;
  end;
Chart1.LeftAxis.Title.Caption :='Число частиц N на глубине '+
  FloatToStr(RoundTo(Trajects.Traject[0].t_arr[Nmax]/epsilon*d_small,-2))+ ' A';
Chart1.BottomAxis.Title.Caption:='Энергии частиц E на данной глубине, с учетом
потерь';
tmpLineSeries:=TFastLineSeries.Create(self);
tmpLineSeries.Stairs:=True;
if (AdvOptF.CheckBox6.Checked=True) then tmpLineSe-
ries.AddXY(StrToFloat(AdvOptF.LabeledEdit6.Text),0,"clBlack)
  else if ((AdvOptF.CheckBox5.Checked=False) and (AdvOptF.CheckBox6.Checked=False))
then
  begin
    AdvOptF.LabeledEdit6.Text:=FloatToStr(Emax[0]-(Emax[Nm-1]-Emax[0]));
    tmpLineSeries.AddXY((Emax[0]-(Emax[Nm-1]-Emax[0])),0,"clBlack);
  end;
  for m:=0 to Nm_kus-1 do
    tmpLineSeries.AddXY(Emax[0]+m*Eshag,Nkusochkov[m],"clBlack);
    if (AdvOptF.CheckBox6.Checked=True) then tmpLineSe-
ries.AddXY(StrToFloat(AdvOptF.LabeledEdit7.Text),0,"clBlack)
      else if ((AdvOptF.CheckBox5.Checked=False) and (AdvOptF.CheckBox6.Checked=False))
then
      begin
        AdvOptF.LabeledEdit7.Text:=FloatToStr(Emax[Nm-1]+(Emax[Nm-1]-Emax[0]));
        tmpLineSeries.AddXY((Emax[Nm-1]+(Emax[Nm-1]-Emax[0])),0,"clBlack);
      end;
    Chart1.AddSeries(tmpLineSeries);

```

```

    tmpLineSeries.Color := clBlack;
end;
13,14: begin
    VizibleChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(13);
    with TreeView1.Selected do
    begin
    if CheckBox11.Checked=True then
    begin
        if StateIndex=14 then norm_koef_b:=100
        else if StateIndex=13 then norm_koef_b:=1000000;
    end
    else norm_koef_b:=1;
    minEmax:=0;
    maxEmax:=0;
    SetLength(Glub_deltaE,Nm);
    if AdvOptF.CheckBox7.Checked=True then
    begin
        if StateIndex=14 then
            for i:=0 to Nm-1 do Glub_deltaE[i]:=(deltaE[i,glub]-deltaE[i,glub-
Podstoi])/ (Podstoi_t/epsilon*d_small)
        else if StateIndex=13 then
            for i:=0 to Nm-1 do Glub_deltaE[i]:=(deltaE[i,glub]-deltaE[i,glub-Podstoi]);
        end
        else for i:=0 to Nm-1 do Glub_deltaE[i]:=deltaE[i,glub];
    if AdvOptF.RadioButton4.Checked=True then
    begin
        razresh:=StrToFloat(AdvOptF.LabeledEdit8.Text);
        if AdvOptF.CheckBox7.Checked=False then
        begin
            if StateIndex=14 then
            begin
                for i:=0 to Nm-1 do
Glub_deltaE[i]:=Glub_deltaE[i]/(Trajects.Traject[0].t_arr[glub]/epsilon*d_small);
                Ad-
vOptF.LabeledEdit1.Text:=FloatToStr(RoundTo(razresh*(Trajects.Traject[0].t_arr[glub]/epsilon*d_small),-2));
            end
            else if StateIndex=13 then
            begin
                razresh:=StrToFloat(AdvOptF.LabeledEdit1.Text);
                AdvOptF.RadioButton3.Checked:=True;

```



```

        end;
    end;
end;
if AdvOptF.RadioButton3.Checked=True then
begin
    Ad-
vOptF.LabeledEdit8.Text:=FloatToStr(RoundTo(razresh/(Trajects.Traject[0].t_arr[glub]/epsilon*d_
small),-7));
    end;
    for i:=1 to Nm-1 do
        if Glub_deltaE[i]<Glub_deltaE[minEmax] then minEmax:=i;
        Tmp:=Glub_deltaE[0];
        Glub_deltaE[0]:=Glub_deltaE[minEmax];
        Glub_deltaE[minEmax]:=Tmp;
        for j:=1 to Nm-1 do
            if Glub_deltaE[j]>Glub_deltaE[maxEmax] then maxEmax:=j;
            Tmp:=Glub_deltaE[Nm-1];
            Glub_deltaE[Nm-1]:=Glub_deltaE[maxEmax];
            Glub_deltaE[maxEmax]:=Tmp;
            Eshag:=razresh;
            Nm_kus:=Round((Glub_deltaE[Nm-1]-Glub_deltaE[0])/Eshag);
            if ((Glub_deltaE[Nm-1]-Glub_deltaE[0])/Eshag)>Nm_kus then Nm_kus:=Nm_kus+2 else
Nm_kus:=Nm_kus+1;
            if AdvOptF.CheckBox5.Checked=True then
                if Nm_kus>Nm+1 then
                    begin
                        MessageDlg('Задано слишком маленькое разрешение, увеличиваем в соответствии с
кол-вом траекторий',mtInformation,[mbOk],0);
                        Eshag:=(Glub_deltaE[Nm-1]-Glub_deltaE[0])/Nm;
                        Nm_kus:=Round((Glub_deltaE[Nm-1]-Glub_deltaE[0])/Eshag);
                        AdvOptF.LabeledEdit1.Text:=FloatToStr(Eshag);
                        if ((Glub_deltaE[Nm-1]-Glub_deltaE[0])/Eshag+0.00000001)>=Nm_kus then
Nm_kus:=Nm_kus+1 else Nm_kus:=Nm_kus;
                    end;
                SetLength(Nkusochkov,Nm_kus);
                if CheckBox1.Checked=True then
                    begin
                        Gauge1.MaxValue:=Nm;
                        for v:=0 to Nm-1 do
                            begin
                                Number_kus:=0;
                                Emax_lev:=Glub_deltaE[0];
                                Emax_prav:=Emax_lev+Eshag;

```

```

        if (Glub_deltaE[v]>=Emax_lev) and (Glub_deltaE[v]<Emax_prav) then
inc(Nkusochkov[0]) else
    begin
        repeat
            Emax_lev:=Emax_lev+Eshag;
            Emax_prav:=Emax_lev+Eshag;
            inc(Number_kus);
        until ((Glub_deltaE[v]>Emax_lev) and (Glub_deltaE[v]<=Emax_prav+0.0001));
        inc(Nkusochkov[Number_kus]);
    end;
    Gauge1.Progress:=Gauge1.Progress+1;
end;
else
    begin
        if AdvOptF.RadioButton5.Checked=True then
            begin
                razresh2:=StrToFloat(AdvOptF.LabeledEdit10.Text);
                Ad-
vOptF.LabeledEdit11.Text:=FloatToStr(RoundTo(razresh2/(Trajects.Traject[0].t_arr[glub]/epsilon*
d_small),-7));
            end
        else
            if AdvOptF.RadioButton6.Checked=True then
                begin
                    if StateIndex=14 then
                        begin
                            razresh2:=StrToFloat(AdvOptF.LabeledEdit11.Text);
                            Ad-
vOptF.LabeledEdit10.Text:=FloatToStr(RoundTo(razresh2*(Trajects.Traject[0].t_arr[glub]/epsilon
*d_small),-2));
                        end
                    else if StateIndex=13 then
                        begin
                            razresh2:=StrToFloat(AdvOptF.LabeledEdit10.Text);
                            AdvOptF.RadioButton5.Checked:=True;
                        end;
                    end;
                end;
                lev2:=Glub_deltaE[0]-(razresh2-razresh);
                prav2:=lev2+razresh2;
                Gauge1.MaxValue:=Nm*Nm_kus;
                while ((lev2<Glub_deltaE[Nm-1]) and (Prav2>Glub_deltaE[0])) do
                    begin

```

```

for v:=0 to Nm-1 do
  begin
    Number_kus:=0;
    Emax_lev:=Glub_deltaE[0];
    Emax_prav:=Emax_lev+Eshag;
    if (((Glub_deltaE[v]>=Emax_lev) and (Glub_deltaE[v]<Emax_prav)) and
    ((Emax_lev>=lev2) and (Emax_lev<=Prav2-Eshag))) then inc(Nkusochkov[0]) else
      begin
        repeat
          Emax_lev:=Emax_lev+Eshag;
          Emax_prav:=Emax_lev+Eshag;
          inc(Number_kus);
        until (((Glub_deltaE[v]>Emax_lev) and (Glub_deltaE[v]<Emax_prav)) and
        ((Emax_lev>=lev2) and (Emax_lev<=Prav2-Eshag))) or (Emax_lev>Prav2);
        if ((Emax_lev<Prav2) and ((Emax_lev>=lev2) and (Emax_lev<=Prav2-
        Eshag))) then inc(Nkusochkov[Number_kus]);
        end;
        Gauge1.Progress:=Gauge1.Progress+1;
      end;
      lev2:=lev2+razresh;
      prav2:=prav2+razresh;
    end;
  end;

```

```

Chart1.Title.Text.Text:='Loss of Energy on depth '+
  FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
  A'+
  ' (in '+IntToStr(glub+1)+' point)';
Chart1.LeftAxis.Title.Caption :='Число частиц N/No';
if CheckBox2.Checked=True then Chart1.LeftAxis.Title.Caption :='Counts (N/No)'
  else if CheckBox2.Checked=False then Chart1.LeftAxis.Title.Caption :='Counts';
Chart1.BottomAxis.Title.Font.Name:='Symbol';
Chart1.BottomAxis.Title.Font.Size:=11;
if CheckBox11.Checked=True then
  begin
    if StateIndex=14 then Chart1.BottomAxis.Title.Caption:=#68#69#44+'
    '+#77#39#66#47#109#107+' ('+#68#235+'='+'
      FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
      '+#65+')'
    else if StateIndex=13 then Chart1.BottomAxis.Title.Caption:=#68#69#44+'
    '+#77#39#66+' ('+#68#235+'='+'
      FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
      '+#65+')';
  end;

```

```

    end
    else
        begin
            if StateIndex=14 then Chart1.BottomAxis.Title.Caption:=#68#69#47#68#235#44+'
            '+#39#66#47#65+' ('+#68#235+'='+'
                FloatToStr(RoundTo(Trajets.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
            '+#65+')'
                else if StateIndex=13 then Chart1.BottomAxis.Title.Caption:=#68#69#44+' '+#39#66+'
            ('+#68#235+'='+'
                FloatToStr(RoundTo(Trajets.Traject[0].t_arr[glub]/epsilon*d_small,-2))+
            '+#65+')';
        end;
        tmpLineSeries:=TFastLineSeries.Create(self);
        if CheckBox1.Checked=True then
            begin
                tmpLineSeries.Stairs:=True;
            end
            else if CheckBox1.Checked=False then
                begin
                    tmpLineSeries.Stairs:=False;
                end;
            if (AdvOptF.CheckBox6.Checked=True) then tmpLineSe-
            ries.AddXY(StrToFloat(AdvOptF.LabeledEdit6.Text),0,",clBlack)
            else if ((AdvOptF.CheckBox5.Checked=False) and (AdvOptF.CheckBox6.Checked=False))
            then
                begin
                    if AdvOptF.RadioButton3.Checked=True then
                        begin
                            if StateIndex=14 then
                                begin
                                    AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
                                    Glub_deltaE[0]))/(Trajets.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b);
                                    tmpLineSeries.AddXY((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
                                    Glub_deltaE[0]))/(Trajets.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b,0,",clBlack);
                                    if CheckBox1.Checked=False then
                                        begin
                                            AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-
                                            razresh)/(Trajets.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b);
                                            tmpLineSeries.AddXY((Glub_deltaE[0]-
                                            razresh)/(Trajets.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b,0,",clBlack);
                                        end;
                                    end
                                    else if StateIndex=13 then

```

```

        begin
            AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
Glub_deltaE[0]))/norm_koef_b);
            tmpLineSeries.AddXY((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
Glub_deltaE[0]))/norm_koef_b,0,"clBlack);
            if CheckBox1.Checked=False then
                begin
                    AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-
razresh)/norm_koef_b);
                    tmpLineSeries.AddXY((Glub_deltaE[0]-razresh)/norm_koef_b,0,"clBlack);
                end;
            end;
        end
    else if AdvOptF.RadioButton4.Checked=True then
        begin
            AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
Glub_deltaE[0]))/norm_koef_b);
            tmpLineSeries.AddXY((Glub_deltaE[0]-(Glub_deltaE[Nm-1]-
Glub_deltaE[0]))/norm_koef_b,0,"clBlack);
            if CheckBox1.Checked=False then
                begin
                    AdvOptF.LabeledEdit6.Text:=FloatToStr((Glub_deltaE[0]-razresh)/norm_koef_b);
                    tmpLineSeries.AddXY((Glub_deltaE[0]-razresh)/norm_koef_b,0,"clBlack);
                end;
            end
        end;
    for m:=0 to Nm_kus-1 do
        begin
            if AdvOptF.RadioButton3.Checked=True then
                begin
                    if StateIndex=14 then
                        Xe:=(Glub_deltaE[0]+m*Eshag)/(Trajects.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b
                    else if StateIndex=13 then Xe:=(Glub_deltaE[0]+m*Eshag)/norm_koef_b;
                end
                else if AdvOptF.RadioButton4.Checked=True then
                    Xe:=(Glub_deltaE[0]+m*Eshag)/norm_koef_b;
                    if CheckBox2.Checked=True then Ye:=Nkusochkov[m]/Nm
                    else if CheckBox2.Checked=False then Ye:=Nkusochkov[m];
                    tmpLineSeries.AddXY(Xe,Ye,"clBlack);
                end;
            if (AdvOptF.CheckBox6.Checked=True) then tmpLineSe-
ries.AddXY(StrToFloat(AdvOptF.LabeledEdit7.Text),0,"clBlack)

```

```

else if ((AdvOptF.CheckBox5.Checked=False) and (AdvOptF.CheckBox6.Checked=False))
then
begin
  if AdvOptF.RadioButton3.Checked=True then
  begin
    if StateIndex=14 then
    begin
      AdvOptF.LabeledEdit7.Text:=FloatToStr((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/(Trajects.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b);
      tmpLineSeries.AddXY((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/(Trajects.Traject[0].t_arr[glub]/epsilon*d_small)/norm_koef_b,0,"clBlack");
    end
    else if StateIndex=13 then
    begin
      AdvOptF.LabeledEdit7.Text:=FloatToStr((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/norm_koef_b);
      tmpLineSeries.AddXY((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/norm_koef_b,0,"clBlack");
    end;
  end
  else if AdvOptF.RadioButton4.Checked=True then
  begin
    AdvOptF.LabeledEdit7.Text:=FloatToStr((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/norm_koef_b);
    tmpLineSeries.AddXY((Glub_deltaE[Nm-1]+(Glub_deltaE[Nm-1]-Glub_deltaE[0]))/norm_koef_b,0,"clBlack");
  end;
end;
end;
Chart1.AddSeries(tmpLineSeries);
tmpLineSeries.Color := clBlack;
end;
end;
15: begin
  VizableChartWithMaxSpeed(10000);
  CommonInEnLossSeries;
  CommonInSeries(15);
  LabWait.Caption:='Подождите...';
  Chart1.LeftAxis.Title.Font.Name:='Symbol';
  Chart1.LeftAxis.Title.Font.Size:=11;
  Chart1.LeftAxis.Title.Caption :='#68#69#44+'+'#39#66#47#65;
  if AdvOptF.CheckBoxAngstrom.Checked=True then
Chart1.BottomAxis.Title.Caption:='Глубина, Ангстрем'
  else Chart1.BottomAxis.Title.Caption:='Глубина, тау';

```

```

Gauge1.MaxValue:=Nm;
if AdvOptF.CheckBoxAngstrom.Checked=True then
begin
    norm_koef:=d_small/epsilon;
    SetLength(tmp_t_arr,Nmax+1);
    for i:=0 to Nmax do
        tmp_t_arr[i]:=Trajects.Traject[0].t_arr[i]*norm_koef;
    SetLength(tmp_y_arr,Nm,Nmax+1);
    if Trajects.Traject[0].t_arr[0]=0 then
        begin
            for s:=0 to Nm-1 do
                begin
                    tmp_y_arr[s,0]:=0;
                    for i:=1 to Nmax do
                        tmp_y_arr[s,i]:=deltaE[s,i]/(Trajects.Traject[s].t_arr[i]*norm_koef);
                    end;
                end
            end
        else
            begin
                for s:=0 to Nm-1 do
                    for i:=0 to Nmax do
                        tmp_y_arr[s,i]:=deltaE[s,i]/(Trajects.Traject[s].t_arr[i]*norm_koef);
                    end;
                end
            for s:=0 to Nm-1 do
                begin
                    tmpLineSeries:=TFastLineSeries.Create(self);
                    with tmpLineSeries.XValues do
                        begin
                            Value:=TChartValues(tmp_t_arr);
                            Count:=Nmax+1;
                            Modified:=True;
                        end;
                    with tmpLineSeries.YValues do
                        begin
                            Value:=TChartValues(tmp_y_arr[s]);
                            Count:=Nmax+1;
                            Modified:=True;
                        end;
                    Chart1.AddSeries(tmpLineSeries);
                    tmpLineSeries.Color := clBlack;
                    Gauge1.Progress:=i+1;
                end;
            end
        end

```

```

else
begin
norm_koef:=d_small/epsilon;
SetLength(tmp_y_arr,Nm,Nmax+1);
if Trajects.Traject[0].t_arr[0]=0 then
begin
for s:=0 to Nm-1 do
begin
tmp_y_arr[s,0]:=0;
for i:=1 to Nmax do
tmp_y_arr[s,i]:=deltaE[s,i]/(Trajects.Traject[s].t_arr[i]*norm_koef);
end;
end
else
begin
for s:=0 to Nm-1 do
for i:=0 to Nmax do
tmp_y_arr[s,i]:=deltaE[s,i]/(Trajects.Traject[s].t_arr[i]*norm_koef);
end;
for s:=0 to Nm-1 do
begin
tmpLineSeries:=TFastLineSeries.Create(self);
with tmpLineSeries.XValues do
begin
Value:=TChartValues(Trajects.Traject[s].t_arr);
Count:=Nmax+1;
Modified:=True;
end;
with tmpLineSeries.YValues do
begin
Value:=TChartValues(tmp_y_arr[s]);
Count:=Nmax+1;
Modified:=True;
end;
Chart1.AddSeries(tmpLineSeries);
tmpLineSeries.Color := clBlack;
Gauge1.Progress:=i+1;
end;
end;
end;
16: begin
if ModelDechanneling = 1 then begin
VizibleChartWithMaxSpeed(10000);

```



```

CommonInEnLossSeries;
CommonInSeries(16);
Chart1.LeftAxis.Title.Caption := '1-Nch/No';
Chart1.BottomAxis.Title.Caption := 'Ангретрер';
Lev_dE := StrToFloat(AdvOptF.LabeledEdit3.Text);
Prav_dE := StrToFloat(AdvOptF.LabeledEdit4.Text);
if AdvOptF.CheckBox7.Checked = True then
  begin
    SetLength(Nch_na_No, Nsloev_for_this_sloi);
    Gauge1.MaxValue := Nm * (Nsloev_for_this_sloi) + Nsloev_for_this_sloi;
    d := 0;
    i := 0;
    if AllDataAveraging = False then begin
      for s := 0 to Nm-1 do
        begin
          while d <= Nsloev_for_this_sloi-1 do
            begin
              dE_na_dL := (deltaE[s, (d+1)*Podsloi+i] -
deltaE[s, d*Podsloi+i]) / (Podsloi_t/epsilon*d_small);
              if (dE_na_dL > Lev_dE) and (dE_na_dL < Prav_dE) then
Nch_na_No[d] := Nch_na_No[d] + 1/Nm;
              Gauge1.Progress := Gauge1.Progress + 1*Podsloi;
              d := d + 1;
              i := i + 1;
            end;
            d := 0;
            i := 0;
          end;
        end
      else
        begin
          for s := 0 to Nm-1 do
            begin
              i := 0;
              for d := 0 to Nsloev_for_this_sloi-1 do
                begin
                  while i <> (Podsloi+1)*(d+1) do
                    begin
                      dE_na_dL := deltaE[s, i] / (Trajects.Traject[s].t_arr[i] / epsilon*d_small);
                      if (dE_na_dL > Lev_dE) and (dE_na_dL < Prav_dE) then
Nch_na_No[d] := Nch_na_No[d] + 1 / (Nm*(Podsloi+1));
                      inc(i);
                      Gauge1.Progress := Gauge1.Progress + 1;

```

```

        end;
    end;
end;
end;
end
else
begin
    SetLength(Nch_na_No,Nmax+1);
    Gauge1.MaxValue:=Nm*(Nmax+1)+Nmax+1;
    for s:=0 to Nm-1 do
        for d:=0 to Nmax do
            if Trajects.Traject[0].t_arr[d]=0 then
                begin
                    Nch_na_No[d]:=1;
                    Gauge1.Progress:=Gauge1.Progress+1;
                end
            else
                begin
                    dE_na_dL:=deltaE[s,d]/(Trajects.Traject[s].t_arr[d]/epsilon*d_small);
                    if ((dE_na_dL)>Lev_dE) and ((dE_na_dL)<Prav_dE) then
Nch_na_No[d]:=Nch_na_No[d]+1/Nm;
                        Gauge1.Progress:=Gauge1.Progress+1;
                    end
                end;
            tmpLineSeries:=TFastLineSeries.Create(self);
            tmpLineSeries.Stairs:=True;
            tmpLineSeries.InvertedStairs:=True;
            if (RoundTo(Trajects.Traject[0].t_arr[0],-2))=0 then
                tmpLineSeries.AddXY(0,0,"clBlack)
            else tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[0]/epsilon*d_small, 0,"clBlack);
            if AdvOptF.CheckBox8.Checked=True then norm_koef:=StrToFloat(AdvOptF.Edit11.Text)
        else norm_koef:=1;
            if AdvOptF.CheckBox7.Checked=True then
                begin
                    tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[0]/epsilon*d_small,(1-
Nch_na_No[0])/norm_koef,"clBlack);
                    i:=0;
                    for d:=0 to Nsloev_for_this_sloi-1 do
                        begin
                            tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[(d+1)*Podsloi+i]/epsilon*d_small,(1-
Nch_na_No[d])/norm_koef,"clBlack);
                            Gauge1.Progress:=Gauge1.Progress+1;
                            i := i+1;

```

```

    end;
    tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[Nsloev_for_this_sloi*Podsloi+i-
1]/epsilon*d_small,0,"clBlack);
    end
    else
    begin
    for d:=0 to Nmax do
    begin
    tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[d]/epsilon*d_small,(1-
Nch_na_No[d])/norm_koef,"clBlack);
    Gauge1.Progress := Gauge1.Progress+1;
    end;
    tmpLineSeries.AddXY(Trajects.Traject[0].t_arr[Nmax]/epsilon*d_small, 0,"clBlack);
    end;
    Size_ar := Length(Nch_na_No);
    for d := 0 to Size_ar do Nsrednee := Nsrednee + Nch_na_No[d];
    AdvOptF.Label15.Caption := 'Dechannaling average = ' + FloatToStr(RoundTo((1-
Nsrednee/Size_ar)/norm_koef,-3));
    Chart1.AddSeries(tmpLineSeries);
    tmpLineSeries.Color := clBlack;
    end;
    if ModelDechanneling = 2 then begin
    VizibleChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(16);
    Chart1.LeftAxis.Title.Caption := '1-Nch/No';
    Chart1.BottomAxis.Title.Caption:='Англстрем';
    if AdvOptF.CheckBox7.Checked=True then
    begin
    SetLength(Nch_na_No,Nsloev_for_this_sloi);
    Gauge1.MaxValue:=Nm*(Nsloev_for_this_sloi)+Nsloev_for_this_sloi;
    d:=0;
    i:=0;
    if AllDataAveraging = False then begin
    for s:=0 to Nm-1 do
    begin
    while d <= Nsloev_for_this_sloi-1 do
    begin
    tmp_Uxx := ( U_xx(Trajects.Traject[s].x_arr[(d+1)*Podsloi+i]) -
    U_xx(Trajects.Traject[s].x_arr[(d)*Podsloi+i]) )/(Podsloi_t/epsilon*d_small);
    if (tmp_Uxx > 0) then Nch_na_No[d] := Nch_na_No[d] + 1/Nm;
    Gauge1.Progress := Gauge1.Progress+1;
    d := d + 1;

```

```

        i := i + 1;
    end;
    d := 0;
    i := 0;
end;
end
else
begin
    for s:=0 to Nm-1 do
        begin
            i:=0;
            for d:=0 to Nsloev_for_this_sloi-1 do
                begin
                    while i<>(Podsloi+1)*(d+1) do
                        begin
                            tmpEp :=
Vmax*Sqr(Trajects.Traject[s].v_arr[i])/2+U(Trajects.Traject[s].v_arr[i]);
                            if tmpEp < Vmax then
                                begin
                                    tmp_Uxx := U_xx(Trajects.Traject[s].x_arr[i]);
                                    if tmp_Uxx > 0 then Nch_na_No[d]:=Nch_na_No[d]+1/(Nm*(Podsloi+1));
                                end;
                                    inc(i);
                                end;
                                    Gauge1.Progress := Gauge1.Progress+1;
                                end;
                            end;
                        end;
                    end;
                end;
            end;
        end
    else
        begin
            SetLength(Nch_na_No, Nmax+1);
            Gauge1.MaxValue := Nm*(Nmax+1)+Nmax+1;
            tmpEp := 0;
            for s:=0 to Nm-1 do
                begin
                    d := 0;
                    while d < Nmax do
                        begin
                            tmpEp :=
Vmax*Sqr(Trajects.Traject[s].v_arr[d])/2+U(Trajects.Traject[s].v_arr[d]);
                            if tmpEp < Vmax then
                                begin

```

```

        tmp_Uxx := U_xx(Trajets.Traject[s].x_arr[d]);
        if tmp_Uxx > 0 then Nch_na_No[d] := Nch_na_No[d] + 1/Nm;
        end;
        d := d + 1;
        Gauge1.Progress:=Gauge1.Progress+1;
        end;
        end;
        Gauge1.Progress:=Gauge1.Progress+1;
        end;
        tmpLineSeries:=TFastLineSeries.Create(self);
        tmpLineSeries.Stairs:=True;
        tmpLineSeries.InvertedStairs:=True;
        if (RoundTo(Trajets.Traject[0].t_arr[0],-2))=0 then
            tmpLineSeries.AddXY(0,0,"clBlack)
            else tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[0]/epsilon*d_small, 0,"clBlack);
        if AdvOptF.CheckBox8.Checked=True then
norm_koef:=StrToFloat(AdvOptF.Edit11.Text) else norm_koef:=1;
        if AdvOptF.CheckBox7.Checked=True then
            begin
                tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[0]/epsilon*d_small,(1-
Nch_na_No[0])/norm_koef,"clBlack);
                i:=0;
                for d:=0 to Nsloev_for_this_sloi-1 do
                    begin
                        tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[(d+1)*Podsloi+i]/epsilon*d_small,(1-
Nch_na_No[d])/norm_koef,"clBlack);
                        Gauge1.Progress:=Gauge1.Progress+1;
                        i := i+1;
                    end;
                    tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[Nsloev_for_this_sloi*Podsloi+i-
1]/epsilon*d_small,0,"clBlack);
                end
            else
                begin
                    for d:=0 to Nmax do
                        begin
                            tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[d]/epsilon*d_small,(1-
Nch_na_No[d])/norm_koef,"clBlack);
                            Gauge1.Progress := Gauge1.Progress+1;
                        end;
                        tmpLineSeries.AddXY(Trajets.Traject[0].t_arr[Nmax]/epsilon*d_small, 0,"clBlack);
                    end;
                end;
                Size_ar := Length(Nch_na_No);

```

```

    for d := 0 to Size_ar do Nsrednee := Nsrednee + Nch_na_No[d];
    AdvOptF.Label15.Caption := 'Dechannaling average = ' + FloatToStr(RoundTo((1-
Nsrednee/Size_ar)/norm_koef,-3));
    Chart1.AddSeries(tmpLineSeries);
    tmpLineSeries.Color := clBlack;
  end;
end;
19: begin
  VizableChartWithMaxSpeed(10000);
  CommonInEnLossSeries;
  CommonInSeries(19);
  LabWait.Caption:='Подождите...';
  SetLength(ArrLimNmOnV, Nm);
  HistogramVX(1, Nm, True, razresh, ArrLimNmOnV, clBlack);
  PlotHistogramMain := True;
  PlotHistogramSecond := False;
end;
20: begin
  VizableChartWithMaxSpeed(10000);
  CommonInEnLossSeries;
  CommonInSeries(20);
  LabWait.Caption:='Подождите...';
  Chart1.LeftAxis.Title.Caption:='N';
  Chart1.BottomAxis.Title.Caption:='Поперечная координата x';
  tmpXmin := Trajects.Traject[0].x_arr[glub];
  tmpXmax := Trajects.Traject[0].x_arr[glub];
  for i:=1 to Nm-1 do
    begin
      tmpX := Trajects.Traject[i].x_arr[glub];
      if tmpX < tmpXmin then tmpXmin := tmpX
      else if tmpX > tmpXmax then tmpXmax := tmpX;
    end;
  tmpNInterval := Abs(Round(Int((tmpXmax - tmpXmin)/razresh))) + 1;
  SetLength(tmpN_X, tmpNInterval);
  for i:=0 to Nm-1 do
    begin
      tmpa := tmpXmin;
      tmpb := tmpa + razresh;
      tmpX := Trajects.Traject[i].x_arr[glub];
      tmpInterval := 0;
      tmpHit := False;
      while not tmpHit do
        begin

```

```

    if (tmpX >= tmpa) and (tmpX < tmpb) then
        begin
            tmpN_X[tmpInterval] := tmpN_X[tmpInterval] + 1;
            tmpHit := True;
        end
    else
        begin
            tmpa := tmpa + razresh;
            tmpb := tmpb + razresh;
            inc(tmpInterval);
        end;
    end;
end;

tmpLineSeries:=TFastLineSeries.Create(self);
tmpa := tmpXmin;
tmpLineSeries.AddXY(tmpa, 0, ", clBlack);
for i:=0 to tmpNInterval - 1 do
    begin
        tmpLineSeries.AddXY(tmpa, tmpN_X[i], ", clBlack);
        tmpa := tmpa + razresh;
    end;
tmpLineSeries.AddXY(tmpa, tmpN_X[tmpNInterval - 1], ", clBlack);
tmpLineSeries.AddXY(tmpa, 0, ", clBlack);
Finalize(tmpN_X);
Chart1.Title.Text.Text:='На глубине '+
    FloatToStr(RoundTo(Trajects.Traject[0].t_arr[glub]/epsilon*d_small,-2))+ ' A'+
    ' (в '+IntToStr(glub+1)+' точке)';
tmpb := (tmpa - tmpXmin) * 0.15;
Chart1.BottomAxis.SetMinMax(tmpXmin - tmpb, tmpa + tmpb);
tmpLineSeries.Stairs:=True;
Chart1.AddSeries(tmpLineSeries);
tmpLineSeries.Color := clBlack;
end;
21: begin
    Form1.InitialConditions;
    VizableChartWithMaxSpeed(10000);
    MinMaxValues(21);
    tmpTeeFunction:=TCustomTeeFunction.Create(Self);
    tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculateUx;
    tmpTeeFunction.StartX:=tmpXValuesMin;
    tmpPeriod:=1/1000;
    tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);
    tmpTeeFunction.Period:=tmpPeriod;

```

```

tmpTeeFunction.NumPoints:=tmpNumPoints;
tmpLineSeries:=TFastLineSeries.Create(self);
tmpLineSeries.SetFunction(tmpTeeFunction);
Chart1.AddSeries(tmpLineSeries);
Chart1.LeftAxis.Title.Caption:='Ux (x)';
Chart1.BottomAxis.Title.Caption:='x/d';
end;
22: begin
    Form1.InitialConditions;
    VizableChartWithMaxSpeed(10000);
    MinMaxValues(22);
    tmpTeeFunction:=TCustomTeeFunction.Create(Self);
    tmpTeeFunction.OnCalculate:=tmpTeeFunctionCalculateUxx;
    tmpTeeFunction.StartX:=tmpXValuesMin;
    tmpPeriod:=1/1000;
    tmpNumPoints:=Round((tmpXValuesMax-tmpXValuesMin)/tmpPeriod);
    tmpTeeFunction.Period:=tmpPeriod;
    tmpTeeFunction.NumPoints:=tmpNumPoints;
    tmpLineSeries:=TFastLineSeries.Create(self);
    tmpLineSeries.SetFunction(tmpTeeFunction);
    Chart1.AddSeries(tmpLineSeries);
    Chart1.LeftAxis.Title.Caption:='Uxx (x)';
    Chart1.BottomAxis.Title.Caption:='x/d';
end;
23: begin
    VizableChartWithMaxSpeed(10000);
    CommonInEnLossSeries;
    CommonInSeries(23);
    LabWait.Caption:='Подождите...';
    Chart1.LeftAxis.Title.Caption:='Ср. кв. флуктуаций поперечной энергии';
    if AdvOptF.CheckBoxAngstrom.Checked=False then
        Chart1.BottomAxis.Title.Caption:='ray';
    if AdvOptF.CheckBoxAngstrom.Checked=True then
        Chart1.BottomAxis.Title.Caption:='глубина, Анг.';
    Gauge1.MaxValue:=Nm;
    if AdvOptF.CheckBoxAngstrom.Checked then
        begin
            norm_koef:=d_small/epsilon;
            SetLength(tmp_t_arr,Nmax+1);
            for i:=0 to Nmax do
                tmp_t_arr[i]:=Trajects.Traject[0].t_arr[i]*norm_koef;
            for i:=0 to Nm-1 do
                begin

```



```

tmpLineSeries:=TFastLineSeries.Create(self);
with tmpLineSeries.XValues do
  begin
    Value:=TChartValues(tmp_t_arr);
    Count:=Nmax+1;
    Modified:=True;
  end;
with tmpLineSeries.YValues do
  begin
    Value:=TChartValues(Trajects.Traject[i].f_arr);
    Count:=Nmax+1;
    Modified:=True;
  end;
Chart1.AddSeries(tmpLineSeries);
tmpLineSeries.Color := clBlack;
Gauge1.Progress:=i+1;
end;
end
else
  begin
    norm_koef:=1;
    for i:=0 to Nm-1 do
      begin
        tmpLineSeries:=TFastLineSeries.Create(self);
        with tmpLineSeries.XValues do
          begin
            Value:=TChartValues(Trajects.Traject[i].t_arr);
            Count:=Nmax+1;
            Modified:=True;
          end;
        with tmpLineSeries.YValues do
          begin
            Value:=TChartValues(Trajects.Traject[i].f_arr);
            Count:=Nmax+1;
            Modified:=True;
          end;
        Chart1.AddSeries(tmpLineSeries);
        tmpLineSeries.Color := clBlack;
        Gauge1.Progress:=i+1;
      end;
    end;
  end;
end;
end;

```

```

Chart1.Repaint;
Chart1.AutoRepaint := True;
Gauge1.Progress := Gauge1.MaxValue;
LabWait.Caption := "";
end;

class procedure TPlotF.ChangeLanguage(ALanguage:Integer);
begin
  Case ALanguage of
    0: TeeSetRussian;
  else
    TeeSetEnglish;
  end;
end;

procedure TPlotF.ChooseLangChange(Sender: TObject);
begin
  ChangeLanguage(ChooseLang.ItemIndex);
end;

procedure TPlotF.FormShow(Sender: TObject);
begin
  ChooseLang.ItemIndex:=0;
  ChooseLangChange(Self);
  ChartEditor1.TreeView:=TeeReadBoolOption(TeeMsg_TreeMode,True);
end;

procedure TPlotF.AdvOptBtnClick(Sender: TObject);
begin
  AdvOptF.Show;
end;

procedure TPlotF.FormClose(Sender: TObject; var Action: TCloseAction);
var
  i,tmpSeriesCount: integer;
begin
  tmpSeriesCount:=Chart1.SeriesCount-1;
  for i:=0 to tmpSeriesCount do
    Chart1.Series[0].Free;
  end;
end;

procedure TPlotF.LimitFShow(Sender: TObject);
begin

```

```

LimitF.Show;
end;

procedure TPlotF.SetLimit(LimV,LimX,LimColor,LimViz,LimHistog,LimEndPoint: boolean);
var
Nm,tmpSeriesCount,EndValue,i: integer;
v1,v2,x1,x2,razresh: double;
ArrLim: array of byte;
procedure ColorLim(i:integer; b: byte);
begin
  case b of
    0: begin
      Chart1.Series[i].Color := clRed;
    end;
    1: begin
      Chart1.Series[i].Color := clBlack;
    end;
  end;
end;
end;
procedure VizLim(i:integer; b: byte);
begin
  case b of
    0: begin
      Chart1.Series[i].Visible := True;
    end;
    1: begin
      Chart1.Series[i].Visible := False;
    end;
  end;
end;
end;
begin
  tmpSeriesCount:=Chart1.SeriesCount;
  Nm:=Trajects.CntOfTraject;
  if LimHistog = False then
    if tmpSeriesCount<>Nm then
      begin
        MessageDlg('Ограничение не установлено. Не совпадение
данных.',mtInformation,[mbOk], 0);
        Exit;
      end;
    if (LimColor = False) and (LimViz = False) and (LimHistog = False) then
      begin

```

```

    MessageDlg('Ограничение не установлено. Не выбран тип ограниче-
ния.', mtInformation, [mbOk], 0);
    Exit;
end;
if LimitOnSetNoFirst then OffLimit;
if LimEndPoint then
    EndValue := Trajects.Traject[0].Nmax
    else EndValue := LimitF.SpinEditPoint.Value;
SetLength(ArrLim, Nm);
if LimV then
    begin
        v1:=StrToFloat(LimitF.LimitVst.Text);
        v2:=StrToFloat(LimitF.LimitVend.Text);
    end;
if LimX then
    begin
        x1:=StrToFloat(LimitF.LimitXst.Text);
        x2:=StrToFloat(LimitF.LimitXend.Text);
    end;
if (LimV) then
    for i:=0 to Nm-1 do
        begin
            with Trajects do
                begin
                    if (Traject[i].v_arr[EndValue]<v1) or (Traject[i].v_arr[EndValue]>v2) then
                        ArrLim[i]:=1;
                    end;
                end;
        end;
if (LimX) then
    for i:=0 to Nm-1 do
        begin
            with Trajects do
                begin
                    if (Traject[i].x_arr[EndValue]<x1) or (Traject[i].x_arr[EndValue]>x2) then
                        ArrLim[i]:=1;
                    end;
                end;
        end;
if (LimX) or (LimV) then
    begin
        Chart1.AutoRepaint:=False;
        if (PlotHistogramMain) and (LimHisto) then
            begin
                razresh := StrToFloat(AdvOptF.EdResolution.Text);

```

```

if (PlotHistogramSecond) and (Chart1.SeriesCount = 2) then
  begin
    Chart1.Series[1].Free;
  end;
  HistogramVX(1, Nm, False, razresh, ArrLim, clRed);
  PlotHistogramSecond := True;
end
else
  begin
    if LimColor then
      for i:=0 to Nm-1 do
        ColorLim(i, ArrLim[i]);
      if LimViz then
        for i:=0 to Nm-1 do
          VizLim(i, ArrLim[i]);
        end;
      Chart1.Repaint;
      Chart1.AutoRepaint:=True;
      Chart1.Refresh;
    end;
    Finalize(ArrLim);
    LimitOnSetNoFirst := True;
  end;

procedure TPlotF.OffLimit;
var
  i,tmpSeriesCount: integer;
begin
  tmpSeriesCount:=Chart1.SeriesCount-1;
  Chart1.AutoRepaint:=False;
  if PlotHistogramSecond then
    Chart1.Series[1].Free
  else
    for i:=0 to tmpSeriesCount do
      begin
        with Chart1.Series[i] do
          begin
            Color := clBlack;
            Visible := True;
          end;
        end;
      end;
    Chart1.Repaint;
    Chart1.AutoRepaint:=True;

```

```
Chart1.Refresh;  
end;
```

```
procedure TPlotF.SpinEdit1Change(Sender: TObject);  
var  
k:integer;  
begin  
k:=SpinEdit1.Value;  
Chart1.LeftAxis.LabelsFont.Size:=k;  
Chart1.BottomAxis.LabelsFont.Size:=k;  
end;
```

```
procedure TPlotF.CheckBox10Click(Sender: TObject);  
begin  
if CheckBox10.Checked=True then Chart1.BottomAxis.Inverted:=True  
else Chart1.BottomAxis.Inverted:=False;  
end;
```

```
procedure TPlotF.IncrementOYClick(Sender: TObject);  
begin  
with Sender as TCheckBox do  
begin  
Checked:=Checked;  
if IncrementOY.Checked=True then Chart1.LeftAxis.Increment:=StrToFloat(Edit12.Text)  
else Chart1.LeftAxis.Increment:=0;  
end;  
end;
```

```
procedure TPlotF.IncrementOXClick(Sender: TObject);  
begin  
with Sender as TCheckBox do  
begin  
Checked:=Checked;  
if IncrementOX.Checked=True then Chart1.BottomAxis.Increment:=StrToFloat(Edit13.Text)  
else Chart1.BottomAxis.Increment:=0;  
end;  
end;
```

```
procedure TPlotF.CheckBox1Click(Sender: TObject);  
begin  
with Sender as TCheckBox do  
begin  
Checked := Checked;
```

```

    AdvOptF.GroupBox4.Enabled := not Checked;
    AdvOptF.LabeledEdit10.Enabled :=not Checked;
    AdvOptF.LabeledEdit11.Enabled := not Checked;
    AdvOptF.RadioButton5.Enabled :=not Checked;
    AdvOptF.RadioButton6.Enabled :=not Checked;
  end;
end;

procedure TPlotF.RadioButton1Click(Sender: TObject);
begin
  AllDataAveraging := True;
end;

procedure TPlotF.RadioButton2Click(Sender: TObject);
begin
  AllDataAveraging := False;
end;

procedure TPlotF.Button2Click(Sender: TObject);
begin
  ModelDechF.Show;
end;

procedure TPlotF.TabSheet3Show(Sender: TObject);
begin
  ChartStat;
end;

procedure TPlotF.ButThClassicClick(Sender: TObject);
var
  t : TChartThemeClass;
  tmp : TChartTheme;
begin
  t:=ChartThemes[2];
  tmp:=t.Create(Chart1);
  tmp.Apply;
  tmp.Free;
  Chart1.Walls.Visible := False;
  Chart1.Border.Visible := False;
  Chart1.LeftAxis.Grid.SmallDots := True;
  Chart1.BottomAxis.Grid.SmallDots := True;
end;

```

```

procedure TPlotF.ButThXPClick(Sender: TObject);
var
  t : TChartThemeClass;
  tmp : TChartTheme;
begin
  t:=ChartThemes[5];
  tmp:=t.Create(Char1);
  tmp.Apply;
  tmp.Free;
  Chart1.Border.Width := 2;
  Chart1.Walls.Visible := True;
  Chart1.Border.Visible := True;
  Chart1.Walls.Back.Gradient.Balance := 1;
  Chart1.Walls.Back.Gradient.Direction := gdFromTopLeft;
  Chart1.Walls.Bottom.Color := clWhite;
end;

```

```

procedure TPlotF.Button4Click(Sender: TObject);
begin
with Chart1 do
  begin
    BottomAxis.Title.Caption := 'Глубина, см';
    BottomAxis.Automatic := True;
  end
end;

```

```

procedure TPlotF.ButChangeChartClick(Sender: TObject);
begin
with Chart1.LeftAxis do
  begin
    Axis.Width := 2;
    LabelsFont.Name := 'Arial';
    LabelsFont.Style := [fsBold];
    LabelsFont.Size := 11;
    Title.Font.Name := 'Arial';
    Title.Font.Style := [fsBold];
    Title.Font.Size := 12;
    Grid.Visible := False;
    Increment := 200;
  end;
with Chart1.BottomAxis do
  begin
    Axis.Width := 2;

```



```
LabelsFont.Name := 'Arial';
LabelsFont.Style := [fsBold];
LabelsFont.Size := 11;
Title.Font.Name := 'Arial';
Title.Font.Style := [fsBold];
Title.Font.Size := 12;
Grid.Visible := False;
Increment := 1;
end;
end;

end.
```

unit Constants;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Buttons, ExtDlgs, TeEngine, Series, ExtCtrls,
TeeProcs, Math;

type

DbArr=array of double;
PDbArr=^DbArr;
TFuncdouble=**function**(x:double):double;
TFunc2pDouble=**function**(x,y:double):double;

const

e2=14.4;
a0=0.529;
hc=1978.5;
mec2=511000;
alpha:array [1..3] of double=(0.35,0.55,0.10);
beta:array [1..3] of double=(0.3,1.2,6.0);

var

Phi_kl,Theta_kl,Pi4Z1Z2e2d2,a_st,sigma_st,sigma_st2,Vmax,Umin,epsilon,
Curv,DeltaTheta:double;
E,LnuclBor,LnuclD,Lel,Lel2,Lel4,LelD,LnuclRadiation,A,B,v2_na_c2,m0c2,M1:double;
k,l,ntr,Z1,nxRange,z_small,tipV:integer;
XY3x: array of array of double;
JK,nuzlov,Ploskost:integer;
kt,cDDT,Ib,hwp,dIE,rho_07,a1,a2,Pi4Z2e2,Pi2a0e2Z1: double;
glubina:real;
tip_ploskost:string;
Xvost, Vvost: array of array of double;
ax,ay,az:double;
x_stj:array [1..8] of double;
y_stj:array [1..8] of double;
z_stj:array [1..8] of double;
alphaVDT:array [1..4] of double;
betaVDT:array [1..4] of double;
alphaDDT:array [1..4] of double;
betaDDT:array [1..4] of double;
gamev,OverBarrierOff,BorOff,SrKvFlEp,VMoilerOff,Z2,TrDataFormat:integer;

EnergyLossVariant4Off:boolean;
 porog_min_eV2,porog_max_eV2,shag_A,min_shag_A:double;
 a_small,d_small,sigma,M2:double;
 Crystal:string;
 VxU, VklU, VxUx, VklUx, VxUxx, VklUxx, Fgx2exp,
 expSgx2, Pi2nx: array of double;
 Pi2, Pi2ax, d_small_3, PhiEpsilon: double;
 MoilerKoef: array[1..3] of double;
 xdax, kyday_lzdaz: array [1..8] of double;
 Auto_Solving, Auto_Save, N_Point_Save, Last_Layer_Save, Last_Layer_ReSave: boolean;
 Auto_Solving_**with**_step, N_Point_Save_Value_Str, DataBase_Name: string;
 N_Point_Save_Value_Int: integer;
 tau_As_Step_Saving: double;
 cnt_for_Saving_if_N_Point_True, cnt_for_Saving_if_N_Point_True_on_one_step_tau: integer;
 step_on_tau: double;
 Auto_Solving_Exit, Three_point_bending: boolean;
 sublayer_on_formoptions, normalization_on_formoptions: boolean;
 sublayer_on_formoptions_Value: integer;
 normalization_on_formoptions_Value: double;
 deltaE: array of array of double;
 glub,Z1_Znak: integer;
 OutPathForData, OutPathForDataBack, PathForSpecLoad: string;
 ModelDechanneling: integer;
 Potential_Axil_Plane, AllDataAveraging, ScatteringWithY: boolean;
 StartDistributionOnVXSet: boolean;
 LoadWithParameters: boolean;
 DirMyTableName, MyTableName, MyTableNameedb, MyTableNameedl: string;
 TypeMyFile: integer;

function gx(nx:integer):double;
function gx2(nx:integer):double;
function g2_kl(nx:integer):double;
function V(g2:double;tt:integer):double;
function U(x_st:double):double;
function U_x(x_st:double):double;
function U_xx(x_st:double):double;
 Procedure InitConstants;
 Procedure InitCrystalConstants;
function Usp(x_st,T_st:double):double;
function Usp_x(x_st,T_st:double):double;
function F(g3:double):double;
function rho_nucl(x_st:double):double;
function rho_el(x_st:double):double;

```
function Diffuzia(x_st:double):double;
procedure DirAndNameMyDB(tmpOpenFileNameDB: string);
```

implementation

```
function gx(nx:integer):double;
begin
  gx:=Pi2ax*nx;
end;
```

```
function gx2(nx:integer):double;
begin
  gx2:=Sqr(Pi2ax*nx);
end;
```

```
function g2_kl(nx:integer):double;
begin
  g2_kl:=Sqr(Pi2)*(Sqr(nx/ax)+Sqr(k/ay)+Sqr(l/az));
end;
```

```
function V(g2:double;tt:integer):double;
var
  sum,V2:double;
  i:integer;
begin
  sum:=0;
  if tt=0 then
    begin
      if VMoilerOff=0 then
        begin
          for i:=1 to 3 do sum:=sum+alpha[i]/(MoilerKoeff[i]+g2);
          V:=Pi4Z1Z2e2d2*sum;
        end
      else if VMoilerOff=1 then
        begin
          for i:=1 to 4 do sum:=sum+alphaVDT[i]*exp(-betaVDT[i]*g2/(Sqr(2*Pi2)));
          V:=Pi2a0e2Z1*sum;
        end;
      end
    else if tt=1 then
      begin
        V2:=Pi4Z2e2/(Sqr(beta[2]/a2)+g2);
        V:=(Z1-(Z1-z_small)*Sqr(beta[2]/a1))/(Sqr(beta[2]/a1)+g2))*V2;
      end
    end
```

```

    end;
end;

function U(x_st:double):double;
var
    sum:double;
    nx,j,Nmax,n:integer;
begin
    nMax:=2*nRange;
    sum:=0;
    for j:=1 to 8 do
        for n:=0 to nMax do
            sum:=sum+VxU[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
        U:=Z1_Znak*sum/d_small_3-Umin;
    end;
end;

```

```

function U_x(x_st:double):double;
var
    sum:double;
    nx,j,Nmax,n:integer;
begin
    nMax:=2*nRange;
    sum:=0;
    for j:=1 to 8 do
        for n:=0 to nMax do
            sum:=sum-VxUx[n]*sin(Pi2nx[n]*(x_st-xdax[j]));
        U_x:=Z1_Znak*sum/d_small_3;
    end;
end;

```

```

function U_xx(x_st:double):double;
var
    sum: double;
    n, nMin, nMax, j: integer;
begin
    nMin:=nRange+1;
    nMax:=2*nRange;
    sum:=0;
    for j:=1 to 8 do
        for n:=nMin to nMax do
            sum:=sum-VxUxx[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
        sum:=2*sum;
        U_xx:=Z1_Znak*sum/d_small_3;
    end;
end;

```

Procedure InitCrystalConstants;

begin

if Crystal='Ge' **then**

begin

Z2:=32;

sigma:=0.085;

a_small:=0.148;

d_small:=5.657;

M2:=72.59;

alphaVDT[1]:=2.4467; alphaVDT[2]:=2.7015; alphaVDT[3]:=1.6157; alphaVDT[4]:=0.6009;

betaVDT[1]:=55.893; betaVDT[2]:=14.393; betaVDT[3]:=2.4461; betaVDT[4]:=0.3415;

alphaDDT[1]:=10.0816; alphaDDT[2]:=6.3747; alphaDDT[3]:=3.7068; alphaDDT[4]:=3.6830;

betaDDT[1]:=2.8509; betaDDT[2]:=0.2516; betaDDT[3]:=11.4468; betaDDT[4]:=54.7625;

cDDT:=2.1313;

end

else if Crystal='Si' **then**

begin

Z2:=14;

sigma:=0.075;

a_small:=0.194;

d_small:=5.431;

M2:=28.09;

alphaVDT[1]:=2.1293; alphaVDT[2]:=2.5333; alphaVDT[3]:=0.8349; alphaVDT[4]:=0.3216;

betaVDT[1]:=57.7748; betaVDT[2]:=16.4756; betaVDT[3]:=2.8796; betaVDT[4]:=0.3860;

alphaDDT[1]:=6.2915; alphaDDT[2]:=3.0353; alphaDDT[3]:=1.9891; alphaDDT[4]:=1.5410;

betaDDT[1]:=2.4386; betaDDT[2]:=32.3337; betaDDT[3]:=0.6785; betaDDT[4]:=81.6937;

cDDT:=1.1407;

end

else if Crystal='C' **then**

begin

Z2:=6;

sigma:=0.04;

a_small:=0.258;

d_small:=3.567;

M2:=12.01;

alphaVDT[1]:=0.7307; alphaVDT[2]:=1.1951; alphaVDT[3]:=0.4563; al-

phaVDT[4]:=0.1247;

betaVDT[1]:=36.9951; betaVDT[2]:=11.2966; betaVDT[3]:=2.8139; betaVDT[4]:=0.3456;

alphaDDT[1]:=2.31; alphaDDT[2]:=1.02; alphaDDT[3]:=1.5886; alphaDDT[4]:=0.865;

betaDDT[1]:=20.8439; betaDDT[2]:=10.2075; betaDDT[3]:=0.5687; betaDDT[4]:=51.6512;

cDDT:=0.2156;

end;

end;

Procedure InitConstants;

var

ii,nx:integer;

begin

Theta_kl:=-ArcTan(l/k*ay/az);

Phi_kl:=k*d_small/ay*sin(Theta_kl+DeltaTheta)+
l*d_small/az*cos(Theta_kl+DeltaTheta);

if tipV=1 **then**

begin

a1:=0.88534*a0/Power((Z1-z_small),1/3);

a2:=0.88534*a0/Power(Z2,1/3);

end;

Pi4Z1Z2e2d2:=4*Pi*Z1*Z2*e2;

Pi4Z2e2:=4*Pi*Z2*e2;

Pi2a0e2Z1:=2*a0*e2*Z1*Pi;

a_st:=a_small;

sigma_st:=sigma;

sigma_st2:=Sqr(sigma_st)/2;

MoilerKoeff[1]:=Sqr(beta[1]/a_st);

MoilerKoeff[2]:=Sqr(beta[2]/a_st);

MoilerKoeff[3]:=Sqr(beta[3]/a_st);

for ii:=1 **to** 8 **do**

begin

xdax[ii]:=x_stj[ii]*d_small/ax;

kyday_lzdaz[ii]:=k*y_stj[ii]*d_small/ay+l*z_stj[ii]*d_small/az;

end;

Pi2:=2*Pi;

Pi2ax:=Pi2/ax;

Finalize(Pi2nx); SetLength(Pi2nx, 2*n xrange+1);

Finalize(expSgx2); SetLength(expSgx2, 2*n xrange+1);

Finalize(VxU); SetLength(VxU, 2*n xrange+1);

Finalize(VklU); SetLength(VklU, 2*n xrange+1);

Finalize(VxUx); SetLength(VxUx, 2*n xrange+1);

Finalize(VklUx); SetLength(VklUx, 2*n xrange+1);

Finalize(VxUxx); SetLength(VxUxx, 2*n xrange+1);

Finalize(VklUxx); SetLength(VklUxx, 2*n xrange+1);

Finalize(Fgx2exp); SetLength(Fgx2exp, 2*n xrange+1);

for ii:=0 **to** (2*n xrange) **do**

begin

nx:=ii-n xrange;

expSgx2[ii]:=exp(-sigma_st2*gx2(nx));

```

Pi2nx[ii]:=Pi2*nx;
VxU[ii]:=V(gx2(nx),tipV)*expSgx2[ii];
VklU[ii]:=V(g2_kl(nx),tipV)*exp(-sigma_st2*g2_kl(nx));
VxUx[ii]:=Pi2nx[ii]*VxU[ii];
VklUx[ii]:=Pi2nx[ii]*VklU[ii];
VxUxx[ii]:=Sqr(Pi2nx[ii])*VxU[ii];
VklUxx[ii]:=Sqr(Pi2nx[ii])*VklU[ii];
Fgx2exp[ii]:=F(gx2(nx))*expSgx2[ii];
end;
d_small_3:=d_small*d_small*d_small;
if tip_ploskost='100' then
begin
Umin:=0; Umin:=U(0.125)
end
else if tip_ploskost='110' then
begin
Umin:=0; Umin:=U(0.25)
end
else if tip_ploskost='111' then
begin
Umin:=0; Umin:=U(0.375);
end;
Vmax:=U(0);
Ib:=11.5*Z2;
rho_07:=8*Z2/(d_small*d_small*d_small);
v2_na_c2:=1-1/Sqr(E/m0c2+1);
hwp:=Sqrt(4*Pi*Z2*8/(d_small*d_small*d_small*mec2)*e2*Sqr(hc));
dlE:=2*ln(hwp*sqrt(v2_na_c2)*sqrt(1/(1-v2_na_c2)))/Ib-1;
LnuclBor:=ln(1.29*a_small*M2*E/(Z1*Z2*(M1+M2)*e2));
LnuclRadiation:=2*ln(183/Power(Z2,1/3));
if BorOff=1 then LnuclD:=LnuclRadiation
else if BorOff=0 then LnuclD:=LnuclBor;
Lel4:=ln(2*mec2*v2_na_c2/(Ib*(1-v2_na_c2)))-v2_na_c2-dlE/2;
Lel2:=ln(2*mec2*v2_na_c2/Ib);
if EnergyLossVariant4Off=True then Lel:=Lel2 else Lel:=Lel4;
LelD:=ln(2*mec2*v2_na_c2/(Ib*(1-v2_na_c2)))-v2_na_c2;
A:=2*Pi*Sqr(Z1)*Sqr(Z2)*Sqr(e2)*LnuclD;
B:=2*Pi*Sqr(Z1)*Sqr(e2)*LelD;
end;

function Usp(x_st,T_st:double):double;
var
sum:double;

```



```

nx,n,j:integer;
begin
sum:=0;
for j:=1 to 8 do
  for nx:=(-nxRange) to nxRange do
    begin
      n:=nx+nxRange;
      sum:=sum+VxU[n]*cos(Pi2nx[n]*(x_st-xdax[j]))
      +VklU[n]*cos(Pi2*(nx*(x_st-xdax[j])+PhiEpsilon*T_st-kyday_lzdaz[j]));
    end;
  Usp:=Z1_Znak*sum/d_small_3 - Umin;
end;

function Usp_x(x_st,T_st:double):double;
var
sum:double;
nx,j,n:integer;
begin
sum:=0;
for j:=1 to 8 do
  for nx:=(-nxRange) to nxRange do
    begin
      n:=nx+nxRange;
      sum:=sum-VxUx[n]*sin(Pi2nx[n]*(x_st-xdax[j]))
      -VklUx[n]*sin(Pi2*(nx*(x_st-xdax[j])+PhiEpsilon*T_st-kyday_lzdaz[j]));
    end;
  Usp_x:=Z1_Znak*sum/d_small_3;
end;

function F(g3:double):double;
var
sum:double;
i:integer;
begin
sum:=0;
if (VMoilerOff=1) then
  begin
    for i:=1 to 4 do sum:=sum+alphaDDT[i]*exp(-betaDDT[i]*g3/(157.91367));
    F:=sum+cDDT;
  end
else
  begin
    for i:=1 to 3 do

```

```

    sum:=sum
      +(alpha[i]*MoilerKoeff[i])/(g3+MoilerKoeff[i]);
    F:=Z2*sum;
  end;
end;

function rho_nucl(x_st:double):double;
var
  sum:double;
  nx,j,n,nMin,nMax:integer;
begin
  nMin:=nxRange+1;
  nMax:=2*nxRange;
  sum:=0;
  for j:=1 to 8 do
    for n:=nMin to nMax do
      sum:=sum+expSgx2[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
    sum:=2*sum+8;
    rho_nucl:=sum/d_small_3;
  end;

function rho_el(x_st:double):double;
var
  sum:double;
  nx,j,nMin,nMax,n:integer;
begin
  nMin:=nxRange+1;
  nMax:=2*nxRange;
  sum:=0;
  for j:=1 to 8 do
    for n:=nMin to nMax do
      sum:=sum+Fgx2exp[n]*cos(Pi2nx[n]*(x_st-xdax[j]));
    sum:=2*sum;
    sum:=sum+Fgx2exp[nxRange]*8;
    rho_el:=sum/d_small_3;
  end;

function Diffuzia(x_st:double):double;
begin
  Diffuzia:=A*rho_nucl(x_st)+B*rho_el(x_st);
end;

procedure DirAndNameMyDB(tmpOpenFileNameDB: string);

```

```

var
tmpDirServer, tmpTableName, tmpDirTableName, tmpFullTableName, tmpTableNamedb,
tmpTableNamedl, Reserved_word, Reserved_word_2, Reserved_word_3: string;
Length_Reserved_word, Length_TableName, Position_Reserved_word, Position_Number_Particle:
integer;
Length_Reserved_word_2, Position_Reserved_word_2, Position_Number_Particle_2: integer;
Length_Reserved_word_3, Position_Reserved_word_3, Position_Number_Particle_3: integer;
tmpf, tmpd, i: integer;
begin
    tmpFullTableName:=tmpOpenFileNameDB;
    tmpDirTableName:=ExtractFilePath(tmpFullTableName);
    tmpTableName:=ExtractFileName(tmpFullTableName);
    tmpTableNamedb := tmpTableName;
    tmpTableNamedl := tmpTableName;
    Reserved_word := '_particles';
    Reserved_word_2 := '_lastlayer_';
    Reserved_word_3 := '_layer';
    Length_Reserved_word := Length(Reserved_word);
    Length_Reserved_word_2 := Length(Reserved_word_2);
    Length_Reserved_word_3 := Length(Reserved_word_3);
    Length_TableName := Length(tmpTableName);
    Position_Reserved_word := Pos(Reserved_word, tmpTableName);
    Position_Reserved_word_2 := Pos(Reserved_word_2, tmpTableName);
    Position_Reserved_word_3 := Pos(Reserved_word_3, tmpTableName);
    if (Position_Reserved_word<>0) then
        begin
            Position_Number_Particle := Length_TableName - (Position_Reserved_word-
1+Length_Reserved_word);
            Delete(tmpTableNamedl, (Length_TableName-Position_Number_Particle+1), Posi-
tion_Number_Particle);
            Delete(tmpTableName, Position_Reserved_word, (Length_Reserved_word + Posi-
tion_Number_Particle));
            Delete(tmpTableNamedb, (Length_TableName - 3), 4);
            tmpTableNamedl := tmpTableNamedl + '_dl';
            TypeMyFile := 1;
        end
    else if (Position_Reserved_word_2<>0) then
        begin
            Position_Number_Particle_2 := Length_TableName - (Position_Reserved_word_2-
1+Length_Reserved_word_2);
            Delete(tmpTableNamedl, (Length_TableName-Position_Number_Particle_2+1), Posi-
tion_Number_Particle_2);

```

```

Delete(tmpTableName, Position_Reserved_word_2, (Length_Reserved_word_2 + Position_Number_Particle_2));
tmpTableNamedb := tmpTableNamedl + 'db';
tmpTableNamedl := tmpTableNamedl + 'dl';
TypeMyFile := 2;
end
else if (Position_Reserved_word_3>0) then
  begin
    Position_Number_Particle_3 := Length_TableName - (Position_Reserved_word_3-1+Length_Reserved_word_3);
    Delete(tmpTableName, Position_Reserved_word_3, (Length_Reserved_word_3 + Position_Number_Particle_3));
    Delete(tmpTableNamedb, (Length_TableName - 5), 6);
    tmpTableNamedl := tmpTableNamedb;
    tmpTableNamedb := tmpTableNamedb + 'db';
    tmpTableNamedl := tmpTableNamedl + 'dl';
    TypeMyFile := 3;
  end
else
  begin
    Delete(tmpTableName, (Length(tmpTableName)-5), 6);
    tmpTableNamedb:=tmpTableName + 'db';
    tmpTableNamedl:=tmpTableName + 'dl';
    TypeMyFile := 4;
  end;
tmpf := 0;
tmpDirServer := tmpDirTablename;
tmpd := Length(tmpDirServer);
Delete(tmpDirServer, tmpd, 1);
tmpd := tmpd-1;
for i := 0 to tmpd do
  if tmpDirServer[tmpd-i]='\ ' then
    begin
      tmpDirServer[tmpd-i]:='/';
      if tmpf = 0 then tmpf := i;
    end;
Delete(tmpDirServer, tmpd-tmpf, tmpf+1);
DirMyTableName := tmpDirServer;
MyTableName := tmpTableName;
MyTableNamedb := tmpTableNamedb;
MyTableNamedl := tmpTableNamedl;
end;
end.

```

unit Solver;

interface

uses

Constants, Math;

type

db3parfn=**function**(x,u,v:double):double;

OneTrajectory=object

t_arr,v_arr,x_arr:DbArr;

t_Start,t_Depth:double;

v_Start,x_Start:double;

sloi:double;

f_arr:DbArr;

v_y: double;

lRange,rRange,lrPeriod:double;

PartDone:double;

OverBarrier:boolean;

CriticalPoint:integer;

NodeCount:integer;

k,q,s:array[1..4] of double;

Solved,CanSolve:boolean;

tau,tau_na_2:double;

n,Nmax:integer;

constructor Init(start,depth,lRng,rRng:double;NodeCnt:integer;vst,xst,ssl:double);

destructor Done;

function RightForV(t,v,x:double):double;

function RightForX(t,v,x:double):double;

function RightForV2(t,v,x:double):double;

function RightForF(v,x:double):double;

function RightForFt(t,v,x:double):double;

function RightForFtWithY(t,v,x:double):double;

function E_transversal(v,x:double):double;

procedure SolveForXV;

procedure SolveForF;

procedure MyGameV;

Procedure Solve;

Procedure SvTable2Dat(fname:string);

Procedure SaveV(fname:string);

Procedure SaveX(fname:string);

end;

implementation

constructor OneTrajectory.Init(start,depth,lRng,rRng:double;NodeCnt:integer;vst,xst,ssl:double);

begin

```
t_Start:=start;
t_Depth:=depth;
sloi:=ssl;
NodeCount:=NodeCnt;
v_Start:=vst;
x_Start:=xst;
lRange:=lRng;
rRange:=rRng;
lRPeriod:=rRng-lRng;
Solved:=false;
SetLength(t_arr,NodeCount);
SetLength(x_arr,NodeCount);
SetLength(v_arr,NodeCount);
PartDone:=0.0;
Nmax:=NodeCount-1;
tau:=(t_Depth-t_Start)/Nmax;
tau_na_2:=tau/2;
v_arr[0]:=v_Start;
x_arr[0]:=x_Start;
t_arr[0]:=t_Start;
SetLength(f_arr,NodeCount);
f_arr[0]:=0;
v_y:=0;
n:=0;
CanSolve:=true;
```

end;

destructor OneTrajectory.Done;

begin

```
Finalize(t_arr);
Finalize(x_arr);
Finalize(v_arr);
Finalize(f_arr);
```

end;

function OneTrajectory.RightForV(t,v,x:double):double;

begin

```
RightForV:=-Z1_Znak*Usp_x(x,t)/Vmax+Curv+kt*t;
```

end;

function OneTrajectory.RightForX(t,v,x:double):double;

begin

RightForX:=v;

end;

function OneTrajectory.RightForV2(t,v,x:double):double;

begin

RightForV2 := -Z1_Znak*U_x(x)/Vmax+Curv+kt*t;

end;

function OneTrajectory.RightForFt(t,v,x:double):double;

begin

RightForFt:=epsilon*d_small*Sqr(v)*Diffuzia(x);

end;

function OneTrajectory.RightForF(v,x:double):double;

begin

RightForF:=epsilon*d_small*Sqr(v)*Diffuzia(x);

end;

function OneTrajectory.RightForFtWithY(t,v,x:double):double;

begin

RightForFtWithY:=epsilon*d_small*(Sqr(v)+Sqr(v_y))*Diffuzia(x);

end;

function OneTrajectory.E_transversal(v,x:double):double;

begin

E_transversal := Vmax*Sqr(v)/2+U(x);

end;

procedure OneTrajectory.SolveForXV;

begin

case Potential_Axil_Plane of

True:

begin

t_arr[n+1]:=t_arr[n]+tau;

k[1]:=RightForV(t_arr[n],v_arr[n],x_arr[n]);

q[1]:=RightForX(t_arr[n],v_arr[n],x_arr[n]);

k[2]:=RightForV(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);

q[2]:=RightForX(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);

k[3]:=RightForV(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);

```

    q[3]:=RightForX(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);
    k[4]:=RightForV(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
    q[4]:=RightForX(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
    v_arr[n+1]:=v_arr[n]+tau*(k[1]+2*k[2]+2*k[3]+k[4])/6;
    x_arr[n+1]:=x_arr[n]+tau*(q[1]+2*q[2]+2*q[3]+q[4])/6;
end;
False:
begin
    t_arr[n+1]:=t_arr[n]+tau;
    k[1]:=RightForV2(t_arr[n],v_arr[n],x_arr[n]);
    q[1]:=RightForX(t_arr[n],v_arr[n],x_arr[n]);
    k[2]:=RightForV2(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);
    q[2]:=RightForX(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);
    k[3]:=RightForV2(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);
    q[3]:=RightForX(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);
    k[4]:=RightForV2(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
    q[4]:=RightForX(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
    v_arr[n+1]:=v_arr[n]+tau*(k[1]+2*k[2]+2*k[3]+k[4])/6;
    x_arr[n+1]:=x_arr[n]+tau*(q[1]+2*q[2]+2*q[3]+q[4])/6;
end;
end;
    if (x_arr[n+1] > rRange) then
        x_arr[n+1]:=x_arr[n+1]-lrPeriod
    else
        if (x_arr[n+1] < lRange) then
            x_arr[n+1]:=x_arr[n+1]+lrPeriod;
end;

procedure OneTrajectory.SolveForF;
var
    tmpf:double;
begin
    if SrKvFLEp=1 then
        case ScatteringWithY of
            True: f_arr[n+1] :=
f_arr[n]+tau_na_2*(RightForFtWithY(t_arr[n],v_arr[n],x_arr[n])+RightForFtWithY(t_arr[n+1],v_a
rr[n+1],x_arr[n+1]));
            False:
                f_arr[n+1] :=
f_arr[n]+tau_na_2*(RightForFt(t_arr[n],v_arr[n],x_arr[n])+RightForFt(t_arr[n+1],v_arr[n+1],x_arr[
n+1]));
        end
    else

```



```

case ScatteringWithY of
  True:
    begin
      s[1]:=RightForFtWithY(t_arr[n],v_arr[n],x_arr[n]);

s[2]:=RightForFtWithY(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);

s[3]:=RightForFtWithY(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);
      s[4]:=RightForFtWithY(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
      f_arr[n+1] := f_arr[n]+tau*(s[1]+2*s[2]+2*s[3]+s[4])/6;
    end;
  False:
    begin
      s[1]:=RightForFt(t_arr[n],v_arr[n],x_arr[n]);
      s[2]:=RightForFt(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[1],x_arr[n]+tau_na_2*q[1]);
      s[3]:=RightForFt(t_arr[n]+tau_na_2,v_arr[n]+tau_na_2*k[2],x_arr[n]+tau_na_2*q[2]);
      s[4]:=RightForFt(t_arr[n+1],v_arr[n]+tau*k[3],x_arr[n]+tau*q[3]);
      f_arr[n+1] := f_arr[n]+tau*(s[1]+2*s[2]+2*s[3]+s[4])/6;
    end;
  end;

procedure OneTrajectory.MyGameV;
var
p_random,Eps,tmpRadius,tmpFi:double;
znak:integer;
begin
case ScatteringWithY of
  True:
    begin
      Eps := E_transversal(v_arr[n+1], x_arr[n+1]);
      p_random := RandG(Eps, Sqrt(f_arr[n+1]));
      tmpRadius := Sqrt( Abs( 2/Vmax*(p_random - U(x_arr[n+1])) ) );
      tmpFi := 2*Pi*Random;
      v_arr[n+1] := tmpRadius*cos(tmpFi);
      v_y := tmpRadius*sin(tmpFi);
    end;
  False:
    begin
      Eps := E_transversal(v_arr[n+1],x_arr[n+1]);
      p_random:=RandG(Eps,Sqrt(f_arr[n+1]));
      znak:=Round(Power(-1,Random(100)));

```

```

    v_arr[n+1]:=v_arr[n+1]+znak*Abs(Sqrt(2*Abs(p_random-U(x_arr[n+1])))/Abs(Vmax))-
    Abs(v_arr[n+1]));
    end;
end;

end;

Procedure OneTrajectory.Solve;
var
Ep,epsilon_na_d:double;
begin
    Ep := E_transversal(v_arr[0],x_arr[0]);
    OverBarrier:=false;
    epsilon_na_d:=epsilon/d_small;
    Randomize;
    while (n <= (Nmax-1)) and CanSolve do
        begin
            SolveForXV;
            if (gamev=1) then
                begin
                    SolveForF;
                    if (t_arr[n+1]>min_shag_A*epsilon_na_d+sloi) then
                        if Overbarrier=false then
                            if (OverBarrierOff=0) or ((OverBarrierOff=1) and (Ep<Vmax)) then
                                begin
                                    if (t_arr[n+1]>=sloi+shag_A*epsilon_na_d) or (f_arr[n+1]>porog_max_eV2) then
                                        begin
                                            MyGameV;
                                            if (OverBarrierOff=1) then
                                                begin
                                                    Ep := E_transversal(v_arr[n+1],x_arr[n+1]);
                                                    if (Ep>Vmax) then
                                                        begin
                                                            CriticalPoint:=n+1;
                                                            OverBarrier:=true;
                                                        end
                                                    end
                                                begin
                                                    if (t_arr[n+1]>=sloi+shag_A*epsilon_na_d) then
                                                        begin
                                                            if (f_arr[n+1]>porog_max_eV2) then f_arr[n+1]:=0;
                                                        end
                                                    else

```

```

        if (f_arr[n+1]>porog_max_eV2) then
            begin
                f_arr[n+1]:=0;
            end;
        sloi:=t_arr[n+1];
    end;
end
else
    begin
        if (t_arr[n+1]>=sloi+shag_A*epsilon_na_d) then
            begin
                if (f_arr[n+1]>porog_max_eV2) then f_arr[n+1]:=0;
            end
            else
                if (f_arr[n+1]>porog_max_eV2) then
                    begin
                        f_arr[n+1]:=0;
                    end;
                sloi:=t_arr[n+1];
            end;
        end;
    end;
end
else
    begin
        CriticalPoint:=n+1;
        OverBarrier:=true;
    end;
end;
inc(n);
PartDone:=n/Nmax;
end;
if (n=Nmax) then begin
    Solved:=true;
end;
end;

```

Procedure OneTrajectory.SvTable2Dat(fname:string);

var

i:integer;

DestFile:TextFile;

begin

AssignFile(DestFile,fname);

ReWrite(DestFile);

```

for i:=0 to NodeCount-1 do
  WriteLn(DestFile,x_arr[i], ' ',v_arr[i], ' ',t_arr[i]);
  CloseFile(DestFile);
end;

```

```

Procedure OneTrajectory.SaveX(fname:string);

```

```

var
  i:integer;
  DestFile:TextFile;
begin
  AssignFile(DestFile,fname);
  ReWrite(DestFile);
  for i:=0 to NodeCount-1 do
    WriteLn(DestFile,x_arr[i]);
    CloseFile(DestFile);
  end;

```

```

Procedure OneTrajectory.SaveV(fname:string);

```

```

var
  i:integer;
  DestFile:TextFile;
begin
  AssignFile(DestFile,fname);
  ReWrite(DestFile);
  for i:=0 to NodeCount-1 do
    WriteLn(DestFile,v_arr[i]);
    CloseFile(DestFile);
  end;

```

```

end.

```

unit AdvOptToOurPlot;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, ComCtrls;

type

TAdvOptF = **class**(TForm)
 GroupBox3: TGroupBox;
 RadioButton3: TRadioButton;
 LabeledEdit1: TLabeledEdit;
 RadioButton4: TRadioButton;
 LabeledEdit8: TLabeledEdit;
 GroupBox4: TGroupBox;
 RadioButton5: TRadioButton;
 LabeledEdit10: TLabeledEdit;
 RadioButton6: TRadioButton;
 LabeledEdit11: TLabeledEdit;
 LabeledEdit5: TLabeledEdit;
 Label20: TLabel;
 Label19: TLabel;
 Button3: TButton;
 Button2: TButton;
 CheckBox7: TCheckBox;
 LabeledEdit9: TLabeledEdit;
 CheckBox8: TCheckBox;
 Edit11: TEdit;
 Label16: TLabel;
 Label15: TLabel;
 CheckBoxAngstrom: TCheckBox;
 CheckBox5: TCheckBox;
 CheckBox6: TCheckBox;
 LabeledEdit6: TLabeledEdit;
 LabeledEdit7: TLabeledEdit;
 LabeledEdit3: TLabeledEdit;
 LabeledEdit4: TLabeledEdit;
 CheckBoxDec: TCheckBox;
 LabelEx: TLabel;
 Label12: TLabel;
 Label13: TLabel;
 CheckBox4: TCheckBox;

```

Label14: TLabel;
Edit8: TEdit;
Edit9: TEdit;
EdResolution: TEdit;
Label1: TLabel;
GroupBox1: TGroupBox;
GroupBox2: TGroupBox;
GroupBox5: TGroupBox;
GroupBox6: TGroupBox;
procedure FormCreate(Sender: TObject);
procedure LabeledEdit1Click(Sender: TObject);
procedure LabeledEdit8Click(Sender: TObject);
procedure LabeledEdit10Click(Sender: TObject);
procedure LabeledEdit11Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    AdvOptF: TAdvOptF;

implementation

uses STetraject, Constants, OurPlotting;

{$R *.dfm}

procedure TAdvOptF.FormCreate(Sender: TObject);
begin
    AdvOptF.FormStyle:=fsStayOnTop;
end;

procedure TAdvOptF.LabeledEdit1Click(Sender: TObject);
begin
    RadioButton3.Checked:=True;
end;

procedure TAdvOptF.LabeledEdit8Click(Sender: TObject);
begin

```

```

    RadioButton4.Checked:=True;
end;

procedure TAdvOptF.LabeledEdit10Click(Sender: TObject);
begin
    RadioButton5.Checked:=True;
end;

procedure TAdvOptF.LabeledEdit11Click(Sender: TObject);
begin
    RadioButton6.Checked:=True;
end;

procedure TAdvOptF.Button2Click(Sender: TObject);
var
    Tmp:integer;
    Nmax:integer;
begin
    Nmax:=Trajects.Traject[0].Nmax;
    Tmp:=glub;
    Button3.Enabled:=True;
if Checkbox5.Checked=True then
        begin
            glub:=glub-Round((Nmax+1)/10);
        end
        else glub:=glub-Round(StrToInt(LabeledEdit5.Text));
if glub<0 then
        begin
            glub:=Tmp;
            Button2.Enabled:=False;
            Button3.Enabled:=True;
            MessageDlg('Запрашиваемая глибуна меньше нулевого значения',mtInformation,[mbOk],0)
        end
        else if glub=0 then
            begin
                Button2.Enabled:=False;
                Button3.Enabled:=True;
            end;
    PlotF.PlotGraphics(Self);
end;

procedure TAdvOptF.Button3Click(Sender: TObject);
var

```

```

Tmp:integer;
Nmax:integer;
begin
  Tmp:=glub;
  Nmax:=Trajects.Traject[0].Nmax;
  Button2.Enabled:=True;
  if Checkbox5.Checked=True then
    begin
      glub:=glub+Round((Nmax+1)/10);
    end
    else glub:=glub+Round(StrToInt(LabeledEdit5.Text));
  if glub>Nmax then
    begin
      glub:=Tmp;
      Button3.Enabled:=False;
      Button2.Enabled:=True;
      MessageDlg('Запрашиваемая глибуна больше рассчитаного',mtInformation,[mbOk],0);
    end
    else if glub=Nmax then
      begin
        Button3.Enabled:=False;
        Button2.Enabled:=True;
      end;
  PlotF.PlotGraphics(Self);
end;

end.

```


unit Limit;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, Spin;

type

```
TLimitF = class(TForm)
  GroupBox1: TGroupBox;
  LimitVst: TLabeledEdit;
  LimitVend: TLabeledEdit;
  GroupBox2: TGroupBox;
  LimitVCheck: TCheckBox;
  Memo1: TMemo;
  GroupBox3: TGroupBox;
  LimitXst: TLabeledEdit;
  LimitXend: TLabeledEdit;
  LimitXCheck: TCheckBox;
  CheckColor: TCheckBox;
  CheckNoVisible: TCheckBox;
  LimitStart: TButton;
  LimitEnd: TButton;
  CheckHistog: TCheckBox;
  Button1: TButton;
  Label1: TLabel;
  CheckEndPoint: TCheckBox;
  Label2: TLabel;
  SpinEditPoint: TSpinEdit;
  Label3: TLabel;
  procedure LimitEndClick(Sender: TObject);
  procedure LimitStartClick(Sender: TObject);
  procedure CheckHistogClick(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure CheckEndPointClick(Sender: TObject);
  procedure SpinEditPointChange(Sender: TObject);
  procedure FormShow(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

var

LimitF: TLimitF;

implementation

uses OurPlotting, STEtraject, Math, Constants;

{ \$R *.dfm }

procedure TLimitF.LimitEndClick(Sender: TObject);

begin

PlotF.OffLimit;

end;

procedure TLimitF.LimitStartClick(Sender: TObject);

begin

PlotF.SetLimit(LimitVCheck.Checked,LimitXCheck.Checked,CheckColor.Checked,
CheckNoVisible.Checked,CheckHistog.Checked,CheckEndPoint.Checked);

end;

procedure TLimitF.CheckHistogClick(Sender: TObject);

begin

CheckColor.Checked := False;

CheckNoVisible.Checked := False;

CheckColor.Enabled := False;

CheckNoVisible.Enabled := False;

end;

procedure TLimitF.Button1Click(Sender: TObject);

begin

LimitF.Close;

end;

procedure TLimitF.CheckEndPointClick(Sender: TObject);

begin

with Sender as TCheckBox **do**

begin

Checked := Checked;

if Checked **then** SpinEditPoint.Value := Trajects.Traject[0].Nmax;

end;

end;

```

procedure TLimitF.SpinEditPointChange(Sender: TObject);
begin
  if SpinEditPoint.Text <> " then
    begin
      Label3.Caption :=
        ' = ' +
        FloatToStr(RoundTo(Trajects.Traject[0].t_arr[SpinEditPoint.Value],-5))
        + ' ray = ' +
        FloatToStr(RoundTo(Trajects.Traject[0].t_arr[SpinEditPoint.Value]/epsilon*d_small,-2))
        + ' A';
      Label3.Width := 265;
      CheckEndPoint.Checked := False;
    end;
  end;

procedure TLimitF.FormShow(Sender: TObject);
begin
  if (SolveMyDataDone) or (LoadMyDataDone) then
    SpinEditPoint.MaxValue := Trajects.Traject[0].Nmax;
  end;

end.

```

unit ModelDechan;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, TeeProcs, TeeDraw3D;

type

TModelDechF = **class**(TForm)
 Draw3D1: TDraw3D;
 GroupBox6: TGroupBox;
 ChDechEn: TCheckBox;
 ChDechDistance: TCheckBox;
 GroupBox8: TGroupBox;
 RadioButton1: TRadioButton;
 RadioButton2: TRadioButton;
 procedure ChoiceModelDechanneling;
 procedure ChDechEnClick(Sender: TObject);
 procedure ChDechDistanceClick(Sender: TObject);
 procedure RadioButton1Click(Sender: TObject);
 procedure RadioButton2Click(Sender: TObject);
 private
 { Private declarations }
 public
 { Public declarations }
end;

var

ModelDechF: TModelDechF;

implementation

uses AdvOptToOurPlot, Constants, STEoptions;

{ \$R *.dfm }

procedure TModelDechF.ChoiceModelDechanneling;
begin
 if ChDechEn.Checked **then**
 begin
 ModelDechanneling := 1;
 end;

```

if ChDechDistance.Checked then
    begin
        ModelDechanneling := 2;
    end;
end;

procedure TModelDechF.ChDechEnClick(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            Checked := Checked;
            ChDechDistance.Checked := not Checked;
            FormOptions.Edit13.Enabled := Checked;
            FormOptions.Label11.Enabled := Checked;
            FormOptions.Label12.Enabled := Checked;
            AdvOptF.LabelEx.Enabled := Checked;
            AdvOptF.LabeledEdit3.Enabled := Checked;
            AdvOptF.LabeledEdit4.Enabled := Checked;
        end;
        ChoiceModelDechanneling;
end;

procedure TModelDechF.ChDechDistanceClick(Sender: TObject);
begin
    with Sender as TCheckBox do
        begin
            Checked := Checked;
            ChDechEn.Checked := not Checked;
            FormOptions.Edit13.Enabled := not Checked;
            FormOptions.Label11.Enabled := not Checked;
            FormOptions.Label12.Enabled := not Checked;
            AdvOptF.LabelEx.Enabled := not Checked;
            AdvOptF.LabeledEdit3.Enabled := not Checked;
            AdvOptF.LabeledEdit4.Enabled := not Checked;
        end;
        ChoiceModelDechanneling;
end;

procedure TModelDechF.RadioButton1Click(Sender: TObject);
begin
    AllDataAveraging := True;
end;

```

```
procedure TModelDechF.RadioButton2Click(Sender: TObject);  
begin  
    AllDataAveraging := False;  
end;  
  
end.
```

unit SpecialLoadFails;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ComCtrls, ShellCtrls, Spin;

type

TFSpLoad = **class**(TForm)

ShellTreeView1: TShellTreeView;

ShellComboBox1: TShellComboBox;

ShellListView1: TShellListView;

ButCaseFiles: TButton;

Memo1: TMemo;

ButCompatibilityFiles: TButton;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

LabStepOnTau: TLabel;

LabPointOnStep: TLabel;

LabCountfor1Tr: TLabel;

Label10: TLabel;

Edit1: TEdit;

SpinEdit1: TSpinEdit;

Label11: TLabel;

LabCountTr: TLabel;

ButRunLoad: TButton;

Button1: TButton;

Label7: TLabel;

ButSelAll: TButton;

procedure ButCaseFilesClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormDestroy(Sender: TObject);

procedure ButCompatibilityFilesClick(Sender: TObject);

procedure ButRunLoadClick(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure ButSelAllClick(Sender: TObject);

private

{ Private declarations }

```

    tmpFullDirFirstTable: string;
public
    { Public declarations }
    ResultS: TStringList;
end;

var
    FSpLoad: TFSpLoad;

implementation

uses STetraject, Constants;

{$R *.dfm}

procedure TFSpLoad.FormCreate(Sender: TObject);
begin
    ResultS := TStringList.Create;
end;

procedure TFSpLoad.FormDestroy(Sender: TObject);
begin
    ResultS.Free;
end;

procedure TFSpLoad.ButCaseFilesClick(Sender: TObject);
var
    Result: TStringList;
    i, tmpL: integer;
    tmpS: string;
begin
    Result := TStringList.Create;
if ShellListView1.SelCount >= 1 then
    begin
        for i := 0 to ShellListView1.Items.Count - 1 do
            if ShellListView1.Items[i].Selected = True then
                if ShellListView1.folders[i].IsFolder = False then
                    Result.Add(ShellListView1.Folders[i].PathName);
    ResultS.Clear;
    for i := 0 to Result.Count - 1 do
        begin
            tmpS := Result.Strings[i];
            if ExtractFileExt(tmpS) = '.MYD' then

```



```

    begin
        tmpL := Length(tmpS);
        if (tmpS[tmpL-4] = 'b') and (tmpS[tmpL-5] = 'd') and (Pos('_layer',tmpS)>0) then
            ResultS.Add(Result.Strings[i]);
        end;
    end;
Result.Destroy; Memo1.Clear;
for i := 0 to ResultS.Count - 1 do
    Memo1.Lines.Add(' ' + IntToStr(i+1) + ' ' + ResultS.Strings[i]);
tmpFullDirFirstTable := ResultS.Strings[0];
PathForSpecLoad := ExtractFilePath(tmpFullDirFirstTable);
for i := 0 to ResultS.Count - 1 do
    begin
        tmpS := ExtractFileName(ResultS.Strings[i]);
        tmpL := Pos('db.MYD',tmpS);
        Delete(tmpS, tmpL, tmpL + 6); ResultS.Strings[i] := tmpS;
    end;
ButCompatibilityFiles.Enabled := True;
end;
end;

procedure TFSpLoad.ButCompatibilityFilesClick(Sender: TObject);
begin
    N_step_uzel := SpinEdit1.Value;
    Form1.CompatibilityMyFiles(tmpFullDirFirstTable, ResultS);
end;

procedure TFSpLoad.ButRunLoadClick(Sender: TObject);
begin
    Form1.Load_STE_File(True, 's');
end;

procedure TFSpLoad.Button1Click(Sender: TObject);
begin
    if PathForSpecLoad <> '' then
        ShellTreeView1.Path := PathForSpecLoad;
end;

procedure TFSpLoad.ButSelAllClick(Sender: TObject);
begin
    ShellListView1.SelectAll;
end;
end.

```

unit NormalDistribution;

interface

uses

Math, Constants;

var

v_ND:DbArr;

a_ND:double;

n_ND:integer;

Procedure ArrayNormalDistribution(sigma,alpha,beta:double);

implementation

var

epsilon_ND:double;

k,itermax,K_for_graph:integer;

PabTrap,PabEulerMcLoren:double;

sigma2:double;

sigma122:double;

s1sqrt2pi:double;

s13sqrt2pi:double;

s15sqrt2pi:double;

TestEX, TestDX, TestSigma:double;

TestSigma122, TestS1sqrt2pi: double;

function FindEX(arr:DbArr;last:integer):double;

var

i:integer;

sum:double;

begin

sum:=0;

for i:=0 **to** last **do**

sum:=sum+arr[i];

FindEX:=sum/(last+1);

end;

function FindDX(arr:DbArr;last:integer;EX:double):double;

var

i:integer;

sum:double;

```

begin
  sum:=0;
  for i:=0 to last do
    sum:=sum+Sqr(arr[i] - EX);
  FindDX:=sum/(last+1);
end;

function RhoGauss(v:double):double;
begin
  RhoGauss:=s1sqrt2pi * exp(- sqr(v-a_ND)*sigma122 );
end;

function RhoGaussTest(v:double):double;
begin
  RhoGaussTest:=TestS1sqrt2pi * exp(- sqr(v-TestEX)*TestSigma122 );
end;

function RhoGaussX(x:double):double;
begin
  RhoGaussX:=(a_ND-x)*s13sqrt2pi * exp(- sqr(x-a_ND)*sigma122 );
end;

function RhoGaussXX(x:double):double;
begin
  RhoGaussXX:=(Sqr(a_ND-x)-sigma2)*s15sqrt2pi * exp(- sqr(x-a_ND)*sigma122 );
end;

function NonLinF(x:double;k:integer):double;
begin
  NonLinF:= 0.5*( RhoGauss(v_ND[k-1]) + RhoGauss(x) )*
    ( x - v_ND[k-1]) - PabTrap/n_ND;
end;

function NonLinFforgraph(x:double):double;
begin
  NonLinFforgraph:= 0.5*( RhoGauss(v_ND[K_for_Graph-1]) + RhoGauss(x) )*
    ( x - v_ND[K_for_Graph-1]) - PabTrap/n_ND;
end;

function NonLinFEulerMcLoren(x:double;k:integer):double;
begin
  NonLinFEulerMcLoren:=NonLinF(x,k) + ( RhoGaussX(v_ND[k-1]) - RhoGaussX(x) )*
    Sqr( x - v_ND[k-1])/12;

```

end;

function NonLinFEulerMcLorenforgraph(x:double):double;

begin

NonLinFEulerMcLorenforgraph:=NonLinF(x,K_for_Graph) + (Rho-
GaussX(v_ND[K_for_Graph-1]) - RhoGaussX(x)) *
Sqr(x - v_ND[K_for_Graph-1])/12;

end;

function NonLinFx(x:double;k:integer):double;

begin

NonLinFx:= 0.5*(RhoGauss(v_ND[k-1]) + RhoGauss(x) +
RhoGaussX(x)*(x - v_ND[k-1]));

end;

function NonLinFxforgraph(x:double):double;

begin

NonLinFxforgraph:= 0.5*(RhoGauss(v_ND[K_for_Graph-1]) + RhoGauss(x) +
RhoGaussX(x)*(x - v_ND[K_for_Graph-1]));

end;

function NonLinFxEulerMcLoren(x:double;k:integer):double;

begin

NonLinFxEulerMcLoren:= NonLinFx(x,k) + (2*(RhoGaussX(v_ND[k-1]) - RhoGaussX(x)) +
RhoGaussXX(x)*(x - v_ND[k-1]))*(x - v_ND[k-1])/12;

end;

function NonLinFxEulerMcLorenforgraph(x:double):double;

begin

NonLinFxEulerMcLorenforgraph:= NonLinFx(x,K_for_Graph) + (
2*(RhoGaussX(v_ND[K_for_Graph-1]) - RhoGaussX(x)) +
RhoGaussXX(x)*(x - v_ND[K_for_Graph-1]))*(x - v_ND[K_for_Graph-1])/12;

end;

Function IntegrateTrap(f:TFuncdouble;a,b:double;nmax:integer):double;

var

i:integer;
tmpI,dx,x:double;

begin

dx:=(b-a)/nmax;
x:=a;
tmpI:=0;
for i:=1 **to** nmax **do**

```

begin
  tmpI:=tmpI+0.5*(f(x)+f(x+dx))*dx;
  x:=x+dx;
end;
IntegrateTrap:=tmpI;
end;

Function IntegrateEulerMcLoren(f,f1:TFuncdouble;a,b:double;nmax:integer):double;
var
  i:integer;
  tmp,dx,x:double;
begin
  dx:=(b-a)/nmax;
  x:=a+dx;
  tmp:=f(x);
  for i:=2 to nmax-1 do
    begin
      x:=x+dx;
      tmp:=tmp+f(x);
    end;
  IntegrateEulerMcLoren:=tmp+0.5*(f(a)+f(b))+(f1(a)-f1(b))/12;
end;

Procedure FindVEuler(k:integer);
var
  i:integer;
  p,xk:double;
begin
  xk:=v_ND[k-1];
  i:=0;
  repeat
    inc(i);
    p:= NonLinFEulerMcLoren(xk,k)/NonLinFxEulerMcLoren(xk,k);
    xk:=xk - p;
  until (i >= itermax) or (abs(p) < epsilon_ND);
  v_ND[k]:=xk;
end;

Procedure FindVTrap(k:integer);
var
  i:integer;
  p,xk:double;
begin

```

```

xk:=v_ND[k-1];
i:=0;
repeat
  inc(i);
  p:= NonLinF(xk,k)/NonLinFx(xk,k);
  xk:=xk - p;
until (i >= itermax) or (abs(p) < epsilon_ND);
v_ND[k]:=xk;
end;

```

Procedure ArrayNormalDistribution(sigma,alpha,beta:double);

```

begin
  itermax:=10;
  epsilon_ND:=1E-20;
  SetLength(v_ND,(n_ND+1));
  Sigma2:=Sqr(sigma);
  sigma122 := 1/(2*sigma2);
  s1sqrt2pi := 1/(sigma*sqrt(2*pi));
  s13sqrt2pi := 1/(sigma*sigma2*sqrt(2*pi));
  s15sqrt2pi := 1/(sigma*sqr(sigma2)*sqrt(2*pi));
  PabTrap:=IntegrateTrap(RhoGauss,alpha,beta,n_ND);
  PabEulerMcLoren:=IntegrateEulerMcLoren(RhoGauss,RhoGaussX,alpha,beta,n_ND);
  K_for_graph:=1;
  v_ND[0]:=alpha;
  for k:=1 to n_ND do
    begin
      FindVEuler(k);
    end;
  end;
end.

```

unit Plotting;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ComCtrls, ExtCtrls, TeEngine, Series, TeeProcs, Chart, ExtDlgs,
StdCtrls, Buttons, Gauges, Constants, STEtraject, Math, STEaddTrj,
ImgList, Spin, ComObj;

type

TFormPlotting = **class**(TForm)

Panel1: TPanel;

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

TabSheet3: TTabSheet;

Panel2: TPanel;

PageControl3: TPageControl;

TabSheet4: TTabSheet;

TabSheet5: TTabSheet;

BitBtn2: TBitBtn;

Edit7: TEdit;

UpDown1: TUpDown;

GroupBox2: TGroupBox;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

Edit8: TEdit;

Edit9: TEdit;

LabeledEdit2: TLabeledEdit;

UpDown2: TUpDown;

Edit10: TEdit;

BitBtn3: TBitBtn;

Button1: TButton;

Label11: TLabel;

Label12: TLabel;

Label13: TLabel;

CheckBox4: TCheckBox;

BitBtn4: TBitBtn;

Button2: TButton;

Button3: TButton;

LabeledEdit1: TLabeledEdit;

LabeledEdit3: TLabeledEdit;

```

LabeledEdit4: TLabelEdit;
BitBtn6: TBitBtn;
BitBtn7: TBitBtn;
BitBtn8: TBitBtn;
BitBtn9: TBitBtn;
BitBtn10: TBitBtn;
Label14: TLabel;
CheckBox5: TCheckBox;
LabeledEdit5: TLabelEdit;
CheckBox6: TCheckBox;
LabeledEdit6: TLabelEdit;
LabeledEdit7: TLabelEdit;
LabeledEdit8: TLabelEdit;
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
GroupBox3: TGroupBox;
CheckBox7: TCheckBox;
LabeledEdit9: TLabelEdit;
CheckBox8: TCheckBox;
Edit11: TEdit;
Label15: TLabel;
Label16: TLabel;
BitBtn13: TBitBtn;
BitBtn14: TBitBtn;
BitBtn15: TBitBtn;
Label19: TLabel;
Label20: TLabel;
GroupBox4: TGroupBox;
RadioButton5: TRadioButton;
LabeledEdit10: TLabelEdit;
RadioButton6: TRadioButton;
LabeledEdit11: TLabelEdit;
procedure FormCreate(Sender: TObject);
procedure TntBitBtn1Click(Sender: TObject);
function Epoperechprav(v,x:double):double;
procedure TntBitBtn2Click(Sender: TObject);
procedure CalculationOfLoseE(Estart: array of double);
procedure CalculateEmax(NmE,NmaxE: integer);
private
    Nsloev,Podsloi:integer;
    { Private declarations }
public
    { Public declarations }

```


end;

var

FormPlotting: TFormPlotting;
f_arr12:array of array of double;
Ep12:array of double;
saveplotnost:integer;
ShowNumberOverBarrier:boolean;
Emax:array of double;
ButtonOfDissipation:integer;
NumberSeries:integer;

implementation

{ \$R *.dfm }

uses printers;

procedure TFormPlotting.FormCreate(Sender: TObject);

begin

saveplotnost:=0;
ShowNumberOverBarrier:=False;

end;

function TFormPlotting.Epoperechprav(v,x:double):double;

begin

Epoperechprav:=2*epsilon*d_small*Sqr(v)*Diffuzia(x);

end;

procedure TFormPlotting.TntBitBtn1Click(Sender: TObject);

var

Nm,Nmax,N,sd:integer;
tau:double;
Ep_s,Ep_p,v_arr_new,p_random:double;

begin

Nm:=Allseria.AllTrCnt;
Nmax:=Trajects.Traject[0].Nmax;
SetLength(f_arr12,Nm,Nmax+1);
SetLength(Ep12,Nm);
tau:=Trajects.Traject[0].tau;
N:=0;

while N<=Nm-1 **do**

begin

```

f_arr12[N,0]:=0;
sd:=0;
Ep12[N]:=Vmax*Sqr(Trajects.Traject[N].v_arr[0])/2+U(Trajects.Traject[N].x_arr[0]);
while sd <= (Nmax-1) do
    begin
        f_arr12[N,sd+1]:=f_arr12[N,sd]+tau/2*(Epoperechprav(Trajects.Traject[N].v_arr[sd],
            Trajects.Traject[N].x_arr[sd]) + Epoerechprav(Trajects.Traject[N].v_arr[sd+1],
            Trajects.Traject[N].x_arr[sd+1]));
        inc(sd);
    end;
    inc(N);
end;
end;

```

```

procedure TFormPlotting.CalculateEmax(NmE,NmaxE: integer);

```

```

var
    n:integer;
begin
    SetLength(Emax,NmE);
for n:=0 to NmE-1 do
        Emax[n]:=E-deltaE[n,NmaxE];
end;

```

```

procedure TFormPlotting.CalculationOfLoseE(Estart: array of double);

```

```

var
    n,k:integer;
    Somnogitel,Sum,tau:double;
    Nm,Nmax:integer;
begin
    Nm:=Trajects.CntOfTraject;
    Nmax:=Trajects.Traject[0].Nmax;
    Somnogitel:=4*Pi*Sqr(Z1)*Sqr(e2)*Lel*d_small/(epsilon*mec2*v2_na_c2);
    tau:=Trajects.Traject[0].tau;
    glub:=Nmax;
    Form1.Gauge1.MaxValue:=Nm*Nmax;
    Form1.Gauge1.Progress:=0;
    SetLength(deltaE,Nm,Nmax+1);
for n:=0 to Nm-1 do
        begin
            Sum:=Estart[n]/Somnogitel;
            deltaE[n,0]:=Estart[n];
            for k:=1 to Nmax do
                begin

```

```

Sum:=Sum+(rho_el(Trajects.Traject[n].x_arr[k-1])+
    rho_el(Trajects.Traject[n].x_arr[k])+2*rho_07)*tau/4;
deltaE[n,k]:=Sum*Sommogitel;
Form1.Gauge1.Progress:=Form1.Gauge1.Progress+1;
Application.ProcessMessages;
end;
end;
CalculateEmax(Nm,Nmax);
Form1.Gauge1.MaxValue:=100;
end;

procedure TFormPlotting.TntBitBtn2Click(Sender: TObject);
var
    i:integer;
    Elose:array of double;
begin
    SetLength(Elose,Trajects.CntOfTraject);
    if (TControl(Sender).Name='BitBtn14') or (calc=0) then
        for i:=0 to Trajects.CntOfTraject-1 do Elose[i]:=0
        else
            for i:=0 to Trajects.CntOfTraject-1 do Elose[i]:=deltaE[i,Trajects.Traject[0].Nmax];
        CalculationOfLoseE(Elose);
    end;

end.

```