Результаты тестирования программного комплекса EULER на сохранение энергии при движении консервативных механических систем

В консервативных механических системах в любой точке их движения сумма потенциальной и кинетической энергий должна сохранять постоянное значение. Для проверки точности сохранения энергии при расчете движения консервативных систем в EULER были проведены расчеты следующих примеров.

- Пример 1. Одно тело, связанное вращательным шарниром, в поле силы тяжести.
- Пример 2. Три тела, связанные вращательными шарнирами, в поле силы тяжести.
- Пример 3. Неуравновешенный гироскоп в поле силы тяжести.

Подробное описание этих примеров и результаты тестирования представлены ниже. Во всех представленных примерах в процессе движения консервативных механических систем производится расчет относительной погрешности сохранения суммарной энергии (датчик с именем energy_REL_ERROR). Эта погрешность зависит от шага и времени численного интегрирования. В таблице 1 представлены максимальные значения относительных погрешностей сохранения энергии на интервале расчета 10 [s].

Таблица 1. Максимальные значения относительных погрешностей сохранения энергии на интервале расчета 10 [s].

Шаг интегрирования [s]	Относительная погрешность сохранения энергии		
	Пример 1	Пример 2	Пример 3
0.01	5. e-6	7. e-3	4. e-3
0.001	2. e-9	6. e-7	5. e-8
0.0001	2. e-9	1. e-8	3. e-10

Пример 1.

Одно тело, связанное вращательным шарниром, в поле силы тяжести. Пример содержится в файле «body_1_rot_1.elr». Вид модели примера представлен на рисунке 1.1. Графики погрешностей сохранения энергии системы при различных шагах численного интегрирования представлены на рисунках 1.2, 1.3, 1.4.

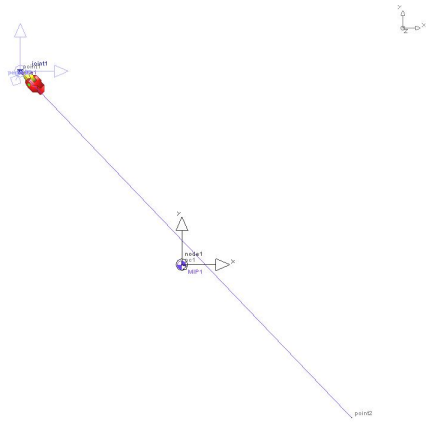

Рис. 1.1. Вид модели примера 1 (файл «body_1_rot_1.elr»)

Текст EULER-проекта примера 1 (файл «body_1_rot_1.elr»):

```
// Тест на сохранение энергии консервативной механической системы.
//    Одно тело, связанное вращательным шарниром, в поле силы тяжести.
//
scalar m=1[kg];
scalar Ix=0.001[kg m2];
scalar Iy=0.01[kg m2];
scalar Iz=0.02[kg m2];
tensor tensor1=mainTensor( Ix, Iy, Iz );
scalar g=10[m/s2];
//
point point1=point( 0 [ m ], 0 [ m ], 0 [ m ] );
point point2=point( 4.1000e-001 [ m ], -4.3000e-001 [ m ], 0.0000e+000 [ m ] );
line line1=polyLine( list( point1, point2 ) );
point pc1=point( 2.0000e-001 [ m ], -2.4000e-001 [ m ], 0.0000e+000 [ m ] );
node node1=nodePoint( pc1 );
MIP MIP1=massNode( node1, m, tensor1 );
//
color color_base=index( 6 );
color color_G=index( 72 );
body base=body( color = color_base );
set ground = base;
body G=body( color = color_G );
body G < ( line1, MIP1 );
//
vector vector1=vector( -1 [ m ], -2 [ m ], -7 [ m ], point = point1 );
joint joint1=rotational( base, G, point1, vector1 );
//
gravity gravity1=parallel( projectY, g = g );
//
sensor Vx1=transVelocity( base, projectX, G, pc1 );
sensor Vy1=transVelocity( base, projectY, G, pc1 );
sensor Vz1=transVelocity( base, projectZ, G, pc1 );
sensor Wx1=rotVelocity( G, vectorX( node1, visible = hide: ), base );
sensor Wy1=rotVelocity( G, vectorY( node1, visible = hide: ), base );
```

```
sensor Wz1=rotVelocity( G, vectorZ( node1, visible = hide: ), base );
sensor EK "кинетическая энергия"=m*(Vx1*Vx1+Vy1*Vy1+Vz1*Vz1)/2+
          Ix*Wx1*Wx1/2[rad2]+Iy*Wy1*Wy1/2[rad2]+Iz*Wz1*Wz1/2[rad2];
sensor h1=bodyDisplacement( base, pc1, projectY, G, pc1 );
sensor EP "потенциальная энергия"=-m*g*h1;
sensor ES "суммарная энергия"=EK+EP;
sensor Ebase "примерный базовый уровень для контроля ошибки"=1[kg m2/s2];
//
sensor energy_REL_ERROR "относительная ошибка энергии"=ES/Ebase;
//

/\////////////////////////////////////////////////////////////////////////////////
/\   Единицы измерения;
set units = SI;
```

energy_tests\body_1_rot_1.elr | EULER 8.12 | step_const=0.01[s]

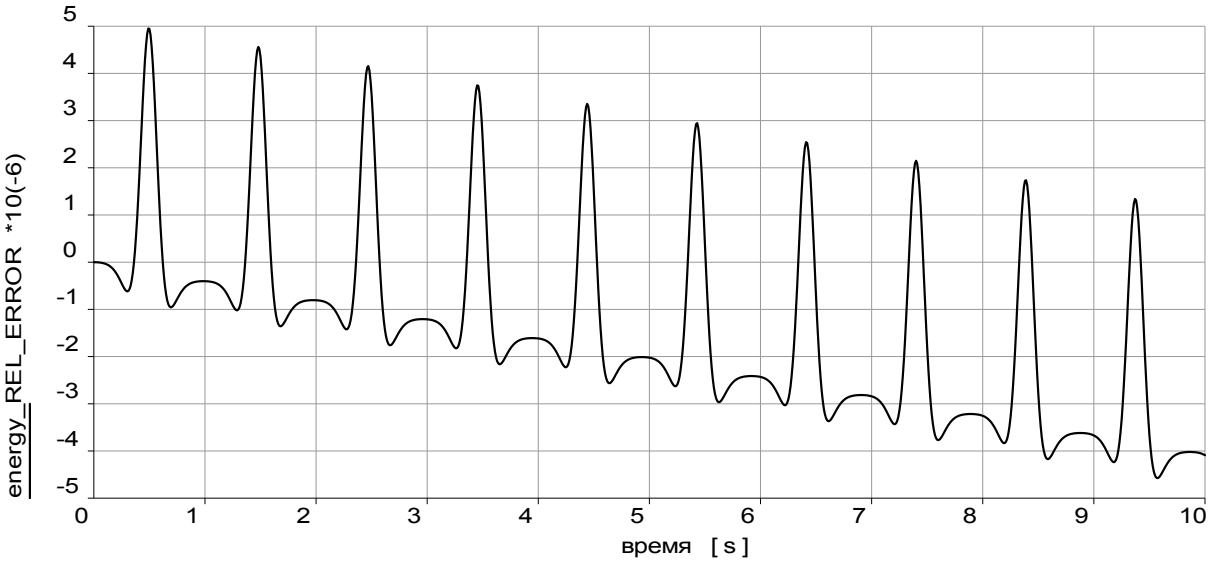

Рис. 1.2. Погрешность сохранения энергии при шаге интегрирования 0.01[s]

energy_tests\body_1_rot_1.elr | EULER 8.12 | step_const=0.001[s]

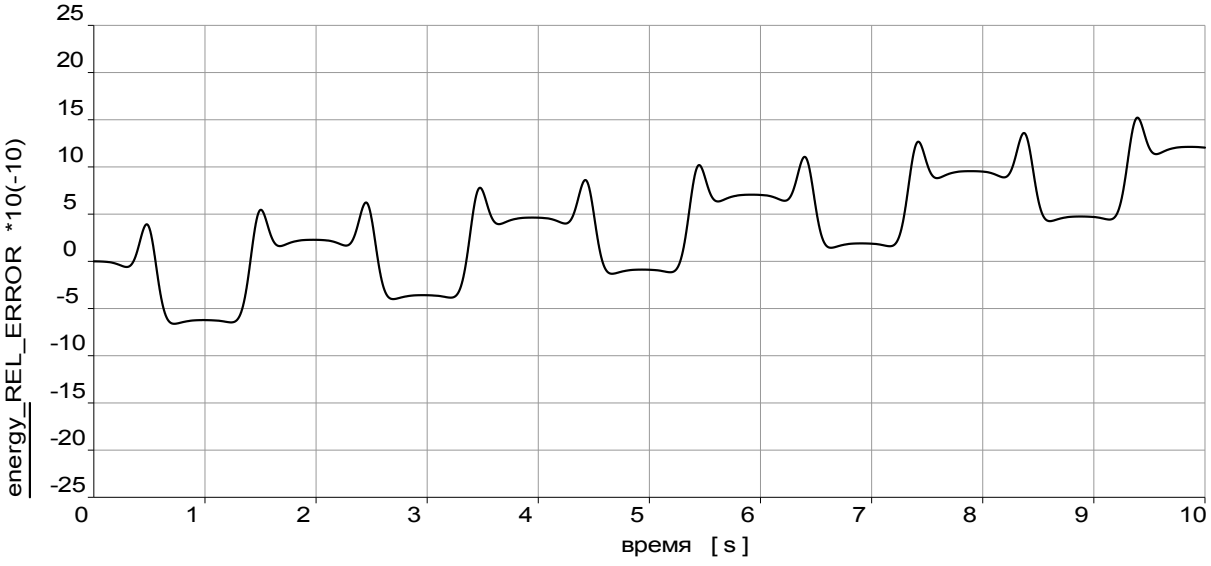

Рис. 1.3. Погрешность сохранения энергии при шаге интегрирования 0.001[s]

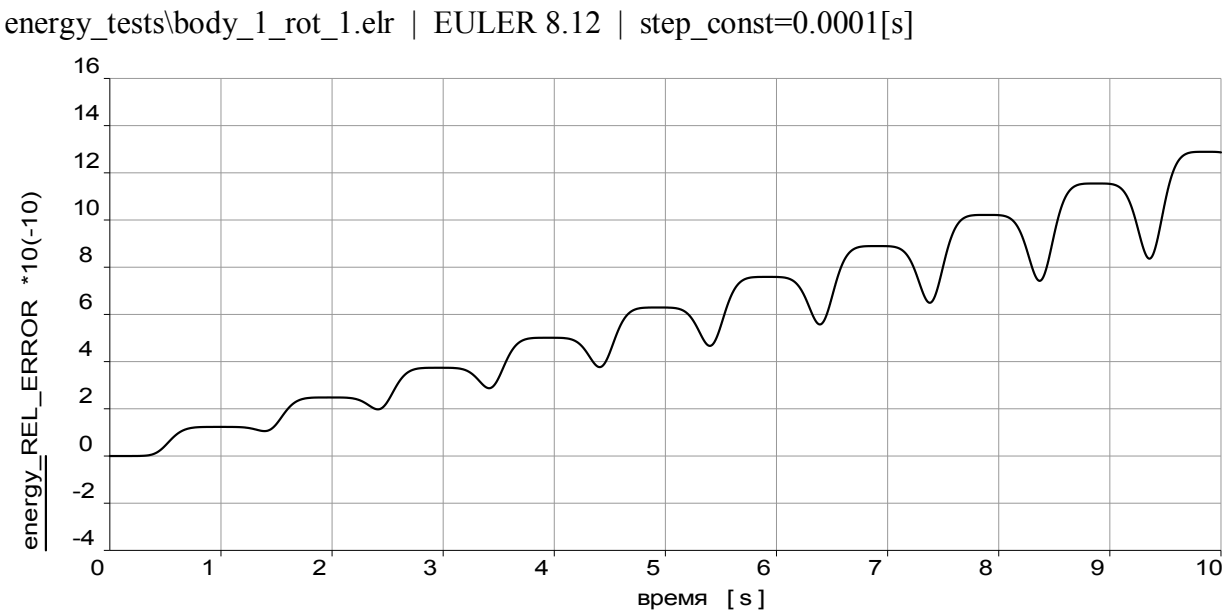

Рис. 1.4. Погрешность сохранения энергии при шаге интегрирования 0.0001[s]

Пример 2.

Три тела, связанные вращательными шарнирами, в поле силы тяжести. Пример содержится в файле «body_3_rot_1.elr». Вид модели примера представлен на рисунке 2.1. Графики погрешностей сохранения энергии системы при различных шагах численного интегрирования представлены на рисунках 2.2, 2.3, 2.4.

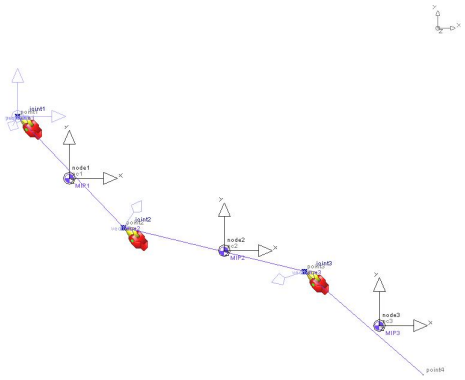

Рис. 2.1. Вид модели примера 2 (файл «body_3_rot_1.elr»)

Текст EULER-проекта примера 2 (файл «body_2_rot_1.elr»):

```
// Тест на сохранение энергии консервативной механической системы.
//    Три тела, связанные вращательными шарнирами, в поле силы тяжести.
//
scalar m=1[kg];
scalar Ix=0.001[kg m2];
scalar Iy=0.01[kg m2];
scalar Iz=0.02[kg m2];
tensor tensor1=mainTensor( Ix, Iy, Iz );
scalar g=10[m/s2];
//
point point1=point( 0 [ m ], 0 [ m ], 0 [ m ] );
point point2=point( 4.1000e-001 [ m ], -4.3000e-001 [ m ], 0.0000e+000 [ m ] );
point point3=point( 1.1100e+000 [ m ], -6.0000e-001 [ m ], 0.0000e+000 [ m ] );
point point4=point( 1.5700e+000 [ m ], -1.0000e+000 [ m ], 0.0000e+000 [ m ] );
line line1=polyLine( list( point1, point2 ) );
line line2=polyLine( list( point2, point3 ) );
line line3=polyLine( list( point3, point4 ) );
point pc1=point( 2.0000e-001 [ m ], -2.4000e-001 [ m ], 0.0000e+000 [ m ] );
point pc2=point( 8.0000e-001 [ m ], -5.2000e-001 [ m ], 0.0000e+000 [ m ] );
point pc3=point( 1.4000e+000 [ m ], -8.1000e-001 [ m ], 0.0000e+000 [ m ] );
node node1=nodePoint( pc1 );
node node2=nodePoint( pc2 );
node node3=nodePoint( pc3 );
MIP MIP1=massNode( node1, m, tensor1 );
MIP MIP2=massNode( node2, m, tensor1 );
MIP MIP3=massNode( node3, m, tensor1 );
//
color color_base=index( 6 );
color color_G=index( 72 );
body base=body( color = color_base );
set ground = base;
body G1=body( color = color_G );
body G1 < ( line1, MIP1 );
body G2=body( color = color_G );
```

```
body G2 < ( line2, MIP2 );
body G3=body( color = color_G );
body G3 < ( line3, MIP3 );
//
vector vector1=vector( -1 [ m ], -2 [ m ], -7 [ m ], point = point1 );
vector vector2=vector( 2 [ m ], 3 [ m ], 4 [ m ], point = point2 );
vector vector3=vector( -3 [ m ], -1 [ m ], 3 [ m ], point = point3 );
joint joint1=rotational( base, G1, point1, vector1 );
joint joint2=rotational( G1, G2, point2, vector2 );
joint joint3=rotational( G2, G3, point3, vector3 );
//
gravity gravity1=parallel( projectY, g = g );
//
sensor Vx1=transVelocity( base, projectX, G1, pc1 );
sensor Vy1=transVelocity( base, projectY, G1, pc1 );
sensor Vz1=transVelocity( base, projectZ, G1, pc1 );
sensor Vx2=transVelocity( base, projectX, G2, pc2 );
sensor Vy2=transVelocity( base, projectY, G2, pc2 );
sensor Vz2=transVelocity( base, projectZ, G2, pc2 );
sensor Vx3=transVelocity( base, projectX, G3, pc3 );
sensor Vy3=transVelocity( base, projectY, G3, pc3 );
sensor Vz3=transVelocity( base, projectZ, G3, pc3 );
sensor Wx1=rotVelocity( G1, vectorX( node1, visible = hide: ), base );
sensor Wy1=rotVelocity( G1, vectorY( node1, visible = hide: ), base );
sensor Wz1=rotVelocity( G1, vectorZ( node1, visible = hide: ), base );
sensor Wx2=rotVelocity( G2, vectorX( node2, visible = hide: ), base );
sensor Wy2=rotVelocity( G2, vectorY( node2, visible = hide: ), base );
sensor Wz2=rotVelocity( G2, vectorZ( node2, visible = hide: ), base );
sensor Wx3=rotVelocity( G3, vectorX( node3, visible = hide: ), base );
sensor Wy3=rotVelocity( G3, vectorY( node3, visible = hide: ), base );
sensor Wz3=rotVelocity( G3, vectorZ( node3, visible = hide: ), base );
sensor EK1 "кинетическая энергия G1"=m*(Vx1*Vx1+Vy1*Vy1+Vz1*Vz1)/2+
        Ix*Wx1*Wx1/2[rad2]+Iy*Wy1*Wy1/2[rad2]+Iz*Wz1*Wz1/2[rad2];
sensor EK2 "кинетическая энергия G2"=m*(Vx2*Vx2+Vy2*Vy2+Vz2*Vz2)/2+
        Ix*Wx2*Wx2/2[rad2]+Iy*Wy2*Wy2/2[rad2]+Iz*Wz2*Wz2/2[rad2];
sensor EK3 "кинетическая энергия G3"=m*(Vx3*Vx3+Vy3*Vy3+Vz3*Vz3)/2+
        Ix*Wx3*Wx3/2[rad2]+Iy*Wy3*Wy3/2[rad2]+Iz*Wz3*Wz3/2[rad2];
sensor h1=bodyDisplacement( base, pc1, projectY, G1, pc1 );
sensor h2=bodyDisplacement( base, pc2, projectY, G2, pc2 );
sensor h3=bodyDisplacement( base, pc3, projectY, G3, pc3 );
sensor EP "потенциальная энергия"=-m*g*h1-m*g*h2-m*g*h3;
sensor ES "суммарная энергия"=EK1+EK2+EK3+EP;
sensor Ebase "примерный базовый уровень для контроля ошибки"=10[kg m2/s2];
//
sensor energy_REL_ERROR "относительная ошибка энергии"=ES/Ebase;
//

/\////////////////////////////////////////////////////////////////////////////////
/\   Единицы измерения;
set units = SI;
```

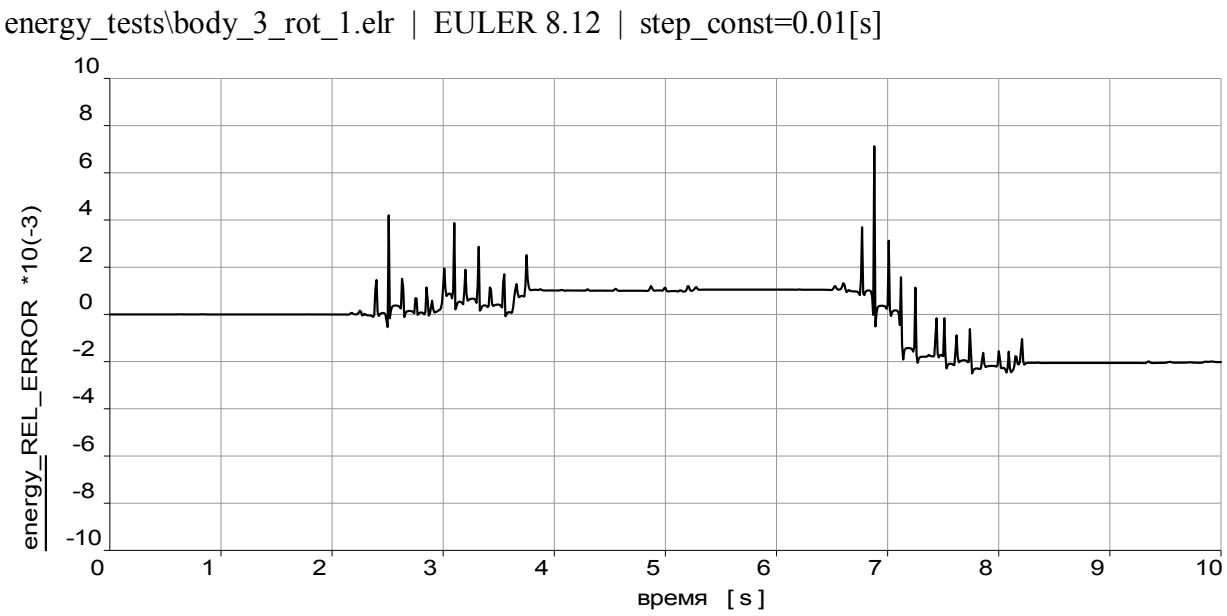

Рис. 2.2. Погрешность сохранения энергии при шаге интегрирования 0.01[s]

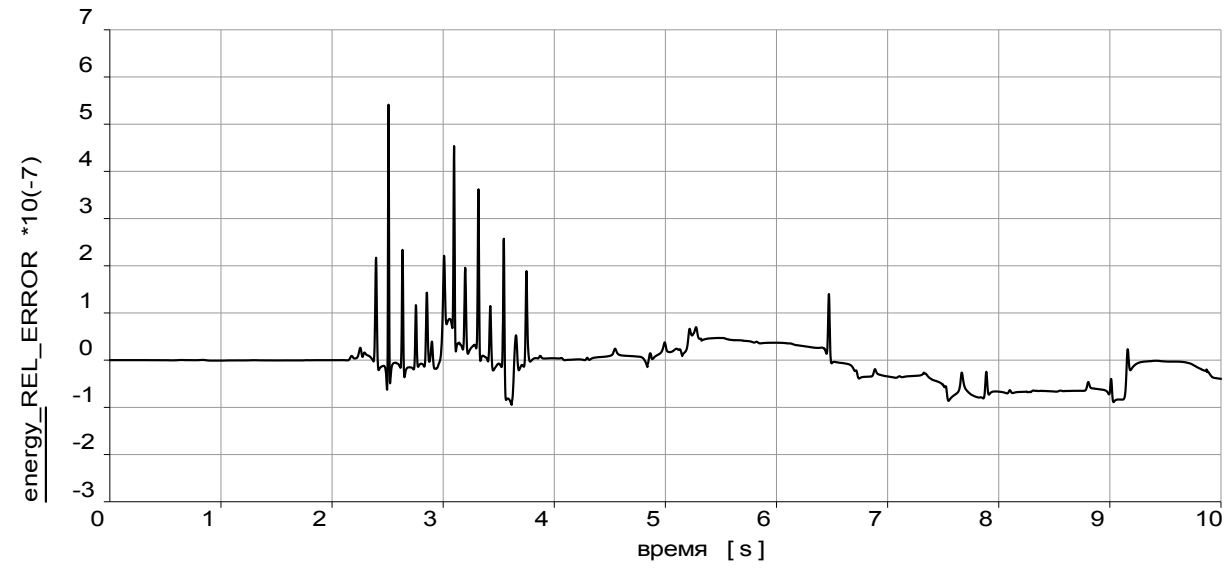

Рис. 2.3. Погрешность сохранения энергии при шаге интегрирования 0.001[s]

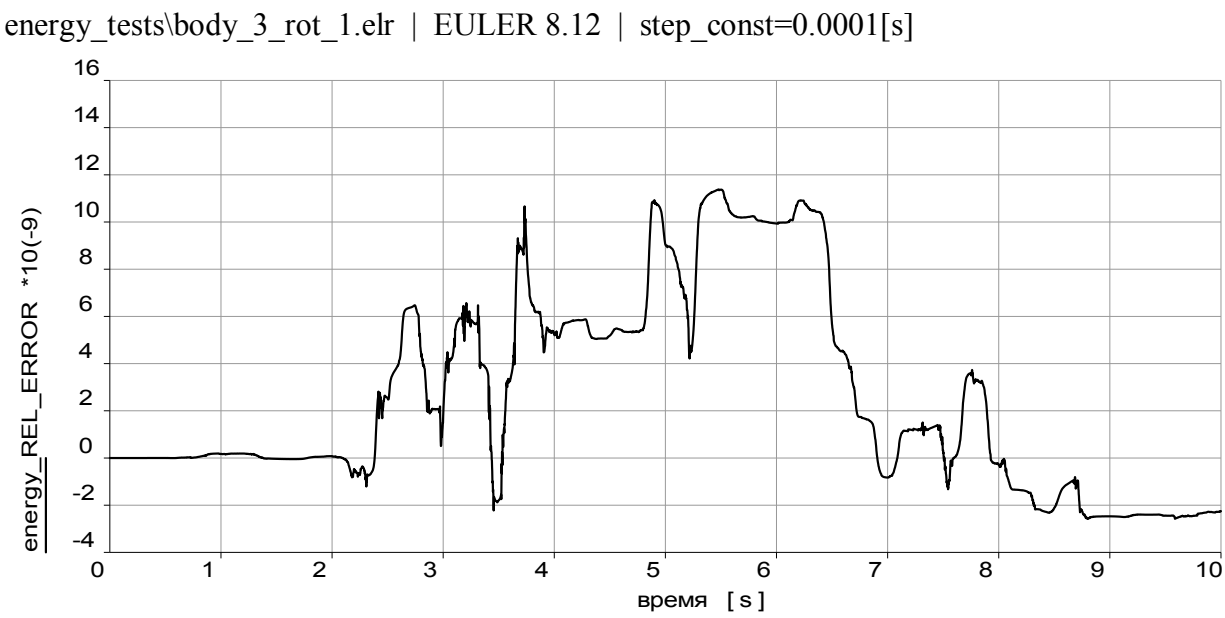

Рис. 2.4. Погрешность сохранения энергии при шаге интегрирования 0.0001[s]

Пример 3.

Неуравновешенный гироскоп в поле силы тяжести. Пример содержится в файле «gyroscope_1.elr». Вид модели примера представлен на рисунке 3.1. Графики погрешностей сохранения энергии системы при различных шагах численного интегрирования представлены на рисунках 3.2, 3.3, 3.4.

Рис. 3.1. Вид модели примера 3 (файл «gyroscope_1.elr»)

Текст EULER-проекта примера 3 (файл «gyroscope_1.elr»):

```
// Тест на сохранение энергии консервативной механической системы.
//    Неуравновешенный гироскоп в поле силы тяжести.
//
scalar W "угловая скорость закрутки гироскопа"=1000/(2*PI)*1[rad/s];
scalar m=1[kg];
scalar I=0.0001[kg m2];
scalar g=10[m/s2];
scalar L=0.1[m];
//
point point1=point( 0 [ m ], 0 [ m ], 0 [ m ] );
point point2=point( L, 0 [ m ], 0 [ m ] );
line line1=polyLine( list( point1, point2 ) );
solid solid1=sphere( point2, L/10 );
tensor tensor1=mainTensor( I, I, I );
node node1=nodePoint( point2 );
MIP MIP1=massNode( node1, m, tensor1 );
//
color color_base=index( 6 );
color color_G=index( 72 );
body base=body( color = color_base );
set ground = base;
body G=body( color = color_G );
body G < ( MIP1, line1, solid1 );
//
joint joint1=spherical( base, G, point1 );
//
condition condition1=rotVelocity( base, projectX, G, W );
//
gravity gravity1=parallel( projectY, g = g );
//
sensor Vx=transVelocity( base, projectX, G, point2 );
sensor Vy=transVelocity( base, projectY, G, point2 );
sensor Vz=transVelocity( base, projectZ, G, point2 );
sensor Vs=sqrt(Vx*Vx+Vy*Vy+Vz*Vz);
sensor Wx=rotVelocity( base, projectX, G );
sensor Wy=rotVelocity( base, projectY, G );
sensor Wz=rotVelocity( base, projectZ, G );
sensor Ws=sqrt(Wx*Wx+Wy*Wy+Wz*Wz)/1[rad];
sensor EK "кинетическая энергия"=m*Vs*Vs/2+I*Ws*Ws/2;
sensor h=bodyDisplacement( base, point2, projectY, G, point2 );
sensor EP "потенциальная энергия"=-m*g*h;
sensor ES "суммарная энергия"=EK+EP;
```

```
scalar ES_0=I*W*W/2/1[rad2];
//
sensor energy_REL_ERROR "относительная ошибка энергии"=(ES-ES_0)/ES_0;
//
```

energy_tests\ gyroscope_1.elr | EULER 8.12 | step_const=0.01[s]

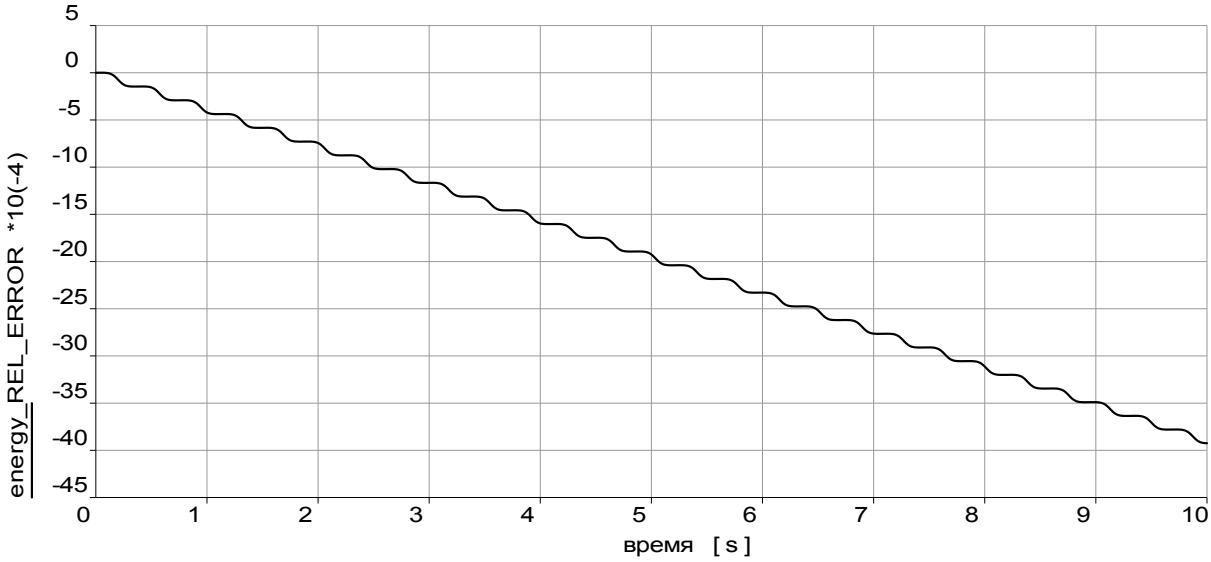

Рис. 3.2. Погрешность сохранения энергии при шаге интегрирования 0.01[s]

energy_tests\ gyroscope_1.elr | EULER 8.12 | step_const=0.001[s]

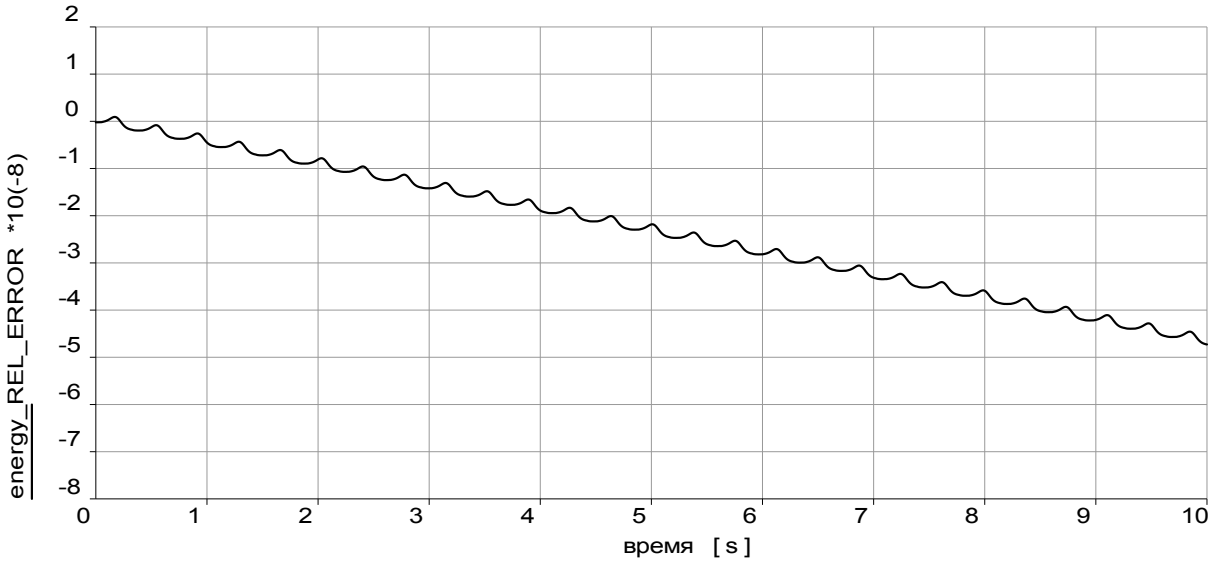

Рис. 3.3. Погрешность сохранения энергии при шаге интегрирования 0.001[s]

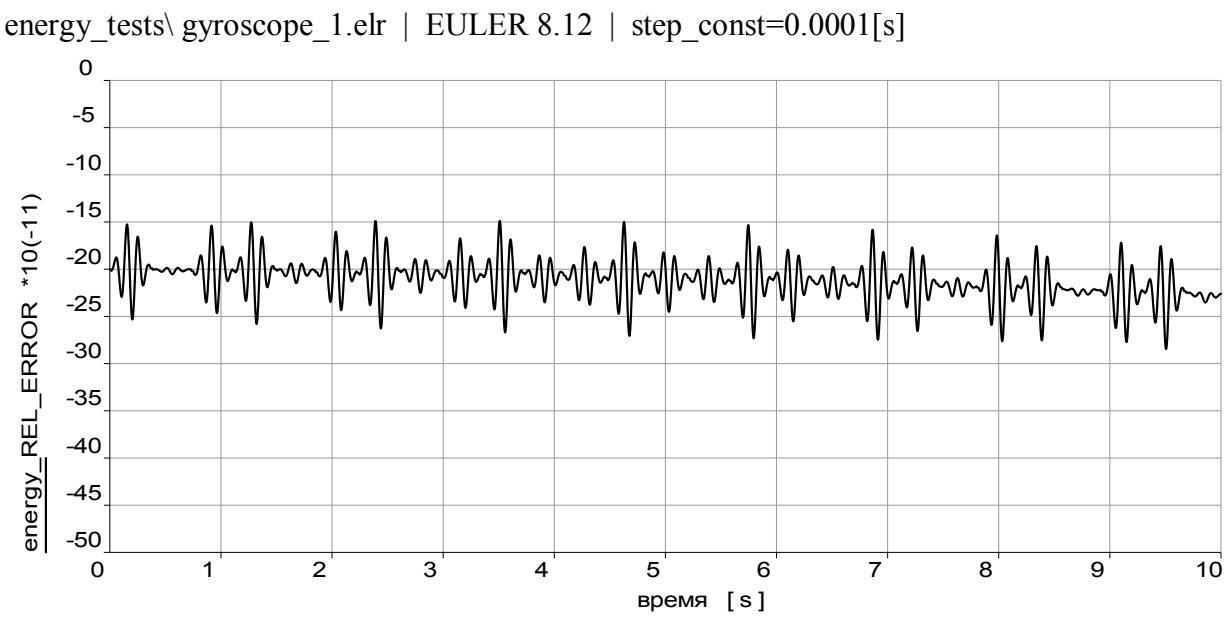

Рис. 3.4. Погрешность сохранения энергии при шаге интегрирования 0.0001[s]