Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

Физический факультет

УТВЕРЖДАЮ:
декан физического факультета
С.Н. Филимонов

Оценочные материалы по дисциплине

Программная визуализация геоданных

по направлению подготовки
03.04.02 Физика

Направленность (профиль) подготовки:
«Фундаментальная и прикладная физика»

Форма обучения
Очная

Квалификация
Магистратура

Год приема
2025

СОГЛАСОВАНО:
Руководитель ОП
Т.В. Бордовицына

Председатель УМК
О.М. Сюсина

Томск – 2025

1. Компетенции и индикаторы их достижения, проверяемые данными оценочными материалами

Целью освоения дисциплины является формирование следующих компетенций:

– ПК-1 – Способен самостоятельно ставить конкретные задачи научных исследований в области физики и решать их с помощью современной аппаратуры и информационных технологий с использованием новейшего российского и зарубежного опыта;

Результатами освоения дисциплины являются следующие индикаторы достижения компетенций:

– ИПК-1.1 – Знает основные стратегии исследований в выбранной области физики, критерии эффективности, ограничения применимости;

– ИПК-1.2 – Умеет выделять и систематизировать основные цели исследований в выбранной области физики, извлекать информацию из различных источников, включая периодическую печать и электронные коммуникации, представлять её в понятном виде и эффективно использовать.

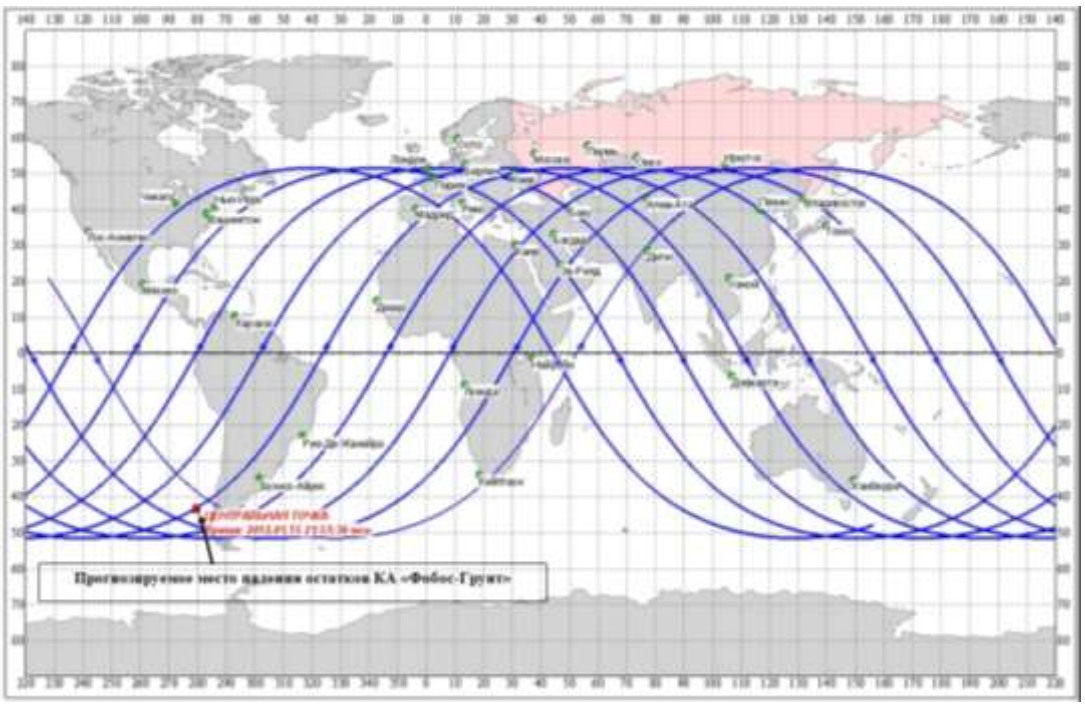
2. Оценочные материалы текущего контроля и критерии оценивания

Элементы текущего контроля:
– практическое задание.

Пример

1. Разработать в figma.com UI/UX дизайн ПО визуализации геоданных. Нарисовать карту, оси, примерные трассы спутников. (ПК-1)

Ключ:



2. Написать процедуру на языке высокого уровня по экспортированию карты мира map.jpg на форму и рассчитать коэффициент перехода от км к пикселам и изобразить сетку широт и долгот на карте на форме. (ИПК-1.1, ИПК-1.2)

Ключ:

```
begin
  x0 := 10;
  y0 := 450; // оси начинаются в точке (40,250)
  dx := 39;
  dy := 46; // шаг координатной сетки 40 пикселей
  dcross := 1; // помечать линии сетки X: 1 - каждую;
  //                          2 - через одну;
  //                          3 - через две;
  dlx := 15; // шаг меток оси X
  dly := 10.0; // шаг меток оси Y, метками будут: 1, 2, 3 и т.д.

  h := 440;
  w := 925;

  with form1.Canvas do
  begin
    cross := dcross;
    MoveTo(x0, y0);
    LineTo(x0, y0 - h); // ось X
    MoveTo(x0, y0);
    LineTo(x0 + w, y0); // ось Y

    // засечки, сетка и оцифровка по оси X
    x := x0 + dx;
    lx := dlx;
    repeat
      MoveTo(x, y0 - 3);
      LineTo(x, y0 + 3); // засечка
      cross := cross - 1;
      if cross = 0 then //оцифровка
      begin
        TextOut(x - 8, y0 + 5, FloatToStr(lx));
        cross := dcross;
      end;
      Pen.Style := psDot;
      MoveTo(x, y0 - 3);
      LineTo(x, y0 - h); // линия сетки
      Pen.Style := psSolid;
      lx := lx + dlx;
      x := x + dx;
    until (x > x0 + w);
    // засечки, сетка и оцифровка по оси Y
    y := y0 - dy;
    ly := dly;
    repeat
      MoveTo(x0 - 3, y);
      LineTo(x0 + 3, y); // засечка
      cross:=cross-1;
      if cross = 0 then //оцифровка
      begin
        TextOut(x0 - 5, y, FloatToStr(ly)); // оцифровка
        cross := dcross;
      end;
      //TextOut(x0 - 20, y, FloatToStr(ly)); // оцифровка

      Pen.Style := psDot;
      MoveTo(x0 + 3, y);
      LineTo(x0 + w, y); // линия сетки

      Pen.Style := psSolid;
      ly := ly + dly;
      y := y - dy;

    until (y < y0 - h);
  end;
end;
```

```
begin
StringGrid1.RowCount:=1;
 grid();
 if OpenTextFileDialog1.Execute then
     begin
      AssignFile(F,OpenTextFileDialog1.Filename);
      Reset(F);

      x0:=Form1.Image1.Width div 2;
      y0:=Form1.Image1.Height div 2;

      for I := 1 to 2400 do
       begin
        Readln(F,x,y);

         Form1.Canvas.Ellipse(x0+round((x-r)*mash),y0+round((y-r)*mash),
x0+round((x+r)*mash),y0+round((y+r)*mash));
         StringGrid1.RowCount:=StringGrid1.RowCount+1;
         StringGrid1.Cells[0,i]:=floattostr(i);
          StringGrid1.Cells[1,i]:=floattostr(x);
          StringGrid1.Cells[2,i]:=floattostr(y);

       end;
      CloseFile(F);
     end;
end;
```

3. Реализация невозмущенного кеплеровского движения; Определение начальных значений элементов орбиты каждого КА из спутников системы; Переход от элементов орбиты к векторам положения и скорости и обратно; Переход от календарной к юлианской дате; Вычисления звездного времени; Вычисление матрицы поворота; Вычисление сферических координат; Переход от сферических координат к пикселам; Визуализация трасс спутников на карте. (ПК-1, ИПК-1.1, ИПК-1.2)

Ключ:

```
double MainWindow::sid2000(double jd)
{
```

```
    const int jd2000 = 2451545;
    const int jdyear = 36525;
    double m, mm, d, t, s, sr;
    m = frac(jd) - 0.5;
    d = jd - m - jd2000;
    t = (d + m) / jdyear;
    mm = m * 86400;
    s = (24110.54841+mm+236.555367908*(d+m)+(0.093104*t+6.21E-
6*t*t)*t)/86400*2*M_PI;
    sr = reduce(s);
    return sr;
}

double MainWindow::reduce(double a)
{
    const double pi2 = M_PI * 2;
    double B;
    B = a - trunc(a/pi2)* pi2;
    if (a<0)
        B = B + pi2;
    return B;
}

double MainWindow::arcsin(double x)
{
    if (abs(x)>1)
    {
      exit(0);
    }
    if (x==1)
    {
      return 2*M_PI;
    }
    if (x==-1)
      return -2*M_PI;

    return atan(x/sqrt(1-x * x));
}

double MainWindow::date_jd(int year, int month, double day)
{
    const int han = 100;
    int i, me, ja, jb;
    double m1, d, date, jd, result;
    date = year + month/100. + day / 1e4;
    i = trunc(date);
    m1 = (date-i) * han;
    me = trunc(m1);
    d = (m1 - me) * han;
    if (me > 2)
        goto first;
    i = i - 1;
```

```
    me = me + 12;
first:
    jd = trunc(365.25*i) + trunc(30.6001 * (me + 1)) + d + 1720994.5;
    if (date < 1582.1015)
    {
        result = jd;
        return result;
    }
    ja = trunc(i / 100.);
    jb = 2 - ja + trunc(ja / 4.);
    jd = jd + jb;
    result = jd;
    return result;
}

double MainWindow::newthon_method(double M0, double e, double T_per, double del_t)
{
    const double eps = 1e-12;
    double n, M, E0, E1, r, res;
    n = 2 * M_PI / T_per;
    M = n * del_t + M0;
    E0 = M;
    r = 1;
    while (r > eps)
    {
        E1 = E0 - ((E0 - e * sin(E0) - M) / (1 - e * cos(E0)));
        r = abs(E1 - E0);
        E0 = E1;
        res = E1;
    }
    return res;
}

QVector<double> MainWindow::eleps(double M0, double a, double e, double i, double W1,
double w, double t, double T_per)
{
    double n, Ee, r, alpha, betta, gamma, dalpha, dbetta, dgamma;

    QVector<double> result(6);

    n = 2 * M_PI / T_per;
    Ee = newthon_method(M0, e, T_per, t);
    alpha = cos(w) * cos(W1) - sin(w) * sin(W1) * cos(i);
    betta = cos(w) * sin(W1) + sin(w) * cos(W1) * cos(i);
    gamma = sin(w) * sin(i);

    dalpha = -sin(w) * cos(W1) - cos(w) * sin(W1) * cos(i);
    dbetta = -sin(w) * sin(W1) + cos(w) * cos(W1) * cos(i);
    dgamma = cos(w) * sin(i);

    result[0] = a * (alpha * (cos(Ee) - e) + dalpha * sqrt(1 - e * e) * sin(Ee));
    result[1] = a * (betta * (cos(Ee) - e) + dbetta * sqrt(1 - e * e) * sin(Ee));
```

```
    result[2] = a * (gamma * (cos(Ee) - e) + dgamma * sqrt(1 - e * e) * sin(Ee));
    r = sqrt(result[0] * result[0] + result[1] * result[1] + result[2] * result[2]);
    result[3] = n * a * a / r * (-alpha * sin(Ee) + dalpha * sqrt(1 - e * e) * cos(Ee));
    result[4] = n * a * a / r * (-betta * sin(Ee) + dbetta * sqrt(1 - e * e) * cos(Ee));
    result[5] = n * a * a / r * (-gamma * sin(Ee) + dgamma * sqrt(1 - e * e) * cos(Ee));

    return result;
}

QVector<double> MainWindow::ToTRS(double phi, double lam, double H, const
QVector<double> &coord_dec)
{
    double z_t, x_t, y_t, H_1, r, h_o, Az;
    QVector<double> r_xyz(3), r1(3), r_sez(3), result(3);
    QVector<QVector<double>> A_t(3, QVector<double> (3)), M_t(3, QVector<double> (3));
    int i, j;

    z_t = Re*sin(phi);

    H_1 = H+lam;

    x_t = Re*cos(phi)*cos(H_1);
    y_t = Re*cos(phi)*sin(H_1);

    r_xyz[0] =coord_dec[1] - x_t;
    r_xyz[1] =coord_dec[2] - y_t;
    r_xyz[2] =coord_dec[3] - z_t;

    A_t[0][0] = cos(H_1);
    A_t[0][1] = sin(H_1);
    A_t[0][2] = 0;
    A_t[1][0] = -sin(H_1);
    A_t[1][1] = cos(H_1);
    A_t[1][2] = 0;
    A_t[2][0] = 0;
    A_t[2][1] = 0;
    A_t[2][2] = 1;

    M_t[0][0] = sin(phi);
    M_t[0][1] = 0;
    M_t[0][2] = -cos(phi);
    M_t[1][0] = 0;
    M_t[1][1] = 1;
    M_t[1][2] = 0;
    M_t[2][0] = cos(phi);
    M_t[2][1] = 0;
    M_t[2][2] = sin(phi);

    for (i = 0; i < 3; i++)
    {
        r1[i] = 0;
        for (j = 0; j < 3; j++)
```

```
            r1[i] = r1[i]+A_t[i][j]*r_xyz[j];
    }
    for (i = 0; i < 3; i++)
    {
       r_sez[i] = 0;
       for (j = 0; j < 3; j++)
         r_sez[i] = r_sez[i]+M_t[i][j]*r1[j];
    }

    r = sqrt(r_sez[0] * r_sez[0] + r_sez[1] * r_sez[1] + r_sez[2] * r_sez[2]);
    h_o = asin(r_sez[2]/r);
    Az = atan2(-r_sez[1], r_sez[0]);
    result[0]  = r;
    result[1]  = h_o;
    result[2]  = Az;

    return result;
}

QVector<double> MainWindow::ToCRS(double r_z, double Az, double H, double phi, double
lam)
{
    double rz_s, rz_e, rz_z, H_1, x_t, y_t, z_t, x_2, y_2, z_2, r;
    QVector<double> r_sez(3), r_xyz(3), x(3), y(3);
    QVector<QVector<double>> A_t(3, QVector<double> (3)), M_t(3, QVector<double> (3));
    int ii, jj;
    QVector<double> r_2(3);
    QVector<double> result(2);

    rz_s  = r_z * cos(Az * rad) * cos(h_min);
    rz_e  = -r_z * sin(Az * rad) * cos(h_min);
    rz_z = r_z * sin(h_min);

    H_1  = H + lam;

    z_t = Re*sin(phi);
    x_t = Re*cos(phi)*cos(H_1);
    y_t = Re*cos(phi)*sin(H_1);

    r_sez[0] = rz_s;
    r_sez[1] = rz_e;
    r_sez[2] = rz_z;

    M_t[0][0] = sin(phi);
    M_t[0][1] = 0;
    M_t[0][2] = cos(phi);
    M_t[1][0] = 0;
    M_t[1][1] = 1;
    M_t[1][0] = 0;
    M_t[2][0] = -cos(phi);
    M_t[2][1] = 0;
    M_t[2][2] = sin(phi);
```

```
    for (ii = 0; ii < 3; ii++)
    {
        r_2[ii] = 0;
        for (jj = 0; jj < 3; jj++)
            r_2[ii] = r_2[ii]+M_t[ii][jj]*r_sez[jj];
    }

    A_t[0][0] = cos(H_1);
    A_t[0][1] = -sin(H_1);
    A_t[0][2] = 0;
    A_t[1][0] = sin(H_1);
    A_t[1][1] = cos(H_1);
    A_t[1][2] = 0;
    A_t[2][0] = 0;
    A_t[2][1] = 0;
    A_t[2][2] = 1;

    for (ii = 0; ii < 3; ii++)
    {
        r_xyz[ii] = 0;
        for (jj = 0; jj < 3; jj++)
            r_xyz[ii] = r_xyz[ii]+A_t[ii][jj]*r_2[jj];
    }

    x_2 = r_xyz[0]+x_t;
    y_2 = r_xyz[1]+y_t;
    z_2 = r_xyz[2]+z_t;

    x[0] = x_2;
    x[1] = y_2;
    x[2] = z_2;

    y[0] = cos(H)*x[0]+sin(H)*x[1];
    y[1] = -sin(H)*x[0]+cos(H)*x[1];
    y[2] = x[2];

    r = sqrt(y[0] * y[0] + y[1] * y[1] + y[2] * y[2]);

    result[0]  = atan2(y[1], y[0]) * deg;   // лямбда
    result[1]  = asin(y[2]/r) * deg;          // фи

    return result;
}

// Считываю файл эфемерид, вычисляю лямбда и фи, записываю в двумерный вектор,
//чтобы можно было вывести отдельные трассы
void MainWindow::makeCalculation()
{
    int j, jj;
    QByteArray data;
    int day, month, year;
```

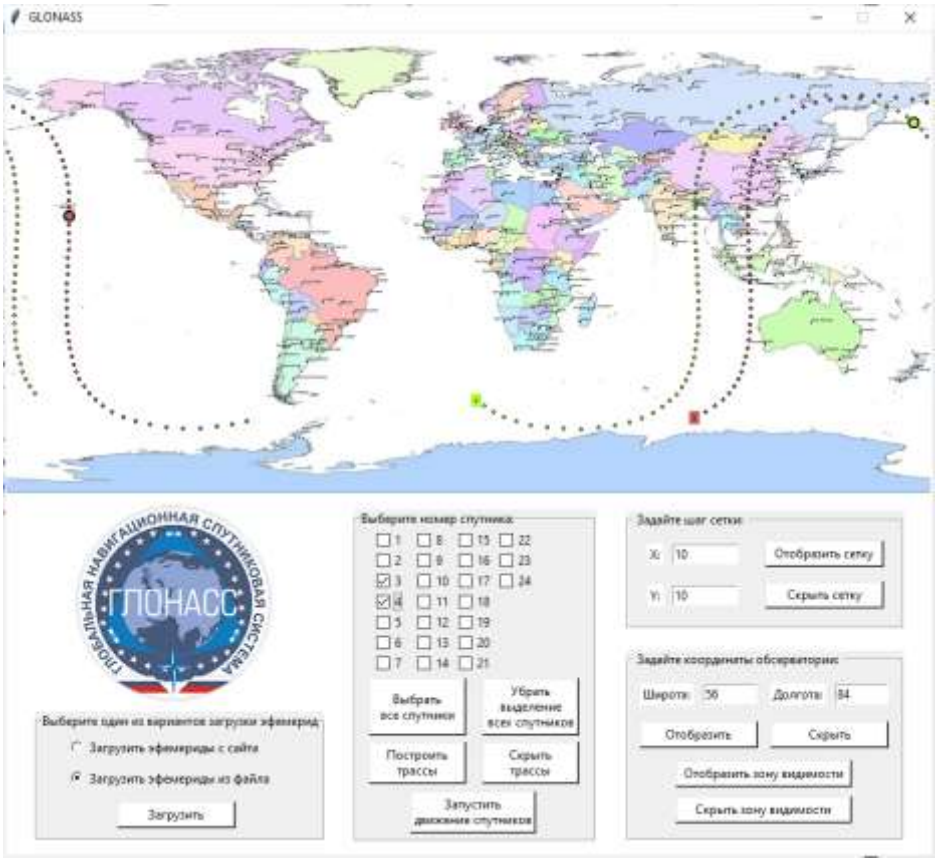
```
    double T_O, T_per, e_s, i_s, L_O, w_s;
    double JD_O, JD, JD0;
    double H_O, O_s, dt, H_ot, r_ot, lam_ot, phi_ot;
    double v_o, E_O, M_O, a_s, del_t, H, r, lam, phi;
    QVector<double> coord_dec(6), xv(6);
    QVector<double> y(3), y_ot(3);

    QFile myFile(":/efem/efemer.txt");
    myFile.open(QFile::ReadOnly | QFile::Text);
    JD0 = date_jd(lineEditValues[0], lineEditValues[1], lineEditValues[2] + lineEditValues[3] /
24 + lineEditValues[4] / (60 * 24)) - 3 / 24;

    for (j = 1; j < 25; j++)
    {
       data = myFile.readLine();
       day = data.split('\t')[1].split(' ')[0].toInt();
       month = data.split('\t')[1].split(' ')[1].toInt();
       year = data.split('\t')[1].split(' ')[2].toInt();
       T_O = data.split('\t')[2].toDouble();
       T_per = data.split('\t')[3].toDouble();
       e_s = data.split('\t')[4].toDouble();
       i_s = data.split('\t')[5].toDouble();
       L_O = data.split('\t')[6].toDouble();
       w_s = data.split('\t')[7].toDouble();

       i_s = i_s * rad;
       L_O = L_O * rad;
       w_s = w_s * rad;

       JD = date_jd(2000 + year, month, day) - 3 / 24;
       JD_O = JD + T_O / 86400;
       H_O = sid2000(JD_O);
       O_s = L_O + H_O;

       v_o = -w_s;
       E_O = 2*atan2(tan(v_o/2), sqrt((1 + e_s)/(1 - e_s)));
       M_O = E_O -  e_s * sin(E_O);
       a_s = pow(pow((T_per / (2 * M_PI)), 2) * my, 1. / 3);

       del_t = (JD0 - JD_O) * 86400;
       coord_dec = eleps(M_O, a_s, e_s, i_s, O_s, w_s, del_t, T_per);

       H = sid2000(JD0);
       y[0] = cos(H) * coord_dec[0] + sin(H) * coord_dec[1];
       y[1] = -sin(H) * coord_dec[0] + cos(H) * coord_dec[1];
       y[2] = coord_dec[2];
       r = sqrt(y[0] * y[0] + y[1] * y[1] + y[2] * y[2]);
       lam = atan2(y[1], y[0]) * deg;
       phi = asin(y[2] / r) * deg;

       vec_lam.push_back(lam);
       vec_phi.push_back(phi);
```

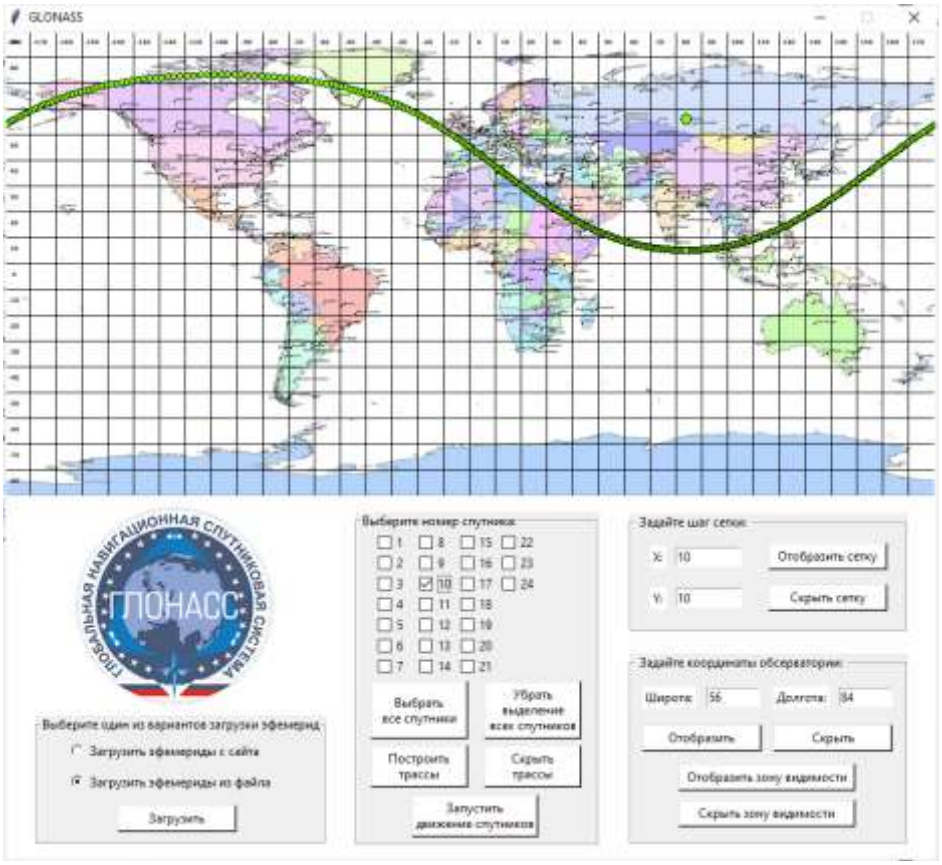
```
        for (jj = 0; jj < 401; jj++)
        {
            dt = (JD0 - JD_O) * 86400 + jj * T_per / 400;
            xv = eleps(M_O, a_s, e_s, i_s, O_s, w_s, dt, T_per);
            H_ot = sid2000(JD0 + jj / 400. * T_per / 86400);
            y_ot[0] = cos(H_ot) * xv[0] + sin(H_ot) * xv[1];
            y_ot[1] = -sin(H_ot) * xv[0] + cos(H_ot) * xv[1];
            y_ot[2] = xv[2];
            r_ot = sqrt(y_ot[0] * y_ot[0] + y_ot[1] * y_ot[1] + y_ot[2] * y_ot[2]);
            lam_ot = atan2(y_ot[1], y_ot[0]) * deg;
            phi_ot = asin(y_ot[2] / r_ot) * deg;

            vec_lam_ot.push_back(lam_ot);
            vec_phi_ot.push_back(phi_ot);

        }
        vec_lam2d.push_back(vec_lam);
        vec_phi2d.push_back(vec_phi);
        vec_lam_ot2d.push_back(vec_lam_ot);
        vec_phi_ot2d.push_back(vec_phi_ot);
        vec_lam_ot.clear();
        vec_phi_ot.clear();
        vec_lam.clear();
        vec_phi.clear();
    }

    myFile.close();
}

// отвечает за отрисовку выбранных трасс
void MainWindow::makeTracks(QVector<int> input_vect)
{
    for (auto j = 0; j < input_vect.size(); j++)
    {
        if (input_vect[j])
        {
            ui->widget->addGraph();
            ui->widget->graph()->setLineStyle(QCPGraph::LineStyle(QCPGraph::lsNone));
            ui->widget->graph()->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssDisc, 4));
            ui->widget->graph()->addData(vec_lam_ot2d[j], vec_phi_ot2d[j]);
            graphStates.push_back(1);  // нужно, чтобы потом можно было менять цвет только
выбранных трасс
        }
        else
            graphStates.push_back(0);
    }
}

// отвечает за отрисовку точек положения Спутников
void MainWindow::makeSattPoint(QVector<int> input_vect)
{
    for (auto j = 0; j < input_vect.size(); j++)
```

```
    {
        if (input_vect[j])
        {
            ui->widget->addGraph();
            ui->widget->graph()->setLineStyle(QCPGraph::LineStyle(QCPGraph::lsNone));
            ui->widget->graph()->setScatterStyle(QCPScatterStyle(*sattPng));
            ui->widget->graph()->addData(vec_lam2d[j], vec_phi2d[j]);
            QCPItemText* textLabel;
            textLabel = new QCPItemText(ui->widget);
            textLabel->setText(QString::number(j + 1));
            textLabel->setColor(Qt::black);
            textLabel->setFont(QFont("sans", 10));
            textLabel->setPen(QPen(Qt::black));
            textLabel->position->setCoords(vec_lam2d[j][0], vec_phi2d[j][0]);
        }
    }
}

// Нажатие на кнопку "Построить трассы"
void MainWindow::on_pushButton_clicked()
{
    if (checkBoxStates.isEmpty())
    {
        makeTracks();
        makeSattPoint();
    }
    else
    {
        makeTracks(checkBoxStates);
        makeSattPoint(checkBoxStates);
    }
    ui->widget->replot();
}
```

4. Создание интерфейса проекта. Построение трасс и зон видимости КА навигационной системы. (ИПК-1.1, ИПК-1.2)

Ключ:


GLONASS
ГЛОНАСС
Выберите номер спутника:
Задайте шаг сетки:
X: 10
Y: 10
Отобразить сетку
Скрыть сетку
Задайте координаты обсерватории:
Широта: 56
Долгота: 84
Отобразить
Скрыть
Отобразить зону видимости
Скрыть зону видимости
Выбрать все спутники
Убрать выделение всех спутников
Построить трассы
Скрыть трассы
Запустить движение спутников
Выберите один из вариантов загрузки эфемерид
Загрузить эфемериды с сайта
Загрузить эфемериды из файла
Загрузить


GLONASS
ГЛОНАСС
Выберите номер спутника:
Задайте шаг сетки:
X: 10
Y: 10
Отобразить сетку
Скрыть сетку
Задайте координаты обсерватории:
Широта: 56
Долгота: 84
Отобразить
Скрыть
Отобразить зону видимости
Скрыть зону видимости
Выбрать все спутники
Убрать выделение всех спутников
Построить трассы
Скрыть трассы
Запустить движение спутников
Выберите один из вариантов загрузки эфемерид
Загрузить эфемериды с сайта
Загрузить эфемериды из файла
Загрузить

3. Оценочные материалы итогового контроля (промежуточной аттестации) и критерии оценивания

Зачет с оценкой во втором семестре проводится в виде защиты своего проекта. Выступление перед группой с презентацией и демонстрацией ПО. Продолжительность зачета 1,5 часа.

Результаты зачета с оценкой определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Оценка «отлично» зависит от посещаемости (не менее 80% занятий), дизайна ПО (соответствовала правилам UI/UX дизайна), функциональной части (все заявленные элементы ПО были внедрены и выполнены), творческий подход к созданию ПО (придумать новую опцию).

Оценка «хорошо» зависит от посещаемости (не менее 50% занятий), дизайна ПО (не обязательно 100% соответствовать всем правилам UI/UX дизайна), функциональной части (все заявленные элементы ПО были внедрены и выполнены).

Оценка «удовлетворительно» зависит от посещаемости (не менее 30% занятий), дизайн не рассматривается, функциональная часть (все заявленные элементы ПО были внедрены и выполнены).

4. Оценочные материалы для проверки остаточных знаний (сформированности компетенций)

Тесты:

1. Поставьте в соответствии: (ИПК-1.1, ИПК-1.2)
   1) Galileo
   2) Цикада
   3) Transit

   и

   a) 6 спутников на круговых околоземных орбитах (высотой около 1000 км) с равномерным распределением плоскостей орбиты вдоль экватора. (Ω). *i*=64,8
   b) 4 спутника. Круговые орбиты высотой 1000 км, *i*=83; Плоскости орбит наклонены на 45° друг к другу. Равномерное распределение плоскостей орбит вдоль экватора (Ω).
   c) e=0; *i*=56; a=23222 км; Ω=0, 120, 240. С равномерным распределением 9 спутников на орбите (ω).

   Ключ: 1 – c, 2 – b, 3 – a.

2. Поставьте в соответствии: (ИПК-1.1, ИПК-1.2)
   1) IRNSS
   2) Бэйдоу
   3) GPS

   и

   a) 5 спутников на геостационарной орбите (ГСО), 3 спутника в трех плоскостях (высота 35 786 км, наклонение 55 град) и 27 спутников в трех плоскостях на средних орбитах (высота 21 500 км, наклонение 55 град).
   b) Круговая орбита с высотой порядка 20200 км. является орбитой суточной кратности с периодом обращения 11 часов 58 минут; таким образом, спутник совершает два витка вокруг Земли за одни звёздные сутки (23 часа 56 минут). Наклонение орбиты (55°) является также общим для всех спутников системы. Единственным отличием орбит спутников является

долгота восходящего узла, или точка, в которой плоскость орбиты спутника пересекает экватор: данные точки отстоят друг от друга приблизительно на 60 градусов. Таким образом, несмотря на одинаковые (кроме долготы восходящего узла) параметры орбит, спутники обращаются вокруг Земли в шести различных плоскостях, по 4 аппарата в каждой.

c) Количество спутников 7, три из них геостационарные: a=42000,0 км, i=0, e=0, λ=34, 83, 132. Параметры остальных четырёх спутников: a=18753,1 км, i=29°, e=0,63323, две орбиты пересекают экватор в точке λ=55 и две λ=111.

Ключ: 1 – c, 2 – a, 3 – b.

3. Сколько орбитальных областей занимают спутники ГЛОНАСС? (ИПК-1.1, ИПК-1.2)
   a) 3
   b) 2
   c) 1
   d) 4
   e) 6
   f)

   Ключ: a

4. Поставьте в соответствии навигационные системы и количество спутников в ней. (ИПК-1.1, ИПК-1.2)
   1. IRNSS
   2. Бэйдоу
   3. GPS
   4. Galileo
   5. Цикада
   6. Transit
   7. ГЛОНАСС

и

   a) 24
   b) 24
   c) 35
   d) 7
   e) 6
   f) 4
   g) 9

   Ключ: 1 – d, 2 – c, 3 – a, 4 – g, 5 – f, 6 – e, 7 – b.

Информация о разработчиках

Баньщикова Мария Александровна, к.ф.-м.н., доцент, доцент, ФФ ТГУ