ON THE NUMERICAL SOLUTION OF THE CYLINDRICAL POISSON EQUATION FOR ISOLATED SELF-GRAVITATING SYSTEMS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Physics and Astronomy

by
Howard S. Cohl
B.S., Indiana University, Bloomington, 1990
M.S., Louisiana State University and A&M College, Baton Rouge, 1994
August, 1999

Acknowledgments

Thanks to Joel for his generosity, his concern, his wisdom, his writing skills, and his direction. Thanks to Dick Durisen for sharing his vision with me. Overwhelming thanks to Mom and Dad, for without their generosity and love, last second efforts would have been impossible or at minimum, a potsticker. In the words of Charles Bukowski, "Another drink! For all my friends!..." and then again and then again and then again and thanks to my roommate Nova for being so great. Thanks to Ali and Steve and Donnie and Ronnie and Mike and John and Dagger and Jody. Thanks to Naomi, Cindy, Ralph and Grover. Thanks to the Thirsty Tiger Tavern, the capitol grocery crew and all my Spanish Town, downtown, and garden district friends. Thanks to Steve and Don and Eddie and Bala and Kevin and Gianna and Bob and Bill and Jim. Thanks to Linda and Lillian. Thanks to USENET. Thanks to Tony. Thanks for sunshine, Bessel functions, Unix and IDL. I am grateful to Dana Browne, Ganesh Chanmugam, A.R.P. Rau, Juhan Frank, and to the entire Louisiana State University department of Physics and Astronomy. Thanks to the Indiana University Astronomy Department. Thanks to my office mate John Cazes and thanks to Eric Barnes and Patrick Motl. Thanks to John Woodward, Paul Fisher, Kim Barker, Dimitris Christodoulou and Saied Andalib. I would like to thank Sandeep Dani for his invaluable assistance in the development of the MasPar MPF swap_pm jacket. I acknowledge the Scalable Computing Lab at Iowa State University for the use of their MasPar MP-2 and support that has been

received from the Louisiana State Board of Regents through the Louisiana Education Quality Support Fund for the establishment of the Concurrent Computing Laboratory on whose facilities this research was partially conducted. I acknowledge support from the U.S. National Science Foundation through grants AST-9528424 and DGE-9355007, the latter of which has been issued through the NSF's Graduate Traineeships Program. This work also has been supported, in part, by grants of high-performance-computing time on NPACI facilities at SDSC and UT, Austin, and through the PET program of NAVOCEANO DoD Major Shared Resource Center in Stennis, MS.

Table of Contents

List of Tables

List of Figures

Abstract

This dissertation addresses the need for an accurate and efficient technique which solves the Poisson equation for arbitrarily complex, isolated, self-gravitating fluid systems. Generally speaking, a potential solver is composed of two distinct pieces: a boundary solver and an interior solver. The boundary solver computes the potential, $\Phi(\mathbf{x_B})$ on a surface which bounds some finite volume of space, V, and contains an isolated mass-density distribution, $\rho(\mathbf{x})$. Given $\rho(\mathbf{x})$ and $\Phi(\mathbf{x_B})$, the interior solver computes the potential $\Phi(\mathbf{x})$ everywhere within V. Herein, we describe the development of a numerical technique which efficiently solves Poisson's equation in cylindrical coordinates on massively parallel computing architectures.

First, we report the discovery of a compact cylindrical Green's function (CCGF) expansion and show how the CCGF can be used to efficiently compute the exact numerical representation of $\Phi(\mathbf{x_B})$. As an analytical representation, the CCGF should prove to be extremely useful wherever one requires the isolated azimuthal modes of a self-gravitating system.

We then discuss some mathematical consequences of the CCGF expansion, such as it's applicability to all nine axisymmetric coordinate systems which are $\mathcal{R}$-separable for Laplace's equation. The CCGF expansion, as applied to the spherical coordinate system, leads to a *second* addition theorem for spherical harmonics.

Finally, we present a massively parallel implementation of an interior solver which is based on a data-transpose technique applied to a Fourier-

ADI (Alternating Direction Implicit) scheme. The data-transpose technique is a parallelization strategy in which all communication is restricted to global 3D data-transposition operations and all computations are subsequently performed with perfect load balance and zero communication.

The potential solver, as implemented here in conjunction with the CCGF expansion, should prove to be an extremely useful tool in a wide variety of astrophysical studies, particularly those requiring an accurate determination of the gravitational field due to extremely flattened or highly elongated mass distributions.

1. Introduction

A great many astrophysical problems require the determination of a gravitational field. The field, for the most part, can be adequately described by Newtonian gravity and often can be derived from a potential function. From a mathematical viewpoint there are two methods for obtaining the potential: by solving a partial differential equation, *i.e.* Poisson's equation; or by solving an integral equation, *i.e.* employing the Green's function method (Jackson 1975). As Arfken (1985) has explained, boundary conditions are directly built into the integral equation rather than being imposed at the final stage of the solution of a partial differential equation. Also, mathematical problems such as existence and uniqueness can be easier to handle when cast in integral form. On the other hand, solving differential equations is often more tractable than solving integral equations, particularly when dealing with multidimensional problems.

In building realistic models of steady-state galaxies, a considerable amount of effort has been devoted in recent years toward identifying analytically prescribable potential–density pairs. In some cases a reasonable three dimensional density distribution can be represented by a sum over a finite set of "basis density functions" in which case Poisson's equation can be solved using the corresponding basis sets of the potential-density pairs (Earn 1996; Robijn & Earn 1996). Some useful steady-state models also can be constructed by superposing other special density (or surface-density) distributions with

known potentials, such as those derivable from Stäckel potentials (de Zeeuw 1985; Evans & de Zeeuw 1992).

When following the time-evolutionary behavior of models whose structures are changing on a dynamical timescale, however, one must develop an efficient technique for solving Poisson's equation that works for arbitrary mass distributions. Furthermore, simulations of time-evolving systems often are carried out on grids that cover a finite (rather than an infinite) region of space, in which case one must also determine the potential on the boundary of that region. In practice, then, in many astrophysical studies a Green's function method is used to find the potential only on a boundary *outside* of a mass distribution, then a technique is developed to solve Poisson's equation to obtain the interior solution. A standard technique for calculating the boundary potential has been to expand the Green's function in spherical coordinates, resulting in what is often referred to as a "multipole method" (Black & Bodenheimer 1975; Norman & Wilson 1978; Barnes & Hut 1986; see also §2.1.1, below) in which the potential is grouped into an infinite sum over a basis set of spherical harmonics described by two quantum numbers — one meridional, l, and the other azimuthal, m.

Because very flattened mass distributions are poorly described in a spherical coordinate system, we have examined whether it might be advantageous in numerical simulations to cast the Green's function in a cylindrical coordinate system. The "familiar" expression for the cylindrical Green's function expansion can be found in variety of references (cf., Morse & Feshbach 1953; Jackson 1975; Arfken 1985). It is expressible in terms of an infinite sum over

the azimuthal quantum number m and an infinite integral over products of Bessel functions of various orders multiplied by an exponential function (see eq. [2.13], below). We note a previous attempt by Villumsen (1985) to solve the potential problem in this manner; he presents a technique where each infinite integral over products of Bessel functions is evaluated numerically using a Gauss-Legendre integrator. In that paper Villumsen states, "Cylindrical coordinates are a more natural coordinate system for disk systems." He then emphasizes the obvious problem that, due to the infinite integrals involved, a calculation of the potential via this straightforward application of the familiar cylindrical Green's function expansion is numerically much more difficult than a calculation of the potential using a spherical Green's function expansion.

In chapter 2 of this dissertation, we derive an extraordinarily compact expression for the Green's function in cylindrical coordinates. Our expression (see eq. [2.15], below) completely removes the need for a numerical evaluation of the infinite integrals involved since we have found an analytical expression for the integral in terms of half-integer degree Legendre functions of the second kind. As we discuss in subsequent sections of chapter 2, our technique should prove to be a particularly powerful tool for studying self-gravitating systems that conform well to a cylindrical coordinate mesh, such as highly flattened (disk systems) or highly elongated (jet or bipolar flow) mass distributions. As far as we have been able to ascertain, this result has not been previously derived. At the very least, based on published re-

search over the past thirty years, the result appears to be unfamiliar to the astrophysics community.

In chapter 3 of this dissertation, we demonstrate how the CCGF can be extended to all nine axisymmetric coordinate systems which are $\mathcal{R}-$ separable for Laplace's equation. The first coordinate system we address in chapter 3 is the spherical coordinate system, where the result is particularly interesting and ends up leading to a *second* addition theorem for spherical harmonics. The standard addition theorem for spherical harmonics demonstrates how one might collapse the summation over all m terms into a single special function expression, whereas the *second* addition theorem shows how one may now collapse the summation over all l terms in the Green's function. In this representation, one is capable of isolating each and every azimuthal mode in the spherical Green's function. We prove the new addition theorem's exactness in one limiting case. Furthermore, we show how this result can be extended to the rest of the axisymmetric Green's functions and how in future investigations this result is likely to lead to a better general understanding of how gravity represents itself in axisymmetric coordinate systems.

In chapter 4 we describe our numerical implementation of an efficient scheme to solve Poisson's equation numerically on massively parallel architectures. The groundwork on serial algorithms for solving Poisson's equation is extensive. In particular, for some time, extremely efficient methods have been known for solving the set of sparse matrices that result from a second-order accurate finite-differencing of the Poisson equation in cylindrical coordinates given the boundary solution. In Cartesian coordinates there has been a large

successful effort in order to find accurate and highly parallel methods for solving Poisson's equation (i.e. Fast Poisson solver using Fourier methods). The situation is not so simple in cylindrical coordinates. Due to the non-constant variation of the matrix elements that result from the finite-discretization of the cylindrical Poisson equation, direct Fourier methods are not possible. It is only in the naturally periodic azimuthal coordinate direction, where one can take advantage of this technique which reduces the complexity of the problem, in terms of coupled dimensions, from three-dimensions to two-dimensions. Techniques like Buneman cyclic reduction can obtain the direct solution of the resulting two-dimensional problems in an extremely accurate fashion, other direct techniques aren't even so efficient when implemented in serial. When one asks the question of how to solve these problems in parallel one quickly sees that the global nature of the two-dimensional solution methods are very difficult to implement in parallel and do not result in a load-balanced solution of the matrix problem. It is here that we present the Fourier-ADI method, which is iterative, although very accurate, and takes advantage of the highly parallel data-transpose technique. In this computational strategy all computations are performed without communication, and all communications are restricted to highly parallel, global three-dimensional data-transpositions. We describe in detail how this algorithm is implemented and give a theoretical operation count which demonstrates the highly parallel nature of this algorithm. It is the Fourier-ADI technique, combined with the CCGF technique for evaluating the boundary potential that yields an extremely efficient and accurate potential solver.

It is important to recognize that the focus of this dissertation is not on obtaining a detailed solution to one particular astrophysical problem. Instead, by developing an accurate and efficient technique for solving the Poisson equation for arbitrarily complex mass distributions, we are laying the groundwork necessary to support future advances in a large number of subfields of astrophysics. Examples of studies that are certain to benefit from the developments presented here are: the fragmentation of molecular cloud cores in order to study star formation processes (Boss 1993; Boss 1998a; Truelove et al. 1997); the formation of giant gaseous protoplanets (Boss 1998b); the dynamical bar-mode instability that arises in rapidly rotating gas clouds (Cazes 1999; Toman et. al. 1998, Pickett, Durisen & Davis 1996); protostellar disks (Pickett et. al. 1998); nonexplosive contraction of the cores of massive stars (Hayashi, Eriguchi, & Hashimoto 1998) and estimates of the gravitational radiation that should be emitted from such configurations (Yoshida & Eriguchi 1995); steady-state structures of triaxial galaxies (Earn 1996; Robijn & Earn 1996); self-consistent field techniques (Hachisu 1986); mass transfer in close binary systems (Motl, Frank, and Tohline 1999) and the ultimate merger of such systems (New and Tohline 1997); and binary star formation (Cazes 1999).

2. A Compact Cylindrical Green's Function Expansion

2.1 A Comparison of Potential Evaluating Techniques

In general, the integral solution to the potential problem may be written in terms of the Green's function $\mathcal{G}(\mathbf{x}, \mathbf{x}')$ as follows (cf., eq. [1.42] of Jackson 1975):

$$\begin{aligned} \Phi(\mathbf{x}) &= -G \int_V \rho(\mathbf{x}') \mathcal{G}(\mathbf{x}, \mathbf{x}') d^3x' \\ &+ \frac{G}{4\pi} \oint_S \left[\Phi(\mathbf{x}') \frac{\partial \mathcal{G}(\mathbf{x}, \mathbf{x}')}{\partial n'} - \mathcal{G}(\mathbf{x}, \mathbf{x}') \frac{\partial \Phi}{\partial n'} \right] da', \end{aligned} \tag{2.1}$$

where Φ is the potential, G is the gravitational constant, ρ is the mass density, $\mathbf{x}$ denotes the position vector from the origin to the point at which the potential is being evaluated, $\mathbf{x}'$ denotes the position vector over which the mass integration is performed, V is the volume over which $\mathbf{x}'$ is integrated, and S is the bounding surface of V. For the case of no bounding surfaces — as in most astrophysical systems — the surface integral in eq. (2.1) vanishes due to the requirement that both Φ and the derivative of Φ normal to the surface $\partial\Phi/\partial n'$ vanish at infinity. In this case the Green's function reduces to

$$\mathcal{G}(\mathbf{x}, \mathbf{x}') = \frac{1}{|\mathbf{x} - \mathbf{x}'|}. \tag{2.2}$$

These requirements therefore reduce eq. (2.1) to the more often quoted integral expression for the gravitational potential, namely

$$\Phi(\mathbf{x}) = -G\int_V \rho(\mathbf{x}')\mathcal{G}(\mathbf{x},\mathbf{x}')d^3x' = -G\int_V \frac{\rho(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|}d^3x'. \tag{2.3}$$

2.1.1 The Multipole Method

In spherical coordinates, the expansion of the Green's function is (cf., eq. [3.70] of Jackson 1975)

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = 4\pi\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{1}{2l+1}\frac{r_<^l}{r_>^{l+1}}Y_{lm}^*(\theta',\phi')Y_{lm}(\theta,\phi), \tag{2.4}$$

where r represents the radial distance from the origin, θ is the polar angle, ϕ is the azimuthal angle, and Y_{lm} is the spherical harmonic function. (For a complete specification of the spherical harmonic function, see eq. [3.13] and the discussion associated with it.) If we insert eq. (2.4) into eq. (2.3), we obtain an expression for the potential at an exterior point ($r > r'$),

$$\Phi_{ext}(\mathbf{x}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{4\pi}{2l+1}\frac{Y_{lm}(\theta,\phi)}{r^{l+1}}q_{lm}^<, \tag{2.5}$$

where the coefficients

$$q_{lm}^< \equiv \int_V Y_{lm}^*(\theta',\phi')r'^l\rho(\mathbf{x}')d^3x', \tag{2.6}$$

are called *multipole moments*. In the case of an axisymmetric configuration, only the $m = 0$ terms in expression (2.4) survive, reducing it to

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|}\Big|_{m=0} = \sum_{l=0}^{\infty}\frac{r_<^l}{r_>^{l+1}}P_l(\cos\theta')P_l(\cos\theta). \tag{2.7}$$

The corresponding expression for the axisymmetric potential is therefore given by,

$$\Phi_{ext}(r,\theta)\Big|_{m=0} = -G\sum_{l=0}^{\infty} P_l(\cos\theta) r^{-(l+1)} M_l, \tag{2.8}$$

where now the *axisymmetric* multipole moments,

$$M_l \equiv \int_V \rho(r',\theta') r'^l P_l(\cos\theta') d^3x'. \tag{2.9}$$

Expressions (2.5) or (2.8) for the gravitational potential have been adopted by many groups when developing numerical techniques to follow self-gravitating fluid flows on spherical or cylindrical coordinate meshes (Black & Bodenheimer 1975; Norman & Wilson 1978; Boss 1980; Tohline 1980; Stone & Norman 1992; Boss & Myhill 1995; Müller & Steinmetz 1995; Yorke & Kaisig 1995).

As mentioned earlier, usually this multipole technique has been used to determine the potential everywhere along the bounding surface of the computational grid, then a separate technique has been developed to solve the Poisson equation (see chapter 4, eq. [4.1] and the relevant discussion given therein) in order to obtain the gravitational potential throughout the volume of the grid. But when utilizing this multipole method an exact determination of Φ for a discrete mass distribution is not possible because of the required infinite sum over the quantum number l. Instead, a decision must be made regarding when the series should be truncated in order to achieve a desired degree of accuracy for a given $\rho(\mathbf{x}')$ distribution. For example, referring to an expression for the axisymmetric potential analogous to our eq. (2.8), Stone & Norman (1992) state that, "As implemented in ZEUS-2D, we continue

to add higher moments until Φ_B has converged to one part in 10^3, up to a maximum of 100 terms."

One must also be sure that every location on the boundary of the computational grid $\mathbf{x}_B$ at which the exterior potential is being evaluated is at a radial location r_B that is greater than *all* interior grid locations at which matter resides. Otherwise $\Phi(\mathbf{x}_B)$ must be evaluated in two parts, namely,

$$\Phi(\mathbf{x}_B) = \Phi_{ext}(\mathbf{x}_B) + \Phi_{int}(\mathbf{x}_B), \tag{2.10}$$

where $\Phi_{int}(\mathbf{x}_B)$ must be determined through a separate integration over the mass that lies at radial locations greater than r_B. Specifically, the potential at an interior point ($r < r'$),

$$\Phi_{int}(\mathbf{x}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \frac{4\pi}{2l+1} r^l Y_{lm}(\theta, \phi) q_{lm}^{>}, \tag{2.11}$$

where the coefficients

$$q_{lm}^{>} = \int_V \frac{Y_{lm}^*(\theta', \phi')}{r'^{l+1}} \rho(\mathbf{x}') d^3x'. \tag{2.12}$$

As we illustrate more fully in §2.2.2, below, unless the boundary of a cylindrical grid is carefully designed so that it lies entirely outside the interior mass distribution (usually this means placing the grid boundary far away from the surface of the mass distribution), it will become necessary to calculate a separate set of "interior" and "exterior" moments of the mass distribution for the majority of boundary locations. This requirement will make the multipole method very computationally demanding, unless accuracy is sac-

rificed through a reduction in the number of terms that are included in the l summation.

2.1.1.1 General Expressions

In terms of the cylindrical coordinates (R, ϕ, z) the Green's function may be written as (cf., problem [3.14] of Jackson 1975),

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')} \int_0^{\infty} dk\ J_m(kR)J_m(kR')\ e^{-k(z_>-z_<)}, \qquad (2.13)$$

where J_m is an order m Bessel function of the first kind. Especially when faced with the problem of determining the gravitational potential on a cylindrical coordinate mesh, it would seem that this is a more appropriate expression to use for the Green's function than eq. (2.4). As we discussed in the introduction, however, devising an efficient numerical technique by which to accurately evaluate the infinite integral over products of Bessel functions has proven to be a difficult task.

Using eq. (13.22.2) in Watson (1944) we recently have realized that,

$$\int_0^{\infty} e^{-at} J_m(bt)J_m(ct)dt = \frac{1}{\pi\sqrt{bc}}\ Q_{m-\frac{1}{2}}\left(\frac{a^2+b^2+c^2}{2bc}\right), \qquad (2.14)$$

where $Q_{m-\frac{1}{2}}$ is the half-integer degree Legendre function of the second kind. Hence, it becomes possible to rewrite eq. (2.13) as,

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \frac{1}{\pi\sqrt{RR'}} \sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')}\ Q_{m-\frac{1}{2}}(\chi), \qquad (2.15)$$

with

$$\chi \equiv \frac{R^2 + R'^2 + (z - z')^2}{2RR'}. \tag{2.16}$$

We note that this same result for the Green's function can be obtained by combining eq. (3.148) in Jackson (1975) with eq. (6.672.4) in Gradshteyn & Ryzhik (1994). Although relationship (2.14) and, hence, the ability to derive (2.15) from (2.13), has been known for some time, apparently the astrophysics community has not been aware that the cylindrical Green's function can be expressed in this extraordinarily compact form. As we shall demonstrate, highly accurate and efficient means of evaluating $\Phi(\mathbf{x})$ can be developed from expression (2.15).

Realizing that $Q_{-\frac{1}{2}+m}(\chi) = Q_{-\frac{1}{2}-m}(\chi)$ (cf., eq. [8.736.7] in Gradshteyn & Ryzhik 1994), and that $e^{i\theta} + e^{-i\theta} = 2\cos\theta$, we can express eq. (2.15) in terms of all $m \geq 0$ as

$$\frac{1}{|\mathbf{x} - \mathbf{x}'|} = \frac{1}{\pi\sqrt{RR'}} \sum_{m=0}^{\infty} \epsilon_m \cos[m(\phi - \phi')]\, Q_{m-\frac{1}{2}}(\chi), \tag{2.17}$$

where ϵ_m is the Neumann factor (Morse & Feshbach 1953), that is $\epsilon_0 = 1$ and $\epsilon_m = 2$ for $m \geq 1$. Now we substitute eq. (2.17) into eq. (2.3) obtaining

$$\Phi(\mathbf{x}) = -\frac{G}{\pi\sqrt{R}} \int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'}} \sum_{m=0}^{\infty} \epsilon_m \cos[m(\phi - \phi')]\, Q_{m-\frac{1}{2}}(\chi), \tag{2.18}$$

which may also be rewritten as,

$$\Phi(\mathbf{x}) \;=\; -\frac{G}{\pi\sqrt{R}} \int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'}}\, Q_{-\frac{1}{2}}(\chi)$$

$$- \frac{2G}{\pi\sqrt{R}} \sum_{m=1}^{\infty} \cos(m\phi) \int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'}} \cos(m\phi') \; Q_{m-\frac{1}{2}}(\chi) \quad (2.19)$$
$$- \frac{2G}{\pi\sqrt{R}} \sum_{m=1}^{\infty} \sin(m\phi) \int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'}} \sin(m\phi') \; Q_{m-\frac{1}{2}}(\chi).$$

Finally, an azimuthal discrete Fourier transform of this last expression yields the following elegant representation of the gravitational potential in Fourier space:

$$\Phi_m^{1,2}(R,z) = -\frac{G\epsilon_m}{\sqrt{R}} \int_\Sigma d\sigma' \sqrt{R'} \; \rho_m^{1,2}(R',z') \; Q_{m-\frac{1}{2}}(\chi), \quad (2.20)$$

where Σ refers to the area over which the meridional integration is to be carried, $d\sigma' = dR'dz'$, and the Fourier components of Φ and ρ are defined such that,

$$\left\{ \begin{array}{c} \Phi \\ \rho \end{array} \right\} (\mathbf{x}) = \sum_{m=0}^{\infty} \cos(m\phi) \left\{ \begin{array}{c} \Phi_m^1 \\ \rho_m^1 \end{array} \right\} (R,z) + \sum_{m=0}^{\infty} \sin(m\phi) \left\{ \begin{array}{c} \Phi_m^2 \\ \rho_m^2 \end{array} \right\} (R,z). \quad (2.21)$$

(Note that $\Phi_0^2 = \rho_0^2 = 0$.)

2.1.1.2 Functional Forms of $Q_{m-\frac{1}{2}}$

Useful expressions for $Q_{-\frac{1}{2}}(\chi)$ and $Q_{\frac{1}{2}}(\chi)$ may be obtained from eqs. (8.13.3) and (8.13.7), respectively, of Abramowitz & Stegun (1965), namely,

$$Q_{-\frac{1}{2}}(\chi) = \mu K(\mu), \quad (2.22)$$

and

$$Q_{\frac{1}{2}}(\chi) = \chi\mu K(\mu) - (1+\chi)\mu E(\mu), \quad (2.23)$$

where K represents the Complete Elliptic Integral of the First Kind, E is the Complete Elliptic Integral of the Second Kind, and

$$\mu \equiv \sqrt{\frac{2}{1+\chi}} = \sqrt{\frac{4RR'}{(R+R')^2+(z-z')^2}}. \tag{2.24}$$

One can then obtain the higher degree half-integer Legendre functions of the second kind using the recurrence relation (cf., eq. [8.5.3] in Abramowitz & Stegun 1965)

$$Q_{m-\frac{1}{2}}(\chi) = 4\frac{m-1}{2m-1}\chi Q_{m-\frac{3}{2}}(\chi) - \frac{2m-3}{2m-1}\, Q_{m-\frac{5}{2}}(\chi). \tag{2.25}$$

For example, substituting eqs. (2.22) and (2.23) into eq. (2.25) gives the following useful expression for $Q_{\frac{3}{2}}(\chi)$:

$$Q_{\frac{3}{2}}(\chi) = (\frac{4}{3}\chi^2 - \frac{1}{3})\mu K(\mu) - \frac{4}{3}\chi(1+\chi)\mu E(\mu). \tag{2.26}$$

According to Table XIII in Tables of Associated Legendre Functions (United States. National Bureau of Standards. Computation Laboratory 1945), we may also express $Q_{m-\frac{1}{2}}(\chi)$ in terms of Gauss's Hypergeometric function as follows:

$$Q_{m-\frac{1}{2}}(\chi) = \frac{\sqrt{\pi}\ \Gamma(m+\frac{1}{2})}{2^{m+\frac{1}{2}}\ \Gamma(m+1)\ \chi^{m+\frac{1}{2}}}\ {}_2F_1\Big(\frac{2m+3}{4}, \frac{2m+1}{4}; m+1; \frac{1}{\chi^2}\Big), \tag{2.27}$$

where the specific Hypergeometric function

$${}_2F_1(a,b;c;y) = \frac{\Gamma(c)}{\Gamma(a)\ \Gamma(b)} \sum_{n=0}^{\infty} \frac{\Gamma(a+n)\ \Gamma(b+n)\ y^n}{\Gamma(c+n)\ n!}, \tag{2.28}$$

and Γ is the Gamma function (see eq. [6.1.1] of Abramowitz & Stegun 1965). Inserting eq. (2.28) into eq. (2.27), we derive the following expression:

$$Q_{m-\frac{1}{2}}(\chi) = \frac{\Gamma(m+\frac{1}{2})\ \sqrt{\pi}}{\Gamma(\frac{2m+3}{4})\ \Gamma(\frac{2m+1}{4})\ (2\chi)^{m+\frac{1}{2}}} \sum_{n=0}^{\infty} \frac{\Gamma(\frac{2m+4n+3}{4})\ \Gamma(\frac{2m+4n+1}{4})}{\Gamma(m+1+n)\ \Gamma(1+n)\ \chi^{2n}}. \tag{2.29}$$

It is well known (see Abramowitz & Stegun 1965), that Legendre functions of the second kind are singular when their arguments are unity. Evaluating the limit of $Q_{m-\frac{1}{2}}(\chi)$ in eq. (2.29) for large values of χ gives the asymptotic behavior of $Q_{m-\frac{1}{2}}(\chi)$ (with only the $n = 0$ term in the sum surviving),

$$\lim_{\chi\to\infty} Q_{m-\frac{1}{2}}(\chi) = \frac{\Gamma(m+\frac{1}{2})\ \sqrt{\pi}}{\Gamma(m+1)\ (2\chi)^{m+\frac{1}{2}}}, \tag{2.30}$$

which decays as $1/\chi^{m+\frac{1}{2}}$.

2.2 Substantiations

In this section, we verify the correctness and highlight the utility of the compact cylindrical Green's function (hereafter, CCGF) representation by comparing expressions for the Newtonian potential derived from it with previously known results. We show that the familiar expression for the potential of an infinitesimally thin, axisymmetric disk in terms of complete elliptic integrals can be readily derived from eq. (2.19). We also show how this expression can be generalized to axisymmetric systems of arbitrary vertical thickness and how an analogous expression for any other isolated azimuthal Fourier mode can now be readily derived. In the context of nonaxisymmetric fields, we show how Kalnajs' reduced potential for an infinitesimally thin, nonaxisymmetric disk can be readily derived via our CCGF expression, and

we draw on one more specific problem from magnetostatics to demonstrate how the CCGF reproduces the exact analytical expression for a potential problem where the solution can be expressed entirely in terms of the $m = 1$ ($Q_{\frac{1}{2}}$) nonaxisymmetric term.

Finally, for several "geometrically thick" configurations of uniform density, we provide numerical comparisons between $\Phi(\mathbf{x}_B)$ as derived from the CCGF method and as determined from (a) the traditional multipole method and (b) analytical prescriptions, where available. In §2.2.3 we comment on the computational advantages and disadvantages of the CCGF method when the objective is to determine values of the gravitational potential outside, but in close proximity to, flattened or elongated mass distributions. Generally speaking, for a given computational grid resolution we find that the CCGF method provides more accurate values of $\Phi(\mathbf{x}_B)$ in equal or less computational time than can be derived using the multipole method, but in certain situations the CCGF method can be quite demanding in terms of memory storage requirements.

2.2.1 Analytical Verifications and Propositions

2.2.1.1 Axisymmetric Systems with Vertical Extent

For an axisymmetric mass distribution, eq. (2.19) reduces to the form,

$$\Phi_0(R,z) = -\frac{2G}{\sqrt{R}}\, q_0, \tag{2.31}$$

with

$$q_0 \equiv \int_\Sigma d\sigma' \sqrt{R'}\, \rho(R',z')\, Q_{-\frac{1}{2}}(\chi) \tag{2.32a}$$

$$= \int_{\Sigma} d\sigma' \sqrt{R'}\, \rho(R', z')\, \mu K(\mu), \tag{2.32b}$$

where χ and μ have been defined by eqs. (2.16) and (2.24) respectively. As we shall illustrate in §2.2.2, this expression can be used effectively to compute the potentials outside of oblate spheroids, prolate spheroids, tori, or thick disks with arbitrarily complex $\rho(R, z)$ distributions.

It is important to note that eq. (2.31) provides an expression for the gravitational potential of an axisymmetric mass distribution that contains a *single* term and a single moment of the mass distribution q_0. In contrast to this, the corresponding expression for the potential in spherical coordinates [eq. (2.8)] requires a summation over an infinite number of terms, each containing a different moment of the mass distribution. Hence, eq. (2.31) provides an expression for the potential that is easier to evaluate and guaranteed to be more accurate (for a given computational grid resolution) than eq. (2.8). We strongly recommend its adoption in numerical algorithms that are designed to study self-gravitating, axisymmetric fluid flows.

2.2.1.2 Behavior on the Axis

In cylindrical coordinates,

$$\frac{1}{|\mathbf{x} - \mathbf{x}'|} = [R^2 + R'^2 - 2RR' \cos(\phi - \phi') + (z - z')^2]^{-\frac{1}{2}}. \tag{2.33}$$

Inserting eq. (2.33) into eq. (2.3) and taking the limit as R approaches zero, we see that the integral solution to the potential along the z-axis due to a mass distribution $\rho(\mathbf{x}')$ is

$$\lim_{R\to 0}\Phi(\mathbf{x}) = -G\int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'^2+(z-z')^2}}. \tag{2.34}$$

Here we demonstrate that this familiar, general solution to the potential along the z-axis can be derived from our compact expression for the cylindrical Green's function.

First we examine the axisymmetric component of the potential which, according to eqs. (2.22), (2.32b) and (2.24) is,

$$\Phi_0(\mathbf{x}) = -\frac{2G}{\pi}\int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{(R+R')^2+(z-z')^2}}\; K\Big(\Big[\frac{4RR'}{(R+R')^2+(z-z')^2}\Big]^{\frac{1}{2}}\Big). \tag{2.35}$$

On the z-axis, eq. (2.35) becomes

$$\lim_{R\to 0}\Phi_0(\mathbf{x}) = -\frac{2G}{\pi}\int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'^2+(z-z')^2}}\; \lim_{R\to 0} K\Big(\Big[\frac{4RR'}{R'^2+(z-z')^2}\Big]^{\frac{1}{2}}\Big). \tag{2.36}$$

According to Abramowitz & Stegun (1965; eq. [17.3.11]), $K(0) = \pi/2$, so from expression (2.36) we obtain

$$\lim_{R\to 0}\Phi_0(\mathbf{x}) = -G\int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'^2+(z-z')^2}}, \tag{2.37}$$

which exactly matches the familiar result for the axis potential given above.

We now demonstrate that all Φ_m vanish on the axis for $m \geq 1$. According to eq. (2.18), the nonaxisymmetric components of the potential are,

$$\Phi_m(\mathbf{x}) = -\frac{2G}{\pi}\int_V d^3x' \frac{\rho(\mathbf{x}')}{\sqrt{R'}}\cos[m(\phi-\phi')]\frac{1}{\sqrt{R}}\; Q_{m-\frac{1}{2}}(\chi). \tag{2.38}$$

If we insert eq. (2.29) into eq. (2.38), we obtain

$$\begin{aligned}\Phi_m(\mathbf{x}) &= -\frac{2G}{\sqrt{\pi}}\frac{\Gamma(m+\frac{1}{2})}{\Gamma(\frac{2m+3}{4})\ \Gamma(\frac{2m+1}{4})\ 2^{m+\frac{1}{2}}}\int_V d^3x'\frac{\rho(\mathbf{x}')}{\sqrt{R'}}\cos[m(\phi-\phi')] \\ &\times \sum_{n=0}^{\infty}\frac{\Gamma(\frac{2m+4n+3}{4})\ \Gamma(\frac{2m+4n+1}{4})}{\Gamma(m+1+n)\ \Gamma(1+n)}\left[\sqrt{R}\chi^{2n+m+\frac{1}{2}}\right]^{-1}. \qquad (2.39)\end{aligned}$$

From this expression we can see that, on the axis, the radial contribution to the nonaxisymmetric components of the potential is governed by the behavior of

$$\lim_{R\to 0}(\sqrt{R}\chi^{2n+m+1/2})^{-1} = \lim_{R\to 0}\left[\frac{2R'}{R'^2+(z-z')^2}\right]^{2n+m+\frac{1}{2}}R^{2n+m}, \qquad (2.40)$$

which vanishes for all $2n+m>0$. But, by definition in expression (2.39), $n \geq 0$ and $m \geq 0$. Hence, all of the nonaxisymmetric components of the potential vanish on the axis, thus providing a critical check on the validity of our cylindrical Green's function expansion.

2.2.1.3 Infinitesimally Thin Axisymmetric Systems

In the case of an *infinitesimally thin* axisymmetric disk located in the plane $z'=0$, the density distribution can be written as

$$\rho(R',z') = \Sigma(R')\delta(z'), \qquad (2.41)$$

where $\Sigma(R')$ is the surface density of the disk and $\delta(z')$ is a Dirac delta function. Inserting this expression for $\rho(R',z')$ into eq. (2.32b) and integrating over z' we obtain the following exact expression for the gravitational potential of any infinitesimally thin, axisymmetric disk:

$$\Phi_{0,disk}(R,z) = -\frac{2G}{\sqrt{R}}\int_0^\infty dR'\sqrt{R'}\ \Sigma(R')\ \mu_d K(\mu_d), \tag{2.42}$$

where

$$\mu_d \equiv \sqrt{\frac{4RR'}{(R+R')^2+z^2}}. \tag{2.43}$$

This equation exactly matches the expression for the potential of an infinitesimally thin, axisymmetric galaxy disk given, for example, by eq. (2-142a) of Binney & Tremaine (1987). It is now clear through eqs. (2.31) and (2.32) that this familiar expression can be generalized to axisymmetric configurations with arbitrary vertical extent.

2.2.1.4 Nonaxisymmetric Systems and Kalnajs Logarithmic Spirals

Here we demonstrate that the expression for the reduced potential of an infinitesimally thin, nonaxisymmetric disk that has been developed by Kalnajs (1971; see also, for example, §2.4b of Binney & Tremaine 1987) can be readily derived from our CCGF. Guided by a key functional relationship found in Morse & Feshbach (1953), we show through a brief derivation in Appendix A (see specifically eq. [A.5]) that,

$$\sum_{m=0}^{\infty} \epsilon_m \cos(m\phi)\ Q_{m-\frac{1}{2}}(\cosh\mu) = \frac{\pi}{\sqrt{2}}\frac{1}{\sqrt{\cosh\mu-\cos\phi}}. \tag{2.44}$$

Hence, expression (2.17) for the Green's function can be rewritten as,

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \frac{1}{\sqrt{2RR'}}\frac{1}{\sqrt{\cosh\xi-\cos(\phi-\phi')}}, \tag{2.45}$$

where,

$$\xi \equiv \cosh^{-1}(\chi) = \ln(\chi + \sqrt{\chi^2 - 1}). \tag{2.46}$$

Combining this expression with eq. (2.3), we may therefore also conclude that the "reduced potential,"

$$V(\mathbf{x}) \equiv \sqrt{R}\Phi(\mathbf{x}) = -G \int_V \frac{1}{\sqrt{R'}} \frac{\rho(\mathbf{x}')}{\sqrt{2[\cosh\xi - \cos(\phi - \phi')]}} d^3\mathbf{x}', \tag{2.47}$$

or,

$$V(\mathbf{x}) = -G \int_0^\infty \frac{dR'}{R'} \int_0^{2\pi} d\phi' \int_{-\infty}^\infty dz' \left\{ \frac{R'^{\frac{3}{2}} \rho(\mathbf{x}')}{\sqrt{2[\cosh\xi - \cos(\phi - \phi')]}} \right\}. \tag{2.48}$$

Now, if we consider an infinitesimally thin disk located in the plane $z' = 0$, the density distribution can be written as,

$$\rho(\mathbf{x}') = \delta(z')\ \Sigma(R', \phi'), \tag{2.49}$$

where $\Sigma(R', \phi')$ represents an arbitrary nonaxisymmetric surface density distribution, and the integral over z' in eq. (2.48) can be completed giving,

$$V(R, \phi) = -G \int_0^\infty \frac{dR'}{R'} \int_0^{2\pi} d\phi' [R'^{\frac{3}{2}} \Sigma(R', \phi')] \times \frac{1}{\sqrt{2[\cosh[\ln(R/R')] - \cos(\phi - \phi')]}}. \tag{2.50}$$

If, finally, we define a reduced surface density,

$$S(R, \phi) \equiv R^{\frac{3}{2}} \Sigma(R, \phi), \tag{2.51}$$

and adopt in place of R the independent variable,

$$u \equiv \ln R, \tag{2.52}$$

we obtain,

$$V(u,\phi) = -G \int_{-\infty}^{\infty} du' \int_{0}^{2\pi} d\phi' S(u',\phi') \mathcal{K}_{2D}(u-u',\phi-\phi'), \tag{2.53}$$

where

$$\mathcal{K}_{2D}(u-u',\phi-\phi') \equiv \frac{1}{\sqrt{2[\cosh(u-u')-\cos(\phi-\phi')]}}. \tag{2.54}$$

Eq. (2.53) is the expression Kalnajs (1971) has provided for the reduced potential of an infinitesimally thin, nonaxisymmetric disk. It is via this expression that Kalnajs has realized the utility of viewing nonaxisymmetric surface density distributions in terms of their various "logarithmic spiral" components.

Our expression (2.48) may now be viewed as a generalization of Kalnajs' reduced potential that applies to nonaxisymmetric structures of arbitrary vertical thickness, the key difference being that, in our more generalized expression for the reduced potential, the function $\mathcal{K}_{2D}(u-u',\phi-\phi')$ must be replaced by the function,

$$\mathcal{K}_{3D}(\chi,\phi-\phi') \equiv \frac{1}{\sqrt{2[\chi-\cos(\phi-\phi')]}}, \tag{2.55}$$

where, as defined in eq. (2.16), χ itself is a function that involves a non-trivial coupling between the coordinate variables R, R', z and z', which we

will further describe below. Although, as indicated by expression (2.46), it is possible to rewrite $\cosh^{-1}(\chi)$ in terms of a logarithmic function and, in so doing, transform eq. (2.55) into a form that more closely resembles Kalnajs' function $\mathcal{K}_{2D}$, the nontrivial coupling between coordinate variables within χ makes such a formulation less compelling in the full three-dimensional problem.

2.2.1.5 The $m = 1$ mode and the Magnetic Field of a Current Loop

A derivation of the magnetic field of a time-independent circular current loop of radius a, and current I has been provided in a multitude of classical electromagnetism textbooks (e.g., Landau & Lifshitz 1960; Jackson 1975). Here we demonstrate that this classic problem can be readily solved via the CCGF. In a magnetostatics problem we may calculate the magnetic field from a vector potential, $\mathbf{A}(\mathbf{x})$ as follows,

$$\mathbf{B}(\mathbf{x}) = \nabla \times \mathbf{A}(\mathbf{x}). \tag{2.56}$$

Then in the *Coulomb* gauge, the vector potential satisfies the following vector Poisson equation,

$$\nabla^2 \mathbf{A}(\mathbf{x}) = -\frac{4\pi}{c}\mathbf{J}(\mathbf{x}), \tag{2.57}$$

where $\mathbf{J}(\mathbf{x})$ is the current density and c is the speed of light. The integral solution of this vector Poisson equation produces the magnetic analogue of eq. (2.3), namely,

$$\mathbf{A}(\mathbf{x}) = \frac{1}{c}\int_V \frac{\mathbf{J}(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|} d^3x'. \tag{2.58}$$

In the case of a circular current loop located in the equatorial plane, $z' = 0$, the current density has only a ϕ component which is

$$\mathbf{J}(\mathbf{x}) = \hat{\phi} J_\phi, \tag{2.59}$$

where

$$J_\phi(\mathbf{x}') = I \cos(\phi')\delta(z')\delta(R' - a). \tag{2.60}$$

Since the final solution must be invariant under rotation, we choose our observing point to be at $\phi = 0$. Substituting eqs. (2.59) and (2.17) into eq. (2.58), we obtain the following expression for the ϕ component of the vector potential:

$$A_\phi = \frac{I}{\pi c}\sqrt{\frac{a}{R}} \int_0^{2\pi} d\phi' \cos(\phi') \sum_{m=0}^{\infty} \epsilon_m \cos(m\phi') Q_{m-\frac{1}{2}}(\chi_l), \tag{2.61}$$

where

$$\chi_l \equiv \frac{R^2 + a^2 + z^2}{2Ra}. \tag{2.62}$$

The only term in the summation that contributes is the $m = 1$ term, so eq. (2.61) becomes,

$$A_\phi = \frac{2I}{\pi c}\sqrt{\frac{a}{R}} Q_{\frac{1}{2}}(\chi_l) \int_0^{2\pi} \cos^2(\phi') d\phi' = \frac{2I}{c}\sqrt{\frac{a}{R}} Q_{\frac{1}{2}}(\chi_l), \tag{2.63}$$

which via eq. (2.23), can be rewritten as,

$$A_\phi = \frac{4Ia}{c\sqrt{(R+a)^2 + z^2}} \Big[\frac{(2 - \mu_l^2)K(\mu_l) - 2E(\mu_l)}{\mu_l^2} \Big]. \tag{2.64}$$

This identically reproduces the previously known result for the vector potential of a current loop (cf., eq. [5.37] in Jackson 1975).

2.2.1.6 The $m = 2$ and Other Isolated Fourier Modes

In §2.2.1.1, we used the CCGF method to derive a general expression that describes the $m = 0$ (axisymmetric) Fourier mode contribution to the gravitational potential for any mass distribution. Here we illustrate how similarly simple expressions for any other isolated azimuthal mode of a self-gravitating system can be derived via eq. (2.20). For an $m = 2$ distortion, for example, the two relevant Fourier components of the potential are,

$$\Phi_2^{1,2}(R,z) = -\frac{2G}{\sqrt{R}} \int_\Sigma d\sigma' \sqrt{R'}\ \rho_2^{1,2}(R',z')\ Q_{\frac{3}{2}}(\chi). \tag{2.65}$$

Utilizing eq. (2.26), which was derived in §2.1.2.2 via the recurrence relation for half-integer degree Legendre functions of the second kind, we are able to rewrite this expression for $\Phi_2^{1,2}(R,z)$ in terms of more familiar elliptic integrals as follows:

$$\Phi_2^{1,2}(R,z) = -\frac{2G}{3\sqrt{R}} \int_\Sigma d\sigma' \sqrt{R'}\ \rho_2^{1,2}(R',z')\ \mu[(4\chi^2-1)K(\mu)-4\chi(1+\chi)E(\mu)]. \tag{2.66}$$

Furthermore, in the case of an infinitesimally thin disk the Fourier components of the density can be written as,

$$\rho_2^{1,2}(R',z') = \Sigma_2^{1,2}(R')\ \delta(z'), \tag{2.67}$$

and we obtain the following exact expression for the $m = 2$ Fourier components of the potential of any infinitesimally thin, self-gravitating disk:

$$\Phi^{1,2}_{2,disk}(R,z) = -\frac{2G}{3\sqrt{R}}\int_0^\infty dR'\sqrt{R'}\ \Sigma_2^{1,2}(R')\ \mu_d$$
$$\times\Big[(4\chi_d^2-1)K(\mu_d) - 4\chi_d(1+\chi_d)E(\mu_d)\Big], \tag{2.68}$$

where $\chi_d \equiv 2/\mu_d^2 - 1$. This compact analytical expression should prove useful in, for example, studies of $m = 2$ spiral-arm instabilities in self-gravitating galaxy or protostellar disks.

2.2.2 Numerical Evaluations

Here we perform a variety of numerical tests in which we have discretized selected mass-density distributions on a uniformly-zoned cylindrical coordinate mesh. We have selected these models in order to elucidate the power that the CCGF method offers as a numerical technique for evaluating exterior potentials surrounding self-gravitating objects. Our comparison incorporates three methods for potential evaluation: (1) analytical potential-density expressions, as drawn from the works of other authors and detailed here in Appendix B; (2) the multipole method described in §2.1.1; and (3) our CCGF method, as outlined in §2.1.2. Where available, analytical solutions provide extremely useful verification of numerical methods for potential evaluation since any valid method should yield asymptotic convergence towards the analytical solution with increased grid resolution. Most of the models we have selected have known analytical solutions. In cases where the analytical solution does not exist, we simply compare the potentials obtained through the CCGF and multipole methods.

Table 2.1 lists the five models we have selected and Table 2.2 summarizes the seven tests that we have conducted using these models. Each of the five selected models has a uniform density distribution that is enclosed within a surface of a well-defined geometry as described by the "Type of Object" column in Table 2.1. Fig. 2.1 portrays the above described models through a three-dimensional isosurface visualization of each homogeneous object's boundary.

The oblate, prolate and toroidal objects are all axisymmetric. For the two oblate spheroids (Models I and II), the aspect ratios listed in Table 2.1 define the size of the equatorial axis relative to the polar axis. For the prolate spheroid (Model III), the 20:1 aspect ratio describes the size of the polar axis relative to the equatorial axis. For the torus (Model IV), the aspect ratio describes the size of the major radius of the torus relative to its minor, cross-sectional radius. Finally, we also have chosen one nonaxisymmetric model (Model V) which is a 20:10:1 triaxial homogeneous ellipsoid.

The column labeled "Grid Resolution" in Table 2.2 specifies the size of the computational grid or grids that was used in each test. For each axisymmetric model (Tests 1 – 6), the stated resolution $J \times K$ refers to the number of radial (J) and vertical (K) zones used; for Model V (Test 7), the stated resolution $J \times K \times L$ includes the number of azimuthal (L) zones that were used as well. For each of the tests identified in Table 2.2, we have determined the fractional error of a given numerical solution for the potential Φ by measuring at every location along the top and side boundaries of our cylindrical grid, the quantity,

Table 2.1: Models

Model	Type of Object	Aspect Ratio	Equation Number
I	oblate spheroid	5:1	B.2
II	oblate spheroid	20:1	B.2
III	prolate spheroid	20:1	B.2
IV	torus	20:1	—
V	triaxial ellipsoid	20:10:1	B.7

Figure 2.1: Three-dimensional wireframe diagrams illustrating the geometry of the five uniform-density models for which the external gravitational potential has been calculated herein using the CCGF technique (Φ^Q) and compared with approximate solutions obtained via a standard multipole technique (Φ^Y) and (where available) exact analytical expressions (Φ^A). See Table 2.1 for details regarding each test model's selected aspect ratio.

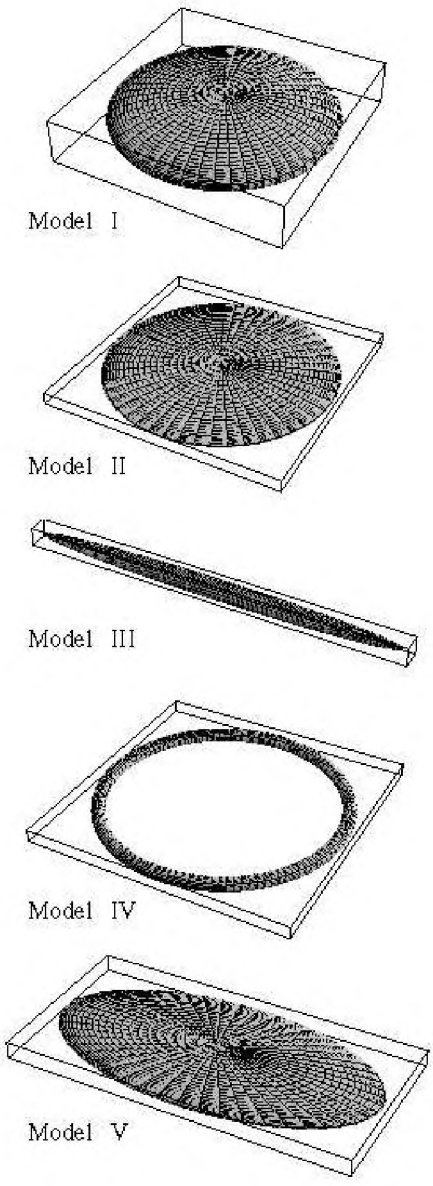

Model I
Model II
Model III
Model IV
Model V

Table 2.2: Tests

Test	Model	Grid Resolution
1	I	128×128
2	I	128×32
3	II	1024×64
4	III	32×512
5	IV	512×32
6[a]	I	$J \times K$
7	V	$512 \times 32 \times 256$

[a] $J = 32i, K = 8i$, with $(1 < i < 25)$

$$\epsilon \equiv \frac{\Phi - \Phi^*}{\Phi^*}, \tag{2.69}$$

where Φ^* is the "known" solution. Figures 2.2 - 2.7 present subsets of these error measurements in various ways.

In presenting the results of these tests, the numerically derived potential Φ is either the Newtonian potential generated via the multipole method, Φ^Y, or via the CCGF method, Φ^Q. Where available, the "known" solution Φ^* is given by the analytical solution, Φ^A, as drawn from the relevant Appendix B expression and identified by the entry in the "Equation Number" column of Table 2.1. Otherwise we take Φ^* to be Φ^Q, since we recognize it as the more correct numerical solution for the discretized model. Note that in Test 6, Model I has been re-examined using 25 different grid resolutions. This has been done in order to ascertain how the determination of Φ^Q relative to Φ^A improves with grid resolution.

2.2.2.1 Axisymmetric Models

For the four axisymmetric models listed in Table 2.1, Φ^Q has been determined via eq. (2.31) and its associated moment of the mass distribution as defined by eq. (2.32b). The thick-dashed curves in Figs. 2.2, 2.3, and 2.4 represent the fractional error obtained by comparing Φ^Q with Φ^A for Models I, II, and III, respectively. Since, as emphasized in §2.2.1.1, eq. (2.31) provides an expression for the gravitational potential that contains only one term, any error that arises in the determination of Φ^Q relative to Φ^A must be entirely attributed to the fact that, at any finite grid resolution, a numerical integration of eq. (2.32b) cannot possibly give the precise analytical answer. It is important to appreciate that this "failing" has nothing to do with our ability to evaluate the special function $K(\mu)$ accurately. Instead, it stems from the fact that the models for which we have analytically known potentials have spheroidal surfaces, and it is impossible to represent such surfaces precisely within a cylindrical coordinate mesh. Indeed, even a straightforward volume integration over the density distribution will give a total mass that is different from the analytically "known" mass because a spheroidal object cannot be perfectly represented in a cylindrical mesh. We shall return to this issue when discussing Test 6, below.

In contrast to this, errors in the determination of Φ^Y are dominated by the fact that, in any practical implementation of the multipole method, the summation over multipole moments must be truncated at some finite number of terms, l_{max}. Only in the limit $l_{max} \to \infty$ will the value of Φ^Y given by eq. (2.8) for an axisymmetric mass distribution converge to the

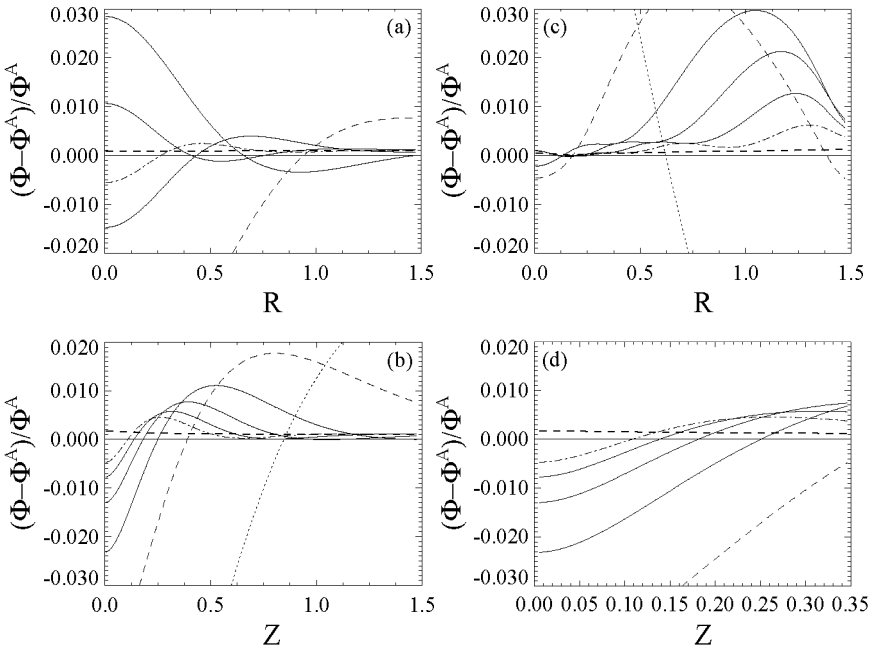
value of Φ^Q given by eq. (2.31), for example. Because the contribution that each multipole moment makes to the potential drops off as $r^{-(l+1)}$, a reasonably small error can be realized with a reasonably small value of l_{max} *if* the boundary cells at which Φ^Y is to be evaluated are placed at locations r that are fairly far from the mass distribution. For each of the seven tests listed in Table 2.2, Φ^Y has been determined for six different even values of l_{max} in the range $0 \leq l_{max} \leq 10$ in an effort to illustrate how rapidly the determination of Φ^Y converges toward Φ^A and Φ^Q as more and more terms are included in the l summation. We illustrate results only for even values of l_{max} because all five models listed in Table 2.1 exhibit reflection symmetry through the equatorial plane and, by design, this symmetry forces all odd multipole moments to be identically zero. In each of the Figs. 2.2 – 2.5 and 2.7, dotted curves illustrate errors in the determinations of Φ^Y when $l_{max} = 0$; thin-dashed curves represent errors resulting from setting $l_{max} = 2$; and the dash-dot curves show errors in Φ^Y resulting from the inclusion of even multipole moments through $l_{max} = 10$. The three solid curves generally lying between the thin-dashed curve and the dash-dot curve in each figure represent, in sequence, errors in Φ^Y that result from setting $l_{max} = 4, 6$, and 8.

Figure 2.2 illustrates results from Tests 1 and 2 on Model I (the 5:1 oblate spheroid). In both of these tests, our computational mesh had 128 radial grid zones of uniform radial (ΔR) and vertical ($\Delta z = \Delta R$) thickness, and the oblate spheroid was positioned such that its equatorial radius extended out to grid location 123. Tests 1 and 2 differed in only one respect, as indicated in Table 2.2: With a cylindrical computational mesh that had four times as

many vertical zones, Test 1 was designed to place the top boundary of the computational grid much farther from the surface of the oblate spheroid than in Test 2. Because every point along the boundary of the grid in Test 1 was at a radial location r_B greater than the equatorial radius of the Model I spheroid, Φ^Y was evaluated using eqs. (2.5) and (2.6), with m set equal to zero, as in eqs. (2.8) and (2.9). However, in Test 2 it was also necessary to include an evaluation of Φ_{int} (eq. [2.11]) and, hence, a separate evaluation of $q_{lm}^{>}$ and $q_{lm}^{<}$, for each zone along the top of the grid boundary. As a result (see the related discussion in §2.2.3, below), the evaluation of Φ^Y in Test 2 was much more computationally demanding than in Test 1. Errors in the determination of the potential along the top boundary of these two different cylindrical computational domains are shown in Figs. 2.2a and 2.2c; corresponding errors along the side boundary are displayed in Figs. 2.2b and 2.2d.

The results presented in Fig. 2.2 highlight three key points that have been discussed in a more general context, above. First, in both tests Φ^Q very nearly follows the analytically derived potential Φ^A at all locations on the grid boundary. It is, however, everywhere offset from Φ^A by a small amount. This small offset is due almost entirely to the effect mentioned above of being unable to properly represent a perfect spheroidal surface within a cylindrical coordinate grid. Second, as l_{max} is increased, the multipole method yields better and better results which converge toward the solution Φ^Q, but in no case is the typical error in Φ^Y smaller than the typical error in Φ^Q. Third, for a given choice of l_{max}, the typical error in Φ^Y measured along the top of the cylindrical grid is smaller in Test 1 (Fig. 2.2a) than it is in Test 2

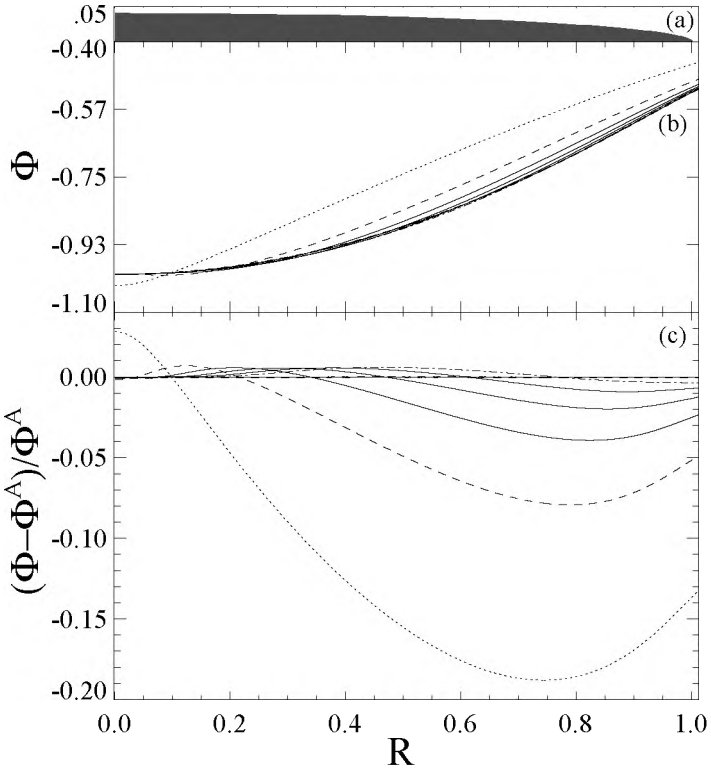
Figure 2.2: Model I (5:1 oblate spheroid). The fractional error in the numerically determined gravitational potential (calculated via two different Green's function techniques) relative to the analytically known potential Φ^A is shown here as a function of position R along the top and Z along the side boundaries of the selected cylindrical computational mesh, as defined in Table 2.2. Frames (a) and (b) illustrate results from Test 1 in which the top boundary of a 128×128 computational mesh has been positioned at the same distance from the center of the grid as the side boundary. A thin, solid horizontal line has been drawn at zero for reference purposes. The thick dashed line running approximately horizontally across both frames shows the errors in the potential as determined via the CCGF technique, *i.e.*, $(\Phi^Q - \Phi^A)/\Phi^A$. (See the discussion associated with Test 6 for an explanation of why these curves are slightly offset from zero.) All other curves illustrate the errors in the potential as determined via the standard multipole technique *i.e.*, $(\Phi^Y - \Phi^A)/\Phi^A$, as the limiting number of terms in the multipole expansion is increased successively by 2 from $l = 0$ (dotted curve) to $l = 2$ (dashed curve), etc., through $l = 10$ (dot-dashed curve). Frames (c) and (d) illustrate the same type of information as displayed in frames (a) and (b), respectively, but for Test 2 in which the top boundary of a 128×32 computational mesh has been placed a factor of 4 closer to the center of the grid, in a position that lies very close to the surface of the Model I spheroid. Results from this Test 2 also appear as the example marked "A" in Fig. 2.6.


(a)
(b)
(c)
(d)
(Φ–Φ^A)/Φ^A
R
Z
0.030
0.020
0.010
0.000
-0.010
-0.020
-0.030
0.0
0.5
1.0
1.5
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35

(Fig. 2.2c). This is because the top of the grid is farther from the surface of the mass distribution in Test 1 than in Test 2.

Figure 2.3 illustrates the results from Test 3 on Model II (the 20:1 oblate spheroid). This test is similar to Test 2 in that the top boundary of the computational grid has been placed very close to the surface of the spheroid. In one quadrant of a meridional plane cutting through Model II, Fig. 2.3a illustrates precisely where the top and side cylindrical boundaries have been placed with respect to the surface of the spheroid. Test 3 differs from Test 2, however, in that the spheroidal model for which the gravitational potential is being determined has a relatively extreme (20:1) axis ratio. In order to maintain a uniformly zoned computational grid, a correspondingly extreme radial to vertical (1024×64) grid resolution was adopted for Test 3. In addition to displaying in Fig. 2.3c the fractional errors that resulted from our determinations of Φ^Y and Φ^Q along the top boundary of the computational grid, we have shown in Fig. 2.3b the functional variation of the boundary potentials from which the errors displayed in Fig. 2.3c have been derived. This is a particularly severe test of the multipole moment method because the potential of extremely flattened mass distributions is not well-represented by an expansion in terms of spherical harmonics. Notice, however, that the CCGF method has no difficulty evaluating the potential for this extremely flattened spheroid; in both Figs. 2.3b and 3c the thick-dashed curve representing Φ^Q is nearly indistinguishable from the thin solid line representing Φ^A.

Figure 2.3: Model II (20:1 oblate spheroid); results from Test 3, as defined in Table 2.2. (a) A meridional cross-section through Model II is shown in which the equatorial radius of the object extends to $R = 1.0$ and the polar radius extends to $Z = 0.05$. The top and right-hand edges of this figure frame illustrate precisely the positioning of the top and side boundaries of the 1024×64 cylindrical computational mesh have been positioned, relative to the highly flattened spheroidal surface. (b) The gravitational potential Φ is plotted as a function of R along the top boundary of the computational mesh, as determined analytically (thin solid curve), via the CCGF technique (thick dashed curve), and via the standard multipole technique as the limiting number of terms in the multipole expansion is increased successively by 2 from $l = 0$ (dotted curve) to $l = 2$ (dashed curve), etc., through $l = 10$ (dot-dashed curve). (c) Similar to frames *a* and *c* of Fig. 2.2, the fractional error in the numerically determined gravitational potential relative to the analytically known potential Φ^A is shown as a function of position R along the top of the selected cylindrical computational mesh. These fractional errors have been derived directly from the values of Φ displayed in (b), and the meaning of the various curves is the same as in (b). Note, in particular, that at all radii the error in Φ^Q (bold dashed curve) is almost indistinguishable from zero.


.05
-0.40
-0.57
-0.75
-0.93
-1.10
Φ
(a)
(b)
(c)
0.00
-0.05
-0.10
-0.15
-0.20
(Φ–Φ^A)/Φ^A
0.0
0.2
0.4
0.6
0.8
1.0
R

In Fig. 2.4 we show results from Test 4 on Model III, the 20:1 prolate spheroid. For this test, the model has been discretized on a 32×512 cylindrical grid. In this case, the primary challenge for both the multipole moment and CCGF methods is to accurately evaluate the potential along the side, rather than the top, of the computational grid. Figure 2.4a shows the fractional error as a function of z along the side of this highly elongated coordinate mesh while Fig. 2.4b shows the fractional error as a function of R along the top of the grid. Once again Φ^Q appears to be tracking the analytical solution extremely well and Φ^Y is seen to be converging toward Φ^Q (and Φ^A) as the maximum number of terms in the multipole expansion is increased. However, for a given value of l_{max}, the typical error in Φ^Y appears to be larger for the prolate model (Fig. 2.4) than for the oblate model with the same aspect ratio (Fig. 2.3).

Figure 2.5 shows results for Test 5 on Model IV, an axisymmetric torus with a 20:1 aspect ratio. The information that has been displayed in the three frames of Fig. 2.5 is analogous to the information that was displayed in Fig. 2.3 for Model II. Specifically, Fig. 2.5a shows a meridional cross-section through the torus, with the symmetry axis of the torus (and the cylindrical computational grid) at the left, while the top and right-hand edges of the frame identify precisely where the top and side cylindrical boundaries were placed with respect to the surface of the torus. In this case we do not have an analytical solution for the potential against which to compare Φ^Y or Φ^Q, but in Fig. 2.5b it is clear that as l_{max} is increased Φ^Y is converging toward

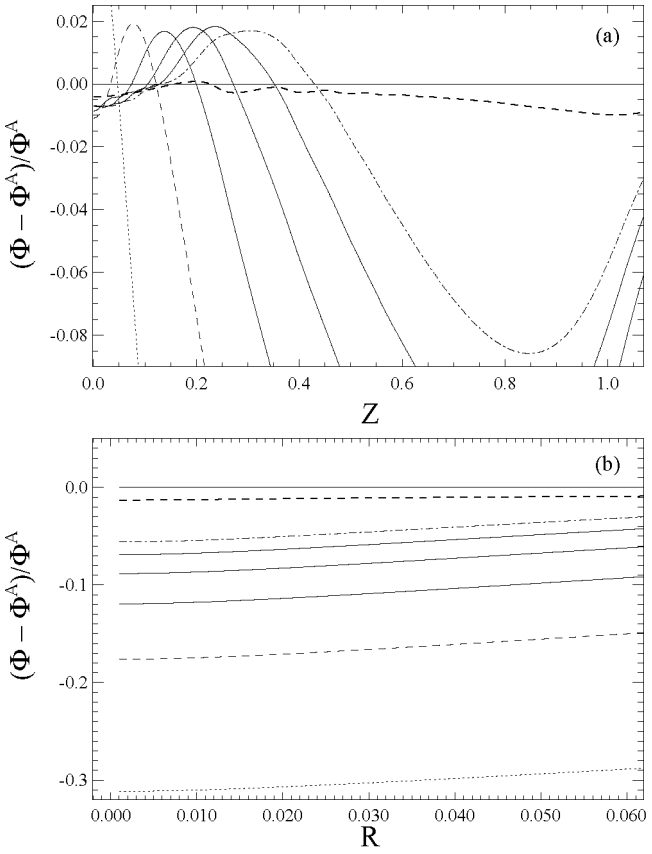


Figure 2.4: Model III (20:1 prolate spheroid); results from Test 4, as defined in Table 2.2. (a) Analogous to frames *b* and *d* of Fig. 2.2, the fractional error in the numerically determined gravitational potential relative to the analytically known potential Φ^A is shown as a function of position Z along the side boundary of the selected cylindrical computational mesh. (b) Analogous to frames *a* and *c* of Fig. 2.2 and frame *c* of Fig. 2.3, the fractional error in the numerically determined gravitational potential relative to the analytically known potential Φ^A is shown as a function of position R along the top boundary of the selected cylindrical computational mesh.

Φ^Q (thick-dashed curve), so in Fig. 2.5c the error in Φ^Y has been measured relative to Φ^Q.

For Test 6 we have returned to Model I to illustrate how the calculated error in Φ^Q improves with increasing computational grid resolution. As indicated in Table 2.2, for this test we have computed the value of the potential on the boundary of 25 different sized grid meshes, all of which are integer multiples of a 32×8 cylindrical (R,z) grid. As is explained in detail in the figure caption, Fig. 2.6a illustrates how the maximum, minimum, and mean fractional error in Φ^Q vary along the top boundary of the cylindrical grid as the radial grid resolution is increased from $J = 32$ to $J = 608$, and Fig. 2.6c illustrates how the maximum, minimum, and mean fractional error in Φ^Q vary along the side boundary of the cylindrical grid as the vertical resolution is increased from $K = 32$ to $K = 200$. Along both the top and side boundaries we have been able to achieve mean fractional errors $\sim 10^{-5}$. For five selected grid resolutions (labeled B, C, D, E, and F in each frame of Fig. 2.6), we also have shown in detail how the fractional error in Φ^Q varies across the top (Fig. 2.6b) and along the side (Fig. 2.6d) boundaries of the grid. The curves in Fig. 2.6b (or Fig. 2.6d) should each be compared directly with the thick-dashed curve plotted in Fig. 2.2b (or Fig. 2.2d), which presents the equivalent information from Test 2 – a relatively low resolution (128×32), but otherwise identical calculation that also shows up and is labeled "A" in the results of Test 6.

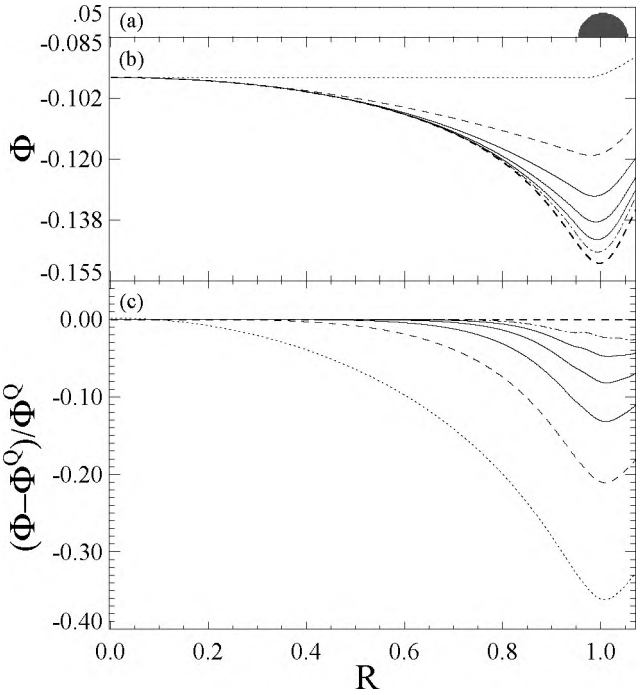
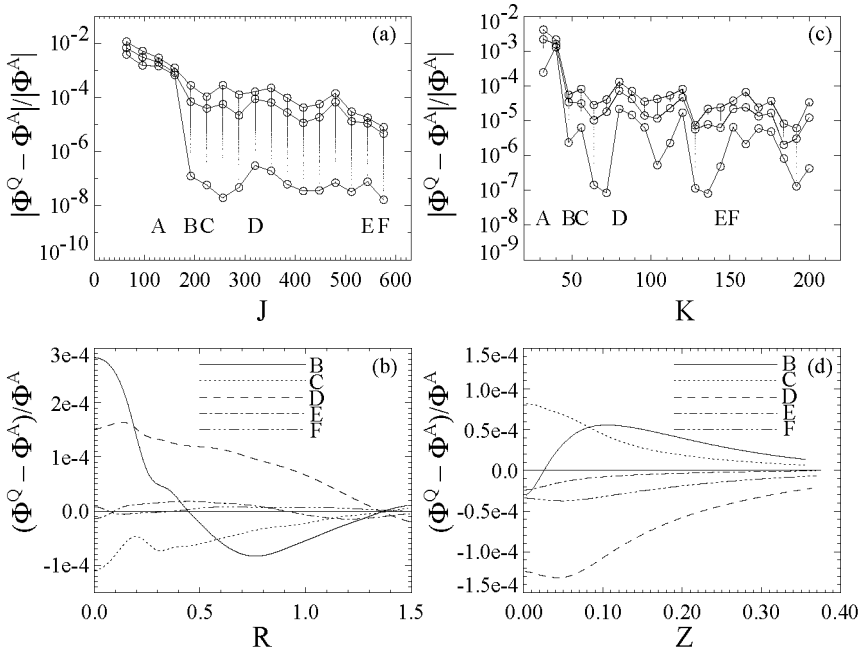


Figure 2.5: Model IV (20:1 torus); results from Test 5. (a) Analogous to Fig. 2.3a, a meridional cross-section through Model IV in which the major and minor radii of the torus are 1.0 and 0.05, respectively. The top and right-hand edges of this figure frame illustrate precisely the positioning of the top and side boundaries of the 512×32 cylindrical computational mesh have been positioned, relative to the surface of the slender torus. (b) Analogous to Fig. 2.3b, the gravitational potential Φ is plotted as a function of R along the top boundary of the computational mesh, as determined via the CCGF technique (thick dashed curve), and via the standard multipole technique as the limiting number of terms in the multipole expansion is increased successively by 2 from $l = 0$ (dotted curve) to $l = 2$ (dashed curve), etc., through $l = 10$ (dot-dashed curve). (c) Analogous to Fig. 2.3c, but because the potential exterior to a torus is not known analytically, the fractional error in the numerically determined gravitational potential is shown here relative to the potential Φ^Q as determined from the CCGF technique. The meaning of the various curves is the same as in (b).

Figure 2.6: Model I (5:1 oblate spheroid); results from Test 6. Fractional errors in the gravitational potential derived via the CCGF technique using 25 different cylindrical grid resolutions (see Table 2.2) to resolve the oblate spheroidal mass distribution. (a) For a specified radial grid resolution J, the vertical column of dots identifies on a logarithmic scale the full range of fractional errors that have been derived along the top boundary of the computational mesh. Each dot identifies the fractional error at a specific radial grid location so, for example, for the column of dots (labeled A) that is drawn from a calculation using a grid resolution $J = 128$ (as in Test 2; see also Fig. 2.3), 128 different dots have been plotted showing errors that range from 1.5×10^{-6} to 2×10^{-3}. At each grid resolution J, an open circle has been drawn to identify the largest, smallest, and median error; a solid line connecting the circles helps the eye recognize an overall trend in computed errors as the resolution of the model is improved. (b) Analogous to the thick dashed curve shown in Fig. 2.2c, the fractional error in the gravitational potential determined via the CCGF technique Φ^Q relative to the analytically known potential Φ^A is shown as a function of position R along the top of the selected cylindrical computational mesh, but for several different grid resolutions. The curves labeled B through F are drawn from models having the grid resolutions J as indicated by the corresponding column labels in frame a of this figure. (c) and (d) Same as frames a and b of this figure, respectively, but showing fractional errors that have been derived along the side boundary of the computational mesh from calculations using various vertical grid resolutions K (see Table 2.2).


(a)
(b)
(c)
(d)
A BC D EF
B
C
D
E
F
J
K
R
Z

We should point out that the fractional errors presented in Fig. 2.6 for Test 6 have all been calculated in a slightly different manner from the fractional errors that have been presented for Tests 1 – 5. Before comparing Φ^Q to Φ^A in Test 6, we have renormalized the total mass that has been used in the determination of Φ^A to correspond with the total mass that results from a discretization of Model I inside our cylindrical computational grid of the specified $(J \times K)$ resolution. As explained earlier in the context of Tests 1 and 2, the thick-dashed curves in Fig. 2.2 are slightly offset from zero primarily because of a slight discrepancy in mass that arises from trying to map a perfect spheroid onto a cylindrical coordinate mesh. By adjusting the mass that is being used in the analytical determination of the gravitational potential for Model I to account for this discrepancy, we are able to present the fractional errors in such a way that they asymptotically approach zero at the largest illustrated values of R (Fig. 2.6b) and z (Fig. 2.6d). We also suspect that geometric imperfections arising from the discretization of the flattened spheroid are also responsible for the fact that the typical fractional errors shown in Figs. 2.6a and 2.6c level out around 10^{-5} and do not continue to decrease with increasing grid resolution.

2.2.2.2 A Nonaxisymmetric Model

In an effort to illustrate how well the CCGF method works for nonaxisymmetric mass distributions, we have developed a test based on the analytically known potential exterior to a triaxial homogeneous ellipsoid, as given in Appendix B by eq. (B.7). Specifically, as detailed in Tables 2.1 and 2.2 for Test 7, we have embedded an homogeneous triaxial ellipsoid with a 20:10:1

axis ratio in a uniformly zoned cylindrical mesh with $512 \times 32 \times 256$ zones in the R, z, and ϕ directions, respectively. Test 7 is similar to Test 3 in the sense that the top and side boundaries of the computational grid were positioned just outside the surface of the ellipsoid in such a way that a vertical cross-section through the configuration that contains the major and minor axes of the ellipsoid looks identical to Fig. 2.3a. As a result, a vertical cross-section containing the minor and intermediate axes of the ellipsoid would show that, in the equatorial plane of the grid, the ellipsoidal surface extends only half-way out to the side boundary of the computational grid. Hence, we should expect any numerical evaluation of the potential on the top and side boundaries of our cylindrical grid to produce better results at azimuthal angles near the intermediate axis of the ellipsoid (*i.e.*, near $\phi = \pi/2$ and $3\pi/2$) than at azimuthal angles near the ellipsoid's major axis ($\phi = 0$ and π; see Fig. 2.7b, below).

The analytical potential outside of an homogeneous, triaxial ellipsoid contains an infinite number of azimuthal Fourier components. When the ellipsoid is discretized and placed inside of a grid with a finite number of azimuthal zones, L (in our case, $L = 256$), we know by Fourier's Theorem that the "exact" potential corresponding to this discretized object will exhibit, at most, Fourier components extending up to mode $m = L/2$ (in our case, $m = 128$). As we have shown in §2.1.2.1 (specifically, eq. [2.20]), via the CCGF method the amplitude and phase of each one of these Fourier modes can be determined precisely by performing a single integral over the mass distribution, weighted by the appropriate special function, $Q_{m-\frac{1}{2}}(\chi)$. In contrast to this

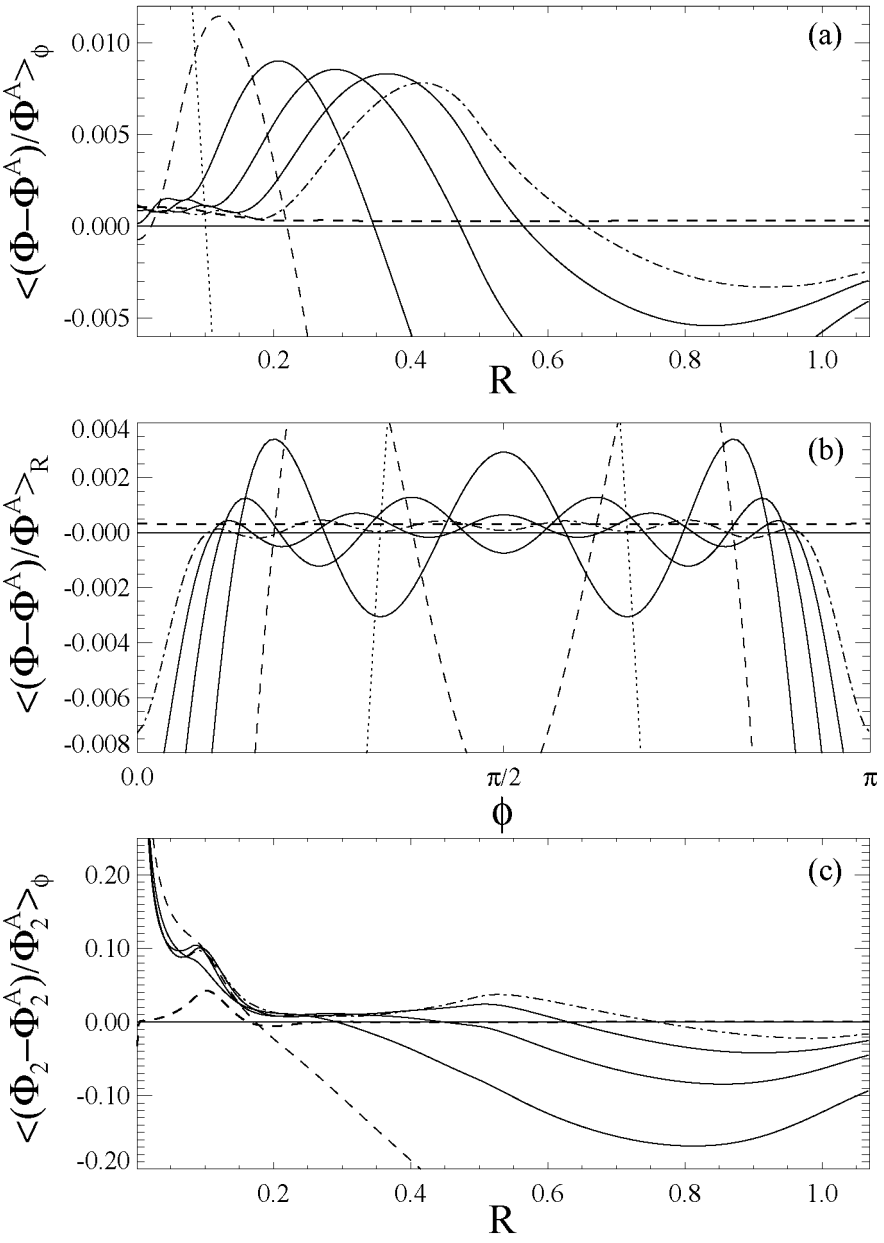
(see §2.1.1), when the method of multipole moments is employed, each of the azimuthal Fourier modes can only be determined exactly via a summation over an infinite number of terms ($l = 0$ to ∞), each one of which requires a separate integral over the mass distribution. Hence, by analogy with our determination of the axisymmetric potential, the multipole method can be implemented in the context of nonaxisymmetric mass distributions only if the l summation is truncated to a finite number of terms for each separate azimuthal Fourier mode. In a practical implementation of either method, it is computationally prudent to limit the calculation of Fourier mode amplitudes to a number substantially smaller than $m = L/2$, in which case one must admit that even the CCGF method can at best produce only an approximation to the "exact" discretized potential. But at least the CCGF method provides an accurate determination of the amplitude and phase of each of the included azimuthal Fourier modes whereas, by truncating the l summation, the multipole method cannot.

In conducting Test 7, we have included in the evaluation of Φ^Y even terms through $l_{max} = 10$ and, for each value of l, even azimuthal modes through $m = \pm l$. (All odd azimuthal moments of the mass distribution are guaranteed to be zero because Model V exhibits a periodic symmetry about the azimuthal angle $\phi = \pi$ as well as about $\phi = 0$ or 2π.) Therefore, in our evaluation of the double summation in eq. (2.5) to calculate Φ^Y in Test 7, 36 separate terms have been included. In addition, we have had to evaluate an entirely independent set of 36 terms associated with the summation in eq. (2.11) because, as in Tests 2, 3, and 5, most of the zones along the top

boundary of our computational grid had radial locations $r_B < a_1$, that is to say, at least some of the material enclosed by Model V's ellipsoidal surface fell outside a sphere of radius r_B. In contrast to this, when evaluating Φ^Q at each grid boundary location via eq. (2.19), we included only 16 terms. But these 16 terms permitted us to include azimuthal Fourier mode contributions to the potential up through mode $m = 30$ because the odd azimuthal modes were guaranteed to be zero.

Figures 2.7a and 2.7b show how closely our determination of Φ^Y and Φ^Q in Test 7 come to matching the analytical potential Φ^A for Model V. Rather than trying to display the errors in Φ^Y and Φ^Q at all grid boundary locations, Fig. 2.7a displays azimuthally averaged errors as a function of R along the top of the computational grid and Fig. 2.7b displays radially averaged errors as a function of ϕ over the same region. Being azimuthally averaged, the error measurements presented in Fig. 2.7a do not tell us much that was not already apparent in our examination of the corresponding axisymmetric spheroid (see Test 3 and, specifically, Fig. 2.3c). However, Fig. 2.7b is clearly illustrating something new. It illustrates that the potential Φ^Q determined through the CCGF method (represented by the thick-dashed line) represents the *azimuthal* variation of the potential outside of triaxial ellipsoid very accurately. We also see in Fig. 2.7b that, as l_{max} is increased, Φ^Y approaches Φ^Q.

Figure 2.7: Model V (20:10:1 triaxial ellipsoid); results from Test 7. (a) Analogous to Fig. 2.3c, except that, at each radius, the fractional error has been derived from an azimuthal average because Model V is not an axisymmetric configuration. (b) In an effort to display information that is complementary to the results shown in frame a for this nonaxisymmetric configuration, the fractional error in the derived potential is shown as a function of azimuthal angle ϕ. The displayed error has been derived from a radial average at each angular position. (c) The error in the $m = 2$ Fourier component of the potential is displayed as a function of R along the top boundary of the computational mesh. In all three frames, by analogy with Fig. 2.3c, fractional errors have been determined via the CCGF technique (thick dashed curve), and via the standard multipole technique as the limiting number of terms in the multipole expansion is increased successively by 2 from $l = 0$ (dotted curve) to $l = 2$ (dashed curve), etc., through $l = 10$ (dot-dashed curve).


(a)
0.010
0.005
0.000
-0.005
0.2
0.4
0.6
0.8
1.0
R
<(Φ−Φ^A)/Φ^A>_φ
(b)
0.004
0.002
-0.000
-0.002
-0.004
-0.006
-0.008
0.0
π/2
π
φ
<(Φ−Φ^A)/Φ^A>_R
(c)
0.20
0.10
0.00
-0.10
-0.20
0.2
0.4
0.6
0.8
1.0
R
<(Φ_2−Φ_2^A)/Φ_2^A>_φ

Finally, via a Fourier analysis of Φ^A, we have determined the correct amplitude as a function of radius of a single, isolated azimuthal mode, Φ_2^A, for Model V, and in Fig. 2.7c we have compared this function with the corresponding $m = 2$ Fourier mode amplitudes of Φ^Y and Φ^Q. As a point of reference, the $m = 2$ Fourier amplitude Φ_2^Q has been derived via the integral expression (2.66) given in §2.2.1.6. Fig. 2.7c shows in a somewhat cleaner manner than does Fig. 2.7b that the CCGF method works as well for the determination of the gravitational potential of nonaxisymmetric mass distributions as it does for axisymmetric systems. At most radii, Φ_2^Q is almost indistinguishable from Φ_2^A. Note, however, that near the z-axis of the grid (*i.e.*, near the polar axis of the ellipsoid), Φ_2^Q does differ from Φ_2^A by a few percent. This deviation almost certainly occurs because we have used only 32 vertical zones to resolve Model V's highly flattened mass distribution. Hence, the upper surface of our discretized mass model does not reproduce well the smooth quadratic surface of the analytically defined ellipsoid. Similar, although lower amplitude, deviations can be found near the z-axis in Fig. 2.2c (Test 2), Fig. 2.4a (Test 4), and Fig. 2.7a. Once again, it is fair to say that Φ_2^Q provides a more correct description of the gravitational potential for the discretized mass model than does Φ_2^A. This statement is supported by the fact that, as l_{max} is increased, Φ^Y is converging toward Φ^Q in Figs 2.7a and 2.7c, rather than toward Φ^A.

2.2.3 Computational Demands

Here we compare the computational demands of the multipole moment and CCGF methods. We do so not from the standpoint of a static problem

whose solution need only be determined once, but from the standpoint of a dynamical problem in which the system's two- or three-dimensional density distribution is changing with time, in which case a solution to the gravitational potential must be frequently redetermined in order to ensure that the potential is at all times consistent with the density distribution.

We will assume that, during such an evolutionary simulation, the cylindrical computational grid and the positions along the grid boundaries at which the potential $\Phi(\mathbf{x}_B)$ is to be determined do not change with time. Under this assumption, it is clear that, whichever Green's function method is being used, the terms included in the Green's function itself do not vary with time because these terms are only functions of the coordinates. Hence, the functions $Y_{lm}(\theta, \phi)$ (for the multipole moment method) or $Q_{m-\frac{1}{2}}(\chi)$ (for the CCGF method) need only be calculated once, as appropriate, for each grid cell location and stored in memory for reuse throughout a time-evolutionary calculation. The primary calculational cost associated with either Green's function method therefore has very little to do with the cost of evaluating various Y_{lm} or the $Q_{m-\frac{1}{2}}$ expressions. Instead, the cost is directly related to the number of integrals N over (moments of) the mass distribution that must be reevaluated each time the mass-density distribution of the evolving system is updated.

For a (two-dimensional) mass distribution that is axisymmetric, but that otherwise exhibits no special geometric symmetries, the multipole moment method includes $l_{max} + 1$ terms in the Green's function expansion whereas the CCGF method contains only one. However, because the argument χ of

the special function $Q_{m-\frac{1}{2}}(\chi)$ is, itself, a function of the boundary coordinates (R, z), a separate moment of the mass distribution must be calculated for each grid boundary location. Hence, for the CCGF method, the number of moments N^Q that must be reevaluated each time the mass-density distribution changes is,

$$N^Q = 2J + K, \tag{2.70}$$

where, as in Table 2.2, J and K specify the radial and vertical grid resolutions, respectively, and the factor of 2 indicates that in general "J" boundary values must be determined along the bottom as well as along the top of the cylindrical grid. In contrast to this, the terms in the multipole moment (*i.e.*, spherical coordinate Green's function) expansion are not explicitly functions of the boundary coordinates, so

$$N^Y = l_{max} + 1. \tag{2.71}$$

Now, as discussed earlier, in order to achieve the same level of accuracy with the multipole moment method as can be achieved with the CCGF method, l_{max} must be set to ∞. But if, in practice, one is satisfied with the level of accuracy achieved by setting l_{max} to a value $l_{max} < (2J + K - 1)$, then $N^Y/N^Q < 1$, and one may conclude that the multipole method is computationally less expensive than the CCGF method.

However, this is not the full story. Even though the terms in the multipole moment expansion are not explicitly functions of the boundary coordinates, the limits on the volume integration for each moment of the mass distribution

will be a function of the boundary coordinates unless every point $\mathbf{x}_B$ along the boundary of the computational grid is at a radial location r_B that is greater than *all* interior grid locations at which matter resides. (See the related discussion associated with eq. [2.10] in §2.1.1.) Test 1 (see Figs. 2.2a and 2.2b) is the only test presented above for which this special condition was true. By setting $J = K$, every point along the top boundary of our cylindrical grid was at a radial location r_B greater than the equatorial radius of the 5:1 oblate spheroid, so the number of separate moments of the mass distribution that had to be evaluated in Test 1 was, indeed, $N^Y = l_{max} + 1$. However, as explained in §2.1.1, for situations in which the boundary of the grid is positioned close to the surface of a flattened or elongated mass distribution, it is necessary to calculate a separate set of "interior" and "exterior" mass moments for the majority of boundary locations.

For example, for mass distributions that are flattened along the symmetry axis, as in our Tests 2, 3, 5, and 6, boundary locations along the side of the grid do not require separate sets of mass moments but most boundary locations along the top and bottom of the grid do. Hence,

$$N^Y \approx 4J(l_{max} + 1), \tag{2.72}$$

where the extra factor of 2 comes from having to determine both interior and exterior moments for each value of l, as shown in eq. (2.10). Therefore, $N^Y/N^Q \sim l_{max}$, and the (less accurate) multipole moment method proves to be more expensive to implement computationally than the CCGF method.

For a nonaxisymmetric (three-dimensional) mass distribution, the CCGF method will require the same number of moments as in the axisymmetric case *for each separate azimuthal Fourier mode.* Hence, if the discrete Fourier series is truncated at mode number m_{max}, the number of moments N^Q that must be reevaluated each time the mass-density distribution changes is,

$$N^Q = 2m_{max} \times (2J + K), \tag{2.73}$$

where the leading factor of 2 comes from the fact that each Fourier mode requires the determination of both an amplitude and a phase. In the optimum situation where the boundary of the computational grid is everywhere outside the mass distribution, in three dimensions the multipole moment method will require the evaluation of

$$N^Y \approx \sum_{l=0}^{l_{max}} [2l + 1] = (l_{max} + 1)^2 \tag{2.74}$$

separate moments (unless the strategic decision is made to set $m_{max} \neq l_{max}$). In most situations, then, N^Y/N^Q will be less than unity, as in the corresponding optimum axisymmetric case, but the ratio will be somewhat larger here.

Again, though, for situations in which the boundary of the grid is positioned close to the surface of a flattened or elongated mass distribution, the number of moments required for the multipole moment method climbs substantially. For example, for a flattened nonaxisymmetric mass distribution like the one examined above in connection with Test 7,

$$N^Y \approx 4J \times \sum_{l=0}^{l_{max}} [2l+1] = 4J(l_{max}+1)^2, \tag{2.75}$$

and the ratio N^Y/N^Q becomes even larger than it was for the corresponding axisymmetric case. Hence, in connection with a broad range of astrophysically interesting, two- and three-dimensional fluid flow problems, we have found the CCGF method to be not only much more accurate but also less expensive to implement than the traditional multipole method.

One note of caution is in order. Because the argument χ of the special function $Q_{m-\frac{1}{2}}(\chi)$ is a function of both coordinates of the interior mass (R', z'), at the beginning of any time–evolutionary simulation a 2D array of "Q" values must be calculated at each location along the boundary of the grid and for each discrete Fourier mode m. Hence, although the expense associated with the calculation of this global "Q" array can be confined to initialization routines, it must generally be a four-dimensional array having dimensions $\sim [J \times K \times m_{max} \times (2J+K)]$. As a result, the CCGF method can be quite demanding in terms of storage space. Because, for a given azimuthal mode number m, the function $Q_{m-\frac{1}{2}}(\chi)$ is very smooth over the entire range of χ, it may prove to be more practical to store only m_{max} one-dimensional arrays that could be referenced by all boundary grid cells in which the particular $Q_{m-\frac{1}{2}}$ function has been evaluated at a reasonably large number and sufficiently wide range of discrete values of χ. Then, when performing its own evaluation of the moments of the mass distribution, each boundary cell could evaluate $Q_{m-\frac{1}{2}}(\chi)$ as needed via an interpolation within the discretized array. We have not yet implemented such a scheme, although as we begin

to investigate problems having sizes larger than the one illustrated in Test 7, above, we will probably need to do so.

3. A Compact Green's Function Expansion for Axisymmetric Coordinate Systems

One primary contribution from chapter 2 was the discovery that the Green's function in cylindrical coordinates can be written in an extraordinarily compact form, namely, eq. (2.15). In order to better understand what it is that we have uncovered, we investigate in this chapter the nature or, specifically, the geometry of our solution. The first time we brought up an image of the meridional variations in $Q_{-\frac{1}{2}}(\chi)$, it appeared to us that the contours of constant χ were circles. In particular, when we first mathematically examined the structure of χ, sure enough, contours of constant χ were circles emanating from the $\mathbf{x} = \mathbf{x}'$ point with ever increasing radius such that the left side of the circle mapped towards the z-axis. In fact, starting with the definition of χ in eq. (2.16), one may derive that

$$(R - R'\chi)^2 + (z - z')^2 = R'^2(\chi^2 - 1), \tag{3.1}$$

which is the equation for a circle in the meridional plane! These circles are centered at the point $(R'\chi, z')$ with radius $R'\sqrt{\chi^2 - 1}$. When these circles are revolved around the z-axis, they describe circular tori.[1] Upon further verification (Morse & Feshbach 1953; Abramowitz & Stegun 1965), we found that the half-integer degree associated Legendre functions are called toroidal harmonics and they provide the principal set of basis functions in toroidal coordinates.

[1] We are grateful to Eric Barnes and Dana Browne for independently bringing to our attention the similarity between our plots of constant χ and the toroidal coordinate system.

Table 3.1: Axisymmeric Coordinate Systems

#	Coordinate System	Miller #
1	Cylindrical	2
2	Spherical	5
3	Prolate Spheroidal	6
4	Oblate Spheroidal	7
5	Parabolic	8
6	Lamé I (unnamed)	14
7	Lamé II (unnamed)	15
8	Lamé III (unnamed)	16
9	Toroidal	17

So, if the half-integer degree Legendre functions of the second kind are toroidal harmonics, we can ask ourselves, "What is the Green's function in toroidal coordinates and how does it compare to the compact cylindrical Green's function expansion?" Furthermore, the $\sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')}$ that appears in our compact cylindrical Green's function expansion appears as well in the Green's function for every other coordinate system that is axisymmetric and $\mathcal{R}$-separable for Laplace's equation. (For a list of most of the Green's functions involved, see chapter 10 of Morse & Feshbach 1953.) The nine coordinate systems that are both axisymmetric and are $\mathcal{R}-$separable for Laplace's equation are listed here in Table 3.1. Table 3.1 has three columns: in the first column, we number the nine coordinate systems; in the second column, we attempt to provide a familiar name for each coordinate system; and in the third column, we cross reference our coordinate system numbers with those listed in Miller (1977).

These nine axisymmetric coordinate systems are a subset of the total seventeen curvilinear orthogonal coordinate systems which are $\mathcal{R}$-separable for Laplace's equation. Bôcher (1894) provides the first complete description of all these coordinate systems. In a more recent discussion, Miller (1977) gives a complete geometrical description of these systems and shows that they can be classified in a number of ways. For instance, in the situation where the modulation factor, $\mathcal{R}$, is unity, there are eleven orthogonal curvilinear coordinate systems. These coordinate systems are all represented as confocal families of quadrics:

$$\frac{x^2}{a_1+\lambda}+\frac{y^2}{a_2+\lambda}+\frac{z^2}{a_3+\lambda}=1. \tag{3.2}$$

All of these coordinate surfaces are limiting cases of the confocal ellipsoidal coordinate surfaces, and the corresponding surfaces are ellipsoids, hyperboloids, and their various limits, such as paraboloids, spheres and planes. More generally, all seventeen coordinate systems may be described as orthogonal families of confocal cyclides, where a cyclide is a surface that satisfies the following equation:

$$a(x^2+y^2+z^2)^2+P(x,y,z)=0, \tag{3.3}$$

where a is a constant and P is any polynomial of order two. If $a=0$, the cyclide reduces to the already discussed eleven coordinate systems which have quadric coordinate surfaces. The remaining nonquadric coordinate systems are of the more general cyclidic form with $a\neq 0$. (For a detailed listing of these coordinate systems see Tables 14 & 17 in Miller 1977).

Now let us return to the nine axisymmetric coordinate systems in Table 3.1 for which Laplace's equation is $\mathcal{R}$-separable. Given in terms of a solution of Laplace's equation, Ψ, Miller (1977) shows that they correspond to the diagonalization of the operator

$$J_3 = x_2 \frac{\partial}{\partial x_1} - x_1 \frac{\partial}{\partial x_2}. \tag{3.4}$$

These special systems have the property that their eigenfunctions take the form,

$$\Psi(\mathbf{x}) = \Phi e^{im\phi}, \tag{3.5}$$

and,

$$iJ_3\Psi = m\Psi, \tag{3.6}$$

where Φ is a function of the remaining two variables. If we substitute this Ψ into Laplace's equation and factor out $e^{im\phi}$, we obtain a differential equation for $\Phi(R, z)$, which in cylindrical coordinates is

$$\frac{\partial^2\Phi}{\partial R^2} + \frac{1}{R}\frac{\partial\Phi}{\partial R} - \frac{m^2}{R^2}\Phi + \frac{\partial^2\Phi}{\partial z^2} = 0. \tag{3.7}$$

This expression for $m \geq 0$ is often referred to as the equation of generalized axisymmetric potential theory. It is clear that the compact cylindrical Green's function expansion we derived in chapter 2 must apply to all nine of these axisymmetric coordinate systems for which Laplace's equation is $\mathcal{R}$-separable through eq. (3.7).

In order to demonstrate this, we need to obtain the standard Green's function expansion for all of these coordinate systems. In our compact rep-

resentation, we express the cylindrical Green's function in terms of a single sum over the azimuthal quantum number, m. Having already described how this occurs in cylindrical coordinates we now show how this result applies to spherical coordinates.

3.1 A Compact Spherical Green's Function Expansion

Here we describe how the toroidal representation of the cylindrical Green's function may be extended to spherical coordinates. The transformation from Cartesian coordinates to spherical coordinates is given by

$$\begin{aligned} x &= r\sin\theta\cos\phi, \\ y &= r\sin\theta\sin\phi, \\ z &= r\cos\theta. \end{aligned} \tag{3.8}$$

The Green's function in spherical coordinates can be written as (cf., eq. [3.38] in Jackson 1975),

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \sum_{l=0}^{\infty} \frac{r_<^l}{r_>^{l+1}} P_l(\cos\gamma), \tag{3.9}$$

where

$$r_< \equiv \begin{cases} r' & \text{if } r' < r \\ r & \text{if } r < r' \end{cases} \quad \text{and} \quad r_> \equiv \begin{cases} r' & \text{if } r' > r \\ r & \text{if } r > r' \end{cases} \tag{3.10}$$

and $r_< = r_>$ if $r = r'$, P_l is the degree l Legendre function of the first kind (Legendre polynomial) with

$$\cos\gamma \equiv \cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\phi-\phi'). \tag{3.11}$$

The addition theorem for spherical harmonics states (cf., Jackson 1975)

$$P_l(\cos\gamma) = \frac{4\pi}{2l+1}\sum_{m=-l}^{l} Y^*_{lm}(\theta',\phi')Y_{lm}(\theta,\phi). \tag{3.12}$$

Now if we insert eq. (3.12) into (3.9) we obtain the familiar expression for the spherical Green's function given earlier by eq. (2.4). The spherical harmonics can be written in terms of the associated Legendre functions of the first kind as follows:

$$Y_{lm}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{\Gamma(l-m+1)}{\Gamma(l+m+1)}}P_l^m(\cos\theta)e^{im\phi}. \tag{3.13}$$

Furthermore, if we insert eq. (3.13) into eq. (2.4) and interchange the l and m sums,[2] we can rewrite eq. (2.4) as follows,

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')}\sum_{l=m}^{\infty}\frac{r_<^l}{r_>^{l+1}}\frac{\Gamma(l-m+1)}{\Gamma(l+m+1)}P_l^m(\cos\theta)P_l^m(\cos\theta'), \tag{3.14}$$

which can be rewritten as

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')}\sum_{l=0}^{\infty}\frac{r_<^{l+m}}{r_>^{l+m+1}}\frac{\Gamma(l+1)}{\Gamma(l+2m+1)}P_{l+m}^m(\cos\theta)P_{l+m}^m(\cos\theta'). \tag{3.15}$$

Now, from eq. (2.15) we are equally certain that

$$\frac{1}{|\mathbf{x}-\mathbf{x}'|} = \frac{1}{\pi\sqrt{rr'\sin\theta\sin\theta'}}\sum_{m=-\infty}^{\infty} e^{im(\phi-\phi')}\ Q_{m-\frac{1}{2}}(\chi), \tag{3.16}$$

where now in spherical coordinates,

[2] We are indebted to Prof. A. R. P. Rau for suggesting that we investigate what happens when the l and m sums are interchanged.

$$\chi = \frac{r^2 + r'^2 - 2rr'\cos\theta\cos\theta'}{2rr'\sin\theta\sin\theta'}. \tag{3.17}$$

Hence, by comparing eq. (3.15) and (3.16), we deduce that

$$\sum_{l=0}^{\infty} \frac{r_<^{l+m}}{r_>^{l+m+1}} \frac{\Gamma(l+1)}{\Gamma(l+2m+1)} P_{l+m}^m(\cos\theta) P_{l+m}^m(\cos\theta') = \frac{Q_{m-\frac{1}{2}}(\chi)}{\pi\sqrt{rr'\sin\theta\sin\theta'}}. \tag{3.18}$$

We offer this as a valid *second* addition theorem for spherical harmonics. This addition theorem, which using eq. (2.46) may also be written as,

$$\begin{aligned} Q_{m-\frac{1}{2}}(\cosh\xi) &= 4\pi^2\sqrt{rr'\sin\theta\sin\theta'}e^{-im(\phi-\phi')} \\ &\times \sum_{l=m}^{\infty} \frac{r_<^l}{r_>^{l+1}} \frac{1}{2l+1} Y_{lm}(\theta,\phi) Y_{lm}^*(\theta',\phi'), \end{aligned} \tag{3.19}$$

complements the familiar one given above as eq. (3.12) in that it provides a mechanism for collapsing the summation over l instead of m.

As a demonstration of the validity of this formula we now show that it is indeed correct in a certain limit. For $m = 0$ eq. (3.18) becomes,

$$\sum_{l=0}^{\infty} \frac{r_<^l}{r_>^{l+1}} P_l(\cos\theta) P_l(\cos\theta') = \frac{Q_{-\frac{1}{2}}(\chi)}{\pi\sqrt{rr'\sin\theta\sin\theta'}}. \tag{3.20}$$

Now if we further assume that both the primed and unprimed coordinates are located in the equatorial plane $z = z' = 0$, then $\cos\theta = \cos\theta' = 0$, $\sin\theta = \sin\theta' = 1$, and χ and μ become, respectively,

$$\chi = \frac{r^2 + r'^2}{2rr'}, \tag{3.21}$$

and

$$\mu = \frac{2\sqrt{rr'}}{r+r'}. \tag{3.22}$$

It can easily be shown that,

$$[P_l(0)]^2 = \frac{\cos^2[\frac{\pi l}{2}]}{\pi} \frac{\Gamma^2(\frac{1}{2}l+\frac{1}{2})}{\Gamma^2(\frac{1}{2}l+1)}. \tag{3.23}$$

Combining eqs. (3.20) through (3.23) with (2.22) and further assuming $r' < r$, we obtain

$$K\left[\frac{2\sqrt{rr'}}{r+r'}\right] = \frac{r+r'}{2} \sum_{l=0}^{\infty} \frac{r'^{2l}}{r^{2l+1}} \frac{\Gamma^2(\frac{2l+1}{2})}{\Gamma^2(l+1)}. \tag{3.24}$$

Now if we make the substitution $x = r'/r$, the argument of the complete elliptic integral of the first kind becomes $2\sqrt{x}/(1+x)$, and eq. (3.24) becomes

$$K\left[\frac{2\sqrt{x}}{1+x}\right] = \frac{1+x}{2} \sum_{l=0}^{\infty} x^{2l} \frac{\Gamma^2(\frac{2l+1}{2})}{\Gamma^2(l+1)}. \tag{3.25}$$

When compared to eqs. (8.113.1) and (8.126.3) in Gradshteyn & Ryzhik (1994), namely,

$$K(k) = \frac{\pi}{2}\left\{1 + \left(\frac{1}{2}\right)^2 k^2 + \left(\frac{1\cdot 3}{2\cdot 4}\right)^2 k^4 + \cdots + \left[\frac{(2n-1)!!}{2^n n!}\right]^2 k^{2n} + \cdots\right\}, \tag{3.26}$$

and

$$K\left[\frac{2\sqrt{x}}{1+x}\right] = (1+x)K(x), \tag{3.27}$$

the equality is demonstrated and therefore the validity of eq. (3.18) has been demonstrated, at least in one limit.

3.2 A Compact Toroidal Green's Function Expansion

Thus far we have demonstrated via eq. (2.14) that there is an integral expression relating the principal basis functions in cylindrical coordinates (Bessel functions) to the special functions $Q_{m-\frac{1}{2}}(\chi)$ and via eq. (3.18) a summation formula relating the principal basis functions in spherical coordinates (associated Legendre functions of the first kind) to the function $Q_{m-\frac{1}{2}}(\chi)$. It is now clear to us that similar integral or summation formula expressions can be obtained for all nine axisymmetric coordinate systems which are in Table 3.1. In order to complete such an analysis, one must have in hand the general Green's function expansion for each system. In Morse & Feshbach (1953), one may find the Green's function expansions for oblate spheroidal coordinates (eq. [10.3.63]), prolate spheroidal coordinates (eq. [10.3.53]) as well as for parabolic coordinates (eq. [10.3.68]). They also give an expression for the Green's function in toroidal coordinates (eq. [10.3.81]), but there appears to be a typo in this expression and, as yet, we have been unable to ascertain the true form of the Green's function in this crucial coordinate system for our study. Furthermore, we have not yet been able to find any reference which gives the Green's function expansion in the three axisymmetric Lamé coordinate systems, referred to in Table 3.1, in order to compare with the CCGF. In future investigations we plan to derive from first principles, all of the relevant Green's function expansions and thereby obtain what should prove to be new mathematical expressions relating all the basis functions involved to the half-integer degree Legendre function of the second kind.

4. Parallel Implementation of a Data-Transpose Technique for the Solution of Poisson's Equation in Cylindrical Coordinates

Here we describe our numerical implementation of an efficient scheme to solve Poisson's equation numerically on massively parallel architectures. The groundwork on serial algorithms for solving Poisson's equation is extensive. In particular, for some time, extremely efficient methods have been known for solving the set of sparse matrices that result from a second-order accurate finite-differencing of the Poisson equation in cylindrical coordinates given the boundary solution. In Cartesian coordinates there has been a large successful effort in order to find accurate and highly parallel methods for solving Poisson's equation (i.e. Fast Poisson solver using Fourier methods). The situation is not so simple in cylindrical coordinates, however. Due to the non-constant variation of the matrix elements that result from the finite-discretization of the cylindrical Poisson equation, direct Fourier methods are not possible. It is only in the naturally periodic azimuthal coordinate direction that one can take advantage of this technique which reduces the complexity of the problem, in terms of coupled dimensions, from three-dimensions to two-dimensions. Techniques like Buneman cyclic reduction (Swarztrauber 1977) provide the direct solution of the resulting set of two-dimensional problems in an extremely accurate fashion; other direct techniques aren't even so efficient when implemented in serial. When one asks the question of how to solve these problems in parallel one quickly sees that the global nature of the two-dimensional solution methods are very difficult to implement in parallel and do not result in a load-balanced solution of the matrix problem. It is here

that we present the Fourier-ADI method, which is iterative, although very accurate, and takes advantage of the highly parallel data-transpose technique. In this computational strategy all computations are performed without communication, and all communications are restricted to highly parallel, global three-dimensional data-transpositions. We describe in detail how this algorithm is implemented and give a theoretical operation count which demonstrates the highly parallel nature of this algorithm. It is the Fourier-ADI technique, combined with the CCGF technique for evaluating the boundary potential that yields an extremely efficient and accurate potential solver.

We have adopted a finite-difference method for the solution of the cylindrical Poisson equation. In iterative schemes, the solution of a partial differential equation (PDE) is obtained by starting with an initial guess and then proceeding iteratively until the solution is obtained to within desired accuracy. In direct schemes, the solution of the PDE is obtained by direct numerical solution of the finite-difference equations. Direct methods are usually preferred over iterative methods since they are guaranteed to generate an exact solution.

Here we utilize both a fast direct method and an iterative method to solve the problem at hand. Both methods are implemented in parallel using a data-transpose technique. The data-transpose technique is a parallelization strategy in which all communication is restricted to global 3D data-transposition operations and all computations are subsequently performed with perfect load balance and zero communication.

In the remainder of this chapter, we present a detailed description of our parallel algorithm. In §4.1, we describe the sequential algorithms that we use in conjunction with the data-transpose technique in order to solve the cylindrical Poisson equation. In §4.2, we describe the parallel data-transpose technique, how it applies to the two sequential algorithms we presented in section §4.1, and a theoretical description of how the technique can be applied to a 2D mesh topology.

4.1 Sequential Algorithms

We present a parallel method to solve Poisson's equation

$$\nabla^2 \Phi = 4\pi G\rho, \tag{4.1}$$

where ∇^2 denotes the Laplacian operator in three dimensions, Φ is the scalar Newtonian potential, G is the gravitational constant, and ρ is the mass-density scalar distribution function.

In a cylindrical geometry, the Cartesian vertical coordinate, z, remains unchanged, but the Cartesian x and y coordinates are replaced by the polar coordinates R (radial) and ϕ (azimuthal) via the transformation

$$\begin{aligned} x &= R\cos\phi, \\ y &= R\sin\phi. \end{aligned} \tag{4.2}$$

Our domain boundaries are specified by the conditions

$$0 \leq R \leq R_B,$$

$$0 \leq \phi \leq 2\pi, \tag{4.3}$$
$$|z| \leq z_B,$$

where R_B is the radius of the cylindrical domain, $2z_B$ is the height of the cylinder, and the angle ϕ is measured in radians

We perform a spatial discretization of our domain using the indices (j, k, l) to refer to the (R_j, z_k, ϕ_l) positions of each cell center with $\Delta R_j, \Delta z_k$, and $\Delta\phi_l$ being the radial, vertical and azimuthal grid spacing of each cell. Both the potential and the source function are evaluated at the center of each grid cell. As in chapter 2, the 3D discretization is performed using J radial zones, K vertical zones, and L azimuthal zones, but here we are concerned about the solution of eq. (4.1) throughout the interior volume of the grid rather than just on its boundary surface.

In our applications, mixed Dirichlet, Neumann, and periodic boundary conditions are usually assumed for the potential Φ, viz.

$$\begin{aligned} \Phi(R_B, \phi, z) &= g(\phi, z), \\ \Phi(R, \phi, +z_B) &= h_+(R, \phi), \\ \Phi(R, \phi, -z_B) &= h_-(R, \phi), \\ \Phi(R, 2\pi, z_B) &= \Phi(R, 0, z), \end{aligned} \tag{4.4}$$

where the functions g, h_+, and h_- are the boundary potentials computed using the CCGF technique described in chapter 2. The interior of our domain is mapped onto a rectangular computational grid which extends between $2 \leq j \leq J$, $2 \leq k \leq K$, and $1 \leq l \leq L$. Since the azimuthal coordinate

direction is periodic in nature, the index l runs modulo L, i.e. $l + L = l$. We also require continuity of the solution across the z axis, this is evaluated at $j = 1$ by setting $\Phi(1, k, l) = \Phi(2, k, l + L/2)$. The boundary values g, h_- and h_+ are evaluated at $j = J + 1, k = 1$ and $k = K + 1$, respectively.

4.1.1 Finite-Difference Derivation of the Equation of Generalized Axisymmetric Potential Theory

Written in cylindrical coordinates Poisson's equation reads

$$\frac{1}{R}\frac{\partial}{\partial R}(R\frac{\partial \Phi}{\partial R}) + \frac{1}{R^2}\frac{\partial^2 \Phi}{\partial \phi^2} + \frac{\partial^2 \Phi}{\partial z^2} = 4\pi G\rho(R, \phi, z). \tag{4.5}$$

The finite-difference representation of eq. (4.5) is

$$\frac{1}{R_j \Delta R_j}\Big(R_{j+1/2}\frac{\Delta_{j+1/2}\Phi}{\Delta R_{j+1/2}} - R_{j-1/2}\frac{\Delta_{j-1/2}\Phi}{\Delta R_{j-1/2}}\Big) + \frac{1}{R_j^2}\frac{\Delta_l^2 \Phi}{(\Delta\phi_l)^2} + \frac{\Delta_k^2 \Phi}{(\Delta z_k)^2} = 4\pi G\rho_{j,k,l}, \tag{4.6}$$

where, for every index i, the symbols Δ_i and Δ_i^2 indicate the sense that the first and second differences are taken and their proper centering. For example, $\Delta_{j+1/2}$ denotes that the first difference is taken in the radial coordinate direction and is centered at the $(j + 1/2, k, l)$ location, while Δ_l^2 denotes that the second difference is taken in the azimuthal coordinate direction and is centered at the (j, k, l) location. Similarly $\Delta R_j = R_{j+1/2} - R_{j-1/2}$, $\Delta R_{j+1/2} = R_{j+1} - R_j$, $\Delta\phi_l = \phi_{l+1/2} - \phi_{l-1/2}$, and $\Delta z_k = z_{k+1/2} - z_{k-1/2}$. Expanding the differences Δ and Δ^2 up to second order in the grid spacings $\Delta R_j, \Delta z_k$, and $\Delta\phi_l$, one obtains

$$A(j)\ \Phi(j + 1, k, l) + B(j)\ \Phi(j - 1, k, l)$$

$$
\begin{aligned}
&+ C(j,l)\,\Big[\Phi(j,k,l+1) + \Phi(j,k,l-1)\Big] \\
&+ D(k)\,\Big[\Phi(j,k+1,l) + \Phi(j,k-1,l)\Big] \\
&-\Big\{A(j) + B(j) + 2[C(j,l) + D(k)]\Big\}\,\Phi(j,k,l) = 4\pi G\rho(j,k,l).
\end{aligned}
\tag{4.7}
$$

The coefficients in eq. (4.7) are defined by the expressions

$$
\begin{aligned}
A(j) &= R_{j+1/2}(R_j \Delta R_{j+1/2} \Delta R_j)^{-1}, \\
B(j) &= R_{j-1/2}(R_j \Delta R_{j-1/2} \Delta R_j)^{-1}, \\
C(j,l) &= (R_j \Delta\phi_l)^{-2}, \\
D(k) &= (\Delta z_k)^{-2}.
\end{aligned}
\tag{4.8}
$$

The 3D problem represented by eq. (4.7) can be decoupled into a set of independent 2D problems, in a nalogy to eq. (2.21) from chapter 2, by performing a discrete Fourier transform in the azimuthal coordinate direction of the general form

$$
Q(j,k,l) = \sum_{m=0}^{L/2} \Big\{ Q^1_m(j,k)\cos(m\phi_l) + Q^2_m(j,k)\sin(m\phi_l) \Big\},
\tag{4.9}
$$

where Q denotes either Φ or ρ, and Q^i_m, with $i = 1$ and $i = 2$, are the Fourier coefficients of the cosine and the sine terms, respectively. Substituting eq. (4.9) into eq. (4.7), assuming a constant value of $\Delta\phi_l \equiv \Delta\phi = 2\pi/L$, and accounting for the continuity boundary condition across the z-axis, one obtains

$$
A(j)\;\phi^i_m(j+1,k) + B(j)\;\Big\{1 + \delta_{j2}[(-1)^m - 1]\Big\}\phi^i_m(j-1+\delta_{j2},k)
$$

$$+D(k)\ \left[\phi^i_m(j,k+1)+\phi^i_m(j,k-1)\right] \tag{4.10}$$
$$+\Big\{2(\lambda_m-1)C(j)-[A(j)+B(j)+2D(k)]\Big\}\ \phi^i_m(j,k)=4\pi G\rho^i_m(j,k),$$

where δ is the Kronecker symbol,

$$m=0,1,2,\ldots,L/2 \qquad (\text{ for } \ i=1\ ), \tag{4.11}$$
$$m=1,2,3,\ldots,L/2-1 \qquad (\text{ for } \ i=2\ ),$$

and $\lambda_m \equiv \cos(m\Delta\phi)$. The equations for $i=1$ are derived by equating coefficients of the cosine terms in the Fourier expansion and the equations for $i=2$ are derived by equating coefficients of the sine terms. Eq. (4.10) is the finite-difference representation of the equation of generalized axisymmetric potential theory (eq. [3.7]).

4.1.2 Solution Methods for the Equation of Generalized Axisymmetric Potential Theory

4.1.2.1 The ADI Method

The alternating direction implicit (ADI) (Peaceman & Rachford 1955; Strikwerda 1989) method is a widely used iterative method for solving multi-dimensional boundary value problems. It is an operator-splitting scheme which solves implicitly, and in an alternating fashion, each of the dimensions of a multi-dimensional elliptic problem. It combines two ideas, described below, and results in a rapidly convergent and numerically stable algorithm. (See Press et al. 1992, Black & Bodenheimer 1975 and references given therein for implementations of the same technique to the solution of various PDEs.)

The first idea of ADI is to write the original operator equation in the form of a diffusion equation, viz.

$$\frac{\partial \Phi}{\partial t} = \nabla^2 \Phi - 4\pi G\rho. \tag{4.12}$$

The diffusion equation uses a false dimensionless time which helps the algorithm settle into a final steady-state solution of eq. (4.1). We have adopted the prescription proposed by Black and Bodenheimer (1975) to compute the "time steps" for a variable number of iterations. The second idea incorporated in the ADI technique is to implement a partially implicit solution of the 2D finite-difference equations. This is performed by splitting the terms of the 2D equations in such a way that, at each step, the finite-difference terms in two given directions are treated as known and unknown, respectively. When this is done, each 2D equation transforms into a set of tridiagonal equations. We then use the optimal sequential tridiagonal solver, LU (Lower-Upper) decomposition with forward- and back-substitution (hereafter referred to as just LU decomposition) to solve each tridiagonal matrix (Press et al. 1992).

In our specific implementation, the spatial operator in the generalized axisymmetric potential theory (eq. [4.10]) along with the diffusion term in eq. (4.12) is split as follows. For each choice of the Fourier mode elements i and m, during the R-sweep we use

$$\begin{aligned} A(j)\ \phi^{n+1/2}(j+1,k) + B(j)\Big\{1+\delta_{j2}[(-1)^m-1]\Big\}\phi^{n+1/2}(j-1+\delta_{j2},k) \\ -\Big[A(j)+B(j)-2(\lambda-1)C(j)+\tfrac{2}{\Delta t}\Big]\phi^{n+1/2}(j,k) = \\ 4\pi G\rho^n(j,k) - D(k)\ \Big[\phi^n(j,k+1)+\phi^n(j,k-1)\Big] \end{aligned} \tag{4.13}$$

$$+(2D(k) - \tfrac{2}{\Delta t})\ \phi^n(j,k);$$

and during the z-sweep we use

$$\begin{aligned} D(k)\ \Big[\phi^{n+1}(j,k+1) + \phi^{n+1}(j,k-1)\Big] - (2D(k) + \tfrac{2}{\Delta t})\phi^{n+1}(j,k) = \\ 4\pi G\rho^{n+1/2}(j,k) - A(j)\ \phi^{n+1/2}(j+1,k) \\ -B(j)\ \Big\{1 + \delta_{j2}[(-1)^m - 1]\Big\}\phi^{n+1/2}(j-1+\delta_{j2},k) \\ -\Big[2(\lambda-1)C(j) - (A(j) + B(j) - \tfrac{2}{\Delta t})\Big]\ \phi^{n+1/2}(j,k). \end{aligned} \tag{4.14}$$

In both cases, as indicated by the "time step" superscripts n, $n + 1/2$, or $n + 1$, terms on the right-hand side of the expressions are considered known quantities, and terms on the left are considered unknown.

4.1.2.2 Fourier Analysis

One popular method for the solution of Poisson's equation is to use Fourier Analysis (Hockney 1965) in order to convert the 3D problem into a set of completely decoupled 1D problems. This method is highly efficient and takes advantage of the fast Fourier transform (FFT) algorithm. The resulting tridiagonal system can then be solved directly using LU decomposition.

We usually assume Dirichlet-Dirichlet (DD) boundary conditions in the vertical coordinate direction. This boundary condition can be accommodated by applying a *sine* transform in the vertical coordinate direction. In fact, any combination of Neumann and Dirichlet boundary conditions can be dealt with by using the appropriate discrete transform. (Cooley et al. 1970; Swarztrauber 1974; Swarztrauber 1977) For instance, in the case

of Neumann-Dirichlet (ND) boundary conditions, a discrete quarter *cosine* transform would accomplish the decoupling. As described by Cooley et al. (1970), all of the possible combinations can be obtained using certain appropriate pre- and post-conditioning operations on the input and output of a standard FFT routine.

When the appropriate transform is substituted into eq. (4.10) and uniform zoning is utilized (i.e. $\Delta z_k \equiv \Delta z = constant$), one obtains

$$A(j)\ \phi^i_{m,k'}(j+1) + B(j)\ \Big\{1 + \delta_{j2}[(-1)^m - 1]\Big\}\phi^i_{m,k'}(j-1+\delta_{j2}) \tag{4.15}$$
$$+\Big\{2(\lambda_m - 1)C(j) + 2(\lambda_{k'} - 1)D - [A(j) + B(j)]\Big\}\ \phi^i_{m,k'}(j) = 4\pi G\rho^i_{m,k'}(j),$$

where $\lambda_{k'}$ depends on the specific Fourier basis transform. Once the solution to this tridiagonal system is obtained via LU decomposition, the appropriate inverse transform is then applied in the vertical coordinate direction followed by an inverse Fourier transform in the azimuthal coordinate direction in order to bring the solution back into coordinate space.

4.2 Parallel Data-Transpose Technique

A large effort has gone into the development of fast *sequential* algorithms for the solution of Poisson's equation (see Press et al. 1992 for many examples). On shared-memory parallel computing architectures, the sequential algorithms with the lowest operation count are optimal, given a way to distribute the computations uniformly among the processors (Briggs 1990). Similarly, a large effort has gone into the development of fast parallel algorithms for the solution of the Cartesian Poisson equation on distributed-memory architectures (cf., Kumar et al. 1994; Schwardmann 1993). We

adopt a strategy for the parallel solution of the cylindrical Poisson equation on a distributed-memory architecture that has perfect computational load balance.

If a sequential algorithm requires a recursive sweep in one coordinate direction, then this sweep can be performed at each of the other coordinate locations independently. If we distribute our data in such a way that the coordinate direction in which the sweep needs to be performed is stored in the internal memory of each processor (node), then the computation can be performed in parallel on each of the nodes. Since each recursive sweep is performed completely in memory, no communication is required in the part of the calculation. The next sweep that needs to be performed is distributed across a set of processors. If we perform this recursive sweep with no change in the data distribution, then inter-processor communication will be required in order to perform the sweep, with the recursive nature of the sweep leading to poor load balance. Another choice that we have is to perform a global data-transposition operation on the storage array so that the second sweep direction is redistributed into the internal memory of each node. The question of which choice is optimal is architecture dependent.

On a 2D mesh of processors, the most natural way to map a 3D array is to spread two of the array directions out across the processors (X and Y processor grid directions) and to store the third array direction in the internal memory of each node (M). If we perform the data-transposition operation between the X processor grid direction and M, then the global data-transpose can be performed in parallel for each Y processor grid location. Similarly, if

we perform the data-transposition operation between the Y processor grid direction and M, then the global data-transpose operation can be performed in parallel for each X processor grid location (Choi & Walker 1995). We have performed a data-transposition operation between each computational sweep for the preceding two algorithms.

4.3 Analysis

4.3.1 Theoretical Timing Analysis

Since the data-transpose technique allows both sequential algorithms to be implemented, in parallel, with no change in the computational strategy, the sequential operation count gives us a partial measure of the parallel execution time. A more accurate model for the total parallel execution time must include the data-transpose operations in the analysis. In both algorithms, a forward and inverse FFT is applied in the azimuthal coordinate direction. Including the highest order terms, the sequential operation count for a length p real FFT is $\frac{1}{2}(5p\log_2(p)+13p)$ (Swarztrauber 1977). Similarly, the operation count for a length p tridiagonal solution using LU decomposition is $5p$ (Press et al. 1992). The sequential operation count for the real fast *sine* transform is $\frac{1}{2}(5p\log_2(p)+22p)$ (Cooley et al. 1970). In order to simplify the calculation, we assume equal number of grid points in all three coordinate directions, i.e. $N = J = K = L$. Therefore, the total sequential operation count for ADI is $N^2[5N\log_2(N)+13N+10NI]$, where I is the total number of iterations needed to converge to a solution. As an example, the total sequential operation count for Fourier Analysis applied to the

fast *sine* transform method for the Dirichlet-Dirichlet boundary condition is $N^2[10N\log_2(N) + 40N]$.

Given these sequential operation counts we can then estimate the parallel execution time. In the ADI algorithm, the data-transpose function is called $3 + 2I$ times and in the Fourier analysis algorithm it is called 4 times. In the case where each data-element is mapped to a single processor, the parallel operation count will be equal to the sequential operation count, reduced by a factor of N^2. Including this fact and including the amount of time spent in the data-transpose function, we estimate the parallel execution time for the ADI algorithm to be

$$t_{ADI} = [5N\log_2(N) + 13N + 10NI]C + (3 + 2I)t_{TRAN}, \tag{4.16}$$

and we estimate the parallel execution time for the Fourier analysis algorithm to be

$$t_{FA} = [10N\log_2(N) + 40N]C + 4t_{TRAN}, \tag{4.17}$$

where C is a constant that determines how much time is spent on average per operation count and t_{TRAN} is the amount of time that is takes to complete a single data-transpose operation.

We have thus demonstrated that the parallel efficiency of the Fourier-ADI technique is on the same order as the Fourier analysis technique, which is the technique that has been broadly adopted for Cartesian problems. The Fourier-ADI technique has a remarkably short execution time, and the data-transpositions that are required to make it function efficiently on distributed

memory, parallel computers prove not to be the main bottleneck in this solution methodology. This has become our computational strategy of choice for solving the Poisson equation on massively parallel computers.

5. Conclusion

In trying to build a mathematical model that accurately describes the observed appearance or behavior of a complex physical system, what often distinguishes the task of an astronomer from that of a physicist is the need to accurately determine in a self-consistent way the time-varying gravitational field that is associated with the system. The long-range influence of the gravitational field makes this a nontrivial task in virtually all situations; and for all but the simplest systems, a determination of the gravitational potential that is consistent with a given mass distribution can only be achieved with numerical rather than analytical tools. Rather than studying in depth the behavior of one particular type of astrophysical system, the objective of this dissertation has been to identify and implement techniques that can be used to accurately and efficiently determine the gravitational potential of arbitrarily complex systems. In this way we hope to facilitate and indeed accelerate modeling efforts in a variety of important areas. Our focus has been on the numerical solution of the Poisson equation in cylindrical coordinates because a very large number of interesting astrophysical mass distributions are particularly well suited to such a coordinate description. But the techniques that we have developed are fairly readily adaptable to other orthogonal curvilinear coordinate systems and, particularly in connection with our study of Green's function expansions, will almost certainly be useful in analytical studies of related, but less complex systems.

The numerical simulations that we perform are guided by previous attempts at these problems (see for example, Black & Bodenheimer 1975; Miller & Smith 1979). These efforts were performed on the most powerful computers of their day. Even then the need for efficiency was paramount; only the most efficient algorithms could tackle the largest of the small-scale problems computable at that time. We recognize the limitation that a single processor imposes for computing time-dependent, large-scale, three-dimensional astrophysically interesting problems. This limitation has forced us to start developing and implementing our algorithms on massively parallel computing architectures. Only through the usage of a large number of processors, each with its own local memory, may these types of large-scale problems presently be solved. The theorist, while always searching for analytically soluble solutions, is now driven towards computationalist strategies. In the parallel programming paradigm, computation (cpu time) and communication (I/O time) combine to further complicate the already difficult task of implementing an efficient algorithm.

Local problems, such as the numerical solution of the Navier-Stokes equations, prove to be extremely efficient when implemented in parallel. Global problems, however, such as the parallel solution of Poisson's equation, have proven to be more difficult to implement efficiently in parallel. Global problems require more sophisticated communication strategies. For global problems, we must match the most stream-lined computation algorithm with the most effective communication strategy in order to minimize the compute time.

The traditional serial algorithms that have frequently been used in the past to numerically obtain the Newtonian potential are not easily parallelizable. We came to the conclusion that a more beneficial parallelization technique for the Poisson solver was to use the Fourier-ADI technique, described here in detail in chapter 4. This powerful parallel technique relies on fast networks that can perform global 3D data-transpositions quite efficiently. Once we were satisfied with the performance of our parallel Poisson solver, which was first implemented on LSU's 8192 node MasPar MP-1, we then set out to parallelize our boundary solver.

It also has been clear for some time that the traditional multipole method does not conform well to a cylindrical coordinate grid. Its implementation requires that all the mass be contained within a spherical radius vector extending from the origin to each boundary location. Hence, in order to accurately and efficiently compute the boundary potential, we were forced to place the boundary far from the mass distribution. This was particularly problematic when encountering highly flattened mass distributions, since we then had to compute the gravitational potential throughout a grid that contained great amounts of empty space. We then set out to find a more accurate algorithm for the boundary solver which would hopefully conform better to our cylindrical mesh. We examined the possibility of computing the Newtonian potential using a cylindrical Green's function. The cylindrical Green's function, which is discussed in a variety of textbooks (e.g., Jackson 1975; Morse & Feshbach 1953) was expressible in terms of certain special functions, i.e. Bessel functions and exponential functions pieced together with an infinite

integral over a continuous wave number and an infinite summation over the azimuthal quantum number, m. The infinite integral posed a serious impediment to the numerical implementation of the cylindrical Green's function technique. Initially, our attempt to use the cylindrical Green's function to determine the Newtonian potential was hindered by our inability to find an efficient numerical method to compute the infinite integrals involved. We were able to obviate this problem through the propitious discovery of an analytical solution to the integral, which led us to the compact cylindrical Green's function (CCGF) expansion presented in detail here in chapter 2.

The CCGF is expressible as a single sum over the azimuthal quantum number, m and is written in terms of half-integer degree Legendre functions of the second kind. These functions, which are also commonly referred to as toroidal harmonics, are known to be the set of basis functions which separate Laplace's equation in toroidal coordinates. In chapter 3 we have outlined how the toroidal harmonics may be used to secure similarly compact Green's function expansions in other coordinate systems. This has led, in particular, to the indentification of a *second* useful addition theorem in spherical coordinates.

The successful numerical implementation of the CCGF algorithm in parallel, combined with the massively parallel data-transpose Fourier-ADI method for solving Poisson's equation in cylindrical coordinates, has proven to be an effective tool for computing the Newtonian potential for arbitrarily complex, isolated mass distributions. In connection with a broad range of astrophysically interesting, two- and three-dimensional fluid flow problems, we have

found this general tool to be not only more accurate but also less expensive to implement than more traditional methods. We strongly encourage the adoption of these techniques by other groups who are attempting to study large, complex, time-evolving astrophysical systems. In an effort to hasten the adoption of these techniques. we have included in appendices C, D, E, & F, the potential solver code, or respectively, the HPF code, the f77 code, grid.h, and the Makefile we use to compile the code on the MIMD Cray T3E.

References

Abramowitz, M. & Stegun, I.A. 1965, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (New York: Dover)

Arfken, G. 1985, Mathematical Methods for Physicists (New York: Academic Press)

Barnes, J., & Hut, P. 1986, Nature, 324, 446

Binney, J., & Tremaine, S. 1987, Galactic Dynamics (Princeton: Princeton University Press)

Black, D. & Bodenheimer, P. 1975, ApJ, 199, 619

Bôcher, M. 1894, Die Reihenentwickelungen der Potentialtheorie (Leigzig)

Boss, A. P. 1980, ApJ, 242, 699

———. 1993, ApJ, 410, 157

———. 1998a, ApJL, 501, L77

———. 1998b, ApJ, 503, 923

Boss, A. P. & Myhill, E.A. 1995, ApJ, 451, 218

Briggs, F. H. 1990, ApJ, 352, 15

Cazes, J. E. 1999, Ph.D. thesis, Louisiana State University and A&M College, Baton Rouge

Chandrasekhar, S. 1969, Ellipsoidal Figures of Equilibrium (New York: Dover Publications, Inc.)

Choi J.D., & Walker D. 1995, Parallel Computing, 21, 1387

Cooley J.W., Tookey, P.L., & Welch, P. 1970, J. Sounds Vib., 12, 315

de Zeeuw, T. 1985, MNRAS, 216, 273

Earn, D.J.D. 1996, ApJ, 465, 91

Evans, N.W. & de Zeeuw, P.T. 1992, MNRAS, 257, 152

Gradshteyn, I.S. & Ryzhik, I.M. 1994, Table of Integrals, Series, and Products (New York: Academic Press)

Hachisu, I. 1986, ApJS, 62, 461

Hayashi, A., Eriguchi, Y., & Hashimoto, M. 1998, ApJ, 492, 286

Hockney, R. 1965, J. ACM, 12, 95

Jackson, J.D. 1975, Classical Electrodynamics (New York: John Wiley & Sons)

Kalnajs, A. 1971, ApJ, 166, 275

Kumar, V., Grama A., Gupta A., & Karypis G. 1994, Introduction to Parallel Computing, Design and Analysis of Algorithms. (The Benjamin/Cummings Publishing Company, Inc.)

Landau, L.D., & Lifshitz, E.M. 1960, Electrodynamics of Continuous Media (Oxford: Pergamon Press)

Miller, W., Jr. 1977, Symmetry and Separation of Variables (London: Addison-Wesley Publishing Company)

Miller, R. H. & Smith, B. F. 1979, ApJ, 227, 407

Morse, P., & Feshbach, H. 1953, Methods of Theoretical Physics (New York: McGraw-Hill Book Company)

Motl, P., Frank, J., & Tohline, J., 1999, (abstract) in 194th AAS meeting, Chicago, IL

Müller, E. & Steinmetz, M. 1995, Comput. Phys. Commun., 89, 45

New, K. B. & Tohline, J. E. 1997, ApJ, 490, 311

Norman, M., & Wilson, J. R. 1978, ApJ, 224, 497

Peaceman, D. & Rachford, J.H. 1955, J. Soc. Indust. Appl. Math., 3, 28

Pickett, B. K., Cassen P., Durisen, R. H., & Link R. 1998, ApJ, 504, 468

Pickett, B. K., Durisen R. H., & Davis, G. A. 1996, ApJ, 458, 714

Press W., Flannery, B., Teukolsky, S., & Vetterling W. 1992, Numerical Recipes in FORTRAN, The Art of Scientific Computing, (Cambridge University Press) Second Edition.

Ramsey, A.S. 1981, Newtonian Attraction (Cambridge: Cambridge Univ. Press)

Robijn, F.H.A. & Earn, D.J.D. 1996, MNRAS, 282, 1129

Schwardmann, U., 1993, Supercomputer, 55, 4, 1993.

Stone, J. M., & Norman, M. L. 1992, ApJS, 80, 753

Strikwerda, J.C. 1989, Finite Difference Schemes and Partial Differential Equations, Mathematics Series, (Wadsworth & Brooks/Cole)

Swarztrauber, P. 1974, SIAM, J. Numer. Anal., 11, 1136

Swarztrauber, P. 1977, SIAM Review, 19, 490

Tohline, J.E. 1980, ApJ, 235, 866

Toman, J., Imamura, J. N., Pickett, B. K., & Durisen, R. H. 1998, ApJ, 497, 70

Truelove, J. K., Klein, R. I., McKee, C. F., Holliman, J. H. II., Howell, L. H., & Greenough, J. A. 1997, ApJL, 489, L179

United States. National Bureau of Standards. Computation Laboratory 1945, Tables of Associated Legendre Functions (New York: Columbia Univ. Press)

Villumsen, J.V. 1985, ApJ, 290, 75

Watson, G.N. 1944, A Treatise on the Theory of Bessel Functions (Cambridge: Cambridge Univ. Press)

Yorke, W.H., & Kaisig, M. 1995, Comput. Phys. Commun., 89, 29

Yoshida, S. & Eriguchi, Y. 1995, ApJ, 438, 830

Appendix A: A Useful Modal Expansion

Morse & Feshbach (1953; see the expression just above their eq. 10.3.79) have presented the following useful relationship in connection with the integral representation of $Q_{m-\frac{1}{2}}$:

$$Q_{m-\frac{1}{2}}(\cosh\mu) = \frac{1}{2\sqrt{2}} \int_0^{2\pi} \frac{\cos(m\phi')d\phi'}{\sqrt{\cosh\mu - \cos\phi'}}. \tag{A.1}$$

Multiplying both sides of this expression by $e^{im\phi}$ and then summing both sides from $m = -\infty$ to $m = \infty$, yields the following expression,

$$\sum_{m=-\infty}^{\infty} e^{im\phi} Q_{m-\frac{1}{2}}(\cosh\mu) = \frac{1}{2\sqrt{2}} \int_0^{2\pi} \frac{d\phi'}{\sqrt{\cosh\mu - \cos\phi'}} \times \sum_{m=-\infty}^{\infty} \left[e^{im(\phi+\phi')} + e^{im(\phi-\phi')} \right]. \tag{A.2}$$

Utilizing the following representation of the Dirac delta function (cf., eq. [3.139] of Jackson 1975)

$$\delta(\Theta) = \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} e^{im\Theta}, \tag{A.3}$$

the integral on the right-hand-side of eq. (A.2) can be readily performed, giving

$$\frac{1}{\sqrt{\cosh\mu - \cos\phi}} = \frac{\sqrt{2}}{\pi} \sum_{m=-\infty}^{\infty} e^{im\phi} Q_{m-\frac{1}{2}}(\cosh\mu), \tag{A.4}$$

or, written entirely in terms of real functions,

$$\frac{1}{\sqrt{\cosh\mu - \cos\phi}} = \frac{\sqrt{2}}{\pi} \sum_{m=0}^{\infty} \epsilon_m \cos(m\phi)\, Q_{m-\frac{1}{2}}(\cosh\mu). \tag{A.5}$$

Appendix B: Selected Analytical Potential-Density Pairs

For a distinguished class of nonspherical objects there exist analytical solutions for the Newtonian potential given in terms of elementary and special functions. Below we list a few of these distinguished objects and the analytical forms of the exterior potential associated with them. There is a long history associated with these problems (cf., Ramsey 1981 and Binney & Tremaine 1987). Unfortunately, these objects do not represent most of the types of objects for which one might need to calculate gravitational forces. Even so, they are certainly very useful in comparing numerical methods for evaluating potentials.

Consider an homogeneous, axisymmetric spheroid defined such that,

$$\rho(R,z) = \begin{cases} \rho_0 & \text{if } R^2/a_1^2 + z^2/a_3^2 \leq 1 \\ 0 & \text{if } R^2/a_1^2 + z^2/a_3^2 > 1 \end{cases} \tag{B.1}$$

where a_1 and a_3 are the equatorial and polar radii of the spheroid, respectively. From Chandrasekhar (1969), we find that the gravitational potential exterior to the spheroid is,

$$\begin{aligned} \Phi(R,z) &= \pi G \rho_0 a_1^2 a_3 \Bigg\{ \left(1 + \frac{R^2}{2(a_3^2 - a_1^2)} - \frac{z^2}{a_3^2 - a_1^2}\right) I_1 - \frac{R^2 \sqrt{a_3^2 + \lambda}}{(a_3^2 - a_1^2)(a_1^2 + \lambda)} \\ &- \frac{2z^2}{(a_1^2 + \lambda)\sqrt{a_3^2 + \lambda}} - \frac{2z^2 \sqrt{a_3^2 + \lambda}}{(a_3^2 - a_1^2)(a_1^2 + \lambda)} \Bigg\} \end{aligned} \tag{B.2}$$

where

$$\lambda = [(R^2 + z^2 - a_1^2 - a_3^2) + \sqrt{(a_1^2 + a_3^2 - R^2 - z^2)^2 - 4(a_1^2 a_3^2 - R^2 a_3^2 - z^2 a_1^2)}]/2, \tag{B.3}$$

and, for an oblate spheroid ($a_1 > a_3$),

$$I_1 = \frac{\pi}{\sqrt{a_1^2 - a_3^2}} - \frac{2}{\sqrt{a_1^2 - a_3^2}} \tan^{-1} \sqrt{\frac{a_3^2 + \lambda}{a_1^2 - a_3^2}}, \tag{B.4}$$

whereas for a prolate spheroid ($a_1 < a_3$),

$$I_1 = \frac{-1}{\sqrt{a_3^2 - a_1^2}} \ln\left[\frac{(\sqrt{a_3^2 + \lambda} - \sqrt{a_3^2 - a_1^2})^2}{a_1^2 + \lambda}\right]. \tag{B.5}$$

For an homogeneous, triaxial ellipsoid defined such that,

$$\rho(x, y, z) = \begin{cases} \rho_0 & \text{if } x^2/a_1^2 + y^2/a_2^2 + z^2/a_3^2 \leq 1 \\ 0 & \text{if } x^2/a_1^2 + y^2/a_2^2 + z^2/a_3^2 > 1 \end{cases} \tag{B.6}$$

where the three principal axes are defined such that $a_1 > a_2 > a_3$, the potential at any point $\mathbf{x} = (x, y, z)$ exterior to the ellipsoid is,

$$\begin{aligned} \Phi(\mathbf{x}) &= \frac{2\pi\rho_0 a_1 a_2 a_3}{\sqrt{a_1^2 - a_3^2}} \left[\left(1 - \frac{x^2}{a_1^2 - a_2^2} + \frac{y^2}{a_1^2 - a_2^2}\right) F(\theta, k) \right. \\ &+ \left. \left(\frac{x^2}{a_1^2 - a_2^2} - \frac{(a_1^2 - a_3^2) y^2}{(a_1^2 - a_2^2)(a_2^2 - a_3^2)} + \frac{z^2}{a_2^2 - a_3^2}\right) E(\theta, k) \right. \end{aligned} \tag{B.7}$$

$$+ \quad \left(\frac{a_3^2+\lambda}{a_2^2-a_3^2}y^2 - \frac{a_2^2+\lambda}{a_2^2-a_3^2}z^2\right)\frac{\sqrt{a_1^2-a_3^2}}{\sqrt{(a_1^2+\lambda)(a_2^2+\lambda)(a_3^2+\lambda)}}\Bigg].$$

where

$$F(\theta,k) = \int_0^\theta \frac{d\phi}{\sqrt{1-k^2\sin^2\phi}}, \tag{B.8}$$

and,

$$E(\theta,k) = \int_0^\theta d\phi\ \sqrt{1-k^2\sin^2\phi}, \tag{B.9}$$

are Legendre's elliptic integrals of the first and second kind respectively,

$$\theta \equiv \sin^{-1}\sqrt{\frac{a_1^2-a_3^2}{a_1^2+\lambda}}, \tag{B.10}$$

$$k^2 \equiv \frac{a_1^2-a_2^2}{a_1^2-a_3^2}, \tag{B.11}$$

and, λ is defined as the algebraically largest root of the following cubic equation:

$$\frac{x^2}{a_1^2+\lambda} + \frac{y^2}{a_2^2+\lambda} + \frac{z^2}{a_3^2+\lambda} = 1. \tag{B.12}$$

Appendix C: HPF Code

```
C-------------------------------------------------------------
C        BESSEL
C   MODIFICATION HISTORY:
C        H. Cohl,   19 June, 1998 ---  Initial implementation.
C-------------------------------------------------------------

      subroutine bessel(isyma)

C-------------------------------------------------------------

      include 'grid.h'
      include 'pot.h'

c-------------------------------------------------------------------------------


      real, dimension(jmax2,kmax2,lmax) :: rhoc,rhos
!hpf$  distribute  rhoc(block,block,*) onto p2
!hpf$  align  rhos(i,j,k) with rhoc(i,j,k)

      real, dimension(jmax2,kmax2) :: rTMP,zTMP,TMPC,TMPS
!hpf$  distribute  rTMP(block,block) onto p2
!hpf$  align  zTMP(i,j) with rTMP(i,j)
!hpf$  align  TMPC(i,j) with rTMP(i,j)
!hpf$  align  TMPS(i,j) with rTMP(i,j)

      real, dimension(jmax2,lmax) :: phirTMP,potr,phirTMPC,phirTMPS
!hpf$  distribute  phirTMP(block,block) onto p2
!hpf$  align  potr(i,j) with phirTMP(i,j)
!hpf$  align  phirTMPC(i,j) with phirTMP(i,j)
!hpf$  align  phirTMPS(i,j) with phirTMP(i,j)

      real, dimension(kmax2,lmax) :: phizTMP,potz,phizTMPC,phizTMPS
!hpf$  distribute  phizTMP(block,block) onto p2
!hpf$  align  potz(i,j) with phizTMP(i,j)
!hpf$  align  phizTMPC(i,j) with phizTMP(i,j)
!hpf$  align  phizTMPS(i,j) with phizTMP(i,j)

      real, dimension(jmax2,mmax) :: SjC,SjS
!hpf$  distribute  SjC(block,block) onto p2
!hpf$  align  SjS(i,j) with SjC(i,j)

      real, dimension(kmax2,mmax) :: SkC,SkS
!hpf$  distribute  SkC(block,block) onto p2
!hpf$  align  SkS(i,j) with SkC(i,j)

c-------------------------------------------------------------------------------

c     Intialize values.
c     call setup(isyma)

c     Read in Density.
c     open(unit=23,file='/u/home/hcohl/bessel/rho064.dat',form='unformatted',
```

```
c    &            status='unknown')
c     read(23)rho
c     close(23)

c-------------------------------------------------------------------------------

C     Evaluate m=0 contribution to "TOP" & "SIDE" potential.
      TMPC(:,:)=0.0
      do lp=1,lmax
        TMPC(2:jmax,2:kmax)=TMPC(2:jmax,2:kmax)+rho(2:jmax,2:kmax,lp)
      enddo
      do j=2,jmax1
        SjC(j,1)=sum(tmr(2:jmax,2:kmax,j,1)*TMPC(2:jmax,2:kmax))
        SjS(j,1)=0.0
      enddo
      do k=2,kmax1
        SkC(k,1)=sum(smz(2:jmax,2:kmax,k,1)*TMPC(2:jmax,2:kmax))
        SkS(k,1)=0.0
      enddo

c-------------------------------------------------------------------------------

C     Evaluate m=1,mmax-1 contribution to "TOP" & "SIDE" potential.
      do m=2,mmax
        TMPC(:,:)=0.0
        TMPS(:,:)=0.0
        do lp=1,lmax
          TMPC(2:jmax,2:kmax)=TMPC(2:jmax,2:kmax)
     &                        +rho(2:jmax,2:kmax,lp)
     &                        *cos(0.5*dtheta*(m-1)*(2*lp-1))
          TMPS(2:jmax,2:kmax)=TMPS(2:jmax,2:kmax)
     &                        +rho(2:jmax,2:kmax,lp)
     &                        *sin(0.5*dtheta*(m-1)*(2*lp-1))
        enddo
        do j=2,jmax1
          SjC(j,m)=sum(tmr(2:jmax,2:kmax,j,m)*TMPC(2:jmax,2:kmax))
          SjS(j,m)=sum(tmr(2:jmax,2:kmax,j,m)*TMPS(2:jmax,2:kmax))
        enddo
        do k=2,kmax1
          SkC(k,m)=sum(smz(2:jmax,2:kmax,k,m)*TMPC(2:jmax,2:kmax))
          SkS(k,m)=sum(smz(2:jmax,2:kmax,k,m)*TMPS(2:jmax,2:kmax))
        enddo
      enddo

c-------------------------------------------------------------------------------

c     forall(l=1:lmax) phirTMP(j,k,l)=(float(j)-2.0)*delr
      do l=1,lmax
        phirTMP(2:jmax1,l)=SjC(2:jmax1,1)
        phizTMP(2:kmax1,l)=SkC(2:kmax1,1)
      enddo
      do m=2,mmax
        forall(l=1:lmax) phirTMPC(2:jmax1,l)=SjC(2:jmax1,m)*cos(0.5*dtheta*(m-1)*(2*l-1))
        forall(l=1:lmax) phirTMPS(2:jmax1,l)=SjS(2:jmax1,m)*sin(0.5*dtheta*(m-1)*(2*l-1))
        forall(l=1:lmax) phizTMPC(2:kmax1,l)=SkC(2:kmax1,m)*cos(0.5*dtheta*(m-1)*(2*l-1))
        forall(l=1:lmax) phizTMPS(2:kmax1,l)=SkS(2:kmax1,m)*sin(0.5*dtheta*(m-1)*(2*l-1))
        phirTMP(2:jmax1,:)=phirTMP(2:jmax1,:)
     &                     +2*phirTMPC(2:jmax1,:)
     &                     +2*phirTMPS(2:jmax1,:)
        phizTMP(2:kmax1,:)=phizTMP(2:kmax1,:)
```

```
     &                    +2*phizTMPC(2:kmax1,:)
     &                    +2*phizTMPS(2:kmax1,:)
      enddo

      potr(2:jmax1,:)=-deltar*deltaz*dtheta*phirTMP(2:jmax1,:)
      potz(2:kmax1,:)=-deltar*deltaz*dtheta*phizTMP(2:kmax1,:)

c     Equatorial Symmetry
      potr(1,:)=cshift(potr(2,:),shift=lmax/2,dim=1)
      potz(1,:)=potz(2,:)

c-------------------------------------------------------------------------------

c     open(unit=19,file='/u/home/hcohl/bessel/tmr.dat'
c    &            ,status='unknown',form='unformatted')
c     write(19) tmr
c     close(19)

c     open(unit=19,file='/u/home/hcohl/bessel/smz.dat'
c    &            ,status='unknown',form='unformatted')
c     write(19) smz
c     close(19)

c     open(unit=20,file='/u/home/hcohl/bessel/potr.dat'
c    &            ,status='unknown',form='unformatted')
c     write(20) potr
c     close(20)

c     open(unit=20,file='/u/home/hcohl/bessel/potz.dat'
c    &            ,status='unknown',form='unformatted')
c     write(20) potz
c     close(20)

      phip(jmax1,:,:) = potz
      phip(:,kmax1,:) = potr
      phip(:,1,:)     = phip(:,2,:)

      return
      end

C-------------------------------------------------------------
C     HELMADI
C   MODIFICATION HISTORY:
C     H. Cohl,   3 Jan, 1994 ---  Made into a subroutine to
C                                 be put into pot.f.
C     H. Cohl,  16 Oct, 1993 ---  Debugged.
C     H. Cohl,  12 Sep, 1993 ---  Initial implementation.
C-------------------------------------------------------------

      subroutine helmadi(isyma,nsteps)

c-------------------------------------------------------------

      include 'grid.h'
      include 'pot.h'

c-------------------------------------------------------------

      real, dimension (jmax2, kmax2, lmax) :: ffrho,ffphi
!hpf$ distribute(block,block,*) onto p2 :: ffrho,ffphi
```

```
      real,dimension(jmax2,kmax2,lmax)::epsir,knownr,bndryr,rhor,phir
!hpf$ distribute(*,block,block) onto p2 :: epsir,knownr,bndryr,rhor,phir

      real,dimension(jmax2,kmax2,lmax)::knownz,bndryz,rhoz,phiz
!hpf$ distribute(block,*,block) onto p2 :: knownz,bndryz,rhoz,phiz

      real,dimension(nsteps)::dt

      dt(nsteps)=4./r(jmax1,1,1)**2
      alph=(r(jmax1,1,1)/r(3,1,1))**(2./(nsteps-1))
      do i=2,nsteps
        ii=nsteps+1-i
        dt(ii)=alph*dt(ii+1)
      enddo

C     Fourier transform density in azimuthal direction.
      call Realft(rhop,jmax2,kmax2,lmax,+1)
      ffrho(:,:,1)=rhop(:,:,1)
      ffrho(:,:,2:lmax/2)=rhop(:,:,3:lmax:2)
      ffrho(:,:,lmax/2+1)=rhop(:,:,2)
      ffrho(:,:,lmax/2+2:lmax)=-rhop(:,:,4:lmax:2)

C     Fourier transform initial guess for potential in azimuthal direction.
      call Realft(phip,jmax2,kmax2,lmax,+1)
      ffphi(:,:,1)=phip(:,:,1)
      ffphi(:,:,2:lmax/2)=phip(:,:,3:lmax:2)
      ffphi(:,:,lmax/2+1)=phip(:,:,2)
      ffphi(:,:,lmax/2+2:lmax)=-phip(:,:,4:lmax:2)

      phir=ffphi
      rhoz=ffrho
      rhor=rhoz

      do i=1,nsteps

        dtt=dt(i)

C       ADI sweep in radial direction.
        epsir=0.0
        epsir(j1:j2,k1:k2,:)=dtt-2.*gamma
        epsir(j1-1,:,:)=0.0
        epsir(j2+1,:,:)=0.0
        if (isyma.eq.2.or.isyma.eq.3) epsir(j1:j2,k1,:)=dtt-1.*gamma
        br(j1:j2,k1:k2,:)=brb(j1:j2,k1:k2,:)+dtt
        bndryr=0.0
        bndryr(j2,k1:k2,:)=-alphar(j2,k1:k2,:)*phir(j2+1,k1:k2,:)
        knownr(j1:j2,k1:k2,:)=-4*pi*rhor(j1:j2,k1:k2,:)
     $       +epsir(j1:j2,k1:k2,:)*phir(j1:j2,k1:k2,:)
     $       +gamma*phir(j1:j2,k1+1:k2+1,:)
     $       +gamma*phir(j1:j2,k1-1:k2-1,:)*factr(j1:j2,k1:k2,:)
     $       +bndryr(j1:j2,k1:k2,:)


        call tridagr(ar,br,cr,knownr,phir,jmax2,kmax2,lmax,j1,j2,k1,k2)
        phiz=phir

C       ADI sweep in vertical direction.
        bz(j1:j2,k1:k2,:)=bzb(j1:j2,k1:k2,:)+dtt
        elambdaz(j1:j2,k1:k2,:)=elambdazb(j1:j2,k1:k2,:)+dtt
```

```
        bndryz=0.
        if (isyma.eq.1) bndryz(j1:j2,k1,:)=gamma*phiz(j1:j2,k1-1,:)
        bndryz(j1:j2,k2,:)=gamma*phiz(j1:j2,k2+1,:)
        knownz(j1:j2,k1:k2,:)=-4*pi*rhoz(j1:j2,k1:k2,:)
     $         +elambdaz(j1:j2,k1:k2,:)*phiz(j1:j2,k1:k2,:)
     $         -alphaz(j1:j2,k1:k2,:)*phiz(j1+1:j2+1,k1:k2,:)
     $         -factz(j1:j2,k1:k2,:)*betaz(j1:j2,k1:k2,:)
     $         *phiz(j1-1:j2-1,k1:k2,:)
     $         +bndryz(j1:j2,k1:k2,:)

        call tridagz(az,bz,cz,knownz,phiz,jmax2,kmax2,lmax,j1,j2,k1,k2)
        phir=phiz

      enddo

C     Inverse Fourier transform in azimuthal direction.
      ffphi=phir

      phip(:,:,1)=ffphi(:,:,1)
      phip(:,:,2)=ffphi(:,:,lmax/2+1)
      phip(:,:,3:lmax:2)=ffphi(:,:,2:lmax/2)
      phip(:,:,4:lmax:2)=-ffphi(:,:,lmax/2+2:lmax)
      call Realft(phip,jmax2,kmax2,lmax,-1)
      do i=1,lmax
        phip(:,:,i)=phip(:,:,i)/(lmax/2)
      enddo

      if (isyma.eq.3) then
        phip(1,:,:)=phip(2,:,:)
      else
        phip(1,:,:)=cshift(phip(2,:,:),shift=lmax/2,dim=2)
      endif
      if (isyma.eq.2.or.isyma.eq.3) phip(:,1,:)=phip(:,2,:)

      return
      end

C-------------------------------------------------------------
C        POISSON
C   MODIFICATION HISTORY:
C        H. Cohl,  12 Sep, 1993 --- Initial implementation.
C-------------------------------------------------------------

      subroutine poisson(isyma)

C-------------------------------------------------------------

      include 'grid.h'
      include 'pot.h'

C-------------------------------------------------------------

      real :: timef,etime1,etime2

C     Use current potential if available.
      if (itstep.gt.1) then
        nsteps=5
      else
        nsteps=20
        if (isyma.eq.1) then
```

```
          phip(j1-1:j2,k1:k2,:)=0.0
        else
          phip(j1-1:j2,k1-1:k2,:)=0.0
        endif
      endif

      etime1=timef()

      call bessel(isyma)

      etime2=timef()
      write(6,*)" Time elapsed (Boundary: bessel) : "
     &           ,(etime2-etime1)/1000.0, " seconds."
      etime1=timef()

      call helmadi(isyma,nsteps)

      etime2=timef()
      write(6,*)" Time elapsed (Interior: ADI) : "
     &           ,(etime2-etime1)/1000.0, " seconds."

      return
      end

C-------------------------------------------------------------
C      POT
C-------------------------------------------------------------
C   MODIFICATION HISTORY:
C      H. Cohl,  11 Jan, 1994 --- Initial implementation.
C-------------------------------------------------------------

chsc  Remove comment when placed in hydrocode
c     subroutine pot(isyma)

C-------------------------------------------------------------

      include 'grid.h'
      include 'pot.h'

C-------------------------------------------------------------

Chsc  Remove when placed in hydrocode
      call setup(isyma)
Chsc  Place in setup.f for hydrocode
      call potsetup(isyma)

C-------------------------------------------------------------
      call poisson(isyma)

      phi=phip
C-------------------------------------------------------------

Chsc  Remove when placed in hydrocode.
      open(unit=22,
     $     file='/u/home/hcohl/adi/phi.dat',status='unknown',form='unformatted')
      write(22)phi
      close(22)

Chsc  Remove comment when placed in hydrocode.
c     return
```

```
      end
C-------------------------------------------------------------
C     POTSETUP
C   MODIFICATION HISTORY:
C     H. Cohl,  27 Mar, 1997 --- Initial implementation.
C-------------------------------------------------------------

      subroutine potsetup(isyma)

C-------------------------------------------------------------

      USE HPF_LIBRARY
      include 'grid.h'
      include 'pot.h'

C-------------------------------------------------------------

      integer :: shx,shy,pj,pk
      integer, dimension(7) :: shape
      integer :: isyma
      integer :: rank
      real, dimension(jmax2) :: xrhf
      real, dimension(kmax2) :: xzhf

      INTERFACE
        EXTRINSIC (f77_LOCAL) SUBROUTINE
     &         tm (shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,tmr)
          INTEGER, INTENT(IN)  :: shx
          INTEGER, INTENT(IN)  :: shy
          INTEGER, INTENT(IN)  :: pj
          INTEGER, INTENT(IN)  :: pk
          INTEGER, INTENT(IN)  :: jmax2
          INTEGER, INTENT(IN)  :: kmax2
          INTEGER, INTENT(IN)  :: mmax
          REAL,    INTENT(IN)  :: xrhf(jmax2)
          REAL,    INTENT(IN)  :: xzhf(kmax2)
          REAL,    INTENT(OUT) :: tmr(jmax2,kmax2,jmax2,mmax)
       include 'proc.h'
!hpf$  distribute  tmr(block,block,*,*) onto p2
        END SUBROUTINE tm
      END INTERFACE

      INTERFACE
        EXTRINSIC (f77_LOCAL) SUBROUTINE
     &         sm (shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,smz)
          INTEGER, INTENT(IN)  :: shx
          INTEGER, INTENT(IN)  :: shy
          INTEGER, INTENT(IN)  :: pj
          INTEGER, INTENT(IN)  :: pk
          INTEGER, INTENT(IN)  :: jmax2
          INTEGER, INTENT(IN)  :: kmax2
          INTEGER, INTENT(IN)  :: mmax
          REAL,    INTENT(IN)  :: xrhf(jmax2)
          REAL,    INTENT(IN)  :: xzhf(kmax2)
          REAL,    INTENT(OUT) :: smz(jmax2,kmax2,kmax2,mmax)
       include 'proc.h'
!hpf$  distribute  smz(block,block,*,*) onto p2
        END SUBROUTINE sm
      END INTERFACE
```

```
C-------------------------------------------------------------

      real,dimension(jmax2,kmax2,lmax)::elm,m1mode,orhf,orhf2
!hpf$  align  elm(i,j,k) with r(i,j,k)
!hpf$  align  m1mode(i,j,k) with r(i,j,k)
!hpf$  align  orhf(i,j,k) with r(i,j,k)
!hpf$  align  orhf2(i,j,k) with r(i,j,k)

      real,dimension(jmax2,kmax2,lmax)::betar
!hpf$ distribute(*,block,block) onto p2 :: betar

c     Determine number of elements per processor
      call hpf_distribution(tmr,processors_rank=rank,processors_shape=shape)
      shx=shape(1)
      shy=shape(2)
      pj=jmax2/shx
      pk=kmax2/shy
      xrhf(:)=rhf(:,1,1)
      xzhf(:)=zhf(1,:,1)

c     Read in tm & sm arrays.
c     if (itstep.eq.1) then
        call tm(shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,tmr)
        call sm(shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,smz)
c     endif

C-------------------------------------------------------------
      open(unit=20,
     $     file='/u/home/hcohl/adi/rho064.dat',
     $     status='unknown',form='unformatted')
      read(20)rho
      close(20)
C-------------------------------------------------------------
      open(unit=21,
     $     file='/u/home/hcohl/adi/pot064.dat',
     $     status='unknown',form='unformatted')
      read(21)phi
      close(21)
C-------------------------------------------------------------

c     write(6,*)phi(:,2,1)

      phip=phi
c     phip=0.0
      rhop=rho

      j1=2
      k1=2
      j2=jmax
      k2=kmax

      eodr2=1./(deltar**2)
      eodtheta2=1./(dtheta**2)
      gamma=1./(deltaz**2)
      orhf=1./rhf
      orhf2=orhf**2

      lstop=lmax/2+1
      do l=1,lmax
        if (isyma.eq.3) then
```

```
          if (l.le.lstop) mode=(l-1)*2
          if (l.gt.lstop) mode=(l-lstop)*2
        else
          if (l.le.lstop) mode=(l-1)
          if (l.gt.lstop) mode=(l-lstop)
        endif
        m1mode(j1:j2,k1:k2,l)=(-1)**mode
        elm(j1:j2,k1:k2,l)=cos(mode*dtheta)
      enddo

      alphar(j1:j2,k1:k2,:)=-r(j1+1:j2+1,k1:k2,:)*orhf(j1:j2,k1:k2,:)*eodr2
      alphaz(j1:j2,k1:k2,:)=alphar(j1:j2,k1:k2,:)
      betar(j1:j2,k1:k2,:)=-r(j1:j2,k1:k2,:)*orhf(j1:j2,k1:k2,:)*eodr2
      betaz(j1:j2,k1:k2,:)=betar(j1:j2,k1:k2,:)

      ar(j1+1:j2,k1:k2,:)=betar(j1+1:j2,k1:k2,:)
      cr(j1:j2-1,k1:k2,:)=alphar(j1:j2-1,k1:k2,:)

      az(j1:j2,k1+1:k2,:)=-gamma
      cz(j1:j2,k1:k2-1,:)=-gamma

      brb(j1+1:j2,k1:k2,:)=2.*eodr2-2.*
     $  (elm(j1+1:j2,k1:k2,:)-1.)*eodtheta2*orhf2(j1+1:j2,k1:k2,:)

      if (isyma.eq.3) then
        brb(j1,k1:k2,:)=-alphar(j1,k1:k2,:)
c     $     -2.*betar(j1,k1:k2,:)
     $     -2.*(elm(j1,k1:k2,:)-1.)*eodtheta2*orhf2(j1,k1:k2,:)
      else
        brb(j1,k1:k2,:)=-alphar(j1,k1:k2,:)+
     $     (m1mode(j1,k1:k2,:)-1.)*betar(j1,k1:k2,:)
     $     -2.*(elm(j1,k1:k2,:)-1.)*eodtheta2*orhf2(j1,k1:k2,:)
      endif

      bzb(j1:j2,k1:k2,:)=2.*gamma
      if (isyma.eq.2.or.isyma.eq.3) bzb(j1:j2,k1,:)=gamma

      elambdazb(j1+1:j2,k1:k2,:)=-2.*eodr2+2.*
     $        (elm(j1+1:j2,k1:k2,:)-1.)*eodtheta2*orhf2(j1+1:j2,k1:k2,:)

      if (isyma.eq.3) then
        elambdazb(j1,k1:k2,:)=alphaz(j1,k1:k2,:)
c     $        +2.*betaz(j1,k1:k2,:)
     $        +2.*(elm(j1,k1:k2,:)-1.)*eodtheta2*orhf2(j1,k1:k2,:)
      else
        elambdazb(j1,k1:k2,:)=alphaz(j1,k1:k2,:)
     $        -(m1mode(j1,k1:k2,:)-1.)*betaz(j1,k1:k2,:)
     $        +2.*(elm(j1,k1:k2,:)-1.)*eodtheta2*orhf2(j1,k1:k2,:)
      endif

      factr=1.
      if (isyma.eq.2.or.isyma.eq.3) factr(:,k1,:)=0.
      factz=1.
      factz(j1,:,:)=0.

      return
      end

C-------------------------------------------------------------
C     REALFT
```

```
C-------------------------------------------------------------

      subroutine realft(data,nx,ny,nz,isign)

c-------------------------------------------------------------

      include 'proc.h'

      real,dimension(nx,ny,nz)::data
!hpf$  distribute  data(block,block,*) onto p2

      real,dimension(nx,ny)::h1i,h1r,h2i,h2r
!hpf$  distribute(block,block) onto p2 :: h1i,h1r,h2i,h2r

      real::theta,wi,wpi,wpr,wr,wtemp

      theta=3.141592653589793/(nz/2)
      c1=0.5
      if (isign.eq.1) then
        c2=-0.5
        call four1(data,nx,ny,nz/2,+1)
      else
        c2=0.5
        theta=-theta
      endif
      wpr=-2.0*sin(0.5*theta)**2
      wpi=sin(theta)
      wr=1.0+wpr
      wi=wpi
      n2p3=nz+3
      do i=2,nz/4
        i1=2*i-1
        i2=i1+1
        i3=n2p3-i2
        i4=i3+1
        wrs=wr
        wis=wi
        h1r=c1*(data(:,:,i1)+data(:,:,i3))
        h1i=c1*(data(:,:,i2)-data(:,:,i4))
        h2r=-c2*(data(:,:,i2)+data(:,:,i4))
        h2i=c2*(data(:,:,i1)-data(:,:,i3))
        data(:,:,i1)=h1r+wrs*h2r-wis*h2i
        data(:,:,i2)=h1i+wrs*h2i+wis*h2r
        data(:,:,i3)=h1r-wrs*h2r+wis*h2i
        data(:,:,i4)=-h1i+wrs*h2i+wis*h2r
        wtemp=wr
        wr=wr*wpr-wi*wpi+wr
        wi=wi*wpr+wtemp*wpi+wi
      enddo
      if (isign.eq.1) then
        h1r(:,:)=data(:,:,1)
        data(:,:,1)=h1r(:,:)+data(:,:,2)
        data(:,:,2)=h1r(:,:)-data(:,:,2)
      else
        h1r(:,:)=data(:,:,1)
        data(:,:,1)=c1*(h1r(:,:)+data(:,:,2))
        data(:,:,2)=c1*(h1r(:,:)-data(:,:,2))
        call four1(data,nx,ny,nz/2,-1)
      endif
      return
```

```
      end

C--------------------------------------------------------------
C     FOUR1
C--------------------------------------------------------------

      subroutine four1(data,nx,ny,nnz,isign)

c--------------------------------------------------------------

      include 'proc.h'

      real,dimension(nx,ny,2*nnz)::data
!hpf$  distribute  data(block,block,*) onto p2

      real,dimension(nx,ny)::tempi,tempr
!hpf$  distribute(block,block) onto p2 :: tempi,tempr

      real::theta,wi,wpi,wpr,wr,wtemp
      integer::nx,ny,nnz,isign,i,istep,j,m,mmax,nz

      nz=2*nnz
      j=1
      do i=1,nz,2
        if(j.gt.i)then
          tempr=data(:,:,j)
          tempi=data(:,:,j+1)
          data(:,:,j)=data(:,:,i)
          data(:,:,j+1)=data(:,:,i+1)
          data(:,:,i)=tempr
          data(:,:,i+1)=tempi
        endif
        m=nz/2
 1      continue
        if ((m.ge.2).and.(j.gt.m)) then
          j=j-m
          m=m/2
        goto 1
        endif
        j=j+m
      enddo
      mmax=2
 2    continue
      if (nz.gt.mmax) then
        istep=2*mmax
        theta=6.28318530717959/(isign*mmax)
        wpr=-2.*sin(0.5*theta)**2
        wpi=sin(theta)
        wr=1.
        wi=0.
        do m=1,mmax,2
          do i=m,nz,istep
            j=i+mmax
            tempr=wr*data(:,:,j)-wi*data(:,:,j+1)
            tempi=wr*data(:,:,j+1)+wi*data(:,:,j)
            data(:,:,j)=data(:,:,i)-tempr
            data(:,:,j+1)=data(:,:,i+1)-tempi
            data(:,:,i)=data(:,:,i)+tempr
            data(:,:,i+1)=data(:,:,i+1)+tempi
          enddo
```

```
          wtemp=wr
          wr=wr*wpr-wi*wpi+wr
          wi=wi*wpr+wtemp*wpi+wi
        enddo
        mmax=istep
      goto 2
      endif
      return
      end
C-------------------------------------------------------------
C     SETUP
C     /u/home/hcohl/adi
C   MODIFICATION HISTORY:
C     H. Cohl,  27 Mar, 1997 --- Initial implementation.
C-------------------------------------------------------------

      subroutine setup(isyma)

C-------------------------------------------------------------

      include 'grid.h'

C-------------------------------------------------------------

c     Set time step.
      itstep=2
c     itstep=1

c     Set grid geometry.
      isyma=2

c     Set up grid spacing.
c     deltar=1e-2
c     deltaz=1e-2
      deltar=1.88679E-02
      deltaz=1.88679E-02
      pi=3.1415926535e0
      grav=1.0

      if (isyma.eq.3) then
        dtheta=pi/real(lmax)
      else
        dtheta=2.*pi/real(lmax)
      endif

      forall(j=1:jmax2,k=1:kmax2,l=1:lmax) r(j,k,l)=(float(j)-2.0)*deltar
      forall(j=1:jmax2,k=1:kmax2,l=1:lmax) rhf(j,k,l)=(float(j)-1.5)*deltar
      rplus  = eoshift(r,dim=1,shift= 1)
      rplus(jmax2,:,:) = rplus(jmax1,:,:) + deltar

      if (isyma.eq.1) then
        forall(j=1:jmax2,k=1:kmax2,l=1:lmax)
     &         z(j,k,l)=(float(k)-1.0-kmax/2)*deltaz
      else
        forall(j=1:jmax2,k=1:kmax2,l=1:lmax)
     &         z(j,k,l)=(float(k)-2.0)*deltaz
      endif

      if (isyma.eq.1) then
        forall(j=1:jmax2,k=1:kmax2,l=1:lmax)
```

```
     &         zhf(j,k,l)=(float(k)-0.5-kmax/2)*deltaz
      else
        forall(j=1:jmax2,k=1:kmax2,l=1:lmax)
     &         zhf(j,k,l)=(float(k)-1.5)*deltaz
      endif

      return
      end
C-------------------------------------------------------------
C     TRIDAGR
C-------------------------------------------------------------
C   MODIFICATION HISTORY:
C     H. Cohl,  18 Sep, 1993 ---  Initial implementation.
C   NOTES: Given a tridiagonal matrix M of size NxN with
C     diagonal elements M(i,i) are in B(1)..B(N)
C     lower offdiagonal elements M(i+1,i) are in A(2)..A(N)
C     upper offdiagonal elements M(i,i+1) are in C(1)..C(N-1)
C     solve the equation MU=R for the vector U().
C-------------------------------------------------------------

      subroutine tridagr(a,b,c,r,u,nx,ny,nz,j1,j2,k1,k2)

      include 'proc.h'

      real,dimension(nx,ny,nz)::a,b,c,r,u,gam,bet
!hpf$ distribute(*,block,block) onto p2 :: a,b,c,r,u,gam,bet

c-------------------------------------------------------------

C     Forward Pass
      bet(j1,k1:k2,:)=b(j1,k1:k2,:)
      u(j1,k1:k2,:)=r(j1,k1:k2,:)/bet(j1,k1:k2,:)

      do 10 j=j1+1,j2
        gam(j,k1:k2,:)=c(j-1,k1:k2,:)/bet(j-1,k1:k2,:)
        bet(j,k1:k2,:)=b(j,k1:k2,:)-a(j,k1:k2,:)*gam(j,k1:k2,:)
        u(j,k1:k2,:)=(r(j,k1:k2,:)-a(j,k1:k2,:)
     $               *u(j-1,k1:k2,:))/bet(j,k1:k2,:)
 10   continue

C     Back-substitution pass
      do j=j2-1,j1,-1
        u(j,k1:k2,:)=u(j,k1:k2,:)-gam(j+1,k1:k2,:)*u(j+1,k1:k2,:)
      enddo

      return
      end


C-------------------------------------------------------------
C     TRIDAGZ
C-------------------------------------------------------------
C   MODIFICATION HISTORY:
C     H. Cohl,  18 Sep, 1993 ---  Initial implementation.
C   NOTES: Given a tridiagonal matrix M of size NxN with
C        diagonal elements M(i,i) are in B(1)..B(N)
C        lower offdiagonal elements M(i+1,i) are in A(2)..A(N)
C        upper offdiagonal elements M(i,i+1) are in C(1)..C(N-1)
C        solve the equation MU=R for the vector U().
C-------------------------------------------------------------
```

```
      subroutine tridagz(a,b,c,r,u,nx,ny,nz,j1,j2,k1,k2)

c-------------------------------------------------------------

      include 'proc.h'

      real,dimension(nx,ny,nz)::a,b,c,r,u,gam,bet
!hpf$ distribute(block,*,block) onto p2 :: a,b,c,r,u,gam,bet

C     Forward Pass
      bet(j1:j2,k1,:)=b(j1:j2,k1,:)
      u(j1:j2,k1,:)=r(j1:j2,k1,:)/bet(j1:j2,k1,:)

      do k=k1+1,k2
        gam(j1:j2,k,:)=c(j1:j2,k-1,:)/bet(j1:j2,k-1,:)
        bet(j1:j2,k,:)=b(j1:j2,k,:)-a(j1:j2,k,:)*gam(j1:j2,k,:)
        u(j1:j2,k,:)=(r(j1:j2,k,:)-a(j1:j2,k,:)
     $               *u(j1:j2,k-1,:))/bet(j1:j2,k,:)
      enddo

C     Back-substitution pass
      do k=k2-1,k1,-1
        u(j1:j2,k,:)=u(j1:j2,k,:)-gam(j1:j2,k+1,:)*u(j1:j2,k+1,:)
      enddo

      return
      end
```

Appendix D: F77 Code

```
C--------------------------------------------------------------
      FUNCTION elle(phi,ak)
      REAL elle,ak,phi
CU    USES rd,rf
      REAL cc,q,s,rd,rf
      s=sin(phi)
      cc=cos(phi)**2
      q=(1.-s*ak)*(1.+s*ak)
      elle=s*(rf(cc,q,1.)-((s*ak)**2)*rd(cc,q,1.)/3.)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
C--------------------------------------------------------------
      FUNCTION ellf(phi,ak)
      REAL ellf,ak,phi
CU    USES rf
      REAL s,rf
      s=sin(phi)
      ellf=s*rf(cos(phi)**2,(1.-s*ak)*(1.+s*ak),1.)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
C--------------------------------------------------------------
      FUNCTION factrl(n)
      INTEGER n
      REAL factrl
CU    USES gammln
      INTEGER j,ntop
      REAL a(33),gammln
      SAVE ntop,a
      DATA ntop,a(1)/0,1./
      if (n.lt.0) then
        pause 'negative factorial in factrl'
      else if (n.le.ntop) then
        factrl=a(n+1)
      else if (n.le.32) then
        do 11 j=ntop+1,n
          a(j+1)=j*a(j)
11      continue
        ntop=n
        factrl=a(n+1)
      else
        factrl=exp(gammln(n+1.))
      endif
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
C--------------------------------------------------------------
      FUNCTION gammln(xx)
      REAL gammln,xx
      INTEGER j
      DOUBLE PRECISION ser,stp,tmp,x,y,cof(6)
      SAVE cof,stp
```

```
      DATA cof,stp/76.18009172947146d0,-86.50532032941677d0,
     *24.01409824083091d0,-1.231739572450155d0,.1208650973866179d-2,
     *-.5395239384953d-5,2.506628274631000 5d0/
      x=xx
      y=x
      tmp=x+5.5d0
      tmp=(x+0.5d0)*log(tmp)-tmp
      ser=1.000000000190015d0
      do 11 j=1,6
        y=y+1.d0
        ser=ser+cof(j)/y
11    continue
      gammln=tmp+log(stp*ser/x)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
C-------------------------------------------------------------
      FUNCTION rd(x,y,z)
      REAL rd,x,y,z,ERRTOL,TINY,BIG,C1,C2,C3,C4,C5,C6
      PARAMETER (ERRTOL=.000015,TINY=1.e-25,BIG=4.5E21,C1=3./14.,C2=1./6.,
     *C3=9./22.,C4=3./26.,C5=.25*C3,C6=1.5*C4)
      REAL alamb,ave,delx,dely,delz,ea,eb,ec,ed,ee,fac,sqrtx,sqrty,
     *sqrtz,sum,xt,yt,zt
      if(min(x,y).lt.0..or.min(x+y,z).lt.TINY.or.max(x,y,
     *z).gt.BIG)pause 'invalid arguments in rd'
      xt=x
      yt=y
      zt=z
      sum=0.
      fac=1.
1     continue
        sqrtx=sqrt(xt)
        sqrty=sqrt(yt)
        sqrtz=sqrt(zt)
        alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz
        sum=sum+fac/(sqrtz*(zt+alamb))
        fac=.25*fac
        xt=.25*(xt+alamb)
        yt=.25*(yt+alamb)
        zt=.25*(zt+alamb)
        ave=.2*(xt+yt+3.*zt)
        delx=(ave-xt)/ave
        dely=(ave-yt)/ave
        delz=(ave-zt)/ave
      if(max(abs(delx),abs(dely),abs(delz)).gt.ERRTOL)goto 1
      ea=delx*dely
      eb=delz*delz
      ec=ea-eb
      ed=ea-6.*eb
      ee=ed+ec+ec
      rd=3.*sum+fac*(1.+ed*(-C1+C5*ed-C6*delz*ee)+delz*(C2*ee+delz*(-C3*
     *ec+delz*C4*ea)))/(ave*sqrt(ave))
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
C-------------------------------------------------------------
      FUNCTION rf(x,y,z)
      REAL rf,x,y,z,ERRTOL,TINY,BIG,THIRD,C1,C2,C3,C4
      PARAMETER (ERRTOL=.000025,TINY=1.5e-38,BIG=3.E37,THIRD=1./3.,
     *C1=1./24.,C2=.1,C3=3./44.,C4=1./14.)
```

```
      REAL alamb,ave,delx,dely,delz,e2,e3,sqrtx,sqrty,sqrtz,xt,yt,zt
      if(min(x,y,z).lt.0..or.min(x+y,x+z,y+z).lt.TINY.or.max(x,y,
     *z).gt.BIG)pause 'invalid arguments in rf'
      xt=x
      yt=y
      zt=z
1     continue
        sqrtx=sqrt(xt)
        sqrty=sqrt(yt)
        sqrtz=sqrt(zt)
        alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz
        xt=.25*(xt+alamb)
        yt=.25*(yt+alamb)
        zt=.25*(zt+alamb)
        ave=THIRD*(xt+yt+zt)
        delx=(ave-xt)/ave
        dely=(ave-yt)/ave
        delz=(ave-zt)/ave
      if(max(abs(delx),abs(dely),abs(delz)).gt.ERRTOL)goto 1
      e2=delx*dely-delz**2
      e3=delx*dely*delz
      rf=(1.+(C1*e2-C2-C3*e3)*e2+C4*e3)/sqrt(ave)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software .
c*************************************************************************

      subroutine sm(isyma,shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,smz)

c*************************************************************************

      include '/usr/local/pgi/t3e/include/pglocal.f'

      parameter(irmax = 111)
      integer n,m,mm,nprocs,myproc
      integer isyma,shx,shy,pj,pk
      integer jmax2,kmax2,mmax
      integer jstart,kstart,jfinish,kfinish
      integer loc(shx*shy,2)
      real xrhf(jmax2),xzhf(kmax2)
      real smz(pj,pk,kmax2,mmax)
      real qp(mmax),qm(mmax)
      real RB,ellf,elle,pi,pi2,coef
      real b,c,ap,am,xp,xm,mup,mum,lap,lam
      real Kmup,Kmum,Emup,Emum
      real nu
      real coefh,dcoefh,gamma,factorial
      real aa,bb,cc,yy,alpha,sum,diff,Fabcy
cjc I added these arrays to speed this process up!
      real myalpha(mmax),mycoefh(mmax)
      real mydcoefh(irmax,mmax)
      real gmlhf


      RB=xrhf(jmax2-1)
      c=RB

      myproc = pghpf_myprocnum()
      nprocs = pghpf_nprocs()
```

```
c     pi = 3.14159265358979324e0
      pi = 3.14159265358979323846264338327950288419716939937510582097494 46
      pi2 = 0.5*pi

      n=1
      do j=1,shy
        do i=1,shx
          loc(n,1)=i-1
          loc(n,2)=j-1
          n=n+1
        enddo
      enddo

      if (loc(myproc+1,1).eq.0) then
        jstart=2
      else
        jstart=1
      endif
      if (loc(myproc+1,2).eq.0) then
        kstart=2
      else
        kstart=1
      endif

      if (loc(myproc+1,1).eq.shx-1) then
        jfinish=pj-2
      else
        jfinish=pj
      endif
      if (loc(myproc+1,2).eq.shy-1) then
        kfinish=pk-2
      else
        kfinish=pk
      endif

cjc   ARRAY SETUP
      gmlhf = gammln(0.5)
      do m=1,mmax
         mm=m-1
    if (isyma.eq.3) mm=2*(m-1)
         mycoefh(m)=exp(gmlhf+gammln(mm+.5))/factrl(mm)
         aa=(mm+1.5)/2.
         bb=(mm+.5)/2.
         cc=mm+1
         myalpha(m)=exp(gammln(cc)-gammln(aa)-gammln(bb))
         do ir=1,irmax
            fr=factrl(ir-1)
            mydcoefh(ir,m)=exp(gammln(aa+ir-1)+gammln(bb+ir-1)-gammln(cc+ir-1))/fr
         enddo
      enddo



c     Equatorial Symmetry
      if (isyma.eq.2) then
        do jj=jstart,jfinish
          b=xrhf(loc(myproc+1,1)*pj+jj)
          coef=sqrt(b/c)/pi
          do kk=kstart,kfinish
```

```
do kkk=2,kmax2-1

  if (loc(myproc+1,2)*pk+kk.lt.kkk) then
    ap=xzhf(kkk)+xzhf(loc(myproc+1,2)*pk+kk)
    am=xzhf(kkk)-xzhf(loc(myproc+1,2)*pk+kk)
  else
    ap=xzhf(loc(myproc+1,2)*pk+kk)+xzhf(kkk)
    am=xzhf(loc(myproc+1,2)*pk+kk)-xzhf(kkk)
  endif

  xp=0.5*(ap**2+b**2+c**2)/(b*c)
  if (xp.lt.1.025) then
    mup=sqrt(2.0/(1.0+xp))
    lap=sqrt(2.0*(1.0+xp))
    Kmup=ellf(pi2,mup)
    Emup=elle(pi2,mup)
    qp(1)=Kmup*mup
    qp(2)=xp*mup*Kmup-lap*Emup
    do m=3,mmax
      nu=(2.*m-5.)/2.
      qp(m)=(2.*nu+1.)/(nu+1.)*xp*qp(m-1)-nu/(nu+1.)*qp(m-2)
    enddo
  else
    do m=1,mmax
      mm=m-1
      coefh=mycoefh(m)/(2*xp)**(mm+.5)
      yy=1./xp**2
      alpha=myalpha(m)
      sum=0.0
      do ir=1,irmax
        diff=mydcoefh(ir,m)*yy**(ir-1)
        sum=sum+diff
      enddo
      Fabcy=alpha*sum
      qp(m)=coefh*Fabcy
    enddo
  endif

  xm=0.5*(am**2+b**2+c**2)/(b*c)
  if (xm.lt.1.025) then
    mum=sqrt(2.0/(1.0+xm))
    lam=sqrt(2.0*(1.0+xm))
    Kmum=ellf(pi2,mum)
    Emum=elle(pi2,mum)
    qm(1)=Kmum*mum
    qm(2)=xm*mum*Kmum-lam*Emum
    do m=3,mmax
      nu=(2.*m-5.)/2.
      qm(m)=(2.*nu+1.)/(nu+1.)*xm*qm(m-1)-nu/(nu+1.)*qm(m-2)
    enddo
  else
    do m=1,mmax
      mm=m-1
      coefh=mycoefh(m)/(2*xm)**(mm+.5)
      yy=1./xm**2
      alpha=myalpha(m)
      sum=0.0
      do ir=1,irmax
        diff=mydcoefh(ir,m)*yy**(ir-1)
        sum=sum+diff
```

```
                enddo
                Fabcy=alpha*sum
                qm(m)=coefh*Fabcy
              enddo
            endif

            do m=1,mmax
              smz(jj,kk,kkk,m)=coef*(qp(m)+qm(m))
            enddo

          enddo
        enddo
      enddo
c     PI-symmetry
      else if (isyma.eq.3) then
        do jj=jstart,jfinish
          if (myproc.eq.1) write(6,*)jj,jfinish
          b=xrhf(loc(myproc+1,1)*pj+jj)
          coef=sqrt(b/c)/pi
          do kk=kstart,kfinish
            do kkk=2,kmax2-1

              if (loc(myproc+1,2)*pk+kk.lt.kkk) then
                ap=xzhf(kkk)+xzhf(loc(myproc+1,2)*pk+kk)
                am=xzhf(kkk)-xzhf(loc(myproc+1,2)*pk+kk)
              else
                ap=xzhf(loc(myproc+1,2)*pk+kk)+xzhf(kkk)
                am=xzhf(loc(myproc+1,2)*pk+kk)-xzhf(kkk)
              endif

              xp=0.5*(ap**2+b**2+c**2)/(b*c)
              if (xp.lt.1.025) then
                mup=sqrt(2.0/(1.0+xp))
                lap=sqrt(2.0*(1.0+xp))
                Kmup=ellf(pi2,mup)
                Emup=elle(pi2,mup)
                qp(1)=Kmup*mup
                qp(2)=(4/3.*xp**2-1/3.)*mup*Kmup-4/3.*xp*lap*Emup
                do m=3,mmax
                  nu=(4.*m-9.)/2.
                  qp(m)=qp(m-1)*((2*nu+3)*(2*nu+1)*xp**2/((nu+2)*(nu+1))
     &                          -(2*nu+3)*nu**2/((2*nu-1)*(nu+2)*(nu+1))
     &                          -(nu+1)/(nu+2))
     &                -qp(m-2)*(2*nu+3)*(nu-1)*nu/((2*nu-1)*(nu+2)*(nu+1))
                enddo
              else
                do m=1,mmax
                  mm=2*(m-1)
                  coefh=mycoefh(m)/(2*xp)**(mm+.5)
                  yy=1./xp**2
                  alpha=myalpha(m)
                  sum=0.0
                  do ir=1,irmax
                    diff=mydcoefh(ir,m)*yy**(ir-1)
                    sum=sum+diff
                  enddo
                  Fabcy=alpha*sum
                  qp(m)=coefh*Fabcy
                enddo
              endif
```

```
            xm=0.5*(am**2+b**2+c**2)/(b*c)
            if (xm.lt.1.025) then
              mum=sqrt(2.0/(1.0+xm))
              lam=sqrt(2.0*(1.0+xm))
              Kmum=ellf(pi2,mum)
              Emum=elle(pi2,mum)
              qm(1)=Kmum*mum
              qm(2)=(4/3.*xm**2-1/3.)*mum*Kmum-4/3.*xm*lam*Emum
              do m=3,mmax
                nu=(4.*m-9.)/2.
                qm(m)=qm(m-1)*((2*nu+3)*(2*nu+1)*xm**2/((nu+2)*(nu+1))
     &                        -(2*nu+3)*nu**2/((2*nu-1)*(nu+2)*(nu+1))
     &                        -(nu+1)/(nu+2))
     &               -qm(m-2)*(2*nu+3)*(nu-1)*nu/((2*nu-1)*(nu+2)*(nu+1))
              enddo
            else
              do m=1,mmax
                mm=2*(m-1)
                coefh=mycoefh(m)/(2*xm)**(mm+.5)
                yy=1./xm**2
                alpha=myalpha(m)
                sum=0.0
                do ir=1,irmax
                  diff=mydcoefh(ir,m)*yy**(ir-1)
                  sum=sum+diff
                enddo
                Fabcy=alpha*sum
                qm(m)=coefh*Fabcy
              enddo
            endif

            do m=1,mmax
              smz(jj,kk,kkk,m)=coef*(qp(m)+qm(m))
            enddo

          enddo
        enddo
      enddo
    endif

    return
    end

c*************************************************************************

    subroutine tm(isyma,shx,shy,pj,pk,jmax2,kmax2,mmax,xrhf,xzhf,tmr)

c*************************************************************************

    include '/usr/local/pgi/t3e/include/pglocal.f'

    parameter(irmax = 111)
    integer n,m,mm,nprocs,myproc
    integer isyma,shx,shy,pj,pk
    integer jmax2,kmax2,mmax
    integer jstart,kstart,jfinish,kfinish
    integer loc(shx*shy,2)
    real xrhf(jmax2),xzhf(kmax2)
    real tmr(pj,pk,jmax2,mmax)
```

```
      real qp(mmax),qm(mmax)
      real zB,ellf,elle,pi,pi2,coef
      real b,c,ap,am,xp,xm,mup,mum,lap,lam
      real Kmup,Kmum,Emup,Emum
      real nu
      real coefh,dcoefh,gamma,factorial
      real aa,bb,cc,yy,alpha,sum,diff,Fabcy

cjc I added these arrays to speed this process up!
      real myalpha(mmax),mycoefh(mmax)
      real mydcoefh(irmax,mmax)
      real gmlhf

      zB=xzhf(kmax2-1)

      myproc = pghpf_myprocnum()
      nprocs = pghpf_nprocs()

c     pi = 3.14159265358979324
      pi = 3.14159265358979323846264338327950288419716939937510582097494 46
      pi2 = 0.5*pi

      n=1
      do j=1,shy
        do i=1,shx
          loc(n,1)=i-1
          loc(n,2)=j-1
          n=n+1
        enddo
      enddo

      if (loc(myproc+1,1).eq.0) then
        jstart=2
      else
        jstart=1
      endif
      if (loc(myproc+1,2).eq.0) then
        kstart=2
      else
        kstart=1
      endif

      if (loc(myproc+1,1).eq.shx-1) then
        jfinish=pj-2
      else
        jfinish=pj
      endif
      if (loc(myproc+1,2).eq.shy-1) then
        kfinish=pk-2
      else
        kfinish=pk
      endif

cjc      array setup
      gmlhf = gammln(0.5)
      do m=1,mmax
         mm=m-1
         if (isyma.eq.3) mm=2*(m-1)
         mycoefh(m)=exp(gmlhf+gammln(mm+.5))/factrl(mm)
         aa=(mm+1.5)/2.
```

```
          bb=(mm+.5)/2.
          cc=mm+1
          myalpha(m)=exp(gammln(cc)-gammln(aa)-gammln(bb))
          do ir=1,irmax
             fr=factrl(ir-1)
             mydcoefh(ir,m)=exp(gammln(aa+ir-1)+gammln(bb+ir-1)-gammln(cc+ir-1))/fr
          enddo
        enddo

c     Equatorial Symmetry
      if (isyma.eq.2) then
        do jj=jstart,jfinish
          do kk=kstart,kfinish
            do jjj=2,jmax2-1

              b=xrhf(loc(myproc+1,1)*pj+jj)
              c=xrhf(jjj)
              ap=zB+xzhf(loc(myproc+1,2)*pk+kk)
              am=zB-xzhf(loc(myproc+1,2)*pk+kk)
              coef=sqrt(b/c)/pi

              xp=0.5*(ap**2+b**2+c**2)/(b*c)
              if (xp.lt.1.025) then
                mup=sqrt(2.0/(1.0+xp))
                lap=sqrt(2.0*(1.0+xp))
                Kmup=ellf(pi2,mup)
                Emup=elle(pi2,mup)
                qp(1)=Kmup*mup
                qp(2)=xp*mup*Kmup-lap*Emup
                do m=3,mmax
                  nu=(2.*m-5.)/2.
                  qp(m)=(2.*nu+1.)/(nu+1.)*xp*qp(m-1)-nu/(nu+1.)*qp(m-2)
                enddo
              else
                do m=1,mmax
                  mm=m-1
                  coefh=mycoefh(m)/(2*xp)**(mm+.5)
                  yy=1./xp**2
                  alpha=myalpha(m)
                  sum=0.0
                  do ir=1,irmax
                    diff=mydcoefh(ir,m)*yy**(ir-1)
                    sum=sum+diff
                  enddo
                  Fabcy=alpha*sum
                  qp(m)=coefh*Fabcy
                enddo
              endif

              xm=0.5*(am**2+b**2+c**2)/(b*c)
              if (xm.lt.1.025) then
                mum=sqrt(2.0/(1.0+xm))
                lam=sqrt(2.0*(1.0+xm))
                Kmum=ellf(pi2,mum)
                Emum=elle(pi2,mum)
                qm(1)=Kmum*mum
                qm(2)=xm*mum*Kmum-lam*Emum
                do m=3,mmax
                  nu=(2.*m-5.)/2.
                  qm(m)=(2.*nu+1.)/(nu+1.)*xm*qm(m-1)-nu/(nu+1.)*qm(m-2)
```

```
              enddo
            else
              do m=1,mmax
                mm=m-1
                coefh=mycoefh(m)/(2*xm)**(mm+.5)
                yy=1./xm**2
                alpha=myalpha(m)
                sum=0.0
                do ir=1,irmax
                  diff=mydcoefh(ir,m)*yy**(ir-1)
                  sum=sum+diff
                enddo
                Fabcy=alpha*sum
                qm(m)=coefh*Fabcy
              enddo
            endif

            do m=1,mmax
              tmr(jj,kk,jjj,m)=coef*(qp(m)+qm(m))
            enddo

          enddo
        enddo
      enddo
c     PI-symmetry
      else if (isyma.eq.3) then
        do jj=jstart,jfinish
          if (myproc.eq.1) write(6,*)jj,jfinish
          do kk=kstart,kfinish
            do jjj=2,jmax2-1

              b=xrhf(loc(myproc+1,1)*pj+jj)
              c=xrhf(jjj)
              ap=zB+xzhf(loc(myproc+1,2)*pk+kk)
              am=zB-xzhf(loc(myproc+1,2)*pk+kk)
              coef=sqrt(b/c)/pi

              xp=0.5*(ap**2+b**2+c**2)/(b*c)
              if (xp.lt.1.025) then
                mup=sqrt(2.0/(1.0+xp))
                lap=sqrt(2.0*(1.0+xp))
                Kmup=ellf(pi2,mup)
                Emup=elle(pi2,mup)
                qp(1)=Kmup*mup
                qp(2)=(4/3.*xp**2-1/3.)*mup*Kmup-4/3.*xp*lap*Emup
                do m=3,mmax
                  nu=(4.*m-9.)/2.
                  qp(m)=qp(m-1)*((2*nu+3)*(2*nu+1)*xp**2/((nu+2)*(nu+1))
     &                          -(2*nu+3)*nu**2/((2*nu-1)*(nu+2)*(nu+1))
     &                          -(nu+1)/(nu+2))
     &                -qp(m-2)*(2*nu+3)*(nu-1)*nu/((2*nu-1)*(nu+2)*(nu+1))
                enddo
              else
                do m=1,mmax
                  mm=2*(m-1)
                  coefh=mycoefh(m)/(2*xp)**(mm+.5)
                  yy=1./xp**2
                  alpha=myalpha(m)
                  sum=0.0
                  do ir=1,irmax
```

```
              diff=mydcoefh(ir,m)*yy**(ir-1)
              sum=sum+diff
            enddo
            Fabcy=alpha*sum
            qp(m)=coefh*Fabcy
          enddo
        endif

        xm=0.5*(am**2+b**2+c**2)/(b*c)
        if (xm.lt.1.025) then
          mum=sqrt(2.0/(1.0+xm))
          lam=sqrt(2.0*(1.0+xm))
          Kmum=ellf(pi2,mum)
          Emum=elle(pi2,mum)
          qm(1)=Kmum*mum
          qm(2)=(4/3.*xm**2-1/3.)*mum*Kmum-4/3.*xm*lam*Emum
          do m=3,mmax
            nu=(4.*m-9.)/2.
            qm(m)=qm(m-1)*((2*nu+3)*(2*nu+1)*xm**2/((nu+2)*(nu+1))
&                         -(2*nu+3)*nu**2/((2*nu-1)*(nu+2)*(nu+1))
&                         -(nu+1)/(nu+2))
&               -qm(m-2)*(2*nu+3)*(nu-1)*nu/((2*nu-1)*(nu+2)*(nu+1))
          enddo
        else
          do m=1,mmax
            mm=2*(m-1)
            coefh=mycoefh(m)/(2*xm)**(mm+.5)
            yy=1./xm**2
            alpha=myalpha(m)
            sum=0.0
            do ir=1,irmax
              diff=mydcoefh(ir,m)*yy**(ir-1)
              sum=sum+diff
            enddo
            Fabcy=alpha*sum
            qm(m)=coefh*Fabcy
          enddo
        endif

        do m=1,mmax
          tmr(jj,kk,jjj,m)=coef*(qp(m)+qm(m))
        enddo

      enddo
    enddo
  enddo
endif

return
end
```

Appendix E: GRID.H

```
C-------------------------------------------------------------
C     GRID.H
C-------------------------------------------------------------

c This file contains a load of common blocks. Many of these should be
c removed from commons and confined to the subroutines where they are
c used so as to limit the use of memory.

      integer, parameter :: jmax2 = 512, kmax2 = 32, lmax = 128
      integer, parameter :: jmax1 = jmax2 - 1, jmax = jmax2 - 2
      integer, parameter :: kmax1 = kmax2 - 1, kmax = kmax2 - 2

      include 'proc.h'

      real, dimension (jmax2, kmax2, lmax) :: r,z,rhf,zhf
!hpf$  distribute  r(block,block,*) onto p2
!hpf$  align  z(i,j,k) with r(i,j,k)
!hpf$  align  rhf(i,j,k) with r(i,j,k)
!hpf$  align  zhf(i,j,k) with r(i,j,k)
      common /grid/    r,z,rhf,zhf

      real, dimension (jmax2, kmax2, lmax) :: rplus, zplus, rhfminus,zhfminus
!hpf$  align  rplus(i,j,k) with r(i,j,k)
!hpf$  align  zplus(i,j,k) with r(i,j,k)
!hpf$  align  rhfminus(i,j,k) with r(i,j,k)
!hpf$  align  zhfminus(i,j,k) with r(i,j,k)
      common /jgrid/ rplus, zplus, rhfminus, zhfminus

      real, dimension (jmax2, kmax2, lmax) :: phi, rho
!hpf$  align  phi(i,j,k) with r(i,j,k)
!hpf$  align  rho(i,j,k) with r(i,j,k)
      common /pois/    phi,rho

      integer :: itstep
      common /timst/   itstep

      real :: deltar, deltaz,dtheta
      common /jgrid2/ deltar, deltaz,dtheta

      real :: pi, grav
      common /blok6b/  pi,grav

!hpf$ processors p2(8,4)
```

Appendix F: Makefile

```
OFILE_DIR= obj

HPFFILES= main.hpf setup.hpf

FFILES= tm.f sm.f elle.f ellf.f rd.f rf.f gammln.f factrl.f

.SUFFIXES : .hpf

OFILES1= $(HPFFILES:.hpf=.o)

OFILES2= $(FFILES:.f=.o)

OFILES= $(OFILES1) $(OFILES2)

main:$(OFILES)
pghpf -O3 -o /home/hcohl/isyma1/main $(OFILES) ;

.hpf.o: $(HPFFILES)
pghpf -O3 -Mextend -Mreplicate=dims:3 -Moverlap=size:1 -c $<

.f.o:  $(FFILES)
f90 -c -dp -O3 -N 132 $<

cleanall:
/bin/rm -f *.o *.f
```

Vita

Howard S. Cohl was born in Paterson, New Jersey, on April 30, 1968. At the age of 18, Howard attained the rank of Eagle Scout in the Boy Scouts of America. He graduated from Indiana University, Bloomington, Indiana, with a bachelor of science degree in Astronomy and Astrophysics in 1990. He then worked for two years as a research assistant at the National Solar Observatory in Sunspot, New Mexico. In 1992, he began his graduate career in the Department of Physics and Astronomy at Louisiana State University and A&M College (L.S.U.), Baton Rouge, Louisiana. He obtained a master of science degree in physics from L.S.U. in 1994. He expects to receive the degree of Doctor of Philosophy in physics at the end of the summer of 1999.