S.P.R.A.M.P.

Spline **R**egression **A**utonomous **M**otion **P**rofiler

Source Code: https://github.com/HaveANiceDay33/AutoMappingUI

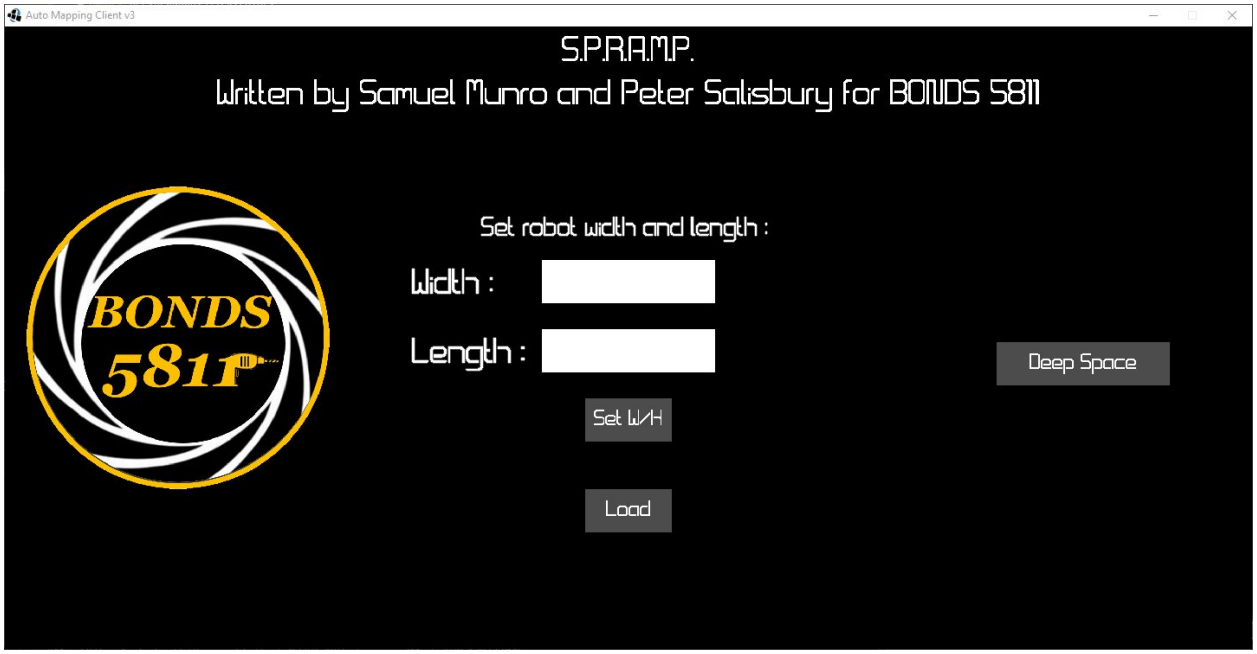
Welcome to SPRAMP, an autonomous motion profiler, written by Samuel Munro and Peter Salisbury for BONDS 5811. This program was written in Java using the graphics library RIDHVL, and performs a variety of tasks:

- Provides a platform to plot motion paths for FRC autonomous periods.
- Renders real time robot frame positions for collision detection.
- Provides an avenue to change motion variables (velocity, acceleration, angular velocity, angular acceleration, and starting angle) for use in the plotted paths.
- Generates a comprehensive motion profile, saved as a *.BOND file, to the user's machine.
- Allows users to load previously plotted profiles for editing and refinement.

There are 3 pieces to the system. First is the feed-forward application, which generates a comprehensive motion profile. Included in this is the Physics Model, which simulates the robot in math as if it were driving the planned path. During movements, sensors onboard the robot collect data that feedback is performed upon.

Feed-forward Simulation

SPRAMP has two main menus to navigate through. The first menu is the *Robot Geometry* menu. This is where users have the opportunity to input the dimensions of their robot (with bumpers, in inches), load previously drawn profiles, and, for BONDS 5811, automatically enter the next menu with the “Deep Space” button, which already contains the dimensions for this year's robot “Moonraker”.



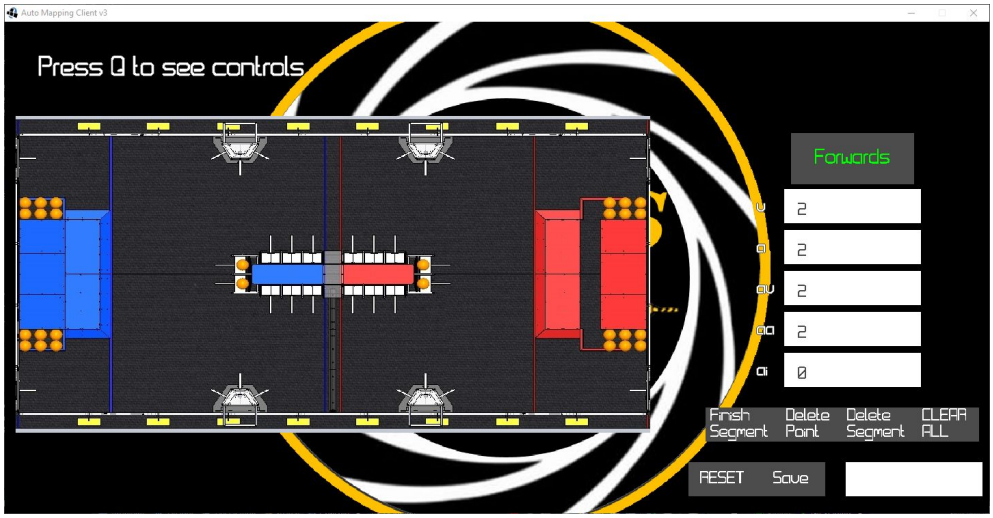
One small but notable feature of the first menu is the loading function. Along with *.BOND files, the program creates a “loader” file, which only contains the graphical coordinates of each point on the virtual field. When a “loader” file is selected from the file dialog that appears upon the press of the “Load” button, the coordinates are read in by the client and displayed on the field. This allows users to revisit previously drawn profiles, where they can edit any of the 4 kinematic parameters, location of points, and the direction of the profile. The functionality accelerates the entire process by not requiring users to redraw an entire path if they only wanted to make a small change.

The second menu is more complex. This is where the users can draw their desired paths, with a variety of tools. The controls on this menu are as follows:

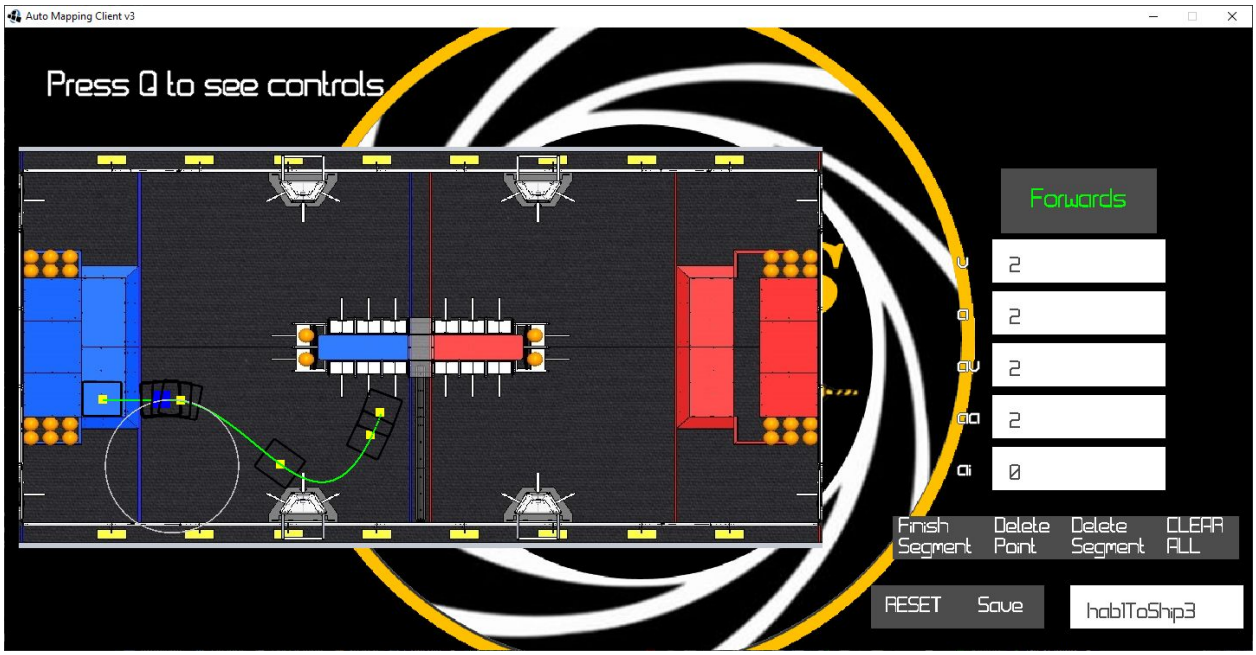
- Pressing Q to see instructions

- A toggle button to change the direction of the moving robot
- 5 dialog boxes with labels
    - Velocity (v)
    - Acceleration (a)
    - Angular Velocity (av)
    - Angular Acceleration (aa)
    - Initial Angle (ai)
- Finish segment button - saves a segment (series of points) to one path. Profiles can be many segments, or just one.
- Delete last point button
- Delete last segment button
- Clear all button
- Reset button - takes the user back to the first menu
- Save button and dialog box - User types in a name for their profile and clicks save, generating the *.BOND file.

Points can be placed by clicking the mouse anywhere on the field, and the field itself can be moved around and zoomed in/out of using right-click and scroll wheel, respectively.



Let's make a path: This path uses two segments, one that moves the robot off of Hab Lv.1 and another that steers the robot to the third bay of the cargo ship. Both paths are run with v = a = av = aa = 2. We can save the file as: hab1ToShip3, as done here:



Once the save button is clicked, the program generates a file called hab1ToShip3.BOND and hab1ToShip3Loader.BOND. The second file purely stores data point locations and is used for reloading the files into the program later on.

The contents of hab1ToShip3.BOND are located at the end of this document, as they are quite lengthy.

Now to dive into the generation of the *.BOND files:

The first step is to calculate the total arc length of each segment. To do this, we use a trapezoidal Riemann Sum that integrates along the curve using the arc length equation:

$$L = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

To calculate derivatives, we move the coefficients in a power rule fashion to adjust the original function generated from the drawing. In the end, the code for this Riemann Sum calculation looks like this:

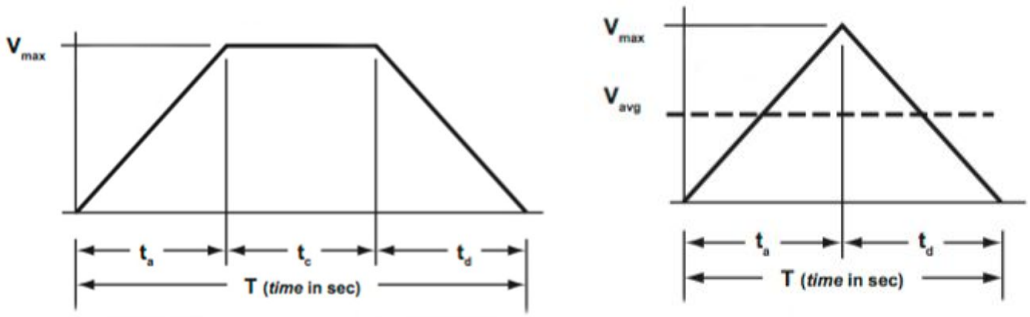
```
for(double i = step; i < finalX; i+=step) { //finalX is the furthest x-displacement the path will run to
   double a = Math.sqrt(1 + Math.pow((deriCoeff[4]*Math.pow(i, 4)) + (deriCoeff[3]*Math.pow(i, 3))+
         (deriCoeff[2]*Math.pow(i, 2))+(deriCoeff[1]*Math.pow(i, 1))+(deriCoeff[0]*Math.pow(i, 0)), 2));

   double b = Math.sqrt(1 + Math.pow((deriCoeff[4]*Math.pow(i-step, 4)) + (deriCoeff[3]*Math.pow(i-step, 3))+
   (deriCoeff[2]*Math.pow(i-step, 2))+(deriCoeff[1]*Math.pow(i-step, 1))+(deriCoeff[0]*Math.pow(i-step, 0)), 2));

   double stepVal = step * ((a+b)/2); //complies with trapezoidal Riemann sum h(f(a) + f(b))/2

   arcLength += stepVal;
}
```

After the calculating an arc length, the time needed to accelerate the robot to the set maximum velocity at the set maximum acceleration and the distance it will travel in that time are calculated using basic kinematic equations:

$$\Delta t_{acc} = \frac{v_{max}}{a_{max}} \qquad \Delta x_{acc} = \frac{1}{2} a_{max} (\Delta t)^2$$

Where Δt_{acc} is the time needed to accelerate to maximum speed and Δx_{acc} is the distance traveled in that time.

Using this Δx, the program decides if it is going to run a triangular or trapezoidal motion profile, depending on if the total arc length is greater than the acceleration distance times 2, which accounts for the deceleration distance, as well. The difference between trapezoidal and triangular profiles is that the trapezoidal will have a period of time where the robot runs at maximum velocity, where triangular will not. These two images dutifully represent this difference:



No matter the shape of the profile, a total time is calculated. This target time is used to halt movement when it is reached.

Trapezoidal Process:

From here, the process becomes quite repetitive, sinking into a series of 3 while-loops, one for each part of the profile. Each iteration around the loop represents 1 time step of the roboRIO in autonomous modes. Each iteration goes like this:

1. Current velocity is updated by the acceleration multiplied by the dt, the time between updates on the roboRIO, equal to 0.02 seconds. $vel = acc_{max} * dt$
2. Current position along the arc is updated by the velocity, times dt. $pos = vel * dt$
3. The instantaneous radius of a theoretical circle at that point in time is then calculated. This value is necessary for voltage calculations, discussed in a later section. We calculate this using the curvature equation, and then taking the inverse of that result. Overall, it looks like this:

$$r = \left(\frac{\frac{d^2y}{dx^2}}{(1+(\frac{dy}{dx})^2)^{\frac{3}{2}}}\right)^{-1}$$

4. After the instantaneous path radius is calculated, we determine the current x-displacement along the curve with a Riemann sum. The derivative of the coordinate at the calculated x-displacement is used to calculate angles with some trigonometry:

$$\theta = arctan(\frac{dy}{dx})$$

5. From here, angular velocity is calculated by taking the difference between the current angle and the previous angle, then dividing by dt.
6. Next, the left and right voltages are calculated with the `solveChassisDynamics()` method, explored more in the physics section of this document.
7. Finally, all of the calculated data is written to a user-defined *.BOND file with this method:

```
public static void writeLine(double vR, double vL, double disp, double vel, double ang, double angVel, double
rad, double pos) {
         try {
                  fileWriter.write(String.format("%.4f", vR) + " " + String.format("%.4f", vL) + " " +
                                    String.format("%.4f", disp) + " " + String.format("%.4f", vel)+ " "
                                   + String.format("%.4f", ang)+ " " + String.format("%.4f", angVel)+ "  " +
                           String.format("%.4f", rad)+ " " + String.format("%.4f", pos));
                  fileWriter.newLine();
          } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
          }
}
```

Triangular Process

The triangular generation process is identical to the previously explained trapezoidal process, except that the middle section (full speed motion) is not included, and time to drive is calculated a little differently:

$$\Delta t_{acc,\,tri} = \sqrt{L_{arc}/acc_{max}} \qquad \Delta t_{total} = 2 * \Delta t_{acc,\,tri}$$

After running through the entire path, the *.BOND file is generated completely. Each line of text represents one timestep or clock cycle of the roboRIO. An example line would look like this:

```
1.7068 1.7068 0.0010 0.0500 0.0000 0.0000 1000000.0000 0.0010
```

Where the numbers represent the right voltage(V), left voltage(V), position on the arc(m), velocity(m/s), angle(Θ), angular velocity(ω), path-radius(m), and x-position(m), respectively. Most of these numbers are used to run feedback on during actual driving, explored in a later section.

Physics Model

To generate a .BOND, the virtual robot moves along the user-specified path. The virtual robot is a rigid body dynamics model and a pair of DC motor models.

A single DC motor model can be used to simulate a linear motorized actuator, including a differential drive robot along a straight path. This model was graciously shared by FRC 971 Spartan Robotics.

The model for an ideal DC motor is as follows:

Eq. 1 $V = IR + \omega/kv$

To compensate for friction, a empirical constant is added to the voltage in the direction of the velocity.

Eq.2 $V = IR + \omega/kv + V_{int}$

To make this model useful for a mechanism, we substitute equations that transform motor specific information to become mechanism relevant: a gear ratio, G and a wheel radius, r.

Eq. 3 $\tau = I * G * kt$

Eq. 4 $F/r = I * G * kt$

Eq. 5 $m * a/r = I * G * kt$

Eq. 6 $V = \omega * G / kv$

Eq. 7 $V = \upsilon / r * G / kv$

The values of kv and kt and calculated from the manufacturer's published nominal values.

Finally, Eq. 4 and Eq. 6 can be substituted into Eq. 2

Eq. 8 $V = \frac{m*a*r}{G*kt} * R + \frac{\upsilon*G}{r*kv} + V_{int}$

With this model an elevator can be precisely controlled or a robot drive along a straight line with only the kinematic inputs of the of acceleration, a, and velocity, v.

By fusing a pair of motor models, the robot can drive along a curve. The model can provide for a driving plan along any curve of at least G2 continuity. This is achieved by decomposing a curve into a sequence of tiny circular arcs, as discussed previously.

The key equation that provides for driving along a circular arc relates the parth arc radius, r, tangential velocity, υ, and angular velocity, ω.

Eq. 9 $\upsilon = \omega * r$

To solve for the voltage to be applied to either side of the drivetrain, the translational velocity and angular velocity must be broken into left and right components. This is done velocity-wise and force-wise. We will start with the velocity-wise side. Right wheel velocity, υR, left wheel velocity, υL, and wheelbase width, W.

Eq. 10 $\upsilon = \frac{\upsilon R + \upsilon L}{2}$

Eq. 11 $\upsilon = \frac{\upsilon R - \upsilon L}{W} * r$

After setting these equations equal to each other, we can solve for the velocity of one side in terms of the other. To simplify the algebraic expressions we use equation 12.

Eq. 12 $j = \frac{2r}{W}$

Eq. 13 $\upsilon L = \frac{\upsilon R(j-1)}{(k+1)}$

Eq. 14 $\upsilon R = \frac{\upsilon R(j+1)}{(j-1)}$

By substituting Eq. 13 and Eq. 14 back into Eq. 10 the velocity for each side is given in terms of the overall translational velocity υ.

Eq. 15 $\upsilon L = \frac{\upsilon(j-1)}{j}$

Eq. 16 $\upsilon R = \frac{\upsilon(j+1)}{j}$

Now we have to determine the force to be applied to each side of the drivetrain. Starting again with a core equation relating the translational and angular acceleration. Translational acceleration, a, angular acceleration, α, and path arc radius, r.

Eq. 17 $a = \alpha * r$

Then we relate the torque applied by the force differential to the robot's angular acceleration. I is the robot's angular inertia.

Eq. 18 $\tau = \alpha * I$

Eq. 19 $(FR - FL) * \frac{W}{2}$

Eq. 20 $F = (FR + FL) = ma$

Now we substitute a rearranged Equation 19 and 20 into Equation 17 and solve for the force applied to one side in terms of the force applied to the other.

Eq. 21 $FL = \frac{FR(k-1)}{k}$

Eq. 22 $FR = \frac{FL(k+1)}{k}$

Now we substitute Equations 21 and 22 into Equation 20 and solve for the force applied to each side in terms of the acceleration. Algebraic expressions are simplified further using Equation 23.

Eq. 23 $k = \frac{W*r*m}{2I}$

Eq. 24 $FL = \frac{ma(k-1)}{2k}$

Eq. 25 $FR = \frac{ma(k+1)}{2k}$

Now, at each program cycle the desired kinematic properties and path radius are used to calculate the voltage needed on either side of the drivetrain.

Feedback Correction

In order to preserve consistency and stay as accurate to the drawn (planned) path as possible, we use onboard sensors to run feedback on our physics model plan. With just two sensors we are able to run feedback on 4 parameters: position(m), velocity(m/s), angle(Θ), and angular velocity(ω).

Collection and analysis:

Position: To collect position data, we survey both rotary encoders every single clock cycle and take the difference between that reading and the reading from the previous cycle to produce `deltaR` and `deltaL`, which represent the distance each encoder has travelled in one cycle. An average of these 2 values is calculated and added to an accumulating variable called `drivenDistanceSensor`. This variable is compared to the position that was planned, a difference is taken, and proportional feedback is performed on that error. For every meter of error, there are a KpPos amount of volts added to each side of the drivetrain.

Velocity: The velocity variable is not directly taken by a sensor; rather, it is calculated by dividing the `drivenDistanceSensor` (the position differential) by dt (the time differential). The calculated value is again compared to the planned value and proportional feedback is ran on the error. For every meter/second of error, there are a KpVel amount of volts added to the each side of the drivetrain.

Angle: The angle variable is calculated by reading the NavX, our onboard gyroscope. The measured angle is taken and the error between it and the plan have proportional feedback run on it. The strength of the proportional gain is tuned per segment, usually being stronger on straighter paths and weaker on large curved arcs. For every radian of error, there are a KpAng amount of volts applied to the drivetrain, in opposite directions.

Angular Velocity: Angular velocity is calculated in a similar manner to velocity. A differential of angle (one reading versus the reading from the cycle before it) is calculated, and divided by dt (the time differential). Again, the error between this value and the planned value has proportional feedback acting on it. For every radian/second of error, there are a KpAngVel amount of volts applied to the drivetrain, again in opposite directions.

By tuning each of the proportional gains for each segment, we can ensure that the robot drives as close as possible to the path that we drew for it in the application. Feedback ensures consistency and repeatability with our paths, solving a common problem with control systems.

Appendix: Contents of hab1ToShip3

```
1.4994 1.4994 0.0008 0.0400 0.0000 0.0000 1000000.0000 0.0010
1.6203 1.6203 0.0024 0.0800 0.0000 0.0000 1000000.0000 0.0030
1.7411 1.7411 0.0048 0.1200 0.0000 0.0000 1000000.0000 0.0050
1.8620 1.8620 0.0080 0.1600 0.0000 0.0000 1000000.0000 0.0090
1.9829 1.9829 0.0120 0.2000 0.0000 0.0000 1000000.0000 0.0120
2.1038 2.1037 0.0168 0.2400 0.0000 0.0000 1000000.0000 0.0170
2.2246 2.2246 0.0224 0.2800 0.0000 0.0000 1000000.0000 0.0230
2.3455 2.3455 0.0288 0.3200 0.0000 0.0000 1000000.0000 0.0290
2.4664 2.4664 0.0360 0.3600 0.0000 0.0000 1000000.0000 0.0360
2.5872 2.5872 0.0440 0.4000 0.0000 0.0000 1000000.0000 0.0440
2.7081 2.7081 0.0528 0.4400 0.0000 0.0000 1000000.0000 0.0530
2.8290 2.8290 0.0624 0.4800 0.0000 0.0000 1000000.0000 0.0630
2.9498 2.9498 0.0728 0.5200 0.0000 0.0000 1000000.0000 0.0730
3.0707 3.0707 0.0840 0.5600 0.0000 0.0000 1000000.0000 0.0850
3.1916 3.1916 0.0960 0.6000 0.0000 0.0000 1000000.0000 0.0970
3.3124 3.3124 0.1088 0.6400 0.0000 0.0000 1000000.0000 0.1090
3.4333 3.4333 0.1224 0.6800 0.0000 0.0000 1000000.0000 0.1230
3.5542 3.5542 0.1368 0.7200 0.0000 0.0000 1000000.0000 0.1370
3.6751 3.6751 0.1520 0.7600 0.0000 0.0000 1000000.0000 0.1530
3.7959 3.7959 0.1680 0.8000 0.0000 0.0000 1000000.0000 0.1690
3.9168 3.9168 0.1848 0.8400 0.0000 0.0000 1000000.0000 0.1850
4.0377 4.0377 0.2024 0.8800 0.0000 0.0000 1000000.0000 0.2030
4.1585 4.1585 0.2208 0.9200 0.0000 0.0000 1000000.0000 0.2210
4.2794 4.2794 0.2400 0.9600 0.0000 0.0000 1000000.0000 0.2400
4.4003 4.4003 0.2600 1.0000 0.0000 0.0000 1000000.0000 0.2600
4.5211 4.5211 0.2808 1.0400 0.0000 0.0000 1000000.0000 0.2810
4.6420 4.6420 0.3024 1.0800 0.0000 0.0000 1000000.0000 0.3030
4.7629 4.7629 0.3248 1.1200 0.0000 0.0000 1000000.0000 0.3250
4.8838 4.8837 0.3480 1.1600 0.0000 0.0000 1000000.0000 0.3480
5.0046 5.0046 0.3720 1.2000 0.0000 0.0000 1000000.0000 0.3720
5.1255 5.1255 0.3968 1.2400 0.0000 0.0000 1000000.0000 0.3970
5.2464 5.2464 0.4224 1.2800 0.0000 0.0000 1000000.0000 0.4230
5.3672 5.3672 0.4488 1.3200 0.0000 0.0000 1000000.0000 0.4490
5.4881 5.4881 0.4760 1.3600 0.0000 0.0000 1000000.0000 0.4760
5.6090 5.6090 0.5040 1.4000 0.0000 0.0000 1000000.0000 0.5040
5.7298 5.7298 0.5328 1.4400 0.0000 0.0000 1000000.0000 0.5330
5.8507 5.8507 0.5624 1.4800 0.0000 0.0000 1000000.0000 0.5630
5.9716 5.9716 0.5928 1.5200 0.0000 0.0000 1000000.0000 0.5930
6.0924 6.0924 0.6240 1.5600 0.0000 0.0000 1000000.0000 0.6240
6.2133 6.2133 0.6560 1.6000 0.0000 0.0000 1000000.0000 0.6560
6.3342 6.3342 0.6888 1.6400 0.0000 0.0000 1000000.0000 0.6890
6.4551 6.4551 0.7224 1.6800 0.0000 0.0000 1000000.0000 0.7230
4.9171 4.9171 0.7552 1.6400 0.0000 0.0000 1000000.0000 0.7560
4.7963 4.7963 0.7872 1.6000 0.0000 0.0000 1000000.0000 0.7880
4.6754 4.6754 0.8184 1.5600 0.0000 0.0000 1000000.0000 0.8190
4.5545 4.5545 0.8488 1.5200 0.0000 0.0000 1000000.0000 0.8490
4.4336 4.4336 0.8784 1.4800 0.0000 0.0000 1000000.0000 0.8790
4.3128 4.3128 0.9072 1.4400 0.0000 0.0000 1000000.0000 0.9080
4.1919 4.1919 0.9352 1.4000 0.0000 0.0000 1000000.0000 0.9360
4.0710 4.0710 0.9624 1.3600 0.0000 0.0000 1000000.0000 0.9630
3.9502 3.9502 0.9888 1.3200 0.0000 0.0000 1000000.0000 0.9890
3.8293 3.8293 1.0144 1.2800 0.0000 0.0000 1000000.0000 1.0150
3.7084 3.7084 1.0392 1.2400 0.0000 0.0000 1000000.0000 1.0400
3.5876 3.5876 1.0632 1.2000 0.0000 0.0000 1000000.0000 1.0640
3.4667 3.4667 1.0864 1.1600 0.0000 0.0000 1000000.0000 1.0870
3.3458 3.3458 1.1088 1.1200 0.0000 0.0000 1000000.0000 1.1090
3.2249 3.2249 1.1304 1.0800 0.0000 0.0000 1000000.0000 1.1310
3.1041 3.1041 1.1512 1.0400 0.0000 0.0000 1000000.0000 1.1520
2.9832 2.9832 1.1712 1.0000 0.0000 0.0000 1000000.0000 1.1720
2.8623 2.8623 1.1904 0.9600 0.0000 0.0000 1000000.0000 1.1910
2.7415 2.7415 1.2088 0.9200 0.0000 0.0000 1000000.0000 1.2090
2.6206 2.6206 1.2264 0.8800 0.0000 0.0000 1000000.0000 1.2270
2.4997 2.4997 1.2432 0.8400 0.0000 0.0000 1000000.0000 1.2440
2.3789 2.3789 1.2592 0.8000 0.0000 0.0000 1000000.0000 1.2600
2.2580 2.2580 1.2744 0.7600 0.0000 0.0000 1000000.0000 1.2750
2.1371 2.1371 1.2888 0.7200 0.0000 0.0000 1000000.0000 1.2890
2.0163 2.0163 1.3024 0.6800 0.0000 0.0000 1000000.0000 1.3030
1.8954 1.8954 1.3152 0.6400 0.0000 0.0000 1000000.0000 1.3160
1.7745 1.7745 1.3272 0.6000 0.0000 0.0000 1000000.0000 1.3280
1.6536 1.6536 1.3384 0.5600 0.0000 0.0000 1000000.0000 1.3390
1.5328 1.5328 1.3488 0.5200 0.0000 0.0000 1000000.0000 1.3490
1.4119 1.4119 1.3584 0.4800 0.0000 0.0000 1000000.0000 1.3590
1.2910 1.2910 1.3672 0.4400 0.0000 0.0000 1000000.0000 1.3680
1.1702 1.1702 1.3752 0.4000 0.0000 0.0000 1000000.0000 1.3760
1.0493 1.0493 1.3824 0.3600 0.0000 0.0000 1000000.0000 1.3830
0.9284 0.9284 1.3888 0.3200 0.0000 0.0000 1000000.0000 1.3890
0.8076 0.8076 1.3944 0.2800 0.0000 0.0000 1000000.0000 1.3950
0.6867 0.6867 1.3992 0.2400 0.0000 0.0000 1000000.0000 1.4000
0.5658 0.5658 1.4032 0.2000 0.0000 0.0000 1000000.0000 1.4040
0.4449 0.4449 1.4064 0.1600 0.0000 0.0000 1000000.0000 1.4070
0.3241 0.3241 1.4088 0.1200 0.0000 0.0000 1000000.0000 1.4090
0.2032 0.2032 1.4104 0.0800 0.0000 0.0000 1000000.0000 1.4110
0.0823 0.0823 1.4112 0.0400 0.0000 0.0000 1000000.0000 1.4120
-1.3785 -1.3785 1.4112 -0.0000 0.0000 0.0000 1000000.0000 1.4120
0.6424 2.3564 0.0008 0.0400 -0.0608 -3.0384 -1.3991 0.0010
0.7372 2.5033 0.0024 0.0800 -0.0622 -0.0716 -1.3996 0.0030
0.8324 2.6499 0.0048 0.1200 -0.0636 -0.0715 -1.4006 0.0050
0.9276 2.7964 0.0080 0.1600 -0.0658 -0.1072 -1.4015 0.0080
1.0232 2.9426 0.0120 0.2000 -0.0686 -0.1428 -1.4030 0.0120
1.1192 3.0883 0.0168 0.2400 -0.0722 -0.1782 -1.4050 0.0170
1.2157 3.2335 0.0224 0.2800 -0.0765 -0.2135 -1.4076 0.0230
1.3127 3.3783 0.0288 0.3200 -0.0807 -0.2131 -1.4108 0.0290
1.4099 3.5228 0.0360 0.3600 -0.0857 -0.2480 -1.4141 0.0360
1.5078 3.6667 0.0440 0.4000 -0.0913 -0.2827 -1.4181 0.0440
1.6064 3.8098 0.0528 0.4400 -0.0977 -0.3171 -1.4228 0.0530
1.7057 3.9522 0.0624 0.4800 -0.1047 -0.3511 -1.4283 0.0630
1.8059 4.0938 0.0728 0.5200 -0.1117 -0.3497 -1.4347 0.0730
1.9066 4.2348 0.0840 0.5600 -0.1194 -0.3831 -1.4415 0.0840
2.0082 4.3749 0.0960 0.6000 -0.1277 -0.4159 -1.4492 0.0960
2.1110 4.5139 0.1088 0.6400 -0.1366 -0.4482 -1.4581 0.1090
2.2151 4.6516 0.1224 0.6800 -0.1456 -0.4456 -1.4682 0.1220
2.3198 4.7886 0.1368 0.7200 -0.1551 -0.4769 -1.4788 0.1360
2.4258 4.9243 0.1520 0.7600 -0.1652 -0.5073 -1.4908 0.1510
2.5335 5.0584 0.1680 0.8000 -0.1760 -0.5370 -1.5043 0.1670
2.6427 5.1908 0.1848 0.8400 -0.1873 -0.5656 -1.5195 0.1840
2.7538 5.3215 0.2024 0.8800 -0.1985 -0.5604 -1.5366 0.2010
2.8658 5.4512 0.2208 0.9200 -0.2102 -0.5875 -1.5546 0.2190
2.9798 5.5790 0.2400 0.9600 -0.2225 -0.6134 -1.5747 0.2380
3.0958 5.7047 0.2600 1.0000 -0.2346 -0.6063 -1.5971 0.2570
3.2130 5.8293 0.2808 1.0400 -0.2479 -0.6616 -1.6207 0.2780
3.3333 5.9507 0.3024 1.0800 -0.2603 -0.6217 -1.6482 0.2980
3.4538 6.0719 0.3248 1.1200 -0.2738 -0.6741 -1.6759 0.3200
3.5778 6.1897 0.3480 1.1600 -0.2871 -0.6637 -1.7081 0.3420
3.7029 6.3063 0.3720 1.2000 -0.3007 -0.6825 -1.7420 0.3650
3.8305 6.4205 0.3968 1.2400 -0.3147 -0.6996 -1.7794 0.3890
3.9605 6.5323 0.4224 1.2800 -0.3284 -0.6866 -1.8207 0.4130
4.0916 6.6428 0.4488 1.3200 -0.3425 -0.7012 -1.8643 0.4380
4.2252 6.7510 0.4760 1.3600 -0.3567 -0.7139 -1.9122 0.4640
4.3612 6.8567 0.5040 1.4000 -0.3707 -0.6981 -1.9649 0.4900
4.4983 6.9614 0.5328 1.4400 -0.3849 -0.7082 -2.0207 0.5170
4.6377 7.0637 0.5624 1.4800 -0.3987 -0.6910 -2.0819 0.5440
4.7780 7.1652 0.5928 1.5200 -0.4127 -0.6984 -2.1466 0.5720
4.9205 7.2644 0.6240 1.5600 -0.4263 -0.6799 -2.2177 0.6000
5.0637 7.3629 0.6560 1.6000 -0.4404 -0.7079 -2.2929 0.6300
5.2104 7.4580 0.6888 1.6400 -0.4537 -0.6641 -2.3783 0.6590
5.3562 7.5539 0.7224 1.6800 -0.4670 -0.6662 -2.4659 0.6890
5.5039 7.6479 0.7568 1.7200 -0.4803 -0.6663 -2.5620 0.7200
5.6534 7.7401 0.7920 1.7600 -0.4932 -0.6441 -2.6676 0.7510
5.8032 7.8321 0.8280 1.8000 -0.5061 -0.6417 -2.7800 0.7830
5.9546 7.9225 0.8648 1.8400 -0.5184 -0.6184 -2.9037 0.8150
6.1061 8.0127 0.9024 1.8800 -0.5303 -0.5953 -3.0357 0.8470
6.2577 8.1028 0.9408 1.9200 -0.5421 -0.5900 -3.1767 0.8800
6.4106 8.1917 0.9800 1.9600 -0.5538 -0.5829 -3.3324 0.9140
6.5648 8.2792 1.0200 2.0000 -0.5649 -0.5578 -3.5047 0.9480
6.2143 7.2126 1.0600 2.0000 -0.5753 -0.5178 -3.6904 0.9810
6.2393 7.1876 1.1000 2.0000 -0.5855 -0.5094 -3.8848 1.0150
6.2643 7.1626 1.1400 2.0000 -0.5949 -0.4714 -4.1013 1.0480
6.2880 7.1390 1.1800 2.0000 -0.6039 -0.4491 -4.3291 1.0810
6.3110 7.1160 1.2200 2.0000 -0.6124 -0.4270 -4.5765 1.1140
6.3334 7.0936 1.2600 2.0000 -0.6205 -0.4053 -4.8462 1.1470
6.3552 7.0718 1.3000 2.0000 -0.6280 -0.3725 -5.1412 1.1790
6.3758 7.0511 1.3400 2.0000 -0.6350 -0.3525 -5.4553 1.2110
6.3960 7.0310 1.3800 2.0000 -0.6417 -0.3328 -5.8013 1.2430
6.4156 7.0113 1.4200 2.0000 -0.6480 -0.3132 -6.1845 1.2750
6.4349 6.9921 1.4600 2.0000 -0.6538 -0.2939 -6.6117 1.3070
6.4537 6.9732 1.5000 2.0000 -0.6593 -0.2747 -7.0914 1.3390
6.4722 6.9548 1.5400 2.0000 -0.6643 -0.2480 -7.6344 1.3700
6.4898 6.9372 1.5800 2.0000 -0.6690 -0.2374 -8.2345 1.4020
6.5076 6.9193 1.6200 2.0000 -0.6733 -0.2120 -8.9486 1.4330
6.5247 6.9023 1.6600 2.0000 -0.6772 -0.1944 -9.7563 1.4640
6.5415 6.8855 1.7000 2.0000 -0.6808 -0.1824 -10.7107 1.4960
6.5587 6.8683 1.7400 2.0000 -0.6840 -0.1589 -11.8990 1.5270
6.5752 6.8518 1.7800 2.0000 -0.6868 -0.1414 -13.3172 1.5580
6.5915 6.8354 1.8200 2.0000 -0.6893 -0.1240 -15.1051 1.5890
6.6078 6.8191 1.8600 2.0000 -0.6914 -0.1033 -17.4345 1.6190
6.6236 6.8034 1.9000 2.0000 -0.6932 -0.0894 -20.4836 1.6500
6.6398 6.7872 1.9400 2.0000 -0.6946 -0.0717 -25.0003 1.6810
6.6561 6.7709 1.9800 2.0000 -0.6957 -0.0539 -32.0908 1.7120
6.6724 6.7545 2.0200 2.0000 -0.6964 -0.0351 -44.8754 1.7420
6.6884 6.7386 2.0600 2.0000 -0.6967 -0.0185 -73.3482 1.7730
6.7050 6.7220 2.1000 2.0000 -0.6967 -0.0001 -216.8649 1.8040
6.7218 6.7052 2.1400 2.0000 -0.6964 0.0176 221.2198 1.8340
6.7383 6.6886 2.1800 2.0000 -0.6957 0.0367 74.1550 1.8650
6.7557 6.6713 2.2200 2.0000 -0.6945 0.0559 43.6549 1.8960
6.7734 6.6536 2.2600 2.0000 -0.6930 0.0754 30.7534 1.9270
6.7915 6.6355 2.3000 2.0000 -0.6912 0.0920 23.6162 1.9570
6.8094 6.6175 2.3400 2.0000 -0.6889 0.1150 19.1975 1.9880
6.8285 6.5985 2.3800 2.0000 -0.6862 0.1359 16.0162 2.0190
```

6.8481 6.5788 2.4200 2.0000 -0.6830 0.1572 13.6803 2.0500
6.8684 6.5585 2.4600 2.0000 -0.6794 0.1791 11.8892 2.0810
6.8894 6.5375 2.5000 2.0000 -0.6754 0.2017 10.4699 2.1120
6.9112 6.5157 2.5400 2.0000 -0.6708 0.2326 9.3158 2.1440
6.9346 6.4923 2.5800 2.0000 -0.6658 0.2497 8.3294 2.1750
6.9583 6.4687 2.6200 2.0000 -0.6603 0.2745 7.5243 2.2060
6.9830 6.4439 2.6600 2.0000 -0.6541 0.3102 6.8342 2.2380
7.0097 6.4172 2.7000 2.0000 -0.6473 0.3386 6.2177 2.2700
7.0378 6.3891 2.7400 2.0000 -0.6399 0.3681 5.6795 2.3020
7.0673 6.3596 2.7800 2.0000 -0.6320 0.3987 5.2055 2.3340
7.0985 6.3285 2.8200 2.0000 -0.6234 0.4306 4.7846 2.3660
7.1313 6.2956 2.8600 2.0000 -0.6138 0.4789 4.4083 2.3990
7.1672 6.2598 2.9000 2.0000 -0.6035 0.5159 4.0598 2.4320
7.2052 6.2217 2.9400 2.0000 -0.5924 0.5546 3.7459 2.4650
7.2456 6.1813 2.9800 2.0000 -0.5805 0.5951 3.4616 2.4980
7.2885 6.1384 3.0200 2.0000 -0.5673 0.6575 3.2034 2.5320
7.3356 6.0914 3.0600 2.0000 -0.5532 0.7048 2.9611 2.5660
7.3857 6.0412 3.1000 2.0000 -0.5381 0.7544 2.7401 2.6000
7.4392 5.9878 3.1400 2.0000 -0.5220 0.8065 2.5382 2.6340
7.4962 5.9307 3.1800 2.0000 -0.5043 0.8873 2.3534 2.6690
7.5588 5.8682 3.2200 2.0000 -0.4853 0.9482 2.1791 2.7040
7.6255 5.8014 3.2600 2.0000 -0.4645 1.0419 2.0197 2.7400
7.6987 5.7283 3.3000 2.0000 -0.4422 1.1128 1.8697 2.7760
7.7766 5.6504 3.3400 2.0000 -0.4185 1.1871 1.7327 2.8120
7.8593 5.5677 3.3800 2.0000 -0.3924 1.3009 1.6077 2.8490
7.9491 5.4778 3.4200 2.0000 -0.3647 1.3864 1.4907 2.8860
8.0438 5.3831 3.4600 2.0000 -0.3344 1.5161 1.3846 2.9240
8.1457 5.2812 3.5000 2.0000 -0.3021 1.6127 1.2861 2.9620
8.2517 5.1753 3.5400 2.0000 -0.2679 1.7117 1.1975 3.0000
8.3607 5.0663 3.5800 2.0000 -0.2307 1.8615 1.1183 3.0390
8.4744 4.9525 3.6200 2.0000 -0.1913 1.9682 1.0461 3.0780
8.5882 4.8388 3.6600 2.0000 -0.1498 2.0741 0.9826 3.1170
8.6997 4.7273 3.7000 2.0000 -0.1051 2.2347 0.9274 3.1570
8.8089 4.6180 3.7400 2.0000 -0.0584 2.3387 0.8791 3.1970
8.9096 4.5173 3.7800 2.0000 -0.0096 2.4355 0.8388 3.2370
8.9982 4.4287 3.8200 2.0000 0.0408 2.5224 0.8062 3.2770
9.0712 4.3558 3.8600 2.0000 0.0927 2.5971 0.7813 3.3170
9.1251 4.3019 3.9000 2.0000 0.1445 2.5900 0.7638 3.3560
9.1565 4.2704 3.9400 2.0000 0.1985 2.6994 0.7540 3.3960
9.1652 4.2617 3.9800 2.0000 0.2517 2.6567 0.7513 3.4350
9.1498 4.2772 4.0200 2.0000 0.3036 2.5951 0.7561 3.4730
9.1122 4.3148 4.0600 2.0000 0.3553 2.5849 0.7679 3.5110
9.0536 4.3734 4.1000 2.0000 0.4051 2.4919 0.7872 3.5480
8.9782 4.4487 4.1400 2.0000 0.4542 2.4533 0.8134 3.5850
8.8875 4.5395 4.1800 2.0000 0.4996 2.2738 0.8473 3.6200
8.7902 4.6368 4.2200 2.0000 0.5440 2.2187 0.8870 3.6550
8.6845 4.7425 4.2600 2.0000 0.5859 2.0951 0.9346 3.6890
8.5763 4.8506 4.3000 2.0000 0.6253 1.9711 0.9888 3.7220
8.4684 4.9585 4.3400 2.0000 0.6623 1.8487 1.0496 3.7540
8.3629 5.0640 4.3800 2.0000 0.6969 1.7294 1.1168 3.7850
8.2614 5.1656 4.4200 2.0000 0.7292 1.6144 1.1900 3.8150
8.1649 5.2621 4.4600 2.0000 0.7603 1.5553 1.2691 3.8450
8.0710 5.3559 4.5000 2.0000 0.7883 1.3980 1.3569 3.8730
7.9864 5.4405 4.5400 2.0000 0.8152 1.3464 1.4471 3.9010
7.9051 5.5218 4.5800 2.0000 0.8402 1.2500 1.5458 3.9280
7.8302 5.5968 4.6200 2.0000 0.8643 1.2032 1.6495 3.9550

7.7588 5.6681 4.6600 2.0000 0.8857 1.0731 1.7621 3.9800
7.6960 5.7309 4.7000 2.0000 0.9064 1.0347 1.8748 4.0050
7.6364 5.7906 4.7400 2.0000 0.9264 0.9971 1.9959 4.0300
6.1865 5.5817 4.7792 1.9600 0.9441 0.8851 2.1260 4.0530
6.0321 5.4943 4.8176 1.9200 0.9604 0.8185 2.2540 4.0750
5.8811 5.4036 4.8552 1.8800 0.9763 0.7917 2.3842 4.0970
5.7325 5.3104 4.8920 1.8400 0.9902 0.6972 2.5223 4.1170
5.5872 5.2140 4.9280 1.8000 1.0037 0.6763 2.6551 4.1370
5.4438 5.1156 4.9632 1.7600 1.0156 0.5913 2.7951 4.1550
5.3030 5.0147 4.9976 1.7200 1.0271 0.5753 2.9273 4.1730
5.1636 4.9124 5.0312 1.6800 1.0383 0.5597 3.0658 4.1910
5.0256 4.8087 5.0640 1.6400 1.0480 0.4848 3.2108 4.2070
4.8893 4.7032 5.0960 1.6000 1.0574 0.4731 3.3452 4.2230
4.7542 4.5966 5.1272 1.5600 1.0661 0.4332 3.4850 4.2380
4.6201 4.4889 5.1576 1.5200 1.0745 0.4235 3.6211 4.2530
4.4870 4.3803 5.1872 1.4800 1.0823 0.3867 3.7623 4.2670
4.3548 4.2707 5.2160 1.4400 1.0893 0.3518 3.8987 4.2800
4.2235 4.1603 5.2440 1.4000 1.0962 0.3449 4.0295 4.2930
4.0927 4.0493 5.2712 1.3600 1.1030 0.3383 4.1643 4.3060
3.9626 3.9377 5.2976 1.3200 1.1086 0.2811 4.3034 4.3170
3.8331 3.8255 5.3232 1.2800 1.1146 0.3014 4.4244 4.3290
3.7040 3.7129 5.3480 1.2400 1.1201 0.2716 4.5599 4.3400
3.5754 3.5997 5.3720 1.2000 1.1249 0.2430 4.6874 4.3500
3.4471 3.4862 5.3952 1.1600 1.1297 0.2395 4.8062 4.3600
3.3193 3.3723 5.4176 1.1200 1.1344 0.2359 4.9277 4.3700
3.1918 3.2581 5.4392 1.0800 1.1386 0.2094 5.0519 4.3790
3.0645 3.1436 5.4600 1.0400 1.1423 0.1838 5.1661 4.3870
2.9374 3.0290 5.4800 1.0000 1.1464 0.2042 5.2696 4.3960
2.8107 2.9140 5.4992 0.9600 1.1500 0.1792 5.3883 4.4040
2.6842 2.7988 5.5176 0.9200 1.1531 0.1551 5.4957 4.4110
2.5577 2.6835 5.5352 0.8800 1.1561 0.1535 5.5913 4.4180
2.4314 2.5680 5.5520 0.8400 1.1592 0.1519 5.6883 4.4250
2.3053 2.4524 5.5680 0.8000 1.1618 0.1290 5.7869 4.4310
2.1793 2.3367 5.5832 0.7600 1.1643 0.1279 5.8726 4.4370
2.0534 2.2209 5.5976 0.7200 1.1668 0.1268 5.9593 4.4430
1.9276 2.1049 5.6112 0.6800 1.1689 0.1048 6.0473 4.4480
1.8018 1.9890 5.6240 0.6400 1.1710 0.1040 6.1214 4.4530
1.6761 1.8729 5.6360 0.6000 1.1731 0.1033 6.1964 4.4580
1.5506 1.7567 5.6472 0.5600 1.1747 0.0821 6.2721 4.4620
1.4249 1.6407 5.6576 0.5200 1.1764 0.0816 6.3333 4.4660
1.2993 1.5245 5.6672 0.4800 1.1780 0.0811 6.3950 4.4700
1.1738 1.4083 5.6760 0.4400 1.1792 0.0606 6.4572 4.4730
1.0481 1.2922 5.6840 0.4000 1.1804 0.0603 6.5043 4.4760
0.9225 1.1761 5.6912 0.3600 1.1816 0.0600 6.5516 4.4790
0.7970 1.0599 5.6976 0.3200 1.1828 0.0598 6.5992 4.4820
0.6715 0.9436 5.7032 0.2800 1.1836 0.0397 6.6472 4.4840
0.5458 0.8276 5.7080 0.2400 1.1844 0.0396 6.6793 4.4860
0.4201 0.7115 5.7120 0.2000 1.1848 0.0198 6.7115 4.4870
0.2941 0.5958 5.7152 0.1600 1.1852 0.0197 6.7277 4.4880
0.1681 0.4800 5.7176 0.1200 1.1856 0.0197 6.7439 4.4890
0.0422 0.3642 5.7192 0.0800 1.1860 0.0197 6.7602 4.4900
-0.0837 0.2484 5.7200 0.0400 1.1860 0.0000 6.7765 4.4900
-1.5501 -1.2070 5.7200 -0.0000 1.1860 0.0000 6.7765 4.4900