The Art of Computational Science

The Kali Code

vol. 11

Initial Conditions:

Plummer's Model

Piet Hut and Jun Makino

September 14, 2007

Contents

Preface

This whole volume is dedicated to a detailed study of Plummer's model. We could have called it 'Plummer for dummies,' because the explanations and derivations given here are far more extensive than is usually the case in text books. Even so, the prefered type of 'dummies' are those that are happy to learn about Abel integral equations and stuff like that.

We present all results in such an easy-going style, that you don't even need pen and paper to follow the argument. The main reason for this approach is that we want to be realistic in presenting the discussion as a dialogue. Whereas text books often state that 'it can be easily seen that', in practice it may well take quite a while, and a number of pages of paper, before the reader finally has seen what was supposed to be easy. In contrast, when two people talk in front of a blackboard, or bend over a note pad, they don't follow such a clinical approach. They are more likely to write things out in long-hand, and that is exactly what our main characters do in this volume.

The main purpose for treating Plummer's model with such respect is that it forms a convenient example for showing how to construct models in phase space. Building a structure in six dimensions, rather than three, is something that you have to get used to. By the time you are really experienced doing so, it is easy to forget how counter-intuitive it probably looked, the first time you gave it a try. Our approach here is to present much of the theoretical argumentation that goes into model building not in an abstract way, but rather with the concrete example of Plummer's model, and specifically with the even more concrete question: here is a computer screen, and here is the keyboard, now sit down and create a star cluster.

It is this hands-on approach that Alice and Bob continue to pursue, as they have done in all previous volumes. In future volumes, they will no doubt come back to construct King models, as well as anisotropic and multi-mass generalizations. They will be able to move along much faster in those cases, having mastered the main principles while juggling Plummer's model. We hope you will enjoy the beauty and elegance of what is the most venerable star cluster model, with a history spanning now more than 120 years.

0.1 Acknowledgments

Besides thanking our home institutes, the Institute for Advanced Study in Princeton and the University of Tokyo, we want to convey our special gratitude to the Yukawa Institute of Theoretical Physics in Kyoto, where this volume was written, during a visit in June 2004, made possible by the kind invitations to both of us by Professor Masao Ninomiya.

We thank Shawn Slavin, Jason Underdown and Marcel Zemp for their comments on the manuscript.

Piet Hut and Jun Makino

Kyoto, July 2004

Chapter 1

A Simple Equilibrium Model

1.1 Time Scales

Alice: Hi Bob, I guess we are now getting close to do some real stellar dynamics of star clusters!

Bob: I just had the same thought. We now have a real N-body code, and it is high time that we use it for what it is designed to do: to follow the evolution of a collisional star system.

Alice: I prefer to avoid the term 'collisional,' it is just too misleading. Students will think that it implies real physical collisions between stars.

Bob: But it is an old and established term. It won't be easy to avoid it. But I agree, it is misleading. It only means that two-body encounters are important. In other words, two-body relaxation plays an important role in collisional systems, whereas you can neglect such relaxation effects in collisionless systems. In yet other words, in collisional systems, there is a significant heat flow through the system.

Alice: Yes. In stellar evolution they talk about a thermal time scale, like the Kelvin-Helmholtz time scale, a hundred million years or so for a star like the sun. This is the time scale at which a star will lose most of its energy from its surface, if the energy would not be replenished by nuclear reactions in the core of the star. In contrast, the dynamical time scale for a star like the sun is only a couple hours. This is the time it takes a star to 'ring', like a drum: the time to cross the star at the speed of sound.

Bob: That is a nice analogy. For a star cluster the dynamical time is called the crossing time. The speed of sound in a star cluster, like in a gas of molecules,

is of the order of the speed of the constituent particles.

For a typical globular cluster, with half-mass radius of the order of 10 parsec, and a velocity dispersion of order 10 km/sec, the crossing time is of order a million years – more than a billion times larger than the dynamical time scale for a typical star.

But the thermal time scale for a globular cluster, the time to redistribute the energy through the collective effects of the diffusion caused by two-body relaxation, is not that much larger than that of the sun: typical values for globular clusters are a billion years, only a factor ten larger than the Sun's Kelvin-Helmholtz time scale.

Alice: Coming back to the term 'collisional stellar dynamics', I wonder why gravitational encounters were called 'collisions.' One reason may be the analogy with molecular diffusion, where the van der Waals forces between molecules drop off so fast with distance that the only significant encounters are the ones where the molecules practically touch. Also, many of the early simulations of star cluster evolution were Monte Carlo simulations, and there the effects of two-body relaxation are modeled as discrete scattering events, with each star going its own merry way until scattered into a different orbit – like molecules in a gas.

Bob: Another reason may have been the fact that there was no competition for the term 'collisions': it was only in the nineteen-nineties that people had enough computer power to begin to treat collisions between stars in a serious and quantitative way. Back in the sixties, when they talked about collisional stellar dynamics, stars were not supposed to collide, because computers were not up to it yet.

Alice: You may have a point there. In any case, I prefer the term 'dense stellar systems' for star systems where encounters are important. Whether the encounters are merely gravitational or also involve occasional physical collisions is less important a distinction.

Bob: Someone else could argue that having collisions or not is the most fundamental distinction. Certainly someone specializing in hydrodynamics is likely to think so. I guess you just betrayed your bias to stellar dynamics. Oh well, classification will also cause heated debates.

Alice: Which I'd rather avoid. Anyway, we're setting out to simulate dense stellar systems, and we know that they have the tendency to show the effects of two-body relaxation: mass segregation, escapers, core collapse, all that good stuff.

Bob: Now that we have an N-body code, let's put up a good show!

1.2 Initial Conditions

Alice: But every show needs a stage that needs to be set up first. What shall

we choose for the initial conditions.

Bob: There are many options. Basically, we can just sprinkle particles into a more or less localized region, and in a few crossing times the initial transients will die down. The remaining system, after the fast particles have escaped, will then slowly undergo core collapse.

So we could start with a homogeneous sphere, with a constant density out to a certain radius, and zero density outside that radius. We could give the particles velocities in accord with the virial theorem, or we could even give each star zero velocity: from such a cold collapse, too, you quickly will get a damped remnant, and you will lose less than half of the particles.

Alice: True, but all those solutions are not very elegant. Besides, there is no good reason to pick one over the other. But what is worse, I don't like to mix the transient dynamical effects with the longer-lasting thermal effects. Remember, for a star cluster with a hundred thousand stars there is not much more than a factor of a thousand difference between the crossing time and the two-body relaxation time. And if we want to play with small simulations of only a few hundred stars, the ratio goes down to ten or less. I much prefer to start with a system that is already in dynamic equilibrium to start with.

Bob: Well, we could start with a King model.

Alice: That would be much better, yes, but still we would have to pick a number, such as the central concentration or the depth of the potential well in dimensionless units, in order to settle upon a particular King model, given that there is a one-dimensional family of models.

Bob: What would you prefer?

Alice: How about good old Plummer's model?

Bob: That one? But that's not very realistic!

Alice: At this point I don't care too much about how realistic our simulations will be, if you mean with realistic that the distribution of stars will resemble that of a globular cluster. First of all, Plummer's model does do a reasonable job of fitting some of the more loose clusters, those with a large core radius.

After all, Plummer got his name attached to the Schuster model, in 1911, because he showed that it could be used well to fit the observed cluster data available at that time. While he specifically referred to Schuster's 1883 publication, in which the model was first derived, it is called Plummer's model because of the astrophysical relevance. Sure, we can do better now, but in 1911 Ivan King wasn't even born yet. And you could still travel around in Europe without the need of a passport.

Bob: Those were the days, I suppose. But your choice of N-body simulations were limited to $N = 2$. I'm glad I'm alive now.

Alice: What I like about Plummer's model, in comparison to King Models, is that: 1) it is one well-defined model, rather than a whole family of models,

so that if two people simulate Plummer's model, they know that they talk about the exact same model; 2) it is a simple model, where everything can be expressed in terms of analytic expressions, which is not the case for King models; 3) most of the venerable early investigations of star cluster dynamics started with Plummer's Model for the initial conditions, so it is easy to make a connection with the literature.

Bob: From an educational point of view, yes, all three aspects carry some weight. But on the other hand, I always like to introduce students quickly to the more dirty nitty-gritty of actual research. And certainly nowadays you will find far more star cluster simulations starting from King Models than from Plummer's model.

Alice: Okay, let's do both. But if so, it really does make sense to start with Plummer's model. Since everything can be done analytically, the students will get a more direct insight into what is going on. After that, we can move on to King models, and whatever else we find time for.

Bob: Fine with me, since I don't feel as strongly about it as you seem to do. Where shall we start? It would be good to give the students a brief handout with some of the basic facts of Plummer's model. All that I remember about it is that a particle that acquires softening morphs from a point particle with a delta function mass distribution to that of Plummer's model.

Alice: Ah, Great! I knew there was a fourth point I could have mentioned to argue for Plummer's model as a favorite starting point: students are likely to already have encountered it without reflecting on it, when they have implemented, or at least used, particles with a softened potential. Thank you!

Bob: You're welcome, even though you didn't need a point 4), since I had already given in. Okay, let me write down what I remember and what I can easily derive. You, as a champion of Plummer's model, can then add whatever analytical elegance you like.

Alice: Go right ahead!

1.3 A Potential-Density pair

Bob: Okay. A model that is in dynamical equilibrium is defined by a potential-density pair that corresponds to a distribution function in phase space that is time-independent. I would have to scratch my head a bit, in fact quite a bit more than a bit, to remember how you derive the distribution function for a given potential-density pair, to show that an equilibrium solution exists.

Alice: We definitely have to prove that, but let's leave that for later.

Bob: Good! There is only so much I can do from scratch.

Alice: But you have to define what you mean by a distribution function. For

a star cluster, it is the density of stars in phase space.

Bob: Yes. Okay, here is the softened potential:

$$\Phi(r) = -GM\frac{1}{(r^2+a^2)^{1/2}} \tag{1.1}$$

Dr. Schuster's softened star system, for short, as you just told us, aka Dr. Plummer's patented potential. And now we have to call upon Dr. Poisson to provide us with the density that corresponds to this potential.

Alice: But first you have to remind the students of the symmetry assumptions that go into this type of construction.

Bob: Ah yes. We assume spherical symmetry in space, in other words, both potential $\Phi(\mathbf{r})$ and density $\rho(\mathbf{r})$ only depend on the distance to the center r, independent of the spherical angles θ, ϕ:

$$\Phi(\mathbf{r}) = \Phi(r)\,; \tag{1.2}$$

$$\rho(\mathbf{r}) = \rho(r). \tag{1.3}$$

In addition, we assume that the velocity dispersion is isotropic. In general, spherical symmetry in configuration space (or position space) still allows cylindrical symmetry in velocity space: the distribution function must be a function of energy per unit mass E and angular momentum per unit mass J. However, by insisting on isotropy we assert spherical symmetry in velocity space as well, which implies:

$$f(E,J) = f(E). \tag{1.4}$$

By the way, the potential $\Phi(r)$ is also the potential energy per unit mass – just like the potential in an electrostatic field, where it is the binding energy per unit charge. To be really specific, take a star with mass m at a a position in a star cluster at a distance r from the center. In order to let that star escape from the cluster, you have to give it a kinetic energy $E_{kin} = \frac{1}{2}mv^2$ that is exactly equal to the binding energy of that star with respect to the cluster, namely the absolute value of the potential energy of that star corresponding to its position in the gravitational well of the cluster: $|E_{pot}| = m\,|\Phi(r)| = -m\Phi(r)$.

So this means that the escape velocity for a star at radial distance r from the center of the cluster can be derived as follows:

$$\tfrac{1}{2}mv_{esc}^2 + m\Phi(r) = 0 \quad \Rightarrow \quad v_{esc}(r) = \sqrt{-2\Phi(r)} \tag{1.5}$$

Alice: Good idea to make that very clear, since when you talk about potential and potential energy, it is at first quite easy to get confused between the potential

energy of the cluster as a whole and between that of individual particles with respect to the rest of the cluster.

Bob: Well, we don't want to get confused, do we.

Alice: Aha, if you put it that way, I can't resist pointing out another possible confusion. When you wrote down the condition for isotropy, you were right, speaking as a physicist, although a mathematician looking over your shoulder would be greatly confused. Using the same symbol for two expressions that have a different functional form, and what is worse, even have a different number of parameters is definitely a no-no in mathematics.

Bob: But what else could I have written but $f(E, J) = f(E)$?

Alice: A proper mathematical expression would have been $\hat{f}(E, J) = f(E)$, for example. This would imply that $\hat{f}$ as a function is very different from the function f. The only connection is that for any choice of a particular value for E and a particular value for J, the relationship $\hat{f}(E, J) = f(E)$ would hold, independently of the value of J.

Bob: Hmm. I prefer to say $f(E, J) = f(E)$. That makes sense and it feels good. As Janis Joplin would have said, 'feeling good is good enough for me.'

Alice: It shows you're a physicist. Just be gentle with the occasional student who may think deeper about these issues than you; she or he may actually have a very good reason to be bothered by the usual glossing over of these questions. After all, it is easy to make mistakes – for example, mistakes in normalization, if you're not careful about what depends in which way on what.

Bob: I suppose you would have wanted me to put hats on the first two equations as well: $\hat{\rho}(\mathbf{r}) = \rho(r)$ and $\hat{\Phi}(\mathbf{r}) = \Phi(r)$.

Alice: Strictly speaking, yes, but once you have made that point, I prefer to then drop the hats, for simplicity.

Bob: Is that called having your cake and eating it, first confusing me by putting a hat on, and then dropping your hat?

Alice: Maybe that's where the expression 'I'll eat my hat' comes from.

1.4 Density

Bob: No comment. Let me go on, and derive the density, given the potential.

To find the density needed to generate a potential, all we have to do is solve Poisson's equation, which in spherical coordinates, under the assumption of spherical symmetry for the potential, takes the relatively simple form

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\Phi(r)\right) = 4\pi G\rho(r) \tag{1.6}$$

Working out the derivatives, this boils down to:

$$\frac{d^2}{dr^2}\Phi(r) + \frac{2}{r}\frac{d}{dr}\Phi(r) = 4\pi G\rho(r) \tag{1.7}$$

It is straightforward to work out the first derivative

$$\frac{d}{dr}\Phi(r) = GM\frac{r}{(r^2+a^2)^{3/2}} \tag{1.8}$$

and the second derivative of the potential

$$\begin{aligned}\frac{d^2}{dr^2}\Phi(r) &= GM\left\{\frac{1}{(r^2+a^2)^{3/2}} - \frac{3r^2}{(r^2+a^2)^{5/2}}\right\} \\ &= GM\frac{-2r^2+a^2}{(r^2+a^2)^{5/2}}\end{aligned}$$

Putting it all together we get:

$$\begin{aligned}\rho(r) &= \frac{1}{4\pi G}\left\{\frac{d^2}{dr^2}\Phi(r) + \frac{2}{r}\frac{d}{dr}\Phi(r)\right\} \\ &= \frac{M}{4\pi}\left\{\frac{-2r^2+a^2}{(r^2+a^2)^{5/2}} + \frac{2}{(r^2+a^2)^{3/2}}\right\} \\ &= \frac{M}{4\pi}\frac{3a^2}{(r^2+a^2)^{5/2}}\end{aligned} \tag{1.9}$$

Alice: It might have been a bit quicker to substitute $\Phi(r)$, given in eq. (1.1), directly in eq. (1.6).

Bob: Probably. But I like to be systematic, and it was nice to get the force on a unit mass, $-d\Phi(r)/dr$, for free.

So now we can also see what the physical interpretation was of the softened particles that we have been using occasionally: the potential–density pair of a softened point particle is Plummer's pair:

$$\left\{\begin{aligned}\Phi(r) &= -GM\frac{1}{(r^2+a^2)^{1/2}} \\ \rho(r) &= \frac{3M}{4\pi}\frac{a^2}{(r^2+a^2)^{5/2}}\end{aligned}\right. \tag{1.10}$$

Alice: That looks right. It has the right dimensions: energy for Φ and mass per cubic length, in other words, mass divided by volume for ρ.

Bob: To get more of a feeling for the behavior of these two functions, I find it helpful to factor out the radial distance squared in units of the structural length a squared:

$$\begin{cases} \Phi(r) &= -\dfrac{GM}{a}\left(1+\dfrac{r^2}{a^2}\right)^{-1/2} \\ \\ \rho(r) &= \dfrac{3M}{4\pi a^3}\left(1+\dfrac{r^2}{a^2}\right)^{-5/2} \end{cases} \tag{1.11}$$

or the opposite: the inverse radial distance squared in units of the structural length squared:

$$\begin{cases} \Phi(r) &= -\dfrac{GM}{r}\left(1+\dfrac{a^2}{r^2}\right)^{-1/2} \\ \\ \rho(r) &= \dfrac{3Ma^2}{4\pi r^5}\left(1+\dfrac{a^2}{r^2}\right)^{-5/2} \end{cases} \tag{1.12}$$

1.5 Physical Interpretations

Alice: Yes, that makes it easy to see immediately how the functions behave for small and for large radii. In fact, to bring out the physics even more, I suggest that for your first choice, you introduce the central potential $\Phi_o = \Phi(0)$ and the central density $\rho_o = \rho(0)$:

$$\begin{cases} \Phi(r) &= \Phi_o\left(1+\dfrac{r^2}{a^2}\right)^{-1/2} \\ \\ \rho(r) &= \rho_o\left(1+\dfrac{r^2}{a^2}\right)^{-5/2} \end{cases} \tag{1.13}$$

with

$$\begin{cases} \Phi_o &= -\dfrac{GM}{a} \\ \\ \rho_o &= \dfrac{3M}{4\pi a^3} \end{cases} \tag{1.14}$$

It is then clear right away how the softening works: instead of the infinitely deep potential well of a point mass, the bottom of the well corresponds to the surface potential of an object with mass equal to the total mass of the particle, and radius equal to the structural length. Similarly, the central density is exactly the density that such an object would have. It is nice that it all corresponds to such a simple picture, not only in order of magnitude, but even without any correction factor. That makes it easy to remember the formulas in this way.

Bob: Ah, yes, that is an even nicer way to write things. But I don't like to remember formulas, easy or not! As long as I keep my notes, I can always quickly look them up.

Alice: Or, in practice, rederive them. Don't try to make me believe that you, a), actually keep all your notes, and, b), have a way to find them when you need them!

Bob: Hmmm, well, yes, I'm afraid I often fail at both.

Alice: So do I, and most everyone I know. Actually, this in and of itself is already a good reason for us to write these notes out in book form, as we have started doing. And the more detail, the better: I hate having to spend a few pages of pen and paper work in order to get from one line to the next in a text book. Back in the days that printed paper was expensive, that might have been a good thing, but I prefer to save time, rather than paper. Actually, we're saving paper too, when we just put it all up on the net. So let's keep these notes, in full detail, just the way we're deriving them, in real time.

Bob: Fine with me. And I must admit, since we started writing these notes, I've gone back to them already several times, to look up things that I knew I knew just a few weeks ago, but which had started to slip out of my working memory.

Coming back to your recasting of the potential-density expressions, I suppose we can do something similar for my second choice, but there we are dealing with quantities at infinity, for which the a^2/r^2 vanishes.

Alice: Indeed, but instead of constants, we are now dealing with functions, basically because we cannot reach infinity, while we can reach the center, at zero distance. What you have factored out are the asymptotic behaviors of the function when you go out to very large radii;

$$\begin{cases} \Phi(r) &= \Phi_\infty(r)\left(1+\dfrac{a^2}{r^2}\right)^{-1/2} \\ \\ \rho(r) &= \rho_\infty(r)\left(1+\dfrac{a^2}{r^2}\right)^{-5/2} \end{cases} \tag{1.15}$$

What you have defined, effectively, are the asymptotic form of the potential $\Phi_\infty(r)$ as the Kepler potential of a point mass, and the asymptotic form of the

density $\rho_\infty(r)$ as the leading power law function, proportional to the inverse fifth power of the radius:

$$\begin{cases} \Phi_\infty(r) &= -\dfrac{GM}{r} \\ \\ \rho_\infty(r) &= \dfrac{3Ma^2}{4\pi r^5} \end{cases} \tag{1.16}$$

Bob: I thought I had done something interesting, even though I was mainly doodling, enjoying the equations I had derived by looking at them from different directions. But pretty as they may be, I feel it's time to actually construct a model along Plummer's line. Let me see what I can do, before we next see each other again.

Chapter 2

A Minimal Code

2.1 A Classic Recipe

Alice: Did you start building a star cluster, using Plummer's model as a blueprint?

Bob: Yes, here is a quick and dirty version that I cobbled together. It was a lot easier than I expected, since I stumbled upon a nice recipe, which must be a classic in stellar dynamics. I followed it line by line, in my implementation. It was published in the paper *A comparison of Numerical Methods for the Study of Star Cluster Dynamics*, by Sverre Aarseth, Michel Henon, and Roland Wielen, which appeared ages ago, in 1974, in Astron. Astroph. **37**, 183.

Alice: I remember reading that paper as a student.

Bob: You are *that* ancient?

Alice: Not quite. I must have just started high school at that time. I mean that I read it as an undergraduate when I wanted to do a small N-body project for my final thesis. My adviser recommended it. And indeed, it was an influential paper by some of the masters in the field, around that time.

I remember it well: the paper made the first detailed quantitative comparison between the Monte-Carlo Fokker-Planck method and direct N-body integration. They even showed the results for runs with a few hundred particles. Hard to believe that they got so much computing done, given the slow speed of computers at that time.

Bob: That is impressive, given that computers have increased their speed by a factor of ten every five years or so. That means that in 2004 computers are a million times faster than what they had available. I bet that my camera is a lot more powerful than the computers in their university centers.

Alice: And I'm sure your camera has way more storage as well. But what was

actually most impressive was how much information they were able to squeeze out of their calculations, given their limited resources. Anyway, why did you bring up this paper?

Bob: Because it contains a nifty recipe for constructing a Plummer's model.

Alice: Hey, I didn't remember that. Really?

Bob: They wrote it as an appendix. Everything was done analytically, except for the fact that they used a rejection technique at the last step, to get the velocities.

Alice: I must not have looked at the appendix. Well, that is convenient. Can you show me your code?

2.2 Classes

Bob: Here it is, and I warned you, I haven't commented it or cleaned it up: I just translated their recipe straight from the appendix into Ruby.

There are three parts to the file `mkplummer1.rb`. The first part contains the class definitions. It is rather short, since I realized I could do everything with stripped-down versions of the `Body` class and the `Nbody` class:

```
require "acs"

class Body

  attr_accessor :mass, :pos, :vel

  def initialize(mass = 0, pos = Vector[0,0,0], vel = Vector[0,0,0])
    @mass, @pos, @vel = mass, pos, vel
  end

end

class NBody

  attr_accessor :time, :body

  def initialize(n = 0)
    @body = []
    for i in 0...n
      @body[i] = Body.new
    end
  end
```

```
end
```

Alice: That all looks familiar. So you only need to create an N-body system, and then print it out.

2.3 Where the Work is Done

Bob: Indeed. The actual work is done here:

```
include Math

def frand(low, high)
  low + rand * (high - low)
end

def mkplummer(n, seed)
  if seed == 0
    srand
  else
    srand seed
  end
  nb = NBody.new(n)
  nb.body.each do |b|
    b.mass = 1.0/n
    radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
    theta = acos(frand(-1, 1))
    phi = frand(0, 2*PI)
    b.pos[0] = radius * sin( theta ) * cos( phi )
    b.pos[1] = radius * sin( theta ) * sin( phi )
    b.pos[2] = radius * cos( theta )
    x = 0.0
    y = 0.1
    while y > x*x*(1.0-x*x)**3.5
      x = frand(0,1)
      y = frand(0,0.1)
    end
    velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
    theta = acos(frand(-1, 1))
    phi = frand(0, 2*PI)
    b.vel[0] = velocity * sin( theta ) * cos( phi )
    b.vel[1] = velocity * sin( theta ) * sin( phi )
    b.vel[2] = velocity * cos( theta )
  end
```

```
   STDERR.print "             actual seed used\t: ", srand, "\n"
   nb.acs_write
end
```

The `mkplummer` method takes two arguments: `n`, which is the number of stars in our star cluster, and `seed`, which is the seed for the random number generator.

If you want to create a different model, each time you invoke `mkplummer`, you can choose `seed = 0`. That will invoke `srand` without any argument, as you can see from the first two lines. Ruby guarantees that such a call to `srand` will return a different seed each time.

However, if you do want to reproduce the same result, you can give a specific seed by choosing `seed = k`, where `k` is a positive integer. Calling `srand` with that number will actually use that seed. This will be useful when you are debugging the `mkplummer` code itself, or when you are debugging another code, that takes the `mkplummer` results as input.

I'm a bit puzzled about one thing, though: the manual tells me that the call `srand 0` would have the same effect as calling `srand` with nu arguments. This would suggest that I could replace the first four lines of `mkplummer` by

```
   srand seed
```

without having to include the `if...else...` statements. However, when I did that, I got the same result each time. Well, perhaps my manual is outdated. In any case, with this construction, it worked.

After seeding the random number generator, I create a new N-body system with `n` particles, and then I enter a loop, in which I initialize each of those particles in turn.

Alice: Let's look at that loop later. I would like to see the overall structure of the code first.

Bob: Okay: after the loop finishes for the last particle, I print out the random number seed that was used, as a check to make sure that we can later recreate the same model realization.

Alice: Why would you do that? You already gave the seed value as an argument.

Bob: If I give a positive value, then indeed this is superfluous. But if I give a seed value of zero, the system chooses a pseudo-random seed for me. And if I would not echo the seed, I would not be able to reproduce the run later on.

Alice: I see. But why do you call `srand` again in the print statement?

Bob: Because Ruby tells me that the value that `srand` returns is the value of the *previous* seed. Don't ask why. It is defined that way. So by calling `srand` again, I get the value of the seed that I have to give the next time, in order

to reproduce the previous run, even if at that time I gave a value zero for the second argument of `mkplummer`. It works: I tested it out.

Alice: I take your word for it. Can you show me how you invoke `mkplummer`?

2.4 The Driver

Bob: Here it is:

```
options_text= <<-END

  Description: Plummer's Model Builder
  Long description:
    This program creates an N-body realization of Plummer's Model.
    (c) 2004, Piet Hut and Jun Makino; see ACS at www.artcompsi.org

    The algorithm used is described in Aarseth, S., Henon, M., & Wielen, R.,
    Astron. Astroph. 37, 183 (1974).


  Short name:           -n
  Long name:            --n_particles
  Value type:           int
  Default value:        1
  Variable name:        n
  Print name:           N
  Description:          Number of particles
  Long description:
    Number of particles in a realization of Plummer's Model.

    Each particles is drawn at random from the Plummer distribution,
    and therefore there are no correlations between the particles.

    Physical Units are used in which G = M = a = 1, where
      G is the gravitational constant
      M is the total mass of the N-body system
      a is the structural length, with potential U(r) = GM/(r^2 + a^2)^{1/2}


  Short name:           -s
  Long name:            --seed
  Value type:           int
  Default value:        0
  Description:          pseudorandom number seed given
  Print name:
```

```
  Variable name:      seed
  Long description:
    Seed for the pseudorandom number generator.  If a seed is given with
    value zero, a preudorandom number is chosen as the value of the seed.
    The seed value used is echoed separately from the seed value given,
    to allow the possibility to repeat the creation of an N-body realization.

      Example:

        |gravity> kali mkplummer1.rb -n 42 -s 0
        . . .
        pseudorandom number seed given  : 0
                    actual seed used    : 1087616341
        . . .
        |gravity> kali mkplummer1.rb -n 42 -s 1087616341
        . . .
        pseudorandom number seed given  : 1087616341
                    actual seed used    : 1087616341
        . . .


  END

c = parse_command_line(options_text)

mkplummer(c.n, c.seed)
```

It is mostly command line argument parser, using our nifty new `Clop` class based parser. It contains the 'here document' that defines the options, followed first by the command `parse_command_line` that does what it says it does, and then by the command `mkplummer`.

2.5 The Basic Idea

Alice: Let us now look at the inner loop of `mkplummer`, where each particle is given its initial values for its mass, position, and velocity. The first line is simple:

```
b.mass = 1.0/n
```

Each particle acquires a mass that is $1/n$: you are creating an equal-mass system, where the total mass is normalized to be unity.

Bob: Indeed. It would be possible to give a mass spectrum, of course, but that can be done later, when we are ready for that. For now, I just wanted to construct a minimal model.

Alice: Fair enough. Then you pick values for position and velocity components. What is the basic idea here?

Bob: The idea is to proceed in two steps. You start by sprinkling particles in space, as a random realization of the mass density distribution of Plummer's model. This means that you have to be careful to determine the radial position of the particles with the right statistical weight. Because of spherical symmetry, the angles can be randomly chosen from a two-dimensional spherical surface.

The second step is to give each particle a velocity with a random direction and a magnitude that is also random, but drawn from the appropriate velocity distribution at that point in space.

Alice: In other words, you really are sprinkling particles into phase space, the six-dimensional space that is the direct product of configuration space and velocity space.

Bob: Yes, but even though you can look at phase space as a single six-dimensional space, that still leaves the fact that there is the three-by-three structure left for the two quite different subspaces.

What I mean is: you have to start by picking an appropriately random point in what you call configuration space, and what is normally just called 'space', containing all possible positions in three dimensions. Only after you know the position of a particle in that subspace, can you determine the potential energy of that particle. And only when you know the potential energy, can you know what type of velocities are admissible, in order to keep the particle bound and to give each velocity the correct statistical weight.

To sum up, the order of picking a point in configuration space, and then picking a point in velocity space, is important: you couldn't do it the other way around.

Alice: Ah, that must correspond to what mathematicians mean when they call the velocity space the tangent bundle to the configuration space. Each point in configuration space has its own tangent space, and it is only because we work in Newtonian flat space that we can pretend that phase space is a single six-dimensional space, shared by all particles.

Bob: Whatever. Enough mathematical terms! Let me show you how the particles get sprinkled into normal space first, and how we then populate the velocity space. In practice, rather than doing all the configuration space sprinkling first, followed by a second loop in which we populate velocity space, I find it easier to do everything in one loop. For each particle I determine the three position coordinates, and with that information I can then immediately give the three velocity coordinates.

2.6 Sprinkling Particles in Space

Alice: Okay, lets look at configuration space first. I see that you invoke some magical expression to determine the value of the radius:

```
radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
```

Bob: Yes, this is part of the recipe provided by Aarseth *et al.*, for choosing a correctly randomized `radius`, the distance of the star from the center of the star cluster.

Alice: Let's look at that in a moment, after we've gone through the body of the loop first. Given the value for `radius`, I see that you assign the three Cartesian coordinates of the star in the usual way from spherical coordinates `radius`, `theta`, and `phi`:

```
b.pos[0] = radius * sin( theta ) * cos( phi )
b.pos[1] = radius * sin( theta ) * sin( phi )
b.pos[2] = radius * cos( theta )
```

Bob: Yes, and randomizing the two angular coordinates was relatively simple. The value for ϕ, for example, is a random number between 0 and 2π, since every azimuthal angle is equally likely:

```
phi = frand(0, 2*PI)
```

By the way, here I have adapted the Ruby defined random number call `rand` in the following way, by defining a general floating point version `frand`:

```
def frand(low, high)
  low + rand * (high - low)
end
```

Since `rand` returns a value uniformly distributed throughout the range $\{0, 1\}$, `frand(a,b)` returns a value uniformly distributed throughout the range $\{a, b\}$.

Alice: And this is how you initialize the angle `theta` between the positive `z` axis and the position vector of your star:

```
theta = acos(frand(-1, 1))
```

Bob: To pick a random value for θ, we have to make sure that the spherical integration element $\sin\theta d\theta$ gets an equal weight for any θ value. In other words, any value for $d(\cos\theta)$ should be equally likely.

Now the highest and lowest values occur for $\theta = 0$, along the positive z axis, and for $\theta = \pi$, along the negative z axis. So $\cos\theta$ runs from $+1$ to -1.

All we have to do is to pick a floating point number at random, somewhere in the interval $\{-1, +1\}$. Let's call the number x. This determines a number y defined as $y = \arccos x$. By construction, the value of $\cos y = \cos(\arccos x) = x$ is uniformly random, as was desired for θ. Hence y has the right distribution of values for θ, and we can simply take $\theta = y$.

Alice: It sounds complicated when you say it in words, but I see what you mean. Indeed, that must be correct.

Bob: I sometimes think that anything to do with probability gets more confusing when you try to talk about it. Easy to fool others, and to fool yourself for that matter. No wonder people talk about lies, damn lies, and statistics!

2.7 Populating Velocity Space

Alice: Moving right along, we now come across something really mysterious:

```
x = 0.0
y = 0.1
while y > x*x*(1.0-x*x)**3.5
  x = frand(0,1)
  y = frand(0,0.1)
end
velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
```

Bob: Yeah, and fun too. But since you postponed a discussion of the proper weighting function for choosing the radial position, we should also postpone a discussion of what goes on here. Briefly, I am using a rejection technique, following Aarseth *et al.* in order to determine the magnitude `velocity` of the velocity vector, in the last line above.

Alice: Okay, I'm happy to wait till later. Now in the next lines you repeat the same spherical distribution trick you applied in order to find the position coordinates.

Bob: Indeed, and this gives me the velocity coordinates.

Alice: You could have put those five lines into a separate method. Remember the DRY principle: don't repeat yourself.

Bob: Good point. Let me do that in the next version of this code. This is a rather minimal one, and I can think already of several improvements. For example, we may want to recenter the center of mass of the star cluster we have created onto the center of the coordinates. We can also dampen some of the fluctuations we are introducing by layering the particles more evenly in radial

bins, rather than sprinkling every particle in space, independently of any other particle.

Alice: That's fine: I also like to start with a minimal script, so that we can really test and understand its behavior, before we start adding bells and whistles. Testing the first version of a new code is half the work. Once you have one well-tested version, you can use that to make comparisons, to check whether each new addition still reproduces the old result. The hardest part is to get an initial result, and make sure that that one is correct.

Chapter 3

Sprinkling Particles in Space

3.1 Choosing a Distance

Alice: Now I understand what your code is doing, except for a few crucial lines. First there is the one-liner about choosing the distance between a new star and the center of the star cluster:

```
radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
```

Can you tell me what this expression means, and how it is derived? It must somehow be related to the density distribution $\rho(r)$, which you have already derived from the potential. How exactly do Aarseth, Henon and Wielen use the density for particle sprinkling?

Bob: They describe their technique in a few words, and I had to read those words carefully, and do some head scratching, to figure out what it meant. But as always, once you see it, it is really easy. Let me try to summarize it in my own words.

First we introduce the cumulative mass distribution

$$m(r) = \int_0^r 4\pi r^2 \rho(r) dr \tag{3.1}$$

which is the amount of mass contained within the star cluster inside a distance r from the center. When we create a new star, and place it at radius r, that star will see an amount of mass $m(r)$ of the cluster at positions closer to the center than its own position, and therefore an amount of mass $M - m(r)$ at positions further from the center.

In other words, it will see a fraction $m(r)/M$ of the total mass inside its radial position. Now that fraction could be anything between 0 and 1. It will be 0 if the particle is placed exactly in the center, and it will approach 1 if the particle is placed very far away, reaching 1 when the particle is placed at infinity.

The ranking of each particle, in terms of the enclosed mass, is random and uniform in the mass fraction. In other words, $m(r)/M$ will be a random value between 0 and 1, with each value equally likely.

So here is the idea: spin a random number generator in order to obtain a a random number m_i, with $0 \leq m_i \leq M$, and we consider that to be the fractional mass contained within the radius r_i of particle i. So all we know is that $m(r_i) = m_i$, but what we need to know is r_i itself. So the procedure is to invert (3.1) to obtain a function $r(m)$, and then life is simple: $r_i = r(m_i)$.

3.2 Following the Recipe

Alice: That sounds straightforward. Can you show me the expressions you found for $m(r)$ and $r(m)$?

Bob: I simply took their expressions. They use a system of units in which the total mass M, the gravitational constant G and the structural length scale a that we used above are all unity. The mass enclosed within a radius r then becomes:

$$m(r) = r^3 \left(r^2 + 1\right)^{-3/2} \tag{3.2}$$

and the radius that corresponds to a mass fraction becomes:

$$r(m) = \left(m^{-2/3} - 1\right)^{-1/2} \tag{3.3}$$

As you can see in the second line of the inner loop in my `mkplummer` method, this is how I determine the radial position of each particle, using Ruby's random number generator `rand`:

```
radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
```

Alice: Hmm. You didn't check whether they had done their math correctly?

Bob: No need to. This is a paper by Aarseth, Henon, and Wielen. Besides, it is thirty years old and has been cited zillions of times by others. I'm sure this is a result that can be trusted.

Alice: I don't like to accept things on trust, no matter what the authority may be behind it. Not that I expect them to be wrong, I agree that that would be highly improbable. Still, I would feel much better if we derive the results

ourselves. Besides, if we work it out now, we can both use those notes when we have to teach it in class to the students. Better still, we just put it into the material we prepare for them on the web.

Bob: Okay, if you like. Your turn, though, I already derived the density. Do you want to use a package from symbolic integration? Differentiation is easy enough by hand, but I must admit, I'm a bit rusty in my integration.

3.3 Cumulative Mass

Alice: So am I, and that is a really good reason to do it with pen and paper, tempting as it is to use a symbolic package. Okay. I'll start with the density you derived:

$$\rho(r) = \frac{M}{4\pi}\frac{3a^2}{(r^2+a^2)^{5/2}} \tag{3.4}$$

By definition, this gives us for the cumulative mass, as a function of radius:

$$\begin{aligned} m(r) &= \int_0^r 4\pi r^2 \rho(r) dr \\ &= \int_0^r \frac{3M}{a^3} r^2 \left(1+\frac{r^2}{a^2}\right)^{-5/2} dr \end{aligned} \tag{3.5}$$

The variable r appears in the integrand only in terms of the combination r^2/a^2, so a natural change of variables is:

$$x = \frac{r^2}{a^2} \quad \Rightarrow \quad r = a\sqrt{x} \quad \Rightarrow \quad dr = \frac{a}{2}\frac{dx}{\sqrt{x}} \quad \Rightarrow$$

which gives us:

$$\begin{aligned} m(r) &= \frac{3M}{a^3}\int_0^{r^2/a^2} (a^2 x)(1+x)^{-5/2}\frac{a}{2}\frac{dx}{\sqrt{x}} \\ &= \frac{3M}{2}\int_0^{r^2/a^2} x^{1/2}(1+x)^{-5/2} dx \end{aligned}$$

It is easier to bring the total mass to the other side, as an expression for the fractional cumulative mass. I don't like the high power 5/2 in the integrand. I'll use integration by parts to lower the power:

$$\begin{aligned}
\frac{m(r)}{M} &= -\int_0^{r^2/a^2} x^{1/2}\frac{d}{dx}\left\{(1+x)^{-3/2}\right\}dx \\
&= -\int_0^{r^2/a^2} d\left\{x^{1/2}(1+x)^{-3/2}\right\} + \tfrac{1}{2}\int_0^{r^2/a^2}(1+x)^{-3/2}\frac{d}{dx}\left\{x^{1/2}\right\}dx \\
&= -x^{1/2}(1+x)^{-3/2}\Big|_0^{r^2/a^2} + \tfrac{1}{2}\int_0^{r^2/a^2} x^{-1/2}(1+x)^{-3/2}dx
\end{aligned}$$

That looks a bit better already. How about another change of variables:

$$y = \frac{1}{x} \quad \Rightarrow \quad x = \frac{1}{y} \quad \Rightarrow \quad dx = -\frac{1}{y^2}\,dy \quad \Rightarrow$$

This gives us:

$$\begin{aligned}
\frac{m(r)}{M} &= -\frac{r}{a}\left(1+\frac{r^2}{a^2}\right)^{-3/2} + \tfrac{1}{2}\int_\infty^{a^2/r^2} y^{1/2}\left(1+\frac{1}{y}\right)^{-3/2}\left\{-\frac{1}{y^2}\,dy\right\} \\
&= -\frac{r}{a}\left(1+\frac{r^2}{a^2}\right)^{-3/2} - \tfrac{1}{2}\int_\infty^{a^2/r^2} y^{-3/2}\left(1+\frac{1}{y}\right)^{-3/2}dy \\
&= -\frac{r}{a}\left(\frac{r^2}{a^2}\left(\frac{a^2}{r^2}+1\right)\right)^{-3/2} - \tfrac{1}{2}\int_\infty^{a^2/r^2}(y+1)^{-3/2}dy \\
&= -\frac{r}{a}\left(\frac{r}{a}\right)^{-3}\left(\frac{a^2}{r^2}+1\right)^{-3/2} + (y+1)^{-1/2}\Big|_\infty^{a^2/r^2} \\
&= -\frac{a^2}{r^2}\left(1+\frac{a^2}{r^2}\right)^{-3/2} + \left(1+\frac{a^2}{r^2}\right)^{-1/2} \\
&= \left(1+\frac{a^2}{r^2}\right)^{-3/2}\left\{-\frac{a^2}{r^2}+\left(1+\frac{a^2}{r^2}\right)\right\} \\
&= \left(1+\frac{a^2}{r^2}\right)^{-3/2} \qquad (3.6)
\end{aligned}$$

So here is what we were looking for:

$$m(r) \;=\; M\left(1+\frac{a^2}{r^2}\right)^{-3/2} \qquad (3.7)$$

Indeed eq. (3.2), with their choice of units.

3.4 Physical Intuition

Bob: Well, if you are rusty in your integrations, then I don't know what to call myself. Nice job! It is always surprising to me how the result of that type of calculation can come out in such a simple form.

Alice: There probably is a good physical reason for it to be this simple. Let's think. I started with density, something that you had found by differentiation, and then I integrated the product of the density and the geometric opening angle factor of $4\pi r^2$. Apart from that factor, integration and differentiation would have canceled. Pity.

Bob: Hey, wait a minute. I found the density by integrating alright, but in the following way, using Poisson's equation:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\Phi(r)\right) = 4\pi G\rho(r) \tag{3.8}$$

Doesn't that have exactly the factor r^2 you were looking for?

Alice: It does . . . Hey, I could have started there! I could have written:

$$\begin{aligned} m(r) &= \int_0^r 4\pi r^2\rho(r)dr \\ &= \frac{1}{G}\int_0^r \frac{d}{dr}\left(r^2\frac{d}{dr}\Phi(r)\right) \\ &= \frac{1}{G}\left.\frac{d}{dr}\left(r^2\frac{d}{dr}\Phi(r)\right)\right|_0^r \\ &= \frac{1}{G}\,r^2\frac{d\Phi}{dr} \end{aligned}$$

Bob: Ah, I remember telling you that it might come in handy to have the derivatives of the potential at hand.

Alice: Not only that, here is the physical meaning we were looking for! You also mentioned that the gradient of the potential is the gravitational force, apart from a minus sign. So what this equation is telling us, is simply that the physical force is proportional to mass and inversely proportional to the radius squared: Newton's gravity! We could have started that way. The magnitude of the force on a particle with mass m_p at distance r from the center is of course:

$$F = G\,\frac{m_p m(r)}{r^2}$$

and also equal to:

$$F \;=\; \left| -\, m_p \, \frac{d\Phi}{dr} \right| \;=\; m_p \, \frac{d\Phi}{dr}$$

since $d\Phi/dr \geq 0$ everywhere.

Bob: You're right. If we would have started with those two lines, we could have written

$$m(r) \;=\; \frac{1}{G}\, r^2 \frac{d\Phi}{dr}$$

right away. And with the expression I wrote down yesterday,

$$\frac{d}{dr}\Phi(r) \;=\; GM \frac{r}{\left(r^2 + a^2\right)^{3/2}}$$

this would have given us:

$$m(r) \;=\; \frac{1}{G}\, r^2 \frac{d\Phi}{dr} \;=\; M\, \frac{r^3}{\left(r^2 + a^2\right)^{3/2}} \;=\; M \left(1 + \frac{a^2}{r^2}\right)^{-3/2}$$

Alice: Quite a bit faster than my juggling of integrals! We could have used a healthy dose of physical intuition, before embarking on that lengthy computation.

3.5 An Intuitive Derivation

Bob: Now that we have found the radius dependence of the cumulative mass, we only have to invert that relationship, to get the dependence of cumulative mass on radius. That shouldn't be too hard. However, I'm getting tired of carrying along the factors M and a which we may as well consider to be the physical units used for measuring m and r, so that the latter are used as dimensionless parameters. Setting $M = a = 1$, we can write:

$$m(r) \;=\; r^3 \left(r^2 + 1\right)^{-3/2} \;\Rightarrow$$

$$m^{-2/3} \;=\; r^{-2} \left(r^2 + 1\right) \;\Rightarrow$$

$$r^2 m^{-2/3} \;=\; \left(r^2 + 1\right) \;\Rightarrow$$

$$r^2 \left(m^{-2/3} - 1\right) \;=\; 1 \;\Rightarrow$$

$$r(m) = \left(m^{-2/3} - 1\right)^{-1/2} \tag{3.9}$$

which is indeed eq. (3.3) that you asked me to derive. No need to trust anybody anymore! We have derived it from first principles.

Alice: But you'd better explain your students how you can restore the correct factors of M and a, otherwise they will think that you were cutting corners.

Bob: Good point. It takes a while to learn to think in terms of dimensionless quantities, and to transform easily and confidently between those and the corresponding physical quantities. In this case, I can just point out that the dimensionless quantity r has to be replaced by r/a, and the dimensionless quantity m has to be replaced by m/M. We then get:

$$r(m) = \frac{a}{\sqrt{\left(\frac{M}{m}\right)^{2/3} - 1}} \tag{3.10}$$

Alice: That is correct, but if I were your student, I would be rather surprised. I would argue that a piece of wood with a length of 1 foot has also a length of 12 inches. So you have to multiply the unit of length with the dimensionless number 1 or 12, as the case may be, to get the physical length. So I would ask: why are you *dividing* r by the length scale a ?

Bob: Yes, students do indeed ask such questions! It just means that they have to practice more with simple examples, until it comes to them naturally.

Alice: Well, that is not really answering the question. My answer would be to introduce two different sets of symbols, to remove the confusion between the physical quantities and the dimensionless quantities.

Bob: I won't stop you!

3.6 A Formal Derivation

Alice: If you define

$$\begin{cases} r &= \xi a \\ m &= \mu M \end{cases} \tag{3.11}$$

you can point out that r and a are physical quantities, while ξ is the dimensionless quantity connecting them. Similarly, m and M are physical quantities, while μ is the dimensionless quantity giving the variable quantity m in terms of the mass unit m. We can then write:

$$\begin{cases} \xi &= \dfrac{r}{a} \\ \mu &= \dfrac{m}{M} \end{cases} \tag{3.12}$$

This makes is possible to write your derivation without any shortcuts, in a mathematically precise way.

$$m = Mr^3\left(r^2+a^2\right)^{-3/2} \Rightarrow$$

$$\mu M = M(\xi a)^3\left((\xi a)^2+a^2\right)^{-3/2} \Rightarrow$$

$$\mu = \xi^3\left(\xi^2+1\right)^{-3/2} \Rightarrow$$

$$\mu^{-2/3} = \xi^{-2}\left(\xi^2+1\right) \Rightarrow$$

$$\xi^2\mu^{-2/3} = \left(\xi^2+1\right) \Rightarrow$$

$$\xi^2\left(\mu^{-2/3}-1\right) = 1 \Rightarrow$$

$$\xi(\mu) = \left(\mu^{-2/3}-1\right)^{-1/2} \Rightarrow$$

$$\frac{r(m)}{a} = \left(\left(\frac{M}{m}\right)^{2/3}-1\right)^{-1/2} \Rightarrow$$

$$r(m) = \frac{a}{\sqrt{\left(\frac{M}{m}\right)^{2/3}-1}}$$

Bob: Yes, that is full mathematical rigor. But of course, in practice, you wouldn't go to such extravagance. This is like what you were pushing for earlier, with your request of putting hats on all kind of variables, just because their mathematical functional form changed. Since I'm a physicist, I prefer to change notation only if the physical meaning of the variables change.

Alice: I must admit, I often do use this type of switching of variables, along the lines of what I just illustrated. I can see that you're comfortable with omitting that step, and that is mostly a matter of taste, I guess. Still, you can't insist or

wish that your students have the same quirks or abilities as you. So I suggest we at least add my derivation as well.

Bob: In that case I insist that we leave my derivation in too, for those younger versions of me who already got the physics, and don't want to accrete unnecessary mathematical niceties.

Alice: So be it.

Chapter 4

Populating Velocity Space

4.1 Choosing a Velocity

Alice: Now that we completely understand how you choose the radial distance for a particle, there is only one thing left to do: to choose the magnitude of its velocity. In your code you had a more complicated construction than the one-liner you used for the position. For the velocity you wrote:

```
x = 0.0
y = 0.1
while y > x*x*(1.0-x*x)**3.5
  x = frand(0,1)
  y = frand(0,0.1)
end
velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
```

You are throwing dice a number of times, until some criterion is satisfied, and then you move on to a new one-liner that gives you the value for the variable `velocity`, which is the magnitude of the velocity.

Bob: Here is the recipe for that part. Aarseth *et al.* start with the observation that the maximum velocity allowed at a radius r is the escape velocity $v_e(r)$, itself a function that depends on the radius. The escape velocity can be determined by requiring that a particle at radius r has exactly zero total energy, *i.e.* its kinetic energy is just enough for a parabolic escape to infinity. Since the potential energy for a test particle moving in Plummer's model is given as

$$\Phi(r) = -\frac{1}{(r^2+1)^{1/2}} \tag{4.1}$$

per unit mass of the test particle, we can equate that to the kinetic energy, also per unit mass, of a particle moving at the escape velocity:

$$\tfrac{1}{2}\left\{v_e(r)\right\}^2 + \Phi(r) = 0 \Rightarrow \tag{4.2}$$

$$\left\{v_e(r)\right\}^2 = 2\left(r^2+1\right)^{-1/2} \Rightarrow \tag{4.3}$$

$$v_e(r) = \sqrt{2}\left(r^2+1\right)^{-1/4} \tag{4.4}$$

This is the maximum velocity allowed at radius r, and we also know that the minimum velocity at radius r is zero. The question is: what is the probability distribution for $0 \leq v \leq v_e$.

Alice: I see that you are using again the choice of units given by Aarseth *et al.*, where $G = M = a = 1$.

Bob: Yes, they are by far the most convenient, they save time when writing down and manipulating the equations, and they make it also less likely to make mistakes.

Alice: I'm not sure about the last part of what you said. The advantage of keeping the full physical quantities is that you always have a few extra checks you can make, at the end: if the physical dimensions of length, mass and time are not exactly the same, at the left and right hand side of an equation, the equation must be wrong. If you work only with dimensionless quantities, you loose that advantage.

Bob: I don't consider it an advantage to have to do so much more work that you are likely to make more mistakes, so that you can then happily catch them. But clearly we are talking about matters of taste, and we have already decided we will present our results either way, now that we know exactly how to transform in both directions, between physical and dimensionless variables.

4.2 A Meta-Recipe

Alice: Yes, that is what we decided. How did you find the probability distribution for the velocities?

Bob: I started with the distribution function for the energy of the particles:

$$f(\mathbf{r},\mathbf{v})d\mathbf{r}d\mathbf{v} = f(E(r,v)\,4\pi r^2 dr\,4\pi v^2 dv = \frac{384\sqrt{2}}{7\pi m}(-E)^{7/2}r^2v^2 dr\,dv \tag{4.5}$$

Alice: Where did you get this expression from?

Bob: It's just what it is, for Plummer's model. I found it in a useful table in *The Gravitational Million-Body Problem*, by Douglas Heggie and Piet Hut [Cambridge University Press, 2003]. It is table 8.1 on p. 73, a whole page full of useful properties of Plummer's model. By the way, m here is the mass of a single star, assuming that all stars have equal mass. If you multiply both sides of the equation with m, you get fm, the mass density of stars in phase space.

Alice: It is a power law in energy, and it looks like a polytrope. Ah, yes, now I remember: Plummer's model is nothing else but a polytrope of index five. Polytropes are defined in general for index n through a distribution function:

$$f(E) \propto (|E|)^{n-3/2} \tag{4.6}$$

for negative energy, in other words for bound particles, while $f(E) = 0$ for $E \geq 0$, otherwise you would get the whole universe filled with escapers.

But it is not fair to use such remembered knowledge, or equations that you pluck from a book. Our whole approach is aimed at spelling out everything, both to help us in our research, to see new aspects we might otherwise have overlooked, and to help us in our teaching, to make things crystal clear to the students.

Bob: How do you expect to get new insight into the fact that the distribution function of Plummer's model is a seven-halves power of the energy? That fact will not change, no matter how long you stare at it.

Alice: That's not the point. Once we spell out in complete detail how you derive and verify all aspects of one recipe for one star cluster model, you can then follow that example approach to construct any other recipe for any other cluster model. In other words, we are using Plummer's model as a case study in order to present a meta-recipe for constructing recipes for constructing models for star clusters.

Bob: I had no idea we were doing something that fancy. But whatever words you want to hang on it, I cannot deny that it is a good thing to check things for yourself, and most importantly, I'll have to explain at least some of these things in class pretty soon, so okay, let's go through the derivation.

However, we would probably lose the thread of our argument if we would go into deriving eq. (4.5) right now. Let's postpone that a bit, and first see whether we can reconstruct what I have written in my program. As you can expect, here too I just followed the recipe from Aarseth *et al.*. Let us first see whether we can at least derive that recipe, assuming the validity of eq. (4.5).

Alice: Sounds like a good plan.

4.3 Following the Recipe

Bob: Here is what I have understood, so far, of the recipe. Given the distribution function for the energy, you have to transform that into an equation for the magnitude for the velocity. What makes life simpler, is the fact that we are comparing particles with different velocity choices at a given point, so we know that their potential energies are all the same.

This means that the probability $g(v)$ to have an absolute value for the velocity $v = |\mathbf{v}|$ at radial position $r = |\mathbf{r}|$ is given by

$$g(v)dv \propto (-E(r,v))^{7/2} v^2 dv \tag{4.7}$$

where the energy per unit mass $E(r,v) = \Phi(r) + \frac{1}{2}v^2$ can be written in terms of the escape velocity v_e as $E(r,v) = -\frac{1}{2}v_e^2 + \frac{1}{2}v^2$. If we introduce the variable

$$q = \frac{v}{v_e} \tag{4.8}$$

we can write $E(q) \propto q^2 - 1$, for a given fixed r. The distribution function for v then becomes, in terms of q, proportional to the following function:

$$g(q) = \left(1 - q^2\right)^{7/2} q^2 \tag{4.9}$$

Alice: And the range of admissible q values is $0 \leq q \leq 1$. This looks exactly like the problem we had for determining the radial positions. There we knew the density, *i.e.* the probability function to find a particle at a given position. By integrating the density we obtained the cumulative mass function $m(r)$, and then we inverted that to obtain $r(m)$.

So I guess the next step is to invert $g(q)$. However, that doesn't look so easy.

Bob: To say the least. Therefore, for the velocities, they choose a different approach. If you plot the function $g(q)$, then the height of that curve, for each q value, gives you the relative probability that q would lie in a region of small fixed width around that value.

You can imagine that you can obtain a distribution of the required weighting by throwing darts at that graph. If you hit a point somewhere above the graph, you throw a new dart, and you keep throwing new darts until you hit a point below the graph, anywhere between $q = 0$ and $q = 1$. If you follow that procedure, you are automatically guaranteed that you score more hits at places where the graph is higher, and exactly so in proportion to the height of a graph.

4.4 A Rejection Technique

Alice: That is a clever solution. It is called a rejection technique. Didn't

John von Neumann first apply that? You start by allowing more solutions than the minimal set of correct ones, and then you weed out the incorrect ones, by rejecting them.

Bob: Indeed. And to make the procedure efficient, you don't want to throw darts way above the graph, so you limit yourself to the maximum value that the graph attains in the interval of interest, or perhaps a slightly higher value. The authors of the paper choose a value of 0.1.

Alice: Is that a safe value? Let's check for ourselves. The derivative of $g(q)$ is

$$\begin{aligned} \frac{dg(q)}{dq} &= 2q\left(1-q^2\right)^{7/2} - 7q^3\left(1-q^2\right)^{5/2} \\ &= q\left(2-9q^2\right)\left(1-q^2\right)^{5/2} \end{aligned} \tag{4.10}$$

To find the extrema for $g(q)$, we set the derivative to zero, and solve for $0 < q_x < 1$:

$$q_x^2 = 2/9 \Rightarrow g(q_x) = (2/9)(7/9)^{7/2} \approx 0.092 \tag{4.11}$$

Indeed: 0.1 is a rather tight upper limit.

Bob: You can now see what I did when I wrote:

```
x = 0.0
y = 0.1
while y > x*x*(1.0-x*x)**3.5
  x = frand(0,1)
  y = frand(0,0.1)
end
velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
```

Alice: Ah, yes. So `x` stands for q and `y` stands for $g(q)$. You keep throwing darts until you find a `y` value under the graph. That gives you the corresponding `x` value. Since this value is equal to $q = v/v_e$, you have to multiply `x` with the escape velocity v_e, which we found in eq. (4.4) to be:

$$v_e(r) = \sqrt{2}\left(r^2+1\right)^{-1/4} \tag{4.12}$$

Okay, I understand the procedure now, and it looks correct.

4.5 Distribution Function

Bob: The only thing left to do now is to derive the form of the distribution function.

Alice: Yes. Let us see how far we can get. Clearly we need a little help from our friends: here is the classic reference *Galactic Dynamics*, by James Binney and Scott Tremaine [Princeton University Press, 1987]. I have found it to be a very useful book, whenever I had to look up some properties of particular models in stellar dynamics. It also has helped me a number of times to refresh my memory about Jeans equations, the tensor virial theorem and those sort of things.

Bob: I see that your browsing was rather uneven: there is a piece, about one third along the way, which has a gray strip. Let me open it up there. Aha, chapter 4: Equilibria of Collisionless Systems. How come those pages are so well-used? I thought you were mainly interested in collisional systems?

Alice: I didn't realize my copy of Binney and Tremaine betrayed my past browsing so well. You're right, that's the chapter I tend to consult most. And precisely because we are interested in *collisional* systems, we have to find a way to start our simulations with a *collisionless* system.

In other words, we run a numerical simulation of a collisional N-body system in order to see the effects of two-body relaxation and all that. But we need to have a well-defined starting point. A formal way to define this is to say that we ignore collisions from time $t = -\infty$ all the way to $t = 0$. Then, at $t = 0$, we switch on the effects of close encounters, and through our simulations we can see in what way our star clusters then begin to deviate from the dynamical equilibrium we started with.

I'm sure you don't like such a formal way of speaking about it. But you can't complain: you started asking philosophical questions about well-read chapters in someone else's book! So there.

Bob: Okay, let's go get our distribution function. Does this book tell us how to do that?

Alice: Yes, but let us first see how far we can get under our own steam. Let us go back all the way to how we got started, namely with the Plummer potential:

$$\Phi(r) \;=\; -\,GM\frac{1}{(r^2+a^2)^{1/2}}$$

Let us define $\tilde{f}(\mathbf{r},\mathbf{v})$ as the density of stars in phase space. This means that in a six-dimensional volume element $d\mathbf{r}d\mathbf{v}$, you will find a number of particles equal to $\tilde{f}(\mathbf{r},\mathbf{v})d\mathbf{r}d\mathbf{v}$. We will restrict ourselves to equal-mass systems, where each star has a mass m. This means that the volume element $d\mathbf{r}d\mathbf{v}$ contains on average an amount of mass equal to $m\tilde{f}(\mathbf{r},\mathbf{v})d\mathbf{r}d\mathbf{v}$.

In general, $\tilde{f}$ will be time dependent, *i.e.* of the form $\tilde{f}(\mathbf{r}, \mathbf{v})$, but here we will restrict ourselves to dynamical equilibrium situations, where $\tilde{f}$ is constant, at least on a dynamical time scale, on the order of the crossing time, the time it will take for a typical particle to cross the system.

Each star moving with velocity v at radial position r has a kinetic energy $\frac{1}{2}mv^2$ and a potential energy $m\Phi(r)$. It is most convenient to talk about specific energies, namely the energy per unit mass, E, which is given as:

$$E = \Phi(r) + \tfrac{1}{2}v^2 \tag{4.13}$$

In the case of spherical symmetry and isotropy, we have $\tilde{f}(\mathbf{r}, \mathbf{v}) = f(E)$ where the energy $E(\mathbf{r}, \mathbf{v})$ is the sum of the kinetic and potential energy per unit mass of the particles that reside in the volume element $d\mathbf{r}d\mathbf{v}$.

Now you see why I started with that funny tilde over the distribution function: since we will be dealing primarily with $f(E)$, I preferred to use the f notation for a distribution function with an E dependence, leaving the $\tilde{f}$ notation for the dependence of Cartesian phase space coordinates.

Bob: I still would be happy to not use tildes at all, but we'll each follow our own way.

Alice: There is one more thing we definitely have to stress here. Even though we have switched to E dependence, we still have to write $f(E)d\mathbf{r}d\mathbf{v}$, in order to find the number of stars in the volume element $d\mathbf{r}d\mathbf{v}$.

This is something students always get confused about. It is tempting to write $f(E)dE$ instead, but that would be wrong: the element dE would include all particles with specific energy between E and $E + dE$, globally in the system, which will amount to far more mass than is present in the local volume element $d\mathbf{r}d\mathbf{v}$ around the position $\mathbf{r}, \mathbf{v}$ in phase space.

Bob: Noted!

4.6 The Density as a Projection

Alice: The most fundamental stage for a star system in stellar dynamics is phase space. Because our physical eyes see what happens in configuration space, we think about the density as a rather basic function. But really, what we call density is already a type of shadow: it is a projection down to 3D from the distribution function in 6D.

In a star cluster with spherical symmetry in configuration space, we can show this projection effect as follows. The mass density in stars at a distance r from the center is

$$\rho(r) = \int_0^\infty f(E)\, d\mathbf{v} \tag{4.14}$$

where the integral is over all of velocity space. If in addition to spherical symmetry, the distribution function is isotropic, we know that there is no direction dependence in $\mathbf{v}$. In that case only the magnitude $v = |\mathbf{v}|$ can come into play. We can thus substitute:

$$d\mathbf{v} \;=\; 4\pi\, v^2 dv$$

Also, we know that the distribution function has to be zero for positive energies, since unbound particles will escape, and their density in an (almost) infinite universe will become (almost) zero:

$$f(E) \;=\; 0 \quad (E \geq 0)$$

For a given radial distance r, the escape velocity is given by definition as the velocity for which a particle can just reach infinity, which means that the total energy is zero for that particle, and therefore also the energy per unit mass. As you already showed right at the beginning of our discussion, this implies:

$$E = \Phi(r) + \tfrac{1}{2} v_{esc}^2 = 0 \;\Rightarrow\; v_{esc}(r) = \left(-2\Phi(r)\right)^{1/2}$$

This means that we can rewrite eq. (4.14) as:

$$\rho(r) = 4\pi \int_0^{v_{esc}} f(E)\, v^2 dv$$

Now

$$E \;=\; \Phi + \tfrac{1}{2} v^2 \;\;\Rightarrow\;\; v^2 = 2(E - \Phi) \;\;\Rightarrow$$

$$v \;=\; 2^{1/2}(E - \Phi)^{1/2} \;\;\Rightarrow$$

$$dv \;=\; 2^{-1/2}(E - \Phi)^{-1/2}\, dE$$

So we get for the density:

$$\rho(r) \quad = \quad 4\pi \int_0^{v_{esc}} f(E)\, v^2 dv$$

$$
\begin{aligned}
&= 4\pi \int_0^{v_{esc}} f(E)\, 2(E-\Phi)\, 2^{-1/2}(E-\Phi)^{-1/2}\, dE \\
&= 2^{5/2}\pi \int_\Phi^0 f(E)(E-\Phi)^{1/2} dE
\end{aligned} \tag{4.15}
$$

Here both Φ and E are negative. It is easier to introduce quantities that are positive. Following the notation used by Binney and Tremaine, we can define:

$$
\begin{cases}
\Psi &= -\Phi \\
\mathcal{E} &= -E
\end{cases} \tag{4.16}
$$

We can write again explicitly how the density in configuration space is a projection down from phase space, through an integration over $\mathcal{E}$:

$$
\rho(r) = 2^{5/2}\pi \int_0^\Psi f(\mathcal{E})(\Psi - \mathcal{E})^{1/2} d\mathcal{E} \tag{4.17}
$$

Strictly speaking I should have changed the symbol for f again since the functional form is different: $\hat{f}(\mathcal{E}) = f(E)$.

Bob: I'm glad you didn't!

Alice: I decided to take my hat off for you, or at least fs hat. Even for me, too much formality gets bothersome.

4.7 A Change of Variables

Bob: Looking over your shoulder, I see that your eq. (4.17) is exactly eq. (4-137) in Binney and Tremaine, if you pull out their $\sqrt{2}$ in front of the integral.

Alice: Yes. Let us follow their next few steps, for our particular Plummer case. They remark that Ψ is a monotonic function of r, and therefore it is possible to use Ψ as the independent variable in the expression for ρ.

Bob: That makes sense. If $\Psi(r)$ would not be monotonic, there would be some values r_1 and r_2 with $r_1 \neq r_2$ for which $\Psi(r_1) = \Psi(r_2)$. In that case, if you would write $\rho(\Psi)$, you would not know whether you would be referring to $\rho(r_1)$ or $\rho(r_2)$.

Alice: Right. But here everything is monotonic: the density keeps dropping off when you move away from the center, the radius keeps increasing, and the potential keeps increasing as well, or equivalently, $\Psi = -\Phi$ keeps decreasing.

Bob: So how do we get from $\rho(r)$ to $\rho(\Psi)$? That can't be difficult. We have

$$
\rho(r)dr = \rho(\Psi)d\Psi \;\Rightarrow\; \rho(\Psi) = \frac{dr}{d\Psi}\rho(r)
$$

And the conversion factor is nothing less than the inverse of the gravitational acceleration, which I derived early on, with the prophetic remark that it would come in handy later on.

Alice: No.

Bob: No?

Alice: No, there is no conversion factor. Think about what density means, in this context. The mass density $\rho(r)d\mathbf{r}$ is the amount of mass in stars within the volume $d\mathbf{r}$, and *not* within the whole shell of the cluster between r and $r + dr$. The latter amount of mass would be $4\pi r^2 \rho dr$.

This is the same thing as what we had seen with the phase space distribution function f: the physical meaning does not change, where we write it as $f(\mathbf{r}, \mathbf{v})$ or as $f(E)$. In both cases we multiply f with the volume element $d\mathbf{r}d\mathbf{v}$.

Bob: You're right! How tricky. Or how stupid of me.

Alice: Let's call it tricky. Believe me, I've made these mistakes often enough myself.

4.8 Deprojecting the Density

Bob: So eq. (4.17) becomes simply:

$$\rho(\Psi) \;=\; 2^{5/2}\pi \int_0^\Psi f(\mathcal{E})(\Psi - \mathcal{E})^{1/2} d\mathcal{E}$$

Alice: Indeed. If we divide both sides of the equation by $\sqrt{8}\pi$, we get Binney and Tremaine's eq. (4-138):

$$\frac{1}{\sqrt{8}\pi}\,\rho(\Psi) \;=\; 2\int_0^\Psi f(\mathcal{E})(\Psi - \mathcal{E})^{1/2} d\mathcal{E} \tag{4.18}$$

Bob: Why would they divide by that funny factor?

Alice: In order to get a simple looking right-hand side in their next equation (4-139), which they obtain by differentiating eq. (4.18) with respect to Ψ:

$$\frac{1}{\sqrt{8}\pi}\,\frac{d\rho}{d\Psi} \;=\; \int_0^\Psi f(\mathcal{E})\,\frac{d\mathcal{E}}{\sqrt{\Psi - \mathcal{E}}} \tag{4.19}$$

Bob: Ah, and here they do their deprojecting: they claim that this equation can be inverted. Eq. (4.19) gives you $d\rho/d\Psi$, when you give it $f(\mathcal{E})$. What we want is to obtain $f(\mathcal{E})$, and we can certainly figure out how to compute $d\rho/d\Psi$. I see the logic now.

Alice: Yes, and they give the solution of inverting Eq. (4.19), as their eq. (4-140a):

$$f(\mathcal{E}) = \frac{1}{\sqrt{8}\pi^2}\frac{d}{d\mathcal{E}}\int_0^{\mathcal{E}}\frac{d\rho}{d\Psi}\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} \tag{4.20}$$

and in alternative form as their (4-140b):

$$f(\mathcal{E}) = \frac{1}{\sqrt{8}\pi^2}\left[\int_0^{\mathcal{E}}\frac{d^2\rho}{d\Psi^2}\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} + \frac{1}{\sqrt{\mathcal{E}}}\left(\frac{d\rho}{d\Psi}\right)_{\Psi=0}\right] \tag{4.21}$$

Bob: Great! That will make it finally possible for us to determine the distribution function: with a little luck we can solve that integral.

Alice: Not so quick: how do we know that they inverted eq. (4.19) correctly?

Bob: There you go again! You don't take anything on authority, do you?

Alice: Well, no, preferably not. The whole reason to embark on this exercise is to prove things for ourselves, from scratch. I won't call it starting from scratch if we simply accept their inversion. We might as well have accepted their expression for the distribution function in the first place.

Bob: Which would have been fine with me, to be honest. But okay, we got started, we may as well finish. Though I really begin to doubt that our students will have the stomach to go through all of this.

Alice: The best ones will. And those are the ones we'd like to stay in touch with. So this may a good way to find good students, to see who is interested in figuring this out, all the way to the bottom.

Bob: I'm not so sure. You might just get formal and pedantic students. I prefer to work with students who get codes to run and who get results with codes, rather than students who can solve this inversion equation – what do they call it? – an Abel integral equation.

Alice: The *very* best students will be able to do both.

Bob: As able as Abel.

Alice: Such a silly pun would work better if you had an accent. Come, let's take our usual approach: first let's assume that eqs. (4.20) and (4.21) are indeed correct, and let us check whether we can then derive the correct distribution function for Plummer's model. Having done that, we will come back, and check for ourselves whether we can prove this Abel integral equation inversion.

Bob: Something tells me it will be a long day.

Alice: Perhaps. Whatever it takes! But I agree with you, this may well take longer than we thought. Let's continue tomorrow.

Bob: That sounds better already. See you then!

Chapter 5

Getting the Physics Right

5.1 An Analytic Recipe

Alice: Hi Bob! Well, are you ready to derive the distribution function of Plummer's model from first principles?

Bob: If we're really going to get to the bottom of this, I *insist* on using $G = M = a = 1$. Let's roll up our sleeves then. Here is the, by now very familiar, potential-density pair for our Plummer's model:

$$\begin{cases} \Psi(r) &= -\Phi(r) = \left(1+r^2\right)^{-1/2} \\ \rho(r) &= \dfrac{3}{4\pi}\left(1+r^2\right)^{-5/2} \end{cases} \tag{5.1}$$

Substituting the right-hand side of the first equation in the second equation, we find:

$$\rho(\Psi) = \frac{3}{4\pi}\,\Psi^5 \tag{5.2}$$

Now we have a choice: we can use this expression either in eq. (4.20) or in eq. (4.21). If I were to guess, the sheer fact that Binney and Tremaine have added the second equation would suggest that that one is the easiest to work with; otherwise they would have limited themselves to the first one. They don't seem to add anything more than necessary.

Alice: Unlike our style of writing.

Bob: I'd say! It's a good thing that we are writing this for our students . . .

Alice: . . . and for ourselves . . .

Bob: . . . and for ourselves, yes, but not for our colleagues. I certainly wouldn't want to show them how many small steps it takes me to derive these types of equations.

Alice: But don't forget, we are planning to put this up on the web.

Bob: Ah, yes, I'd forgotten about that already. Oh, well, we can just call each page 'class notes', and that will chase our colleagues away.

Alice: I'm not so sure, but I don't want to discourage you. Let's move on!

Bob: Yes, let's. As I was saying, my bet is on eq. (4.21). Let's write it down again here, and then substitute $\rho(\Psi)$ with eq. (5.2):

$$\begin{aligned} f(\mathcal{E}) &= \frac{1}{\sqrt{8}\pi^2}\left[\int_0^{\mathcal{E}} \frac{d^2\rho}{d\Psi^2}\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} + \frac{1}{\sqrt{\mathcal{E}}}\left(\frac{d\rho}{d\Psi}\right)_{\Psi=0}\right] \\ &= \frac{1}{\sqrt{8}\pi^2}\frac{3}{4\pi}\left[\int_0^{\mathcal{E}} \left(\frac{d^2}{d\Psi^2}\left(\Psi^5\right)\right)\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} + \frac{1}{\sqrt{\mathcal{E}}}\left(\frac{d}{d\Psi}\left(\Psi^5\right)\right)_{\Psi=0}\right] \\ &= \frac{3}{8\sqrt{2}\pi^3}\left[\int_0^{\mathcal{E}} 20\frac{\Psi^3 d\Psi}{\sqrt{\mathcal{E}-\Psi}} + \frac{5}{\sqrt{\mathcal{E}}}\left(\Psi^4\right)_{\Psi=0}\right] \end{aligned}$$

The second term is zero, and the expression in the denominator invites us to introduce the following change of variables:

$$x = \mathcal{E} - \Psi \quad \Rightarrow \quad \Psi = \mathcal{E} - x \quad \Rightarrow \quad d\Psi = -dx \quad \Rightarrow$$

$$\begin{aligned} f(\mathcal{E}) &= \frac{3\times 20}{8\sqrt{2}\pi^3}\int_{\mathcal{E}}^{0} (\mathcal{E}-x)^3\frac{(-dx)}{\sqrt{x}} \\ &= \frac{15\sqrt{2}}{4\pi^3}\int_0^{\mathcal{E}}\left\{-x^{5/2}+3\mathcal{E}x^{3/2}-3\mathcal{E}^2x^{1/2}+\mathcal{E}^3x^{-1/2}\right\}dx \\ &= \frac{15\sqrt{2}}{4\pi^3}\left\{-\frac{2}{7}x^{7/2}+\frac{6}{5}\mathcal{E}x^{5/2}-2\mathcal{E}^2x^{3/2}+2\mathcal{E}^3x^{1/2}\right\}\Bigg|_0^{\mathcal{E}} \\ &= \frac{15\sqrt{2}}{4\pi^3}\left\{-\frac{2}{7}+\frac{6}{5}-2+2\right\}\mathcal{E}^{7/2} \\ &= \frac{15\sqrt{2}}{4\pi^3}\frac{32}{35}\mathcal{E}^{7/2} \\ &= \frac{24\sqrt{2}}{7\pi^3}\mathcal{E}^{7/2} \end{aligned} \tag{5.3}$$

5.2 The Full Form

Well, I guess I was not as rusty in these kind of manipulations as I had thought I was.

Alice: You shouldn't, since you did your undergraduate studies a lot more recently than I did!

Bob: Yet it seems so long ago! I guess everything is relative.

Alice: At least the power 7/2 came out correctly, as expected for a polytrope of index 5. A while ago you wrote down an expression you got from some book, where was that, ah, eq. (4.5). Let's write it here again:

$$f(E(r,v)\,4\pi r^2 dr\,4\pi v^2 dv \;=\; \frac{384\sqrt{2}}{7\pi m}(-E)^{7/2} r^2 v^2 dr\,dv \tag{5.4}$$

Bob: Since $384/16 = 24$, we got exactly what was ordered. It was a bit of work, but I must admit, it is fun to derive things from scratch. But wait, what is that factor m doing there? That one is missing from what I just derived.

Alice: It was the mass of a single star, in our star cluster in which all masses are the same. The book you picked your expression for the distribution function from, must have defined $f(E)$ as the *number* density of the stars. If you multiply both sides with m, you get on the right-hand side what you found. This then must be the *mass* density of the stars in phase space, namely the number density multiplied by the mass of a single star.

Bob: That must be the answer.

Alice: You may remember that I did not protest when you decided to do your derivation purely in dimensionless units. As a reward, I insist on you writing the final answer with all the G 's, M 's, and a 's put back in.

Bob: That's easy. Here it is:

$$f(\mathcal{E}) \;=\; \frac{24\sqrt{2}}{7\pi^3}\,\frac{a^2}{G^5 M^4}\,\mathcal{E}^{7/2} \tag{5.5}$$

But don't ask me to prove it to you with hats or tildes or a substantial part of the Greek alphabet!

Alice: I won't. But since that is how I would do it, I'm curious to know what you just did. How did you figure this out?

5.3 Dimensional Analysis

Bob: Well, I used dimensional analysis, but not in a very systematic way, just substituting what seems to work in the quickest way. If we denote the unknown

combination of G, M, a factors by A, we can start by writing

$$f(\mathcal{E}) \;=\; A\,\mathcal{E}^{7/2}$$

where I have dropped all numerical factors, since they won't make any difference as far as dimensional analysis is concerned. Since you just pointed out that we have computed the *mass* density in phase space, we know that the expression

$$f(\mathcal{E})d\mathbf{r}d\mathbf{v}$$

has the dimension of mass, something we can write as

$$[f(\mathcal{E})d\mathbf{r}d\mathbf{v}] \;=\; [\text{mass}]$$

Therefore

$$[f] \;=\; [\text{mass}]\;[\text{length}]^{-3}\;[\text{velocity}]^{-3}$$

Since energy is proportional to mass times velocity squared, think mc^2, our $\mathcal{E}$, the energy per unit mass, has a dimension of velocity squared, so

$$\left[\mathcal{E}^{7/2}\right] \;=\; [\text{velocity}]^7$$

Before you protest that I should use only physical dimensions of [mass], [length] and [time] . . .

Alice: . . . which is what I would have done . . .

Bob: . . . I want to point out that I have a freedom of choice, as long as my three units are not degenerate. I find it more convenient to work with [mass], [length] and [velocity].

Alice: Come to think of it, that makes sense, since the last two reflect the two different spaces of which phase space is the product, and mass is what we have been sprinkling into phase space.

Bob: I guess. You always find a way to suggest more abstract reasons for what I am doing intuitively. It just has been my experience that velocity is a more useful quantity than time in this kind of dimensional analysis.

Alice: I think I'll follow your suggestion, from now on. Learn something new every day!

5.4 Getting The Details Right

Bob: Where was I? Ah, yes, I started with

$$f(\mathcal{E}) \,=\, A\,\mathcal{E}^{7/2}$$

and using the last two expression above, I can find that A has the following dimension:

$$\begin{aligned}
[A] &= [f]\left[\mathcal{E}^{-7/2}\right] \\
&= [\text{mass}]\,[\text{length}]^{-3}\,[\text{velocity}]^{-3}\,[\text{velocity}]^{-7} \\
&= [\text{mass}]\,[\text{length}]^{-3}\,[\text{velocity}]^{-10}
\end{aligned}$$

Now the crucial point is that only G can help us in giving us a velocity dimension. Velocity involves time, since its dimension is length over time, and neither M nor a have a time component.

The dimension of G follows from the fact that potential energy must have the same dimension as kinetic energy:

$$\left[\frac{GMm}{r}\right] \,=\, \left[\tfrac{1}{2}mv^2\right]$$

for whatever M's, m's, r's, *etc.* you care to use. This implies:

$$\left[v^2\right] \,=\, \left[\frac{GM}{r}\right] \,=\, [G]\,[\text{mass}]\,[\text{length}]^{-1}$$

So this tells us that we can factor out G as follows:

$$\begin{aligned}
[A] &= [\text{mass}]\,[\text{length}]^{-3}\left[G^{-5}\right]\,[\text{mass}]^{-5}\,[\text{length}]^{5} \\
&= \left[G^{-5}\right]\,[\text{mass}]^{-4}\,[\text{length}]^{2}
\end{aligned}$$

Since the only mass we have is M, and the only length we have is a, we have to conclude that

$$A \,\propto\, G^{-5}M^{-4}a^2$$

This then means that our dimensionless expression, that we derived with the help of Binney and Tremaine:

$$f(\mathcal{E}) \,=\, \frac{24\sqrt{2}}{7\pi^3}\,\mathcal{E}^{7/2}$$

has to be expanded in the following way, to make it again dimensionally correct:

$$f(\mathcal{E}) = \frac{24\sqrt{2}}{7\pi^3}\,\frac{a^2}{G^5M^4}\,\mathcal{E}^{7/2}$$

5.5 From Implicit to Explicit

Alice: This is indeed what you wrote down before, eq. (5.5).

Bob: I must say, I'm surprised at how many words and lines of equations I have to write down to make it all explicit. I did it mostly in my head, with a few scribbles here and there to provide some help.

Alice: That's because you've grown so familiar by now with these kinds of manipulations. But I definitely think it is a good idea to show your students how you do this, if only once.

Bob: Don't you think they will figure it out for themselves, sooner or later.

Alice: Probably later. I like the idea of being a catalyst, speeding up the process of letting them discover things. And by making your implicit stream of thoughts explicit, tedious as it may seem, we may be doing them a real favor.

Besides, to be really honest, I think we can still learn a lot as well. I had no idea that your approach would be so different from mine, and I can see the advantage of your way of thinking, at least in some cases such as these. It would definitely have taken me longer to derive your result in my way.

Bob: It is interesting, isn't it, that normally we don't talk very much about how we do these kind of little, or not so little, derivations. And I must admit, I too have learned quite a bit from the way you approach these problems. Besides, it is more fun to struggle with all this together.

Well, I think we've struggled enough now. Let's call it a day.

Alice: call it a day? But we haven't yet checked the truth behind the Abel integral transform, whether that way of inverting the equation was correct. We have just taken Binney and Tremaine's word for it!

Bob: I'd forgotten all about that. Well, hmm. Do you really insist on checking it all the way?

Alice: I do.

Bob: Okay, okay. Let's get it over with. Can I look at that page, where they talk about Mr. Abel and his transforms? You would at least expect them to give us a hint.

Alice: They do give a hint: go read appendix 1.B.4.

5.6 Abel Integral Transforms

Bob: So they do. Here it is. They start with the following two lines.

Let

$$f(x) = \int_0^x \frac{g(t)dt}{(x-t)^\alpha} \qquad\qquad 0 < \alpha < 1 \tag{5.6}$$

Then

$$\begin{aligned} g(t) &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\left[\int_0^t \frac{df}{dt}\frac{dx}{(t-x)^{1-\alpha}} + \frac{f(0)}{t^{1-\alpha}}\right] \end{aligned} \tag{5.7}$$

I like that terse style: only "Let . . then . .", what a difference from our constant chattering!

Alice: It is a good move of them to reduce the more complex equations (4.20) and (4.21) to these simplified forms. But let us first check whether the "Let . . then . ." claim, if true, will solve our problem. If it really does, we then have to verify the claim itself, to make sure that the problem really has been solved.

Let us start with eq. (5.6). For this to be equal to our previous equation (4.19):

$$\frac{1}{\sqrt{8}\pi}\frac{d\rho}{d\Psi} = \int_0^\Psi f(\mathcal{E})\frac{d\mathcal{E}}{\sqrt{\Psi-\mathcal{E}}}$$

We have to use the following dictionary, with their appendix notation on the left and their main text notation on the right:

$$\begin{cases} f(x) & \to & \dfrac{1}{\sqrt{8}\pi}\dfrac{d\rho}{d\Psi} \\ x & \to & \Psi \\ g(t) & \to & f(\mathcal{E}) \\ t & \to & \mathcal{E} \\ \alpha & \to & \frac{1}{2} \end{cases} \tag{5.8}$$

Bob: Now that's what I call confusing. They use f symbols for two very different functions, in their two languages. We'll have to be careful with this dictionary.

Alice: But I thought you didn't like hats and tildes, and preferred to use the same symbol for different things?

Bob: Only if they have the same physical meaning. Here we're talking mathematics. I would have prefered that they would have used $k(x)$ instead of $f(x)$ in the appendix. Oh well, small point, as long as we're careful.

Alice: Now let us assume that their eq. (5.9) indeed provides the solution for eq. (5.6). We then have to translate eq. (5.6) back into main text quantities, using the dictionary in the other direction. Here we go:

$$\begin{aligned} g(t) &= \frac{\sin\pi\alpha}{\pi}\,\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\left[\int_0^t \frac{df}{dt}\,\frac{dx}{(t-x)^{1-\alpha}} + \frac{f(0)}{t^{1-\alpha}}\right] \quad \Rightarrow \end{aligned}$$

$$\begin{aligned} f(\mathcal{E}) &= \frac{\sin\left(\frac{\pi}{2}\right)}{\pi}\,\frac{d}{d\mathcal{E}}\int_0^{\mathcal{E}} \frac{1}{\sqrt{8}\pi^2}\,\frac{d\rho}{d\Psi}\,\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} \\ &= \frac{\sin\left(\frac{\pi}{2}\right)}{\pi}\left[\int_0^{\mathcal{E}} \frac{1}{\sqrt{8}\pi^2}\,\frac{d^2\rho}{d\Psi^2}\,\frac{d\Psi}{\sqrt{\mathcal{E}-\Psi}} + \frac{1}{\sqrt{8}\pi^2}\,\frac{1}{\sqrt{\mathcal{E}}}\left(\frac{d\rho}{d\Psi}\right)_{\Psi=0}\right] \end{aligned}$$

Bob: Indeed, these are exactly the eqs. (4.20) and (4.21) that we set out to prove. Great! Now can we start testing my code?

Alice: Ho! Not so quick. All we have done is verify that Binney and Tremaine's claim in the appendix leads to their claim in the main text. Now we have to prove the claim they make in their appendix.

Chapter 6

Getting the Math Right

6.1 A Mathematical Proof

Bob: I started out with the recipe provided by Aarseth, Henon and Wielen, and you insisted that we use Binney and Tremaine to check them. We now have five luminaries in stellar dynamics to vouch for our results to be correct. On top of that, we got quite a bit of insight in the underlying physics. Isn't that enough?

Alice: Not to my taste. We have blindly accepted what Binney and Tremaine claim in their appendix, without proving it for ourselves.

Bob: But that is mathematics! Look, there is no physics in the relation between eqs. (5.6) and (5.9). It's just a matter of some kind of mathematical transformation, like Fourier transforms or Laplace transforms.

Alice: Even so, we set out to prove things from first principles, and I certainly would feel more comfortable to do exactly that. And I don't feel I can call Abel transforms first principles. Besides, they may well come in handy when we will start constructing more complicated models, and I wouldn't mind getting some more practice in these type of manipulations.

Come on, let's just do it.

Bob: Okay, but not a bit more than is really necessary. Look, they introduce α where we only needed the number 1/2.

Alice: I'll do it for α, why not. You can just watch. So how is the problem posed exactly? In the "if . . then . . " sentence in their appendix . . .

Bob: . . . their "Let . . then . . " sentence . . .

Alice: . . . yes, of course, they are mathematical physicists, not computer scientists. Okay, in the "Let . . then . . " sentence in their appendix, Binney

and Tremaine actually assert two results. Let me start trying to prove the first one, which is:

Let

$$f(x) = \int_0^x \frac{g(t)dt}{(x-t)^\alpha} \qquad 0 < \alpha < 1 \tag{6.1}$$

Then

$$g(t) \;=\; \frac{\sin \pi\alpha}{\pi} \frac{d}{dt} \int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \tag{6.2}$$

Now how shall we prove this?

6.2 The Problem

Bob: They give a terse hint about substituting the first equation in the second and interchanging the order of the integration. Now why would you want to substitute the first equation in the second? I like brevity, but this is a bit too brief for me.

Alice: Well, given that their book as it is runs already well over 700 pages, they probably thought they couldn't afford more room for explanations.

Bob: Good thing we don't suffer from that constraint. The world wide web is wide enough for our meanderings.

Alice: But let's not meander too far. I want to crack this nut. Let's see. Ah, they must mean that they take the right hand side of eq. (6.2), and work that out, in order to see whether they really get $g(t)$ back in the end.

Bob: What do you mean? They have already called it $g(t)$.

Alice: Well, they have *asserted* that the right hand side is the answer to the question of what $g(t)$ is. We now have to prove it.

Bob: I find that confusing.

Alice: It is a bit confusing. Here, let's be more precise and explicit, without confusing name spaces. Forget about eq. (6.2) . . .

Bob: . . . with pleasure and glee. I'd just as soon forget about this whole derivation . . .

Alice: Patience, please. Forget about eq. (6.2), and write instead

$$h(t) \;=\; \frac{\sin \pi\alpha}{\pi} \frac{d}{dt} \int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \tag{6.3}$$

Bob: Exactly the same, but now you're calling it $h(t)$.

Alice: Exactly. Exactly exactly, I mean. The point is that now we don't make any claim about what $h(t)$ is supposed to mean, it is just a function we happen to define this way, right? Now, this new function $h(t)$ also happens to have this factor $f(x)$ in the integrand, and nothing stops us from substituting the expression for $f(x)$, given in eq. (6.1) into eq. (6.3).

Bob: Yes, and yes. I agree. That is much clearer, since there are only one $f(x)$ and one $g(t)$ and one $h(t)$ in the game, rather than two different $g(t)$'s with different status. And ah, now I see why you would want to substitute eq. (6.1) into eq. (6.3). Is that what they meant?

Alice: I guess it is.

Bob: Okay, now I see why. Boy, they *are* terse!

Alice: In math, more than half the work is often to find out exactly *what* to do. Actually doing it is typically the least of the problem.

Bob: But let's do it, and get it over with.

6.3 The Answer

Alice: Yes, let's. Here is the result of substituting eq. (6.1) into eq. (6.3):

$$\begin{aligned} h(t) &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t \frac{dx}{(t-x)^{1-\alpha}}\int_0^x \frac{g(t')dt'}{(x-t')^{\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t dx\int_0^x dt'\frac{g(t')}{(t-x)^{1-\alpha}(x-t')^{\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t dt'\int_{t'}^t dx\frac{g(t')}{(t-x)^{1-\alpha}(x-t')^{\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t dt' g(t')\int_{t'}^t \frac{dx}{(t-x)^{1-\alpha}(x-t')^{\alpha}} \end{aligned} \tag{6.4}$$

Bob: What did you do exactly when you went from the second to the third line?

Alice: I followed Binney and Tremaine's advice, to interchange the order of integration.

Bob: But how did you get the new limits for the two integral signs in the third line so quickly?

Alice: If you draw a picture, you can see what is going on here. Instead of t' use y instead. We then have a double integral over the $\{x, y\}$ plane. The area over which we take the integral is a triangle bounded on top by the diagonal

$x = y$, on the bottom by the positive x axis, and on the right by the line $x = t$. Now in the second line the inner integral runs over vertical lines, and in the third integral, integration runs over horizontal lines.

[We should probably draw a picture here]

— **picture** —

— **picture** —

— **picture** —

— **picture** —

— **picture** —

Bob: Ah, yes, I see now. It has been a while since I interchanged the order of integrations, I must admit. But your picture makes it clear. Onward!

Alice: Let us focus on the inner integral, and let us call it

$$I(t,t') \;=\; \int_{t'}^{t} \frac{dx}{(t-x)^{1-\alpha}(x-t')^{\alpha}} \tag{6.5}$$

It is natural to introduce:

$$y = x - t' \quad \Rightarrow \quad x = y + t' \quad \Rightarrow \tag{6.6}$$

$$I(t,t') \;=\; \int_{0}^{t-t'} \frac{dy}{(t-t'-y)^{1-\alpha}y^{\alpha}} \tag{6.7}$$

This in turn invites a second change of variables:

$$z \;=\; \frac{y}{t-t'} \quad \Rightarrow \quad y = (t-t')z \quad \Rightarrow \tag{6.8}$$

$$t - t' - y \;=\; (t-t')(1-z) \quad \Rightarrow \tag{6.9}$$

$$\begin{aligned} I(t,t') \quad &= \quad \int_0^1 \frac{(t-t')dz}{(t-t')^{1-\alpha}(1-z)^{1-\alpha}(t-t')^{\alpha}z^{\alpha}} \\ &= \quad \int_0^1 \frac{dz}{(1-z)^{1-\alpha}z^{\alpha}} \end{aligned} \tag{6.10}$$

That is nice: both the t and the t' dependences have dropped out, and we are left with a definite integral that only has one parameter, α.

Bob: And the answer is:

$$I(t,t') \;=\; \int_0^1 \frac{dz}{(1-z)^{1-\alpha} z^{\alpha}} \;=\; \frac{\pi}{\sin \pi\alpha} \tag{6.11}$$

6.4 All the Way

Alice: How did you get that so quickly? Did you use a symbolic integrator? That is cheating?

Bob: No, I used Binney and Tremaine again, their appendix eq. (1B-58). You see, they give you just the minimal amount of information needed to get this all worked out.

Alice: But we haven't worked out the integral yet.

Bob: Are you kidding? Perhaps I should call up a symbolic integrator!

Alice: Look, Bob, we're nearly there, a few feet from the finish line, and you want to give up now? Let's just work it out, and then we can tell all of our friends and family that we have just derived the distribution function corresponding to a softened potential, really from first principles.

Bob: My mom will be thrilled to hear that. I can't get you out of this room until you've proved everything, right? Well, I'll play along under one condition: this time we really will use only $\alpha = \frac{1}{2}$. That is all that we needed to prove Binney and Tremaine's main text results, eqs. (4.20) and (4.21). Anything more would be masochism.

Alice: Okay, okay, $\alpha = \frac{1}{2}$ it will be. I'll take a deep breath, and then:

$$\int_0^1 \frac{dz}{(1-z)^{1/2} z^{1/2}} \;=\; \int_0^1 \frac{dz}{(z-z^2)^{1/2}} \;=\; \int_0^1 \frac{dz}{\sqrt{\frac{1}{4} - \left(z-\frac{1}{2}\right)^2}} \;=$$

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{dx}{\sqrt{\frac{1}{4}-x^2}} \;=\; \int_{-1}^{1} \frac{\frac{1}{2}dy}{\sqrt{\frac{1}{4} - \left(\frac{y^2}{4}\right)}} \;=\; \int_{-1}^{1} \frac{dy}{\sqrt{1-y^2}}$$

Aha! Now we're getting there. And rather than looking *this* one up, I do remember how it went. It is all coming back to me now from my freshman years. You start with the following tautology, given that the arcsin function is the inverse of the sin function:

$$\sin(\arcsin x) = x$$

When you differentiate this with respect to x, you get

$$\frac{d}{dx}\,\sin(\arcsin x) \;=\; \cos(\arcsin x)\,\frac{d}{dx}\,\arcsin x \;=\; 1$$

This implies:

$$\frac{d}{dx}\,\arcsin x \;=\; \frac{1}{\sqrt{1-[\sin(\arcsin x)]^2}} \;=\; \frac{1}{\sqrt{1-x^2}}$$

This means that the solution of our integral is:

$$\int_0^1 \frac{dz}{(1-z)^{1/2}z^{1/2}} \;=\; \int_{-1}^1 \frac{dy}{\sqrt{1-y^2}} \;=\; \{\arcsin x\}\Big|_{-1}^{1} \;=\; \frac{\pi}{2}+\frac{\pi}{2} \;=\; \pi$$

Now that makes me feel good! From first principles, all the way.

6.5 Q.E.D.

Bob: Congratulations with going back to your youth!

Alice: Do I detect a slight sense of sarcasm there?

Bob: Only slight. Next thing you'll do is prove that $1+1=2$. Do you have a more first principles all-the-way way of proving that too?

Alice: Well, you start with the empty set, and the notion of a successor mapping, which can be implemented by constructing a set containing the previous number, and then . . .

Bob: . . . I shouldn't have asked!

Alice: Not if you want us to finish today. We're not quite done yet.

Bob: Anyway, I'm glad to see that you're getting at least a wee bit more terse in your derivations, not writing out every change in variables explicitly anymore.

Alice: Yeah, only a wee bit. So. Now we have to substitute our nice result, π, for the inner integral in eq. (6.4). Remembering the original definition of $h(t)$ in eq. (6.3), we can use both of these equations to write, for our $\alpha = \frac{1}{2}$:

$$\begin{aligned} h(t) &= \frac{1}{\pi}\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1/2}} \\ &= \frac{1}{\pi}\frac{d}{dt}\int_0^t dt' g(t') \int_{t'}^t \frac{dx}{(t-x)^{1/2}(x-t')^{1/2}} \\ &= \frac{1}{\pi}\frac{d}{dt}\int_0^t dt' g(t')\pi \end{aligned}$$

$$\begin{aligned} &= \frac{d}{dt}\int_0^t dt' g(t') \\ &= g(t) \end{aligned}$$

Quod erat demonstrandum.

Bob: quad what?

Alice: *quod*, as in *quod licet Jovi.* Never mind, that is Latin. It means 'what was to be demonstrated.' Mathematicians used to write *q.e.d* at the end of a proof.

Bob: I thought that stood for quantum electrodynamics.

Alice: That too, but we'll keep quantum field theory for volume 137 in our series.

Bob: Remind me, I'm losing track. So you have proved that $h(t) = g(t)$. Why again did we want to know that?

Alice: You're like my students: lack of motivation leads to loss of memory! We wrote down a definition for the function $h(t)$ in (6.3), and then we substituted the $f(x)$ expression within $h(t)$ using eq. (5.6). And then — but I can see your eyes glazing over. You must be getting tired.

Bob: Too much math, I'm afraid.

Alice: I'll write it down again. We started with

$$f(x) = \int_0^x \frac{g(t)dt}{(x-t)^\alpha} \qquad\qquad 0 < \alpha < 1 \tag{6.12}$$

and then we proved that

$$h(t) \;=\; \frac{\sin\pi\alpha}{\pi}\,\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \;=\; g(t) \tag{6.13}$$

In words: eq. (6.12) shows you have to compute $f(x)$, if you start with $g(t)$. Now in order to find the inverse expression, you start instead with $f(x)$, and then you can compute $h(t)$ given in (6.13), and lo and behold, that actually gives you $g(t)$. So the expression between the two equal signs in eq. (6.13) is the inverse of the expression given in eq. (6.12): our friend the Abel transform. And the good thing is: we have now proved it completely.

Bob: How nice. And yes, I think you are right: I am getting a bit tired. Can we go home now?

6.6 The End of Let-Then

Alice: Almost. Remember Binney and Tremaine's appendix? Here it is again,

there famous "Let . . then . ." claim:

Let

$$f(x) = \int_0^x \frac{g(t)dt}{(x-t)^\alpha} \qquad 0 < \alpha < 1 \tag{6.14}$$

Then

$$\begin{aligned} g(t) &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\left[\int_0^t \frac{df}{dt}\frac{dx}{(t-x)^{1-\alpha}} + \frac{f(0)}{t^{1-\alpha}}\right] \end{aligned} \tag{6.15}$$

See, we have proved the first equality in the last expression, but we haven't proved the second equality yet. I promise, this will be the last thing we'll do today. I hope it is just a matter of writing things out, since I'm getting a bit tired to, to tell you the truth. One more deep breath, and here we go.

A natural change of variables to simplify the denominator in the integrand for $g(t)$ is:

$$w = t - x \quad \Rightarrow \quad x = t - w \quad \Rightarrow \quad dx = -dw$$

Starting now with the first equality in the last equation above, I'll just see what happens when I do the differentiation with respect to t:

$$\begin{aligned} g(t) &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t \frac{f(x)dx}{(t-x)^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_t^0 \frac{f(t-w)(-dw)}{w^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\frac{d}{dt}\int_0^t \frac{f(t-w)dw}{w^{1-\alpha}} \\ &= \frac{\sin\pi\alpha}{\pi}\left[\frac{f(t-w)}{w^{1-\alpha}}\bigg|_{w=t} + \int_0^t \left(\frac{d}{dt}f(t-w)\right)\frac{dw}{w^{1-\alpha}}\right] \\ &= \frac{\sin\pi\alpha}{\pi}\left[\frac{f(0)}{t^{1-\alpha}} + \int_0^t \left(\frac{d}{dx}f(x)\right)\frac{dx}{(t-x)^{1-\alpha}}\right] \\ &= \frac{\sin\pi\alpha}{\pi}\left[\frac{f(0)}{t^{1-\alpha}} + \int_0^t \frac{df}{dx}\frac{dx}{(t-x)^{1-\alpha}}\right] \end{aligned}$$

Bob: And that is exactly what you wanted to prove. Are you happy now?

Alice: I'm very happy.

Bob: And I even forgot to protest against the fact that you smuggled α back into the game again.

Alice: So, we've pulled it off!

Bob: And look how much we have pulled. We have taken a couple sentences from an appendix from Binney and Tremaine, and we have expanded their discussion by a factor of, what? More than ten pages I bet – an increase of two orders of magnitude! Amazing. No wonder I'm tired. Let's call it a day.

Alice: Fine with me. And tomorrow we'll see your code in action.

Bob: Looking forward to it!

Chapter 7

Energy Checks

7.1 More Modular

Alice: Hi Bob! Did you recover from our mathematical adventures?

Bob: Well, I must say, I'm glad to be back in the normal world. Let me show you how to deal with a *real* example of Plummer's model, rather than with a mathematical abstraction.

Alice: You mean the code you showed me, which triggered our lengthy discussion?

Bob: Yes, but I made one modification, following your suggestion that I could factor out the spherical angle treatment, which was so similar for creating the position and velocity vectors. I've named it `mkplummer2.rb`. There is now one new method:

```
def spherical(r)
  vector = Vector.new
  theta = acos(frand(-1, 1))
  phi = frand(0, 2*PI)
  vector[0] = r * sin( theta ) * cos( phi )
  vector[1] = r * sin( theta ) * sin( phi )
  vector[2] = r * cos( theta )
  vector
end
```

It takes only one argument, the absolute value of the vector that will be returned by this method with random values for the two spherical angles θ and ϕ. You can see how I invoke this method from within the inner loop of the `mkplummer` method:

```
nb.body.each do |b|
  b.mass = 1.0/n
  radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
  b.pos = spherical(radius)
  x = 0.0
  y = 0.1
  while y > x*x*(1.0-x*x)**3.5
    x = frand(0,1)
    y = frand(0,0.1)
  end
  velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
  b.vel = spherical(velocity)
end
```

Alice: I like this better, yes, it is more modular.

Bob: And a lot shorter, so you can see more clearly what is happening now.

7.2 Repeatability

Alice: just to see the raw output, can you show me what a 3-body realization looks like, of Plummer's model?

Bob: Sure. Here is one:

```
|gravity> kali mkplummer2.rb -n 3
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
            actual seed used: 898477328012380325888046941761925151 36
ACS
  NBody
    Array body
      Body body[0]
        Float mass
             3.3333333333333331e-01
        Vector pos
             1.5011979414230079e+00  -2.3790824944281405e-01  -8.5527439631222768e
        Vector vel
             4.0773664926673316e-02  -1.0952878979900857e-01   3.1418781269054652e
```

```
      Body body[1]
        Float mass
             3.3333333333333331e-01
        Vector pos
            -3.4745331723355716e-01   1.0459146044533737e-02  -9.1898956388538386e-02
        Vector vel
            -2.5078179240334225e-01   4.9729801541029434e-01   1.9706180990947048e-01
      Body body[2]
        Float mass
             3.3333333333333331e-01
        Vector pos
             6.5257165308596238e-02  -1.6698017505293156e-01  -1.0951348661040470e+00
        Vector vel
            -2.2702458895205430e-01   2.3119688110276201e-01  -7.8480845814976263e-01
SCA
```

Alice: I see that you are echoing the seed from random number generator, as you explained before. Let us check whether we indeed can repeat the same model generation. I'd be happy to just compare the last particle. What value should we take for the seed of the random number generator?

Bob: 42.

Alice: Of course, that is the answer. But what a mundane question it was!

Bob: Different questions can have the same answer. Here is the first attempt:

```
|gravity> kali mkplummer2.rb -n 3 -s 42 | tail -2
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 42
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 42
             1.0707835396652617e-01   3.2726929775243468e-01   2.3909866955874151e-01
SCA
```

And now let me repeat the command with the same seed:

```
|gravity> kali mkplummer2.rb -n 3 -s 42 | tail -2
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 42
```

```
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 42
             1.0707835396652617e-01   3.2726929775243468e-01   2.3909866955874151e
SCA
```

And now with a different seed:

```
|gravity> kali mkplummer2.rb -n 3 -s 43 | tail -2
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 43
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 43
             1.0397916156880645e-01  -1.4513970203331489e-01   1.6854199026403588e
SCA
```

Alice: Good! And the full values from the first 3-body version look plausible. But of course we cannot check, just by staring at them, whether they really correspond to Plummer's model. We'll have to come up with some checks.

7.3 Energy Diagnostics

Bob: I thought about this checking question, so I cobbled together a tool to do a quick energy check. Here it is:

```
#!/usr/local/bin/ruby -w

require "acs"

class Body

  attr_accessor :mass, :pos, :vel

  def initialize(mass = 0, pos = Vector[0,0,0], vel = Vector[0,0,0])
    @mass, @pos, @vel = mass, pos, vel
  end
```

```
  def ekin                          # kinetic energy
    0.5*@mass*(@vel*@vel)
  end

  def epot(body_array)                    # potential energy
    p = 0
    body_array.each do |b|
      unless b == self
        r = b.pos - @pos
        p += -@mass*b.mass/sqrt(r*r)
      end
    end
    p
  end

end

class NBody

  attr_accessor :time, :body

  def initialize
    @body = []
  end

  def ekin                          # kinetic energy
    e = 0
    @body.each{|b| e += b.ekin}
    e
  end

  def epot                          # potential energy
    e = 0
    @body.each{|b| e += b.epot(@body)}
    e/2                             # pairwise potentials were counted twice
  end

  def write_diagnostics
    etot = ekin + epot
    print <<END
  E_kin = #{sprintf("%.3g", ekin)} ,\
 E_pot =  #{sprintf("%.3g", epot)} ,\
 E_tot = #{sprintf("%.3g", etot)}
END
  end
```

```
end

include Math

nb = ACS_IO.acs_read(NBody)
nb.write_diagnostics
```

This is all very similar to what we did within our N-body integrators.

7.4 Measurements

Alice: Let's start with a few small realizations, to see how large the fluctuations are from case to case, in the kinetic and potential energies, as well as the total energy.

```
|gravity> kali mkplummer2.rb -n 10 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 10
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 256998274192395533564317965818307356187
  E_kin = 0.203 , E_pot =  -0.472 , E_tot = -0.269
|gravity> kali mkplummer2.rb -n 10 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 10
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 946173897344225072951199417330177225 32
  E_kin = 0.153 , E_pot =  -0.259 , E_tot = -0.106
|gravity> kali mkplummer2.rb -n 10 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 10
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
```

```
Incremental indentation: add_indent = 2
             actual seed used: 19426478893611149359282198958273232067 1
  E_kin = 0.153 , E_pot =  -0.234 , E_tot = -0.0808
```

Bob: Better to increase the number of particles, to see whether we actually converge to something reasonable. An $N = 1000$ particle system should have $\sqrt{N}$ deviations around three percent or so.

```
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 29634114401708237048639934459956246016 5
  E_kin = 0.14 , E_pot =  -0.295 , E_tot = -0.155
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 29568503343167221084000432684410294834 5
  E_kin = 0.146 , E_pot =  -0.292 , E_tot = -0.146
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 17472379098586688010096623525556688155 8
  E_kin = 0.142 , E_pot =  -0.288 , E_tot = -0.146
```

Bob: Good enough! Also note that the virial theorem is obeyed quiet well: the total energy has a magnitude comparable to the kinetic energy, and the potential energy has a magnitude that is twice as large.

Alice: Yes, that is encouraging. The question is: are these magnitudes right?

Bob: Well, we can check by looking up what the kinetic and potential energy should be for a Plummer model. But something tells me that you'd rather compute it for yourself.

Alice: How did you guess! But we have to compute only one quantity. Since we start with a model that is in equilibrium, we can assume the virial theorem to hold, so as soon as we know one of the three energy diagnostics that you are printing out here, we can determine the other two immediately.

Bob: In that case, the total potential energy of the system would seem to be the best candidate.

Alice: I agree. We already know the density and the potential, so it is just a matter of integration.

7.5 Two Roads

Bob: Let's see. The total potential energy must be the sum of the potential energy that each mass element feels, with respect to the rest of the star cluster:

$$E_{pot} = \int_0^\infty \rho(r)\Phi(r)4\pi r^2 dr$$

Alice: Ah, but now you're double counting all pairwise interactions. You have to divide this by a factor two:

$$E_{pot} = \frac{1}{2}\int_0^\infty \rho(r)\Phi(r)4\pi r^2 dr \tag{7.1}$$

Bob: I guess that is right, or is it? The more I think about it, I wonder. You are talking about pairwise interactions, but we are letting each mass element feel the rest of the cluster through the whole potential $\Phi(r)$. If I were to think about pairwise interactions, I would immediately think about how the cluster can be build up by bringing each new mass element in from an infinite distance. At first there would be little mass, and later, when more and more mass is accumulated, each new mass element feels more attraction, and gains more energy falling in. Yet each mass element attracts, and is in turn attracted by the earlier mass that already has fallen in. So . . . I admit, I'm a bit confused.

Alice: I'm sure it is wrong to leave out the factor two. in the above equation. But I like the notion of building a cluster up, bit by bit. Given that we have spherical symmetry, we can make life easiest by letting each radial shell fall in, as if it was prefabricated. Like building a pre-fab house: you order the parts and put it together.

Bob: So every shell with destination radius r and destination thickness dr then has a mass $4\pi\rho(r)r^2dr$. Even if the shell has a much lower density at first, and

perhaps a much larger thickness, and certainly a much larger radius, when it settles in it must have that amount of mass, and since mass is conserved, this must be the mass that the pre-fab shell had before it got compressed into place. The term pre-fab is not a very good metaphor, perhaps, when we have to crunch the components, but let's not worry about words for now.

At the moment that the shell reaches its proper place, the material inside that shell has already been put there, while the material outside that shell is still waiting to fall in. So during the trip from an infinite distance with zero binding energy, down to its final destination, the infalling shell acquires an amount of potential energy of

$$-\frac{Gm(r)\rho(r)4\pi r^2 dr}{r}$$

Now if we look at it this way, and repeat the procedure for all mass shells from the inside outward, we must have for the total potential energy:

$$E_{top} = -\int_0^\infty \frac{Gm(r)\rho(r)4\pi r^2 dr}{r} \tag{7.2}$$

And I'm sure that there is no factor two here, no matter what you may argue about pairwise interactions.

Alice: Why do you call this energy *top* rather than *pot*?

Bob: Because I want to stress the difference with the expression you wrote down above, and you already claimed the label *pot*. So I just reversed the letters. Besides, my way of constructing the star cluster by letting mass rain down from infinity is surely a top-down method.

Alice: As you wish. And I agree that there should not be a factor two in your case, probably because on average, each mass element sees only half of the rest of the mass. But I must admit, I'm not totally sure that both expressions are correct. Why don't we work them both out, and see whether they boil down to the same result.

7.6 One Destination

Bob: You go first, since you wrote down the first expression. I'll hand you the expressions for the density and the potential, from way back when, eq. (1.11). Here they are again:

$$\begin{cases} \Phi(r) &= -\dfrac{GM}{a}\left(1+\dfrac{r^2}{a^2}\right)^{-1/2} \\ \\ \rho(r) &= \dfrac{3M}{4\pi a^3}\left(1+\dfrac{r^2}{a^2}\right)^{-5/2} \end{cases} \tag{7.3}$$

Alice: Okay. Eq. (7.1) then becomes:

$$\begin{aligned} E_{pot} &= \frac{1}{2}\int_0^\infty \rho(r)\Phi(r)4\pi r^2 dr \\ &= -\frac{1}{2}\int_0^\infty \frac{3M}{4\pi a^3}\left(1+\frac{r^2}{a^2}\right)^{-5/2}\frac{GM}{a}\left(1+\frac{r^2}{a^2}\right)^{-1/2} 4\pi r^2 dr \\ &= -\frac{3}{2}\frac{GM^2}{a^4}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-3} r^2 dr \end{aligned} \tag{7.4}$$

Before working this out further, let's see whether you get at least the same expression. Your turn!

Bob: I'll try. I first need to recover the expression for the accumulative mass, eq. (3.7):

$$m(r) = M\left(1+\frac{a^2}{r^2}\right)^{-3/2} = M\frac{r^3}{a^3}\left(1+\frac{r^2}{a^2}\right)^{-3/2} \tag{7.5}$$

where I added the right-hand term to make the expression more similar to the expressions that give us density and potential. My *top* expression then becomes:

$$\begin{aligned} E_{top} &= -\int_0^\infty \frac{Gm(r)\rho(r)4\pi r^2 dr}{r} \\ &= -\int_0^\infty GM\frac{r^3}{a^3}\left(1+\frac{r^2}{a^2}\right)^{-3/2}\frac{3M}{4\pi a^3}\left(1+\frac{r^2}{a^2}\right)^{-5/2}\frac{4\pi r^2 dr}{r} \\ &= -3\frac{GM^2}{a^6}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-4} r^4 dr \end{aligned} \tag{7.6}$$

Bob: Hmm. That doesn't look like your expression.

Alice: It is more complicated than my expression, alright, but an integration by parts may help out. May I try?

$$E_{top} = -3\frac{GM^2}{a^6}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-4} r^4 dr$$

$$
\begin{aligned}
&= -3\frac{GM^2}{a^6}\left\{-\frac{a^2}{6}\left(1+\frac{r^2}{a^2}\right)^{-3} r^3\Big|_0^\infty + \int_0^\infty \frac{a^2}{6}\left(1+\frac{r^2}{a^2}\right)^{-3} 3r^2 dr\right\} \\
&= -\frac{3}{2}\frac{GM^2}{a^4}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-3} r^2 dr \qquad (7.7)
\end{aligned}
$$

where the first term in the next-to-last line vanishes, because the contributions at both zero and infinity are zero.

Bob: Thanks! Now it is clear that E_{top} and E_{pot} are identically the same.

7.7 Potential Energy

Alice: All that is left to do is to solve it. Let me try another integration by parts:

$$
\begin{aligned}
E_{pot} &= -\frac{3}{2}\frac{GM^2}{a^4}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-3} r^2 dr \\
&= -\frac{3}{2}\frac{GM^2}{a^4}\left\{-\frac{a^2}{4}\left(1+\frac{r^2}{a^2}\right)^{-2} r\Big|_0^\infty + \int_0^\infty \frac{a^2}{4}\left(1+\frac{r^2}{a^2}\right)^{-2} dr\right\} \\
&= -\frac{3}{8}\frac{GM^2}{a^2}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-2} dr \qquad (7.8)
\end{aligned}
$$

where again the first term in the next-to-last line vanishes for the same reason as before.

Now how shall we tackle this integral? I wonder how we can simplify this further.

Bob: Now this is really simple enough for my taste. Yesterday I let you get away with going back to square one, but if we keep doing that, we'll never build an N-body environment.

We should be able to find this integral easily in a book with a table of integrals, or by using a symbolic package. Ah, you see, here it is, in good old Abramowitz and Stegun, right at the beginning of their list of most common integrals:

$$
\int \frac{dx}{(x^2+a^2)^2} = \frac{1}{2a^3}\arctan\frac{x}{a} + \frac{x}{2a^2(x^2+a^2)} \qquad (7.9)
$$

Alice: Okay, I give in.

Bob: Substituting eq. (7.9) into eq. (7.8), we get:

$$
E_{pot} = -\frac{3}{8}\frac{GM^2}{a^2}\int_0^\infty \left(1+\frac{r^2}{a^2}\right)^{-2} dr
$$

$$
\begin{aligned}
&= -\frac{3}{8} GM^2a^2 \int_0^\infty \left(a^2+r^2\right)^{-2} dr \\
&= -\frac{3}{8} GM^2a^2 \left\{ \frac{1}{2a^3} \arctan\frac{r}{a} + \frac{r}{2a^2\left(r^2+a^2\right)} \right\}\Bigg|_0^\infty \\
&= -\frac{3}{8} GM^2a^2 \left\{ \frac{1}{2a^3} \frac{\pi}{2} \right\} \\
&= -\frac{3\pi}{32} \frac{GM^2}{a} \qquad (7.10)
\end{aligned}
$$

Alice: It looks plausible: at least it has the right dimensions and the right dependency on mass and structural parameter.

7.8 Validation

Bob: I'm really curious to see whether my code gives the potential energy that we have now calculated, at least within the statistical noise. In my code, following Aarseth *et al.*, I have $G = M = a = 1$, so I should find a value of

$$E_{pot} = -\frac{3\pi}{32} \approx -0.2945$$

For the kinetic and total energy, we should similarly expect:

$$E_{kin} = -E_{tot} = \frac{3\pi}{64} \approx 0.1473$$

Alice: And look, those are indeed the type of numbers we got. Why don't you create a few more 1000-particle realizations.

Bob: Here are a few more.

```
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
            actual seed used: 289599681324346027028475083960495163734
  E_kin = 0.145 , E_pot =  -0.288 , E_tot = -0.143
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
```

```
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 110493489622357455400959075383471675626
  E_kin = 0.145 , E_pot =  -0.289 , E_tot = -0.145
|gravity> kali mkplummer2.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 372105320728436010054724616167392048 13
  E_kin = 0.151 , E_pot =  -0.286 , E_tot = -0.135
```

You're right: the numbers work out beautifully.

Alice: Congratulations!

Chapter 8

Quartile Checks

8.1 Quartiles

Bob: Hi Alice, what's next? I can think of a few further improvements to my Plummer code, but do you have something specific, you'd like to add?

Alice: I'm happy that we have done this energy checking, which seems to indicate that your code probably does produce Plummer's model correctly. However, I'd like to be even more sure, since it would really be terrible if we start our future simulations with the wrong initial conditions. That would make debugging our evolution codes quite confusing, to say the least.

Bob: There are many ways to check things. The question is, what would be the simplest way. Ah, how about just looking at mass shells, shells with a certain mass fraction, and check that they have the correct radius? After all, we computed the cumulative mass $m(r)$ as a function of the radius already.

Alice: That is a good idea. For one thing, the half-mass radius should come out with the correct value. And for good measure, we may as well compute the quarter-mass radius and the three-quarter-mass radius. In other words: the radii of the three quartiles, within which 1/4, 1/2, and 3/4 of the total mass are enclosed.

Bob: That we should be able to do from scratch. We can start again with minimal versions of the `Body` and `Nbody` classes, just enough to read a snapshot in, and initialize everything properly. The only thing to do is add a method `quartiles` that prints out the three quartiles.

Fortunately, Ruby has a handy `sort` method, that comes with the `Array` class. As usual, it has an associated `sort!` method that affects the array itself, unlike `sort` that returns a new array. Let's see. We have to sort the particles in radial order, using $r = |\mathbf{r}|$, the distance from the center. It will be easier to use the square instead, $r^2 = \mathbf{r} \cdot \mathbf{r}$. This can be done in three lines:

```
def order_squared_radii
  a = []
  @body.each{|b| a.push b.pos*b.pos}
  a.sort!
end
```

The first line defines `a` to be an empty array. The second line fills the array with the squares of the radial positions of all particles. Then the third line sorts that array. Couldn't be simpler!

8.2 Coding

Alice: Can I write the `quartiles` method? I'd like to get some more experience with Ruby.

Bob: Here is the keyboard!

Alice: Since you have defined `order_squared_radii` as a member function, we may as well define `quartiles` as a member function too. We can then immediately invoke `order_squared_radii` to create our order list of particle positions. All we have to do then is to cut the list in four equal parts, and document the places where we cut the list. Hey, that is too simple to learn much about Ruby! This should do the job:

```
def quartiles
  a = order_squared_radii
  n = a.size
  r_1 = a[(n/4.0).round - 1]
  r_2 = a[(n/2.0).round - 1]
  r_3 = a[(n*3/4.0).round - 1]
  print "The values of the three quartiles for r(M) are:\n"
  print "  r(1/4) = "
  printf("%.4g\n", r_1)
  print "  r(1/2) = "
  printf("%.4g\n", r_2)
  print "  r(3/4) = "
  printf("%.4g\n", r_3)
end
```

Bob: I bet it does. I saw you looking up `round`: indeed, that is a predefined method that rounds a floating point number to an integer. In case of a number exactly in between, such as 2.5, it rounds up, to 3.0. You could have used integer

division directly, instead of floating point division, but I agree that what you wrote might be easier to understand.

Alice: We could quibble about exactly where to choose the boundaries for the quartiles. For example, for a four-particle system, things come out just right, but for a five particle system, the half mass radius encloses 3/5 of the mass. We could have take the average of the two-particle and three-particle distance, but that would have been overkill, in my opinion. Anyway, the fluctuations in the positions of the particles, in an N-body system will be of order $\sqrt{N} \gg 1$, so we don't have to worry about enclosing one particle more or less, at least for $N \gg 1$.

8.3 Code

Bob: Let me print out the whole program, and write it out to a file called `quartiles1.rb`:

```
require "acs"

class Body

  attr_accessor :mass, :pos, :vel

  def initialize(mass = 0, pos = Vector[0,0,0], vel = Vector[0,0,0])
    @mass, @pos, @vel = mass, pos, vel
  end

end

class NBody

  attr_accessor :time, :body

  def initialize
    @body = []
  end

  def order_squared_radii
    a = []
    @body.each{|b| a.push b.pos*b.pos}
    a.sort!
  end

  def quartiles
    a = order_squared_radii
```

```
    n = a.size
    r_1 = a[(n/4.0).round - 1]
    r_2 = a[(n/2.0).round - 1]
    r_3 = a[(n*3/4.0).round - 1]
    print "The values of the three quartiles for r(M) are:\n"
    print "  r(1/4) = "
    printf("%.4g\n", r_1)
    print "  r(1/2) = "
    printf("%.4g\n", r_2)
    print "  r(3/4) = "
    printf("%.4g\n", r_3)
  end

end

include Math

nb = ACS_IO.acs_read(NBody)
nb.quartiles
```

8.4 Testing

Alice: Let's try it out:

```
|gravity> kali mkplummer2.rb -n 100 | kali quartiles1.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 173605212232326196847944258433114314381
The values of the three quartiles for r(M) are:
  r(1/4) = 0.5172
  r(1/2) = 1.209
  r(3/4) = 3.945
```

```
|gravity> kali mkplummer2.rb -n 100 | kali quartiles1.rb
==> Plummer's Model Builder <==
```

```
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 31058758856593447541572879260623764110 8
The values of the three quartiles for r(M) are:
  r(1/4) = 0.771
  r(1/2) = 2.1
  r(3/4) = 3.997
```

```
|gravity> kali mkplummer2.rb -n 100 | kali quartiles1.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 313271208733788070196627795604435025509
The values of the three quartiles for r(M) are:
  r(1/4) = 0.6326
  r(1/2) = 1.9
  r(3/4) = 4.575
```

Quite noisy. Not too surprising: for the inner quartile we are dealing with only 25 particles, so we should expect $1/\sqrt{25}$ noise. Indeed, there are fluctuations of about twenty percent in the first quartile.

Bob: Let's try 10,000 particles instead:

```
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles1.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 14057914387865372667907540486041835743 8
The values of the three quartiles for r(M) are:
  r(1/4) = 0.6639
  r(1/2) = 1.722
  r(3/4) = 4.778
```

```
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles1.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 161364654345763278421094667025804062569
The values of the three quartiles for r(M) are:
  r(1/4) = 0.6711
  r(1/2) = 1.728
  r(3/4) = 4.707
```

```
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles1.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 327124622031046078183213047956695327215
The values of the three quartiles for r(M) are:
  r(1/4) = 0.6475
  r(1/2) = 1.722
  r(3/4) = 4.808
```

Not bad! Fluctuations of a few percent, at most, as it should be. Encouraging.

8.5 Checking the Math

Alice: Now let's check the numbers. We have used already a few times eq. (3.7):

$$m(r) = M\left(1+\frac{a^2}{r^2}\right)^{-3/2} \tag{8.1}$$

Or in our units, with $M = a = 1$:

$$m(r) \;=\; r^3 \left(1 + r^2\right)^{-3/2} \tag{8.2}$$

While we walked through your code, we reproduced the inverted form, expressing $r(m)$, in eq. (3.3) as:

$$r(m) \;=\; \left(m^{-2/3} - 1\right)^{-1/2} \tag{8.3}$$

Bob: I have one problem with this kind of testing. First we write a code, using the above equation. Then we test the code, using the same equation. This will catch implementation bugs alright, but it won't catch any mistake we might have made in the above equation. In this case the equation is simple, but as a general procedure, I'm not sure how much this really buys us.

Alice: Excellent point! And there is an easy check: Let us insert eq. (8.3) into the right-hand side of eq. (8.2), to check whether we get the original amount of mass back. Let us call the left-hand side of eq. (8.2) $\tilde{m}$, so that we can distinguish it from the mass m we start with, in eq. (8.3). Eq. (8.2) then reads:

$$\begin{aligned}
\tilde{m}(r) &= r^3 \left(1 + r^2\right)^{-3/2} \\
&= \left(m^{-2/3} - 1\right)^{-3/2} \left(1 + \left(m^{-2/3} - 1\right)^{-1}\right)^{-3/2} \\
&= \left(m^{-2/3} - 1\right)^{-3/2} \left(\frac{\left(m^{-2/3} - 1\right) + 1}{m^{-2/3} - 1}\right)^{-3/2} \\
&= \left(m^{-2/3} - 1\right)^{-3/2} \left(m^{-2/3}\right)^{-3/2} \left(m^{-2/3} - 1\right)^{3/2} \\
&= m
\end{aligned} \tag{8.4}$$

This was an almost trivial exercise. But I'm glad we checked.

Bob: Yes, many bugs turn out to be 'almost trivial' as well, and yet they can waste many hours of your time while you're chasing them. By the way, wouldn't it have been much faster to substitute eq. (8.3) into the right-hand side of eq. (8.1)? Calling the $m(r)$ used in eq. (8.1) $\hat{m}(r)$, we would have gotten:

$$\begin{aligned}
\hat{m}(r) &= \left(1 + \frac{1}{r^2}\right)^{-3/2} \\
&= \left(1 + \left(m^{-2/3} - 1\right)\right)^{-3/2} \\
&= \left(m^{-2/3}\right)^{-3/2}
\end{aligned}$$

$$= \quad m \tag{8.5}$$

Alice: You're right, that is faster. And if we were to write a text book, we would certainly hide the fact that we did it first the hard way around. But in practice, who cares? The important thing is to check that things are correct. And they are, we now know.

8.6 Checking the Code

Bob: Let's compute the expected quartile radii, from eq. (8.3):

$$r(1/4) = \left(2^{4/3} - 1\right)^{-1/2} \approx 0.8111 \tag{8.6}$$

$$r(1/2) = \left(2^{2/3} - 1\right)^{-1/2} \approx 1.305 \tag{8.7}$$

$$r(3/4) = \left(2^{4/3}3^{-2/3} - 1\right)^{-1/2} \approx 2.175 \tag{8.8}$$

Alice: Those numbers look *completely* different from what we got from our code! And the strange thing is, the first number comes out too small, while the next ones are too large, especially the last one. So it is not a matter of having overlooked a scale factor, or something like that.

Bob: What a pity! I was more or less convinced that my Plummer code was correct. Obviously there is something seriously wrong with the way particles are sprinkled into space.

Alice: What is really strange is the fact that we got the energy correct. First of all, the virial theorem did hold, something that is typically violated if you make a random mistake in the energies, since it is unlikely that you will make the same mistake in the kinetic as in the potential energy. But secondly, we actually computed the energy, and we found that your code produced it correctly.

Bob: Yes, that is puzzling. Unless . . .

Alice: . . . unless my quartile code is wrong. But that is also very unlikely. How much can there be wrong in just a few lines of code?

Bob: It wouldn't be the first time, to find a nasty bug in a small code. I guess that's why they are called bugs! Bugs are small, and can hide even in a code of a single line. Let's have a look.

Alice: Yes, the good thing about a small code is that it makes sense to go through each line, one by one. And there are less than ten lines here where actual calculations are done.

8.7 Checking the Code-Checking Code

Bob: Let's be systematic. The driver part calls `quartiles`, and that method in turn starts by calling `order_squared_radii`. You like long names, don't you? You could have called it `place_an_order_to_order_squared_radii`.

Alice: Yes, I do like long names, since that way I will remember later what was doing what. I hate that old style in which it probably would have been called `srtrsq` for 'sorting r squared'. Forty years ago there was a reason, when single memory locations were expensive, but that excuse has long since gone.

Bob: Well, I can't for the life of me see anything wrong with `order_squared_radii`. Back to `quartiles`. In the second line you find the number of particles. Nothing wrong with that either. In the third line you cut the list at the place of the first quarter. Indeed, in that way you are bound to find the mass shell that contains one quarter of the mass, and precisely the inner quarter.

Alice: Iron logic, and I, too, cannot see anything wrong with that.

Bob: So we both agree that `a[(n/4.0).round - 1]` returns what you have ordered: the squared radius of the inner quarter of . . .

Alice: . . . the *squared* radius!!

Bob: Ah, yes, that's the problem! You had forgotten that you ordered the *squares* of the radial distances, for convenience. Instead, you just assigned the values to the quartile radii themselves.

Alice: That must have been the problem. And that immediately explains why the errors grew bigger for larger values of r. Okay, that is easy to fix. Let us rewrite these three lines, and put the result in a file called `quartiles.rb`:

```
r_1 = sqrt(a[(n/4.0).round - 1])
r_2 = sqrt(a[(n/2.0).round - 1])
r_3 = sqrt(a[(n*3/4.0).round - 1])
```

And let's test it right away:

```
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 280304577016587325675314010204715072766
The values of the three quartiles for r(M) are:
```

```
  r(1/4) = 0.81
  r(1/2) = 1.312
  r(3/4) = 2.166
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 151057629702332869562586198630961298683
The values of the three quartiles for r(M) are:
  r(1/4) = 0.8066
  r(1/2) = 1.301
  r(3/4) = 2.187
|gravity> kali mkplummer2.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 187229995952768761158374448960407058 55
The values of the three quartiles for r(M) are:
  r(1/4) = 0.8226
  r(1/2) = 1.322
  r(3/4) = 2.178
```

Bob: Wonderful! So my Plummer code was correct after all. But I'm glad we checked so thoroughly.

Alice: You may laugh, and perhaps I'm getting to much addicted to testing, but let me do one final little check. Having just made a really silly mistake determining the quartile radii, I feel I cannot take anything for granted. Let me see whether the quartile radii are *really* quartile radii. Starting with:

$$m(r) \;=\; \left(1 + r^{-2}\right)^{-3/2} \tag{8.9}$$

We have

$$m(r(1/4)) = \left(1 + \left(\left(2^{4/3} - 1\right)^{-1/2}\right)^{-2}\right)^{-3/2} = \left(1 + 2^{4/3} - 1\right)^{-3/2} = 2^{-2} = 1/4 \tag{8.10}$$

Similarly with the half-mass radius $m(r(1/2)) = 1/2$, and:

$$m(r(3/4)) = \left(1 + 3^{-2/3}2^{4/3} - 1\right)^{-3/2} = 3.2^{-2} = 3/4 \tag{8.11}$$

Bob: Now that is what I would call overkill. But you're right, it doesn't hurt, and better safe than sorry.

Alice: Indeed!

Chapter 9

Standard Units

9.1 Confusion

Alice: Now that we have a working and well-tested code to generate realizations of Plummer's model, we have to decide how to scale the output. We simply started with the coordinate system that was used by Aarseth and his friends, in which $G = M = a = 1$. While that is a reasonable choice, there are other choices as well.

The problem is: if we make one choice in one program, and another choice in another program, we are inviting disaster. If you then put those programs together, as different modules in a single larger context, you will generate nonsense.

Bob: Well, let's make a particular choice then, and stick to it.

Alice: What have you been using so far?

Bob: I have never settled on one particular choice. It always depended on the problem at hand, the code I used, and the preferences of my collaborators.

Alice: People always underestimate the importance of standardization. It takes some work, of course, for a community to settle on a standard. However, it takes far more work if you always have to transform between different systems, even in the unlikely case that you do not make mistakes. Scaling the results from one system to the others is tricky.

Bob: Tell me about it! Or better, tell my students about it. One thing students always have great trouble with is scaling the results from an N-body simulation back to physical quantities, expressed in physical or astrophysical units.

Alice: That is not surprising. And I must say, I often have to think carefully about such questions. It may be trivial, from a scientific point of view, but in practice, it is easy to make a mistake. It is one thing to get as an answer that

your star cluster retains a final mass of `0.765` at the end of a run, but it is quite another to translate that into grams, or into solar masses.

Bob: At least the physical units are standardized. Mass comes in grams, or in kilograms. It is annoying that there are still two sets of units in general use in astrophysics, *MKS* and *cgs*, but at least the conversion between those two is relatively straightforward, just a factor of a thousand in the case of masses. And similarly, astrophysical units are standardized: mass general comes in units of a solar mass. But in computer simulations, everybody uses whatever convention they like.

I often find it difficult to interpret the results that come from running someone else's code. Not everybody clearly documents what units they used. I know, I know – I just told you that I have never settled on a single system of units either. I wish there was such a system.

9.2 A Standard

Alice: But there *is* a standard for computer simulations in stellar dynamics. And I have been using that standard ever since I started running simulations.

Bob: What are they called?

Alice: Hmm. They are generally referred to as 'standard units.' Perhaps that is one reason that they haven't found general acceptance yet. Maybe we should give them a real name! They are sometimes also referred to as 'Heggie units,' since Douglas Heggie was the first one, as far as I know, to stress the need for such standardization.

Bob: Did he publish the definition of his standards?

Alice: Yes, in a paper with Bob Mathieu, back in 1986, as a contribution to a conference where this issue of standardization was discussed. The reference is *Standardised Units and Time Scales*, by Douglas Heggie and Robert Mathieu, and it appeared in 1986, on page 233 in *The Use of Supercomputers in Stellar Dynamics*, edited by Steve McMillan and Piet Hut, and published by Springer.

Bob: How did they define their units?

Alice: They took the gravitational constant and total mass of a star cluster to be unity, and they took the total energy to be -1/4.

Bob: Why one quarter? Unlike the first two choices, that doesn't sound very natural to me. Why not unity?

Alice: Actually, the origin of their definition stems from the fact that they took the virial radius r_V to be unity. I should have introduced their choice as:

$$G = M = r_V = 1 \tag{9.1}$$

The fact that $E_{tot} = -0.25$ is a consequence from this fact.

Bob: I know the virial *theorem*, but what is a virial *radius*?

Alice: For an equal mass system, an elegant definition is: the virial radius is the inverse of the average inverse distance between particles in an N-body system. Expressed as a formula:

$$\frac{1}{r_V} = \left\langle \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \right\rangle_{i \neq j} \tag{9.2}$$

averaged over all particle pairs i, j. For a general system of particles with masses m_i and total mass M, the definition is:

$$\frac{M^2}{r_V} = \sum_i \sum_{\substack{j \\ j \neq i}} \frac{m_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{9.3}$$

and of course, we recognize that this is the potential energy of an N-body system, for $G = 1$, apart from a factor of two, because we have now counted every pair twice. So we can write the potential energy of an N-body system as:

$$E_{pot} = -G \sum_{i<j} \frac{m_i m_j}{|\mathbf{r}_i - \mathbf{r}_j|} = -\tfrac{1}{2} \frac{GM^2}{r_V} \tag{9.4}$$

Bob: And now we can use the virial theorem, which tells us that the magnitude of the total energy is half that of the potential energy, to write:

$$E_{tot} = \tfrac{1}{2} E_{pot} = -\frac{1}{4} \frac{GM^2}{r_V} \tag{9.5}$$

and with our previous choice of $G = M = 1$, we get:

$$E_{tot} = -\frac{1}{4} \frac{1}{r_V} \tag{9.6}$$

Now I see what you meant, with the total energy of -1/4 being a consequence the choice of $G = M = r_V = 1$. But I'm still puzzled. What is so special about the virial radius that you want to set it equal to unity? Why not set the total energy equal to unity, say?

9.3 Motivation

Alice: I think that the original idea was that we would like to choose the most natural units for the three basic physical units, namely those of mass, length,

and time. But since we also like to scale the gravitational constant to unity, we have only two degrees of freedom left over. The total mass is an obvious candidate, since it appears in many equations as GM, which is nice to forget about by equating it to 1. It also means that for equal-mass N-body systems, you can count on each particle always having a mass of $1/N$.

The only remaining question is: what to do with the last degree of freedom. Do you want to find a natural length scale or a natural time scale, or do you want to take a more derived quantity, not directly coupled to the basic physical units, such as the energy? I for one agree with Heggie and Mathieu that it is more elegant to choose a basic physical quantity, either a length or time scale.

Bob: I think I would prefer total energy, derived or not. But if you insist on purity, well, a natural length scale for a star cluster is the half-mass radius r_h. And a natural time scale in turn is the crossing time t_h at the half-mass radius, the typical time for a particle to cross the system, starting at the half-mass radius.

Alice: It is already clear from your suggestion that choosing a length scale is somewhat more natural than choosing a time scale, since in your time scale definition you make use of an earlier length scale definition.

Bob: So what is wrong with the choice of $G = M = r_h = 1$?

Alice: There is nothing wrong with that, and people have used that choice as well. The problem is that in that case the total energy has a non-obvious value, typically somewhere like $|E_{tot}| = 0.2$, but not exactly.

Bob: Well, my original suggestion was to make $|E_{tot}| = 1$. Not only is that an exact number, it is a very simple number. I'm still not sure what is wrong with that.

Alice: If you choose the absolute value of the total energy to be one, your half-mass radius comes out to be very small, about 1/5, and that is not a very natural value.

Bob: Hmm, yes, it would be a bit of a nuisance, to deal with a core radius of $r_c = 0.02$, say, and then having to remember that we are dealing with a not very concentrated cluster, since $r_h = 0.2$ and therefore $r_c/r_h = 0.1$.

Alice: To sum up, we really would like to have a system of length units, in which the half-mass radius is close to unity. However, the half-mass radius is not a conserved quantity, and as soon as you start a simulation, the half-mass radius may change. Therefore, it is better to take a conserved quantity, such as the total energy, as a gauge, and give it a simple value in such a way that it implies that the unit of length is at least close to the half-mass radius. This must have been the sort of thinking that went into the definition of the standard units, I'm pretty sure.

Bob: That all makes sense. But in practice, the half-mass radius does not change much, if you simulate a star cluster. Only after core collapse does the

half-mass radius begin to expand.

Alice: You are used to dealing with a system that starts in dynamic equilibrium. However, if you start with a cold collapse, or a system that has too much kinetic energy and starts off expanding, in both cases the half-mass radius will change right away, while the total energy will remain conserved.

Bob: Okay, I see the advantages of the standard units. And since I don't feel very strong about my other two candidates for standardization, I'm happy to use those virial units, what did you call them, Heggie units?

Alice: Yes, were it not for the fact that Douglas Heggie is a modest gentleman, who would be the first to point out that those units have been used by others before he suggested them. Virial units might actually be a reasonable name; I haven't heard that expression yet.

Bob: Hmmm. It just slipped out, but to me it sounds too much like a medical term, reminding me of a virus. And I certainly don't like computer viruses. I prefer the term Heggie units: he should get credit for his suggestion.

Alice: We'll see what the field decides.

9.4 Approximations

Bob: By the way, I was impressed by the fact that you juggled those numbers so easily, like that value 0.2 that you pulled out of a hat. What was that again?

Alice: That was the value for the half-mass radius, if you would insist on a total mass of unity.

Bob: Ah yes, did you make that up to impress me, or did you calculate or guestimate that quickly?

Alice: None of the above. Since I have been working with these standard units for a long time, and especially since I have been teaching it to my students, some of these numbers just stick in my mind. You mentioned from the start that students always have problems with scaling, and my students are no exception.

Bob: I guess the counter-intuitive aspect is that if your ruler shrinks, everything you measure becomes bigger, and similarly, if you take a ruler with larger units of length, the whole world gets smaller, in terms of the values you read off. Knowing where to multiply and where to divide is something that requires some thought. With one ruler changing, you have to be careful, and if you simultaneously change your ruler, your clock, and your scales, changing your units of length, time, and mass, it is real easy to go wrong.

Alice: I know from experience! And still, I always have to double check.

Bob: Glad to hear we share the same problem. And just to make sure that I can buy into your story, shall we quickly check with Plummer's model as a concrete application how we can derive the numbers you mentioned?

Alice: Good idea! It never hurts to check, as we've now seen a number of times. But I don't like working with factors like $3\pi/16$ and $(2^{2/3}-1)^{-1/2}$. Let's make some simplifications, trying to use only fractions like 1/3 and 1/4 and the like, but let's not get more accurate. That should be enough to show our main point.

So let us start with the original expressions for the total energy E_{tot} which we may as well abbreviate as E, the half-mass radius r_h and the virial radius r_V, given in terms of the structural length a. Remember that a was what we first encountered as the softening length, when we smoothed the potential of point particles. We started off with our Plummer potential

$$U(r) \;=\; -\,GM\frac{1}{(r^2+a^2)^{1/2}} \tag{9.7}$$

We also computed the potential energy of Plummer's model in eq. (7.10). The virial theorem tells us that the total energy is just half that value:

$$E \;=\; -\,\frac{3\pi}{64}\,\frac{GM^2}{a} \tag{9.8}$$

In terms of a, we can write, with $G = M = 1$:

$$E \;=\; -\,\frac{3\pi}{64}\frac{1}{a} \;\approx\; \frac{1}{10}\times\frac{3}{2}\frac{1}{a} \tag{9.9}$$

$$r_h \;=\; \left(2^{2/3}-1\right)^{-1/2} a \;\approx\; \frac{4}{3}a \tag{9.10}$$

$$r_V \;=\; \frac{16}{3\pi}a \;\approx\; \frac{5}{3}a \tag{9.11}$$

The expression for the half mass radius r_h we already determined earlier, in (8.7), and the last result follows directly from eq. (9.6), which tells us that the virial radius is $r_V = -1/(4E)$.

I have split off the factor 1/10 in the expression for the energy, so that we can deal with remaining numbers that are all of order unity, in the form of fractions of small integers.

All we have to do now is to choose different values for a. For each choice of a, we can see explicitly how everything else will receive different values. The structural length a plays the role of our ruler.

9.5 Three Round Numbers

Bob: That is a nice way to lay it all out. Okay, so we have talked about three

choices of units, based on what we choose to set equal to one: the energy, the half-mass radius, and the virial radius. Let us start with the energy

Alice: If we take $E = 1$, we are forced to take $a = 3/20$ which implies $r_h = 1/5$ and $r_V = 1/4$.

Bob: How simple! Yes, it is clear now, and you were right about that value of roughly 0.2 for the half-mass radius. I'm curious to see what the other two choices will lead to. Let us make a list:

$$E = -1 \Rightarrow r_h \approx 1/5, r_V = 1/4 \tag{9.12}$$

$$r_h = 1 \Rightarrow E \approx -1/5, r_V \approx 5/4 \tag{9.13}$$

$$r_V = 1 \Rightarrow E = -1/4, r_h \approx 4/5 \tag{9.14}$$

Alice: Yes, that is a good summary, and it is important to indicate which relations are approximate and which are exact because they follow from the definitions.

Bob: And these are useful numbers to remember. Actually, now that we have decided to adopt the 'standard units' as standard units (we have to come up with a better name), it is only the bottom line that is really worth remembering.

Alice: And since $E = -1/4$ for all models in the standard units, for Plummer's model there is only one number to remember: the fact that the half-mass radius is roughly 4/5.

Bob: Ah, but there is also the structural length. Let's see, in standard units that is roughly 3/5. That is a second useful number to remember, since it gives a measure for the size of the core of the potential. So we have:

$$r_V : r_h : a \approx 5 : 4 : 3 \tag{9.15}$$

And this shows that Plummer's model is not very centrally condensed: the core is barely smaller than the half-mass radius.

Alice: Talking about central concentrations, it would be nice to throw in the core radius as well, for good measure.

9.6 Surface Density

Bob: The problem with the core radius is that there are several definitions. Which one do we choose?

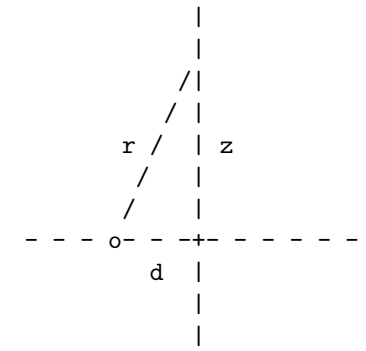
Alice: There may be several, but the definition that I have seen used most often is the one that appeals to observers: it defines the core radius as the distance from the center on the sky where the projected light density Σ has dropped by a factor of two, with respect to the central value. In other words

$$\Sigma(r_c) = \tfrac{1}{2}\Sigma(0)$$

Bob: It may be useful for observers, but for a theorist it is rather unnatural to do a line-of-sight integration through a model. But I too have come across this definition quite often, so let's adopt it.

Alice: We'll have to do the integral of course. Better first to draw a picture.

[We should put a picture here]



From this figure it is clear that the surface density, at projected distance d from the center, is given by:

$$\begin{aligned}
\Sigma(d) &= \int_{-\infty}^{\infty} \rho(r(z))dz \\
&= \int_{-\infty}^{\infty} \rho(\sqrt{d^2+z^2}dz \\
&= \frac{3}{4\pi}\int_{-\infty}^{\infty} (1+d^2+z^2)^{-5/2}dz \qquad (9.16)
\end{aligned}$$

where I have used eq. (1.10) for the density.

Bob: Now before you will begin to solve this with pen and paper, let me give you the answer, by using a symbolic manipulation program. Here it is: the answer for the definite integral is $4/(3(1+d^2)^2)$.

Alice: The impatience of youth! But okay, I'll use your value. We then have:

$$\begin{aligned}\Sigma(d) &= \frac{3}{4\pi}\frac{4}{3}\frac{1}{\left(1+d^2\right)^2} \\ &= \frac{1}{\pi}\frac{1}{\left(1+d^2\right)^2}\end{aligned}$$

Bob: And if you want to see the complete dependence on the original variables, dimensional analysis shows that:

$$\Sigma(r) = \frac{M}{\pi a^2}\left(1+\frac{r^2}{a^2}\right)^{-2} \tag{9.17}$$

where I have switched back to r notation.

9.7 More Round Numbers

Alice: We're almost there: the core radius is defined as the place where the surface density has dropped by a factor of two:

$$\Sigma(r_c) = \tfrac{1}{2}\Sigma(0) \quad \Rightarrow \quad \left(1+\frac{r_c^2}{a^2}\right)^{-2} = \tfrac{1}{2} \quad \Rightarrow$$

$$r_c = \frac{a}{\sqrt{2}} \tag{9.18}$$

So this is the value for the core radius.

Bob: And that gives us a third number to remember for Plummer's model:

$$r_V : r_h : a : r_c \approx 5 : 4 : 3 : 2 \tag{9.19}$$

Alice: I like that! An unlikely simple progression. I thought I knew Plummer's model by now, but I had not realized how simple the ratios for these main numbers are. Okay, let's remember them all!

Bob: Or, to be a bit more lazy, I may just remember that the half-mass radius is twice as large as the core radius, that the structural radius lies about half-way in between, and that the virial radius is a bit larger than the half-mass radius.

Alice: If we want to be really lazy, we should group all these results together, so that we can later easily come back to look them up:

$$\begin{cases} r_c &= 2^{-1/2}a \\ r_h &= \left(2^{2/3}-1\right)^{-1/2}a \\ r_V &= \dfrac{16}{3\pi}a \end{cases} \tag{9.20}$$

Actually, what we really need is to know what these quantities look like in standard units:

$$\begin{cases} r_c &= \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2}} &\approx 0.4165 &\sim 0.4 \\ a &= \dfrac{3\pi}{16} &\approx 0.5890 &\sim 0.6 \\ r_h &= \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{2/3}-1}} &\approx 0.7686 &\sim 0.8 \\ r_V &= 1 &= 1 &= 1 \end{cases} \tag{9.21}$$

In the last column we recognize those nice round numbers we just discovered. And while we're at it, we may as well throw in the first and third quartile radii, which we determined before, in eqs. (8.6) and (8.8):

$$\begin{cases} r(1/4) &= \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{4/3}-1}} &\approx 0.4778 &\sim 0.5 \\ r(3/4) &= \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{4/3}3^{-2/3}-1}} &\approx 1.2811 &\sim 1.3 \end{cases} \tag{9.22}$$

Bob: That is a very useful collection of numbers. For example, it tells us immediately that the core of Plummer's model contains a little less than one quarter of the total mass.

Also, if you are using softened particles, you may want to know how extended the mass distribution is. These relations tell you that within one softening length there is just a bit more than a quarter of the total mass, but that within two softening lengths you already have almost three quarters of the total mass.

Alice: And if we agree to stick to these standard units, let us adapt your code, so that it generates Plummer model realizations with the right units.

Bob: Okay, that should be easy.

Chapter 10

Two More Code Versions

10.1 Standard Units

Alice: In order to adapt your code `mkplummer2.rb` to standard units, we have to change scales, both in position space and in velocity space. Let us start with positions. In our old system, we had a structural length $a = 1$. In that system, the virial radius was:

$$r_V = \frac{16}{3\pi}$$

Let us call this quantity our scale factor. In standard units, the virial radius $r_V = 1$. This means that we have to divide the virial radius by the scale factor, when we switch from the old units to standard units.

Bob: And this means that *all* distances have to be scaled this way, since that is the definition of a scale factor after all. I'll copy the previous code in a file called `mkplummer3.rb`, and add this scaling. That will take only two lines. Here, right at the top of the `mkplummer` method, I'm adding this line:

```
scalefactor = 16.0 / (3.0 * PI)
```

Then at the place where we initialize the positions for each body, we have to divide by that scale factor, just as we had to divide the virial radius by that factor:

```
b.pos = spherical(radius) / scalefactor
```

Alice: That must be right. Now the velocity scaling is a bit more tricky. How shall we approach that?

Bob: Velocities come into the kinetic energy. Since the masses of the particles are not affected, the velocities must scale like the square root of the energy.

Alice: Good point! The total mass of a cluster with N stars is unity, and each star therefore has a mass of $m = 1/N$ in both systems, or old system of units, and the standard system of units. It is only a change in the magnitude of their velocities that can make up for a change in kinetic energy – or in total or potential energies, for that matter, since all these quantities are related. And we know that the potential energy can only scale with the distances, since $G = M = 1$ in both the old system and the new one.

Bob: In other words, we have for the potential energy of the cluster as a whole:

$$E_{pot} \propto \frac{1}{r}$$

and

$$E_{kin} \propto v^2$$

Since they must remain proportional to each other, with the virial factor of two between them, we know that

$$v \propto r^{-1/2}$$

Alice: That is an extremely sloppy notation, using r and v as arbitrary lengths and velocities, but I see what you mean, and yes, obviously the square of the velocities transform inversely proportional to the distances.

Bob: This means that I can add the following conversion factor to the code, at the point where the velocities are calculated:

```
b.vel = spherical(velocity) * sqrt(scalefactor)
```

10.2 Checking Quartiles

Alice: That must be correct, but let's make sure that we get the right energy and the right quartiles again – and this time in standard units.

Bob: Let's start with the quartiles:

```
|gravity> kali mkplummer3.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
```

```
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 29334875186857296662463761519600967784 9
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4808
  r(1/2) = 0.7803
  r(3/4) = 1.292
|gravity> kali mkplummer3.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 736950900700500896792996485970238541 50
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4728
  r(1/2) = 0.7697
  r(3/4) = 1.301
|gravity> kali mkplummer3.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 309346344472717883480813094012221223722
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4848
  r(1/2) = 0.7706
  r(3/4) = 1.287
```

Alice: These are indeed what we had derived a bit earlier, in eqs. (9.21) and (9.22):

$$\left\{ \begin{array}{lcll} r(1/4) & = & \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{4/3}-1}} & \approx \quad 0.4778 \\ r_h & = & \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{2/3}-1}} & \approx \quad 0.7686 \\ r(3/4) & = & \dfrac{3\pi}{16}\dfrac{1}{\sqrt{2^{4/3}3^{-2/3}-1}} & \approx \quad 1.2811 \end{array} \right. \tag{10.1}$$

Well done!

10.3 Checking Energy

Bob: Now let's see whether the energies come out standardized as well. That should be easy to verify: in the standard units, the total energy should be -1/4. Let's check:

```
|gravity> kali mkplummer3.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 10425161316532150531833103071535950804 5
  E_kin = 0.241 , E_pot =  -0.504 , E_tot = -0.262
|gravity> kali mkplummer3.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 32114771548028494957601546171898744302 4
  E_kin = 0.249 , E_pot =  -0.497 , E_tot = -0.248
|gravity> kali mkplummer3.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 28438604808819711409059433102041153967 2
  E_kin = 0.256 , E_pot =  -0.524 , E_tot = -0.268
```

Alice: Just as it should be, within the expected statistical errors of a few percent. Great! Now we can generate a truly standard realization of Plummer's model.

10.4 Quiet Start

Bob: There are a couple other features I'd like to add to the code: a quiet start, and center-of-mass adjustment. Let's start with the first one.

In my original code, each star is given a position in the star cluster independently of all other stars. If you make a small star cluster, it is possible that you wind up with a significant excess of stars, in the core, say, or equally likely a significant lack of stars, there or somewhere else.

I have found it helpful in some of my simulations to start in a more quiet way, in which the stars are initially layered, each occupying a position somewhere in their proper mass shell. In other words, you divide the N-body system as an onion into N different concentric shells, centered on the center of the cluster, and each having the same amount of mass. You then sprinkle one star into each different shell.

Alice: I'm not sure whether that really helps. After you let the system evolve for a while, you will soon develop fluctuations that obey Poissonian statistics. You won't keep your neat layering for long. As a matter of fact, within a small fraction of the crossing time, your particles will visit shells other than the ones they were born in.

Bob: That is true, but even so, I like to start with a more orderly bunch of stars. For one thing, the quartiles will come out better initially.

Alice: I'm still not convinced that it will help, but it won't hurt either. Go right ahead!

Bob: I'll copy the previous version in a new file `mkplummer4.rb` Now I have to be careful. Given me a minute. . . . . Ah, this should do it. Here is the new version of the `mkplummer` method:

```
def mkplummer(n, seed)
  if seed == 0
    srand
  else
    srand seed
  end
  scalefactor = 16.0 / (3.0 * PI)
  nb = NBody.new(n)
  cumulative_mass_min = 0
  cumulative_mass_max = 1.0/n
  nb.body.each do |b|
    b.mass = 1.0/n
    cumulative_mass = frand(cumulative_mass_min, cumulative_mass_max)
    cumulative_mass_min = cumulative_mass_max
    cumulative_mass_max += 1.0/n
    radius = 1.0 / sqrt( cumulative_mass ** (-2.0/3.0) - 1.0)
```

```
    b.pos = spherical(radius) / scalefactor
    x = 0.0
    y = 0.1
    while y > x*x*(1.0-x*x)**3.5
      x = frand(0,1)
      y = frand(0,0.1)
    end
    velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
    b.vel = spherical(velocity) * sqrt(scalefactor)
  end
  STDERR.print "             actual seed used\t: ", srand, "\n"
  nb.acs_write
end
```

10.5 Quiet Indeed

Alice: Can you point out what you changed?

Bob: There are only two places where I added something. Before entering the `nb.body.each` loop, I define the inner mass shell – in fact, an inner mass sphere; the central layer is a sphere, and all the subsequent ones are shells – as follows:

```
cumulative_mass_min = 0
cumulative_mass_max = 1.0/n
```

Then, at the beginning of the loop, I pick a random value for the cumulative mass within the constraint that the value for the cumulative mass has to lie within the range present within the current mass shell:

```
    cumulative_mass = frand(cumulative_mass_min, cumulative_mass_max)
```

As soon as I have done that, I shift the boundaries of the mass shell up by one shell, to make them ready for the next traversal of the loop:

```
    cumulative_mass_min = cumulative_mass_max
    cumulative_mass_max += 1.0/n
```

The last difference is that the radius is now determined from the value for the cumulative mass that I had just found, rather than from using the random number generator here directly:

```
    radius = 1.0 / sqrt( cumulative_mass ** (-2.0/3.0) - 1.0)
```

Alice: How about testing the new version?

Bob: The total energy should remain unchanged, still -1/4. Here it is:

```
|gravity> kali mkplummer4.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 165382740514104340268431945502952096955
  E_kin = 0.253 , E_pot =  -0.5 , E_tot = -0.247
```

Alice: Fair enough. And your quartiles should come out wonderfully accurate, even for, say, 100 particles, by construction.

Bob: I hope so! Let's try:

```
|gravity> kali mkplummer4.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 148758676395007771729895383666632406243
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4693
  r(1/2) = 0.7587
  r(3/4) = 1.258
|gravity> kali mkplummer4.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 3161414165343725478458236113102536048 27
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4752
  r(1/2) = 0.759
```

```
  r(3/4) = 1.275
|gravity> kali mkplummer4.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 146273320387826226617393042233913220311
The values of the three quartiles for r(M) are:
  r(1/4) = 0.469
  r(1/2) = 0.7556
  r(3/4) = 1.253
```

Indeed, as expected. And of course it will be even more accurate for 10,000 particles:

```
|gravity> kali mkplummer4.rb -n 10000 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 10000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 19989003003489936288830478433000250349 2
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4777
  r(1/2) = 0.7685
  r(3/4) = 1.281
```

Alice: Quiet indeed. You have managed to shove the statistical noise under the rug.

Chapter 11

Centering

11.1 Center of Mass Adjustment

Bob: I could see that you were not too impressed with my quiet start. However, I expect that you may be more interested in a more important improvement I would like to make, the center of mass adjustment.

Alice: Indeed. I think it would be better to shift to a coordinate system in which the newly created star cluster has its center of mass in the origin of the coordinate system. In addition, it would be nice to give the coordinate system a boost in such a way that the velocity of the center of mass is zero in that coordinate system. Is this what you had in mind?

Bob: Exactly. It would of course be possible to sprinkle particles in space, and in velocity space, in pairs, so that you would cancel the contributions: you could place them at opposite sides of the center, and give them opposite velocities. But that would create artificial correlations, and I don't like to do that. Better to create a realization first, and then to shift the coordinate system in the way you suggested.

After creating our model, we measure the center of mass position $\mathbf{r}_{\rm com}$, which I will name `pos_com` in the code, as follows:

$$\mathbf{r}_{\rm com} = \frac{\sum_{i=0}^{N-1} m_i \mathbf{r}_i}{\sum_{i=0}^{N-1} m_i} = \frac{1}{M} \sum_{i=0}^{N-1} m_i \mathbf{r}_i$$

and similarly for the velocity of the center of mass, which I will call `vel_com` in the code:

$$\mathbf{v}_{\rm com} = \frac{\sum_{i=0}^{N-1} m_i \mathbf{v}_i}{\sum_{i=0}^{N-1} m_i} = \frac{1}{M} \sum_{i=0}^{N-1} m_i \mathbf{v}_i$$

If we then subtract $\mathbf{r}_{\rm com}$ from each particle's position, and also subtract $\mathbf{v}_{\rm com}$ from each particle's velocity, we will be guaranteed that $\mathbf{r}_{\rm com} = \mathbf{v}_{\rm com} = 0$. This is then the shift that we ordered.

11.2 Implementation

Here is a straightforward implementation, in file `mkplummer5.rb`. At the end of the `mkplummer`, after all the work is done, I am adding a line:

```
nb.adjust_center_of_mass
```

which invokes the following method:

```
def adjust_center_of_mass
  vel_com = pos_com = @body[0].pos*0     # null vectors of the correct length
  @body.each do |b|
    pos_com += b.pos*b.mass
    vel_com += b.vel*b.mass
  end
  @body.each do |b|
    b.pos -= pos_com
    b.vel -= vel_com
  end
end
```

Alice: Straightforward indeed. Normally you would have to divide the positions and the velocities by the total mass, but here the total mass is unity, so you can skip that. Okay, that looks good, but as always, let's first do a couple checks.

Bob: Never hurts. Here they are. First the energy:

```
|gravity> kali mkplummer5.rb -n 1000 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1000
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 22213636604931115936986984484948008407 0
  E_kin = 0.247 , E_pot =  -0.499 , E_tot = -0.252
```

That certainly looks fine. Now the quartiles:

```
|gravity> kali mkplummer5.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 191113185073940108092009029233757184345
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4956
  r(1/2) = 0.7521
  r(3/4) = 1.269
```

11.3 A Bit Disquieting

Hmm, a bit less quiet than before, it seems. Let me try a few more:

```
|gravity> kali mkplummer5.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 259054883668678010594557371152873789580
The values of the three quartiles for r(M) are:
  r(1/4) = 0.5143
  r(1/2) = 0.7532
  r(3/4) = 1.326
|gravity> kali mkplummer5.rb -n 100 | kali quartiles.rb
==> Plummer's Model Builder <==
Number of particles: N = 100
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
```

```
Incremental indentation: add_indent = 2
             actual seed used: 291077572111311493350897272444411366849
The values of the three quartiles for r(M) are:
  r(1/4) = 0.4574
  r(1/2) = 0.7835
  r(3/4) = 1.29
```

Definitely less quiet than before. How can shifting . . . ah, shifting the center of mass also shifts the positions of my idealized mass shells which provided the scaffolding for sprinkling particles in such a nicely layered way. Of course!

Alice: Yes, that must be the reason. Well, that's the price you have to pay for preventing your model for being off-center!

Bob: Perhaps layering was not such a hot idea after all. Oh, well. I may as well leave it in, for now.

11.4 Checking the One-Body Problem

Alice: So far, so good, but we should check that the center of mass is indeed in the center, and will stay there.

Bob: That may not be so easy to check, unless we write a new analysis tool to report the center of mass position and motion.

Alice: And that tool would reflect the same equations you just entered in the code, making it less of an independent check.

Bob: Ah, wait a minute: we can look at a few-body system. Starting with one body, it should sit happily in the center, and two bodies should now be placed opposite each other, in position as well as in velocity.

Alice: Yes, of course. That's a good way to check. Better first run those cases with the version you created when you went to standard units, and then to repeat them for your shifted version.

Bob: Okay, here is a one-body system without shifting:

```
|gravity> kali mkplummer3.rb -n 1
==> Plummer's Model Builder <==
Number of particles: N = 1
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 193832903954810924263329174508991544727
```

```
ACS
  NBody
    Array body
      Body body[0]
        Float mass
             1.0000000000000000e+00
        Vector pos
            -1.8567992923753585e+00  -9.2594814522491742e-01   5.0303486786525831e-01
        Vector vel
             2.2659631503271319e-01  -7.3258520882712438e-02   5.3950878591061713e-01
SCA
```

---

and here with the proper center of mass shifts:

```
|gravity> kali mkplummer5.rb -n 1
==> Plummer's Model Builder <==
Number of particles: N = 1
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 208700341407249682305261004678131538430
ACS
  NBody
    Array body
      Body body[0]
        Float mass
             1.0000000000000000e+00
        Vector pos
             0.0000000000000000e+00   0.0000000000000000e+00   0.0000000000000000e+00
        Vector vel
             0.0000000000000000e+00   0.0000000000000000e+00   0.0000000000000000e+00
SCA
```

Alice: Proper indeed.

11.5 Checking the Two-Body Problem

Bob: And here for the two-body system, unshifted:

```
|gravity> kali mkplummer3.rb -n 2
==> Plummer's Model Builder <==
Number of particles: N = 2
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 345807909785296468290666715994337205 89
ACS
  NBody
    Array body
      Body body[0]
        Float mass
             5.0000000000000000e-01
        Vector pos
             3.1652779708472756e-01  -1.6122213508487826e-01  -6.9169937689279692e
        Vector vel
             5.5200426553264202e-02   3.0624479376193575e-01   1.3253863487752043e
      Body body[1]
        Float mass
             5.0000000000000000e-01
        Vector pos
            -2.5359871597587352e-01  -4.5332999684125568e-02  -5.7089852844379374e
        Vector vel
            -1.0054280731866456e+00  -9.9576524816940327e-03  -4.2666524232859754e
SCA
```

and shifted:

```
|gravity> kali mkplummer5.rb -n 2
==> Plummer's Model Builder <==
Number of particles: N = 2
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 159870946621639406074815189578997451138
ACS
  NBody
    Array body
      Body body[0]
        Float mass
             5.0000000000000000e-01
```

```
    Vector pos
         8.7192183081543928e-01   8.0364762542453172e-01   2.1438145962418126e-01
    Vector vel
        -7.8691888328259768e-02  -1.5125295684987217e-01  -5.9572661089055678e-03
  Body body[1]
    Float mass
         5.0000000000000000e-01
    Vector pos
        -8.7192183081543906e-01  -8.0364762542453172e-01  -2.1438145962418126e-01
    Vector vel
         7.8691888328259768e-02   1.5125295684987222e-01   5.9572661089055748e-03
SCA
```

Alice: Good! I believe the code now. We have acquired a well-adjusted codes that speaks in standard units.

Chapter 12

Scaling

12.1 Units Adjustment

Bob: We now have a code that produces particle positions and velocities, drawn from a Plummer distribution function represented in proper standard units of length, time, and mass, and properly centered at the center of mass, in position as well as velocity. What more could we possibly want? I think we can call it a day.

Alice: Before doing so, there is just one thing that is still bothering me. Even though our Plummer realizations are now perfectly centered, their units are not quite right.

Bob: But I thought we had checked that? Starting with `mkplummer3.rb` we have made sure to use standard units.

Alice: Well, let's see what happens for really low N values:

```
|gravity> kali mkplummer5.rb -n 3 -s 1 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 1
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 1
  E_kin = 0.0661 , E_pot =  -0.319 , E_tot = -0.252
|gravity> kali mkplummer5.rb -n 3 -s 2 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
```

```
pseudorandom number seed given: 2
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 2
  E_kin = 0.121 , E_pot =  -0.273 , E_tot = -0.152
|gravity> kali mkplummer5.rb -n 3 -s 3 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 3
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 3
  E_kin = 0.204 , E_pot =  -0.386 , E_tot = -0.182
|gravity> kali mkplummer5.rb -n 3 -s 4 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 4
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 4
  E_kin = 0.124 , E_pot =  -0.184 , E_tot = -0.0595
```

Bob: Hmm, that doesn't look like the desired total energy value of minus one quarter. And I certainly don't like the total energy to become positive! That means that some of our realizations are actually unbound!

Alice: The problem is that, even though our underlying distribution function has been scaled correctly, any small-number realization will introduce fluctuations in the actual numbers that are picked out.

Bob: Ah, of course, this is just what happened with the center of mass. Even though an ensemble of many realizations will shown the average center of mass position and velocity values to be almost zero, individual realizations need to be adjusted, as we just did. Similarly, we will have to rescale the positions and velocities, to make any single realization come out with the right energy.

Alice: Effectively, we have to rescale length and time units; length for the positions, and in addition separately time for the velocities.

12.2 Implementation

Bob: Let's create a new file. How about calling it `mkplummer.rb`, without a number attached to it now, to show that this will be our final version, that we can use as a work horse?

Alice: Hope springs eternal. But go ahead, we can always rename it to `mkplummer.rb6`, like in a few minutes.

Bob: No, I think this will be really it. Well, we will see.

Alice: We have to do two separate things. Just getting the total energy to be -1/4 is not good enough: we want to make sure that we have an accurate virial equilibrium, with a kinetic energy of +1/4 and a potential energy of -1/2. This is the reason that we have to scale length and time units separately.

Bob: In analogy with our way to adjust the center of mass, we can now adjust the units. If I understand correctly what you just said, it should be this, right?

```
def adjust_units
  alpha = -epot / 0.5
  beta = ekin / 0.25
  @body.each do |b|
    b.pos *= alpha
    b.vel /= sqrt(beta)
  end
end
```

Alice: That indeed looks correct. And you invoke that function *after* you invoke the center-of-mass adjustment.

Bob: Ah, yes, the order is important. Not for the potential energy, since that only depends on relative distances. But the kinetic energy depends on the square of the velocity differences between each particle and the center of mass. Good point.

You know, let me modularize the `mkplummer` method a bit further. I'm sure you would love that!

Alice: Any move to more modularity is welcome, within reason, but something tells me you won't unreasonably overshoot toward modularity.

Bob: You bet I won't. But with all the adjustments we are now making, it would seem more clear to isolate the actual sampling procedure in a separate methods `plummer_sample`, and to leave the rest of the administrative details to the higher-level function `mkplummer`, which can deal with picking the right seed, invoking the sampling function, pushing the sampling results on the stack, and doing all the final adjustments.

Alice: You're becoming a true modularity spokesman!

Bob: I'll ignore that. This is just common sense.

Alice: Let's hope it becomes more common.

12.3 Treating Even the Vacuum

Bob: Here is the top level method:

```
def mkplummer(c)
  if c.seed == 0
    srand
  else
    srand c.seed
  end
  nb = NBody.new
  c.n.times do |i|
    b = plummer_sample
    b.mass = 1.0/c.n
    b.body_id = i
    nb.body.push(b)
  end
  nb.adjust_center_of_mass if c.n > 0
  nb.adjust_units if c.n > 1
  nb.acs_log(1, "             actual seed used\t: #{srand}\n")
  nb.acs_write($stdout, false, c.precision, c.add_indent)
end
```

and here is where the actual sampling is done, the place where a single particle receives its initial position and velocity values, before they later will be adjusted:

```
def plummer_sample
  b = Body.new
  scalefactor = 16.0 / (3.0 * PI)
  radius = 1.0 / sqrt( rand ** (-2.0/3.0) - 1.0)
  b.pos = spherical(radius) / scalefactor
  x = 0.0
  y = 0.1
  while y > x*x*(1.0-x*x)**3.5
    x = frand(0,1)
    y = frand(0,0.1)
  end
  velocity = x * sqrt(2.0) * ( 1.0 + radius*radius)**(-0.25)
  b.vel = spherical(velocity) * sqrt(scalefactor)
  b
end
```

Alice: Ah, I see that you only invoke the `adjust_units` method if you have two or more particles. Of course, if you have only a single particle, then its velocity, after adjusting it to the center of mass frame, will be zero.

Bob: Yes, I didn't want to risk dividing by zero, and in any case, there is no need to scale anything for a one-particle system already at rest.

Alice: But what is the meaning of the `if` statement for the center-of-mass adjustment?

Bob: Oh, I thought I might as well make it work for any reasonable value, including the vacuum, with zero particles. Without that clause, for `c.n = 0` the method `adjust_center_of_mass` would try to set the length of the position and velocity vectors of the center of mass to the same length as that of the position vector of the first particle, which would be non-existent, and so you would get an error message – even though the rest of the body of `adjust_center_of_mass` would work fine for zero particles.

Alice: I like building in such careful statements. Who knows, at some point we may decide to build slews of N-body models with different particle numbers, and it is nice to have them all behave well, for any reasonable and even not so reasonable numbers, including 0. However, what will happen for negative numbers?

Bob: I haven't tested that. But when I look at `mkplummer` above, it would seem that the do loop never gets executed, so effectively if will still give a zero-particle system, without running into any more serious error.

Alice: That may or may not be what we want. In any case, we can discuss more careful exception handling some other time.

12.4 Bells and Whistles

Bob: I agree. What next. Ah, let me take out this quiet start business. It was an interesting idea, but as we noticed above, with center of mass adjustment it will get partly screwed up anyway. Actually, I just realized a much more important reason *not* to do a radial layering of particles. Pretty soon we may want to give some particles extra properties. We could introduce a mass spectrum, or promordial binaries, or what not. If we make sure that the original distribution of particles is truly random, without any layering bias, we are less likely to wind up with unrealistic distributions . . .

Alice: . . . such as having the lighter particles all in the center and the heavier ones outside. Yes, I see what you mean. Let's keep it simple and forget about being too quiet.

Bob: Finally, we can use some of the bells and whistles that we have added

before, when we wrote the ACS I/O routines. There we allowed the user to specify the number of digits accuracy and indentation prefered. If you're dealing with only a thousand particles, you may not want double precision, if that will cut down your inital file length by a factor half. Let us test the options we have provided:

```
|gravity> kali mkplummer.rb -h
Plummer's Model Builder
-n  --n_particles: Number of particles [default: 1]
-s  --seed: pseudorandom number seed given [default: 0]
--verbosity: Screen Output Verbosity Level [default: 1]
--acs_verbosity: ACS Output Verbosity Level [default: 1]
--precision: Floating point precision [default: 16]
--indentation: Incremental indentation [default: 2]
-h  --help: Help facility
---help: Program description (the header part of --help)
|gravity> kali mkplummer.rb -n 3 --precision 7 --indentation 4
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 7
Incremental indentation: add_indent = 4
             actual seed used: 258014049084773028176397665329844112845
ACS
    NBody
        Array body
            Body body[0]
                Fixnum body_id
                    0
                Float mass
                       3.3333333e-01
                Vector pos
                      -2.5644760e-01   1.3257792e-01   4.5680920e-01
                Vector vel
                       5.4383604e-01   8.9160370e-02  -1.6318676e-01
            Body body[1]
                Fixnum body_id
                    1
                Float mass
                       3.3333333e-01
                Vector pos
                      -8.2653137e-02   5.5067278e-02  -1.3863486e-01
                Vector vel
```

```
                    -2.0330925e-01   6.6077956e-01   8.4353819e-02
            Body body[2]
                Fixnum body_id
                    2
                Float mass
                       3.3333333e-01
                Vector pos
                       3.3910074e-01  -1.8764520e-01  -3.1817434e-01
                Vector vel
                      -3.4052678e-01  -7.4993993e-01   7.8832938e-02
        String story 4
                         actual seed used: 258014049084773028176397665329844112845

SCA
```

Alice: Fine to have the extra freedom, but I doubt we'll ever use it; as soon as you start doing a run, you'll probably want to keep the output to full precision.

Bob: In any case, at least for display purposes it all fits within the good old VT100 80-column wide screen.

Alice: Which actually came from the 80-column punched card format. But you're too young to remember that.

Bob: Next time we go to a museum you can point out to me what the world was like when you grew up.

12.5 Checking the Output

Alice: Let's check again, starting with the zero-body problem, while working our way up. And we might as well use your precision cap to keep it punched card printable:

```
|gravity> kali mkplummer.rb -n 0 --precision 10
==> Plummer's Model Builder <==
Number of particles: N = 0
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 10
Incremental indentation: add_indent = 2
             actual seed used: 207499680399256420416434074598264822880
ACS
  NBody
    Array body
```

```
    String story 2
                  actual seed used: 207499680399256420416434074598264822880

SCA
|gravity> kali mkplummer.rb -n 1 --precision 10
==> Plummer's Model Builder <==
Number of particles: N = 1
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 10
Incremental indentation: add_indent = 2
            actual seed used: 312865614098536403818900489640401762769
ACS
  NBody
    Array body
      Body body[0]
        Fixnum body_id
          0
        Float mass
             1.0000000000e+00
        Vector pos
             0.0000000000e+00   0.0000000000e+00   0.0000000000e+00
        Vector vel
             0.0000000000e+00   0.0000000000e+00   0.0000000000e+00
    String story 2
                  actual seed used: 312865614098536403818900489640401762769

SCA
|gravity> kali mkplummer.rb -n 2 --precision 10
==> Plummer's Model Builder <==
Number of particles: N = 2
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 10
Incremental indentation: add_indent = 2
            actual seed used: 27501119533328509525832556146120520876 3
ACS
  NBody
    Array body
      Body body[0]
        Fixnum body_id
          0
        Float mass
             5.0000000000e-01
```

```
        Vector pos
             7.4648885924e-04  -2.4989123038e-01   7.3359209282e-03
        Vector vel
            -1.4548783709e-01   6.3957148088e-01   2.6416209060e-01
      Body body[1]
        Fixnum body_id
          1
        Float mass
             5.0000000000e-01
        Vector pos
            -7.4648885924e-04   2.4989123038e-01  -7.3359209282e-03
        Vector vel
             1.4548783709e-01  -6.3957148088e-01  -2.6416209060e-01
    String story 2
                  actual seed used: 27501119533328509525832556146120520876 3

SCA
```

12.6 Checking the Energy

That all works as it should. Now an energy check:

```
|gravity> kali mkplummer.rb -n 0 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 0
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 23529713768571183817147144222057497866 0
  E_kin = 0 , E_pot =  0 , E_tot = 0
|gravity> kali mkplummer.rb -n 1 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 1
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 25797359119924397470559126176136662900 0
  E_kin = 0 , E_pot =  0 , E_tot = 0
|gravity> kali mkplummer.rb -n 2 | kali energy.rb
```

```
==> Plummer's Model Builder <==
Number of particles: N = 2
pseudorandom number seed given: 0
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 171592391979247212523424164411050984728
  E_kin = 0.25 , E_pot =  -0.5 , E_tot = -0.25
```

Looks good so far. Even the zero-body behaves! A single particle at rest in the center has neither potential nor kinetic energy. And the two-particle realization is nicely scaled to virial equilibrium.

Bob: Let's try the same four realizations you made earlier, with the very same seeds:

```
|gravity> kali mkplummer.rb -n 3 -s 1 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 1
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 1
  E_kin = 0.25 , E_pot =  -0.5 , E_tot = -0.25
|gravity> kali mkplummer.rb -n 3 -s 2 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 2
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 2
  E_kin = 0.25 , E_pot =  -0.5 , E_tot = -0.25
|gravity> kali mkplummer.rb -n 3 -s 3 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 3
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
```

```
             actual seed used: 3
  E_kin = 0.25 , E_pot =  -0.5 , E_tot = -0.25
|gravity> kali mkplummer.rb -n 3 -s 4 | kali energy.rb
==> Plummer's Model Builder <==
Number of particles: N = 3
pseudorandom number seed given: 4
Screen Output Verbosity Level: verbosity = 1
ACS Output Verbosity Level: acs_verbosity = 1
Floating point precision: precision = 16
Incremental indentation: add_indent = 2
             actual seed used: 4
  E_kin = 0.25 , E_pot =  -0.5 , E_tot = -0.25
```

Alice: Good! Believe it or not, we may not need to call this version `mkplummer6.rb`, after all. It may remain `mkplummer.rb` and become our standard initial conditions generator.

Bob: I told you so!

Chapter 13

Literature References

A comparison of Numerical Methods for the Study of Star Cluster Dynamics, by Sverre Aarseth, Michel Henon, and Roland Wielen, 1974, Astron. Astroph. **37**, 183.

Galactice Dynamics, by James Binney and Scott Tremaine, 1987 [Princeton University Press].

The Gravitational Million-Body Problem, by Douglas Heggie and Piet Hut, 2003 [Cambridge University Press].

Standardised Units and Time Scales, by Douglas Heggie and Robert Mathieu, 1986, in *The Use of Supercomputers in Stellar Dynamics*, edited by Steve McMillan and Piet Hut [Springer], p. 233.