Mon. Not. R. Astron. Soc. **000**, 000–000 (0000) Printed 25 November 2015 (MN LATEX style file v2.2)

Torus mapper: a code for dynamical models of galaxies

James Binney[1]⋆ and Paul J. McMillan[1,2]
[1] *Rudolf Peierls Centre for Theoretical Physics, 1 Keble Road, Oxford, OX1 3NP, UK*
[2] *Lund Observatory, Lund University, Department of Astronomy and Theoretical Physics, Box 43, SE-22100, Lund, Sweden*

25 November 2015

ABSTRACT
We present a freely downloadable software package for modelling the dynamics of galaxies, which we call the Torus Mapper (TM). The package is based around 'torus mapping', which is a non-perturbative technique for creating orbital tori for specified values of the action integrals. Given an orbital torus and a star's position at a reference time, one can compute its position at any other time, no matter how remote. One can also compute the velocities with which the star will pass through any given point and the contribution it will make to the time-averaged density there. A system of angle-action coordinates for the given potential can be created by foliating phase space with orbital tori. Such a foliation is facilitated by the ability of TM to create tori by interpolating on a grid of tori.

We summarise the advantages of using TM rather than a standard time-stepper to create orbits, and give segments of code that illustrate applications of TM in several contexts, including setting up initial conditions for an N-body simulation. We examine the precision of the orbital tori created by TM and the behaviour of the code when orbits become trapped by a resonance.

Key words: Galaxy: kinematics and dynamics – galaxies: kinematics and dynamics – methods: numerical

1 INTRODUCTION

Dynamical models are playing an increasingly important role in the interpretation of observations. Moreover, the volume and quality of available observational data for both external galaxies and our own Galaxy are growing explosively through the increase in the scale and sensitivity of large-format CCD detectors and integral field spectrographs, and, in the case of our own Galaxy, the taking of astrometric data by the Gaia satellite.

N-body models have played a large role in the development of our understanding of galaxy formation and dynamics, but they have important limitations. 1) They are significantly degraded by discreteness noise, which is particularly acute in the case of our Galaxy because the data are strongly focused on the immediate vicinity of one star, the Sun. 2) Tailoring an N-body model to a specific body of observations is difficult given the complex and ill-understood nature of the connection between the initial conditions that define an N-body simulation and the equilibrium model that emerges. It is true that the made-to-measure technique (Syer & Tremaine 1996; Dehnen 2009; Morganti & Gerhard 2012) can be used to refine the fit between a model's predictions and data, but making such adjustments increases the level of discreteness noise in the model, so the starting point for adjustments should be reasonably close to the truth. 3) There is no compact and transparent representation of an N-body model: the model is specified by $6N$ phase-space coordinates, but if the equations of motion are integrated for a fraction of a dynamical time, all these numbers change while the model remains the same. This degeneracy of representation makes comparison of models hard. It also obscures the physical significance of a given model and therefore restricts the harvest of scientific understanding that can be drawn from a given set of observations. 4) N-body models with acceptable levels of discreteness noise are computationally costly, so it is unlikely to be feasible to find an N-body model that provides an excellent fit to a sophisticated set of observational data.

N-body models do have one important advantage, namely, that they can be end-points of simulations of cosmological clustering. This advantage is, however, strongly tempered by the fact that it is currently not feasible to follow the dynamics of baryons in such simulations using only basic physics. Consequently, resort is made to "subgrid recipes" that should reproduce, in a statistical sense, the impact on simulated scales of physical processes that take place on smaller scales. It is not even known whether the

⋆ E-mail: binney@thphys.ox.ac.uk

neglected physics *can* be reproduced by such recipes, still less is it clear if any of the recipes currently used is reasonably accurate. Consequently, cosmological simulations of galaxy formation lack rigour and should be considered instructive ways of producing N-body galaxy models, rather than definitive exercises.

Decades ago, when the data for galaxies were orders-of-magnitude more sparse than they now are, data were widely modelled with the Jeans equations (e.g. Bacon et al. 1983; Binney et al. 1990; van der Marel et al. 1990). Some studies still rely on the Jeans equations, but the technique has strong limitations. 1) It recovers only the low-order velocity moments, which do not specify the actual (non-Gaussian) velocity distributions. 2) In practice Jeans modelling is restricted to systems with distribution functions (DFs) that depend on only energy E and angular momentum L, whereas we know that real DFs typically depend on three isolating integrals.

As data for external galaxies grew richer, the dominant technique for modelling galaxies became that introduced by Schwarzschild (1979) and extended by van der Marel et al. (1998), Gebhardt et al. (2003) and others, in which orbits in a given gravitational potential are assigned weights in such a way that the fit of a model to observational data is optimised. This technique is flexible but suffers from many of the shortcomings of N-body models, in particular a high degree of discreteness noise and the lack of compact and transparent model specification. The keys to obtaining such a specification are (a) to assign physically meaningful labels to orbits, and (b) to establish the density with which phase space is sampled by the orbit library so the weights orbits are assigned can be converted into the value taken by the DF on each orbit. These tasks can be accomplished in the context of a classical Schwarzschild model (e.g. Häfner et al. 2000), but there are distinct advantages to abandoning the use of orbits in favour of phase-space tori, using the technique described in this paper.

1.1 Orbital tori and action-angle coordinates

Numerical integration of orbits in spherical and axisymmetric potentials reveals that they are nearly always quasiperiodic. That is, the Fourier decomposition of the time series $x_i(t)$ of any coordinate that one obtains by integrating the equations of motion contains only discrete spectral lines, and the frequencies of these lines can be represented as integer linear combinations of three *fundamental frequencies* Ω_i. From this finding it follows that these orbits are *integrable* in the sense that along them three independent functions $I_i(\mathbf{x},\mathbf{v})$ of the phase-space coordinates are constant (Arnold 1989). The Jeans theorem (Jeans 1915) tells us that the DF of a steady-state galaxy with the given potential can be taken to be a function $f(I_1,I_2,I_3)$ of these constants of motion.

Since any function $J(I_1,I_2,I_3)$ inherits from the I_i the property of being a constant of motion, there is considerable freedom in what functions of $(\mathbf{x},\mathbf{v})$ one uses for the arguments of the DF. However, if one requires that the adopted functions J_i are capable of being complemented by canonically conjugate coordinates θ_i, so together $(\boldsymbol{\theta},\mathbf{J})$ form a system of canonical coordinates, then almost all the freedom in the choice of constants of motion disappears. If one further specifies that J_r quantifies the amplitude of a star's oscillations in radius and J_z quantifies the amplitude of a star's oscillations in the latitudinal direction (along ϑ in conventional spherical polar coordinates), then the last vestiges of freedom disappear and one has a uniquely defined set of canonical coordinates, with the momenta labelling orbits and the conjugate variables specifying position within the orbit.[1]

It follows trivially from Hamilton's equations of motion, that angle coordinates increase linearly in time: $\theta_i(t) = \theta_i(0) + \Omega_i t$. Hence stellar dynamics becomes trivial once the mapping $(\boldsymbol{\theta},\mathbf{J}) \to (\mathbf{x},\mathbf{v})$ has been constructed.

The prerequisite of angle-action coordinates $(\boldsymbol{\theta},\mathbf{J})$ playing a useful role in astronomy is algorithms that alow one to compute $\mathbf{x}(\boldsymbol{\theta},\mathbf{J})$ and $\mathbf{v}(\boldsymbol{\theta},\mathbf{J})$, and the inverse functions, $\mathbf{J}(\mathbf{x},\mathbf{v})$ etc. A fully analytic algorithm of this type is known only for the multi-dimensional harmonic oscillator and the isochrone potential (Henon 1959; Binney & Tremaine 2008) (which includes the Kepler potential as a limiting case). McGill & Binney (1990) showed how these fully analytic algorithms can be leveraged into an effective numerical scheme for a general axisymmetric potential by *torus mapping.* The idea is that the analytic formulae provide an explicit representation of a phase-space surface $\mathbf{J}$ = constant. This three-dimensional surface in six-dimensional phase space is topologically a 3-torus (Arnold 1989), and this 3-torus is special in the sense that any two-surface within it has vanishing Poincaré invariant $\sum_i \mathrm{d}q_i\mathrm{d}p_i$. Consequently, the orbital surface $\mathbf{J}$ = constant is called a *null torus.* If such a null torus can be mapped into the target phase space such that on it the target Hamiltonian

$$H(\mathbf{x},\mathbf{v}) = \tfrac{1}{2}v^2 + \Phi(\mathbf{x}) \tag{1}$$

is constant, then the image torus becomes an orbital torus of the target potential $\Phi(\mathbf{x})$.

McGill & Binney (1990) mapped *toy tori* labelled by $\mathbf{J}^{\mathrm{T}}$ and furnished with angle coordinates $\boldsymbol{\theta}^{\mathrm{T}}$, into tori of a target potential labelled by $\mathbf{J}$ with canonical transformations that have generating functions of the form

$$S(\mathbf{J},\boldsymbol{\theta}^{\mathrm{T}}) = \mathbf{J}\cdot\boldsymbol{\theta}^{\mathrm{T}} + \sum s_{\mathbf{n}}(\mathbf{J})\mathrm{e}^{\mathrm{i}\mathbf{n}\cdot\boldsymbol{\theta}^{\mathrm{T}}}. \tag{2}$$

They showed that the image tori became excellent approximations to orbital tori once the parameters $S_{\mathbf{n}}$ had been adjusted to yield a small RMS variation δH in the Hamiltonian over the image torus. Kaasalainen (1994) showed that certain target tori cannot be reached using generating functions of this form, and for these tori it is necessary to compound such a transformation with a point transformation. The primary purpose of this paper is to introduce a freely downloadable code, TM, that computes these canonical transformations for an arbitrary axisymmetric potential, and provides a wide range of routines for exploiting the resulting tori.

[1] J_r is sometimes denoted J_R or J_u. Similarly, J_z is identical to what in Binney & Tremaine (2008) is denoted J_θ. These alternative notations arise through solving the Hamilton-Jacobi equation in different coordinate systems: unique underlying actions J_r, J_z are being approached through differing limiting forms.

1.2 Torus-mapping code TM

Users of TM do not need to engage with the process of computing canonical transformations. The only code they need to write is that which computes the potential and its first derivatives in R and z – thus precisely the code required to compute an orbit by numerical integration of the equations of motion. Given this code and values for the three actions J_r, J_ϕ and J_z, TM provides functions that return (i) $(\mathbf{x}, \mathbf{v})$ for specified $\boldsymbol{\theta}$, (ii) the Jacobian $\partial(\mathbf{x})/\partial(\boldsymbol{\theta})$, and (iii) $\boldsymbol{\theta}$ given $(\mathbf{x}, \mathbf{J})$. In Section 2.4 we will see that the last two functions allow TM to perform the role in Schwarzschild modelling that has previously been played by an integrator of the Runge-Kutta, Bulirsch-Stoer, or leap-frog type. Displacing an integrator with TM brings substantial advantages.

1 A torus is specified by ~ 100 numbers, namely the $S_\mathbf{n}$, their derivatives $\partial S_\mathbf{n}/\partial \mathbf{J}$, and a handful of other parameters, rather than by some thousands of phase-space coordinates along a time sequence, so using TM dramatically shrinks the size of an orbit library at a given resolution.

2 It is far easier to construct an orbit library that provides appropriate phase-space coverage by systematically marching through action space than by designing a grid of initial conditions $(\mathbf{x}, \mathbf{v})$ for orbit integrations. In particular, two very different initial conditions will specify the same orbit if an orbit passes through both phase-space points. Consequently, when a grid of initial conditions is used to generate an orbit library, it is not straightforward to ensure that essentially identical orbits are not obtained at different points of the grid. When the actions are specified, there is no danger of such double-computation.

3 When TM is used, every orbit has a unique label that allows it to be compared with orbits computed by other investigators, possibly in a slightly different potential.

4 On account of this last point, it is trivial to establish, indeed predetermine, the density with which the orbit library samples phase space. This knowledge makes it possible to infer the value taken by the DF on an orbit from the orbit's weight in the Schwarzschild model.

5 When TM is used, it is trivial to find with what velocities an orbit will pass through a given point, whereas when an integrator is used, the whole time sequence must be searched for points at which the orbit comes near to the specified point, for in finite time it is unlikely to reach that point exactly. This capability makes it much easier to compute velocity distributions when TM is used in place of an integrator.

The only disadvantage of using TM rather than an orbit integrator is that TM cannot accurately represent resonantly trapped or chaotic orbits. We briefly address this issue in Section 6, but a full discussion of the response of TM to resonant trapping lies beyond the scope of this paper and will be the topic of a forthcoming paper. Fortunately, in typical axisymmetric potentials, resonant trapping is of limited significance.

1.3 Extensions of torus mapping

In its simplest form torus mapping returns $(\mathbf{x}, \mathbf{v})$ given $(\boldsymbol{\theta}, \mathbf{J})$ or $(\mathbf{v}, \boldsymbol{\theta})$ given $(\mathbf{x}, \mathbf{J})$, but it is not well suited to computing $(\boldsymbol{\theta}, \mathbf{J})$ given $(\mathbf{x}, \mathbf{v})$. Consequently, much recent work has used radically different approaches to the computation of $(\boldsymbol{\theta}, \mathbf{J})$ from $(\mathbf{x}, \mathbf{v})$ (Binney 2010; Binney & McMillan 2011; Binney 2012a,b; Bovy & Rix 2013; McMillan & Binney 2013; Bovy 2014; Sanders & Binney 2013; Piffl et al. 2014, 2015; Binney & Piffl 2015). The methods used in these studies are inferior to torus mapping as regards precision in the sense that they cannot be systematically refined, while by increasing the number of coefficients $S_\mathbf{n}$ used in the specification (2) of the generating function, torus mapping can be systematically refined. Approximate actions often suffice for the computation of the observables of a model that is specified by a DF that is an analytic function of the actions (e.g Binney 2012b; Sanders & Binney 2015), but lack of precision is a serious issue when the value of the DF varies significantly with changes in action comparable to the error in $\mathbf{J}(\mathbf{x}, \mathbf{v})$. The classic example of this phenomenon is the DF of a stellar stream (Bovy 2014; Sanders 2014).

Recently Sanders & Binney (2015) gave an algorithm that allows one to compute $(\boldsymbol{\theta}, \mathbf{J})$ from $(\mathbf{x}, \mathbf{v})$ to high precision: the Stäckel Fudge is first used to estimate $\mathbf{J}(\mathbf{x}, \mathbf{v})$. Then the torus is constructed for $\mathbf{J}$ and the point $(\mathbf{x}_1, \mathbf{v}_1)$ on the torus that is nearest to $(\mathbf{x}, \mathbf{v})$ is located. Then the Stäckel Fudge is used to estimate $\mathbf{J}_1(\mathbf{x}_1, \mathbf{v}_1)$. Under the assumption that the error $\boldsymbol{\Delta}_1 = \mathbf{J}_1 - \mathbf{J}$ in the actions returned by the Stäckel Fudge is a smooth function on phase space, an improved estimate of the actions of $(\mathbf{x}, \mathbf{v})$ is $\mathbf{J} - \boldsymbol{\Delta}_1$. This improved estimate can be refined by constructing the torus $\mathbf{J} - \boldsymbol{\Delta}_1$ and finding the point $(\mathbf{x}_2, \mathbf{v}_2)$ on it that is closest to $(\mathbf{x}, \mathbf{v})$. Then the refined estimate of the actions of $(\mathbf{x}, \mathbf{v})$ is $\mathbf{J} - \boldsymbol{\Delta}_2 = 3\mathbf{J} - \mathbf{J}_1 - \mathbf{J}_2$, where $\mathbf{J}_2$ are the Stäckel-Fudge actions of $(\mathbf{x}_2, \mathbf{v}_2)$ and $\boldsymbol{\Delta}_2 = \mathbf{J}_2 - (\mathbf{J} - \boldsymbol{\Delta}_1)$. Sanders & Binney (2015) showed that in practice $\mathbf{J} - \boldsymbol{\Delta}_1$ is already an excellent estimate of the actions, but that the algorithm can be iterated to convergence in the sense that $(\mathbf{x}, \mathbf{v})$ lies on one of the constructed tori.

In this paper we address an important topic that has been rather neglected in recent papers on torus mapping, namely resonances. The importance of resonances for Hamiltonian mechanics is well known (Chirikov 1979; Lichtenberg & Lieberman 1983). In a generic potential the fundamental frequencies Ω_i are functions of the actions, and since rational numbers are densely distributed on the real line, it often happens that the frequencies are commensurable: that is, a vector $\mathbf{N}$ with integer components exists such that $\mathbf{N} \cdot \boldsymbol{\Omega} = 0$. If the potential has a separable Hamilton-Jacobi equation (as Stäckel potentials do, e.g. Binney & Tremaine 2008), these resonances are accidental in the sense that they leave no mark on the dynamics. But in general resonant orbits for which $\mathbf{N}$ has components of modest size trap neighbouring orbits, so the latter *librate* around the resonant orbit, rather than circulating past it (for an account of resonant trapping, see e.g. Binney 2013). Torus mapping can be used to construct an integrable Hamiltonian that is close to the real Hamiltonian (Kaasalainen & Binney 1994b), so trapping can be studied with high-precision perturbation theory (Kaasalainen 1995). In Section 6 we show how resonant trapping manifests itself with torus mapping, and explain two different approaches to resonant trapping; the approach that is appropriate depends on the astronomical context. In Section 6.1 we speculate that on account of resonant trapping there may be points in the disc at which

Table 1. Coordinate systems. By default the HEQ system is at epoch J2000. s denotes distance from Sun, $\mu_{\alpha*} \equiv \dot\alpha\cos\delta$ and similarly $\mu_{l*} \equiv \dot l\cos b$. In the HCA system the x axis points to the Galactic centre and the y axis points along $l = 90°$. In the GCA system the x axis points towards the Sun. In the GCY system the Sun is at $\phi = 0$ and has $v_\phi < 0$. The default local circular speed is $\Theta_0 = 244.5\,\mathrm{km\,s^{-1}}$ (from the "convenient potential" in McMillan 2011). The Sun's position is assumed to be $(R, z) = (8.5, 0.014)$ kpc, and from Schönrich et al. (2010) the Sun's velocity in the LSR system is $(v_x, v_y, v_z) = (-11.1, -12.24, 7.25)\,\mathrm{km\,s^{-1}}$.

Coordinate system	Point	Symbol
Heliocentric equatorial (J2000)	$(s, \alpha, \delta, v_\parallel, \mu_{\alpha*}, \mu_\delta)$	HEQ
Heliocentric Galactic polar	$(s, l, b, v_\parallel, \mu_{l*}, \mu_b)$	HGP
Heliocentric Cartesian	(x, y, z, v_x, v_y, v_z)	HCA
Local standard of rest	(x, y, z, v_x, v_y, v_z)	LSR
Galactocentric Cartesian	(x, y, z, v_x, v_y, v_z)	GCA
Galactocentric cylindrical	$(R, z, \phi, v_R, v_z, v_\phi)$	GCY

there are more stars moving up through the plane at a given speed v_z than there are moving down at the same speed.

1.4 Layout of the paper

Section 2 is the core of the paper. It aims to demonstrate how easily properties of individual orbits in any axisymmetric potential can be computed with TM, and to explain how distribution functions are used in conjunction with tori, for example to obtain an N-body realisation of a model galaxy. We have kept this section brief by moving as many technical details as possible to four appendices that normal users do not need to study. Section 3 clarifies the meaning of angle variables and presents some tests of the precision achievable with TM. Section 4 shows how to create tori by interpolating on a grid of previously created tori. Section 5 presents a test of an N-body realisation created with the tools included in TM. Section 6 discusses the implications of resonant trapping for torus mapping. Section 7 sums up and looks to the future.

2 CLASSES AND NAMESPACES

2.1 Units

Internally TM takes as units $M_\odot$, kpc and Myr. In this system Newton's constant is $G = 4.99\times10^{-12}$ so for our Galaxy $GM \sim 1$. The unit of velocity, $\mathrm{kpc\,Myr^{-1}} = 978\,\mathrm{km\,s^{-1}}$, so the local circular speed is ~ 0.25 in code units. To specify a speed of $220\,\mathrm{km\,s^{-1}}$ one writes

```
Vc=220 * Units::kms
```

TM works with angles in radians, so to input 10 degrees one writes

```
theta=10 * Units::degree
```

The namespace `Units` (defined in the file Units.h) contains a large range of conversion factors between commonly used units and code units.

2.2 Coordinate systems

A remarkably large number of coordinate systems are useful in Galaxy modelling, because we can consider motion to occur in the meridional (R, z) plane, or in full 3d space, and for the latter we might prefer coordinates centred on Sgr A* or on the Sun, and we might prefer to use Cartesian, cylindrical or spherical coordinates. If we are using heliocentric polar coordinates, we can cover the sky with either right-ascension and declination, or Galactic longitude and latitude. Sometimes we require only spatial coordinates, sometimes velocities, and at other times we require phase-space coordinates. Table 1 lists the coordinate systems available in TM. By defining distinct data types for each of these numerous options, the code limits the scope for error by calling a function with inappropriate data. Table 2 lists the available data types.

Table 2. Data types. Types listed in the lower portion of the table are classes that include methods for adding instances, multiplying components by a constant, etc.

Type	dimension	content
Frequencies	3	$(\Omega_r, \Omega_z, \Omega_\phi)$
Actions	3	(J_r, J_z, J_ϕ)
Angles	3	$(\theta_r, \theta_z, \theta_\phi)$
Position	3	(R, z, ϕ)
Velocity	3	(v_R, v_z, v_ϕ)
HEQ	6	$(s, \alpha, \delta, v_\parallel, \mu_{\alpha*}, \mu_\delta)$
HGP	6	$(s, l, b, v_\parallel, \mu_{l*}, \mu_b)$
HCA	6	(x, y, z, v_x, v_y, v_z)
LSR	6	(x, y, z, v_x, v_y, v_z)
GCA	6	(x, y, z, v_x, v_y, v_z)
GCY	6	$(R, z, \phi, v_R, v_z, v_\phi)$
PSPD	4	(R, z, v_R, v_z) or $(J_r, J_z, \theta_r, \theta_z)$
PSPT	6	$(R, z, \phi, v_R, v_z, v_\phi)$ or $(J_r, J_z, J_\phi, \theta_r, \theta_z, \theta_\phi)$

A class `OmniCoords` is defined to facilitate transformations between coordinate systems. For example

```
OmniCoords OC; LSR w1;
w1[0]=1;w1[1]=2;w1[2]=3;
w1[3]=.01;w1[4]=.004;w1[5]=.005;
GCA w2=OC.GCAfromLSR(w1);
```

will take the phase-space point with heliocentric location $(x, y, z) = (1, 2, 3)$ kpc and LSR velocity $(U, V, W) \simeq (10, 4, 5)\,\mathrm{km\,s^{-1}}$ and return its galactocentric position and velocity in Cartesian coordinates. The class has methods for changing the solar motion, radius, etc., that are used in the transformations. For example

```
OC.change_sol_pos(8.3,0.014);
OC.change_vc(220*Units::kms);
```

The default epoch is J2000, but the method `change_epoch` will change this: for example

```
OC.change_epoch(1950)
```

2.3 Class Potential

TM needs to be able to evaluate a model potential and its derivatives with respect to R and z – the code is currently restricted to axisymmetric potentials. Hence `Phi(R,z)` must return the potential value, and `Phi(R,z,dPR,dPz)` must additionally leave in `dPR` and `dPz` the potential's radial and vertical derivatives. Using these derivatives, the base class

Table 3. Pre-defined potentials. The middle column gives the quantities fixed by the arguments of the creator. For example, to create a log potential, write `Potential *Phi = new LogPotential(.2,.8,.5);`

Class	arguments	formula
`IsochronePotential`	`(M,b)`	$\dfrac{-GM}{b+\sqrt{r^2+b^2}}$
`MiyamotoNagaiPotential`	`(M,a,b)`	$\dfrac{-GM}{[R^2+(a^2+\sqrt{b^2+z^2})^2]^{1/2}}$
`LogPotential`	`(V0,q,Rc)`	$\dfrac{V_0^2}{2}\log\left(R^2+\dfrac{z^2}{q^2}+R_c^2\right)$

provides a method $R_c(L_z)$ that returns the radius of the circular orbit with given angular momentum. Since many distribution functions involve epicycle frequencies, the class `Potential` has a method `KapNuOm` that computes for a circular orbit of given radius the radial and vertical epicycle frequencies and the azimuthal frequency. However, tori can be constructed and manipulated without defining this function, which requires second derivatives of the potential.

The TM package includes code for the three potentials listed in Table 3, and for the FALPOT package. The latter computes potentials that are generated by one or more double-exponential discs, usually representing the thin and thick stellar discs and the gas disc, and one or more spheroidal, double-power-law components, to represent the bulge and dark halo. The algorithm encoded in FALPOT is described by Dehnen & Binney (1998) and the code was extracted from Walter Dehnen's FALCON package. The lines

```
ifstream ifile; ifile.open("pot/PJM11_best.Tpot");
Potential *Phi=new GalaxyPotential(ifile);
```

initialise a potential of this class. The file opened in this example defines the "best" potential in McMillan (2011), which is produced by thin and thick stellar discs, a flattened bulge and a spherically symmetric dark halo. We use this "PM11 potential" in all our examples. The line

```
P=(*Phi)(R,z,dPR,dPz);
```

will set `P` to $\Phi(R,z)$ and leave the values of $\partial\Phi/\partial R$ and $\partial\Phi/\partial z$ in `dPR` and `dPz`, respectively, and

```
Frequencies epicycle=Phi->KapNuOm(R);
```

will set the components of `epicycle` to the frequencies κ, ν and Ω of the circular orbit of radius R.

Appendix C explains how users can define additional potentials, and gives details of a class `MultiPotential` that allows one to combine into a single potential several previously defined potentials, for example the potentials of a disc, a bulge and a dark halo.

2.4 Class Torus

All the needs of a typical end user of TM should be covered by the methods of the class `Torus`. A torus `T` with actions $(J_r, J_z, J_\phi) = (0.1, 0.2, 1)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ is created by the lines

```
Actions J; J[0]=.1; J[1]=.2; J[2]=1;
Torus T;
int flag = T.AutoFit(J,Phi);
```

where `Phi` is a pointer to a previously initialised model potential. The integer `flag` returned by `AutoFit` is zero if the call was entirely successful, with values −1 to −4 signalling various kinds of failure. Invoked thus, `AutoFit` uses default values of several arguments that could be explicitly set.

By far the most important of these optional parameters is TOL$_J$, which sets the precision of the constructed torus as follows. TM seeks to diminish the RMS fluctuation δH in $H(\mathbf{x},\mathbf{v})$ over the torus until $\delta H <$ TOL$_J\widetilde{\Omega}\widetilde{J}$, where

$$\widetilde{\Omega} \equiv \sqrt{\Omega_r^2+\Omega_z^2}, \tag{3}$$

$$\widetilde{J} \equiv \begin{cases} \sqrt{J_rJ_z} & \text{if } J_rJ_z \neq 0 \\ J_r+J_z & \text{otherwise.} \end{cases} \tag{4}$$

One may show that when $\delta H <$ TOL$_J\widetilde{\Omega}\widetilde{J}$, the fluctuations in J_r or J_z over the constructed torus are $\lesssim$ TOL$_J\widetilde{J}$. The choice of expression (4) for the scale action $\widetilde{J}$ is a compromise between expressions that take the scale to be of order the larger or smaller of the two actions. This is necessary because if, say, we have $J_r \gg J_z$ then taking $\widetilde{J} \sim J_r$ would imply the constraint was very weak compared to the energy that could plausibly be associated with motion in the z-direction, whereas taking $\widetilde{J} \sim J_z$ would provide an excessively stringent constraint on the accuracy required in describing the radial motion.

Once `T` has been created, it can be written to a file by

```
T.write_ebf(outname,"T1","w");
```

and read back with

```
T.read_ebf(outname,"T1");
```

Files are written in Sanjib Sharma's *Efficient Binary Format* so TM requires the EBF package for c++ to be installed. It can be found at sourceforge.net/projects/ebfformat/files/libebf/c-cpp/.

The energy of the created torus is

```
double E=T.energy();
```

The statements

```
Angles thetas;
thetas[0]=1; thetas[1]=2; thetas[2]=3;
GCY w=T.FullMap(thetas);
```

will put into the instance `w` of the class `GCY` of phase-space points (Table 2) the coordinates $(R, z, \phi, v_R, v_z, v_\phi)$ of the phase space point with the specified angle coordinates. The lines

```
Frequencies Om; Om=T.omega();
for(int i=0; i<2048; i++){
    thetas+=Om*2;
    w=T.FullMap(thetas);
    cout << w[0] << " " << w[1] << "\n";
}
```

will print 2048 points (R, z) along the orbit covering the elapse of 4094 Myr (Top panel of Fig. 1).

The lines

```
Position Rzphi;
Rzphi[0]=3; Rzphi[1]=.4; Rzphi[2]=1;
double s=T.DistancetoPoint(Rzphi);
```

will put `s` equal to the minimum distance between the point

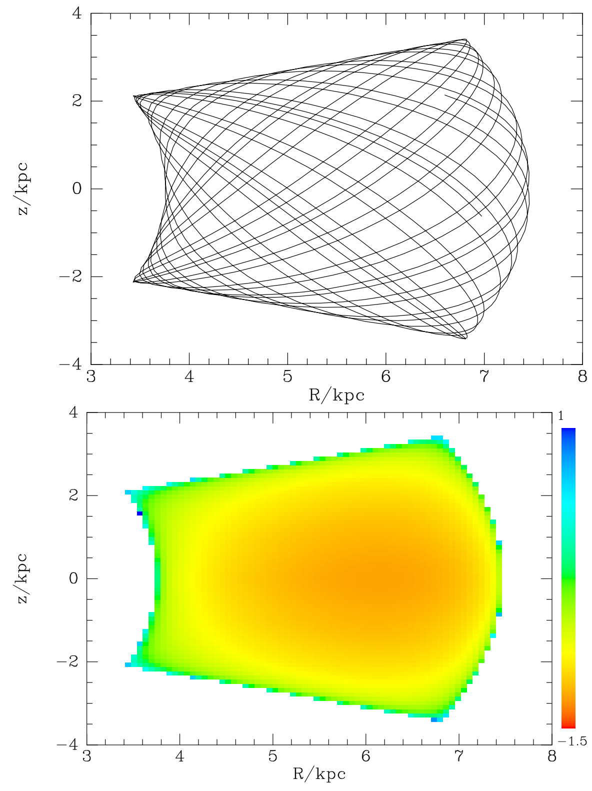

Figure 1. Top: a time sequence along the torus $\mathbf{J} = (0.1, 0.2, 1)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$. Bottom: logarithm of the density generated by this torus.

$(R, z, \phi) = (3, 0.4, 1)$ and a point on the orbit. If the given point lies within the three-dimensional region visited by the orbit, the returned value of s is naturally zero. Variants of `DistancetoPoint` exist that return information about the nearest point on the torus.

An analogous function `DistancetoPSP` returns the distances in $\mathbf{x}$ and $\mathbf{v}$ to the point on a torus that is closest to a given phase-space point when the velocity separation is scaled to a distance using a user-specified time-span.

```
PSPD w4;
w4[0]=8.5; w4[1]=0.014; w4[2]=0; w4[3]=0;
double t=4;
Vector<double,2> out=T.DistancetoPSP(w4,t,thetas);
```

will return in `out[0]` and `out[1]`, respectively, the real-space and velocity-space distances of `w4` from the nearest point on the orbit, when velocities are converted to distances by multiplication by 4 Myr (specified above by the value of `t`).

A method `containsPoint_Vel` is provided to discover if a given location lies on an orbit (when 1 is returned): and, if so, find the velocities (v_R, v_z) that the star will have at that location – two velocities (v_R, v_z) are returned, from which the four possible velocities can be recovered by multiplying by ± 1:

```
Velocity v1,v2;
if(T.containsPoint_Vel(Rzphi,v1,v2)==1)
printf("(%f,%f)(%f,%f)",v1[0],v1[1],v2[0],v2[1]);
```

An analogous method

```
containsPoint_Ang(Rzphi,theta1,theta2)
```

returns the angles (θ_r, θ_z) at which two visits occur – the angles of the remaining visits are $(-\theta_r, \pi - \theta_z)$.

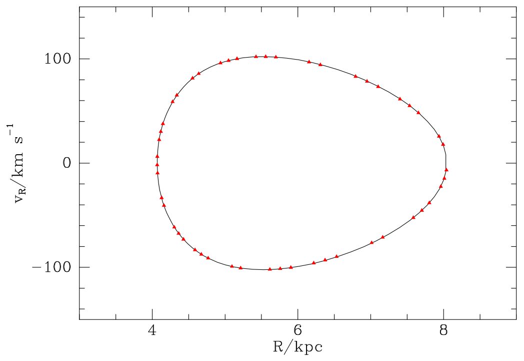

Figure 2. Full curve: surface of section of the torus $\mathbf{J} = (0.1, 0.2, 1)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ produced by the method `SOS` of the class `Torus`. Red points: consequents along a numerically integrated orbit.

The two methods just described call methods that can compute more than just two velocities or two angles of a visit. All these more sophisticated methods are called simply `containsPoint`. Some of them compute the matrix $\partial(R, z)/\partial(\theta_r, \theta_z)$ and/or the determinant $|\partial(x, y, z)/\partial(\theta_r, \theta_z, \theta_\phi)|$. The inverse of this determinant, which vanishes at the edge of the orbit, is proportional to the orbit's contribution to the density at the given point. The bottom panel of Fig. 1 shows the logarithm of the density generated by the orbit shown in the top panel. The relevant code reads

```
double det1,det2,**rho;
rho=PJM::matrix<double>(NX,NX);
for(int i=0;i<NX;i++){
  Rzphi[0]=xmin+i*(xmax-xmin)/(float)(NX-1);
  for(int j=0;j<NX;j++){
    Rzphi[1]=zmin+j*(zmax-zmin)/(float)(NX-1);
    int k=T.containsPoint(Rzphi,v1,det1,v2,det2);
    if(k==1)
      rho[i][j]=log10(1/fabs(det1)+1/fabs(det2));
    else
      rho[i][j]=-26;
  }
}
```

Since motion in an axisymmetric potential can be reduced to motion in the (R, z) plane, surfaces of section give valuable insight into the structure of phase space. The method `SOS()` computes the (R, v_R) surface of section of a torus, so the lines

```
ofstream to; to.open("SOS.dat");
T.SOS(to);
```

will write to the file `SOS.dat` $(R, z, v_R, v_z, \theta_z)$ for 200 points that have $z = 0$ and $v_z > 0$ (Fig. 2).

2.5 Class DF

An abstract base class `DF` provides basic distribution functions $f(\mathbf{J})$. A derived class `multidisk_DF` provides DFs that are the sum of any number of the "quasi-isothermal" DFs. The latter are defined by equation (5) below (Binney 2010; Binney & McMillan 2011; McMillan & Binney 2013). The values taken by these DFs depend explicitly on the actions, and implicitly on the potential through the epicycle frequencies. Instances of derived classes store the parameters of a

DF, and code that will provide the value of the DF given a value of **J** and a potential.

One often needs to evaluate a DF at a fixed value of **J**, in a fixed potential, for many different values of the DF's parameters. This is required, for example, when seeking the parameter values that provide the best fits to observational data (e.g. McMillan & Binney 2012, 2013). Significant computing time is saved by saving values, such as the epicycle frequencies, that will be needed by all DFs. The abstract base class `quickDF` makes it possible to store the parameters of the DF and any values associated with **J** and the potential that are required to re-evaluate the DF. Instances of the derived class `multidisk_quickDF` provide this functionality for DFs of the type implemented by `multidisk_DF`.

As defined by Binney & McMillan (2011), a quasi-isothermal DF is

$$f(\mathbf{J}) \equiv \left.\frac{\Omega_c\nu\Sigma}{2\pi^2 M\sigma_r^2\sigma_z^2\kappa}\right|_{R_c} \mathrm{cut}(J_\phi)\,\mathrm{e}^{-\kappa J_r/\sigma_r^2}\,\mathrm{e}^{-\nu J_z/\sigma_z^2}, \quad (5)$$

where R_c is the radius of a circular orbit with angular momentum J_ϕ, and the epicycle frequencies κ, ν and the circular frequency Ω_c are evaluated at R_c. The choice

$$\Sigma(J_\phi) \equiv \Sigma_0 \mathrm{e}^{-R_\mathrm{c}/R_\mathrm{d}} \quad (6)$$

ensures that disc's surface-density is an approximately exponential function of radius with scale-length R_d. $M = 2\pi\Sigma_0 R_\mathrm{d}^2$ is a normalisation chosen to ensure that $(2\pi)^3\int \mathrm{d}^3\mathbf{J}\, f(\mathbf{J}) = 1$. The factor $\mathrm{cut}(J_\phi)$ ensures we have different numbers of stars rotating in each direction. We use the form

$$\mathrm{cut}(J_\phi) = \tfrac{1}{2}\left[1 - \tanh(J_\phi/L_0)\right], \quad (7)$$

where the value of L_0 is small compared to the angular momentum of the Sun.

The functions $\sigma_z(J_\phi)$ and $\sigma_r(J_\phi)$ control the vertical and radial velocity dispersions,

$$\begin{aligned}\sigma_r(J_\phi) &= \sigma_{r0}\,\mathrm{e}^{(R_0-R_\mathrm{c})/R_\sigma}\\ \sigma_z(J_\phi) &= \sigma_{z0}\,\mathrm{e}^{(R_0-R_\mathrm{c})/R_\sigma},\end{aligned} \quad (8)$$

where R_σ is the scalelength on which the velocity dispersions decline.

We provide a function `set_DF` that reads the parameters of a DF from a file and returns a pointer to a DF. Several example input files are included with TM. For example

```
ifstream ifile; ifile.open("df/TwoDisk_MW.df");
DF *distfunc = set_DF(ifile);
```

initialises a DF with the parameters given in the file `TwoDisk_MW.df`. This file reads

```
m
2 8.5
27 20 3.0 6.67 10 1
48 44 3.5 7.78 10 0.3
```

where `m` indicates that this is a `multidisk_DF`, `2 8.5` indicates that it is the sum of two quasi-isothermal DFs with $R_0 = 8.5\,$kpc. The final two lines of the file give the parameters of the two quasi-isothermal DFs $(\sigma_{r0}, \sigma_{z0}, R_\mathrm{d}, R_\sigma, L_0, \mathrm{weight})$, where the weights of the two DFs need not sum to unity, and for convenience σ_{r0} and σ_{z0} are given in $\mathrm{km\,s^{-1}}$, and L_0 in $\mathrm{kpc\,km\,s^{-1}}$.

Code to find the value taken by the DF at a given **J** in a given potential reads

```
double f = distfunc->df(Phi,J);
```

where, as ever, `J` specifies the actions and `Phi` points to a `Potential`.

A `quickDF` cannot be set up from a file (as their raison d'etre is that they allow for changes in the DF). They are instead set up using a method `setup`, which takes as input (i) a pointer to a `Potential`, (ii) a value **J**, (iii) a pointer to an array of parameters, (iv) a pointers to a array of boolean values, and (v) the dimension of these arrays. For example

```
quickDF *qdf = new multidisk_quickDF;
double par[8]={1,8.5,0.027,0.02,3,6.67,0.01,1};
bool changeable[8];
for(int i=0;i<8;i++) changeable[i] = false;
changeable[2] = true;
qdf->setup(Phi,J,par,changeable,8);
double fJ = qdf->df();
```

sets up a `quickDF` with essentially the same parameters as the first of the two quasi-isothermal DFs in the `DF` set up above, and computes $f(\mathbf{J})$. Note that for this `setup` the parameters are given in code units. The array of boolean values specifies which parameters are liable to change: if a parameter will not change, numbers derived from it will not need to be recalculated. In the above example, only $\sigma_{r,0}$ is allowed to change. To find the value of the DF for a different value of $\sigma_{r,0}$ one would write

```
par[2]=0.028; fJ = qdf->df(par);
```

2.6 Class `tunableMCMC`

N-body realisations of a particular Galaxy model can be used to generate mock catalogues, or to initialise an N-body simulation with which to probe the response of the model to a perturbation or its behaviour during a merger. Instances of the class `tunableMCMC` use a simple Metropolis-Hastings Markov Chain Monte Carlo (MCMC) algorithm (Metropolis et al. 1953) to create a sample of N actions $\mathbf{J}_i$ with associated integer weights w_i such that for any function $h(\mathbf{J})$ of the actions

$$\int \mathrm{d}^3\mathbf{J}\, f(\mathbf{J})h(\mathbf{J}) \simeq \sum_i w_i h(\mathbf{J}_i) \Big/ \sum_i w_i, \quad (9)$$

where $f(\mathbf{J})$ is any given DF normalised such that $\int \mathrm{d}^3\mathbf{J}\, f(\mathbf{J}) = 1$. This MCMC algorithm avoids trying any negative values of J_r and J_z by working in terms of their square roots.

The class `tunableMCMC` has methods: `burn_in` designed to deal with burn-in of the MCMC chain; `find_sigs` which determines the standard deviations in $\sqrt{J_r}$, $\sqrt{J_z}$ and J_ϕ to provide a sensible proposal density for the MCMC chain (a Gaussian with these standard deviations in each direction); `tune` to tune the step size to a more appropriate value in each direction; and `output` which gives the output from the chain, either to a file or to vectors.

Included with TM are three main programs that perform the steps required to create an N-body model from a DF. `Choose_any_df` outputs a list of actions and their associated weights sampled from a `DF` using `tunableMCMC`. `Create_df_tori` takes this list and creates a torus for each one, storing it and its weight in a file. `Sample_list_limits` takes this file and outputs points in Galactocentric Cartesian coordinates. These points are uniformly distributed in $\boldsymbol{\theta}$ over tori that are randomly selected from the previously written list using the weights w_i.

Code that does a similar job, although omitting output at intermediate steps and sampling the tori in a random order, is as follows

```
time_t cpu=time(NULL);
Random3 R3(7*int(cpu)),R3b(123*int(cpu));
Gaussian Gau(&R3,&R3b);
J[0] = J[1] = 0.001; J[2] = -1.8;
tunableMCMC tMC(distfunc,Phi,J,&Gau,&R3);
int nTor=3000, nEach=10;
tMC.burn_in(nTor/10);
tMC.find_sigs(nTor/10);
tMC.tune(nTor/10,2);
vector<Actions> Jtab; vector<int> wtab;
tMC.output(nTor,&Jtab,&wtab);
for(int i=0;i<nTor;i++){
  T.AutoFit(Jtab[i],Phi);
  for(int j=0;j<nEach*wtab[i];j++){
    for(int k=0;k<3;k++)
        thetas[k] = 2*Pi*R3.RandomDouble();
    cout << T.FullMap(thetas) << '\n';
  }
}
```

3 ANGLE COORDINATES

In principle any point on the torus may be assigned $\boldsymbol{\theta} = 0$, but there are natural choices for the zero point. TM takes the zero point to be where the star is at pericentre with $z = 0$ and at azimuth $\phi = 0$. Given this zero point, there are some useful rules of thumb for interpreting angle coordinates.

- The value θ_ϕ is the ϕ coordinate of the guiding centre of the star's epicycles.
- $\theta_r \sim 0$ corresponds to pericentre, and $\theta_r \sim \pi$ to apocentre. We therefore typically have $v_R > 0$ for $0 \lesssim \theta_r \lesssim \pi$, and $v_R < 0$ for $\pi \lesssim \theta_r \lesssim 2\pi$.
- The orbit passes upwards through $z = 0$ at $\theta_z \sim 0$, reaches maximum distance above the plane ($z > 0$) at $\theta_z \sim \pi/2$, passes through the plane again (now with $v_z < 0$) at $\theta_z \sim \pi$, then reaches maximum distance below the plane at $\theta_z \sim 3\pi/2$.

These relations would be exact if the motion were separable in R and z. Fig. 3 shows a typical case.

The requirement that $\boldsymbol{\theta}$ increase linearly in time along an orbit computed using a Runge-Kutta or similar integrator provides a rigorous test of the accuracy of fitted tori. Fig. 4 shows this test for four tori. At top left we have a nearly circular orbit, at top right an orbit that is quite eccentric but lies nearly in the equatorial plane, at bottom left a less eccentric orbit that moves far from the plane, and at bottom right an eccentric and highly inclined orbit such as might be occupied by a halo globular cluster. Paths computed using the tori are shown by solid black lines and those computed by Runge-Kutta are shown with dashed red lines. The near coincidence of these lines demonstrates the precision that TM achieves.

All four tori were fitted with tolerance threshold $\text{TOL}_J = 0.0002$, which is significantly smaller than we typically use, in order to demonstrates the accuracy with which torus mapping can reproduce orbits that are far from resonance. As we explain below, the phenomenon of resonant trapping makes

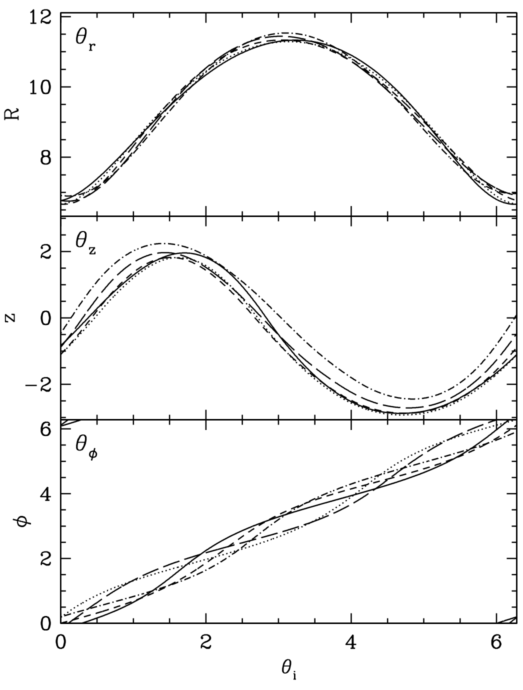

Figure 3. Typical relations between $\boldsymbol{\theta}$ and the real-space position of a thick-disc star. Each line shows how a real-space variable changes with its associated angle variable with the other two angle variables held constant. Different lines arise from different values of the constant angles. We see that $\phi \approx \theta_\phi$, $z \sim \sin\theta_z$ and $R - R_{\rm g} \sim -\cos\theta_r$, where $R_{\rm g}$ is the guiding centre radius. In the limit $J_r, J_z \to 0$ these approximations become exact.

it expedient to employ the larger value $\text{TOL}_J = 0.003$ in general.

4 INTERPOLATING BETWEEN TORI

In some applications one needs to sample densely a small region of action space – a prime example is when modelling stellar streams (Sanders 2014; Bovy 2014). Dense sampling of action space is also required when implementing Hamiltonian perturbation theory using an integrable Hamiltonian defined by tori (Kaasalainen 1995). When dense sampling is required, tori are best obtained by interpolation on a grid of fitted tori rather than by fitting all tori directly to the Hamiltonian. Moreover, we will see in Section 6 that when resonant trapping is important, it is absolutely essential to obtain tori for certain "missing actions" by interpolating between tori obtained for actions that are not subject to trapping.

Interpolation between tori was introduced by Kaasalainen & Binney (1994a) to implement Hamiltonian perturbation theory. They interpolated the Fourier coefficients $S_{\mathbf{n}}$ of the generating function (eqn. 2), their derivatives with respect to $\mathbf{J}$, and the parameters of the toy potential $\Phi^{\rm T}$. TM implements this scheme by making it possible to multiply tori by any real number and add them: these operations are interpreted as the corresponding operations on each $S_{\mathbf{n}}$ and each parameter of $\Phi^{\rm T}$. Kaasalainen & Binney (1994a) showed that linear

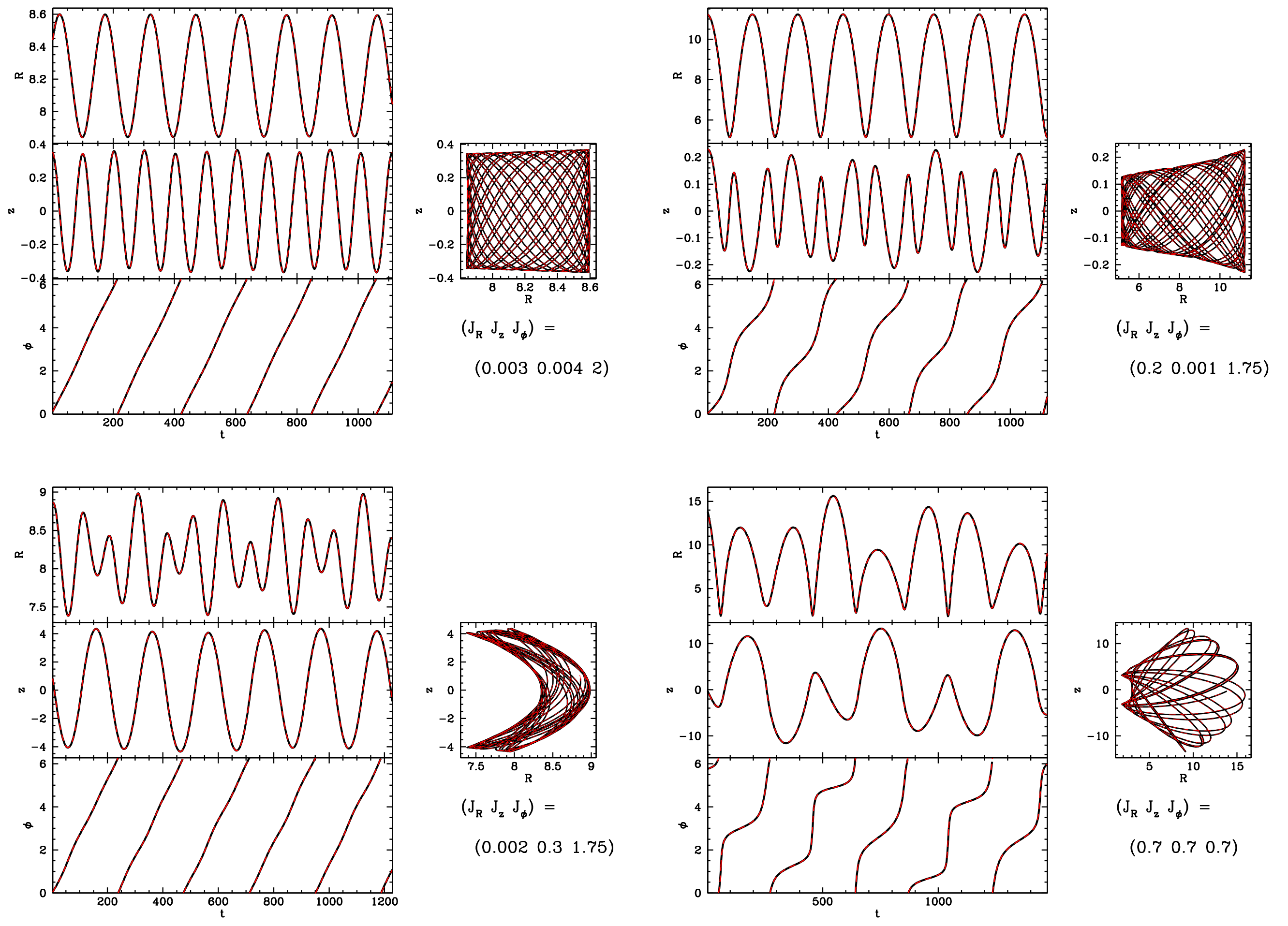

Figure 4. Orbits found by tracing the path associated with the expected linear increase in $\boldsymbol{\theta}$ using the torus machinery (solid black lines) or by Runge-Kutta integration in the gravitational potential (red dashed lines). The two agree to high precision.

interpolation in $\mathbf{J}$ provides an acceptable approximation, and we employ this method here, although in principle a higher-order interpolation scheme could be developed that takes advantage of the values of the derivatives $\partial S_{\mathbf{n}}/\partial \mathbf{J}$ for each fitted torus.

Code to enable the construction of tori near the torus T reads

```
Actions Jbar=T.actions(),dJ;
for(int j=0;j<3;j++) dJ[j]=.03;
Torus Tg; Actions Jg;
Torus ***Tgrid = PJM::matrix<Torus>(2,2,2);
for(int i=0;i<2;i++){
    Jg[0]=Jbar[0]+(i-.5)*dJ[0];
    for(int j=0;j<2;j++){
        Jg[1]=Jbar[1]+(j-.5)*dJ[1];
        for(int k=0;k<2;k++){
            Jg[2]=Jbar[2]+(k-.5)*dJ[2];
            Tg.SetToyPot(Phi,Jg);
            Tgrid[i][j][k].FitWithFixToyPot
              (Jg,Tg.TP(),Phi,.001);
        }
    }
}
Torus T2=InterpTorus(Tgrid,Jbar,dJ,Jbar);
```

Here we first declare an array of eight tori arranged at the corners of a cube in action space surrounding T and $0.1\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ on a side. Then we create a torus at each grid point by a two-step process: `SetToyPot` finds the parameters of a suitable toy potential, and then `FitWithFixToyPot` optimises the $S_{\mathbf{n}}$ for this Φ^{T}. Finally by interpolation on the corners of the cube we create a torus `T2` that should be very close to the original torus `T`.[2] Analogously to Fig. 4, Fig. 6 shows comparisons of integrated orbits and time sequences over tori constructed by this interpolation procedure. The grid was formed by the corners of a cuboid in action space centred on $\overline{\mathbf{J}} = (0.5, 0.5, 3.55)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$. The grid's diagonal is $\Delta\mathbf{J} = (0.2, 0.2, 0.7)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$. The actions at which two tori were obtained were randomly chosen from within the cuboid and are given below each right-hand panel of Fig. 6. The dashed red lines show the result of Runge-Kutta

[2] Normally tori are fitted by `AutoFit`, which adjusts the toy potential in parallel with the $S_{\mathbf{n}}$. Here we use `FitWithFixToyPot`, which fits only the $S_{\mathbf{n}}$ because when Φ^{T} is fitted simultaneously with the $S_{\mathbf{n}}$, trade-offs between them tend to impair smooth variation of the parameters with $\mathbf{J}$.

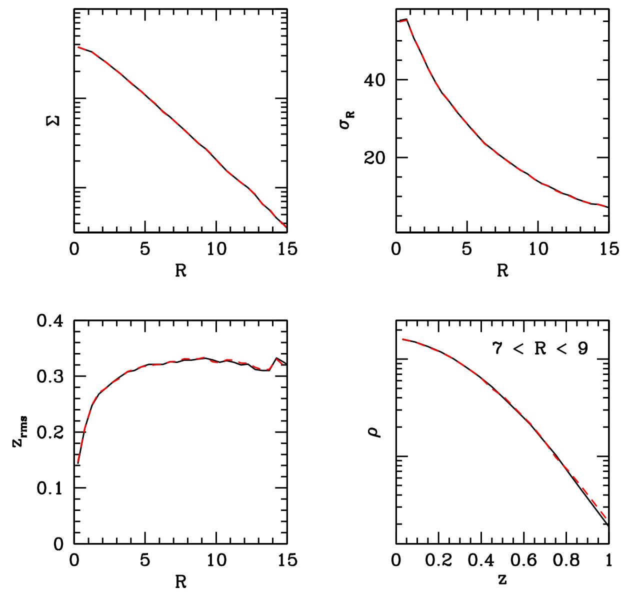

Figure 5. Properties of an $f(\mathbf{J})$ model sampled by TM (solid black line) and the same model evolved by Runge-Kutta integration of the orbits in the Galactic potential for 5 Gyr (red dashed line). The figures show the surface density Σ (top left), radial velocity dispersion σ_R (top right) and root-mean-squared distance from the plane $z_{\rm rms}$ (bottom left) as a function of Galactocentric radius, and the density profile ρ as a function of z for particles with Galactocentric radius $7\,{\rm kpc} < R < 9\,{\rm kpc}$. The lines are almost identical because the model starts from statistical equilibrium, as it should.

integration, while the black lines show time sequences along two tori obtained by interpolation. The curves overlie each other to a satisfactory extent.

An independent check on the accuracy of an interpolated torus is provided by the RMS variation δH of the Hamiltonian (1) over the torus. The grid tori were fitted with tolerance parameter $\text{TOL}_J = 0.001$, which gives us an average $\delta H = 2\times10^{-3}\,{\rm kpc}^2\,{\rm Myr}^{-2}$ for these tori. The average interpolated torus has $\delta H = 4\times10^{-3}\,{\rm kpc}^2\,{\rm Myr}^{-2}$, which indicates perfectly acceptable accuracy.

In a future paper we will use torus interpolation to model stellar streams.

5 N-BODY MODELS

As described in Section 2.6, TM provides a means of choosing initial conditions for N-body simulations of realistic disc galaxies that start in perfect equilibrium. Here we illustrate this technique in the case of a tracer population that moves in a fixed potential. However, now that Binney & Piffl (2015) have shown how to determine the potential that is jointly created by populations of stars and dark-matter particles that each have prescribed DFs $f(\mathbf{J})$, one could in principle use tori to set up the initial conditions of a self-consistent galaxy by first relaxing the joint potential as Binney & Piffl (2015) describe and then using a pointer `Phi` to that potential. Here `Phi` points to the PM11 potential.

Fig. 5 shows plots for a population sampled from a DF of the form (5). The parameters of the DF are $R_{\rm d} = 3\,{\rm kpc}$, $R_\sigma = 6.67\,{\rm kpc}$, $\sigma_{r0} = \sigma_{z0} = 20\,{\rm km\,s^{-1}}$ and $L_0 = 10\,{\rm kpc\,km\,s^{-1}}$. 50 000 values of $\mathbf{J}$ were sampled from this DF using an instance of `tunableMCMC`.[3] We find the corresponding tori, and from each torus sample 20 values of $\boldsymbol{\theta}$, so we have 10^6 particles in all. On a single core of a typical laptop it takes 90 minutes to fit the 50 000 tori (a rate of $\sim 10\ {\rm s}^{-1}$) and 2 minutes to sample the 10^6 particles (a rate of $\sim 10\,000\ {\rm s}^{-1}$). Both tasks are trivially parallelisable.

The full black lines in Fig. 5 show the surface density, radial and vertical velocity dispersions, disk thickness and vertical profile in the range $7\,{\rm kpc} < R < 9\,{\rm kpc}$ that result from the sampling. The dashed red lines show the same profiles after the orbit of every particle has been evolved with a Runge-Kutta integrator for 5 Gyr. The differences between the initial and evolved distributions are tiny and consistent with Poisson noise. This demonstrates that the DF is has been properly sampled, so the initial distribution is consistent with the Jeans theorem.

6 RESONANCES

Real galaxy potentials are almost certainly not integrable. That is, these potentials manifest the phenomenon of resonant trapping, in which orbits librate around an orbit with commensurable fundamental frequencies, so $\mathbf{N}\cdot\boldsymbol{\Omega} = 0$ for a vector $\mathbf{N}$ with integer components. The extent of this phenomenon varies widely from potential to potential, for reasons that are not fully understood (see Binney & Tremaine 2008, §3.7.3 for an interesting example).

When orbits are effectively two-dimensional (as in an axisymmetric potential), the extent of resonant trapping can be determined from surfaces of section. Resonant trapping of fully three-dimensional orbits is best probed with a frequency map (Laskar 1993; Binney & Tremaine 2008, §3.7.3(b)). In general the fraction of phase space occupied by trapped orbits is small in typical axisymmetric potentials, and moderate in triaxial potentials that have large core radii and low pattern speeds. Shrinking the core radius of a triaxial potential increases the fraction of trapped orbits (Merritt & Valluri 1999), and even with a large core radius, orbits comparable in size to the potential's corotation radius have a significant probability of being resonantly trapped.

Resonantly trapped orbits are quasiperiodic like their untrapped cousins. But when the regions of influence of individual resonances overlap, orbits cease to be quasiperiodic (Chirikov 1979) and one says that they are chaotic. Numerical experiments suggest that many chaotic orbits are successively trapped by different resonances, with the consequence that their frequencies are stable for only brief periods of time.

When we use a torus-mapping code, we pre-determine the gross structure of the image torus by specifying the toy potential and any point transformation, and the code merely

[3] As explained in Section 2.6, some values of $\mathbf{J}$ occur $w > 1$ times in the MC chain, and in this case $\mathbf{J}$ is specified only once but given weight $w > 1$. We sample 50 000 distinct tori, some with weights $w_i > 1$. Then the probability that the ith torus will be drawn from this library is $w_i/\sum_j w_j$, and on the selected torus 20 points are chosen.

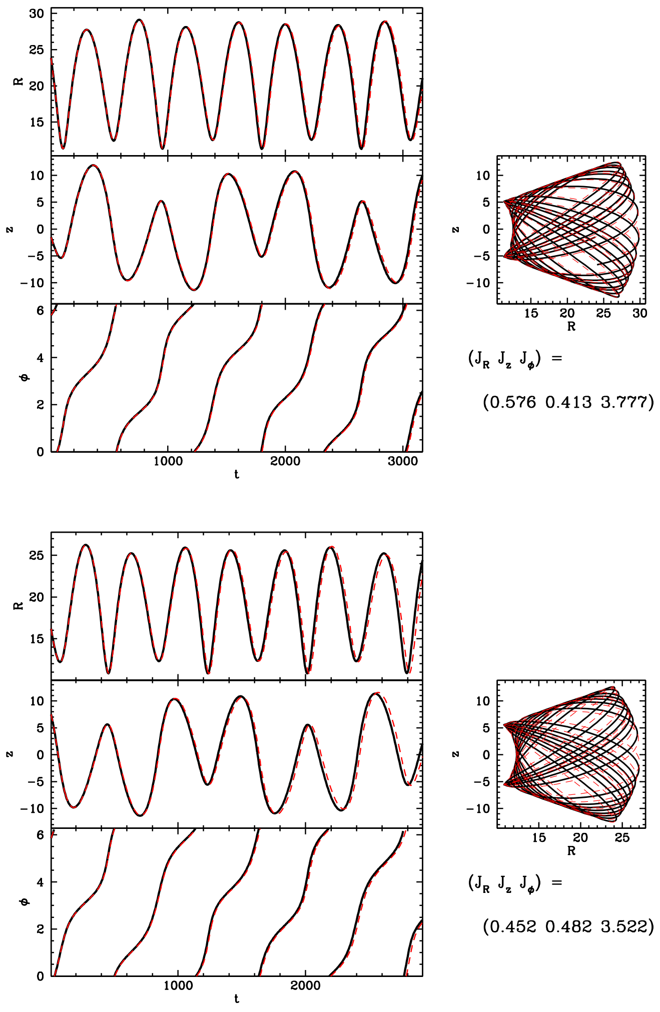

Figure 6. Orbits found by tracing the path associated with the expected linear increase in $\boldsymbol{\theta}$ using interpolated tori (solid black lines) or by Runge-Kutta integration in the gravitational potential (red dashed lines).

tweaks the image torus to make H as constant as possible over its surface.

In the case of TM, the tori have the gross structure of non-resonant tori. When such a non-resonant orbital torus exists with the specified actions, TM should in principle be able to drive δH to arbitrarily small values. When there is no non-resonant torus with the specified actions, it is impossible for TM to drive δH to zero. Hence when TM is employed, resonant trapping can be signalled by failure of `AutoFit` to achieve a small specified value of TOL_J and consequently return a non-zero value.

Near the Sun the most important resonance from the perspective of trapping is that for which $\Omega_r = \Omega_z$. At any given energy, satisfaction of this condition on some orbits is guaranteed by the fact that for nearly circular orbits $\Omega_z > \Omega_r$ because the disc is massive and thin, while orbits that make large excursions in z, and thus perceive only a mildly flattened potential, have $\Omega_z < \Omega_r$.

Each row of Fig. 7 shows an orbit that is trapped by the resonance $\Omega_r = \Omega_z$. The orbit shown in the upper row is trapped such that the R and z oscillations are always roughly in phase,[4] so every time the star moves up through the Galactic plane, it is moving outwards. In consequence of this, the consequents of this orbit are confined to the upper half ($v_R > 0$) of the surface of section (second panel from the left of Fig. 7).

The orbit shown in the lower row of Fig. 7 is trapped such that its z oscillations lead its R oscillations by $\sim \pi/2$ in phase, with the consequence that the orbit has a definite sense of circulation in the (R, z) plane. The white square at the centre of the orbit is analogous to the unvisited circle at the centre of a familiar rosette orbit in an axisymmetric potential. In the surface of section (second panel from the left), the orbit's consequents are confined to the left half of the diagram because the orbit is always near pericentre when it rises through the plane.

On account of the potential's symmetry in z, each of the two plotted orbits is associated with a mirror orbit obtained by mapping $z \to -z$. In the third column of Fig. 7, the mirror of the top orbit slopes from upper left to lower right, and in the second column its consequents would lie in the lower half of the diagram. The mirror of the lower orbit in Fig. 7 circulates in the opposite sense, so it would look the same as that plotted in the third column, but its consequents would be confined to the right side of the panel in the second column.

In the PM11 potential, trapping such that the relative phases of the R and z oscillations librates around 0 or π occurs at $J_\phi \lesssim 1.15\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ (upper panels of Fig. 7), while at larger values of J_ϕ the relative phase of the R and z librates around $\pm\pi/2$. Thus a surface of section that showed both orbits such as those plotted in Fig. 7 and their mirrors would show a pair of islands that touched at two points on the R axis when $J_\phi < 1.15\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ and at top and bottom when J_ϕ is larger. Orbits with $J_\phi = 1.15\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ are not trapped by the 1:1 resonance.

The black curves in the surfaces of section would run through these islands. They show the cross section of the torus that has the actions these orbits would have if they were not resonant. In reality *no* orbit has these values of J_r and J_z – Binney & Spergel (1984) called these "missing actions". Since these resonant orbits are quasiperiodic, they do have actions, but one of these actions measures the extents of their libration around the trapping resonant orbit and not the extent of their radial and vertical excursions.

In summary, with TM we can generate tori for any values of the actions. Even though the frequencies of some tori will be commensurable, all tori represent non-resonant orbits in the sense that the tori impose no phase relationship between the R and z oscillations. For many values of $\mathbf{J}$, non-resonant orbits exist with these actions, and TM should be able to generate these orbits to arbitrary precision, with the consequence that on the computed tori the RMS fluctuation in H can be made arbitrarily small. In some ranges of $\mathbf{J}$ it is impossible to fit tori of the specified form into surfaces of constant H. Consequently, on any tori with these values of $\mathbf{J}$ that TM constructs, H will fluctuate by a non-negligible amount.

[4] In phase in the sense that smallest z occurs at smallest R. But with TM's zero-points for $\boldsymbol{\theta}$ this implies $\theta_r - \theta_z \simeq \pi/2$.

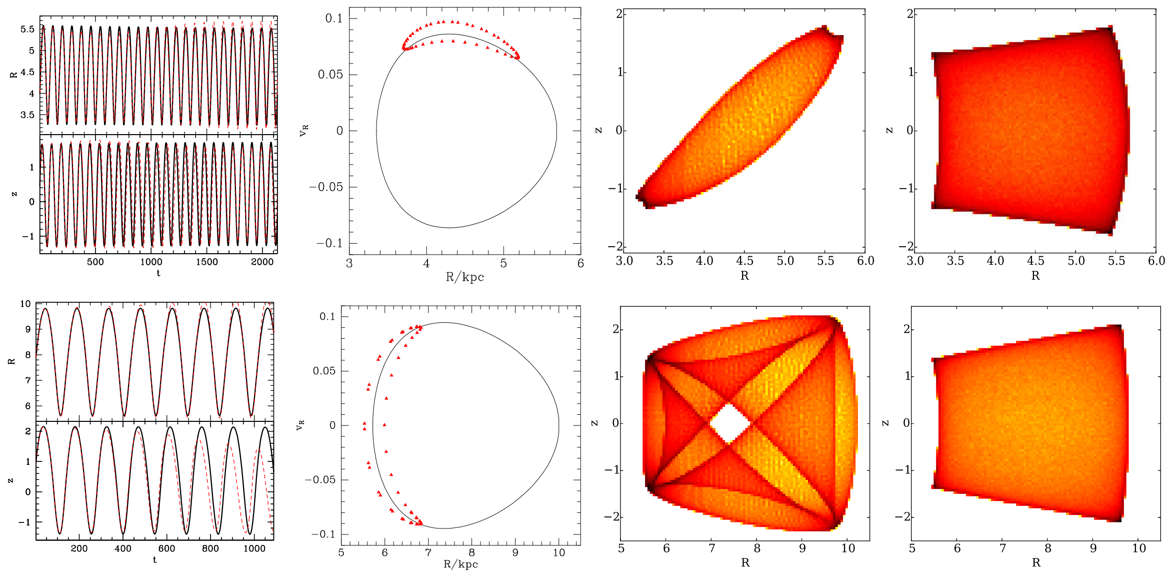

Figure 7. Plots illustrating the differences when near resonance between orbits found by following the expected path on a fitted torus and found by Runge-Kutta integration. The upper and lower panels show different orbits, both near a 1:1 resonance. The upper panels are for a torus with actions $\mathbf{J} = (0.05, 0.085, 0.9)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ and frequency ratio $\Omega_z/\Omega_r = 0.99674$ while the lower panels are for a torus with actions $(0.1, 0.07, 1.7)\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$ and frequency ratio 1.0015. In each case the leftmost panel shows R and z as a function of t for the torus (black) and for the integrated orbit (red); the second-left panel shows the surface of section (i.e. values of R and v_R as the orbit passes $z = 0$ with $v_z > 0$), again with that for the torus in black and that for the integrated orbit in red. The density plots show the density in the (R, z) plane associated with the integrated orbit (second-right) or fitted torus (rightmost). Darker colours correspond to higher densities.

If we use TM to foliate phase space with tori, and assign to the torus $\mathbf{J}$ the average value, $\overline{H}(\mathbf{J})$ that H takes on that torus, then $\overline{H}$ is an integrable Hamiltonian for which we have angle-action coordinates. The difference

$$\Delta(\boldsymbol{\theta}, \mathbf{J}) \equiv H(\boldsymbol{\theta}, \mathbf{J}) - \overline{H}(\mathbf{J}) \tag{10}$$

is a small perturbation. The existence of this perturbation to an integrable Hamiltonian explains why real orbits are trapped by the resonance, and the trapping process can be studied with first-order Hamiltonian perturbation theory (Kaasalainen & Binney 1994a; Kaasalainen 1995). In the example just discussed, $\Delta(\boldsymbol{\theta}, \mathbf{J})$ at a given value of $\boldsymbol{\theta}$ evidently changes sign as J_ϕ varies across the value $J_\phi = 1.15\,\mathrm{kpc}^2\,\mathrm{Myr}^{-1}$, and vanishes at this particular value of J_ϕ.

Crucial for the construction of $\overline{H}$ is that TM creates tori for adjacent values of the actions which touch but do not cross: if we are to have a global system of angle-action coordinates, each point in phase space must be on precisely one torus. This requirement that tori constructed for neighbouring values of $\mathbf{J}$ do not cross is non-trivial.

The existence of resonant trapping has implications for the choice of the tolerance parameter TOL_J. First, given the logical impossibility of driving δH to zero when trapping occurs, very small values of TOL_J are inappropriate: the specified value of TOL_J should be large enough for the task TM is set to be feasible. Moreover, in parts of action space that correspond to orbits that are strongly influenced by a resonance that nevertheless does not trap them, it may be expedient to avoid small values of TOL_J and closely fitting tori in order to prevent adjacent tori crossing. That is, if you want a clean foliation of phase space with tori, you should limit the extent to which resonances distort tori by using a moderate value of TOL_J. We plan to examine more carefully the many issues raised by orbital trapping in a forthcoming paper.

6.1 Star-streaming in the Rz plane?

Given that there are orbits with a well defined sense of circulation in the Rz plane, one could construct an equilibrium Galaxy model in which $\langle v_z\rangle \neq 0$ in the Galactic plane. This is a remarkable possibility. Naively one would interpret a non-vanishing value of $\langle v_z\rangle$ in the Galactic plane as evidence for a non-equilibrium process, such as flapping or warping. Careful examination of the range of values of $|v_z|$ in which there were more stars moving up/down than down/up could distinguish between the equilibrium and non-equilibrium possibilities: in the former case the up/down asymmetry would be confined to velocities characteristic of the 1:1 resonance. Moreover, a net upward flow of stars at one radius would be balanced by a net downward flow at a small/larger radius in a velocity range that could be accurately predicted given knowledge of the potential. The Sun's motion perpendicular to the plane is measured relative to stars that are too tightly confined to the plane to be affected by the resonance, so the conventional value, $v_z = 7.25\,\mathrm{km\,s}^{-1}$, is secure.

As a massive star cluster is tidally disrupted and donates its stars to the thick disc, it will be moving either down or up through the disc at a particular radius. If this radius is one associated with orbits appropriately trapped by the 1:1 resonance, its stars will subsequently have a net sense of circulation in the Rz plane. A burst of star formation induced by a massive, dense cloud hitting the disc at an appropriate radius could likewise induce a net circulation in part of the thick disc. Consequently, hunting for this phenomenon is star catalogues seems a worthwhile activity.

7 CONCLUSION

We have presented a package of C++ code, TM, that allows the user to find the orbital torus associated with given values of the actions $\mathbf{J}$ in any axisymmetric potential Φ. Once a torus has been fitted, it is trivial to determine the star's position, velocity and contribution to the local density for any value $\boldsymbol{\theta}$ of the star's angle variables. Alternatively, one can choose a location $\mathbf{x}$ and determine whether the star will ever visit that location, and if so with which velocities and how much it will contribute to the density there. Since the orbital frequencies are available, the star's trajectory from any location on the orbit can be readily recovered without integrating the equations of motion. Equally, one can recover the curve on which its consequents will lie in a surface of section.

On a typical single processor ~ 10 tori can be fitted per second. Evaluating quantities for a given torus takes just a few ms. Although a torus completely describes a star's motion for all times, it can be stored in ~ 100 numbers. TM provides for the construction of new tori by interpolation on fitted tori, which is an exceedingly fast operation.

Initial conditions that are in statistical equilibrium in a given potential are readily chosen with tools provided in TM: in addition to torus-finding and manipulating routines, the package includes distribution functions for realistic discs and a MCMC sampler.

Our discussion of the connection between resonant trapping and torus mapping has been rather superficial because this topic merits a paper on its own. The key points are (i) that torus mapping provides accurate representation of all orbits that are not resonantly trapped, and (ii) that it provides the means to construct an integrable Hamiltonian that can be used to compute resonant trapping precisely, and to gain a deeper understanding of this phenomenon and its implications for galactic dynamics. Examination of trapping by the 1:1 resonance is a realistic Galactic potential indicates that at some radii and amplitudes of vertical excursions, stars may have a well defined sense of circulation in the Rz plane.

The TM package can be downloaded from github.com/PaulMcMillan-Astro/Torus. It contains a number of example programmes, including ones that produce the data plotted here. Development of TM continues, and improvements and additions will be made from time to time. Relevant software can be downloaded from github.com/GalacticDynamics-Oxford/ABGal, that includes provision for linking TM to code implementing the Stäckel fudge, so that approaches combining the techniques (see e.g. Sanders & Binney 2015) are easy to use.

ACKNOWLEDGEMENTS

Several colleagues have made major contributions to this project. In the early 1990s Colin McGill and Mikko Kaasalainen contributed to most of the key ideas and wrote early FORTRAN codes. In 1995–6 Walter Dehnen rewrote Kaasalainen's code in C++ and most of the class structure of TM derives from this rewrite. The project has been generously supported by successive UK research councils: SERC prior to 1994, PPARC from 1994–2007, and latterly by STFC under grants ST/G002479/1, ST/J00149X/1, and ST/K00106X/1.

The research leading to these results has also received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no. 321067.

PJM gratefully acknowledges financial support from the Swedish National Space Board and the Royal Physiographic Society in Lund.

We thank Payel Das and Eugene Vasiliev for useful comments on a draft.

REFERENCES

Arnold V., 1989, Mathematical methods of classical mechanics. Vol. 60, Springer
Bacon R., Simien F., Monnet G., 1983, A&A, 128, 405
Binney J., 2010, MNRAS, 401, 2318
Binney J., 2012a, MNRAS, 426, 1324
Binney J., 2012b, MNRAS, 426, 1328
Binney J., 2013, Dynamics of secular evolution. Cambridge University Press, p. 259
Binney J., Kumar S., 1993, MNRAS, 261, 584
Binney J., McMillan P., 2011, MNRAS, 413, 1889
Binney J., Piffl T., 2015, MNRAS, p. xx
Binney J., Spergel D., 1984, MNRAS, 206, 159
Binney J., Tremaine S., 2008, Galactic Dynamics: Second Edition. Princeton University Press
Binney J. J., Davies R. L., Illingworth G. D., 1990, ApJ, 361, 78
Bovy J., 2014, ApJ, 795, 95
Bovy J., Rix H.-W., 2013, ApJ, 779, 115
Chirikov B. V., 1979, Phys. Rep., 52, 263
Dehnen W., 2009, MNRAS, 395, 1079
Dehnen W., Binney J., 1998, MNRAS, 294, 429
Gebhardt K., Richstone D., Tremaine S., Lauer T. R., Bender R., Bower G., Dressler A., Faber S. M., Filippenko A. V., Green R., Grillmair C., Ho L. C., Kormendy J., Magorrian J., Pinkney J., 2003, ApJ, 583, 92
Häfner R., Evans N. W., Dehnen W., Binney J., 2000, MNRAS, 314, 433
Henon M., 1959, Annales d'Astrophysique, 22, 126
Jeans J. H., 1915, MNRAS, 76, 70
Kaasalainen M., 1994, MNRAS, 268, 1041
Kaasalainen M., 1995, MNRAS, 275, 162
Kaasalainen M., Binney J., 1994a, Physical Review Letters, 73, 2377
Kaasalainen M., Binney J., 1994b, MNRAS, 268, 1033
Laakso T., Kaasalainen M., 2013, Physica D Nonlinear Phenomena, 243, 14
Laskar J., 1993, Celestial Mechanics and Dynamical Astronomy, 56, 191
Lichtenberg A. J., Lieberman M. A., 1983, Regular and stochastic motion. Springer
McGill C., Binney J., 1990, MNRAS, 244, 634
McMillan P. J., 2011, MNRAS, 414, 2446
McMillan P. J., Binney J. J., 2012, MNRAS, 419, 2251

McMillan P. J., Binney J. J., 2013, MNRAS, 433, 1411
Merritt D., Valluri M., 1999, AJ, 118, 1177
Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E., 1953, Journal of Chemical Physics, 21, 10871092
Morganti L., Gerhard O., 2012, MNRAS, 422, 1571
Piffl T., Binney J., McMillan P. J., et al. 2014, MNRAS, 445, 3133
Piffl T., Penoyre Z., Binney J., 2015, MNRAS, 451, 639
Press W. H., Flannery B. P., Teukolsky S. A., 1986, Numerical recipes. The art of scientific computing. Cambridge: University Press, 1986
Sanders J. L., 2014, MNRAS, 443, 423
Sanders J. L., Binney J., 2013, MNRAS, 433, 1813
Sanders J. L., Binney J., 2015, MNRAS, 447, 2479
Schönrich R., Binney J., Dehnen W., 2010, MNRAS, 403, 1829
Schwarzschild M., 1979, ApJ, 232, 236
Syer D., Tremaine S., 1996, MNRAS, 282, 223
van der Marel R. P., Binney J., Davies R. L., 1990, MNRAS, 245, 582
van der Marel R. P., Cretton N., de Zeeuw P. T., Rix H.-W., 1998, ApJ, 493, 613

APPENDIX A: WHAT TM ACTUALLY DOES

Classically, angle-action coordinates $(\boldsymbol{\theta}, \mathbf{J})$ are obtained by solving the Hamilton-Jacobi equation for the generating function $S(\mathbf{x}, \mathbf{J})$ of the canonical transformation $(\mathbf{x}, \mathbf{v}) \leftrightarrow (\boldsymbol{\theta}, \mathbf{J})$. Since analytic solutions of the Hamilton-Jacobi equation are not available for generic potentials, TM obtains the required transformation by compounding up to three canonical transformations:

$$(\boldsymbol{\theta}, \mathbf{J}) \xrightarrow{S_{\mathbf{n}}} (\boldsymbol{\theta}^{\rm T}, \mathbf{J}^{\rm T}) \xrightarrow{\text{HJ eqn}} (\mathbf{x}^{\rm T}, \mathbf{v}^{\rm T}) \xrightarrow{\text{Point transf}} (\mathbf{x}, \mathbf{v}). \tag{A1}$$

The point transformation $\mathbf{x}^{\rm T}(\mathbf{x})$ is employed only if $J_z/J_r > 0.05$ and an attempt to fit a torus without a point transformation has failed. If it is required, TM finds a point transformation that maps the relevant circular orbit $J_r^{\rm T} = 0$ in the toy potential $\Phi^{\rm T}$ into the shell orbit $J_r = 0$ in the Galactic potential. Otherwise it just uses the identity transformation. Next it solves for the coefficients $S_{\mathbf{n}}$ of the generating function (2) and the optimum parameters of $\Phi^{\rm T}$. Finally it determines the derivatives $\partial S_{\mathbf{n}}/\partial \mathbf{J}$, which are required to map between toy and true angle variables,

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{\rm T} + 2 \sum_{\mathbf{n}>0} \frac{\partial S_{\mathbf{n}}(\mathbf{J})}{\partial \mathbf{J}} \sin(\mathbf{n} \cdot \boldsymbol{\theta}^{\rm T}). \tag{A2}$$

Fig. A1 illustrates the central idea of torus mapping with a simple one-dimensional example that does not require a point transformation. In each panel the full black curve shows the phase space trajectory of the orbit we seek to model. The angle-action coordinates of a default toy potential $\Phi^{\rm T}$ constitute a system of polar coordinates $(\theta^{\rm T}, J^{\rm T})$ for phase space. In the leftmost panel this system is symbolised by a series of circles of constant $J^{\rm T}$. The dashed red curve shows the coordinate curve that encloses the same area as the orbit's curve, and therefore has $J^{\rm T} = J$, the orbit's action.

In the next panel the parameters of $\Phi^{\rm T}$ have been adjusted to bring the dashed red curve $J^{\rm T} = J$ into closer alignment with the orbit. In the next panel a generating function with single coefficient $S_{\mathbf{n}}$ has been used to deform the red curve whilst leaving the area it encloses constant. In the final panel a generating function with several $S_{\mathbf{n}}$ has been used to bring the red curve into alignment with the black curve to the precision required by the user.

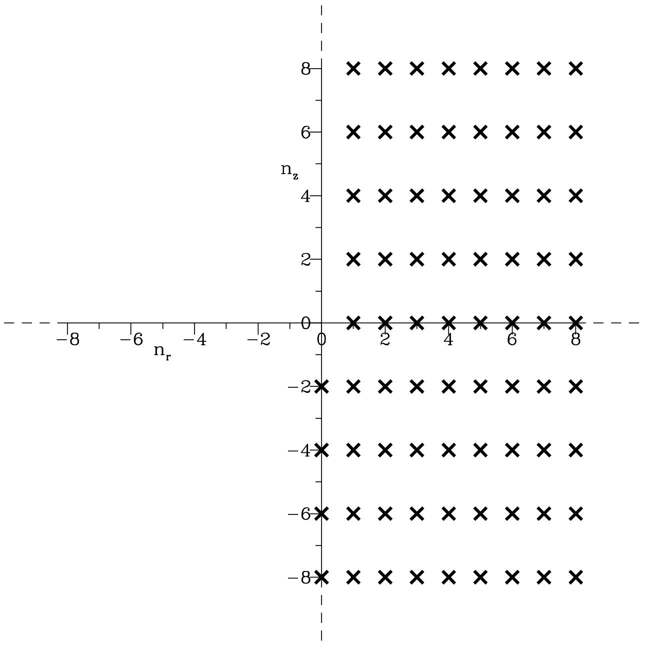

Figure A2. Figure illustrating the pattern of allowed values of (n_r, n_z) in the generating function (eq. A3). The figure stops at $n_i = 8$, but terms can be used (if they follow the same rules) for $n \to \infty$

A1 The generating function

Given that Φ is mirror symmetric in the Galactic plane, the general form (2) of the generating function can be specialised to (McGill & Binney 1990)

$$S(\boldsymbol{\theta}^{\rm T}, \mathbf{J}) = \boldsymbol{\theta}^{\rm T} \cdot \mathbf{J} + 2 \sum_{\mathbf{n}} S_{\mathbf{n}}(\mathbf{J}) \sin(\mathbf{n} \cdot \boldsymbol{\theta}^{\rm T}). \tag{A3}$$

where the $S_{\mathbf{n}}$ are real, $n_r > 0$ for $n_z \geqslant 0$, and only even values of n_z occur – the allowed values of $\mathbf{n}$ are illustrated in Figure A2. At any point $\boldsymbol{\theta}^{\rm T}$ on the torus with actions $\mathbf{J}$, the toy actions are then given by

$$\mathbf{J}^{\rm T} = \mathbf{J} + 2 \sum_{\mathbf{n}>0} \mathbf{n} S_{\mathbf{n}}(\mathbf{J}) \cos(\mathbf{n} \cdot \boldsymbol{\theta}^{\rm T}). \tag{A4}$$

For given $\mathbf{J}$ and a regularly-spaced grid of values $\boldsymbol{\theta}^{\rm T}$, TM evaluates $\mathbf{J}^{\rm T}$ and uses the analytic relation between $(\boldsymbol{\theta}^{\rm T}, \mathbf{J}^{\rm T})$ and $(\mathbf{x}^{\rm T}, \mathbf{v}^{\rm T})$ and then the point transformation $\mathbf{x}(\mathbf{x}^{\rm T})$ to evaluate $H(\mathbf{x}, \mathbf{v})$, and thus determines the variance $(\delta H)^2$ of H around the torus. The Levenberg-Marquardt algorithm (Press et al. 1986) is then used to adjust the $S_{\mathbf{n}}$ and possibly the parameters of $\Phi^{\rm T}$ to minimise $(\delta H)^2$. The Levenberg-Marquardt algorithm requires the derivatives of H with respect to the $S_{\mathbf{n}}$ and the parameters of $\Phi^{\rm T}$, and these are all analytically computed using the chain rule. On account of the symmetries of the potential, the grid in $\boldsymbol{\theta}^{\rm T}$ can be restricted to $0 \leqslant \theta_r^{\rm T} < \pi$ and $0 \leqslant \theta_z^{\rm T} < \pi$.

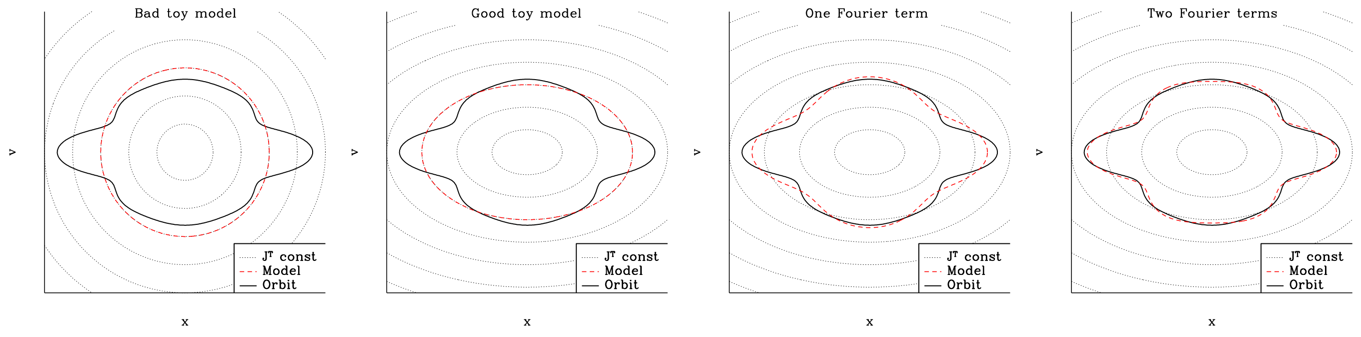

Figure A1. Illustration of the torus method in the 1D case. Angle-action coordinates in the toy potential provide a polar coordinate system (dotted lines) in the q-p plane with the actions giving a radius and the angle as a polar angle (left). To fit the true orbit with actions J' (solid line) we start with the line $J^T = J'$ in a toy potential (red dashed - first panel), then adjust a parameter of the toy potential such that this provides a better fit to the orbit (second panel). We then add terms periodic in θ^T that conserve the total area within the red dashed curve (i.e. conserve the action), which allow us to describe the true orbit with greater and greater accuracy as we add more terms (third and fourth panels). *Modify (i) to use* (x, v_x) *instead of (q,p); (ii) delete coord curves from 3rd & 4th panels.*

TM starts with a small number of coefficients $S_{\mathbf{n}}$, by default those with $\mathbf{n} = (1,0)$, $(2,0)$ $(3,0)$, $(0,-2)$, $(0,-4)$, $(1,2)$, $(1,-2)$, $(1,4)$. After an optimisation of these coefficients it adds coefficients at values of $\mathbf{n}$ that are adjacent to points at which $|S_{\mathbf{n}}|$ has been found to be above a threshold value that is a fixed fraction of the largest $|S_{\mathbf{n}}|$. Each $S_{\mathbf{n}}$ is set to zero on its introduction. When additional $S_{\mathbf{n}}$ are introduced, it may be necessary to make the grid in $\boldsymbol{\theta}^{\rm T}$ denser to ensure that the sampling density around the torus comfortably exceeds the Nyquist frequency associated with the largest value of $|\mathbf{n}|$. Consequently, the computational cost of each optimisation step increases quite rapidly as the number of $S_{\mathbf{n}}$ increases. The number of optimisation steps is limited, by default to 10. After 10 steps there are typically ~ 700 non-zero $S_{\mathbf{n}}$.

Once a satisfactory variance in H has been achieved, TM solves for the derivatives of the $S_{\mathbf{n}}$, which appear in the relation (A2) between the toy and true angles. The derivatives are obtained analytically rather than by finite differences. Currently TM does this using the algorithm introduced by Kaasalainen & Binney (1994b).[5] From several initial conditions drawn from the torus, TM integrates the equations of motion for M time-steps and computes the toy angles after each time-step. These must satisfy

$$\boldsymbol{\theta}(0) + \boldsymbol{\Omega} t_i = \boldsymbol{\theta}^{\rm T}(t_i) + 2\sum_{\mathbf{n}>0} \frac{\partial S_{\mathbf{n}}(\mathbf{J})}{\partial \mathbf{J}} \sin[\mathbf{n}\cdot\boldsymbol{\theta}^{\rm T}(t_i)] \qquad \text{(A5)}$$

for $i = 1, \dots, M$. Hence each integration yields $3M$ equations in which $\boldsymbol{\theta}(0)$, $\boldsymbol{\Omega}$, and $\partial S_{\mathbf{n}}(\mathbf{J})/\partial\mathbf{J}$ appear as unknowns. The equations are linear in the unknowns and for $M \gg 1$ the number of available equations increases much faster than the number of unknowns. We solve these equations in a least-squares sense. We refer to this process as an "angle fit".

The various steps just described are each implemented by an instance of a distinct class:

- The $\mathbf{n}$ and corresponding $S_{\mathbf{n}}$, are stored and manipulated by an instance `Sn` of the class `GenPar`.
- The $\mathbf{n}$ and the corresponding values $\partial S_{\mathbf{n}}/\partial\mathbf{J}$ are stored and manipulated by an instance `A` of the class `AngPar`.
- An instance `GF` of the class `GenFnc` handles the mapping from $(\mathbf{J}, \boldsymbol{\theta}^{\rm T})$ to $(\mathbf{J}^{\rm T}, \boldsymbol{\theta}^{\rm T})$ for a fixed value of $\mathbf{J}$. It also finds the derivatives $(\partial\mathbf{J}^{\rm T}/\partial\boldsymbol{\theta}^{\rm T})_{\mathbf{J}}$. `Sn` is a member of `GF`.
- An instance `GFF` of the class `GenFncFit` performs the same tasks as `GF` but focuses on the transformations required when performing an action fit, where we have a known, regular grid of points in $\boldsymbol{\theta}^{\rm T}$ at which the transformation has to be calculated many times. `GFF` also provides the derivatives $\partial\mathbf{J}^{\rm T}/\partial S_{\mathbf{n}}$.
- An instance `AM` of the class `AngMap` handles the mapping between $(\mathbf{J}, \boldsymbol{\theta})$ and $(\mathbf{J}, \boldsymbol{\theta}^{\rm T})$ (eq. A2). This mapping works in either direction, and the values $\partial\theta_i/\partial\theta_j^T$ can also be found. `A` is a member of `AM`.
- An instance `T` of the class `Torus` has `GF` and `AM` as members.

A2 The Toy Potential

The job of the toy potential is to provide analytic angle-action coordinates $(\boldsymbol{\theta}^{\rm T}, \mathbf{J}^{\rm T})$. Candidates include the harmonic oscillator potential, Hénon's isochrone (McGill & Binney 1990), and a Stäckel potential (Laakso & Kaasalainen 2013). TM uses the generalised effective isochrone potential

$$\Phi^T_{\rm eff}(r,\vartheta) = \frac{-GM}{b + \sqrt{b^2 + (r-r_0)^2}} + \frac{L_z^2}{2\left[(r-r_0)\sin\vartheta\right]^2}, \qquad \text{(A6)}$$

where ϑ is the usual spherical polar coordinate (not to be confused with a dynamical angle coordinates), and M, b, L_z and r_0 are the potential's parameters. Since we are modelling the motion in the (R,z) plane, we can treat L_z as a parameter of the toy effective potential that is distinct from the actual z-component of angular momentum J_ϕ. Appendix A2 of McGill & Binney (1990) gives the formulae needed to compute (R, z, v_R, v_z) from $(\theta_r, \theta_z, J_r, J_z)$ with the provisos: (i) for $\cos\theta$ in McGill & Binney read $\sin\vartheta$, (ii) for J_ϑ read J_z, and (iii) for r read $r - r_0$. Expressions for $\theta^{\rm T}_\phi$ are given below. If $\theta^{\rm T}_\phi$ is to increase in the same sense as θ_ϕ, L_z and J_ϕ must have the same sign.

[5] Angle fitting will shortly be shortly changed (Vasiliev et al., in preparation) to the algorithm introduced by Binney & Kumar (1993) as modified by Laakso & Kaasalainen (2013).

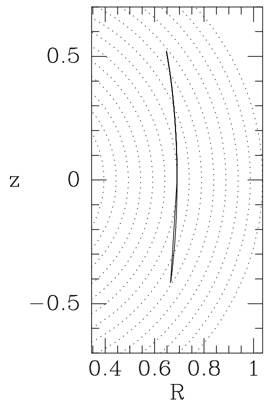

Figure A3. A target shell orbit in the (R, z) plane (solid line) and shell orbits in the toy potential (which are all on circles centred on the origin, dotted lines). The generating function of Section A1 is incapable of mapping the toy orbits on to the target orbit.

The parameters of Φ^{T} actually used by TM are L_z, r_0,

$$\gamma \equiv \sqrt{GM} \text{ and } \beta \equiv \sqrt{b}. \tag{A7}$$

This parametrisation ensures that unphysical negative values of M and b cannot be inadvertently chosen during torus fitting.

The class `ToyMap` is a base class for any mapping between $(\boldsymbol{\theta}^{\mathrm{T}}, \mathbf{J}^{\mathrm{T}})$ and $(\mathbf{x}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}})$. Having this base class facilitates future use of the tori of a three-dimensional harmonic oscillator, but currently the only class of this kind that is implemented is `ToyIsochrone`, which specifies the mapping in case of the generalised effective isochrone potential. It also specifies the calculation of the partial derivatives of $(\mathbf{x}^{\mathrm{T}}, \mathbf{v}^{\mathrm{T}})$ with respect to $\mathbf{J}^{\mathrm{T}}$, $\boldsymbol{\theta}^{\mathrm{T}}$ and the parameters of the toy potential.

A3 Point transformation

Unless J_r/J_z is small, a torus of Φ^{T} can be mapped onto a torus of the Galactic potential using only the generating function (A3). Unfortunately for small values of J_r/J_z a more sophisticated treatment is necessary.

As $J_r \to 0$ eq. (A4) implies negative values of J_r^{T} unless all the $S_{\mathbf{n}} \to 0$, so the transformation generated by $S(\boldsymbol{\theta}^{\mathrm{T}}, \mathbf{J})$ tends to the identity. However, while the shell orbits $J_r^{\mathrm{T}} = 0$ of the isochrone potential lie on spheres, the shell orbits $J_r = 0$ of a flattened Galactic potential do not (Figure A3). Consequently, along a shell orbit in a Galactic potential r oscillates, so $p_r \neq 0$. But if $p_r \neq 0$, then $J_r^{\mathrm{T}} \neq 0$ also, because p_r is non-zero only on eccentric orbits. It follows that the generating function (A3) is insufficiently general to yield image tori that have small radial actions.

Kaasalainen & Binney (1994b) solved this problem by combining the canonical transformation generated by eqn. (A3) with a point transformation $\mathbf{x} \to \mathbf{x}^{\mathrm{T}}(\mathbf{x})$ of the form

$$(r, \vartheta) \to (r^{\mathrm{T}}, \vartheta^{\mathrm{T}}) = (\xi(\vartheta) r,\ \eta(r)\zeta(\vartheta)). \tag{A8}$$

Here ξ, η and ζ are functions to be determined such that the shell orbit of Φ^{T} with the given (J_z, J_ϕ) is mapped into the corresponding shell orbit of the Galactic potential. Qualitatively, we want $\eta \sim 1$ and $\zeta \sim \vartheta$; ξ is chosen such that the shell orbit of the target potential is $r^{\mathrm{T}} = a$, where a is the radius of the toy shell orbit. Since the radius and angular extent of the shell orbit in Φ^{T} depends on the parameters of Φ^{T}, the latter must be fixed before the point transformation is determined, and if it changed by the Levenberg-Marquardt routine as it minimises δH, the shell orbits will no longer be mapped into each other and tori $J_r = 0$ will cease to be accessible. We set $r_0 = 0$, $L_z = J_\phi$, and (rather arbitrarily) $\beta = \sqrt{3}$. γ is chosen such that in Φ^{T} the orbit has $r = 1$ and in the formulae below a can be set to unity.

Numerical integration of the Galactic shell orbit yields the radius of this orbit, $r_{\mathrm{s}}(\vartheta)$ and we immediately have

$$\xi(\vartheta) \equiv \frac{a}{r_{\mathrm{s}}(\vartheta)} \tag{A9}$$

because then on the orbit $r^{\mathrm{T}} = \xi r_{\mathrm{s}} = a$ as required. The upper curve in the right-hand panel of Fig. A4 shows $\xi(\vartheta)$ for the shell orbit plotted in Figure A3.

The generating function of our point transformation is

$$S(r, \vartheta, p_r^{\mathrm{T}}, p_\vartheta^{\mathrm{T}}) = \xi r p_r^{\mathrm{T}} + \eta\zeta p_\vartheta^{\mathrm{T}}, \tag{A10}$$

so

$$p_r = \xi p_r^{\mathrm{T}} + \frac{\mathrm{d}\eta}{\mathrm{d}r}\zeta p_\vartheta^{\mathrm{T}} \quad ; \quad p_\vartheta = \frac{\mathrm{d}\xi}{\mathrm{d}\vartheta} r p_r^{\mathrm{T}} + \eta\frac{\mathrm{d}\zeta}{\mathrm{d}\vartheta} p_\vartheta^{\mathrm{T}}. \tag{A11}$$

From our integration of the target shell orbit we know $p_r(\vartheta)$ and $p_\vartheta(\vartheta)$ along this orbit. On the toy potential's shell orbit $p_r^T = 0$ and

$$p_\vartheta^{\mathrm{T}}(\vartheta^{\mathrm{T}}) = \sqrt{L^2 - \frac{L_z^2}{\sin^2\vartheta^{\mathrm{T}}}}. \tag{A12}$$

Finally we use the known relation $r_{\mathrm{s}}(\vartheta)$ on the target shell orbit to treat η as a function of ϑ. Then equations (A11) yield coupled o.d.e.s for η and ζ

$$\frac{\mathrm{d}\eta}{\mathrm{d}\vartheta} = \frac{p_r}{\zeta p_\vartheta^{\mathrm{T}}}\frac{\mathrm{d}r_{\mathrm{s}}}{\mathrm{d}\vartheta} \quad ; \quad \frac{\mathrm{d}\zeta}{\mathrm{d}\vartheta} = \frac{p_\vartheta}{\eta p_\vartheta^{\mathrm{T}}}. \tag{A13}$$

Figure A4 shows the functions η and ζ that are obtained by integrating these equations. [6]

So far we have obtained the functions ξ, η and ζ only in the ranges of r and ϑ covered by the closed orbit. The functions are extended beyond this range by adding single points at the ends of the required ranges, and then fitting a sum of Chebychev polynomials to both these points and the points returned by the numerical integrations. The fitting algorithm does not require the polynomial to pass through the given points. Rather a function is constructed from the points by fitting a quadratic to each set of three points. Then the Chebychev series is fitted in a least squares sense to this function – the coefficients of the Chebychev polynomials are found from integrals of the products of the relevant Chebychev polynomial and the quadratics and the weight function. These integrals are done analytically. By relieving the polynomial of the obligation to pass through the data points, this techniques avoids the danger that the polynomial makes wild excursions between data points.

Figure A5 shows the new coordinates, while Figure A6

[6] In practice TM changes these o.d.e.s to ones with a new independent variable ψ before integrating the equations, where $\vartheta = \vartheta_{\max}\sin\psi$, with $\vartheta_{\max}$ the largest value of ϑ attained on the closed orbit.

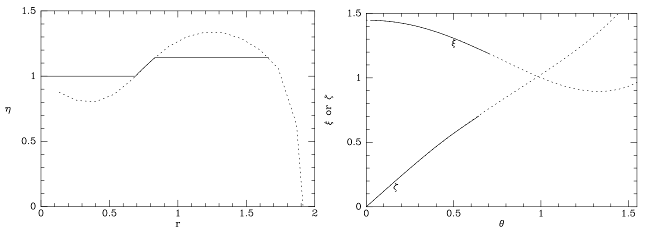

Figure A4. The functions $\eta(r)$ (left) and $\xi(\theta)$, $\zeta(\theta)$ (right) that define the new (r, θ) coordinates. The full curves show results obtained by integrating the defining equations and then adding extra points to fill out the range, while the dashed curves show Chebychev-polynomial L^2 fits to the points defined by these curves.

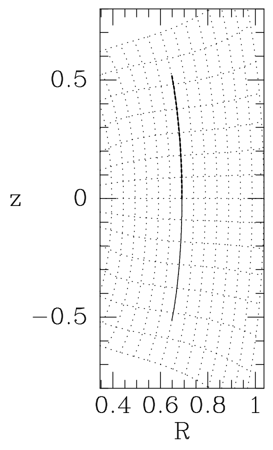

Figure A5. The dotted lines show the (r, ϑ) coordinate system constructed such that a shell orbit in the target potential coincides with a curve of constant r.

shows the fits obtained to the target shell orbit in the (r, p_r) and (r, p_θ) planes. The parameters of the point transform are the coefficients of the Chebychev polynomial.

The class `PoiTra` is the base class for any point transformation $(\mathbf{x}^{\rm T}, \mathbf{v}^{\rm T}) \leftrightarrow (\mathbf{x}, \mathbf{v})$. It uses cylindrical polar coordinates. If no point transformation is required, TM uses an instance of the derived class `PoiNone`. If J_r/J_z is small so a point transformation is required, TM creates an instance of the derived class `PoiClosedOrbit`.

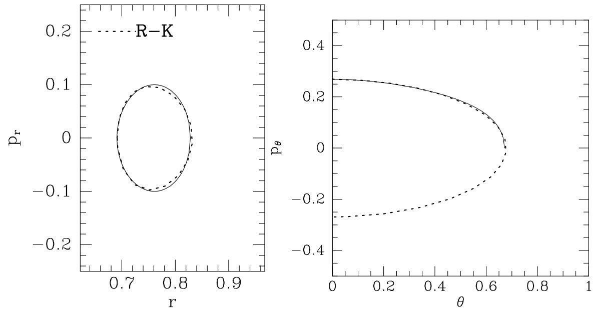

Figure A6. Fits in the two position-momentum planes: dotted curves numerical integration, full curves images of the toy closed orbit under the point transformation.

APPENDIX B: EVALUATING θ_ϕ^T IN THE ISOCHRONE POTENTIAL

Evaluating θ_ϕ^T in the generalised effective isochrone potential potential (eq. A6) is not, perhaps, as straightforward as might appear from the standard texts. We therefore give the details here.

B1 Non-planar orbits

First let us consider the usual case where $J_\vartheta \neq 0$ (with ϑ the usual polar angle). We have found the values of $\theta_r^T, \theta_\vartheta^T$ and $\boldsymbol{J}$ (or equivalently $r, \vartheta, p_r, p_\vartheta$) in the generalised effective isochrone potential potential (eq. A6). For the standard isochrone potential ($r_0 = 0$, no term in L_z), we have from equations (3.229) and (3.232) of Binney & Tremaine (2008) that

$$\theta_\phi = \phi - u + \mathrm{sgn}(L_z)\theta_\vartheta \tag{B1}$$

with ϕ the usual azimuthal angle and

$$\sin u \equiv \cot i \cot \vartheta, \tag{B2}$$

with i the inclination given by $i = \arccos(L_z/L)$.

Care needs to be taken when finding i and u. i is found in the toy Hamiltonian, so L_z is the value used as a parameter in eqn. (A6), and $L = J_\vartheta^T + |L_z|$.

Equation (B2) only provides the value of $\sin u$, while u takes values throughout the range $(0, 2\pi)$, so this is insufficient information. We need to take note of the fact that $u = \phi - \Omega$, where in this case Ω is the longitude of the ascending node, i.e., the line on which an orbit crosses the (x, y) plane with $\dot\vartheta < 0$, (again, in the spherical toy potential, in which the orbital plane does not precess). Assume that $L_z > 0$ and follow the star up from the ascending node (where $u = 0$) until ϑ stops decreasing. Then it is geometrically clear that $\vartheta = \pi/2 - i$, so $\sin u = 1$ and $u = \pi/2$. Now ϑ starts to increase and in order to ensure that u continues to increase we need to shift to the other solution of eqn. (B2), $u = \pi - \arcsin(\cot i \cot \vartheta)$.

In summary for $L_z > 0$ we take

$$u = \begin{cases} \arcsin(\cot i \cot \vartheta) & \text{for } \dot\vartheta < 0, \\ \pi - \arcsin(\cot i \cot \vartheta) & \text{otherwise.} \end{cases} \tag{B3}$$

With this definition, $-\pi/2 < u < 3\pi/2$.

When $L_z < 0$, $i > \pi/2$ and u decreases from zero as the star rises through the ascending node, just as ϕ decreases, and eq (B2) still applies.

B2 Planar orbits

When $J_\vartheta = 0$, the value of θ_ϑ is undefined, so eq. B3 is essentially meaningless. We have to return to the full expression for the generating function, eqn. (3.231) of Binney & Tremaine (2008), fixing $J_2 = |J_1| \equiv |L_z|$, and $\vartheta = \pi/2$:

$$S = \phi L_z + \int_{r_{\rm min}}^{r} \mathrm{d}r\, \epsilon_r \sqrt{2H(L_z, J_r) - 2\Phi(r) - \frac{L_z^2}{r^2}}, \tag{B4}$$

and find $\theta_\phi = \partial S/\partial L_z$. Happily, this is very similar to the equations that have already been solved to find θ_ϑ in the usual case, so we know that we can solve for θ_ϕ:

$$\theta_\phi = \phi + \frac{\Omega_\phi}{\Omega_r}\theta_r - \mathrm{sgn}(L_z)\left[\tan^{-1}\left(\sqrt{\frac{1+e}{1-e}}\tan(\tfrac{1}{2}\eta)\right)\right.$$
$$\left. + \zeta\tan^{-1}\left(\sqrt{\frac{1+e+2b/c}{1-e+2b/c}}\tan(\tfrac{1}{2}\eta)\right)\right], \qquad \text{(B5)}$$

where e, η and c are as defined in eqn. (3.240) of Binney & Tremaine (2008) and

$$\zeta = \frac{1}{\sqrt{1+4GMb/L_z^2}}. \qquad \text{(B6)}$$

APPENDIX C: GRAVITATIONAL POTENTIALS

C1 Multi and predefined potentials

The class `MultiPotential` allows one to build a new potential by adding several previously defined potentials. For example, one can construct the potential of a disc galaxy by adding the potentials of a bulge, a disc and a dark halo.

To create a potential that is the sum of a `MiyamotoNagaiPotential` and a `LogPotential`, one writes

```
Potential **PotList  = new Potential*[2];
PotList[0] =  new MiyamotoNagaiPotential(M,a,b);
PotList[1] =  new LogPotential(V0,q,Rc);
Potential *Phi = new MultiPotential(PotList,2);
```

C2 User defined potentials

To create a new potential class, one has to write two new files (the header file and the main code). As an example, we give below the header and code files for the Kepler potential (which is identical to `IsochronePotential` with b=0).

```
// file KeplerPotential.h
#include <cmath>
#include "Potential.h"

class KeplerPotential : public Potential {
  double GM;
public:
  KeplerPotential     (const double);
  ~KeplerPotential    () {;}
  double operator()   (const double,
                       const double) const;
  double operator()   (const double,
                       double&, double&) const;
  Frequencies KapNuOm (const double) const;
};

// file KeplerPotential.cc
KeplerPotential::KeplerPotential(const double M) {
  GM = Units::G * M;
}
double KeplerPotential::operator(const double R,
                                 const double z) const {
  return -GM / sqrt(R*R+z*z);
}
double KeplerPotential::operator(const double R,
                                 const double z,
                                 double &dPdR,
                                 double &dPdz) const {
  double ir2 = 1./(R*R+z*z), ir = sqrt(r2);
  double dPdr = GM * ir2;
  dPdR = dPdr * R*ir;
  dPdz = dPdr * z*ir;
  return -GM * ir;
}

Frequencies KeplerPotential::KapNuOm(const double R) {
  Frequencies epicycle;
  double allfreqs = sqrt(GM/powf(R,1.5));
  // In Kepler potential, all epicycle freqs equal
  epicycle[0] = allfreqs;
  epicycle[1] = allfreqs;
  epicycle[2] = allfreqs;
  return epicycle;
}
```

APPENDIX D: USER DEFINED DISTRIBUTION FUNCTIONS

To create a new class of DF one adds code to the file `DF.h`. The following example defines a DF that is constant for $J_r < 1$, $J_z < 1$ and $0 < J_\phi < J_{\phi,\max}$, and zero elsewhere.

```
  class Simple_DF : public DF{
  public:
    double JpMax;
    int setup(istream&);
    int setup_full(istream&);
    int NumberofParameters(){return 1;}
    void Parameters(double*);
    double df(Potential*,Actions);
  };
  inline int Simple_DF::setup(istream &ifile){
    char type1; ifile >> type1;
    if(type1!='S') {
      cerr << "improper input file\n";
      return 0;
    }
    ifile >> JpMax;
    return 1;
  }
  inline void Simple_DF::Parameters
          (double* output){
    output[0] = JpMax;
  }
  inline double Simple_DF::df
          (Potential* Phi,Actions J){
    if(J[0]>1 || J[1]>1) return 0;
    if(J[2]<0 || J[2]>JpMax) return 0;
    return 1/JpMax;
  }
```

The line

```
  DF *sdf=set_DF(ifile);
```

will initialise an instance `sdf` of `Simple_DF` with `Jpmax = 2` provided (i) the contents of `ifile` are

```
  S
  2
```

and (ii) the following code has been added to the list of possibilities in the definition of `DF::set_DF` (which is given at the end of the file `DF.h`):

```
if(type1=='S'){
  tmpdf = new Simple_DF;
  tmpdf->setup(ifile);
  return tmpdf;
}
```