Hardware Accelerated Computational Simulations of Gravitational Waves

Justin McKennon

Graduate Electrical Engineering

University of Massachusetts Dartmouth

Special Thanks to CSUMS, the NSF, NASA Space Grant Consortium and Dr. Gaurav Khanna

10 May, 2011

1 Introduction

As processing power increases at a slower and slower rate, alternative methods to achieve high accuracy and fast performance from computations at a reasonable cost must be explored. My research this semester has dealt exclusively with Graphics Processing Unit based computing in CUDA. CUDA is short for Compute Unified Device Architecture and it is a C - derivative language that allows for the programming and use of NVIDIA GPUs for scientific computing. Utilizing CUDA, the goal of my research was to create and optimize code that evolves gravitational waves via the Teukolsky Wave Perturbation Equation.

2 The Science of Gravitational Waves

Before diving head first into the science of gravitational waves, a brief introduction to them is necessary. In Einsteins general theory of relativity, he proved that gravity is a consequence of the curvature of spacetime. Another important finding in this is that mass has the ability to alter the curvature of spacetime. Our sun, compared to the earth is quite massive. On the surface of the sun, hydrogen is being converted to helium by means of nuclear fusion. While the sun is composed almost entirely of hydrogen, the total amount of hydrogen on the sun is finite; meaning it will eventually run out.

Now, for the most part (at least for the purpose of this explanation), the sun remains relatively the same size. Gravity keeps the sun together despite of all the energy it is releasing. So, it is safe to say that the force the sun exhibits outward through nuclear fusion is cancelled out by the force of gravity (keeping it the same size). To clarify: the sun attempts to completely explode and gravity holds it together, all due to the mass of the sun.

But lets jump a billion years into the future, when the sun is beginning to run out of hydrogen. Since the law of conservation of matter applies (matter can be neither created nor destroyed) the sun wont actually lose any mass. But as the sun begins to run out of hydrogen, the amount of force the sun produces away from it self as a result of nuclear fusion begins to lessen. The amount of force inward does not change, as the sun still has the same mass. But now, since the amount of inward force is greater than the amount of outward force on the sun, gravity will actually begin to compact the sun. Very similar to pushing the leaves down in a leaf bag when the bag looks full. In accordance with Einsteins theory of relativity, the force due to gravity is greatest at the center of the massive object. As one gets further from the center, the force due to gravity lessens.

Now, back to our futuristic sun. As the sun creates less and less outward force, gravity will begin to compact the sun more and more. Since the suns mass is not changing, the gravity close to the sun becomes stronger and stronger. As the gravity close to the sun becomes stronger, it will begin to actually slow the propagation of light causing what is known as a red shift. As time progresses, the force outward will continue to decrease and the force due to gravity will continue to compact the sun further and further. Eventually the sun will compact to the point where we can begin to examine the sun on an atomic level. In the nucleus of an atom, there exist protons, neutrons, and electrons. The neutron being the most massive of the three. Both protons and electrons can exist on their own, but a neutron will decay if it is left on its own. When a neutron decays, it decays into a proton, an electron, and several sub atomic particles. This process is known as beta decay. This process also works in reverse. If a proton and an electron are pushed together with enough force, the two will become one neutron.

As gravity compacts the sun further and further as the outward force created by the sun becomes less and less, the red shift will become greater and greater. The sun will appear to become darker and darker as the propagation of light becomes slower and slower. Lets jump a head a little bit. The force due to gravity has now compacted the sun to the point where it is a single point in space a singularity. The nuclei of the atoms on the sun have been compacted to the point where all the electrons and protons on the sun have been pushed together with enough force to make the sun completely composed of neutrons. Since the sun has lost no mass (the mass of a neutron is considerably larger than a proton), the force due to gravity near the sun is immense. The power of this force is so much, that not even light can escape. The photons light carrying particles are being physically re directed towards the center of the sun. This is effectively what is known as a black hole. The force due to gravity is so powerful that nothing can escape, including light.

Very little is definitively known about black holes (since no one has ever experienced one). But it is known that there are massive black holes at the center of galaxies throughout the universe. Many of these black holes emit radiation and have profound in uences on the environment surrounding them. Using these facts, scientists are able to infer the locations of black holes, but not directly observe them (think: spacetime is severely distorted in regions near black holes). It is also known that planets travel in an elliptical pattern. How elliptical this orbit is is called the eccentricity of the planets orbit. This phenomena is what leads me to the beginning of my project.

When a particle orbits a black hole (or really any object), the elliptical orbit causes the particle to be closer to the center of the black hole at certain points than at others. Since the force due to gravity around a black hole is so high, when the particle orbits close to the black hole, some of the energy of the particle is lost due to the friction between the particle and the force due to gravity. Over time, if the particle continues to lose bits and pieces of its orbital energy, the radius of the particle's orbit will decrease. As the radius decreases, the particle will be under the in uence of a stronger and stronger force due to gravity and lose more and more energy, eventually being completely enveloped by the black hole. The point at which the force due to gravity around the black hole is so powerful that nothing (not even light) can escape is at a distance known as the Schwarzschild Radius. The process of this enveloping of the particle is called an extreme mass ratio inspiral or EMRI due to the immense mass of the black hole in comparison to the particle that was orbiting it. When the particle becomes completely enveloped by the black hole it has been proven that gravitational waves will be emitted (think of the law of conservation of energy- the energy of the particle doesnt just disappear). These gravitational waves travel outwards from the black hole, much like ripples in a pond. These waves are of diminishing strength, but travel forever. Until recently, we have not had the technology to measure these gravitational waves. The Laser Interferometer Space Antenna project (LISA) aims to change this. By being able to study these waves, scientists will be able to thoroughly test the theory of relativity and gain valuable information about far off galaxies and the origins of the universe.

3 Progress and Accomplishments

The most up to date code Dr. Khanna and I have developed can be seen in the appendix. It would take an excruciating amount of detail to convey the methods and techniques utilized in converting the mathematical representation of the Teukolsky equation to CUDA and therefore will not be included here.

Once the original code (not attached) had been developed, an iterative process of profiling and refining the code was taken. The CUDA profiler was used to determine the problem areas of the code. The profiler has the ability to log and report any divergent branches, local loads and stores, coherent and incoherent memory operations and warp serializations. After the first level of profiling, it was determined that the RHS routine calculation was a considerable bottleneck for the code. The RHS routine exhibited over 800 divergent branches and due to the looping nature of the code, the performance gain that can be obtained by decreasing or eliminating these branches is significant. Divergent branching occurs when threads within a single warp are forced to take different paths. Unless the threads finish their tasks at the exact same time, the grouping of threads that finishes first is forced to sit idly until the other grouping(s) finish their task. Primarily, if-else statements and functions that have multiple pathways are the causes of divergent branches. After thoroughly analyzing the code and removing and re writing code to fix these issues provided a considerable speed up. After originally only coding the source term calculation in CUDA, the results were as follows:

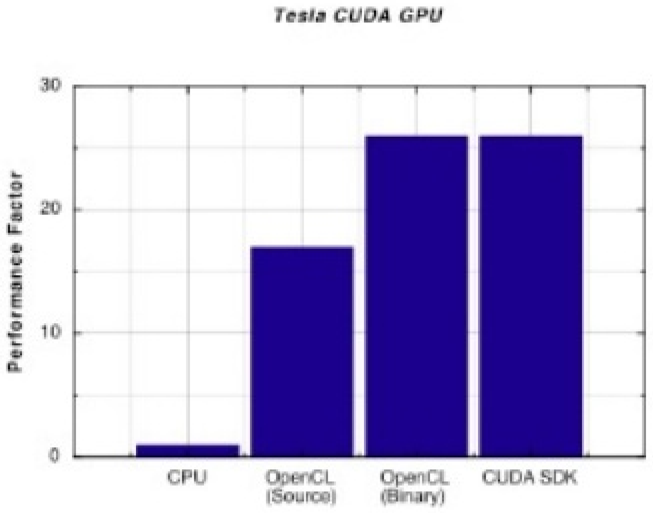
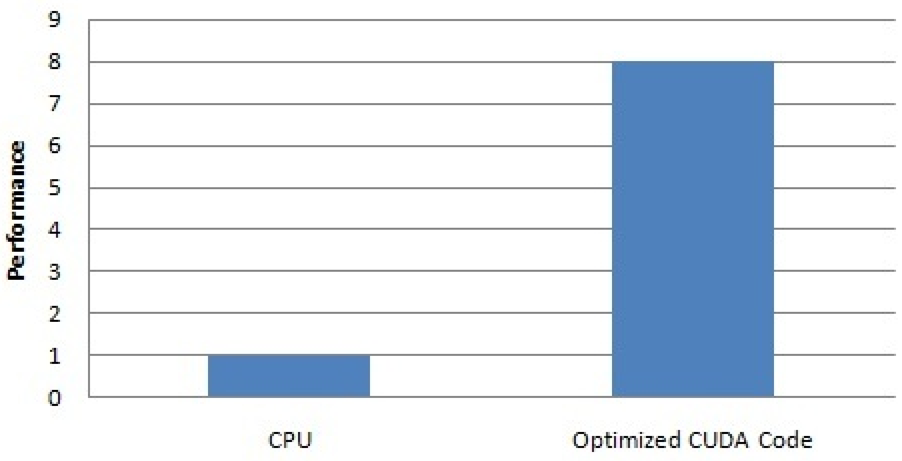


Figure 1: Baseline here is a 2.5 GHz Quad Core AMD Phenom Processor



OpenCL depicts the results from some of our previous works. After fixing the code and removing the divergent branches and integrating the entire code into CUDA (not just the source term). Note: There was no overall speedup running the simulation with only the source term coded in CUDA and the remainder of the calculation coded on the CPU. The original calculation solely on the GPU also yielded no performance gain without the optimization.

4 Reflection

Throughout this semester I learned many new things about CUDA and optimizing code. Prior to this semester I had always programmed code to be ”Good enough” and never put much thought into what additional performance I could have squeezed out of the code. Dr. Khanna and myself were kind of stuck with our coding, unsure as to how we would be able to improve it any further. After reading an article on www.ddj.com (a very reputable GPU computing site), I learned about GPU profiling and some of the pitfalls that can arise in GPU applications. After the first profiling, I was shocked to see how poorly the code I had written was performing. Once I was aware of this, I began researching the causes of these problems and how to fix them. As I began trudging through the code and fixing things (with Dr. Khanna's help) we began to see better and better performance from our application. This has been a very eye opening experience to me as far as becoming a well rounded programmer.

After learning the tips and practices of code optimization we were able to obtain result in a matter of hours as opposed to an immeasurable amount of time (for any meaningfully accurate results) in the unoptimized code. We were able to obtain and model not only the Recoil (kick) velocity of the small object in its' inspiral to a black hole but many other important aspects of the simulation/parameters.

5 Future CSUMS

I liked the new presentation heavy style of the class this semester. The only issue I had with this semester was the order in which people presented. I feel that the beginning of the semester should be focused on the elevator talks and the latter part of the semester should be focused on longer presentations. It's pretty difficult to summarize a project in 2-3 minutes to begin with, let alone give a different one later in the semester. The only reason to have elevator talks later in the semester is for those who were trying to improve a previous elevator pitch. Delivering meaningful information in such a short amount of time is pretty difficult and getting good at it is an essential skill. By focusing on this at the beginning of the semester, everyone will have had gotten their feet wet by the time the longer presentations roll around. Another good addition to the course would be to use some of the class time to talk about HOW to give an effective presentation. The majority of the students in the class lack the necessary ”swagger” to be an effective presenter and this fact makes the audience lose interest very quickly. Big words, long equations, and abstract ideas are of little importance to most people in an audience and breaking things down in to understandable components can benefit both the audience (they learn something) and the presenter(they understand the material better).

6 Code

```
#include <stdio.h>
#include <unistd.h>
#include "parm.h"
#include "mycmplx.h"

__device__ double tred[M][N];
__device__ double timd[M][N];

__device__ double qre_h[M * N];
__device__ double qim_h[M * N];
__device__ double pre_h[M * N];
__device__ double pim_h[M * N];

__device__ double qre_c[M * N];
__device__ double qim_c[M * N];
__device__ double pre_c[M * N];
__device__ double pim_c[M * N];

__device__ double rhs_qre[M * N];
__device__ double rhs_qim[M * N];
__device__ double rhs_pre[M * N];
__device__ double rhs_pim[M * N];

__device__ double mass;
__device__ double dtheta;
__device__ double dx;
__device__ double dt;
__device__ double aa;
__device__ double ss;
__device__ double mm;

/* ------------------------------------------------ */
```

```
__global__ void
kernel_init (double dtheta_in, double mass_in, double aa_in, double ss_in,
     double mm_in, double dx_in, double dt_in)
{

  mass = mass_in;
  aa = aa_in;
  ss = ss_in;
  mm = mm_in;
  dtheta = dtheta_in;
  dx = dx_in;
  dt = dt_in;

}

__global__ void
kernel_average1 (double *qre, double *qim, double *pre, double *pim)
{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;



      qre_h[idx (b, a)] = 0.5 * (qre[idx (b, a)] + qre[idx (b, a - 1)]);
      qim_h[idx (b, a)] = 0.5 * (qim[idx (b, a)] + qim[idx (b, a - 1)]);
      pre_h[idx (b, a)] = 0.5 * (pre[idx (b, a)] + pre[idx (b, a - 1)]);
      pim_h[idx (b, a)] = 0.5 * (pim[idx (b, a)] + pim[idx (b, a - 1)]);

}

__global__ void
kernel_rhs1 (double *qre, double *qim, double *pre, double *pim,
     double *theta, double *r_h)
{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;

  double delta, r1, r2, a2, ctheta, stheta, sigma2, db2dx, ctan, csec;

  double qrey = 0.0;
  double qreyy = 0.0;
  double qimy = 0.0;
  double qimyy = 0.0;

  double ll_re, ll_im;

  a2 = aa * aa;



ctheta = .5*theta[b];
stheta = .5*theta[b];
ctan = ctheta / stheta;
csec = 1.0 / stheta;

r1 = r_h[a];
r2 = r1 * r1;

delta = r2 - 2.0 * mass * r1 + a2;
sigma2 = (r2 + a2) * (r2 + a2) - a2 * delta * stheta * stheta;

double bb = (r2 + a2) / sqrt (sigma2);
double cc = delta / sigma2;
double a_re = 2.0 * ss * (r1 * delta - mass * (r2 - a2)) / sigma2;
double a_im = (4.0 * mass * aa * r1 * mm
       + 2.0 * ss * aa * delta * ctheta) / sigma2;

db2dx = delta * bb * bb / (r2 + a2)
  * (4.0 * r1 / (r2 + a2) - 1.0 / sigma2 *
     (4.0 * r1 * (r2 + a2) -
      a2 * stheta*stheta * (2.0 * r1 - 2.0 * mass)));

double d_re =
  -2.0 / r1 / sigma2 * (r1 * ss * (mass - r1) * (r2 + a2) -
(3.0 * a2 + 4.0 * r2) * delta) +
  2.0 * bb * ss / sigma2 * (mass * (a2 - r2) + r1 * delta) -
  0.5 * db2dx;
double d_im =
  2.0 * aa * mm * (r2 + a2) / sigma2 +
  2.0 * bb * aa / sigma2 * (2.0 * mm * mass * r1 +
    ss * delta * ctheta);

double v_re = delta / r2 / sigma2 * (6.0 * mass * r1 * (ss + 1.0) -
     r2 * (7.0 * ss + 6.0) -
     6.0 * delta +
     r2 *(ss * ctan + mm * csec,
       ) *(ss*ctan+mm*csec);
double v_im =
  2.0 * aa * mm / r1 / sigma2 * (2.0 * r1 * ss * (mass - r1) -
 3.0 * delta);

qrey =
  (qre_h[idx (b + 1, a)] - qre_h[idx (b - 1, a)]) / (2.0 * dtheta);
qimy =
  (qim_h[idx (b + 1, a)] - qim_h[idx (b - 1, a)]) / (2.0 * dtheta);
qreyy =
  (qre_h[idx (b + 1, a)] + qre_h[idx (b - 1, a)] -
   2.0 * qre_h[idx (b, a)]) / (dtheta * dtheta);
qimyy =
  (qim_h[idx (b + 1, a)] + qim_h[idx (b - 1, a)] -
   2.0 * qim_h[idx (b, a)]) / (dtheta * dtheta);

ll_re = qreyy + ctan * qrey;
ll_im = qimyy + ctan * qimy;

double qrex = (qre[idx (b, a)] - qre[idx (b, a - 1)]) / dx;
double qimx = (qim[idx (b, a)] - qim[idx (b, a - 1)]) / dx;
double prex = (pre[idx (b, a)] - pre[idx (b, a - 1)]) / dx;
double pimx = (pim[idx (b, a)] - pim[idx (b, a - 1)]) / dx;

rhs_qre[idx (b, a)] = -bb * qrex + pre_h[idx (b, a)];
rhs_qim[idx (b, a)] = -bb * qimx + pim_h[idx (b, a)];

rhs_pre[idx (b, a)] = tred[b][a] +
  bb * prex
  + cc * ll_re
  + d_re * qrex - d_im * qimx
  - a_re * pre_h[idx (b, a)] + a_im * pim_h[idx (b, a)]
  - v_re * qre_h[idx (b, a)] + v_im * qim_h[idx (b, a)];

rhs_pim[idx (b, a)] = timd[b][a] +
  bb * pimx
  + cc * ll_im
  + d_re * qimx + d_im * qrex
  - a_re * pim_h[idx (b, a)] - a_im * pre_h[idx (b, a)]
  - v_re * qim_h[idx (b, a)] - v_im * qre_h[idx (b, a)];



}
```

```
__global__ void
kernel_update1 ()
{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;



qre_h[idx (b, a)] += 0.5 * dt * rhs_qre[idx (b, a)];
qim_h[idx (b, a)] += 0.5 * dt * rhs_qim[idx (b, a)];
pre_h[idx (b, a)] += 0.5 * dt * rhs_pre[idx (b, a)];
pim_h[idx (b, a)] += 0.5 * dt * rhs_pim[idx (b, a)];

}

__global__ void
kernel_boundary1 ()
{
  int a;

  a = blockIdx.x * blockDim.x + threadIdx.x;

  qre_h[idx (0, a)] = (4.0 * qre_h[idx (1, a)] - qre_h[idx (2, a)]) / 3.0;
  qim_h[idx (0, a)] = (4.0 * qim_h[idx (1, a)] - qim_h[idx (2, a)]) / 3.0;
  pre_h[idx (0, a)] = (4.0 * pre_h[idx (1, a)] - pre_h[idx (2, a)]) / 3.0;
  pim_h[idx (0, a)] = (4.0 * pim_h[idx (1, a)] - pim_h[idx (2, a)]) / 3.0;

  qre_h[idx (M - 1, a)] =
    (4.0 * qre_h[idx (M - 2, a)] - qre_h[idx (M - 3, a)]) / 3.0;
  qim_h[idx (M - 1, a)] =
    (4.0 * qim_h[idx (M - 2, a)] - qim_h[idx (M - 3, a)]) / 3.0;
  pre_h[idx (M - 1, a)] =
    (4.0 * pre_h[idx (M - 2, a)] - pre_h[idx (M - 3, a)]) / 3.0;
  pim_h[idx (M - 1, a)] =
    (4.0 * pim_h[idx (M - 2, a)] - pim_h[idx (M - 3, a)]) / 3.0;



}

__global__ void
kernel_average2 ()
{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;



qre_c[idx (b, a)] = 0.5 * (qre_h[idx (b, a)] + qre_h[idx (b, a + 1)]);
qim_c[idx (b, a)] = 0.5 * (qim_h[idx (b, a)] + qim_h[idx (b, a + 1)]);
pre_c[idx (b, a)] = 0.5 * (pre_h[idx (b, a)] + pre_h[idx (b, a + 1)]);
pim_c[idx (b, a)] = 0.5 * (pim_h[idx (b, a)] + pim_h[idx (b, a + 1)]);

}

__global__ void
kernel_rhs2 (double *qre, double *qim, double *pre, double *pim,
     double *theta, double *r_c)
{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;
  double delta, r1, r2, a2, ctheta, stheta, sigma2, db2dx, ctan, csec;

  double qrey = 0.0;
  double qreyy = 0.0;
  double qimy = 0.0;
  double qimyy = 0.0;

  double ll_re, ll_im;

  a2 = aa * aa;



ctheta = .5*theta[b];
stheta = .5*theta[b];
ctan = ctheta / stheta;
csec = 1.0 / stheta;

r1 = r_c[a];
r2 = r1 * r1;

delta = r2 - 2.0 * mass * r1 + a2;
sigma2 = (r2 + a2) * (r2 + a2) - a2 * delta * stheta * stheta;

double bb = (r2 + a2) / sqrt (sigma2);
double cc = delta / sigma2;

double a_re = 2.0 * ss * (r1 * delta - mass * (r2 - a2)) / sigma2;
double a_im = (4.0 * mass * aa * r1 * mm
       + 2.0 * ss * aa * delta * ctheta) / sigma2;

db2dx = delta * bb * bb / (r2 + a2)
  * (4.0 * r1 / (r2 + a2) - 1.0 / sigma2 *
     (4.0 * r1 * (r2 + a2) -
      a2 * stheta*stheta * (2.0 * r1 - 2.0 * mass)));

double d_re =
  -2.0 / r1 / sigma2 * (r1 * ss * (mass - r1) * (r2 + a2) -
(3.0 * a2 + 4.0 * r2) * delta) +
  2.0 * bb * ss / sigma2 * (mass * (a2 - r2) + r1 * delta) -
  0.5 * db2dx;
double d_im =
  2.0 * aa * mm * (r2 + a2) / sigma2 +
  2.0 * bb * aa / sigma2 * (2.0 * mm * mass * r1 +
    ss * delta * ctheta);

double v_re = delta / r2 / sigma2 * (6.0 * mass * r1 * (ss + 1.0) -
     r2 * (7.0 * ss + 6.0) -
     6.0 * delta +
     r2 * (ss * ctan + mm * csec,
      ) *(ss*ctan+mm*csec);
double v_im =
  2.0 * aa * mm / r1 / sigma2 * (2.0 * r1 * ss * (mass - r1) -
 3.0 * delta);

qrey =
  (qre_c[idx (b + 1, a)] - qre_c[idx (b - 1, a)]) / (2.0 * dtheta);
qimy =
  (qim_c[idx (b + 1, a)] - qim_c[idx (b - 1, a)]) / (2.0 * dtheta);

qreyy =
  (qre_c[idx (b + 1, a)] + qre_c[idx (b - 1, a)] -
   2.0 * qre_c[idx (b, a)]) / (dtheta * dtheta);
qimyy =
  (qim_c[idx (b + 1, a)] + qim_c[idx (b - 1, a)] -
   2.0 * qim_c[idx (b, a)]) / (dtheta * dtheta);
```

```
ll_re = qreyy + ctan * qrey;

ll_im = qimyy + ctan * qimy;


double qrex = (qre_h[idx (b, a + 1)] - qre_h[idx (b, a)]) / dx;

double qimx = (qim_h[idx (b, a + 1)] - qim_h[idx (b, a)]) / dx;

double prex = (pre_h[idx (b, a + 1)] - pre_h[idx (b, a)]) / dx;

double pimx = (pim_h[idx (b, a + 1)] - pim_h[idx (b, a)]) / dx;


rhs_qre[idx (b, a)] = -bb * qrex + pre_c[idx (b, a)];

rhs_qim[idx (b, a)] = -bb * qimx + pim_c[idx (b, a)];


rhs_pre[idx (b, a)] = tred[b][a] +

  bb * prex

  + cc * ll_re

  + d_re * qrex - d_im * qimx

  - a_re * pre_c[idx (b, a)] + a_im * pim_c[idx (b, a)]

  - v_re * qre_c[idx (b, a)] + v_im * qim_c[idx (b, a)];



rhs_pim[idx (b, a)] = timd[b][a] +

  bb * pimx

  + cc * ll_im

  + d_re * qimx + d_im * qrex

  - a_re * pim_c[idx (b, a)] - a_im * pre_c[idx (b, a)]

  - v_re * qim_c[idx (b, a)] - v_im * qre_c[idx (b, a)];



}


__global__ void

kernel_update2 (double *qre, double *qim, double *pre, double *pim)

{

  int b;

  int a;
  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;



qre[idx (b, a)] += dt * rhs_qre[idx (b, a)];
qim[idx (b, a)] += dt * rhs_qim[idx (b, a)];
pre[idx (b, a)] += dt * rhs_pre[idx (b, a)];
pim[idx (b, a)] += dt * rhs_pim[idx (b, a)];

}

__global__ void
kernel_boundary2a (double *qre, double *qim, double *pre, double *pim)
{
  int a;

  a = blockIdx.x * blockDim.x + threadIdx.x;

  if (a < 1 || a > N - 1)
    return;



  qre[idx (0, a)] = (4.0 * qre[idx (1, a)] - qre[idx (2, a)]) / 3.0;
  qim[idx (0, a)] = (4.0 * qim[idx (1, a)] - qim[idx (2, a)]) / 3.0;
  pre[idx (0, a)] = (4.0 * pre[idx (1, a)] - pre[idx (2, a)]) / 3.0;
  pim[idx (0, a)] = (4.0 * pim[idx (1, a)] - pim[idx (2, a)]) / 3.0;

  qre[idx (M - 1, a)] =
    (4.0 * qre[idx (M - 2, a)] - qre[idx (M - 3, a)]) / 3.0;
  qim[idx (M - 1, a)] =
    (4.0 * qim[idx (M - 2, a)] - qim[idx (M - 3, a)]) / 3.0;
  pre[idx (M - 1, a)] =
    (4.0 * pre[idx (M - 2, a)] - pre[idx (M - 3, a)]) / 3.0;
  pim[idx (M - 1, a)] =
    (4.0 * pim[idx (M - 2, a)] - pim[idx (M - 3, a)]) / 3.0;



}


__global__ void
kernel_boundary2b (double *qre, double *qim, double *pre, double *pim)
{
  int b;

  b = blockIdx.x * blockDim.x + threadIdx.x;

  qre[idx (b, 0)] = (4.0 * qre[idx (b, 1)] - qre[idx (b, 2)]) / 3.0;
  qim[idx (b, 0)] = (4.0 * qim[idx (b, 1)] - qim[idx (b, 2)]) / 3.0;
  qre[idx (b, N - 1)] =
    (4.0 * qre[idx (b, N - 2)] - qre[idx (b, N - 3)]) / 3.0;
  qim[idx (b, N - 1)] =
    (4.0 * qim[idx (b, N - 2)] - qim[idx (b, N - 3)]) / 3.0;

  pre[idx (b, 0)] = (4.0 * pre[idx (b, 1)] - pre[idx (b, 2)]) / 3.0;
  pim[idx (b, 0)] = (4.0 * pim[idx (b, 1)] - pim[idx (b, 2)]) / 3.0;
  pre[idx (b, N - 1)] =
    (4.0 * pre[idx (b, N - 2)] - pre[idx (b, N - 3)]) / 3.0;
  pim[idx (b, N - 1)] =
    (4.0 * pim[idx (b, N - 2)] - pim[idx (b, N - 3)]) / 3.0;

}

__global__ void
reset_sourced ()

{
  int b;
  int a;

  b = blockIdx.y * blockDim.y + threadIdx.y;
  a = blockIdx.x * blockDim.x + threadIdx.x;

  tred[b][a] = 0.0;
  timd[b][a] = 0.0;

}

__global__ void
sourced (double *theta, double rp, double phip, double tp,
 double E, double lz, double Q,
 double drdt, double d2rdt2, double d3rdt3, double dthdt,
 double d2thdt2, double d3thdt3, double dphidt, double d2phidt2, double *rr)
{

  cudacomplex i;
  int j, k, ip, it;
  double r, th, wt, wr, stheta, ctheta, pie;
  double xp, cs2, delta, wp;
  double a, nmu, rpl, rm;
  double DelR, DelR1, DelR2;
  double DelTH, DelTH1, DelTH2;

  cudacomplex t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13, t14,
    t15, t16, t17, t18, t19, t20, t21, t22, t23, t24, t25, t26, t27, t28, t29,
    t30, t31, t32, t33, t34, t35, t36, t37, t38, t39, t40, t41, t42, t43, t44,
    t45, t46, t47, t48, t49, t50, t51, t52, t53, t54, t55, t56, t57, t58, t59,
    t60, t61, t62, t63, t64, t65, t66, t67, t68, t69, t70, t71, t72, t73, t74,
    t75, t76, t77, t78, t79, t80, t81, t82, t83, t84, t85, t86, t87, t88, t89,
    t90, t91, t92, t93, t94, t95, t96, t97, t98, t99, t100, t101, t102, t103,
    t104, t105, t106, t107, t108, t109, t110, t111, t112, t113, t114, t115,
    t116, t117, t118, t119, t120, t121, t122, t123, t124, t125, t126, t127,
    t128, t129, t130, t131, t132, t133, t134, t135, t136, t137, t138, t139,
    t140, t141, t142, t143, t144, t145, t146, t147, t148, t149, t150, t151,
    t152, t153, t154, t155, t156, t157, t158, t159, t160, t161, t162, t163,
    t164, t165, t166, t167, t168, t169, t170, t171, t172, t173, t174, t175,
    t176, t177, t178, t179, t180, t181, t182, t183, t184, t185, t186, t187,
    t188, t189, t190, t191, t192, t193, t194, t195, t196, t197, t198, t199,
    t200, t201, t202, t203, t204, t205, t206, t207, t208, t209, t210, t211,
    t212, t213, t214, t215, t216, t217, t218, t219, t220, t221, t222, t223,
    t224, t225, t226, t227, t228, t229, t230, t231, t232, t233, t234, t235,
    t236, t237, t238, t239, t240, t241, t242, t243, t244, t245, t246, t247,
    t248, t249, t250, t251, t252, t253, t254, t255, t256, t257, t258, t259,
    t260, t261, t262, t263, t264, t265, t266, t267, t268, t269, t270, t271,
    t272, t273, t274, t275, t276, t277, t278, t279, t280, t281, t282, t283,
    t284, t285, t286, t287, t288, t289, t290, t291, t292, t293, t294, t295,
    t296, t297, t298, t299, t300, t301, t302, t303, t304, t305, t306, t307,
    t308, t309, t310, t311, t312, t313, t314, t315, t316, t317, t318, t319,
    t320, t321, t322, t323, t324, t325, t326, t327, t328, t329, t330, t331,
    t332, t333, t334, t335, t336, t337, t338, t339, t340, t341, t342, t343,
    t344, t345, t346, t347, t348, t349, t350, t351, t352, t353, t354, t355,
    t356, t357, t358, t359, t360, t361, t362, t363, t364, t365, t366, t367,
    t368, t369, t370, t371, t372, t373, t374, t375, t376, t377, t378, t379,
    t380, t381, t382, t383, t384, t385, t386, t387, t388, t389, t390, t391,
    t392, t393, t394, t395, t396, t397, t398, t399, t400, t401, t402, t403,
    t404, t405, t406, t407, t408, t409, t410, t411, t412, t413, t414, t415,
    t416, t417, t418, t419, t420, t421, t422, t423, t424, t425, t426, t427,
    t428, t429, t430, t431, t432, t433, t434, t435, t436, t437, t438, t439,
    t440, t441, t442, t443, t444, t445, t446, t447, t448, t449, t450, t451,
    t452, t453, t454, t455, t456, t457, t458, t459, t460, t461, t462, t463,
    t464, t465, t466, t467, t468, t469, t470, t471, t472, t473, t474, t475,
    t476, t477, t478, t479, t480, t481, t482, t483, t484, t485, t486, t487,
    t488, t489, t490, t491, t492, t493, t494, t495, t496, t497, t498, t499,
    t500, t501, t502, t503, t504, t505, t506, t507, t508, t509, t510, t511,
    t512, t513, t514, t515, t516, t517, t518, t519, t520, t521, t522, t523,
    t524, t525, t526, t527, t528, t529, t530, t531, t532, t533, t534, t535,
    t536, t537, t538, t539, t540, t541, t542, t543, t544, t545, t546, t547,
    t548, t549, t550, t551, t552, t553, t554, t555, t556, t557, t558, t559,
    t560, t561, t562, t563, t564, t565, t566, t567, t568, t569, t570, t571,
    t572, t573, t574, t575, t576, t577, t578, t579, t580, t581, t582, t583,
    t584, t585, t586, t587, t588, t589, t590, t591, t592, t593, t594, t595,
    t596, t597, t598, t599, t600, t601, t602, t603, t604, t605, t606, t607,
    t608, t609, t610, t611, t612, t613, t614, t615, t616, t617, t618, t619,
    t620, t621, t622, t623, t624, t625, t626, t627, t628, t629, t630, t631,
    t632, t633, t634, t635, t636, t637, t638, t639, t640, t641, t642, t643,
    t644, t645, t646, t647, t648, t649, t650, t651, t652, t653, t654, t655,
    t656, t657, t658, t659, t660, t661, t662, t663, t664, t665, t666, t667,
    t668, t669, t670, t671, t672, t673, t674, t675, t676, t677, t678, t679,
    t680, t681, t682, t683, t684, t685, t686, t687, t688, t689, t690, t691,
    t692, t693, t694, t695, t696, t697, t698, t699, t700, t701, t702, t703,
    t704, t705, t706, t707, t708, t709, t710, t711, t712, t713, t714, t715,
    t716, t717, t718, t719, t720, t721, t722, t723, t724, t725, t726, t727,
    t728, t729, t730, t731, t732, t733, t734, t735, t736, t737, t738, t739,
    t740, t741, t742, t743, t744, t745, t746, t747, t748, t749, t750, t751,
    t752, t753, t754, t755, t756, t757, t758, t759, t760, t761, t762, t763,
    t764, t765, t766, t767, t768, t769, t770, t771, t772, t773, t774, t775,
    t776, t777, t778, t779, t780, t781, t782, t783, t784, t785, t786, t787,
    t788, t789, t790, t791, t792, t793, t794, t795, t796, t797, t798, t799,
    t800, t801, t802, t803, t804, t805, t806, t807, t808, t809, t810, t811,
    t812, t813, t814, t815, t816, t817, t818, t819, t820, t821, t822, t823,
    t824, t825, t826, t827, t828, t829, t830, t831, t832, t833, t834, t835,
    t836, t837, t838, t839, t840, t841, t842, t843, t844, t845, t846, t847,
    t848, t849, t850, t851, t852, t853, t854, t855, t856, t857, t858, t859,
    t860, t861, t862, t863, t864, t865, t866, t867, t868, t869, t870, t871,
    t872, t873, t874, t875, t876, t877, t878, t879, t880, t881, t882, t883,
    t884, t885, t886, t887, t888, t889, t890, t891, t892, t893, t894, t895,
    t896, t897, t898, t899, t900, t901, t902, t903, t904, t905, t906, t907,
    t908, t909, t910, t911, t912, t913, t914, t915, t916, t917, t918, t919,
    t920, t921, t922, t923, t924, t925, t926, t927, t928, t929, t930, t931,
    t932, t933, t934, t935, t936, t937, t938, t939, t940, t941, t942, t943,
    t944, t945, t946, t947, t948, t949, t950, t951, t952, t953, t954, t955,
    t956, t957, t958, t959, t960, t961, t962, t963, t964, t965, t966, t967,
    t968, t969, t970, t971, t972, t973, t974, t975, t976, t977, t978, t979,
    t980, t981, t982, t983, t984, t985, t986, t987, t988, t989, t990, t991,
    t992, t993, t994, t995, t996, t997, t998, t999, t1000, t1001, t1002,
    t1003, t1004, t1005, t1006, t1007, t1008, t1009, t1010, t1011, t1012,
    t1013, t1014, t1015, t1016, t1017, t1018, t1019, t1020, t1021, t1022,
    t1023, t1024, t1025, t1026, t1027, t1028, t1029, t1030, t1031, t1032,
    t1033, t1034, t1035, t1036, t1037, t1038, t1039, t1040, t1041, t1042,
    t1043, t1044, t1045, t1046, t1047, t1048, t1049, t1050, t1051, t1052,
    t1053, t1054, t1055, t1056, t1057, t1058, t1059, t1060, t1061, t1062,
    t1063, t1064, t1065, t1066, t1067, t1068, t1069, t1070, t1071, t1072,
    t1073, t1074, t1075, t1076, t1077, t1078, t1079, t1080, t1081, t1082,
    t1083, t1084, t1085, t1086, t1087, t1088, t1089, t1090, t1091, t1092,
    t1093, t1094, t1095, t1096, t1097, t1098, t1099, t1100, t1101, t1102,
    t1103, t1104, t1105, t1106, t1107, t1108, t1109, t1110, t1111, t1112,
    t1113, t1114, t1115, t1116, t1117, t1118, t1119, t1120, t1121, t1122,
    t1123, t1124, t1125, t1126, t1127, t1128, t1129, t1130, t1131, t1132,
    t1133, t1134, t1135, t1136, t1137, t1138, t1139, t1140, t1141, t1142,
    t1143, t1144, t1145, t1146, t1147, t1148, t1149, t1150, t1151, t1152,
    t1153, t1154, t1155, t1156, t1157, t1158, t1159, t1160, t1161, t1162,
    t1163, t1164, t1165, t1166, t1167, t1168, t1169, t1170, t1171, t1172,
    t1173, t1174, t1175, t1176, t1177, t1178, t1179, t1180, t1181, t1182,
    t1183, t1184, t1185, t1186, t1187, t1188, t1189, t1190, t1191, t1192,
    t1193, t1194, t1195, t1196, t1197, t1198, t1199, t1200, t1201, t1202,
    t1203, t1204, t1205, t1206, t1207, t1208, t1209, t1210, t1211, t1212,
    t1213, t1214, t1215, t1216, t1217, t1218, t1219, t1220, t1221, t1222,
    t1223, t1224, t1225, t1226, t1227, t1228, t1229, t1230, t1231, t1232,
    t1233, t1234, t1235, t1236, t1237, t1238, t1239, t1240, t1241, t1242,
    t1243, t1244, t1245, t1246, t1247, t1248, t1249, t1250, t1251, t1252,
    t1253, t1254, t1255, t1256, t1257, t1258, t1259, t1260, t1261, t1262,
    t1263, t1264, t1265, t1266, t1267, t1268, t1269, t1270, t1271, t1272,
    t1273, t1274, t1275, t1276, t1277, t1278, t1279, t1280, t1281, t1282,
    t1283, t1284, t1285, t1286, t1287, t1288, t1289, t1290, t1291, t1292,
    t1293, t1294, t1295, t1296, t1297, t1298, t1299, t1300, t1301, t1302,
    t1303, t1304, t1305, t1306, t1307, t1308, t1309, t1310, t1311, t1312,
    t1313, t1314, t1315, t1316, t1317, t1318, t1319, t1320, t1321, t1322,
    t1323, t1324, t1325, t1326, t1327, t1328, t1329, t1330, t1331, t1332,
    t1333, t1334, t1335, t1336, t1337, t1338, t1339, t1340, t1341, t1342,
    t1343, t1344, t1345, t1346, t1347, t1348, t1349, t1350, t1351, t1352,
    t1353, t1354, t1355, t1356, t1357, t1358, t1359, t1360, t1361, t1362,
    t1363, t1364, t1365, t1366, t1367, t1368, t1369, t1370, t1371, t1372,
    t1373, t1374, t1375, t1376, t1377, t1378, t1379, t1380, t1381, t1382,
    t1383, t1384, t1385, t1386, t1387, t1388, t1389, t1390, t1391, t1392,
    t1393, t1394, t1395, t1396, t1397, t1398, t1399, t1400, t1401, t1402,
    t1403, t1404, t1405, t1406, t1407, t1408, t1409, t1410, t1411, t1412,
    t1413, t1414, t1415, t1416, t1417, t1418, t1419, t1420, t1421, t1422,
    t1423, t1424, t1425, t1426, t1427, t1428, t1429, t1430, t1431, t1432,
    t1433, t1434, t1435, t1436, t1437, t1438, t1439, t1440, t1441, t1442,
    t1443, t1444, t1445, t1446, t1447, t1448, t1449, t1450, t1451, t1452,
    t1453, t1454, t1455, t1456, t1457, t1458, t1459, t1460, t1461, t1462,
    t1463, t1464, t1465, t1466, t1467, t1468, t1469, t1470, t1471, t1472,
    t1473, t1474, t1475, t1476, t1477, t1478, t1479, t1480, t1481, t1482,
    t1483, t1484, t1485, t1486, t1487, t1488, t1489, t1490, t1491, t1492,
    t1493, t1494, t1495, t1496, t1497, t1498, t1499, t1500, t1501, t1502,
    t1503, t1504, t1505, t1506, t1507, t1508, t1509, t1510, t1511, t1512,
    t1513, t1514, t1515, t1516, t1517, t1518, t1519, t1520, t1521, t1522,
    t1523, t1524, t1525, t1526, t1527, t1528, t1529, t1530, t1531, t1532,
    t1533, t1534, t1535, t1536, t1537, t1538, t1539, t1540, t1541, t1542,
    t1543, t1544, t1545, t1546, t1547, t1548, t1549, t1550, t1551, t1552,
    t1553, t1554, t1555, t1556, t1557, t1558, t1559, t1560, t1561, t1562,
    t1563, t1564, t1565, t1566, t1567, t1568, t1569, t1570, t1571, t1572,
    t1573, t1574, t1575, t1576, t1577, t1578, t1579, t1580, t1581, t1582,
    t1583, t1584, t1585, t1586, t1587, t1588, t1589, t1590, t1591, t1592,
    t1593, t1594, t1595, t1596, t1597, t1598, t1599, t1600, t1601, t1602,
    t1603, t1604, t1605, t1606, t1607, t1608, t1609, t1610, t1611, t1612,
    t1613, t1614, t1615, t1616, t1617, t1618, t1619, t1620, t1621, t1622,
    t1623, t1624, t1625, t1626, t1627, t1628, t1629, t1630, t1631, t1632,
    t1633, t1634, t1635, t1636, t1637, t1638, t1639, t1640, t1641, t1642,
    t1643, t1644, t1645, t1646, t1647, t1648, t1649, t1650, t1651, t1652,
    t1653, t1654, t1655, t1656, t1657, t1658, t1659, t1660, t1661, t1662,
    t1663, t1664, t1665, t1666, t1667, t1668, t1669, t1670, t1671, t1672,
    t1673, t1674, t1675, t1676, t1677, t1678, t1679, t1680, t1681, t1682,
    t1683, t1684, t1685, t1686, t1687, t1688, t1689, t1690, t1691, t1692,
    t1693, t1694, t1695, t1696, t1697, t1698, t1699, t1700, t1701, t1702,
    t1703, t1704, t1705, t1706, t1707, t1708, t1709, t1710, t1711, t1712,
    t1713, t1714, t1715, t1716, t1717, t1718, t1719, t1720, t1721, t1722,
    t1723, t1724, t1725, t1726, t1727, t1728, t1729, t1730, t1731, t1732,
    t1733, t1734, t1735, t1736, t1737, t1738, t1739, t1740, t1741, t1742,
    t1743, t1744, t1745, t1746, t1747, t1748, t1749, t1750, t1751, t1752,
    t1753, t1754, t1755, t1756, t1757, t1758, t1759, t1760, t1761, t1762,
    t1763, t1764, t1765, t1766, t1767, t1768, t1769, t1770, t1771, t1772,
    t1773, t1774, t1775, t1776, t1777, t1778, t1779, t1780, t1781, t1782,
    t1783, t1784, t1785, t1786, t1787, t1788, t1789, t1790, t1791, t1792,
    t1793, t1794, t1795, t1796, t1797, t1798, t1799, t1800, t1801, t1802,
    t1803, t1804, t1805, t1806, t1807, t1808, t1809, t1810, t1811, t1812,
    t1813, t1814, t1815, t1816, t1817, t1818, t1819, t1820, t1821, t1822,
    t1823, t1824, t1825, t1826, t1827, t1828, t1829, t1830, t1831, t1832,
    t1833, t1834, t1835, t1836, t1837, t1838, t1839, t1840, t1841, t1842,
    t1843, t1844, t1845, t1846, t1847, t1848, t1849, t1850, t1851, t1852,
    t1853, t1854, t1855, t1856, t1857, t1858, t1859, t1860, t1861, t1862,
    t1863, t1864, t1865, t1866, t1867, t1868, t1869, t1870, t1871, t1872,
    t1873, t1874, t1875, t1876, t1877, t1878, t1879, t1880, t1881, t1882,
    t1883, t1884, t1885, t1886, t1887, t1888, t1889, t1890, t1891, t1892,
    t1893, t1894, t1895, t1896, t1897, t1898, t1899, t1900, t1901, t1902,
    t1903, t1904, t1905, t1906, t1907, t1908, t1909, t1910, t1911, t1912,
    t1913, t1914, t1915, t1916, t1917, t1918, t1919, t1920, t1921, t1922,
    t1923, t1924, t1925, t1926, t1927, t1928, t1929, t1930, t1931, t1932,
    t1933, t1934, t1935, t1936, t1937, t1938, t1939, t1940, t1941, t1942,
    t1943, t1944, t1945, t1946, t1947, t1948, t1949, t1950, t1951, t1952,
    t1953, t1954, t1955, t1956, t1957, t1958, t1959, t1960, t1961, t1962,
    t1963, t1964, t1965, t1966, t1967, t1968, t1969, t1970, t1971, t1972,
    t1973, t1974, t1975, t1976, t1977, t1978, t1979, t1980, t1981, t1982,
    t1983, t1984, t1985, t1986, t1987, t1988, t1989, t1990, t1991, t1992,
    t1993, t1994, t1995, t1996, t1997, t1998, t1999, t2000, t2001, t2002,
    t2003, t2004, t2005, t2006, t2007, t2008, t2009, t2010, t2011, t2012,
    t2013, t2014, t2015, t2016, t2017, t2018, t2019, t2020, t2021, t2022,
    t2023, t2024, t2025, t2026, t2027, t2028, t2029, t2030, t2031, t2032,
    t2033, t2034, t2035, t2036, t2037, t2038, t2039, t2040, t2041, t2042,
    t2043, t2044, t2045, t2046, t2047, t2048, t2049, t2050, t2051, t2052,
    t2053, t2054, t2055, t2056, t2057, t2058, t2059, t2060, t2061, t2062,
    t2063, t2064, t2065, t2066, t2067, t2068, t2069, t2070, t2071, t2072,
    t2073, t2074, t2075, t2076, t2077, t2078, t2079, t2080, t2081, t2082,
    t2083, t2084, t2085, t2086, t2087, t2088, t2089, t2090, t2091, t2092,
    t2093, t2094, t2095, t2096, t2097, t2098, t2099, t2100, t2101, t2102,
    t2103, t2104, t2105, t2106, t2107, t2108, t2109, t2110, t2111, t2112,
    t2113, t2114, t2115, t2116, t2117, t2118, t2119, t2120, t2121, t2122,
    t2123, t2124, t2125, t2126, t2127, t2128, t2129, t2130, t2131, t2132,
    t2133, t2134, t2135, t2136, t2137, t2138, t2139, t2140, t2141, t2142,
    t2143, t2144, t2145, t2146, t2147, t2148, t2149, t2150, t2151, t2152,
    t2153, t2154, t2155, t2156, t2157, t2158, t2159, t2160, t2161, t2162,
    t2163, t2164, t2165, t2166, t2167, t2168, t2169, t2170, t2171, t2172,
    t2173, t2174, t2175, t2176, t2177, t2178, t2179, t2180, t2181, t2182,
    t2183, t2184, t2185, t2186, t2187, t2188, t2189, t2190, t2191, t2192,
    t2193, t2194, t2195, t2196, t2197, t2198, t2199, t2200, t2201, t2202,
    t2203, t2204, t2205, t2206, t2207, t2208, t2209, t2210, t2211, t2212,
    t2213, t2214, t2215, t2216, t2217, t2218, t2219, t2220, t2221, t2222,
    t2223, t2224, t2225, t2226, t2227, t2228, t2229, t2230, t2231, t2232,
    t2233, t2234, t2235, t2236, t2237, t2238, t2239, t2240, t2241, t2242,
    t2243, t2244, t2245, t2246, t2247, t2248, t2249, t2250, t2251, t2252,
    t2253, t2254, t2255, t2256, t2257, t2258, t2259, t2260, t2261, t2262,
    t2263, t2264, t2265, t2266, t2267, t2268, t2269, t2270, t2271, t2272,
    t2273, t2274, t2275, t2276, t2277, t2278, t2279, t2280, t2281, t2282,
    t2283, t2284, t2285, t2286, t2287, t2288, t2289, t2290, t2291, t2292,
    t2293, t2294, t2295, t2296, t2297, t2298, t2299, t2300, t2301, t2302,
    t2303, t2304, t2305, t2306, t2307, t2308, t2309, t2310, t2311, t2312,
    t2313, t2314, t2315, t2316, t2317, t2318, t2319, t2320, t2321, t2322,
    t2323, t2324, t2325, t2326, t2327, t2328, t2329, t2330, t2331, t2332,
    t2333, t2334, t2335, t2336, t2337, t2338, t2339, t2340, t2341, t2342,
    t2343, t2344, t2345, t2346, t2347, t2348, t2349, t2350, t2351, t2352,
    t2353, t2354, t2355, t2356, t2357, t2358, t2359, t2360, t2361, t2362,
    t2363, t2364, t2365, t2366, t2367, t2368, t2369, t2370, t2371, t2372,
    t2373, t2374, t2375, t2376, t2377, t2378, t2379, t2380, t2381, t2382,
    t2383, t2384, t2385, t2386, t2387, t2388, t2389, t2390, t2391, t2392,
    t2393, t2394, t2395, t2396, t2397, t2398, t2399, t2400, t2401, t2402,
    t2403, t2404, t2405, t2406, t2407, t2408, t2409, t2410, t2411, t2412,
    t2413, t2414, t2415, t2416, t2417, t2418, t2419, t2420, t2421, t2422,
    t2423, t2424, t2425, t2426, t2427, t2428, t2429, t2430, t2431, t2432,
    t2433, t2434, t2435, t2436, t2437, t2438, t2439, t2440, t2441, t2442,
    t2443, t2444, t2445, t2446, t2447, t2448, t2449, t2450, t2451, t2452,
    t2453, t2454, t2455, t2456, t2457, t2458, t2459, t2460, t2461, t2462,
    t2463, t2464, t2465, t2466, t2467, t2468, t2469, t2470, t2471, t2472,
    t2473, t2474, t2475, t2476, t2477, t2478, t2479, t2480, t2481, t2482,
    t2483, t2484, t2485, t2486, t2487, t2488, t2489, t2490, t2491, t2492,
    t2493, t2494, t2495, t2496, t2497, t2498, t2499, t2500, t2501, t2502,
    t2503, t2504, t2505, t2506, t2507, t2508, t2509, t2510, t2511, t2512,
    t2513, t2514, t2515, t2516, t2517, t2518, t2519, t2520, t2521, t2522,
    t2523, t2524, t2525, t2526, t2527, t2528, t2529, t2530, t2531, t2532,
    t2533, t2534, t2535, t2536, t2537, t2538, t2539, t2540, t2541, t2542,
    t2543, t2544, t2545, t2546, t2547, t2548, t2549, t2550, t2551, t2552,
    t2553, t2554, t2555, t2556, t2557, t2558, t2559, t2560, t2561, t2562,
    t2563, t2564, t2565, t2566, t2567, t2568, t2569, t2570, t2571, t2572,
    t2573, t2574, t2575, t2576, t2577, t2578, t2579, t2580, t2581, t2582,
    t2583, t2584, t2585, t2586, t2587, t2588, t2589, t2590, t2591, t2592,
    t2593, t2594, t2595, t2596, t2597, t2598, t2599, t2600, t2601, t2602,
    t2603, t2604, t2605, t2606, t2607, t2608, t2609, t2610, t2611, t2612,
    t2613, t2614, t2615, t2616, t2617, t2618, t2619, t2620, t2621, t2622,
    t2623, t2624, t2625, t2626, t2627, t2628, t2629, t2630, t2631, t2632,
    t2633, t2634, t2635, t2636, t2637, t2638, t2639, t2640, t2641, t2642,
    t2643, t2644, t2645, t2646, t2647, t2648, t2649, t2650, t2651, t2652,
    t2653, t2654, t2655, t2656, t2657, t2658, t2659, t2660, t2661, t2662,
    t2663, t2664, t2665, t2666, t2667, t2668, t2669, t2670, t2671, t2672,
    t2673, t2674, t2675, t2676, t2677, t2678, t2679, t2680, t2681, t2682,
    t2683, t2684, t2685, t2686, t2687, t2688, t2689, t2690, t2691, t2692,
    t2693, t2694, t2695, t2696, t2697, t2698, t2699, t2700, t2701, t2702,
    t2703, t2704, t2705, t2706, t2707, t2708, t2709, t2710, t2711, t2712,
    t2713, t2714, t2715, t2716, t2717, t2718, t2719, t2720, t2721, t2722,
    t2723, t2724, t2725, t2726, t2727, t2728, t2729, t2730, t2731, t2732,
    t2733, t2734, t2735, t2736, t2737, t2738, t2739, t2740, t2741, t2742,
    t2743, t2744, t2745, t2746, t2747, t2748, t2749, t2750, t2751, t2752,
    t2753, t2754, t2755, t2756, t2757, t2758, t2759, t2760, t2761, t2762,
    t2763, t2764, t2765, t2766, t2767, t2768, t2769, t2770, t2771, t2772,
    t2773, t2774, t2775, t2776, t2777, t2778, t2779, t2780, t2781, t2782,
    t2783, t2784, t2785, t2786, t2787, t2788, t2789, t2790, t2791, t2792,
    t2793, t2794, t2795, t2796, t2797, t2798, t2799, t2800, t2801, t2802,
    t2803, t2804, t2805, t2806, t2807, t2808, t2809, t2810, t2811, t2812,
    t2813, t2814, t2815, t2816, t2817, t2818, t2819, t2820, t2821, t2822,
    t2823, t2824, t2825, t2826, t2827, t2828, t2829, t2830, t2831, t2832,
    t2833, t2834, t2835, t2836, t2837, t2838, t2839, t2840, t2841, t2842,
    t2843, t2844, t2845, t2846, t2847, t2848, t2849, t2850, t2851, t2852,
    t2853, t2854, t2855, t2856, t2857, t2858, t2859, t2860, t2861, t2862,
    t2863, t2864, t2865, t2866, t2867, t2868, t2869, t2870, t2871, t2872,
    t2873, t2874, t2875, t2876, t2877, t2878, t2879, t2880, t2881, t2882,
    t2883, t2884, t2885, t2886, t2887, t2888, t2889, t2890, t2891, t2892,
    t2893, t2894, t2895, t2896, t2897, t2898, t2899, t2900, t2901, t2902,
    t2903, t2904, t2905, t2906, t2907, t2908, t2909, t2910, t2911, t2912,
    t2913, t2914, t2915, t2916, t2917, t2918, t2919, t2920, t2921, t2922,
    t2923, t2924, t2925, t2926, t2927, t2928, t2929, t2930, t2931, t2932,
    t2933, t2934, t2935, t2936, t2937, t2938, t2939, t2940, t2941, t2942,
    t2943, t2944, t2945, t2946, t2947, t2948, t2949, t2950, t2951, t2952,
    t2953, t2954, t2955, t2956, t2957, t2958, t2959, t2960, t2961, t2962,
    t2963, t2964, t2965, t2966, t2967, t2968, t2969, t2970, t2971, t2972,
    t2973, t2974, t2975, t2976, t2977, t2978, t2979, t2980, t2981, t2982,
    t2983, t2984, t2985, t2986, t2987, t2988, t2989, t2990, t2991, t2992,
    t2993, t2994, t2995, t2996, t2997, t2998, t2999, t3000, t3001, t3002,
    t3003, t3004, t3005, t3006, t3007, t3008, t3009, t3010, t3011, t3012,
    t3013, t3014, t3015, t3016, t3017, t3018, t3019, t3020, t3021, t3022,
    t3023, t3024, t3025, t3026, t3027, t3028, t3029, t3030, t3031, t3032,
    t3033, t3034, t3035, t3036, t3037, t3038, t3039, t3040, t3041, t3042,
    t3043, t3044, t3045, t3046, t3047, t3048, t3049, t3050, t3051, t3052,
    t3053, t3054, t3055, t3056, t3057, t3058, t3059, t3060, t3061, t3062,
    t3063, t3064, t3065, t3066, t3067, t3068, t3069, t3070, t3071, t3072,
    t3073, t3074, t3075, t3076, t3077, t3078, t3079, t3080, t3081, t3082,
    t3083, t3084, t3085, t3086, t3087, t3088, t3089, t3090, t3091, t3092,
    t3093, t3094, t3095, t3096, t3097, t3098, t3099, t3100, t3101, t3102,
    t3103, t3104, t3105, t3106, t3107, t3108, t3109, t3110, t3111, t3112,
    t3113, t3114, t3115, t3116, t3117, t3118, t3119, t3120, t3121, t3122,
    t3123, t3124, t3125, t3126, t3127, t3128, t3129, t3130, t3131, t3132,
    t3133, t3134, t3135, t3136, t3137, t3138, t3139, t3140, t3141, t3142,
    t3143, t3144, t3145, t3146, t3147, t3148, t3149, t3150, t3151, t3152,
    t3153, t3154, t3155, t3156, t3157, t3158, t3159, t3160, t3161, t3162,
    t3163, t3164, t3165, t3166, t3167, t3168, t3169, t3170, t3171, t3172,
    t3173, t3174, t3175, t3176, t3177, t3178, t3179, t3180, t3181, t3182,
    t3183, t3184, t3185, t3186, t3187, t3188, t3189, t3190, t3191, t3192,
    t3193, t3194, t3195, t3196, t3197, t3198, t3199, t3200, t3201, t3202,
    t3203, t3204, t3205, t3206, t3207, t3208, t3209, t3210, t3211, t3212,
    t3213, t3214, t3215, t3216, t3217, t3218, t3219, t3220, t3221, t3222,
    t3223, t3224, t3225, t3226, t3227, t3228, t3229, t3230, t3231, t3232,
    t3233, t3234, t3235, t3236, t3237, t3238, t3239, t3240, t3241, t3242,
    t3243, t3244, t3245, t3246, t3247, t3248, t3249, t3250, t3251, t3252,
    t3253, t3254, t3255, t3256, t3257, t3258, t3259, t3260, t3261, t3262,
    t3263, t3264, t3265, t3266, t3267, t3268, t3269, t3270, t3271, t3272,
    t3273, t3274, t3275, t3276, t3277, t3278, t3279, t3280, t3281, t3282,
    t3283, t3284, t3285, t3286, t3287, t3288, t3289, t3290, t3291, t3292,
    t3293, t3294, t3295, t3296, t3297, t3298, t3299, t3300, t3301, t3302,
    t3303, t3304, t3305, t3306, t3307, t3308, t3309, t3310, t3311, t3312,
    t3313, t3314, t3315, t3316, t3317, t3318, t3319, t3320, t3321, t3322,
    t3323, t3324, t3325, t3326, t3327, t3328, t3329, t3330, t3331, t3332,
    t3333, t3334, t3335, t3336, t3337, t3338, t3339, t3340, t3341, t3342,
    t3343, t3344, t3345, t3346, t3347, t3348, t3349, t3350, t3351, t3352,
    t3353, t3354, t3355, t3356, t3357, t3358, t3359, t3360, t3361, t3362,
    t3363, t3364, t3365, t3366, t3367, t3368, t3369, t3370, t3371, t3372,
    t3373, t3374, t3375, t3376, t3377, t3378, t3379, t3380, t3381, t3382,
    t3383, t3384, t3385, t3386, t3387, t3388, t3389, t3390, t3391, t3392,
    t3393, t3394, t3395, t3396, t3397, t3398, t3399, t3400, t3401, t3402,
    t3403, t3404, t3405, t3406, t3407, t3408, t3409, t3410, t3411, t3412,
    t3413, t3414, t3415, t3416, t3417, t3418, t3419, t3420, t3421, t3422,
    t3423, t3424, t3425, t3426, t3427, t3428, t3429, t3430, t3431, t3432,
    t3433, t3434, t3435, t3436, t3437, t3438, t3439, t3440, t3441, t3442,
    t3443, t3444, t3445, t3446, t3447, t3448, t3449, t3450, t3451, t3452,
    t3453, t3454, t3455, t3456, t3457, t3458, t3459, t3460, t3461, t3462,
    t3463, t3464, t3465, t3466, t3467, t3468, t3469, t3470, t3471, t3472,
    t3473, t3474, t3475, t3476, t3477, t3478, t3479, t3480, t3481, t3482,
    t3483, t3484, t3485, t3486, t3487, t3488, t3489, t3490, t3491, t3492,
    t3493, t3494, t3495, t3496, t3497, t3498, t3499, t3500, t3501, t3502,
    t3503, t3504, t3505, t3506, t3507, t3508, t3509, t3510, t3511, t3512,
    t3513, t3514, t3515, t3516, t3517, t3518, t3519, t3520, t3521, t3522,
    t3523, t3524, t3525, t3526, t3527, t3528, t3529, t3530, t3531, t3532,
    t3533, t3534, t3535, t3536, t3537, t3538, t3539, t3540, t3541, t3542,
    t3543, t3544, t3545, t3546, t3547, t3548, t3549, t3550, t3551, t3552,
    t3553, t3554, t3555, t3556, t3557, t3558, t3559, t3560, t3561, t3562,
    t3563, t3564, t3565, t3566, t3567, t3568, t3569, t3570, t3571, t3572,
    t3573, t3574, t3575, t3576, t3577, t3578, t3579, t3580, t3581, t3582,
    t3583, t3584, t3585, t3586, t3587, t3588, t3589, t3590, t3591, t3592,
    t3593, t3594, t3595, t3596, t3597, t3598, t3599, t3600, t3601, t3602,
    t3603, t3604, t3605, t3606, t3607, t3608, t3609, t3610, t3611, t3612,
    t3613, t3614, t3615, t3616, t3617, t3618, t3619, t3620, t3621, t3622,
    t3623, t3624, t3625, t3626, t3627, t3628, t3629, t3630, t3631, t3632,
    t3633, t3634, t3635, t3636, t3637, t3638, t3639, t3640, t3641, t3642,
    t3643, t3644, t3645, t3646, t3647, t3648, t3649, t3650, t3651, t3652,
    t3653, t3654, t3655, t3656, t3657, t3658, t3659, t3660, t3661, t3662,
    t3663, t3664, t3665, t3666, t3667, t3668, t3669, t3670, t3671, t3672,
    t3673, t3674, t3675, t3676, t3677, t3678, t3679, t3680, t3681, t3682,
    t3683, t3684, t3685, t3686, t3687, t3688, t3689, t3690, t3691, t3692,
    t3693, t3694, t3695, t3696, t3697, t3698, t3699, t3700, t3701, t3702,
    t3703, t3704, t3705, t3706, t3707, t3708, t3709, t3710, t3711, t3712,
    t3713, t3714, t3715, t3716, t3717, t3718, t3719, t3720, t3721, t3722,
    t3723, t3724, t3725, t3726, t3727, t3728, t3729, t3730, t3731, t3732,
    t3733, t3734, t3735, t3736, t3737, t3738, t3739, t3740, t3741, t3742,
    t3743, t3744, t3745, t3746, t3747, t3748, t3749, t3750, t3751, t3752,
    t3753, t3754, t3755, t3756, t3757, t3758, t3759, t3760, t3761, t3762,
    t3763, t3764, t3765, t3766, t3767, t3768, t3769, t3770, t3771, t3772,
    t3773, t3774, t3775, t3776, t3777, t3778, t3779, t3780, t3781, t3782,
    t3783, t3784, t3785, t3786, t3787, t3788, t3789, t3790, t3791, t3792,
    t3793, t3794, t3795, t3796, t3797, t3798, t3799, t3800, t3801, t3802,
    t3803, t3804, t3805, t3806, t3807, t3808, t3809, t3810, t3811, t3812,
    t3813, t3814, t3815, t3816, t3817, t3818, t3819, t3820, t3821, t3822,
    t3823, t3824, t3825, t3826, t3827, t3828, t3829, t3830, t3831, t3832,
    t3833, t3834, t3835, t3836, t3837, t3838, t3839, t3840, t3841, t3842,
    t3843, t3844, t3845, t3846, t3847, t3848, t3849, t3850, t3851, t3852,
    t3853, t3854, t3855, t3856, t3857, t3858, t3859, t3860, t3861, t3862,
    t3863, t3864, t3865, t3866, t3867, t3868, t3869, t3870, t3871, t3872,
    t3873, t3874, t3875, t3876, t3877, t3878, t3879, t3880, t3881, t3882,
    t3883, t3884, t3885, t3886, t3887, t3888, t3889, t3890, t3891, t3892,
    t3893, t3894, t3895, t3896, t3897, t3898, t3899, t3900, t3901, t3902,
    t3903, t3904, t3905, t3906, t3907, t3908, t3909, t3910, t3911, t3912,
    t3913, t3914, t3915, t3916, t3917, t3918, t3919, t3920, t3921, t3922,
    t3923, t3924, t3925, t3926, t3927, t3928, t3929, t3930, t3931, t3932,
    t3933, t3934, t3935, t3936, t3937, t3938, t3939, t3940, t3941, t3942,
    t3943, t3944, t3945, t3946, t3947, t3948, t3949, t3950, t3951, t3952,
    t3953, t3954, t3955, t3956, t3957, t3958, t3959, t3960, t3961, t3962,
    t3963, t3964, t3965, t3966, t3967, t3968, t3969, t3970, t3971, t3972,
    t3973, t3974, t3975, t3976, t3977, t3978, t3979, t3980, t3981, t3982,
    t3983, t3984, t3985, t3986, t3987, t3988, t3989, t3990, t3991, t3992,
    t3993, t3994, t3995, t3996, t3997, t3998, t3999;
```

```
cudacomplex mgv1;
cudacomplex mgv2;
cudacomplex mgv3;
cudacomplex mgv4;
cudacomplex mgv5;
cudacomplex mgv6;
cudacomplex mgv7;
cudacomplex mgv8;
cudacomplex mgv9;
cudacomplex mgv10;

cudacomplex TT;
cudacomplex rho;

i.real = 0.0;
i.img = 1.0;

j = blockIdx.y * blockDim.y + threadIdx.y;
k = blockIdx.x * blockDim.x + threadIdx.x;

pie = 3.141592653589793;
a = aa;
nmu = 0.01;
wt = dtheta;
wp = dx;

rpl = mass + sqrt (mass * mass - a * a);
rm = mass - sqrt (mass * mass - a * a);
wr = wp * (rp - rpl) * (rp - rm) / (rp * rp + a * a);

xp = rp + (rpl * rpl + aa * aa) / (rpl - rm) * log (rp / rpl - 1.0) -
  (rm * rm + aa * aa) / (rpl - rm) * log (rp / rm - 1.0);
ip = (int) ((xp - Xmin) / dx + 0.5 );
it = (int) (tp / dtheta + 0.5 );
```

```
  r = rr[k + ip - PTS / 2];
  th = theta[j + it - PTS / 2];

  DelR =
    1.0 / sqrt (2.0 * pie) * exp (-(r - rp) * (r - rp) / (2.0 * wr * wr)) /
    wr;
  DelR1 = -(r - rp) / (wr * wr) * DelR;
  DelR2 =
    (-1.0 / (wr * wr) + (r - rp) * (r - rp) / (wr * wr * wr * wr)) * DelR;



  stheta = sin (th);
  ctheta = cos (th);

  DelTH =
    1.0 / sqrt (2.0 * pie) * exp (-(th - tp) * (th - tp) / (2.0 * wt * wt)) /
    wt;
  DelTH1 = -(th - tp) / (wt * wt) * DelTH;
  DelTH2 =
    (-1.0 / (wt * wt) + (th - tp) * (th - tp) / (wt * wt * wt * wt)) * DelTH;

//    Maple generated expression here

  t1 = rp * rp;
  t2 = a * a;
  t3 = cos (tp);
  t4 = t3 * t3;
  t5 = t2 * t4;
  t6 = t1 + t5;
  t7 = 1 / t6;
  t8 = t2 + t1;
  t9 = (-1.0 / 4.0 * i) * a;
  t10 = a * t3;
```

```
t11 = rp + (i) * t10;

t12 = t11 * t11;

t13 = t9 * t12;

t14 = t6 * t6;

t15 = t14 * t14;

t16 = 1 / t15;

t17 = (0.0 - i) * a;

t19 = rp + t17 * t3;

t20 = t16 * t19;

t21 = t20 * nmu;

t22 = t13 * t21;

t23 = t8 * t8;

t26 = t1 + t2 - 2.0 * mass * rp;

t27 = 1 / t26;

t29 = sin (tp);

t30 = t29 * t29;

t31 = t2 * t30;

t34 = a * lz;

t35 = t8 * t27;

t38 = E * (t23 * t27 - t31) + t34 * (1.0 - t35);

t39 = t38 * t38;

t40 = 1 / t39;

t43 = E * t8 + drdt * t38 - t34;

t44 = t40 * t43;

t46 = 1 / t30;

t48 = a * E - lz * t46;

t51 = (i) * t29 * t48 + dthdt * t38;

t52 = t44 * t51;

t53 = 1 / pie;

t56 = cos (-mm * phip) + i * sin (-mm * phip);

t57 = t53 * t56;

t58 = rp * drdt;

t61 = t26 * t26;

t62 = 1 / t61;
```

```
t63 = t23 * t62;

t65 = t58 - mass * drdt;

t67 = t2 * t3;

t68 = t29 * dthdt;

t69 = t67 * t68;

t75 = t8 * t62;

t79 =

  E * (4.0 * t35 * t58 - 2.0 * t63 * t65 - 2.0 * t69) +

  t34 * (-2.0 * t58 * t27 + 2.0 * t75 * t65);

t83 = (1.0 / 2.0 * i) * a;

t84 = t83 * t12;

t85 = 1 / t14;

t86 = t85 * t29;

t87 = sqrt (2.0);

t88 = t86 * t87;

t90 = t11 * t7;

t91 = 1 / tan (tp);

t92 = t91 * t87;

t95 = t84 * t88 - t90 * t92 / 4.0;

t96 = t95 * nmu;

t97 = t11 * t85;

t98 = t97 * t87;

t99 = t96 * t98;

t100 = 1 / t38;

t101 = E * rp;

t106 = 2.0 * t101 * drdt + d2rdt2 * t38 + drdt * t79;

t108 = t100 * t106 * t51;

t109 = 1 / t29;

t110 = t57 * t109;

t111 = t108 * t110;

t114 = (1.0 / 4.0 * i) * a;

t115 = t12 * t16;

t116 = t115 * t19;

t117 = t114 * t116;
```

```
t118 = nmu * t100;
t120 = t51 * t53;
t121 = t120 * t56;
t124 = (-1.0 / 2.0 * i) * a;
t127 = 1 / t15 / t6;
t128 = t127 * t19;
t129 = t128 * nmu;
t131 = t100 * t43;
t132 = t131 * t51;
t133 = t58 - t69;
t138 = 1 / t14 / t6;
t139 = t12 * t138;
t140 = t19 * t26;
t142 = rp - mass;
t143 = t19 * t142;
t144 = t97 * t143;
t145 = t144 - t139 * t140;
t146 = t145 * nmu / 2.0;
t147 = t12 * t85;
t148 = t147 * t100;
t149 = t146 * t148;
t150 = t56 * t109;
t151 = t3 * dthdt;
t153 = (2.0 * i) * t46;
t154 = lz * t3;
t159 = (i) * t151 * t48 + t153 * t154 * dthdt + d2thdt2 * t38 + dthdt * t79;
t160 = t150 * t159;
t161 = t120 * t160;
t163 = t19 * t19;
t164 = t163 * t138;
t165 = t11 * t26;
t167 = t144 - t164 * t165;
t168 = t167 * nmu / 2.0;
t169 = t168 * t148;
```

```
t171 = t97 * t100;

t173 = t51 * t51;

t174 = t173 * t53;

t176 = drdt + t17 * t68;

t177 = t150 * t176;

t178 = t174 * t177;

t180 = t174 * t56;

t181 = t46 * t3;

t182 = t181 * dthdt;

t183 = t180 * t182;

t186 = t139 * t100;

t188 = 2.0 * t150 * t133;

t189 = t174 * t188;

t191 = t131 * t159;

t192 = t191 * t110;

t197 = t163 * t127;

t198 = t12 * t11;

t200 = t198 * t26 * nmu;

t201 = t197 * t200;

t204 = t150 * t79;

t208 = t147 * t40;

t210 = t174 * t204;

t221 = t12 * t26;

t222 = t221 * nmu;

t223 = t197 * t222;

t224 = t100 * t173;

t225 = t224 * t53;

t229 = t96 * t11;

t230 = t85 * t87;

t231 = t230 * t100;

t232 = t229 * t231;

t233 = t43 * t51;

t234 = t233 * t53;

t235 = t56 * t46;
```

```
t236 = t235 * t151;

t237 = t234 * t236;

t240 =

  t22 * t52 * t57 * t79 - t99 * t111 / 4.0 +

  t117 * t118 * t106 * t121 +

  2.0 * t124 * t12 * t129 * t132 * t57 * t133 - t149 * t161 +

  t169 * t161 - t146 * t171 * t178 - t169 * t183 / 2.0 +

  t146 * t186 * t189 - t99 * t192 / 4.0 + t149 * t183 / 2.0 -

  t201 * t40 * t173 * t53 * t204 / 8.0 +

  t146 * t208 * t210 / 2.0 - t168 * t208 * t210 / 2.0 +

  t201 * t100 * t51 * t53 * t160 / 4.0 +

  t223 * t225 * t177 / 4.0 + t232 * t237 / 4.0;

t241 = t230 * t40;

t243 = t234 * t204;

t250 = t132 * t57 * t151;

t251 = t13 * t20 * t109 * nmu * t250;

t252 = (1.0 / 8.0 * i) * t46;

t255 = mm * mm;

t256 = t53 * t255;

t257 = dphidt * t56;

t261 = t46 * nmu;

t263 = t176 * t85 * t87;

t265 = t53 * mm;

t266 = t265 * t56;

t267 = t132 * t266;

t270 = t261 * t11;

t271 = t138 * t87;

t272 = t271 * t100;

t274 = mm * t56;

t275 = 2.0 * t274 * t133;

t280 = t274 * t79;

t284 = t261 * t98;

t291 = dthdt * dthdt;

t294 = t29 * d2thdt2;
```

```
t296 = d2rdt2 + t17 * t3 * t291 + t17 * t294;
t299 = t233 * t110;
t302 = nmu * t176;
t304 = t234 * t188;
t310 = nmu * t11;
t311 = t310 * t272;
t312 = t106 * t51;
t313 = t312 * t53;
t317 = t310 * t241;
t321 = t310 * t85;
t322 = t87 * t40;
t323 = t322 * t43;
t324 = t321 * t323;
t330 = t310 * t231;
t331 = t150 * t291;
t332 = t234 * t331;
t335 = (1.0 / 4.0 * i) * nmu;
t336 = t335 * t11;
t337 = t336 * t231;
t338 = mm * dphidt;
t339 = t338 * t150;
t342 = t16 * t87;
t345 = 4.0 * t133 * t133;
t346 = t150 * t345;
t350 = t310 * t138;
t352 = 2.0 * t109 * t133;
t353 = t352 * t79;
t357 = t87 * t100;
t358 = t357 * t43;
t359 = t350 * t358;
t365 = (-1.0 / 4.0 * i) * nmu;
t366 = t365 * t11;
t367 = t230 * t131;
t368 = t366 * t367;
```

```
 t372 = t321 * t358;
 t373 = t120 * t255;
 t374 = dphidt * dphidt;
 t376 = t374 * t56 * t109;
 t381 = 1 / t30 / t29;
 t383 = t381 * t4 * t291;
 t393 = t291 * lz;
 t403 = drdt * drdt;
 t412 = rp * d2rdt2;
 t416 = 1 / t61 / t26;
 t418 = 4.0 * t65 * t65;
 t422 = t403 + t412 - mass * d2rdt2;
 t424 = t31 * t291;
 t426 = t5 * t291;
 t428 = t67 * t294;
 t445 =
   E * (8.0 * t1 * t403 * t27 - 16.0 * t75 * t58 * t65 +
4.0 * t35 * t403 + 4.0 * t35 * t412 +
2.0 * t23 * t416 * t418 - 2.0 * t63 * t422 + 2.0 * t424 -
2.0 * t426 - 2.0 * t428) + t34 * (-2.0 * t403 * t27 -
  2.0 * t412 * t27 +
  8.0 * t58 * t62 * t65 -
  2.0 * t8 * t416 *
  t418 + 2.0 * t75 * t422);
 t447 =
   (0.0 - i) * t29 * t291 * t48 + (i) * t3 * d2thdt2 * t48 +
   (-2.0 * i) * t4 * t393 * t381 + (-2.0 * i) * t109 * t393 +
   t153 * t154 * d2thdt2 + d3thdt3 * t38 + 2.0 * d2thdt2 * t79 +
   dthdt * t445;
 t452 = t150 * t445;
 t456 =
   -nmu * t296 * t231 * t299 / 8.0 + t302 * t272 * t304 / 2.0 +
   t302 * t241 * t243 / 4.0 + t311 * t313 * t188 / 2.0 +
   t317 * t313 * t204 / 4.0 -
```

```
    t324 * t121 * t46 * t79 * t151 / 4.0 - t330 * t332 / 8.0 +
    t337 * t313 * t339 -
    3.0 / 4.0 * t310 * t342 * t100 * t234 * t346 -
    t350 * t323 * t121 * t353 / 2.0 -
    t359 * t121 * t46 * t133 * t151 + t368 * t120 * t338 * t236 +
    t372 * t373 * t376 / 8.0 - t372 * t121 * t383 / 4.0 -
    t330 * t43 * t447 * t110 / 8.0 + t317 * t234 * t452 / 8.0;
  t457 = t403 + t412 + t424 - t426 - t428;
  t458 = 2.0 * t150 * t457;
  t462 = (-1.0 / 2.0 * i) * nmu;
  t466 = t120 * mm;
  t472 = t121 * t182;
  t477 = t234 * t339;
  t479 = t43 * t159;
  t480 = t479 * t53;
  t484 = 1 / t39 / t38;
  t487 = t79 * t79;
  t488 = t150 * t487;
  t504 =
    2.0 * E * t403 + 2.0 * t101 * d2rdt2 + d3rdt3 * t38 +
    2.0 * d2rdt2 * t79 + drdt * t445;
  t509 = (1.0 / 8.0 * i) * nmu;
  t510 = t509 * t11;
  t511 = t510 * t231;
  t518 = t109 * t79;
  t527 = t159 * t53 * t56;
  t528 = t527 * t182;
  t537 = t302 * t231;
  t538 = t479 * t110;
  t541 = t181 * d2thdt2;
  t545 = t312 * t110;
  t548 =
    t311 * t234 * t458 / 4.0 +
    t462 * t11 * t271 * t131 * t466 * t257 * t352 +
    t302 * t85 * t358 * t472 / 4.0 + t335 * t176 * t231 * t477 +
    t337 * t480 * t339 - t310 * t230 * t484 * t234 * t488 / 4.0 +
    t321 * t357 * t106 * t472 / 4.0 -
    t330 * t504 * t51 * t110 / 8.0 +
    t511 * t234 * mm * d2phidt2 * t150 +
    t366 * t230 * t44 * t466 * t257 * t518 -
    t330 * t106 * t159 * t110 / 4.0 + t372 * t528 / 4.0 +
    t311 * t480 * t188 / 2.0 + t317 * t480 * t204 / 4.0 -
    t537 * t538 / 4.0 + t372 * t121 * t541 / 8.0 - t537 * t545 / 4.0;
  t549 = t456 + t548;
  t552 = t381 * nmu;
  t558 = t552 * t11 * t231 * t234 * t274 * t151 / 8.0;
  t564 = t132 * t110;
  t570 = t118 * t43;
  t576 = t174 * mm;
  t577 = t257 * t109;
  t578 = t576 * t577;
  t580 = a * t12;
  t581 = t580 * t16;
  t582 = t19 * nmu;
  t583 = t582 * t100;
  t585 = t338 * t56;
  t590 = t26 * nmu;
  t591 = t590 * t100;
  t607 = t97 * t357;
  t615 = (-1.0 / 8.0 * i) * t163 * t127;
  t619 = nmu * t12;
  t620 = t16 * t100;
  t625 = t138 * t100;
  t626 = t625 * t173;
  t627 = t619 * t626;
  t628 = t57 * t46;
  t630 = 2.0 * t133 * t3 * dthdt;
  t633 = t85 * t100;
```

```
t634 = t633 * t173;
t635 = t619 * t634;
t636 = t57 * t383;
t639 = t176 * t176;
t642 = t174 * t150;
t645 = t256 * t376;
t648 = t85 * t40;
t649 = t648 * t173;
t650 = t619 * t649;
t652 = t79 * t3 * dthdt;
t656 = t138 * t40;
t659 = t57 * t353;
t661 = t365 * t148;
t663 = d2phidt2 * t56 * t109;
t666 = t619 * t648;
t670 = t619 * t633;
t675 = (1.0 / 2.0 * i) * nmu;
t676 = t675 * t12;
t678 = t265 * dphidt;
t679 = t678 * t204;
t681 = (0.0 - i) * nmu;
t683 = t633 * t51;
t691 = t85 * t484;
t696 =
  3.0 / 2.0 * t619 * t620 * t174 * t346 + t627 * t628 * t630 +
  t635 * t636 / 2.0 + nmu * t639 * t633 * t642 / 2.0 -
  t635 * t645 / 4.0 + t650 * t628 * t652 / 2.0 +
  t619 * t656 * t173 * t659 + t661 * t576 * t663 -
  t666 * t174 * t452 / 4.0 + t670 * t120 * t150 * t447 / 2.0 +
  t676 * t649 * t679 + t681 * t12 * t683 * t678 * t160 +
  t681 * t11 * t634 * t678 * t177 + t619 * t691 * t174 * t488 / 2.0;
t697 = t310 * t634;
t703 = t109 * t176;
t707 = t174 * t331;
```

```
 t726 = t159 * t159;
 t731 = t676 * t634;
 t734 = t619 * t625;
 t738 = t678 * t188;
 t745 = t619 * t683;
 t747 = t57 * t541;
 t750 = t310 * t633;
 t755 =
   0.0 - t697 * t628 * t176 * t3 * dthdt -
   t310 * t649 * t57 * t703 * t79 + t670 * t707 / 4.0 -
   4.0 * t310 * t626 * t57 * t703 * t133 +
   2.0 * t310 * t683 * t57 * t703 * t159 -
   2.0 * t619 * t625 * t51 * t57 * t352 * t159 +
   t670 * t726 * t53 * t150 / 2.0 + t731 * t678 * t236 -
   t734 * t174 * t458 / 2.0 + (i) * t627 * t738 -
   t619 * t648 * t51 * t57 * t518 * t159 - t745 * t528 -
   t635 * t747 / 4.0 + t750 * t174 * t150 * t296 / 2.0;
 t758 = t83 * nmu;
 t760 = t466 * t160;
 t762 = t9 * nmu;
 t764 = t576 * t204;
 t766 = t762 * t148;
 t771 = t576 * t188;
 t773 = a * nmu;
 t774 = t773 * t148;
 t780 = t576 * t177;
 t787 = t163 / t15 / t14;
 t788 = t787 * t200;
 t792 =
   t229 * t241 * t243 / 4.0 + t251 +
   t90 * t87 * (t252 * nmu * t98 * t132 * t256 * t257 -
t261 * t263 * t267 / 8.0 +
t270 * t272 * t234 * t275 / 4.0 +
t270 * t241 * t234 * t280 / 8.0 -
t284 * t108 * t266 / 8.0 -
t284 * t191 * t266 / 8.0 + t17 * t29 * t549 +
t558) / 2.0 - t96 * t263 * t564 / 4.0 +
   t229 * t272 * t304 / 2.0 + t117 * t570 * t527 +
   (1.0 / 4.0 * i) * t95 * nmu * t607 * t477 +
   t581 * t583 * t234 * t585 / 4.0 -
   t197 * t198 * t591 * t183 / 8.0 +
   t114 * t11 * t21 * t176 * t100 * t43 * t121 +
   (1.0 / 4.0 * i) * t145 * nmu * t148 * t578 +
   (-1.0 / 4.0 * i) * t167 * nmu * t148 * t578 +
   t168 * t171 * t178 - t168 * t186 * t189 +
   t615 * t200 * t225 * t339 - t7 * (t8 * (t696 + t755) +
     t758 * t148 * t760 +
     t762 * t208 * t764 +
     t766 * t576 * t236 +
     t124 * nmu * t186 * t771 +
     t774 * t174 * t255 * t577 /
     4.0 +
     t758 * t171 * t780) / 2.0 -
   t788 * t225 * t188 / 4.0;
 t795 = t7 * t87;
 t797 = t537 * t299 / 8.0;
 t799 = t311 * t304 / 4.0;
 t801 = t317 * t243 / 8.0;
 t803 = t330 * t545 / 8.0;
 t805 = t330 * t538 / 8.0;
 t806 = t511 * t477;
 t808 = t372 * t472 / 8.0;
 t809 = t799 - t797 + t801 - t803 - t805 + t806 + t808;
 t814 = t17 * t29 * t809 - t284 * t267 / 8.0;
 t816 = t795 * t814 / 2.0;
 t817 = t87 * t814;
 t819 = t97 * t817 * rp;
 t822 = t57 * rp;
```

```
t824 = t17 * t12 * t129 * t132 * t822;
t825 = t115 * nmu;
t826 = t114 * t825;
t828 = t826 * t131 * t121;
t830 = t11 * t100;
t833 = t114 * t21 * t830 * t43 * t121;
t836 = (-2.0 * i) * a;
t839 = t87 * rp;
t840 = t138 * t29 * t839;
t842 = t7 * t91;
t843 = t842 * t87;
t848 =
  (i) * a * t11 * t88 + t836 * t12 * t840 - t843 / 4.0 +
  t97 * t92 * rp / 2.0;
t849 = t848 * nmu;
t852 = t849 * t98 * t564 / 4.0;
t862 = t57 * t182;
t865 =
  t750 * t178 / 2.0 - t734 * t189 / 2.0 - t666 * t210 / 4.0 +
  t670 * t161 / 2.0 + t661 * t578 - t635 * t862 / 4.0;
t866 = t8 * t865;
t867 = t619 * t85;
t869 = t274 * t109;
t870 = t225 * t869;
t871 = t114 * t867 * t870;
t872 = t866 + t871;
t877 = t11 * t138;
t878 = t877 * t140;
t884 = 2.0 * t19 * t142;
t887 = t85 * t19;
t888 = t887 * t142;
t889 = t888 / 2.0;
t892 = 2.0 * t877 * t143 * rp;
t894 = t97 * t142 / 2.0;
```

```
t896 = t97 * t19 / 2.0;
t897 =
  0.0 - t878 + 3.0 * t115 * t140 * rp - t139 * t26 / 2.0 -
  t139 * t884 / 2.0 + t889 - t892 + t894 + t896;
t898 = t897 * nmu;
t900 = t224 * t110;
t902 = t898 * t147 * t900 / 2.0;
t903 = t163 * t16;
t909 = 2.0 * t11 * t142;
t912 =
  0.0 - t878 + 3.0 * t903 * t165 * rp - t164 * t26 / 2.0 -
  t164 * t909 / 2.0 + t889 - t892 + t894 + t896;
t913 = t912 * nmu;
t916 = t913 * t147 * t900 / 2.0;
t917 = t128 * t200;
t919 = t917 * t900 / 4.0;
t920 = t150 * rp;
t923 = 3.0 / 4.0 * t788 * t225 * t920;
t925 = t223 * t900 / 8.0;
t928 = 2.0 * t197 * t198 * t142 * nmu;
t930 = t928 * t900 / 8.0;
t931 =
  t816 - t819 + t824 + t828 + t833 - t852 + t85 * t872 * rp -
  t7 * rp * t865 - t902 + t916 + t919 - t923 + t925 + t930;
t933 = (-1.0 / 8.0 * i) * nmu;
t936 = t234 * t869;
t940 = t234 * t274 * t352;
t944 = t234 * t274 * t518;
t947 = t933 * t11 * t231;
t958 =
  t933 * t176 * t231 * t936 + t336 * t272 * t940 +
  t510 * t241 * t944 + t947 * t313 * t869 + t947 * t480 * t869 -
  t372 * t373 * t577 / 8.0 + t510 * t367 * t466 * t236;
t961 = (-1.0 / 8.0 * i) * t46;
```

```
t964 = t256 * t56;

t977 = t265 * t150;

t978 = t132 * t977;

t988 = t256 * t577;

t993 = t266 * t182;

t998 = t255 * t56 * t109;

t1012 = (1.0 / 8.0 * i) * t163 * t127;

t1021 = t90 * t817 / 2.0;

t1023 = t117 * t570 * t121;

t1025 = t99 * t564 / 4.0;

t1030 = t146 * t147 * t900 / 2.0;

t1033 = t168 * t147 * t900 / 2.0;

t1035 = t201 * t900 / 8.0;

t1036 = t1021 + t1023 - t1025 - t7 * t872 / 2.0 - t1030 + t1033 + t1035;

t1046 = t146 * t98;

t1053 = (-1.0 / 8.0 * i) * a;

t1054 = t1053 * nmu;

t1057 = t1054 * t98;

t1060 = (1.0 / 8.0 * i) * a;

t1061 = t1060 * nmu;

t1067 = t1061 * t607;

t1071 = t114 * nmu;

t1075 = t773 * t11;

t1076 = t1075 * t231;

t1077 = t255 * dphidt;

t1085 = t146 * t11;

t1092 = t197 * t12;

t1093 = t590 * t87;

t1094 = t1092 * t1093;

t1097 = t590 * t357;

t1101 = t124 * t11;

t1102 = t29 * t87;

t1106 = t1101 * t887 * t1102 + (i) * t580 * t88;

t1107 = t1106 * nmu;
```

```
t1108 = t43 * t43;
t1109 = t1108 * t40;
t1111 = t85 * t53;
t1115 = t1108 * t100;
t1116 = t1107 * t1115;
t1117 = t138 * t53;
t1121 = t1115 * t85;
t1127 = t150 * t106;
t1135 = t1085 * t231;
t1145 = t787 * t12;
t1146 = t1145 * t1093;
t1147 = t57 * t352;
t1151 = t57 * t518;
t1156 = t221 * nmu * t87;
t1162 = nmu * t1108;
t1163 = t1162 * t100;
t1172 = t1162 * t633;
t1175 = t109 * t291;
t1176 = t57 * t1175;
t1179 = nmu * t43;
t1181 = t109 * t106;
t1190 = t633 * t53;
t1199 = t1162 * t648;
t1204 = t1162 * t625;
t1209 = t106 * t106;
t1212 = t1111 * t150;
t1215 = t131 * t85;
t1220 = t1109 * t85;
t1223 = t1115 * t138;
t1226 = t933 * t1121;
t1245 = t1179 * t633;
t1263 =
  t335 * t1220 * t679 + t675 * t1223 * t738 +
  t1226 * t265 * t663 - t1172 * t645 / 8.0 +
    t335 * t1108 * t1190 * t585 * t182 +
    t1162 * t648 * t53 * t235 * t652 / 4.0 +
    3.0 / 4.0 * t1162 * t620 * t57 * t109 * t345 +
    t1245 * t57 * t109 * t504 / 4.0 +
    t1162 * t691 * t57 * t109 * t487 / 4.0 +
    t1162 * t656 * t659 / 2.0 + t1162 * t625 * t53 * t235 * t630 / 2.0;
  t1276 = t261 * t1115;
  t1285 = t56 * t3;
  t1289 = t552 * t1121 * t265 * t1285 * dthdt / 8.0;
  t1294 =
    0.0 - t1107 * t131 * t1111 * t1127 / 4.0 - t1046 * t111 / 4.0 +
    t1094 * t111 / 8.0 + t1135 * t237 / 4.0 +
    t197 * t11 * t590 * t176 * t358 * t120 * t150 / 8.0 -
    t1146 * t132 * t1147 / 4.0 - t1094 * t52 * t1151 / 8.0 +
    t615 * t1156 * t131 * t120 * t339 +
    (1.0 / 8.0 * i) * t1106 * t1163 * t1111 * mm * t577 +
    (1.0 / 8.0 * i) * t145 * nmu * t607 * t477 -
    t90 * t87 * (t17 * t29 *
 (t1172 * t636 / 4.0 + t1172 * t1176 / 8.0 -
  2.0 * t1179 * t625 * t57 * t1181 * t133 -
  t1179 * t648 * t57 * t1181 * t79 / 2.0 -
  t1179 * t1190 * t235 * t106 * t3 * dthdt / 2.0 -
  t1172 * t747 / 8.0 -
  t1199 * t57 * t109 * t445 / 8.0 -
  t1204 * t57 * t109 * t457 / 2.0 +
  nmu * t1209 * t100 * t1212 / 4.0 +
  t462 * t1215 * t678 * t1127 + t1263) +
 t261 * t131 * t1111 * t274 * t106 / 4.0 -
 t261 * t1109 * t1111 * t280 / 8.0 -
 t1276 * t1117 * t275 / 4.0 +
 t961 * t1163 * t1111 * t1077 * t56 - t1289) / 2.0;
  t1300 = t8 * t809 + t1057 * t978;
  t1302 = t67 * t29;
  t1303 = t85 * t1300 * t1302;
```

```
t1304 = t877 * t19;
t1305 = t26 * a;
t1306 = t1305 * t29;
t1309 = t3 * t29;
t1310 = t26 * t2 * t1309;
t1313 = (-1.0 / 2.0 * i) * t12;
t1316 = t124 * t29;
t1317 = t1316 * t888;
t1321 = 2.0 * t1304 * t142 * t2 * t1309;
t1322 = (1.0 / 2.0 * i) * t11;
t1324 = a * t29;
t1326 = t1322 * t85 * t1324 * t142;
t1327 =
  (i) * t1304 * t1306 - 3.0 * t116 * t1310 +
  t1313 * t138 * t1306 + t1317 + t1321 + t1326;
t1328 = t1327 * nmu;
t1331 = t1328 * t98 * t564 / 4.0;
t1335 = t56 * a;
t1338 = (1.0 / 4.0 * i) * t19 * t127 * t222 * t358 * t120 * t1335;
t1339 = t57 * t67;
t1342 = 3.0 / 4.0 * t1146 * t132 * t1339;
t1343 = t1305 * nmu;
t1345 = t11 * t87;
t1346 = t1345 * t100;
t1349 = t615 * t1343 * t1346 * t233 * t57;
t1350 = t83 * t29;
t1353 = t1245 * t57 * t1181 / 4.0;
t1355 = t1199 * t1151 / 8.0;
t1357 = t1204 * t1147 / 4.0;
t1359 = t1226 * t265 * t577;
t1361 = t1172 * t862 / 8.0;
t1362 = t1353 - t1355 - t1357 + t1359 - t1361;
t1364 = t17 * t29 * t1362;
t1367 = t1276 * t1111 * t274 / 8.0;
```

```
 t1368 = t1364 + t1367;
 t1375 = t17 * t3 * t1362;
 t1380 = t552 * t1115 * t1111 * t274 * t3 / 8.0;
 t1388 = t2 * a;
 t1389 = (-2.0 * i) * t1388;
 t1390 = t1389 * t877;
 t1391 = t19 * t30;
 t1392 = t87 * t3;
 t1395 = t2 * t11;
 t1398 = t1395 * t85 * t30 * t87;
 t1405 = t138 * t30 * t1392;
 t1407 = t85 * t3;
 t1408 = t1407 * t87;
 t1412 =
   (0.0 - t31 * t887 * t87 / 2.0 + t1390 * t1391 * t1392 +
    5.0 / 2.0 * t1398 + t1101 * t887 * t1392 +
    (4.0 * i) * t1388 * t12 * t1405 + (i) * t580 * t1408) * nmu * t1115;
 t1414 = t1412 * t1212 / 8.0;
 t1415 = (0.0 - i) * t109;
 t1453 = t1111 * t869;
 mgv1 =
   t17 * t29 * (0.0 - t1046 * t192 / 4.0 - t146 * t263 * t564 / 4.0 +
t7 * (t8 * t549 + t1054 * t263 * t978 +
      t1057 * t108 * t977 +
      t1061 * t97 * t322 * t944 +
      t1057 * t191 * t977 +
      t1067 * t233 * t265 * t236 +
      t1071 * t877 * t357 * t940 -
      t1076 * t234 * t1077 * t150 / 8.0) / 2.0 +
t1085 * t272 * t304 / 2.0 +
t1085 * t241 * t243 / 4.0 + t1094 * t192 / 8.0 -
t1092 * t1097 * t237 / 8.0 +
t1107 * t1109 * t1111 * t204 / 8.0 +
t1116 * t1117 * t188 / 4.0 +
t1107 * t1121 * t862 / 8.0 + t1294) + t1303 -
   t1331 + t1338 + t1342;
 t1457 =
   mgv1 + t1349 + t1350 * t795 * t1368 -
   t98 * t1368 * t2 * t1309 - t90 * t87 * (t1375 - t1380) / 2.0 -
   t1414 +
   t1415 * (t7 * (t8 * t958 + t1076 * t234 * t998 / 8.0) / 2.0 +
    (-1.0 / 8.0 * i) * t145 * nmu * t98 * t978 +
    t1012 * t1156 * t978 -
    t90 * t87 * (t17 * t29 *
 (t335 * t1215 * t265 * t1127 +
  t933 * t1220 * t265 * t204 +
  t365 * t1223 * t265 * t188 +
  t1172 * t988 / 8.0 + t1226 * t993) +
 t252 * t1162 * t633 * t964) / 2.0 +
    (-1.0 / 8.0 * i) * t1106 * t1163 * t1453);
 t1461 = t124 * t97;
 t1462 = t19 * t29;
 t1464 = t7 * t1300 / 2.0;
 t1466 = t1046 * t564 / 4.0;
 t1468 = t1094 * t564 / 8.0;
 t1473 = t1116 * t1212 / 8.0;
 t1474 = t1464 - t1466 + t1468 - t90 * t87 * t1368 / 2.0 - t1473;
 t1475 = t87 * t1474;
 t1483 = (2.0 * i) * a;
 t1484 = t1483 * t12;
 t1489 = t173 * dthdt;
 t1490 = t1489 * t110;
 t1493 = t233 * dthdt;
 t1494 = t1493 * t110;
 t1499 = t224 * dthdt;
 t1503 = dthdt * t53;
 t1504 = t1503 * t56;
 t1505 = t132 * t1504;
```

```
t1507 = t51 * dthdt;
t1508 = t1507 * t53;
t1523 = t235 * t3;
t1537 =
  t368 * t1508 * t339 - t324 * t1508 * t204 / 4.0 -
  t359 * t1508 * t188 / 2.0 + t330 * t312 * dthdt * t110 / 4.0 -
  t372 * t51 * t291 * t53 * t1523 / 4.0 + t537 * t1494 / 4.0 +
  t330 * t233 * d2thdt2 * t110 / 8.0 + t330 * t479 * dthdt * t110 / 4.0;
t1540 = t270 * t231;
t1541 = t1493 * t266;
t1544 =
  t17 * t29 * t1537 - t797 + t799 + t801 - t803 - t805 + t808 +
  t1540 * t1541 / 8.0 + t806;
t1550 = t1503 * t188;
t1553 = t1503 * mm * t577;
t1556 = t291 * t53 * t1523;
t1565 = t1503 * t204;
t1570 = t1489 * t53;
t1571 = t1570 * t869;
t1573 =
  t8 * (0.0 - t697 * t1503 * t177 + t627 * t1550 + t731 * t1553 +
t635 * t1556 / 2.0 - t745 * t1503 * t160 -
t670 * t173 * d2thdt2 * t110 / 4.0 +
t650 * t1565 / 2.0) + t766 * t1571;
t1581 = t1057 * t1505 - t330 * t299 / 8.0;
t1582 = t795 * t1581;
t1584 = t87 * t1581;
t1585 = t1584 * rp;
t1587 = t16 * t8;
t1588 = t619 * t100;
t1589 = t1587 * t1588;
t1598 =
  t26 * (t1582 / 2.0 - t97 * t1585 -
 t1589 * t1570 * t920 / 4.0 + t138 * rp * t1588 * t1490 / 4.0);
```

```
t1602 = t1076 * t1541 / 8.0 + t947 * t936;
t1606 = (1.0 / 8.0 * i) * t138;
t1612 =
  t8 * (0.0 - t169 * t1490 / 2.0 + t232 * t1494 / 4.0 +
t149 * t1490 / 2.0 - t201 * t1499 * t110 / 8.0 +
t22 * t1505 + t90 * t87 * t1544 / 2.0 -
t7 * t1573 / 2.0) - t1598 +
  a * (t90 * t87 * t1602 / 2.0 + t1606 * t8 * t1588 * t1571);
t1617 = t138 * t8;
t1618 = t1617 * t1588;
t1621 = t90 * t1584 / 2.0 + t1618 * t1490 / 8.0;
t1632 = t1493 * t977;
t1634 = t8 * t1537 + t1067 * t1632;
t1637 = t85 * dthdt;
t1638 = t1637 * t110;
t1651 = t335 * t1121;
t1658 =
  t1199 * t1565 / 4.0 - t1172 * d2thdt2 * t53 * t150 / 8.0 +
  t1204 * t1550 / 2.0 + t1651 * t1553 + t1172 * t1556 / 4.0 -
  t1245 * t1503 * t1127 / 2.0;
t1661 = t1637 * t266;
t1664 =
  t17 * t29 * t1658 + t1353 - t1355 - t1357 + t1359 - t1361 -
  t1276 * t1661 / 8.0;
t1668 = t1503 * t150;
t1669 = t132 * t1668;
t1672 =
  t7 * t1634 / 2.0 + t1116 * t1638 / 8.0 + t1135 * t1494 / 4.0 -
  t90 * t87 * t1664 / 2.0 - t1094 * t1669 / 8.0;
t1675 = t1587 * nmu;
t1676 = t1675 * t1346;
t1677 = t1493 * t1339;
t1679 = t1676 * t1677 / 8.0;
t1681 = t633 * t1504;
```

```
t1685 = t1060 * t1162 * t1681 + t1163 * t1212 / 8.0;

t1686 = t795 * t1685;

t1687 = t1350 * t1686;

t1688 = t1685 * t2;

t1690 = t98 * t1688 * t1309;

t1691 = t3 * nmu;

t1697 = t1060 * t1691 * t1108 * t633 * dthdt * t110 + t1364 + t1367;

t1701 = (1.0 / 16.0 * i) * t138;

t1703 = t310 * t357;

t1711 = 0.0 - t773 * t1115 * t1661 / 8.0 + t509 * t1115 * t1453;

t1715 = t1701 * t8 * t1703 * t1632 - t90 * t87 * t1711 / 2.0;

t1717 =

  t17 * t29 * t1672 + t1679 + t1464 - t1466 + t1468 + t1687 -

  t1690 - t90 * t87 * t1697 / 2.0 - t1473 + t1415 * t1715;

t1722 = t1617 * nmu * t1346;

t1724 = t1722 * t1494 / 16.0;

t1727 = t90 * t87 * t1685 / 2.0;

t1728 = t1724 - t1727;

t1729 = t87 * t1728;

t1743 = t26 * t1621;

t1744 = t1324 * t1743;

t1750 = (1.0 / 2.0 * i) * t163 * t138;

t1755 = (-1.0 / 16.0 * i) * t11;

t1757 = t87 * a;

t1758 = t1757 * t3;

t1759 = t1755 * t138 * t1758;

t1761 = t1759 * t1163 * t1668;

t1762 = t1679 + t1687 - t1690 + t1761;

t1763 = t87 * t1762;

t1772 = t91 * t91;

t1774 = (-1.0 - t1772) * t87;

t1794 = t274 * t67;

t1802 = t877 * t87;

t1803 = t10 * nmu;
```

```
t1812 = t127 * t8 * nmu;
t1813 = t1345 * t131;
t1814 = t1812 * t1813;
t1815 = t56 * t29;
t1816 = t2 * t2;
t1817 = t1816 * t4;
t1821 = t1814 * t1508 * t1815 * t1817 / 2.0;
t1822 = t29 * t2;
t1826 = t1676 * t1493 * t57 * t1822 / 8.0;
t1827 = t1675 * t1813;
t1831 = t1827 * t1508 * t150 * t5 / 8.0;
t1832 = (2.0 * i) * t1388;
t1833 = t1832 * t30;
t1836 = t1833 * t230 * t1685 * t3;
t1839 =
  t1415 * ((1.0 / 8.0 * i) * t16 * t8 * t1703 * t233 * t1503 *
   t1794 + t1350 * t795 * t1711 -
   t98 * t1711 * t2 * t1309 +
   t1802 * t1803 * t1115 * dthdt * t977 / 16.0) +
  t1338 + t1821 - t1826 + t1831 + t1303 - t1331 + t1836 +
  t1342 + t17 * t3 * t1672 + t1349;
t1842 = t2 * t138;
t1844 = t1842 * t1392 * nmu;
t1846 = t1844 * t1115 * t1504;
t1848 = t83 * t3;
t1849 = t1848 * t1686;
t1851 = t4 * t30;
t1854 = 4.0 * t1802 * t1685 * t1816 * t1851;
t1858 = t98 * t1688 * t30;
t1860 = t98 * t1688 * t4;
t1861 = (-1.0 / 8.0 * i) * t11;
t1862 = t1861 * t16;
t1863 = t87 * t1388;
t1864 = t1863 * t4;
```

```
 t1866 = t1163 * t1504;
 t1871 = t1053 * t1162;
 t1897 = (-1.0 / 4.0 * i) * t19 * t127;
 t1902 = t1145 * t1097;
 t1913 =
   t1350 * t795 * t1697 - t1846 / 16.0 + t1849 - t1854 +
   (i) * t46 * t1715 * t3 + t1858 - t1860 +
   t1862 * t1864 * t1866 - t98 * t1697 * t2 * t1309 -
   t90 * t87 * (t1871 * t1681 + t1375 - t1380) / 2.0 - t1414 +
   t17 * t29 * (t85 * t1634 * t1302 + t1412 * t1638 / 8.0 +
t1328 * t11 * t231 * t1494 / 4.0 +
t1350 * t795 * t1664 - t98 * t1664 * t2 * t1309 -
t90 * t87 * (t17 * t3 * t1658 + t1289) / 2.0 +
t1897 * t1156 * t132 * t1503 * t1335 -
3.0 / 4.0 * t1902 * t1677 +
t1012 * t1305 * t310 * t358 * t1507 * t57);
 t1920 = t30 * t87;
 t1926 = t1621 * t2 * t1309;
 t1929 =
   (-2.0 * i) * t12 * t138 * t1744 + t9 * t29 * t842 * t1729 +
   t1750 * t1744 + t98 * t1717 * t2 * t1309 +
   t1461 * t1462 * t1763 + t1390 * t1391 * t1729 * t3 +
   t1461 * t19 * t3 * t1729 + t90 * t1774 * t1728 / 4.0 -
   t31 * t85 * t19 * t87 * t1728 / 2.0 +
   t90 * t92 * t1762 / 4.0 + t85 * t1612 * t1302 +
   t1316 * t795 * t1717 + t90 * t87 * (t1839 + t1913) / 2.0 +
   (8.0 * i) * t1388 * t139 * t1920 * t1728 * t3 -
   12.0 * t115 * t140 * t1926;
 t1938 = (-1.0 / 8.0 * i) * t12;
 t1944 = t1570 * t56 * t2 * t3;
 t1947 =
   t1316 * t1582 + t98 * t1581 * t2 * t1309 +
   t1938 * t138 * t1803 * t1669 + t1589 * t1944 / 4.0;
 t1963 = t903 * t11;
```

```
t1967 =
  (0.0 - i) * t11 * t138 * t140 * t1324 - 3.0 * t1963 * t1310 +
  t1750 * t1306 + t1317 + t1321 + t1326;
t1977 = t97 * t91;
t1984 =
  t1398 + t1832 * t12 * t1405 + t84 * t1408 +
  t114 * t29 * t843 - t1977 * t87 * t2 * t1309 / 2.0 - t90 * t1774 / 4.0;
t1998 = t787 * t198 * t591;
t2003 = t12 * t100 * t173;
t2036 =
  0.0 - t1967 * nmu * t148 * t1490 / 2.0 +
  t1984 * nmu * t11 * t231 * t1494 / 4.0 +
  t1328 * t148 * t1490 / 2.0 + t1897 * t200 * t1499 * t57 * a -
  3.0 / 4.0 * t1998 * t1944 + t1012 * t1343 * t2003 * t1504 -
  t1822 * t16 * t582 * t11 * t1505 / 4.0 +
  (0.0 - i) * t1388 * t12 * t128 * t29 * nmu * t250 +
  t2 * t12 * t16 * t29 * nmu * t1505 / 4.0 + t251 +
  t1316 * t795 * t1544 + t98 * t1544 * t2 * t1309 +
  t90 * t87 * (t17 * t3 * t1537 - t558) / 2.0 - t85 * t1573 * t1302;
t2040 = (-1.0 / 8.0 * i) * t138;
t2042 = t1691 * t11;
t2045 = t1324 * t85;
t2048 = rp * t2;
t2049 = t2048 * t1309;
t2052 = (1.0 / 4.0 * i) * t12;
t2053 = t2052 * t16;
t2054 = t10 * t118;
t2057 = t57 * t109 * rp;
t2060 = t1812 * t2003;
t2069 =
  t26 * (t230 * t1581 * t1302 + t2040 * a * t2042 * t1669 +
 (i) * t2045 * t1585 - 4.0 * t877 * t1584 * t2049 +
 t2053 * t2054 * t1493 * t2057 -
 t2060 * t1504 * t2048 * t3 + t16 * rp * t1588 * t1944 / 2.0);
 t2111 =
   9.0 / 2.0 * t1395 * t85 * t1920 * t1728 -
   2.0 * t139 * t140 * t1947 - t164 * t165 * t1947 / 2.0 +
   t97 * t92 * t1728 * t2 * t1309 / 2.0 + t7 * (t8 * t2036 -
t2069 +
a * (t1316 *
     t795 *
     t1602 +
     t98 *
     t1602 * t2 *
     t1309 +
     t139 * a *
     t1691 *
     t100 *
     t1632 /
     8.0 +
     (1.0 / 4.0 *
      i) * t16 *
     t8 * t1588 *
     t1570 *
     t1794)) /
   2.0 + 3.0 / 2.0 * t145 * t1947 - t1967 * t1621 -
   t167 * t1947 / 2.0 + 3.0 * t1327 * t1621 +
   3.0 * t1984 * t1728 + 3.0 * t95 * t1762 +
   t1484 * t1407 * t1729 + t1484 * t86 * t1763 -
   3.0 * t903 * t165 * t1926 + (3.0 * i) * t11 * t138 * t19 * t1744;
 t2114 = t7 * t8;
 t2116 = t57 * t291;
 t2117 = t132 * t2116;
 t2121 = t1061 * t98 * t2117 + t330 * t1494 / 8.0;
 t2127 = t90 * t87 * t2121 / 2.0 - t1618 * t707 / 8.0;
 t2133 = t1871 * t633 * t2116 - t1172 * t1668 / 8.0;
 t2139 = 0.0 - t90 * t87 * t2133 / 2.0 - t1722 * t332 / 16.0;
 t2142 = t17 * t29 * t2139 + t1724 - t1727;
```

```
t2147 = t85 * t8;
t2150 = t795 * t2121;
t2152 = t2121 * t2;
t2155 = (1.0 / 8.0 * i) * t12;
t2160 = t291 * t2;
t2165 =
  t1316 * t2150 + t98 * t2152 * t1309 +
  t2155 * t138 * t1803 * t132 * t1176 - t1589 * t180 * t2160 * t3 / 4.0;
t2168 = t795 * t2142;
t2170 = t2142 * t2;
t2175 = t795 * t2133;
t2177 = t2133 * t2;
t2180 = (1.0 / 16.0 * i) * t11;
t2190 =
  t1350 * t2175 - t98 * t2177 * t1309 +
  t2180 * t138 * t1758 * t1163 * t1176 -
  t1676 * t234 * t56 * t291 * t67 / 8.0;
t2193 =
  t17 * t3 * t2139 + t17 * t29 * t2190 + t1679 + t1687 - t1690 + t1761;
t2207 = t2127 * t2;
t2212 = t124 * t3;
t2214 = t1389 * t30;
t2235 = t138 * a;
t2236 = t2235 * nmu;
t2237 = t1938 * t2236;
t2244 = t29 * t291;
t2253 =
  t2212 * t2150 + t2214 * t230 * t2121 * t3 +
  t1842 * t2042 * t2117 / 4.0 +
  4.0 * t1802 * t2121 * t1816 * t1851 +
  (1.0 / 2.0 * i) * t12 * t16 * t1388 * t4 * nmu * t2117 -
  t98 * t2152 * t30 + t98 * t2152 * t4 + t2237 * t2117 -
  t2060 * t57 * t29 * t291 * t1816 * t4 +
  t1589 * t180 * t2244 * t2 / 4.0 - t1675 * t2003 * t110 * t426 / 4.0;
```

```
t2256 = t17 * t29;
t2276 = (-1.0 / 4.0 * i) * t11 * t16;
t2293 = t1163 * t2116;
t2299 = t271 * a;
t2300 = t1755 * t2299;
t2314 =
  t1848 * t2175 + t1833 * t230 * t2133 * t3 +
  t1844 * t1115 * t2116 / 8.0 -
  4.0 * t1802 * t2133 * t1816 * t1851 +
  (1.0 / 4.0 * i) * t11 * t16 * t1864 * t2293 +
  t98 * t2177 * t30 - t98 * t2177 * t4 + t2300 * t2293 -
  t1814 * t121 * t2244 * t1817 / 2.0 +
  t1676 * t234 * t1815 * t2160 / 8.0 - t1827 * t121 * t1175 * t5 / 8.0;
t2321 = t2180 * t2299;
t2323 =
  t1849 + t2276 * t1864 * t1866 + t1836 + t17 * t29 * t2314 +
  t1821 - t1826 + t1831 + (i) * t1324 * t2139 +
  t836 * t3 * t2190 - t1846 / 8.0 + t2321 * t1866 - t1854 + t1858 - t1860;
t2327 =
  4.0 * t1617 * t2127 * t1817 * t30 +
  2.0 * t2147 * t2165 * t1302 - t2147 * t2207 * t30 +
  t2147 * t2207 * t4 + t2114 * t2253 / 2.0 +
  t2256 * t795 * t2193 + t2212 * t2168 +
  t2214 * t230 * t2142 * t3 +
  4.0 * t1802 * t2142 * t1816 * t1851 +
  2.0 * t98 * t2193 * t2 * t1309 - t98 * t2170 * t30 +
  t98 * t2170 * t4 + t90 * t87 * t2323 / 2.0;
t2335 = drdt * t53;
t2337 = t2335 * mm * t577;
t2341 = t2335 * t204;
t2346 = t2335 * t56;
t2350 = t2335 * t188;
t2352 =
  0.0 - t670 * t173 * d2rdt2 * t110 / 4.0 + t731 * t2337 -
  t697 * t2335 * t177 + t650 * t2341 / 2.0 -
  t745 * t2335 * t160 + t635 * t2346 * t182 / 2.0 + t627 * t2350;
t2363 = t590 * t148;
t2372 = t173 * drdt;
t2373 = t2372 * t53;
t2374 = t2373 * t869;
t2376 =
  t8 * t2352 - t590 * t171 * t178 / 2.0 +
  t590 * t186 * t189 / 2.0 + t590 * t208 * t210 / 4.0 -
  t2363 * t161 / 2.0 + t2363 * t183 / 4.0 +
  (1.0 / 4.0 * i) * t26 * nmu * t148 * t578 + t766 * t2374;
t2379 = t233 * drdt;
t2380 = t2379 * t266;
t2384 = t51 * drdt * t53;
t2388 = t2379 * t110;
t2411 =
  0.0 - t372 * t2384 * t236 / 4.0 + t537 * t2388 / 4.0 -
  t359 * t2384 * t188 / 2.0 + t330 * t312 * drdt * t110 / 4.0 -
  t324 * t2384 * t204 / 4.0 + t330 * t479 * drdt * t110 / 4.0 +
  t368 * t2384 * t339 + t330 * t233 * d2rdt2 * t110 / 8.0;
t2414 = t1540 * t2380 / 8.0 + t17 * t29 * t2411;
t2415 = t87 * t2414;
t2418 = t132 * t2346;
t2421 = t224 * drdt * t110;
t2426 = t2372 * t110;
t2431 =
  0.0 - t7 * t2376 / 2.0 + t90 * t2415 / 2.0 + t22 * t2418 -
  t201 * t2421 / 8.0 + t232 * t2388 / 4.0 + t149 * t2426 / 2.0 -
  t169 * t2426 / 2.0;
t2433 = t773 * t100;
t2435 = t2379 * t822;
t2436 = t2053 * t2433 * t2435;
t2438 = t2040 * t1075 * t2418;
t2439 = t8 * nmu;
```

```
t2440 = t2439 * t148;
t2444 = 0.0 - t2440 * t2426 - t590 * t147 * t900;
t2445 = t85 * t2444 / 4.0;
t2446 = t2445 * rp;
t2447 = rp * nmu;
t2448 = t2447 * t148;
t2450 = t2448 * t2426 / 2.0;
t2451 = 2.0 * t142 * nmu;
t2454 = t2451 * t147 * t900 / 4.0;
t2455 = 0.0 - t2450 + t866 - t2454 + t871;
t2458 =
  t2436 + t1023 + t1021 + t2438 - t1025 + t2446 -
  t7 * t2455 / 2.0 - t1030 + t1033 + t1035;
t2470 =
  (-1.0 / 4.0 * i) * t8 * nmu * t148 * t2374 +
  (-1.0 / 4.0 * i) * t26 * t867 * t870;
t2475 =
  t8 * t2431 - t26 * t2458 + a * (t773 * t186 * t2380 / 8.0 -
  t7 * t2470 / 2.0);
t2479 = t7 * t2444 / 4.0;
t2481 = t2237 * t2418 - t2479 / 2.0;
t2485 = t140 * t2481;
t2495 = t590 * t11;
t2502 = t590 * t98;
t2507 = t2495 * t231;
t2510 = t2379 * t977;
t2516 =
  t8 * t2411 + t590 * t263 * t564 / 8.0 -
  t2495 * t272 * t304 / 4.0 - t2495 * t241 * t243 / 8.0 +
  t2502 * t111 / 8.0 + t2502 * t192 / 8.0 - t2507 * t237 / 8.0 +
  t1067 * t2510 + (-1.0 / 8.0 * i) * t26 * nmu * t607 * t477;
t2521 = t85 * drdt;
t2522 = t2521 * t110;
t2525 = t2335 * t150;
```

```
 t2526 = t132 * t2525;
 t2551 =
   t17 * t29 * (t1651 * t2337 + t1199 * t2341 / 4.0 -
t1172 * d2rdt2 * t53 * t150 / 8.0 -
t1245 * t2335 * t1127 / 2.0 +
t1204 * t2350 / 2.0 +
t1162 * t633 * drdt * t862 / 4.0) -
   t1276 * t2521 * t266 / 8.0;
 t2552 = t87 * t2551;
 t2559 = t2439 * t11 * t231;
 t2562 = t2559 * t2388 + t2502 * t564;
 t2563 = t85 * t2562 / 8.0;
 t2564 = t2563 * t1302;
 t2567 = t1115 * t2346;
 t2569 = t1822 * t271 * nmu * t2567 / 16.0;
 t2572 = t1115 * drdt;
 t2575 = t1862 * t1863 * nmu * t2572 * t57 * t1309;
 t2577 = t1759 * t1163 * t2525;
 t2586 =
   (1.0 / 8.0 * i) * t8 * nmu * t607 * t2510 +
   (1.0 / 8.0 * i) * t26 * nmu * t98 * t978;
 t2589 = t1757 * nmu;
 t2596 =
   t17 * t29 * (t7 * t2516 / 2.0 + t1135 * t2388 / 4.0 +
t1116 * t2522 / 8.0 - t1094 * t2526 / 8.0 -
t90 * t2552 / 2.0) + t2564 - t2569 + t2575 +
   t2577 + t1415 * (t7 * t2586 / 2.0 + t877 * t2589 * t2572 * t266 / 16.0);
 t2597 = t87 * t2596;
 t2601 = t7 * t2562 / 16.0;
 t2603 = t2300 * t1163 * t2346;
 t2604 = t2601 + t2603;
 t2605 = t87 * t2604;
 t2606 = t1462 * t2605;
 t2613 = t86 * t2605;
 t2618 = drdt * dthdt;
 t2622 = t330 * t2388;
 t2624 = t1071 * t98 * t132 * t2618 * t57 + t2622 / 8.0;
 t2625 = t87 * t2624;
 t2628 = t2372 * dthdt * t110;
 t2633 = t2440 * t2628 / 2.0 + t2363 * t1490 / 4.0;
 t2635 = t90 * t2625 - t7 * t2633;
 t2637 = t8 * t2635 / 2.0 - t1743;
 t2639 = t2379 * t1668;
 t2644 = 0.0 - t2559 * t2639 / 4.0 - t2507 * t1494 / 8.0;
 t2651 = t9 * t1163 * t2521 * t1504 - t1172 * t2525 / 8.0;
 t2652 = t87 * t2651;
 t2654 = t7 * t2644 - t90 * t2652;
 t2657 = t17 * t29 * t2654 / 2.0 + t2601 + t2603;
 t2658 = t87 * t2657;
 t2662 = t85 * t2637;
 t2664 = t795 * t2657;
 t2668 = t85 * t2644;
 t2670 = t795 * t2651;
 t2675 = (1.0 / 8.0 * i) * t11;
 t2677 = t1757 * t1691;
 t2679 = t2572 * t1668;
 t2684 =
   t2564 + t2575 + t17 * t3 * t2654 / 2.0 +
   t17 * t29 * (t2668 * t1302 + t1350 * t2670 -
t98 * t2651 * t2 * t1309 +
t2675 * t138 * t2677 * t2679) + t2577 - t2569;
 t2685 = t87 * t2684;
 t2691 = t795 * t2624;
 t2699 = t85 * t2633;
 t2701 =
   t1316 * t2691 + t98 * t2624 * t2 * t1309 +
   t2052 * t138 * t2054 * t2639 - t2699 * t1302;
 t2704 = t8 * t2701 - t26 * t1947;
```

```
t2717 = t26 * t2481;
t2722 = 0.0 - t2450 - t2454;
t2725 = t2438 + t2436 + t2446 - t7 * t2722 / 2.0;
t2746 = t2451 * t148;
t2768 = t9 * t825;
t2789 =
  t85 * t2376 * rp - t7 * (2.0 * rp * t2352 -
   t2451 * t171 * t178 / 2.0 +
   t2451 * t186 * t189 / 2.0 +
   t2451 * t208 * t210 / 4.0 -
   t2746 * t161 / 2.0 +
   t2746 * t183 / 4.0 +
   (1.0 / 2.0 * i) * t142 * nmu * t148 *
   t578) / 2.0 + t795 * t2414 / 2.0 -
  t97 * t2415 * rp + t9 * t11 * t21 * t2418 +
  (i) * t580 * t127 * t583 * t2435 + t2768 * t2418 -
  t917 * t2421 / 4.0 + 3.0 / 4.0 * t1998 * t2373 * t920 -
  t223 * t2421 / 8.0 - t928 * t2421 / 8.0 +
  t849 * t11 * t231 * t2388 / 4.0 + t898 * t148 * t2426 / 2.0 -
  t913 * t148 * t2426 / 2.0;
t2794 = t310 * t100;
t2796 = (1.0 / 2.0 * i) * t16 * a * t2794 * t2435;
t2797 = t826 * t2418;
t2801 = t57 * t1;
t2803 = (0.0 - i) * t12 * t127 * t2433 * t2379 * t2801;
t2806 = t138 * t2444 * t1;
t2808 = t85 * t2722 * rp;
t2811 = t670 * t2426;
t2815 = t670 * t642;
t2820 =
  t828 + t833 + t2796 + t2797 + t824 + t2803 + t816 - t819 -
  t852 - t2806 + t2808 + t2445 + t85 * t2455 * rp -
  t7 * (0.0 - t2811 / 2.0 + 2.0 * rp * t865 - t2815 / 2.0) / 2.0 -
  t902 + t916 + t919 - t923 + t925 + t930;
```

```
t2826 = t274 * rp;
t2850 = t2717 * rp;
t2858 =
  0.0 - t85 * t2475 * rp + t795 * t2596 / 2.0 - 5.0 * t877 * t2485 -
  t164 * t2717 / 2.0 - t97 * t2597 * rp - t167 * t2725 / 2.0 -
  2.0 * t139 * t2717 + 3.0 * t897 * t2481 +
  t7 * (2.0 * rp * t2431 + t8 * t2789 - 2.0 * t142 * t2458 -
t26 * t2820 + a * (t2235 * t2794 * t2380 / 8.0 -
   t581 * t570 * t2384 * t2826 / 4.0 +
   t85 * t2470 * rp -
   t7 * ((-1.0 / 2.0 * i) * rp * nmu *
 t148 * t2374 +
 (-1.0 / 2.0 * i) * t142 *
 t867 * t870) / 2.0)) / 2.0 -
  t912 * t2481 + 3.0 / 2.0 * t145 * t2725 +
  12.0 * t116 * t2850 + 3.0 * t848 * t2604 - t164 * t165 * t2725 / 2.0;
t2869 = t2451 * t11;
t2876 = t2451 * t98;
t2881 = t2869 * t231;
t2895 = t124 * t85;
t2898 = t1483 * t877;
t2937 = t1309 * rp;
t2939 = t138 * t2562 * t2 * t2937 / 2.0;
t2941 = t2447 * t11 * t231;
t2946 = t2941 * t2388 / 4.0 + t2876 * t564 / 8.0;
t2948 = t85 * t2946 * t1302;
t2951 = t2572 * t822;
t2953 = t1822 * t342 * nmu * t2951 / 8.0;
t2959 = t100 * drdt;
t2960 = t2959 * t53;
t2963 =
  (-1.0 / 8.0 * i) * t16 * t87 * t1388 * nmu * t1108 * t2960 * t1285 * t29;
t2964 = t1322 * t127;
t2970 = t2964 * t1863 * t1162 * t2960 * t1285 * t29 * rp;
```

```
  t2971 = (-1.0 / 16.0 * i) * t138;
  t2975 = t2971 * t87 * t1803 * t2572 * t110;
  t2976 = t2675 * t16;
  t2979 = t2976 * t2677 * t2572 * t2057;
  t2993 = t773 * t1108;
  t3013 = t2563 * rp;
  t3015 = t7 * t2946 / 2.0;
  t3017 = t2971 * t2589 * t2567;
  t3019 = t2976 * t2589 * t2951;
  t3020 = 0.0 - t3013 + t3015 + t3017 + t3019;
  t3021 = t87 * t3020;
  t3024 = t2605 * rp;
  mgv3 = 0.0 - t164 * t909 * t2481 / 2.0;
  mgv4 =
    t90 * t87 * (t17 * t29 *
 (0.0 - t85 * t2516 * rp +
  t7 * (2.0 * rp * t2411 +
t2451 * t263 * t564 / 8.0 -
t2869 * t272 * t304 / 4.0 -
t2869 * t241 * t243 / 8.0 +
t2876 * t111 / 8.0 + t2876 * t192 / 8.0 -
t2881 * t237 / 8.0 +
(-1.0 / 4.0 * i) * t142 * nmu * t607 *
t477) / 2.0 +
  t898 * t11 * t231 * t2388 / 4.0 +
  (t2895 * t1462 * t87 + t2898 * t1462 * t839 +
   (3.0 / 2.0 * i) * a * t11 * t88 +
   (-4.0 * i) * a * t12 * t840) * nmu * t1115 *
  t2522 / 8.0 - t128 * t12 * t1093 * t2526 / 4.0 +
  3.0 / 4.0 * t1902 * t2379 * t2057 -
  t197 * t26 * t310 * t87 * t2526 / 8.0 -
  t1092 * t2451 * t87 * t2526 / 8.0 -
  t795 * t2551 / 2.0 + t97 * t2552 * rp) - t2939 +
 t2948 + t2953 + t2963 + t2970 + t2975 + t2979 +
 t1415 * (0.0 - t85 * t2586 * rp +
  t7 * ((1.0 / 4.0 * i) * rp * nmu *
t607 * t2510 +
(1.0 / 4.0 * i) * t142 * nmu *
t98 * t978) / 2.0 +
  t271 * t2993 * t2959 * t266 / 16.0 -
  t11 * t16 * t87 * t2993 * t2960 *
  t2826 / 8.0)) / 2.0 + 3.0 * t1963 * t2850;
  mgv2 = mgv3 + mgv4;
  mgv1 =
    mgv2 + t2895 * t2606 + t1461 * t1462 * t3021 +
    t2898 * t1462 * t3024 - 2.0 * t139 * t884 * t2481;
  t3052 =
    mgv1 - 2.0 * t139 * t140 * t2725 +
    (-8.0 * i) * t12 * t2235 * t1102 * t2604 * rp +
    t842 * t2605 / 4.0 + t90 * t92 * t3020 / 4.0 -
    t1977 * t3024 / 2.0 + t1484 * t86 * t3021 +
    (7.0 / 2.0 * i) * a * t11 * t2613 + 3.0 * t95 * t3020;
  t3059 = t2625 * rp;
  t3065 = t2448 * t2628 + t2746 * t1490 / 4.0;
  t3071 =
    rp * t2635 + t8 * (t2691 / 2.0 - t97 * t3059 + t2699 * rp -
       t7 * t3065 / 2.0) - 2.0 * t142 * t1621 - t1598;
  t3075 = t2658 * rp;
  t3082 = 0.0 - t2941 * t2639 / 2.0 - t2881 * t1494 / 8.0;
  t3088 =
    0.0 - t2668 * rp + t7 * t3082 / 2.0 - t2670 / 2.0 + t97 * t2652 * rp;
  t3091 = t17 * t29 * t3088 - t3013 + t3015 + t3017 + t3019;
  t3180 =
    -4.0 * t138 * t2637 * t2 * t2937 + t85 * t3071 * t1302 +
    (i) * t2045 * t3075 + t1316 * t795 * t3091 +
    t795 * t2684 / 2.0 - t97 * t2685 * rp + t90 * t87 * (0.0 - t2939 +
 t2948 +
 t2963 +
 t2970 +
 t17 *
 t3 *
 t3088 +
 t17 *
 t29 *
 (-4.0 *
  t138 *
  t2644 *
  t2 *
  t2937 +
  t85 *
  t3082 *
  t1302 +
  t2256 *
  t230 *
  t2651 *
  rp -
  t230 *
  t2651 *
  t1302 +
  4.0 *
  t877 *
  t2652 *
  t2049 +
  t1606 *
  t87 *
  t1803 *
  t2679 +
  t2276 *
  t2677 *
  t1115 *
  t2618 *
  t2057) +
 t2975 +
 t2979 +
 t2953) /
    2.0 + t230 * t2657 * t1302 - 4.0 * t877 * t2658 * t2049 +
    t98 * t3091 * t2 * t1309 - t85 * t2704 * rp +
    t7 * (2.0 * rp * t2701 +
  t8 * ((i) * t2045 * t3059 + t230 * t2624 * t1302 -
4.0 * t877 * t2625 * t2049 +
(1.0 / 4.0 * i) * t138 * a * t1691 * t830 *
t2639 +
t1313 * t16 * t2054 * t233 * t2618 * t2057 +
4.0 * t138 * t2633 * t2 * t2937 -
t85 * t3065 * t1302) - 2.0 * t142 * t1947 - t2069) / 2.0;
  t3185 = t173 * t403 * t110;
  t3188 = t2440 * t3185 + t2363 * t2426;
  t3189 = t7 * t3188 / 4.0;
  t3193 = t403 * t53 * t56;
  t3194 = t132 * t3193;
  t3196 = 0.0 - t3189 / 2.0 + t2155 * t2236 * t3194;
  t3198 = t8 * t3196 - t2717;
  t3202 = (-1.0 / 2.0 * i) * t11 * t7;
  t3203 = t233 * t403;
  t3204 = t3203 * t110;
  t3207 = 0.0 - t2559 * t3204 - t2507 * t2388;
  t3210 = t1163 * t3193;
  t3212 = t7 * t3207 / 16.0 + t2321 * t3210;
  t3219 = t85 * t3198;
  t3223 = t85 * t3188 / 4.0;
  t3229 = t2448 * t3185 / 2.0 + t2746 * t2426 / 4.0;
  t3237 = t3203 * t822;
  t3239 =
    t3223 * rp - t7 * t3229 / 2.0 + t1606 * t1075 * t3194 +
    (-1.0 / 4.0 * i) * t12 * t16 * t2433 * t3237;
  t3243 = 2.0 * rp * t3196 + t8 * t3239 - 2.0 * t142 * t2481 - t26 * t2725;
  t3248 = t1324 * t3212;
  t3251 = t1324 * t3212 * rp;
  t3253 = t85 * t3207 / 8.0;
  t3259 = 0.0 - t2941 * t3204 / 4.0 - t2881 * t2388 / 8.0;
  t3266 = t1115 * t403;
  t3267 = t3266 * t822;
  t3269 =
    0.0 - t3253 * rp + t7 * t3259 / 2.0 +
    t1701 * t2589 * t1115 * t3193 + t1862 * t2589 * t3267;
  t3283 = t2052 * t2236;
  mgv5 =
    (t7 *
     (t8 * (t240 + t792) - t26 * t931 +
      a * (t90 * t87 *
   (t17 * t29 * t958 +
    t961 * nmu * t98 * t132 * t964) / 2.0 -
   t580 * t21 * t267 / 4.0 +
   (-1.0 / 4.0 * i) * t95 * nmu * t98 * t978 -
   t7 * (t8 *
 (t675 * t171 * t780 + t462 * t186 * t771 +
  t365 * t208 * t764 + t675 * t148 * t760 +
  t635 * t988 / 4.0 + t365 * t12 * t634 * t993) -
 t774 * t174 * t998 / 4.0) / 2.0 +
   (-1.0 / 4.0 * i) * t145 * t867 * t870 +
   (1.0 / 4.0 * i) * t167 * t867 * t870 +
   t1012 * t200 * t870)) / 2.0 +
     3.0 / 2.0 * t145 * t1036 - t167 * t1036 / 2.0 -
     2.0 * t139 * t140 * t1036 - t164 * t165 * t1036 / 2.0 +
     t90 * t87 * t1457 / 2.0 + t1461 * t1462 * t1475 +
     t90 * t92 * t1474 / 4.0 + 3.0 * t95 * t1474 +
     t1484 * t86 * t1475) * DelTH;
  mgv6 =
    (t7 * t1612 / 2.0 + 3.0 / 2.0 * t145 * t1621 -
     t167 * t1621 / 2.0 - 2.0 * t139 * t140 * t1621 -
     t164 * t165 * t1621 / 2.0 + t90 * t87 * t1717 / 2.0 +
     t1461 * t1462 * t1729 + t90 * t92 * t1728 / 4.0 +
     3.0 * t95 * t1728 + t1484 * t86 * t1729) * DelTH1 - (t1929 +
  t2111) * DelTH;
  mgv4 = mgv5 + mgv6;
  mgv3 =
    mgv4 + (t2114 * t2127 +
    t90 * t87 * t2142) * DelTH2 / 2.0 -
    2.0 * (t2147 * t2127 * t1302 + t2114 * t2165 / 2.0 +
   t1316 * t2168 + t98 * t2170 * t1309 +
   t90 * t87 * t2193 / 2.0) * DelTH1 + t2327 * DelTH;
  mgv4 = DelR;
  mgv2 = mgv3 * mgv4;
  mgv3 =
    ((t7 * t2475 / 2.0 + 3.0 / 2.0 * t145 * t2481 -
      t167 * t2481 / 2.0 - 2.0 * t139 * t2485 -
      t164 * t165 * t2481 / 2.0 + t90 * t2597 / 2.0 +
      t1461 * t2606 + t90 * t92 * t2604 / 4.0 +
      3.0 * t95 * t2604 + t1484 * t2613) * DelTH + (t7 * t2637 +
    t90 * t2658) *
     DelTH1 / 2.0 - (t2662 * t1302 + t1316 * t2664 +
     t90 * t2685 / 2.0 +
     t98 * t2657 * t2 * t1309 +
     t7 * t2704 / 2.0) * DelTH) * DelR1 -
    ((t2858 + t3052) * DelTH +
     (0.0 - t2662 * rp + t7 * t3071 / 2.0 + t2664 / 2.0 - t97 * t3075 +
      t90 * t87 * t3091 / 2.0) * DelTH1 - t3180 * DelTH) * DelR;
  mgv1 = mgv2 + mgv3;
  mgv2 =
    mgv1 + (t7 * t3198 / 2.0 + t3202 * t1757 * t29 * t3212) * DelTH * DelR2;
  mgv3 = mgv2;
  mgv5 =
    -2.0 * (0.0 - t3219 * rp + t7 * t3243 / 2.0 +
    (-1.0 / 2.0 * i) * t7 * t87 * t3248 +
    (i) * t98 * t3251 + t3202 * t1757 * t29 * t3269) * DelTH * DelR1;
  mgv6 =
    (4.0 * t138 * t3198 * t1 - 2.0 * t85 * t3243 * rp - t3219 +
     t7 * (0.0 - t3189 + t3283 * t3194 + 4.0 * rp * t3239 +
   t8 * (0.0 - t138 * t3188 * t1 + 2.0 * t85 * t3229 * rp +
 t3223 - t7 * (t670 * t3185 + t2811) / 4.0 +
 (-1.0 / 2.0 * i) * t16 * a * t2794 * t3237 +
 (i) * t12 * t127 * t2433 * t3203 * t2801 +
 t2768 * t3194) + t3283 * t2418 + t2479 -
   4.0 * t142 * t2725 - t26 * (t2796 + t2803 + t2797 -
       t2806 + 2.0 * t2808 +
       t2445 - t7 * (0.0 - t2811 -
     t2815) /
       4.0)) / 2.0 +
     (2.0 * i) * t85 * t87 * t3251 +
     (0.0 - i) * t7 * t87 * t1324 * t3269 +
     (-4.0 * i) * t11 * t271 * t1324 * t3212 * t1 +
     (2.0 * i) * t11 * t230 * t1324 * t3269 * rp +
     (i) * t98 * t3248 +
     t3202 * t1757 * t29 * (t138 * t3207 * t1 / 2.0 -
    2.0 * t85 * t3259 * rp - t3253 +
    t7 * (0.0 - t330 * t3204 - t2622) / 8.0 +
    (-1.0 / 4.0 * i) * t16 * t2589 *
    t3267 +
    t2964 * t2589 * t3266 * t2801 +
    t1861 * t342 * a * t3210)) * DelTH * DelR;
  mgv4 = mgv5 + mgv6;
  t3364 = mgv3 + mgv4;

//    Maple generated expression ends here

  delta = r * r + a * a - 2.0 * mass * r;
  cs2 = (r * r + a * a) * (r * r + a * a) - a * a * delta * stheta * stheta;
  rho = -1.0 / (r - a * i * ctheta);
  TT = (r * r + a * a * ctheta * ctheta) * t3364 / (rho * rho * rho * rho);


  TT = TT * (cos (-mm * a / (rpl - rm) * log ((r - rpl) / (r - rm))) +
     i * sin (-mm * a / (rpl - rm) * log ((r - rpl) / (r - rm))));


  tred[j + it - PTS / 2][k + ip - PTS / 2] =
    (8.0 * pie * delta * TT.real / cs2 / (r * r * r));
  timd[j + it - PTS / 2][k + ip - PTS / 2] =
    (8.0 * pie * delta * TT.img / cs2 / (r * r * r));



}

/* -------------------------------------------------- */
```