COMPARISON OF GEOMETRIC INTEGRATOR METHODS FOR HAMILTON SYSTEMS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Mathematics

by
Pınar İNECİ

March 2009
İZMİR

We approve the thesis of **Pınar İNECİ**

Assoc. Prof. Dr. Gamze TANOĞLU
Supervisor

Prof. Dr. Oktay PASHAEV
Committee Member

Assist. Prof. Dr. Gürcan ARAL
Committee Member

31 March 2009

Prof. Dr. Oğuz YILMAZ
Head of the Mathematics Department

Prof. Dr. Hasan BÖKE
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGEMENTS

This thesis is the consequence of a three-year study evolved by the contribution of many people and now I would like to express my gratitude to all the people supporting me from all the aspects for the period of my thesis.

Firstly, I would like to thank and express my deepest gratitude to Assoc. Prof. Dr. Gamze TANOĞLU, my advisor, for her help, guidance, understanding, encouragement and patience during my studies and preparation of this thesis. And I would like to thank to TÜBİTAK for its support.

I also would like to express my special thanks to Engin İZGİ for his support, understanding and love.

Last, thanks to Barış ÇİÇEK, Hakan GÜNDÜZ and Duygu DEMİR for their supports. And finally I am also grateful to my family for their confidence to me and for their endless supports.

ABSTRACT

COMPARISON OF GEOMETRIC INTEGRATOR METHODS FOR HAMILTON SYSTEMS

Geometric numerical integration is relatively new area of numerical analysis. The aim of a series numerical methods is to preserve some geometric properties of the flow of a differential equation such as symplecticity or reversibility. In this thesis, we illustrate the effectiveness of geometric integration methods. For this purpose symplectic Euler method, adjoint of symplectic Euler method, midpoint rule, Störmer-Verlet method and higher order methods obtained by composition of midpoint or Störmer-Verlet method are considered as geometric integration methods. Whereas explicit Euler, implicit Euler, trapezoidal rule, classic Runge-Kutta methods are chosen as non-geometric integration methods. Both geometric and non-geometric integration methods are applied to the Kepler problem which has three conserved quantities: energy, angular momentum and the Runge-Lenz vector, in order to determine which those quantities are preserved better by these methods.

ÖZET

HAMİLTON SİSTEMLER İÇİN GEOMETRİK ENTEGRASYON YÖNTEMLERİNİN KARŞILAŞTIRILMASI

Geometrik entegrasyon nümerik analizin nispeten yeni alanlarından biridir. Birçok sayısal metodun amacı diferansiyel denklemlerin çözümünün simplektiklik ya da tersine çevrilebilirlik gibi bazı geometrik özelliklerini korumaktır. Bu tezde geometrik entegrasyon yöntemlerinin etkisini ortaya koyduk. Bu amaç doğrultusunda geometrik entegrasyon yöntemleri olarak simplektik Euler metodu, simplektik Euler metodunun adjonti, Störmer-Verlet metod ve midpoint ya da Störmer-Verlet metodun bileşkesi ile elde edilen yüksek mertebeden metodları kullandık. Aynı zamanda Explicit Euler, Implicit Euler, trapezoidal rule ve klasik runge-kutta yöntemleri geometrik olmayan entegrasyon yöntemleri olarak kullanıldı. Enerji, açısal momentum ve Runge-Lenz vektörü gibi üç tane korunan, geometrik özelliği olan kepler problemine, bu özelliklerin hangi yöntemler tarafından daha iyi korunduğunu saptamak için hem geometrik hem de geometrik olmayan entegrasyon yöntemleri uygulandı.

TABLE OF CONTENS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The topic of this thesis is geometric structure-preserving numerical integration method for ordinary differential equations. We will concentrate mainly on Hamiltonian equations and numerical methods that preserve geometric structures of them.

During the past decade there has been an increasing interest in studying numerical methods that preserve certain properties of some differential equations (Budd and Piggott 2000). The reason is that some physical systems possess conserved quantities, and that the solutions of the systems also should contain these invariants. Typical examples are the symplectic structure in a Hamiltonian system, the energy in a conservative mechanical system and the angular momentum of a rotating rigid body in space. Classical numerical methods normally fail to preserve such as the above mentioned quantities (Budd and Piggott 2000).

Frequently we need to integrate numerically a system of ordinary differential equations (ODEs) having some first integrals or admitting some conservation laws. In what follows we will consider exclusively the ODE case, but our method can also be applied to the case of partial differential equations. Many examples of systems of ODEs having first integrals can be found in classical mechanics, where Hamiltonian equations are of this type, always having the Hamiltonian as a first integral.

In this thesis we mainly concern with symplectic geometric integration methods. We will give the definition of symplectic maps, and call any numerical scheme which induces a symplectic map as the *symplectic numerical method*.

For numerical implementation we choose the Kepler problem. Conserved quantities of the Kepler problem are the energy, the angular momentum and the Runge-Lenz vector. We applied both geometric and non-geometric integration methods to the Kepler problem. We observed that symplectic methods were most successful. We illustrate the effectiveness of geometric integration methods, namely symplectic Euler, adjoint of symplectic Euler, midpoint rule and Störmer-

Verlet method which is composition of symplectic Euler and its adjoint. In this study third and fifth order methods are constructed by composition techniques by considering midpoint rule and Störmer-Verlet method as base method. For comparison reason non-symplectic methods are also chosen.

The outline of this thesis as follows: After giving some preliminaries in Chapter 2, geometric numerical methods and non-geometric methods that we use in our computations are introduced in Chapter 3. And we discuss some geometric properties, first integrals of the differential equations and philosophy of geometric integration in Chapter 4. Finally; both geometric and non-geometric methods, that we introduce in Chapter 4, are applied to the Kepler problem in Chapter 5, in order to see advantages and disadvantages of geometric integration.

CHAPTER 2

PRELIMINARIES

This chapter consist of some preliminary information about geometric integration method. We introduce the exact flow of an ODE and general properties of exact flows. We also give the definition of an integrator and some properties of integrators. Also a brief discussion is given on the geometric integration method idea. Finally we introduce the Hamiltonian systems.

The motion is described by differential equations, which are derived from the laws of physics. These equations contain within them not just a statement of the current acceleration experienced by the object(s), but all the physical laws relevant to the particular situation. Finding these laws and their consequences for the motion has been a major part of physics since the time of Newton (Stuart and Humphries 1996). For example, the equations tell us the space in which the system evolves (its phase space, which may be ordinary Euclidean space or a curved space such as a sphere); any symmetries of the motion, such as left-right or forward-backward symmetries of a pendulum; and any special quantities such as energy, which for a pendulum is either conserved. Most importantly, the laws describe how all motion starting close to the actual one are constrained in relation to each other. These laws are known as symplecticity and volume preservation (Stuart and Humphries 1996).

Standard methods for simulating motion, called numerical integrators, take an initial condition and move the objects in direction specified by the differential equations. They completely ignore all of the above hidden physical laws contained within the equations. Since about 1990, new methods have been developed, called geometric integrators, which obey these extra laws (Budd and Iserles 1999). Since the methods are physically natural, we can hope that the results will be extremely reliable, especially for long-time simulations. Before we list you advantages of the method, we mention three disadvantages:

- The hidden physical law usually has to be known if the integrator is going to obey it. For example to preserve energy, the energy must be known.

- Because we are asking something more in this method, it may turn out to be computationally more expensive then a standard method. Amazingly (because the laws are so natural) sometimes it's actually much cheaper.
- Many systems have several hidden laws, but the methods currently known preserve one of them but not all simultaneously (Quispel and Dyt 1997).

Now the advantages of the method:

- Simulations can be run for enormously long times, because there are no spurious non-physical effects, such as dissipation of energy in a conservative system.
- By studying structure of the equations, very simple, fast, and reliable geometric integrators can often be found;
- In some situations, results can be guaranteed to be *qualitatively* correct, even when the motion is chaotic.
- For some systems, for short, medium and long times even the actual quantitative errors are much smaller than in standard methods (Quispel and Dyt 1997).

2.1. Introduction to ODEs

Coming from classical mechanics ordinary differential equations (ODEs) were developed to model mechanical systems. Most dynamical systems -physical, biological, engineering- are often conveniently expressed in the form of differential equations.

Let $\mathbb{R}$ be the set of all real numbers and I be an open interval on the real line $\mathbb{R}$, that is, $I = \{t : t \in \mathbb{R}, r_1 < t < r_2\}$, where r_1, r_2 are any two fixed points in $\mathbb{R}$. Also, let $\mathbb{R}^n$ denote the real n-dimensional Euclidean space with elements $x = (x_1, x_2, ..., x_n)$, and let $\mathbb{R}^{n+1}$ be the space of elements of (n+1)-tuple $(t, x_1, x_2, ..., x_n)$ or (t, x). Let B be a domain, i.e, an open-connected set in $\mathbb{R}^{n+1}$, and $C[B, \mathbb{R}^n]$ be a class of functions defined and continuous on B.

An ordinary differential equation of the n-th order and of the form

$$F(t, u, u', u'', ..., u^{(n)}) = 0 \tag{2.1}$$

where $u^{(n)}$ is the n-th derivative of the unknown function u with respect to t and F is defined in some subset of $\mathbb{R}^{n+2}$, express a relation between the (n+2)-variables $t, u, u', u'', ..., u^{(n)}$ (Rao 1979).

System of Differential Equations

We shall consider a system of differential equations of the form

$u'_1 = f_1(t, u_1, ..., u_n)$

$u'_2 = f_2(t, u_1, ..., u_n)$

.

.

.

$u'_n = f_n(t, u_1, ..., u_n)$

In vector notation, these equations can be written as

$$u' = f(t, u) \tag{2.2}$$

where $u = (u_1, u_2, ..., u_n)$, $f = (f_1, f_2, ..., f_n)$ are vectors in $\mathbb{R}^n$. We shall assume that $f \in C[B, \mathbb{R}^n]$. With an initial condition

$$u(t_0) = u_0 \tag{2.3}$$

where $u_0 = (u_{10}, u_{20}, ..., u_{n0})$ is a vector in $\mathbb{R}^n$. Relations (2.2) and (2.3) constitute a so-called initial value problem (IVP) (Rao 1979). A set of n-functions $\phi_1, \phi_2, ..., \phi_n$ defined on I is said to be solution of IVP if, for $t \in I$,

- $\phi'_1(t), ..., \phi'_n(t)$ exist,
- *the point* $(t, \phi_1(t), ..., \phi_n(t))$ remains in B,
- $\phi'_i(t) = f_i(t, \phi_1(t), ..., \phi_n(t)), \quad i = 1, 2, ..., n$
- $\phi_i(t_0) = \phi_{i0}, \quad i = 1, 2, ..., n$

2.2. The Exact Flow of an ODE

We first define the exact flow (or solution) of an ordinary differential equation and discuss what properties one would like an integrator to have. Let u(t) be the exact solution of the system of ordinary equations (ODEs)

$$\frac{du}{dt} = f(u(t)), \quad u(t_0) = u_0, \quad u(t) \in \mathbb{R}^n \tag{2.4}$$

the exact flow φ_h is defined by

$$u(t+h) = \varphi_h(u(t)) \quad \forall t,\ h \in \mathbb{R}$$

For each fixed time step h, φ is a map from the phase space to itself,i.e, $\varphi_h : \mathbb{R}^n \to \mathbb{R}^n$.

2.3. Hamiltonian Systems

In this section we give the definition of the Hamiltonian system.

Definition 2.1 *Suppose that H(q,p) is a smooth function of its arguments for q and $p \in \mathbb{R}^n$. Then the dynamical system:*

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \tag{2.5}$$

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} \tag{2.6}$$

(i = 1,2,...,n) is called a Hamiltonian system and H is the Hamiltonian function (or just the Hamiltonian) of the system. Equations (2.5) and (2.6) called Hamiltons equations.

Definition 2.2 *The number of degrees of freedom of a Hamiltonian system is the number of (q_i, p_i) pairs in Hamilton's equations, i.e. the value of n.*

In mechanics, the vector q represents the generalized coordinates of the components of the system (positions, angles, etc.), while p is a set of generalized momenta. Note that the Hamiltonian function is a constant of the motion:

$$\begin{aligned} \frac{dH}{dt} &= \sum_{i=1}^{n} \frac{\partial H}{\partial q_i}\dot{q}_i + \frac{\partial H}{\partial p_i}\dot{p}_i \\ &= \sum_{i=1}^{n} \frac{\partial H}{\partial q_i}\frac{\partial H}{\partial p_i} + \frac{\partial H}{\partial p_i}(-\frac{\partial H}{\partial q_i}) = 0 \end{aligned}$$

Example 2.1 *A harmonic oscillator is a mass-spring system with potential energy $\frac{1}{2}kq^2$, where q is the displacement of the spring from equilibrium. For simple systems like this one, in which the potential energy simply depends on the position, the Hamiltonian is just the total energy:*

$$H(q,p) = \frac{1}{2}kq^2 + \frac{p^2}{2m} \tag{2.7}$$

where k and m are positive constants and p is the momentum. Because H is a constant, the orbits are just the family of ellipses,

$$\frac{1}{2}kq^2 + \frac{p^2}{2m} = E \tag{2.8}$$

The value of E is fixed by the initial conditions. Different values of E correspond to ellipses of different size. If we are interested in the equations of motion, we can recover them from Hamiltons equations:

$$\dot{q} = \frac{\partial H}{\partial p} = p/m \tag{2.9}$$

$$\dot{p} = -\frac{\partial H}{\partial q} = -kq \tag{2.10}$$

CHAPTER 3

NUMERICAL METHODS

In this chapter, we introduce numerical methods approximating the solution of initial value problem for ordinary differential equation.

$$\frac{d}{dt}y = f(y), \qquad y(t_0) = y_0 \in \mathbb{R}^k. \tag{3.1}$$

Here $y(t)$ represents the solution at a particular time t; $y = y(t)$ thus defines a parameterized trajectory in $\mathbb{R}^k$. We assume that trajectories are defined for all initial values $y_0 \in \mathbb{R}^n$ and for all times $t \geq t_0$. For simplicity, we typically take $t_0 = 0$. One also often uses the notation $y(t; y_0)$ to distinguish the trajectory for a given initial value y_0. Define a mapping, or rather a one-parametric family of mappings, $\{\phi_t\}_{t\geq 0}$, which take initial data to later points along trajectories, i.e.

$$\phi_t(y_0) = y(t; y_0), \qquad y_0 \in \mathbb{R}^k$$

A numerical method for solving ordinary differential equations is a mapping Φ_h defined on the phase space that approximates the time-h flow φ_h; if $\Phi_h(y) = \varphi_h(y) + O(h^{r+1})$. The numerical approximation at time $t = nh$ is obtained by $y_n = \Phi_h(y_{n-1})$ (Budd and Iserles 1999).

3.1. One Step Method

We will primarily concerned with the generalized one-step method. By iterating the flow map, we know that we obtain a series of snapshots of the true trajectory

$$y_0, \phi_{\Delta t}(y_0), \phi_{\Delta t} \circ \phi_{\Delta t}(y_0), ...$$

or, compactly, $\{\phi^n_{\Delta t}(y_0)\}^{\infty}_{n=0}$, where the composition power $\phi^n_{\Delta t}$ is the identity if $n = 0$ and is otherwise the n-fold composition of $\phi_{\Delta t}$ with itself. For a one-step method, the approximating trajectory can be viewed as the iteration of another mapping $\psi_{\Delta t} = \mathbb{R}^k \to \mathbb{R}^k$ of the underlying space, so that

$$y^n = \psi^n_{\Delta t}(y_0). \tag{3.2}$$

The mapping $\psi_{\Delta t}$ is generally nonlinear, will depend on the function f and/or derivatives, and may be quite complicated, because one-step method generate a mapping of the phase space (Leimkuhler and Reich 2004).

3.2. Derivation of One-step Method

One step method can be derived in various ways. One way is first to integrate both sides of $\frac{d}{dt}y = f(y)$ on a small interval $[t, t + \Delta t]$, obtaining

$$y(t + \Delta t) - y(t) = \int_0^{\Delta t} f(y(t + \tau))d\tau.$$

The right-hand side can then be replaced by a suitable quadrature formula resulting in an approximation of the form

$$y(t + \Delta t) \approx y(t) + \sum_{i=1}^{s} b_i f(y(t + \tau_i))$$

for an appropriate set of weights $\{b_i\}$ and quadrature points $\{\tau_i\}$ (Leimkuhler and Reich 2004).

3.3. Elementary One Step Method

In this section we will introduce elementary one-step methods.

Explicit Euler Method:

$$y_{n+1} = y_n + \Delta t f(y_n)$$

the quadrature rule used is just

$$\int_0^{\Delta t} f(y(t + \tau))d\tau = \Delta t f(y(t)) + O(\Delta t^2)$$

Implicit Euler Method:

$$y_{n+1} = y_n + \Delta t f(y_{n+1})$$

Trapezoidal Rule:

$$y_{n+1} = y_n + \frac{\Delta t}{2}[f(y_n) + f(y_{n+1})]$$

the trapezoidal rule is based on

$$\int_0^{\Delta t} f(y(t + \tau))d\tau = \frac{1}{2}\Delta t[f(y(t)) + f(y(t + \Delta t))] + O(\Delta t^3)$$

Implicit Midpoint Method:

$$y_{n+1} = y_n + \Delta t f\Big(\frac{y_n + y_{n+1}}{2}\Big)$$

the quadrature rule is defined as

$$\int_0^{\Delta t} f(y(t+\tau))d\tau = \Delta t f\Big(\frac{y(t) + y(t+\Delta t)}{2}\Big) + O(\Delta t^3)$$

3.4. Runge-Kutta Methods

All of the methods discussed so far are special cases of Runge-Kutta methods. We treat non-autonomous systems of first order ordinary differential equations

$$\dot{y} \;=\; f(t,y) \qquad y(t_0) = y_0 \tag{3.3}$$

Let b_i, a_{ij} $(i,j = 1,...,s)$ be real numbers and let $c_i = \sum_{j=1}^{s} a_{ij}$. An *$s-stage\ Runge-Kutta\ method$* is given by

$$\begin{aligned} k_i &= f(t_0 + c_i h, y_0 + h\sum_{j=1}^{s} a_{ij}k_j), \quad i = 1,...,s && (3.4)\\ y_1 &= y_0 + h\sum_{i=1}^{s} b_i k_i && (3.5)\end{aligned}$$

Here we allow a full matrix (a_{ij}) of non-zero coefficients. In this case, the slopers k_i can no longer be computed explicitly, and even do not necessarily exist(Hairer, et al. 2001).

In Butcher tableau the coefficients are usually displayed as follows:

$$\begin{array}{c|cccc} c_1 & a_{11} & . & . & . \; a_{1s} \\ . & . & & & . \\ . & . & & & . \\ . & . & & & . \\ c_s & a_{s1} & . & . & . \; a_{ss} \\ \hline & b_1 & . & . & . \; b_s \end{array}$$

The number of stages s and the constant coefficients $\{b_i\}$,$\{a_{ij}\}$ completely characterize a Runge-Kutta method. In general, such a method is implicit and leads to a

nonlinear system in the s internal stage variables y_n. An example of a fourth-order explicit Runge-Kutta method is given (Butcher 1987).
Let an initial value problem be specified as follows:

$$\dot{y} = f(t, y) \qquad y(t_0) = y_0 \tag{3.6}$$

Then, the $RK4$ method for this problem is given by the following equations:

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.7}$$

$$t_{n+1} = t_n + h \tag{3.8}$$

where y_{n+1} *is the RK4 approximation of* $y(t_{n+1})$ *and*

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\ k_3 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2) \\ k_4 &= f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_3) \end{aligned}$$

3.5. Partitioned Euler Method

For *partitioned* systems

$$\begin{aligned} \dot{u} &= a(u, v), \\ \dot{v} &= b(u, v), \end{aligned} \tag{3.9}$$

such as the Lotka-Volterra problem we consider also *partitioned* Euler methods

$$u_{n+1} = u_n + ha(u_n, v_{n+1}) \tag{3.10}$$

$$v_{n+1} = v_n + hb(u_n, v_{n+1}) \tag{3.11}$$

which treat one variable by the implicit and the other variable by the explicit Euler method. In view of an important property of this method, discovered by de Vogelaere (1956) and we call them *symplectic Euler methods* if we apply the method to Hamiltonian systems (Hairer 2005).

3.6. The Störmer-Verlet Scheme

The equations for the pendulum are of the form

$$\begin{aligned} \dot{p} &= f(q) \\ \dot{q} &= p \end{aligned} \tag{3.12}$$

or

$$\ddot{q} = f(q)$$

which is the important special case of a second order differential equation. The most natural discretisation of (3.12) is

$$q_{n+1} - 2q_n + q_{n-1} = h^2 f(q_n), \tag{3.13}$$

which is just obtained by replacing the second derivative in (3.12) by the central second-order difference quotient. This basic method, or its equivalent formulation given below, is called the *Störmer method* in astronomy, *the Verlet method* in molecular dynamics, the *leap-frog method* in the context of partial differential equations, and it has further names in other areas. Störmer(1907) used higher-order variants for numerical computations in molecular dynamics, where it has become by far the widely used integration scheme (Hairer 2005).

Geometrically, the Störmer-Verlet method can be seen as produced by parabolas, which in the points t_n possess the right second derivative$f(q_n)$. But we can also think of polygons, which possess the right slope in the midpoints (Figure(3.1) to the right).

Approximations to the derivative $p = \dot{q}$ are simply obtained by

$$p_n = \frac{q_{n+1} - q_{n-1}}{2h} \quad and \quad p_{n+1/2} = \frac{q_{n+1} - q_n}{h}$$

One step formulation: The Störmer-Verlet method admits one-step formulation which is useful for actual computations. The value q_n together with the slope p_n and the second derivative $f(q_n)$, all at t_n, uniquely determine the parabola and hence also the approximation (p_{n+1}, q_{n+1}) at t_{n+1}. Writing (3.13) as $p_{n+1/2} - p_{n-1/2} = hf(q_n))$ and using $p_{n+1/2} + p_{n-1/2} = 2p_n$, we get by elimination of either $p_{n+1/2}$ or

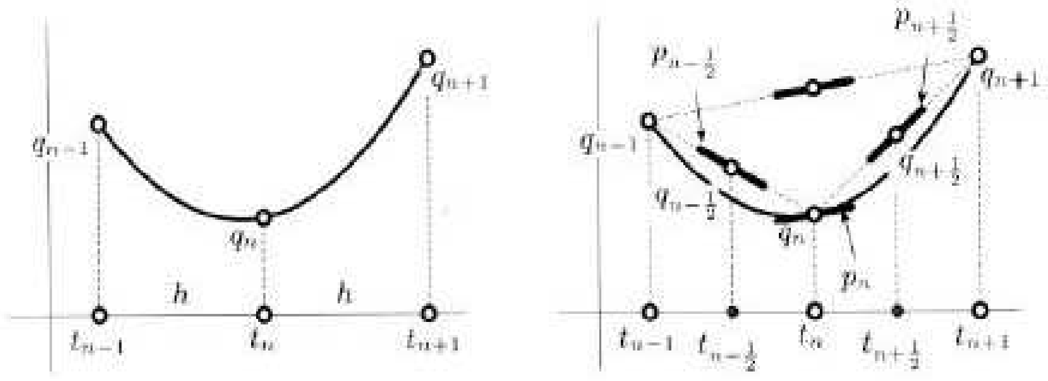

Figure 3.1. Illustration for the Störmer-Verlet method

$p_{n-1/2}$ the formula

$$\begin{aligned} p_{n+1/2} &= p_n + \frac{h}{2} f(q_n) \\ q_{n+1} &= q_n + h p_{n+1/2} \\ p_{n+1} &= p_{n+1/2} + \frac{h}{2} f(q_{n+1}) \end{aligned} \tag{3.14}$$

which is an explicit one-step method $\Phi_h : (q_n, p_n) \mapsto (q_{n+1}, p_{n+1})$ for the corresponding first order system of (3.12). If one is not interested in the values p_n of the derivative, the first and third equations in (3.14) can be replaced by

$$p_{n+1/2} = p_{n-1/2} + h f(q_n))$$

3.7. Partitioned Runge-Kutta Methods

Some interesting numerical methods introduced before(symplectic Euler and the Störmer-Verlet method) do not belong to the class of Runge-Kutta methods. They are important examples of so-called partitioned Runge-Kutta methods. Here we consider differential equations in the partitioned form

$$p' = f(p, q), \quad q' = g(p, q). \tag{3.15}$$

We have chosen the letters p and q for the dependent variables, because Hamiltonian systems are of this form. The idea is to take two different Runge-Kutta methods, and to treat the $y - \textit{variables}$ with the first method (a_{ij}, b_j) and the $z - \textit{variables}$ with the second method $(\hat{a}_{ij}, \hat{b}_j)$ (Hairer, et al. 2001).

Definition 3.1 *Let b_i; a_{ij} and $\hat{b}_i$; $\hat{a}_{ij}$ be the coefficients of two Runge-Kutta methods. A partitioned Runge-Kutta method is given by*

$$
\begin{aligned}
k_i &= f(p_0 + h\sum_{j=1}^{S} a_{ij}k_j, q_0 + h\sum_{j=1}^{S} \hat{a}_{ij}\ell_j) \\
\ell_i &= g(p_0 h\sum_{j=1}^{S} a_{ij}k_j, q_0 + h\sum_{j=1}^{S} \hat{a}_{ij}\ell_j) \\
p_1 &= p_0 + h\sum_{i=1}^{S} b_i k_i, \quad q_1 = q_0 + h\sum_{i=1}^{S} \hat{b}_i \ell_i
\end{aligned}
\tag{3.16}
$$

Methods of this type have originally been proposed by Hofer (1976) and Griepentrog (1978). Their importance for Hamiltonian systems has been discovered only very recently.

An interesting example is the partitioned Euler method (3.9-3.11), where the implicit Euler method $b_1 = 1, a_{11} = 1$ is combined with the explicit Euler method $\hat{b}_1 = 1, \hat{a}_{11} = 0$. The Störmer-Verlet method (3.14) is of the form (3.16) with coefficients given in below table

$$
\begin{array}{c|cc}
0 & 0 & 0 \\
1 & 1/2 & 1/2 \\
\hline
 & 1/2 & 1/2
\end{array}
\qquad
\begin{array}{c|cc}
1/2 & 1/2 & 0 \\
1/2 & 1/2 & 0 \\
\hline
 & 1/2 & 1/2
\end{array}
$$

The theory of Runge-Kutta methods can be extended in a straight-forward way to partitioned methods. Since equation (3.16) is a one-step method $(p_1, q_1) = \Phi_h(p_0, q_0)$, the order, the adjoint method and symmetric methods are defined in the usual way (Hairer, et al. 2001).

3.8. Splitting and Composition Methods

These methods exploit natural decompositions of the problems and particular of the Hamiltonian and have been used with great success in studies of the solar system and of molecular dynamics. Much recent work in this field is due to Yoshida and McLachlan (Budd and Piggott 2000). The main idea behind splitting methods is to decompose the discrete flow Ψ_h as a composition of simpler flows

$$\Psi_h \;=\; \Psi_{1,h} \circ \Psi_{2,h} \circ \Psi_{3,h} \ldots,$$

where each of the sub-flows is chosen such that each represent a simpler integration of the original. A geometrical perspective on this approach is to find useful geometric properties of each of the individual operations which are preserved under combination, symplecticity is just such a property, but we often seek to preserve reversibility and other structures. Suppose that a differential equation takes the form

$$\frac{du}{dt} = f = f_1 + f_2 \tag{3.17}$$

Here the functions f_1 and f_1 may well represent different physical processes in which case there is a natural decomposition of the problem (say into terms to kinetic and potential energy).

The most direct form of splitting methods decompose this equation into two problems

$$\frac{d\mathrm{u}_1}{dt} = f_1 \text{ and } \frac{d\mathrm{u}_2}{dt} = f_2 \tag{3.18}$$

chosen such that these two problems can be integrated *exactly in closed form* to give explicitly computable flows $\Psi_1(t)$ and $\Psi_2(t)$. We denote by $\Psi_{i,h}$ the result of applying the corresponding continuous flows $\psi_i(t)$ over a time h. A simple (first order) splitting is then given by the *Lie-Trotterr formula*

$$\Psi_h = \Psi_{1,h} \circ \Psi_{2,h}$$

Suppose that the original problem has Hamiltonian $H = H_1 + H_2$, then as observed earlier this is the composition of two problems with respective Hamiltonian H_1 and H_2. The differential equation corresponding to each Hamiltonian leads to an evolutionary map $\psi_i(t)$ of the form described above.

The Lie-Trotter splitting introduces local errors proportional to h^2 at each step and a more accurate decomposition is the *Strang splitting* given by

$$\Psi_h = \Psi_{1,h/2} \circ \Psi_{2,h} \circ \Psi_{1,h/2}.$$

This splitting method has a local error proportional to h^3 (Budd and Piggott 2000).

Example 3.1 *Suppose that a Hamiltonian system has a Hamiltonian which can be expressed as a combination of a kinetic energy and a potential energy term as follows*

$$H(u) \;=\; H_1(u) + H_2(u) \equiv T(p) + V(q).$$

so that

$$\frac{d\mathbf{p}}{dt} \;=\; -H_{2,q}(\mathbf{q}),\; \frac{d\mathbf{q}}{dt} \;=\; H_{1,p}(\mathbf{p}) \tag{3.19}$$

This splitting of H is usually referred to as a separable or P-Q splitting. We immediately have that

$$\Psi_{1,h} \;=\; I - hH_{2,q} \;\text{ and }\; \Psi_{2,h} \;=\; I + hH_{1,p}$$

where I represents the identity mapping.
Applying the Lie-Trotter formula directly to this splitting gives the *symplectic Euler method* (Budd and Piggott 2000)

$$p_{n+1} = p_n - hH_{2,q}(q_n), \;\text{ and }\; q_{n+1} = q_n + hH_{1,p}(p_{n+1}).$$

A more sophisticated splitting method for this problem based upon the Strang splitting is

$$p_{n+1/2} = p_n - \frac{h}{2}H_{2,q}(q_n),\; q_{n+1} = q_n + hH1,p(p_{n+1/2}),\; p_{n+1} = p_{n+1/2} - \frac{h}{2}H_{2,q}(q_{n+1})$$

For systems with such separable Hamiltonians we may combine subsequent Strang splitting (as described above) to give (after relabeling) the celebrated *Störmer-Verlet method*(Budd and Piggott 2000).

$$q^{n+1/2} = q^{n-1/2} + hT_p(p^n), \tag{3.20}$$

$$p^{n+1} = p^n - hV_q(q^{n+1/2}). \tag{3.21}$$

A form of this method appeared in the molecular dynamics literature many years before anyone realized that its remarkable success in that field was due to the fact that it was in fact a very efficient symplectic method.

Example 3.2 *Yoshida splitting*

The Strang splitting has the desirable property that it is symmetric so that $\Psi_h^{-1} = \Psi_{-h}$. By combining symmetric splitting, Yoshida derived a series of remarkable high order methods. Given a symmetric second order *base method* $\Psi_h^{(2)}$ for example in Yoshida considers the case where the base method is given by the Strang splitting (Budd and Piggott 2000). Yoshida then constructs the method

$$\Psi_h^{(4)} = \Psi_{x_1 h}^{(2)} \circ \Psi_{x_0 h}^{(2)} \circ \Psi_{x_1 h}^{(2)}.$$

He proved that $\Psi_h^{(4)}$ is indeed a fourth-order symmetric method if the weights are chosen as

$$x_0 = -\frac{2^{1/3}}{2 - 2^{1/3}}, \ x_1 = \frac{1}{2 - 2^{1/3}}.$$

If the problem being considered is Hamiltonian and the second-order method is symplectic then our newly constructed fourth-order method will also be symplectic. This procedure can be extended and generalized as follows. Given a symmetric integrator $\Psi_h^{(2n)}$ of order $2n$, for example when $n = 1$ we are simply restating the above construction and when $n = 2$ we may take the method constructed above. The method given by the composition

$$\Psi_h^{(2n+2)} = \Psi_{x_1 h}^{(2n+2)} \circ \Psi_{x_0 h}^{(2n)} \circ \Psi_{x_1 h}^{(2n)}$$

will be a symmetric method of order $2n + 2$ if the weights are chosen as

$$x_0 = -\frac{2^{1/(2n+1)}}{2 - 2^{1/(2n+1)}}, \ \ x_1 = \frac{1}{2 - 2^{1/(2n+1)}}.$$

3.9. The Adjoint of a Method

The flow φ_t of an autonomous differential equation

$$\dot{y} = f(y) \quad y(t_0) = y_0 \tag{3.22}$$

satisfies $\varphi_{-t}^{-1} = \varphi_t$.

$$\begin{array}{rl} & y_n \underset{\longrightarrow}{\varphi_h} y_{n+1} \\ y_{n+1} = \varphi_h(y_n) \Rightarrow & y_n \underset{\longleftarrow}{\varphi_h^{-1}} y_{n+1} \\ & y_{n+1} \underset{\longrightarrow}{\varphi_{-h}} y_n \end{array} \tag{3.23}$$

Definition 3.2 *The adjoint method Φ_h^* of a method Φ_h is the inverse map of the original method with reversed time step -h, i.e.,*

$$\Phi_h^* := \Phi_{-h}^{-1} \tag{3.24}$$

In other words, $y_1 = \Phi_h^*(y_0)$ is implicitly defined by $\Phi_{-h}(y_1) = y_0$. A method for which $\Phi_h^* := \Phi_h$ is called *symmetric* (J.M. Sanz Serna 1994).

Lemma 3.1 *Implicit Euler Method is the adjoint of explicit Euler Method.*

Proof:

Implicit Euler Method $\rightarrow y_{n+1} = y_n + hf(y_{n+1})$

Explicit Euler Method $\rightarrow y_{n+1} = y_n + hf(y_{n+1})$

$$y_{n+1} = \varphi_h(y_n)$$
$$y_{n+1} = \varphi_{-h}(y_n)$$
$$y_n = \varphi_{-h}^{-1}(y_{n+1}) = \varphi_h^*$$

exchanging h by -h and (n+1) by (n)

$$y_n = y_{n+1} - hf(y_n) \Rightarrow y_{n+1} = y_n + hf(y_n) \tag{3.25}$$

then we proved that the adjoint of explicit Euler method is implicit Euler method.

CHAPTER 4

CONSERVED QUANTITIES AND GEOMETRIC INTEGRATION

In this chapter we give both geometric structures of differential equations and numerical methods that we consider in chapter 3. We consider three geometric structures, first integrals, symplecticity and reversibility and symmetry. We give examples of differential equations which have these geometric features. Then we mention which of the methods introduced in Chapter 3 preserve first integrals and have symplecticity or reversibility properties. Finally we give the philosophy of geometric integration.

4.1. Exact Conservation of Invariants

This section is devoted to the conservation of invariants (first integrals). First we give the definition of first integral or invariant of a differential equation.

Definition 4.1 *: A non-constant function $I(y)$ is called a first integral of the system $\dot{y} = f(y)$, if*

$$\frac{dI}{dt} = I'y' = I'f = 0 \ \ for \ \ all \ \ y \tag{4.1}$$

This implies that every solution y(t) of $I'(y)f(y) = 0$ satisfies $I(y(t)) = I(y_0) = Const$.

For example:

$$u' = u(v-2) \tag{4.2}$$

$$v' = v(1-u) \tag{4.3}$$

is the Lotka-Volterra system of equations. Equations (4.2) and (4.3) constitute an autonomous system of differential equations. This system of equations has $I(u;v) = ln\, u - u + 2\, ln\, v - v \ \ as\ first\ integral.$ (Hairer, et al. 2001)

Since

$$\frac{dI}{dt} = \frac{1}{u}.u' - u' + \frac{2}{v}.v' - v'$$

substituting the (4.2) and (4.3) in above derivation

$$\begin{aligned}\frac{dI}{dt} &= \frac{1}{u}.u(v-2) - u(v-2) + \frac{2}{v}.v(1-u) - v(1-u)\\ &= v - 2 - uv + 2u + 2 - 2u - v + uv = 0.\end{aligned}$$

Lemma 4.1 ***:Conservation of the Total Energy***

Every Hamiltonian system is of the form

$$\dot{p} = -H_q(p,q), \qquad \dot{q} = H_p(p,q) \tag{4.4}$$

the Hamiltonian function $H(p,q)$ is a first integral for every Hamiltonian system is of the above form.

Proof : This follows at once from
$H'(p,q) = (\frac{\partial H}{\partial p} , \frac{\partial H}{\partial q})$ and

$$\frac{dH}{dt} = \frac{\partial H}{\partial p}(-\frac{\partial H}{\partial q}) + \frac{\partial H}{\partial q}(\frac{\partial H}{\partial p}) = 0$$

Example 4.1 *:* ***Conservation of the Total and Angular Momentum of N-Body systems.***

We consider a system of N particles interacting pairwise with potential forces which depend on the distance between the particles. This is formulated as a Hamiltonian system with total energy

$$H(p,q) = \frac{1}{2}\sum_{i=1}^{N}\frac{1}{m_i}p_i^T p_i + \sum_{i=1}^{N}\sum_{i\neq j=1}^{N} V_{ij}(\| q_i - q_j \|) \tag{4.5}$$

Here $p_i, q_i \in \mathbb{R}^3$ represent the position and momentum of the ith particle of mass m_i and V_{ij} $(i > j)$ is the interaction potential between the ith and jth particle. The equations of motion read

$$\dot{q}_i = \frac{1}{m_i}p_i, \qquad \dot{p}_i = \sum_{j=1}^{N} v_{ij}(q_i - q_j) \tag{4.6}$$

where, for $i > j$ we have $v_{ij} = v_{ji} = V'_{ij}(r_{ij})/r_{ij}$ with $r_{ij} =\| q_i - q_j \|$, and v_{ii} is arbitrary, say $v_{ii} = 0$. The conservation of the total linear momentum $P = \sum_{i=1}^{N} p_i$ and the angular momentum $L = \sum_{i=1}^{N} q_i \times p_i$ is the consequences of symmetry relation

$v_{ij} = v_{ji}$

$$\frac{d}{dt}\sum_{i=1}^{N} p_i = \sum_{i=1}^{N}\sum_{j=1}^{N} v_{ij}(q_i - q_j) = 0 \tag{4.7}$$

$$\frac{d}{dt}\sum_{i=1}^{N} q_i \times p_i = \sum_{i=1}^{N} \frac{1}{m_i} p_i \times p_i + \sum_{i=1}^{N}\sum_{j=1}^{N} q_i \times v_{ij}(q_i - q_j) = 0 \tag{4.8}$$

Theorem 4.1 ***(Conservation of Linear First Integrals)*** *: All explicit and implicit Runge-Kutta methods as well as multi-step methods conserve linear first integrals. Partitioned Runge-Kutta methods conserve linear first integrals only if $b_i = \hat{b}_i$ for all i, or if the first integral only depends on p alone or only on q alone.*

Proof : Let $I(y) = b^T y$ with constant vector b, so that $b^T f(y) = 0$ *for all* y. In the case of Runge-Kutta methods we thus have $b^T k_i = 0$, and consequently $b^T y_1 = b^T y_0 + hb^T(\sum_{i=1}^{s} b_i k_i) = b^T y_0$. The statement for partitioned method is proved similarly.

4.1.1. Quadratic Invariants

Quadratic invariants appear often in applications. Conservation of the angular momentum is an example of quadratic invariants. We consider differential equation $y' = f(y)$ and quadratic function

$$Q(y) = y^T C y$$

where C is a symmetric square matrix. It is an invariant of $y' = f(y)$ if $y^T C f(y) = 0$ *for all* y (Hairer, et al. 2001).

Since $y^T C f(y) = 0$

$$\begin{aligned} Q'(y) &= (y')^T C y + y^T C y' = (f(y))^T C y + y^T C f(y) \\ &= (y^T C f(y))^T + y^T C f(y) = 0 \end{aligned}$$

Theorem 4.2 *If the coefficients of a Runge-Kutta method satisfy*

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \tag{4.9}$$

then it conserves quadratic invariants.

Proof: See (Hairer, et al. 2001).

We next consider partitioned Runge-Kutta methods for systems $\dot{y} = f(y,z) \quad \dot{z} = g(y,z)$. Usually such methods cannot conserve general quadratic invariants. We therefore concentrate on quadratic invariants of the form

$$Q(y,z) = y^T Dz$$

where D is a matrix of appropriate dimensions. Observe that the angular momentum is of this form.

Theorem 4.3 *Sörmer-Verlet scheme conserves all quadratic invariants of the form $Q(y,z) = y^T Dz$.*

Proof: See (Hairer, et al. 2001).

Example 4.2 *: **Rigid Body***

The motion of the free rigid body, whose center of mass is at the origin, is described by the Euler equations

$$\dot{y}_1 = a_1 y_2 y_3, \qquad a_1 = (I_2 - I_3)/I_2 I_3 \tag{4.10}$$

$$\dot{y}_2 = a_2 y_3 y_1, \qquad a_2 = (I_3 - I_1)/I_3 I_1 \tag{4.11}$$

$$\dot{y}_3 = a_1 y_2 y_2, \qquad a_3 = (I_1 - I_2)/I_1 I_2 \tag{4.12}$$

where the vector $y = (y_1, y_2, y_3)^T$ represents the angular momentum in the body frame,and I_1, I_2, I_3 are the principal moments of inertia This problem can be written as

$$\begin{pmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{pmatrix} = \begin{pmatrix} 0 & y_3/I_3 & -y_2/I_2 \\ -y_3/I_3 & 0 & y_1/I_1 \\ y_2/I_2 & -y_1/I_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

which is of the form with a skew-symmetric matrix A(Y). By the theorem $y_1^2 + y_2^2 + y_3^2 = R^2$ is first integral. It is interesting to note that the system is actually a Hamiltonian system with Hamiltonian

$$H(y_1, y_2, y_3) = \frac{1}{2}\left(\frac{y_1^2}{I_1} + \frac{y_2^2}{I_2} + \frac{y_3^2}{I_3}\right)$$

This is a second quadratic invariant of the system.

We present in Figure(4.1) the sphere with some of the solutions of corresponding to $I_1 = 2, I_2 = 1$ *and* $I_3 = 2/3$. In the left picture we have included the numerical solution (30 steps) obtained by the implicit midpoint rule with step size h = 0,3 and initial value $y_0 = (cos(1.1); 0; sin(1.1))^T$. It stays exactly on a solution curve. This follows from the fact that the implicit midpoint rule preserves quadratic invariants exactly. For the explicit Euler method (right picture, 320 steps with h = 0.05 and the same initial value as above) we see that the numerical solution shows the wrong qualitative behaviour (it should lie on a closed curve). The numerical solution even drifts away from the manifold (Hairer, et al. 2001)

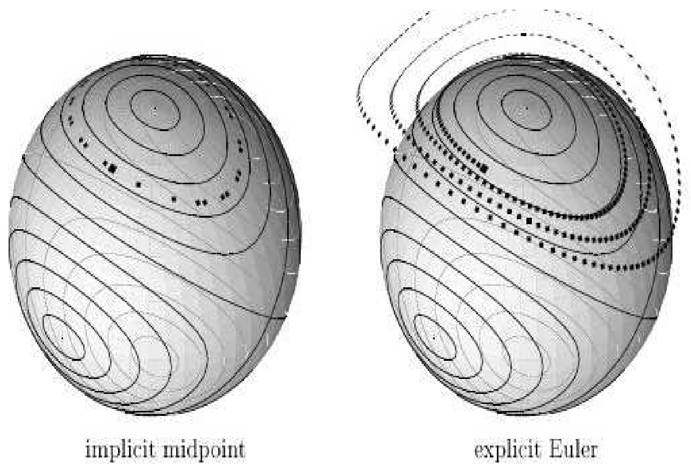

Figure 4.1. Solution of the Euler equations (4.10),(4.11),(4.12) for the rigid body.

4.2. Symplectic Transformations

In this section we discuss the symplecticity property. The basic objects to be studied are two-dimensional parallelograms lying in $\mathbb{R}^{2d}$. We suppose the parallelogram to be spanned by two vectors

$$\xi = \begin{pmatrix} \xi^p \\ \xi^q \end{pmatrix}, \qquad \eta = \begin{pmatrix} \eta^p \\ \eta^q \end{pmatrix}$$

in the (p, q) space ($\xi^p, \xi^q, \eta^p, \eta^q$ are in $\mathbb{R}^d$) as

$$P = \{t\xi + s\eta | 0 \leq t \leq 1, 0 \leq s \leq 1\}.$$

In the case d=1 we consider the *oriented area*

$$Area(P) = det\begin{pmatrix} \xi^p & \eta^p \\ \xi^q & \eta^q \end{pmatrix} = \xi^p\eta^q - \xi^q\eta^p$$

In higher dimensions, we replace this by the sum of the oriented areas of the projections of P onto the coordinate planes (p_i, q_i), i.e., by

$$\omega(\xi,\eta) := \sum_{i=1}^{d} det\begin{pmatrix} \xi_i^p & \eta_i^p \\ \xi_i^q & \eta_i^q \end{pmatrix} = \sum_{i=1}^{d}(\xi_i^p\eta_i^q - \xi_i^q\eta_i^p).$$

This defines a bilinear map acting on vectors of $\mathbb{R}^{2d}$, which will play a central role for Hamiltonian system. In matrix notation, this map has the form

$$\omega(\xi,\eta) = \xi^T J\eta \qquad \text{with} \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$$

where I is the identity matrix of dimension d (Hairer, et al. 2001).

Definition 4.2 *A linear mapping* $A : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ *is called symplectic if*

$$A^T JA = J$$

or, equivalently, if $\omega(A\xi, A\eta) = \omega(\xi,\eta)$ *for all* $\xi, \eta \in \mathbb{R}^{2d}$.

We can find it

$$\omega(A\xi, A\eta) = (A\xi)^T J(A\eta) = \xi^T \underbrace{A^T JA}\, \eta \qquad (\text{since} \quad A^T JA = J)$$

$$= \xi^T J\eta = \omega(\xi,\eta)$$

Definition 4.3 *For nonlinear mapping, the differentiable functions can be locally approximated by linear functions. A differentiable map* $g : U \to \mathbb{R}^{2d}$ *(where* $U \subset \mathbb{R}^{2d}$ *is an open set) is called symplectic if the Jacobian matrix* $g'(p,q)$ *is everywhere symplectic, i.e., if*

$$g'(p,q)^T Jg'(p,q) = J \quad \text{or} \quad \omega(g'(p,q)\xi, g'(p,q)\eta) = \omega(\xi,\eta).$$

Lemma 4.2 *If* ψ *and* φ *are symplectic maps then* $\psi \circ \varphi$ *is symplectic.*

Proof: *Since* ψ *is symplectic*

$$(\psi')^T J\psi' = J, \qquad \text{similarly} \qquad (\varphi')^T J\varphi' = J$$

$$\left[(\psi o\varphi)'\right]^T J(\psi o\varphi)' = (\psi' o\varphi')^T J(\psi' o\varphi') = (\varphi')^T o(\psi')^T J\psi' o\varphi' = J$$

Theorem 4.4 *(Poincaré 1899). Let $H(p,q)$ be a twice continuously differentiable function on $U \subset \mathbb{R}^{2d}$. Then, for each fixed t, the flow φ_t is a symplectic transformation wherever it is defined.*

Proof: Let φ_t be flow of the Hamiltonian system. φ_t' is a Jacobian matrix of the flow, then φ_t' satisfies the variational equation i.e.

$$\frac{d}{dt}\varphi_t' = J^{-1}H''\varphi_t' \qquad \text{where} \quad H'' = \begin{pmatrix} H_{pp} & H_{pq} \\ H_{qp} & H_{qq} \end{pmatrix} \quad \text{is symmetric.}$$

Hence

$$\frac{d}{dt}(\varphi_t'^T J\varphi_t') = [J^{-1}H''\varphi_t']^T J\varphi_t' + \varphi_t'^T \underbrace{JJ^{-1}} H''\varphi_t'$$

since $JJ^{-1} = I$

$$\frac{d}{dt}(\varphi_t'^T J\varphi_t') = \varphi_t'^T H''^T \underbrace{(J^{-1})^T J}\,\varphi_t' + \varphi_t'^T H''\varphi_t'$$

now, we will use $(H'')^T = H''$ and $(J^{-1})^T J = -I$, let us prove it;

$$J^T = -J$$

$$[(J^{-1})^T J]^T = J^T \cdot J^{-1} = -J \cdot J^{-1} = -I$$

then we finally find $(J^{-1})^T J = -I$. We put $(J^{-1})^T J = -I$ in the last equation;

$$\frac{d}{dt}(\varphi_t'^T J\varphi_t') = -\varphi_t'^T H''\varphi_t' + \varphi_t'^T H''\varphi_t' = 0$$

Since $\frac{d}{dt}(\varphi_t'^T J\varphi_t') = 0$ then $\varphi_t'^T J\varphi_t' = \mathbf{C}$. When $t = 0$, we have $\varphi_t'(t_0) = I \Rightarrow C = J$

Theorem 4.5 *Let $f : U \to \mathbb{R}^{2d}$ be continuously differentiable. Then, $\dot{y} = f(y)$ is locally Hamiltonian if and only if its flow $\varphi_t(y)$ is symplectic for all $y \in U$ and for all sufficiently small t.*

Proof: Assume that the flow φ_t is symplectic, and we have to prove the local existence of a function $H(y)$ such that $f(y) = J^{-1}\nabla H(y)$. Using the fact that $\frac{\partial\varphi_t}{\partial y_0}$ is a solution of the variational equation $\dot{\Psi} = f'(\varphi_t(y_0))\Psi$, we obtain

$$\frac{d}{dt}\left(\left(\frac{\partial\varphi_t}{\partial y_0}\right)^T J\left(\frac{\partial\varphi_t}{\partial y_0}\right)\right) = \left(\frac{\partial\varphi_t}{\partial y_0}\right)\left(f'(\varphi_t(y_0))^T J + Jf'(\varphi_t(y_0)\right)\left(\frac{\partial\varphi_t}{\partial y_0}\right) = 0.$$

Putting $t = 0$, it follows from $J = -J^T$ that $Jf'(y_0)$ is a symmetric matrix for all y_0.

Theorem 4.6 *Let $\psi : U \to V$ be a change of coordinates such that ψ and ψ^{-1} are continuously differentiable functions. If ψ is symplectic, the Hamiltonian system $\dot{y} = J^{-1}\nabla H(y)$ becomes in the new variables $z = \psi(y)$*

$$\dot{z} = J^{-1}\nabla K(z) \quad \text{with} \quad K(z) = H(y).$$

Conversely, if ψ transforms every Hamiltonian system to another Hamiltonian system, then ψ is symplectic.

Proof: Since $\dot{z} = \psi'(y)\dot{y}$ and $\psi'(y)^T\nabla K(z) = \nabla H(y)$, the Hamiltonian system $\dot{y} = J^{-1}\nabla H(y)$ becomes

$$\dot{z} = \psi'(y)J^{-1}\psi'(y)^T\nabla K(z)$$

in the new variables. It is equivalent to $\dot{z} = J^{-1}\nabla K(z) \quad \text{with} \quad K(z) = H(y)$ if

$$\psi'(y)J^{-1}\psi'(y)^T = J^{-1}.$$

Multiplying this relation from the right by $\psi'(y)^{-T}$ and from the left by $\psi'(y)^{-1}$ and then taking its inverse yields $J = \psi'(y)^T J\psi'(y)$, which shows that the last equation is equivalent to the symplecticity of ψ.

4.3. Examples of Symplectic Integrators

Definition 4.4 *A numerical one-step method is called symplectic if the one-step map $y_1 = \Phi_h(y_0)$ is symplectic whenever it is applied to a smooth Hamiltonian system.If the method is symplectic:*

$$\Phi_h'(y)^T\, J\, \Phi_h'(y) = J \tag{4.13}$$

where $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$

Theorem 4.7 *The implicit midpoint rule is symplectic.*

Proof: The second order implicit midpoint rule is:

$$U_{n+1} = U_n + hf\Big(\frac{U_n + U_{n+1}}{2}\Big) \tag{4.14}$$

Consider the Hamiltonian problem

$$\dot{y} = J^{-1}\nabla H(y) \tag{4.15}$$

$$U_{n+1} = U_n + hJ^{-1}\nabla H\Big(\frac{U_n + U_{n+1}}{2}\Big) \tag{4.16}$$

$U_{n+1} = \psi_n(U_n)$ and need to show $\psi_n'^T J \psi_n' = J$

$$\psi_n' = \frac{\partial U_{n+1}}{\partial U_n} = I + hJ^{-1}H''\Big(\frac{U_n + U_{n+1}}{2}\Big)\Big(\frac{1}{2}\Big)\Big(\frac{\partial U_{n+1}}{\partial U_n} + I\Big) \tag{4.17}$$

$$\psi_h' = \frac{\partial U_{n+1}}{\partial U_n} = \Big(I - \frac{h}{2}J^{-1}H''\Big)^{-1}\Big(I + \frac{h}{2}J^{-1}H''\Big) \tag{4.18}$$

$\psi_n'^T J \psi_n' = J$ means:

$$\Big(I + \frac{h}{2}J^{-1}H''\Big)J\Big(I + \frac{h}{2}J^{-1}H''\Big)^T = \Big(I - \frac{h}{2}J^{-1}H''\Big)J\Big(I - \frac{h}{2}J^{-1}H''\Big)^T \tag{4.19}$$

By using the equalities

1)$(H'')^T = H''$ (since H is symmetric)

2)$(J^{-1})^T = -J^{-1} = J$

we get

$$\Big(I + \frac{h}{2}J^{-1}H''\Big)^T = I - \frac{h}{2}H''J^{-1}$$
$$\Big(I - \frac{h}{2}J^{-1}H''\Big)^T = I + \frac{h}{2}H''J^{-1}$$

Then

$$\Big(IJ + \frac{h}{2}J^{-1}H''J\Big)\Big(I - \frac{h}{2}H''J^{-1}\Big) = \Big(IJ - \frac{h}{2}J^{-1}H''J\Big)\Big(I + \frac{h}{2}H''J^{-1}\Big) \tag{4.20}$$

$$J + \frac{h}{2}J^{-1}H''J - \frac{h}{2}JH''J^{-1} - \frac{h^2}{4}J^{-1}H''JH''J^{-1} = J + \frac{h}{2}JH''J^{-1} - \frac{h}{2}J^{-1}H''J - \frac{h^2}{4}J^{-1}H''JH''J^{-1} \tag{4.21}$$

And we find

$$\Rightarrow hJH''J^{-1} = hJ^{-1}H''J \tag{4.22}$$
$$J^{-1} = -J \Rightarrow -JH''J = -JH''J$$

4.4. Symplectic Runge-Kutta Methods

It is fortunate that an important class of Runge-Kutta methods namely Gauss-Legendre methods also turn out to be symplectic methods. The implicit midpoint rule is considered earlier is a member of this class. This gives a very useful class of methods applicable to a wide class of general Hamiltonian problems, the resulting methods are highly implicit and for specific problems splitting methods, which maybe explicit, maybe preferable (Budd and Piggott 2000). To explain this Runge-Kutta methods, in these section consider a standard s-stage Runge-Kutta method with Butcher tableau

$$\begin{array}{c|c} \mathrm{c} & \mathrm{A} \\ \hline & b^T \end{array}$$

where $c, b \in R^s$ *and* $A = (a_{i,j}) \in R^{s \times s}$. The following result gives complete characterization of all symplectic Runge-Kutta methods.

Theorem 4.8 *A necessary and sufficient condition for a Runge-Kutta method to be symplectic is that for all* $1 \leq i, j \leq s$

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0 \tag{4.23}$$

Proof: See (Hairer, et al. 2001).

Note that the necessity of condition (4.23) requires that the method have no redundant stages i.e the method be reducible. We can immediately see that if the matrix A was lower triangular then (4.23) would imply that $b_i = 0$, $\forall i$, and therefore we can conclude that symplectic Runge-Kutta methods must be implicit. This not an ideal state of affairs, however we shall see presently that in a number of very important situation methods which are both symplectic and explicit may be found (Budd and Piggott 2000).

The proof of this result is a consequence of the fact that any method satisfying (4.23) automatically preserve any quadratic invariant. of the solution of the equation to which is applied.

The natural partitioning present in Hamiltonian problem suggest that we may think about using different numerical methods for different components of the problem. We shall now consider partitioned Runge-Kutta methods, where we integrate p components with the Runge-Kutta method given by the first Butcher tableau and the q components with the second tableau below.

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \qquad \begin{array}{c|c} \hat{c} & \hat{A} \\ \hline & \hat{b}^T \end{array}$$

As we have done for symplectic Runge-Kutta methods we may now classify partitioned Runge-Kutta methods with the following result.

For special case of problems where the Hamiltonian take the separable form $H(p, q) = T(p) + V(q)$ only the second part of (4.23) is required for symplecticity.

An example of such a method by Ruth's third order method it has been tableau

$$\begin{array}{c|ccc} & 7/24 & 0 & 0 \\ c & 7/24 & 3/4 & 0 \\ & 7/24 & 3/4 & -1/24 \\ \hline & 7/24 & 3/24 & -1/24 \end{array} \qquad \begin{array}{c|ccc} & 0 & 0 & 0 \\ \hat{c} & 2/3 & 0 & 0 \\ & 2/3 & -2/3 & 0 \\ \hline & 2/3 & -2/3 & 1 \end{array}$$

Theorem 4.9 *If the coefficients of a partitioned Runge-Kutta method satisfy*

$$b_i\hat{a}_{ij} + \hat{b}_j a_{ji} = b_i\hat{b}_j \quad for\ i, j = 1, ..., s, \tag{4.24}$$

$$b_i = \hat{b}_i \quad for\ i = 1, ..., s, \tag{4.25}$$

then it is symplectic.

Theorem 4.10 *A symplectic Runge-Kutta method leaves all quadratic first integrals of a hamiltonian system invariant, i.e. if* $y_n = (p_n, q_n)$ *and* $G = G^t \in M_{2d}(\mathbb{R})$,*then* $y^t_{n+1}Gy_{n+1} = y^t_n Gy_n$ *for all n.*

In particular, if a linear autonomous system is integrated with a symplectic Runge-Kutta method, than the energy will be conserved exactly (up to truncation errors, of course).

4.5. Examples of Applications of Symplectic Integrators

In these section we look the application of symplectic integrators to the harmonic oscillator as a Hamiltonian system.

The Harmonic Oscillator

This well studied problem has a separable Hamiltonian of the form

$$H(p, q) = \frac{p^2}{2} + \frac{q^2}{2}, \tag{4.26}$$

and has solutions which are circles in the (p,q) phase space. The associated differential equations are

$$\frac{dq}{dt} = p, \quad \frac{dp}{dt} = -q. \tag{4.27}$$

Consider now the closed curve (circle)

$$\Gamma \equiv p^2 + q^2 = C^2,$$

the action of the solution operator of the differential equation is to map this curve into itself(conserving the area πC^2 of the enclosed region). The standard *forward Euler method* applied to this system gives the scheme

$$p_{n+1} = p_n - hq_n, \quad q_{n+1} = q_n + hp_n, \tag{4.28}$$

so that Ψ_h is the operator given by

$$\Psi_h v = \begin{pmatrix} 1 & -h \\ h & 1 \end{pmatrix} v, \qquad with \;\; det(\Psi_h) = 1 + h^2.$$

It is easy to see that in this case,Γ evolves through the action of the discrete map Ψ_h to the new circle given by

$$p_{n+1}^2 + q_{n+1}^2 = C^2(1 + h^2) \tag{4.29}$$

and the area enclosed within the discrete evolution of Γ has increased by a factor of $1 + h^2$. Periodic orbits are not preserved by the forward Euler method – indeed all such discrete orbits spiral to infinity. Similarly, a discretisation using the backward Euler method leads to trajectories that spiral towards the origin (Budd and Piggott 2000).

Consider now the *symplectic Euler method* applied to this example. This gives rise to the discrete map

$$p_{n+1} = p_n - hq_n, \; q_{n+1} = q_n + h(p_n - hq_n) = (1 - h^2)q_n + hp_n \tag{4.30}$$

The discrete evolutionary is then simply matrix

$$\Psi_h \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 1 & -h \\ h & 1 - h^2 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}$$

which can easily be checked to be symplectic. For example det(Ψ_h)=1. The circle Γ is now mapped to the ellipse

$$(1 - h^2 + h^4)p_{n+1}^2 - 2h^3 p_{n+1}q_{n+1} + (1 + h^2)q_{n+1}^2 \; = \; C^2 \tag{4.31}$$

which has the same enclosed area. The symplectic Euler map is not symmetric in time so that

$$\psi_h^{-1} \; \neq \; \psi_{-h}.$$

Observe that the symmetry of the circle has been destroyed through the application of this mapping.

It is also easy to see that if

$$A \; = \; \begin{pmatrix} 1 & -\frac{h}{2} \\ -\frac{h}{2} & 1 \end{pmatrix}$$

then

$$\Psi_h^T A \Psi_h \; = \; A.$$

Next consider the *Störmer-Verlet* method. This gives the discrete map

$$p_{n+1/2} \; = \; p_n - hq_n/2, \quad q_{n+1} = q_n + h(p_n - hq_n/2), \tag{4.32}$$

$$p_{n+1} \; = \; p_{n+1}/2 - \frac{h}{2}(q_n + h(p_n - \frac{h}{2}q_n)). \tag{4.33}$$

The discrete evolutionary operator ψ_h is now the symplectic matrix

$$\Psi_h v = \begin{pmatrix} 1 - h^2/2 & -h + h^3/4 \\ h & 1 - h^2/2 \end{pmatrix} v.$$

The curve Γ is to order h^2 mapped to a circle of the same radius. The Störmer-Verlet method preserves the symmetry of the circle to this order, and indeed is also symmetric in time so that $\psi^{-1} = \psi_{-h}$.

Finally we consider the implicit midpoint rule. For this we have

$$C \begin{pmatrix} p_{n+1} \\ q_{n+1} \end{pmatrix} = C^T \begin{pmatrix} p_n \\ q_n \end{pmatrix},$$

where

$$C = \begin{pmatrix} 1 & \frac{h}{2} \\ -\frac{h}{2} & 1 \end{pmatrix}.$$

Observe that $CC^T = (1 + h^2/4)I$. The evolution operator is then simply

$$\Psi_h = C^{-1}C^T.$$

Observe further that if $U_n = (p_n, q_n)^T$ then

$$(1 + h^2/4)U_{n+1}^T U_{n+1} = U_{n+1}^T C^T C U_{n+1} = U_n^T C C^T U_n = (1 + h^2/4)U_n^T U_n.$$

Hence the quadratic invariant $U_n^T U_n$ is exactly preserved by this method. This feature is shared by all symplectic Runge-Kutta methods.

4.6. Symmetric Integration and Reversibility

Symmetric methods of this chapter and symplectic methods of preceding chapter play a center role in the geometric integration of differential equations. We discuss reversible differential equations and reversible maps, and we explain how symmetric integrators are related to them. We study symmetric Runge-Kutta and composition methods.

4.6.1. Reversible Differential Equations and Maps

Conservative mechanical systems have the property that inverting the initial direction of the velocity vector and keeping the initial position does not change the solution trajectory, it only inverts the direction of motion (Hairer, et al. 2001). Such systems are 'reversible'. We extend this notion to more general situations.

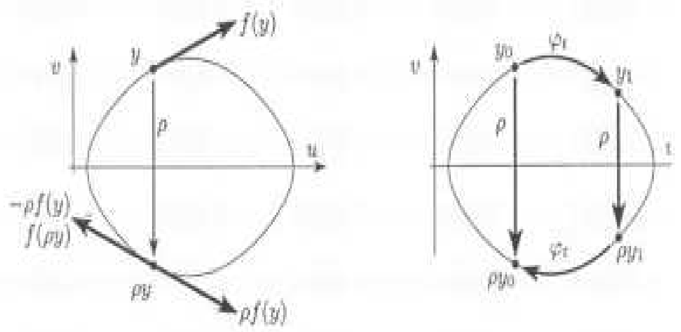

Figure 4.2. Reversible vector field (left picture) and reversible map (right picture)

Definition 4.5 *Let ρ be an invertible linear transformation in the phase space of $\dot{y} = f(y)$. This differential and vector field $f(y)$ called $\rho - reversible$ if*

$$\rho f(y) = -f(\rho y) \quad \text{for all} \quad y. \tag{4.34}$$

This property is illustrated in the left picture of Figure(4.3). For $\rho - reversible$ differential equations the exact flow $\varphi_t(y)$ satisfies

$$\rho \circ \varphi_t = \varphi_{-t} \circ \rho = \varphi_t^{-1} \circ \rho \tag{4.35}$$

Lemma 4.3 *Show the equality is true.*

$$\rho \circ \varphi_t = \varphi_{-t} \circ \rho = \varphi_t^{-1} \circ \rho \tag{4.36}$$

Proof: The right identity is a consequence of the group property

$$\varphi_t \circ \varphi_s = \varphi_{t+s}, \quad \varphi_t \circ \varphi_{-t} = \mathrm{I}, \quad \varphi_{-t} = \varphi_t^{-1},$$

and the left identity follows from $\dot{y} = f(y)$, φ_t is exact flow then $\frac{d}{dt}(\varphi_t) = f(\varphi_t)$ and $\rho(\varphi_t)$ is also solution of equation

$$\frac{d}{dt}(\rho \circ \varphi_t)(y) = \rho f(\varphi_t(y)) = -f((\rho \circ \varphi_t)(y))$$

$$\frac{d}{dt}(\varphi_{-t} \circ \rho)(y) = -f((\varphi_{-t} \circ \rho)(y))$$

Definition 4.6 *A map $\Phi(y)$ is called $\rho - reversible$ if*

$$\rho \circ \Phi = \Phi^{-1} \circ \rho \tag{4.37}$$

Example 4.3 ***: An important example is the partitioned system***

$$\dot{u} = f(u, v), \quad \dot{v} = g(u, v) \tag{4.38}$$

where $f(u, -v) = -f(u, v)$ and $g(u, -v) = g(u, v)$. Here, the transformation ρ is given by $\rho(u, v) = (u, -v)$. If we call a vector field or a map reversible (without specifying the transformation ρ), we mean that it is $\rho - reversible$ with this particular ρ.

$$-\rho f(y) = f(\rho(y))$$

$$f(u,-v) = -f(u,v)$$

$$g(u,-v) = g(u,v)$$

$$\rho(u,v) = (u,-v) \text{ where } (u,v) = Y$$

$$f(\rho(Y)) = f(u,-v) = -f(u,v) \qquad (*)$$

$$f(Y) = f(u,v) = -f(u,-v)$$

$$\rho(f(Y)) = f(u,v) \qquad (**)$$

from $(*)$ and $(**)$

$$f(\rho(Y)) = -\rho(f(Y))$$

$\dot{u} = g(u)$ is always reversible system. $\dot{u} = v = f(u,v), \dot{v} = g(u)$

$$f(u,-v) = -v = -f(u,v)$$

Definition 4.7 *A numerical one-step method Φ_h is called symmetric or time-reversible, if it satisfies*

$$\Phi_h \circ \Phi_{-h} = id \qquad \text{or equaivalently} \qquad \Phi_h = \Phi_{-h}^{-1}$$

With the definition of adjoint method ($\Phi_h^* = \Phi_{-h}^{-1}$), the condition for symmetry reads $\Phi_h = \Phi_h^*$. A method $y_1 = \Phi_h(y_0)$ is symmetric if exchanging $y_0 \leftrightarrow y_1$ and $h \leftrightarrow -h$ leaves the method unaltered. Implicit midpoint rule and the Störmer-Verlet scheme both of which are symmetric.

Theorem 4.11 *If a numerical method, applied to a $\rho -$ reversible differential equation satisfies*

$$\rho \circ \Phi_h = \Phi_{-h} \circ \rho, \tag{4.39}$$

then the numerical flow Φ_h is a $\rho -$reversible map if and only if Φ_h is a symmetric method.

Proof: As a consequence of (4.30) the numerical flow Φ_h is $\rho -$ *reversible* if and only if $\Phi_{-h} \circ \rho = \Phi_h^{-1} \circ \rho$. Since ρ is an invertible transformation, this is equivalent to the symmetry of the method Φ_h.

Example 4.4 *: **Explicit method*** $\Phi_h(y_0) = y_0 + hf(y_0)$

$$\rho \circ \Phi_h(y_0) = \rho(y_0 + hf(y_0)) = \rho y_0 + h\rho f(y_0) = \rho y_0 - hf(\rho(y_0)) = \Phi_{-h}(\rho(y_0))$$

Hence

$$\rho \circ \Phi_h = \Phi_{-h} \circ \rho.$$

Similarly, it is also true that a symmetric method is $\rho - $ *reversible* if and only if the ρ-compatibility condition (4.37) holds. Compared to the system of the method, condition (4.37) is much less restrictive. It is automatically satisfied by most numerical methods. Let us briefly discuss the validity of (4.37) for different classes of methods.

- *Runge-Kutta Methods*(explicit or implicit) satisfy (4.37) without any restriction other than (4.36) on the vector field. Let us illustrate the proof with the explicit Euler method $\Phi_h(y_0) = y_0 + hf(y_0)$:

  $$(\rho \circ \Phi_h)(y_0) = \rho y_0 + h\rho f(y_0) = \rho y_0 - hf(\rho y_0) = \Phi_{-h}(\rho y_0)$$

  (Hairer, et al. 2001)

- *Partitioned Runge-Kutta Methods* applied to a partitioned system satisfy the condition (4.37) if $\rho(u, v) = (\rho_1(u), \rho_2(v))$ with invertible ρ_1 and ρ_2. The proof is the same as for Runge-Kutta methods. Notice that the mapping $\rho(u, v) = (u, -v)$ of example (4.3) is of this special form (Hairer, et al. 2001).

- *Composition methods.* If two methods Φ_h and Ψ_h satisfy (4.37), then so does the adjoint Φ_h^* and the composition $\Phi_h \circ \Psi_h$. Consequently, the composition methods, which compose a basic method Φ_h and its adjoint with different step sizes, have the property (4.37) provided the basic method Φ_h has it (Hairer, et al. 2001).

- *Splitting methods* are based on a splitting $\dot{y} = f^{[1]}(y) + f^{[2]}(y)$ of the differential equation. If both vector fields, $f^{[1]}(y)$ and $f^{[2]}(y)$, satisfy (1), then their exact flows $\varphi_h^{[1]}$ and $\varphi_h^{[2]}$ satisfy (4.36). In this situation, the splitting method has the property (4.37) (Hairer, et al. 2001).

4.7. Examples of Symmetric Methods

Now we introduce some symmetric methods in this section.

Lemma 4.4 *Midpoint rule is a symmetric method.*

Proof: The midpoint rule is

$$y_{n+1} = y_n + hf\Big(\frac{y_{n+1} + y_n}{2}\Big)$$

exchanging h by -h and (n+1) by (n)

$$\begin{aligned} y_n &= y_{n+1} - hf\Big(\frac{y_{n+1} + y_n}{2}\Big) \\ y_{n+1} &= y_n + hf\Big(\frac{y_{n+1} + y_n}{2}\Big) \end{aligned}$$

then we proved that midpoint rule is symmetric.

Lemma 4.5 *Trapezoidal rule is symmetric.*

Proof: Trapezoidal rule is

$$y_{n+1} = y_n + \frac{h}{2}\Big[f(y_n) + f(y_{n+1})\Big]$$

exchanging h by -h and (n+1) by (n)

$$\begin{aligned} y_n &= y_{n+1} - \frac{h}{2}\Big[f(y_{n+1}) + f(y_n)\Big] \\ y_{n+1} &= y_n + \frac{h}{2}\Big[f(y_n) + f(y_{n+1})\Big] \end{aligned}$$

then we proved that trapezoidal rule is symmetric.

4.8. Geometric Structures in Hamiltonian Systems

We can write the Hamiltonian systems of the form:

$$\dot{p} \;=\; -\nabla_q H(p,q), \qquad \dot{q} \;=\; \nabla_p H(p,q) \tag{4.40}$$

where $H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and the dimension d is the number of degrees of freedom. In applications the Hamiltonian is often given in the form:

$$H(p,q) \;=\; \frac{1}{2}p^T M(q)^{-1} p \;+\; U(q) \tag{4.41}$$

with a positive definite symmetric mass matrix $M(q)$ and a potential $U(q)$.In this situation, the function $H(p,q)$ represents the total energy of the system. Such problems arise in mechanics, astrophysics,molecular dynamics, and many other sciences (Hairer 2005).

Due to their special structure, Hamiltonian systems have several interesting properties (in the following we denote the flow of the system, mapping an initial value $y = (p,q)$ onto the solution at time t, by $\varphi_t(y)$):

(P1) the group property $\varphi_t \; o\varphi_s = \varphi_{t+s}$ is satisfied by every differential equation; in particular, one has

$$\varphi_t \; o\varphi_{-t} = \varphi_0 = identity \tag{4.42}$$

(P2) the Hamiltonian $H(p,q)$ is constant along solutions of (5.32) which means that the total energy is conserved quantity,

(P3) the flow φ_t of (4.40) is a symplectic transformation, i.e.

$$\varphi_t'(y)^T J\varphi_t'(y) = J \qquad for \quad t \geq 0, \qquad J = \begin{pmatrix} 0 & I \\ -I & o \end{pmatrix} \tag{4.43}$$

Due to det $\varphi_t'(y) = 1$, this implies the flow is volume preserving,

$$\mu(\varphi_t(A)) = \mu(A) \qquad t \geq 0$$

and for the system with one degree of freedom symplecticity turns out to be equivalent with area-preservation of the flow φ_t,

(P4) if $H(-p,q) = H(p,q)$, the flow φ_t is ρ-reversible with respect to the reflection $\rho(p,q) = (-p,q)$, i.e. it satisfies

$$(\rho \; o \; \varphi_t)(y) = (\varphi_t^{-1} \; o \; \rho)(y) \quad for\ all\ t\ and\ all\ y.$$

It is natural to look for numerical methods that satisfy one or several of these properties.

4.9. Geometric Integration

As we mentioned in chapter 3 that a numerical method for solving ordinary differential equations is a mapping Φ_h defined on the phase space that approximates the time-h flow φ_h; if $\Phi_h(y) = \varphi_h(y) + O(h^{r+1})$. The numerical approximation

at time $t = nh$ is obtained by $y_n = \Phi_h(y_{n-1})$ and the following are of interest:
(S1) the method is symmetric if it satisfies

$$\Phi_h \circ \Phi_{-h} = identity \tag{4.44}$$

(S2) it is energy-preserving if along numerical solutions of differential equation

$$H(p_n, q_n) = Constant \tag{4.45}$$

(S3) it is called symplectic if Φ_h satisfies

$$\Phi'_h(y)^T J \Phi'_h(y) = J \tag{4.46}$$

$$where \qquad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$$

(S4) it is $\rho - reversible$ if, for $H(-p, q) = H(p, q)$,

$$(\rho \circ \Phi_h)(y) = (\Phi_h^{-1} \circ \rho)(y) \tag{4.47}$$

for all h and for all y(Hairer 2005).

After these properties, we can briefly say that a numerical method that satisfies one or several of these properties is called a geometric integrator.

4.9.1. Philosophy of Geometric Integration

In any numerical study, one should
• examine any geometric or structural properties of ODE or its flow
• design numerical methods which also have these structural geometric properties
• examine the consequences, hopefully over and above the immediate ones
This encourages us to
• confront questions of phase space and degrees of freedom
• think about the significance of local, global and qualitative errors
• think about the kinds tools and functions allowed in numerical analysis.

For example, multi-step methods do not define a map on phase space, because more than one initial condition required. They can have geometric properties, but in a different phase space, which can alter the effects of the properties. This puts geometric integration firmly into the single step. If a system is defined on a sphere, one should stay on that sphere.

The direct consequences of geometric integration are that we are

• studying a dynamical system which is close to the true one, and the right class

• this class may have restricted orbit types, stability, and long-time behavior

In addition, because the structural properties are so natural, some indirect consequences have been observed. For example,

• symplectic integrators have good energy behaviors

• symplectic integrators can conserve angular momentum and other conserved quantities

• geometric integrators can have smaller local truncation errors for special problems and smaller global truncation errors for special problems/initial conditions

• some problems can have errors tending to zero at long times.

In designing our numerical method to preserve some structure we hope to see improvements in our computations. Geometric structures often make it easier to estimate errors, and in fact local and global errors may will be smaller for no extra computational expense. Geometric integration methods designed to capture specific qualitative properties may also preserve additional properties of the solution for free. For example symplectic methods for Hamiltonian problems have excellent energy conservation properties and can conserve angular momentum or other invariants.

In conclusion, geometric integration methods can often 'go where other methods cannot' and the motivation for preserving structure include

(i) it may yield methods that are faster, simpler, more stable, and more accurate, as in methods for structured eigenvalue problems for some types of ODEs;

(ii) it may yield methods that are more robust and yield qualitatively better results than standard methods, even if the standard numerical errors are no smaller;

(iii) it may suggest new types of calculations previously thought to be impossible, as in the long-time integration of Hamiltonian systems;

(iv) it may be essential to obtain a useful (e.g convergent) method, as in the discrete differential complexes used in electromagnetism;
(v) the development of general-purpose methods may be nearing completion, as in Runge-Kutta methods for ODEs (Hairer 2005).

4.10. The Kepler Problem and Its Conserved Quantities

In classical mechanics, Keplers problem is a special case of the two-body problem, in which the two bodies interact by a central force F that varies in strength as the inverse square of the distance r between them. The force may be either attractive or repulsive. The "problem" to be solved is to find the position or speed of the two bodies over time given their masses and initial positions and velocities. Using classical mechanics, the solution can be expressed as a Kepler orbit.

Consider two particles in $\mathbb{R}^3$. One is fixed at the origin, and the other one moves under the influence of gravitational field of the fixed particle. The problem, named after Kepler, is to describe the motion of the second particle. The following non-dimensional Kepler Problem is considered as a test problem to perform the geometric and non-geometric integration methods. We define the Kepler Hamiltonian system:

$$\dot{q} = p \tag{4.48}$$

$$\dot{p} = -G\frac{q}{|q|^3} \tag{4.49}$$

where q is position vector, p is momenta and G is the gravitational constant. In this section we consider the conserved quantities of the Kepler Problem such that the energy function, angular momentum and Runge-Lenz vector. The equations of three dimensional Kepler problem have the following first integrals:

1) Hamiltonian Function:

Total energy of the system is the Hamiltonian function

$$H(q_1, q_2, q_3, p_1, p_2, p_3) = \frac{1}{2}(p_1^2 + p_2^2 + p_3^2) - \frac{G}{\sqrt{q_1^2 + q_2^2 + q_3^2}} \tag{4.50}$$

We showed before that the energy is conserved.

2) Angular Momentum:

The system has not only the total energy $H(p,q)$ as a first integral, but also the angular momentum of a point particle with position $\overrightarrow{q}$ and momentum $\overrightarrow{p}$ is equal to the vector cross product of the position and momentum vectors

$$L = \overrightarrow{q} \times \overrightarrow{p} \tag{4.51}$$

Angular momentum is another conserved quantity for the Kepler problem. The time derivative of angular momentum is:

$$\begin{aligned} \frac{dL}{dt} &= \frac{dq}{dt} \times p + q \times \frac{dp}{dt} \\ &= p \times p - \frac{K}{|q|^3} q \times q = 0 \end{aligned}$$

3) Runge-Lenz Vector:

Another conserved quantity of the Kepler Problem is Runge-Lenz vector. For a single particle acted on by an inverse-square central force described by the equation $F(|q|) = \frac{-k}{|q|^2}\hat{q}$, the Runge-Lenz vector A is defined mathematically by the formula:

$$A = p \times L - mk\hat{q}$$

where

- m is the mass of the point particle moving under the central force,
- p is its momentum vector,
- $L = q \times p$ is its angular momentum vector,
- k is a parameter that describes strength of the central force,
- q is the position vector of the particle,
- $\hat{q}$ is the corresponding unit vector, i.e, $\hat{q} = \frac{q}{|q|}$.

The Runge-lenz vector is conserved quantity of the Kepler problem since:
A central force F acting on the particle is $F = f(|q|).\frac{q}{|q|} = f(|q|).\hat{q}$. For some function $f(|q|)$ of the radius q. Since $\frac{dL}{dt} = 0$

$$\frac{d}{dt}(p \times L) = \frac{dp}{dt} \times L = f(|q|).\hat{q} \times (q \times p)$$

where the momentum $p = m\frac{dq}{dt}$.

$$\frac{d}{dt}(p \times L) = f(|q|).\frac{q}{|q|} \times (q \times m\frac{dq}{dt})$$

Using Lagrange's formula

$$q \times (q \times \frac{dq}{dt}) = q(q.\frac{dq}{dt}) - |q|^2 \frac{dq}{dt}$$

then

$$\frac{d}{dt}(p \times L) = f(|q|).\frac{m}{|q|}[q(q.\frac{dq}{dt}) - |q|^2 \frac{dq}{dt}]$$

the identity

$$\frac{d}{dt}(q.q) = 2q.\frac{dq}{dt} = \frac{d}{dt}(|q|^2) = 2|q|.\frac{d|q|}{dt}$$

yields the equation

$$\begin{aligned}\frac{d}{dt}(p \times L) &= -mf(|q|).|q|^2[\frac{1}{|q|}.\frac{dq}{dt} - \frac{q}{|q|^2}.\frac{d|q|}{dt}] \\ &= -mf(|q|).|q|^2.\frac{d}{dt}(\frac{q}{|q|})\end{aligned}$$

for the special case of an inverse square central force $f(|q|) = -\frac{k}{|q|^2}$,

$$\frac{d}{dt}(p \times L) = mk\frac{d}{dt}(\frac{q}{|q|}) = \frac{d}{dt}(mk\hat{q})$$

then the time derivative of the Runge-Lenz vector becomes

$$\frac{dA}{dt} = \frac{d}{dt}(p \times L - mk\hat{q}) = \frac{d}{dt}(p \times L) - \frac{d}{dt}(mk\hat{q}) = \frac{d}{dt}(mk\hat{q}) - \frac{d}{dt}(mk\hat{q}) = 0$$

In next chapter we will apply both geometric and non-geometric integration methods to the Kepler problem and we will compare the results that which of these methods conserve better above conserved quantities.

CHAPTER 5

A COMPARISON OF SYMPLECTIC AND NON-SYMPLECTIC METHODS APPLIED TO KEPLER PROBLEM

In this chapter both geometric and non-geometric integrator methods are applied to the Kepler problem. For these numerical experiments we use the initial conditions $q_0 = (0.8, 0.6, 0)^T \quad and \quad p_0 = (0, 1, 0.5)^T$. All computations are performed on the time interval [0,1000] with time step $\triangle t = h = 0.01$. All these initial conditions and fictitious time are taken from Kozlov's paper (Kozlov 2007). Algorithms are written in MATLAB. Conservation of the trajectory of motion and conserved quantities of the Kepler Problem such that energy, angular momentum and Runge-Lenz vector are compared by using geometric and non-geometric integrator methods.

5.1. First Order Methods

First we consider first order methods, namely, explicit Euler, implicit Euler, symplectic Euler and its adjoint.

5.1.1. Phase Spaces

In this subsection trajectory of motion obtained by using explicit Euler, implicit Euler is shown in Figure(5.1) and Fig (5.2), respectively. Since they are not geometric integration methods, they do not preserve the shape of the trajectory of the motion. Figure(5.3) and Figure(5.4) present the trajectory of motion obtained by symplectic Euler and adjoint of symplectic Euler, respectively. Since both of these methods are symplectic geometric integrators the shape of the trajectory preserved by both these methods.

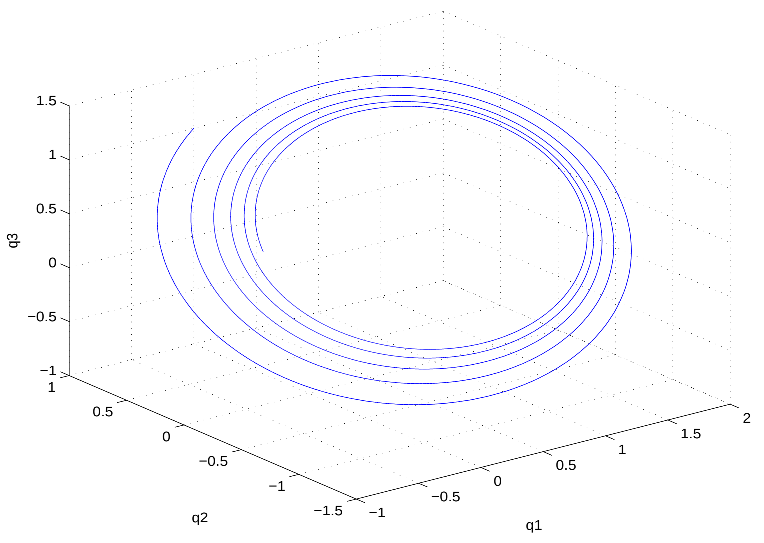

Figure 5.1. Trajectory of motion by Explicit Euler.

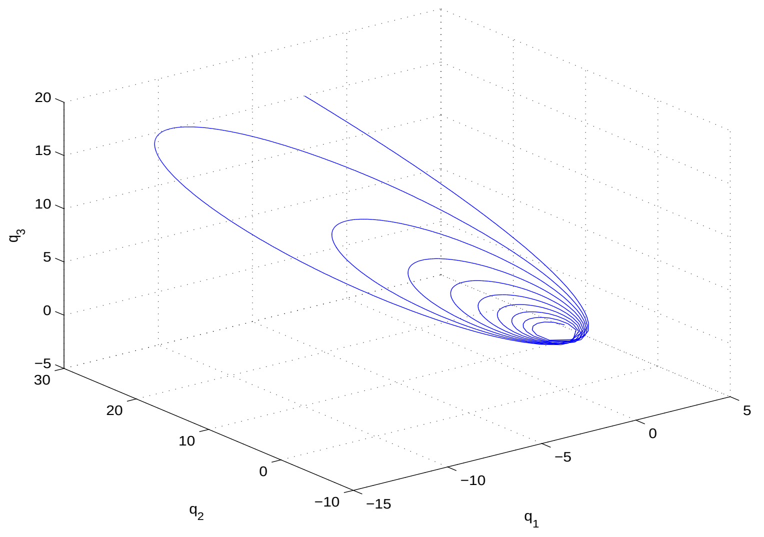

Figure 5.2. Trajectory of motion by Implicit Euler.

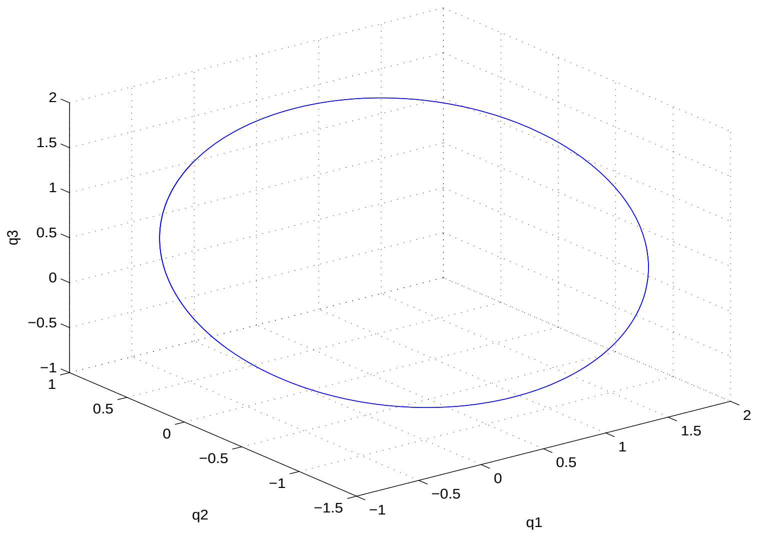

Figure 5.3. Trajectory of motion by Symplectic Euler.

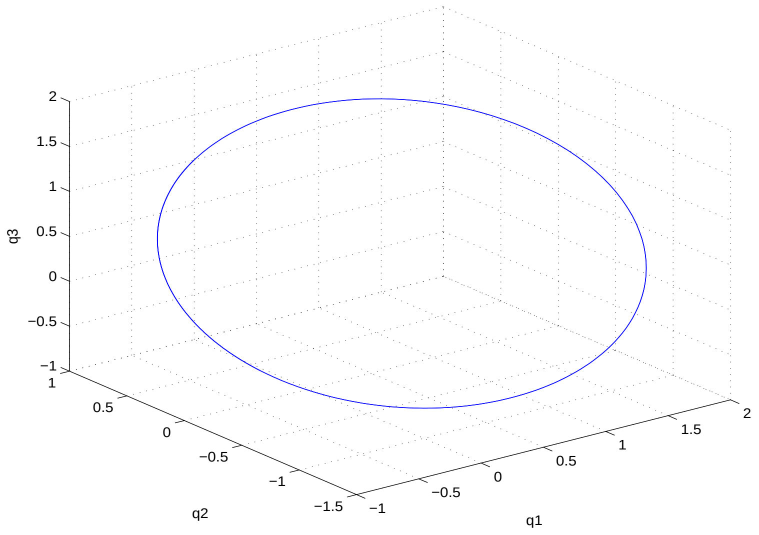

Figure 5.4. Trajectory of motion by Adjoint of Symplectic Euler.

5.1.2. Conservation of Energy

Conservation of the energy is presented in Figure(5.5, 5.6, 5.7, 5.8) by explicit Euler, implicit Euler, symplectic Euler and adjoint of the symplectic Euler method, respectively. Since symplectic Euler and adjoint of symplectic Euler methods are symplectic methods we observed that they conserved the energy better than explicit and implicit Euler methods.

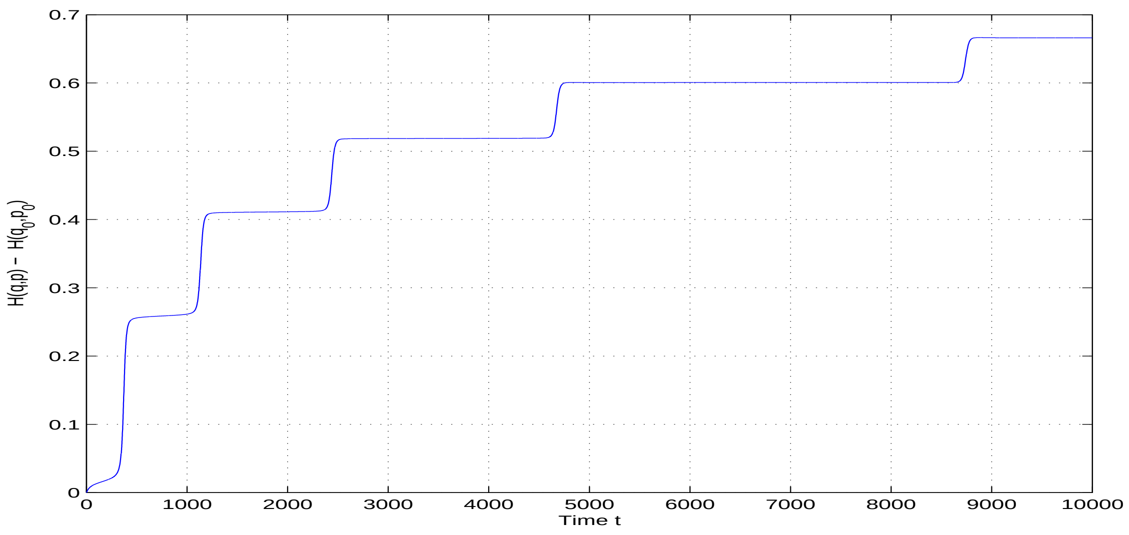

Figure 5.5. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for the Explicit Euler.

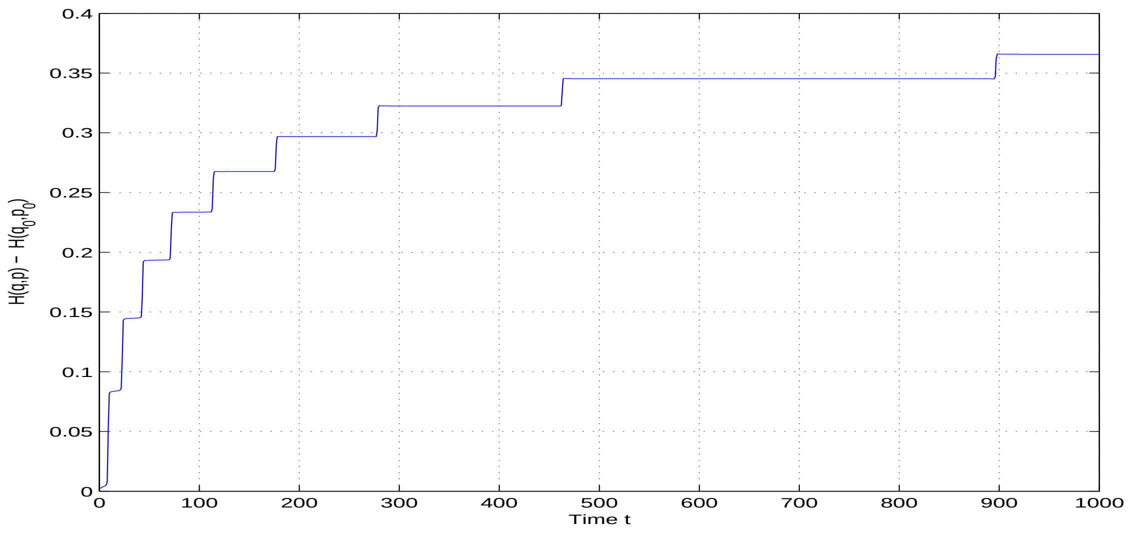

Figure 5.6. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for the Implicit Euler.

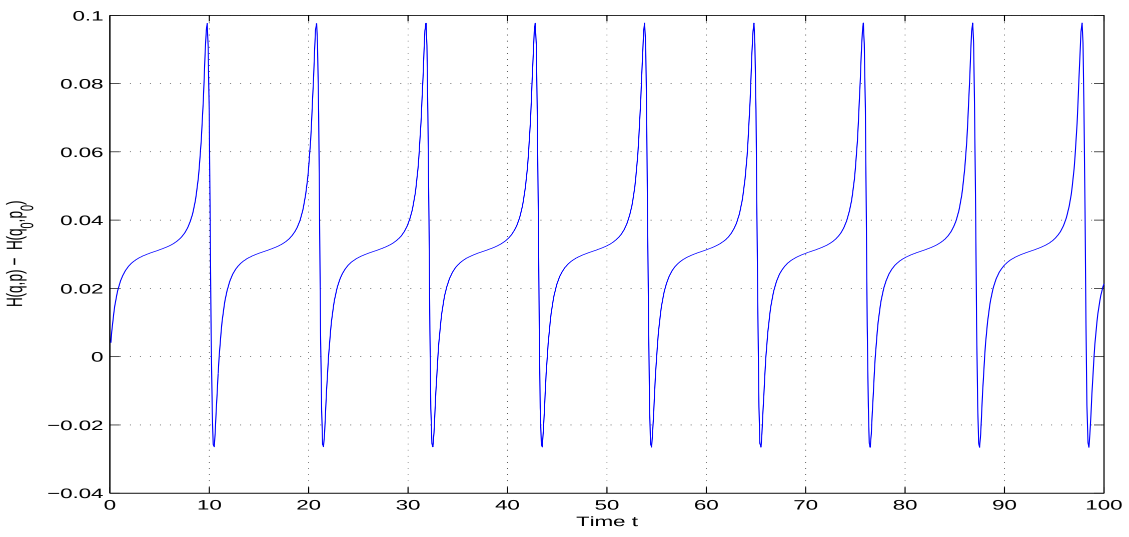

Figure 5.7. The errors in Hamiltonian $H(q, p) - H(q_0, p_0)$ for the Symplectic Euler.

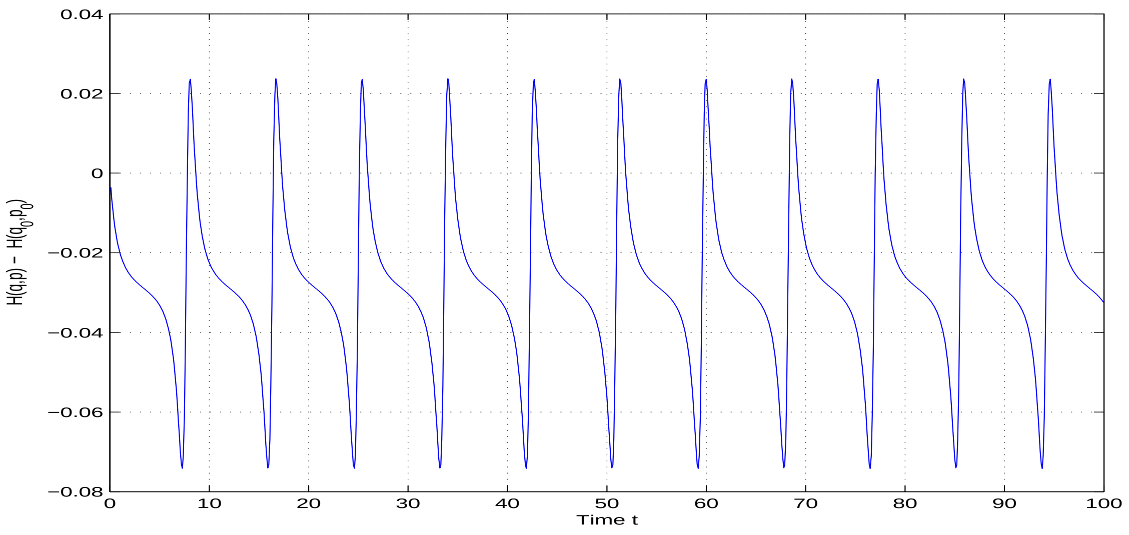

Figure 5.8. The errors in Hamiltonian $H(q, p) - H(q_0, p_0)$ for the Adjoint of Symplectic Euler.

5.1.3. Conservation of Angular Momentum

Conservation of angular momentum is shown in Figure(5.9, 5.10, 5.11, 5.12) by explicit Euler, implicit Euler, symplectic Euler and adjoint of symplectic Euler, respectively. We observed the following: angular momentum is preserved by the symplectic methods but not by any of the explicit and implicit Euler methods.

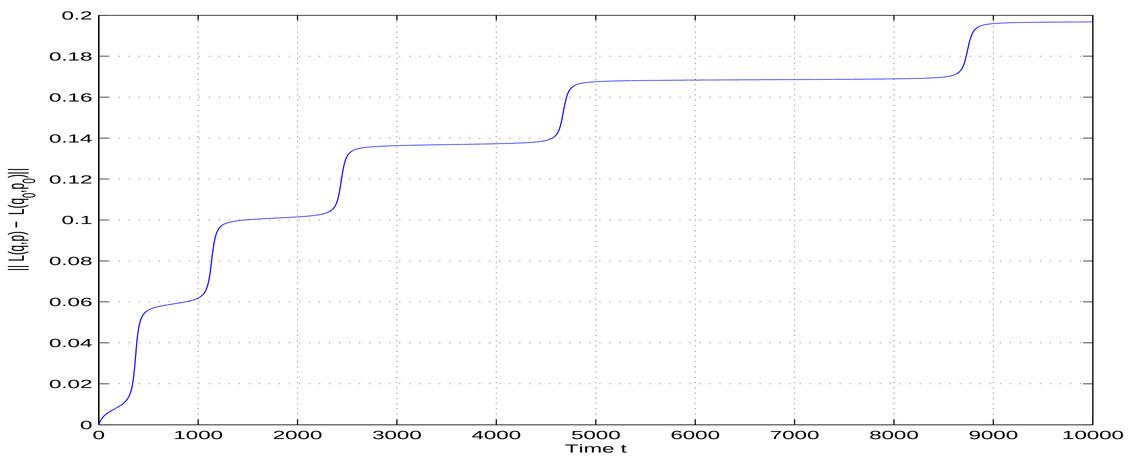

Figure 5.9. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for the Explicit Euler.

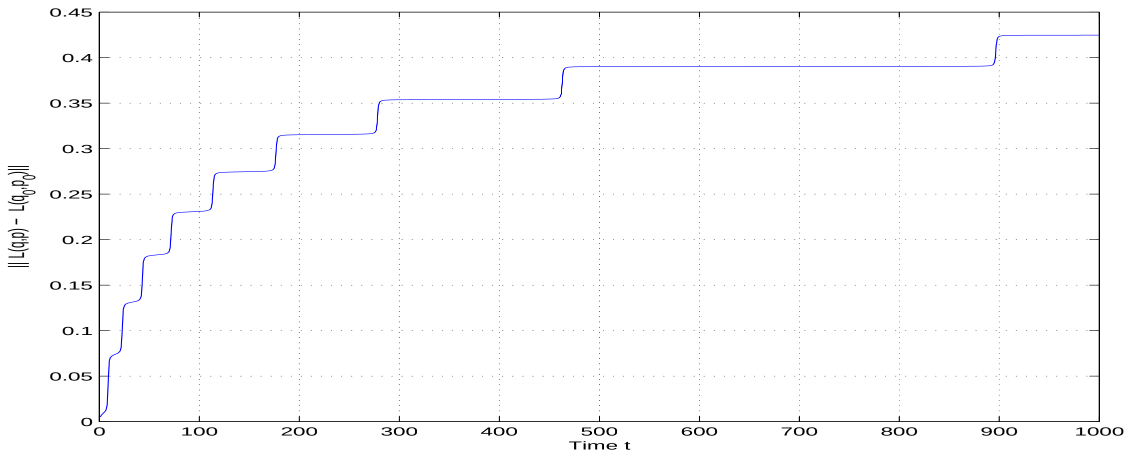

Figure 5.10. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for the Implicit Euler.

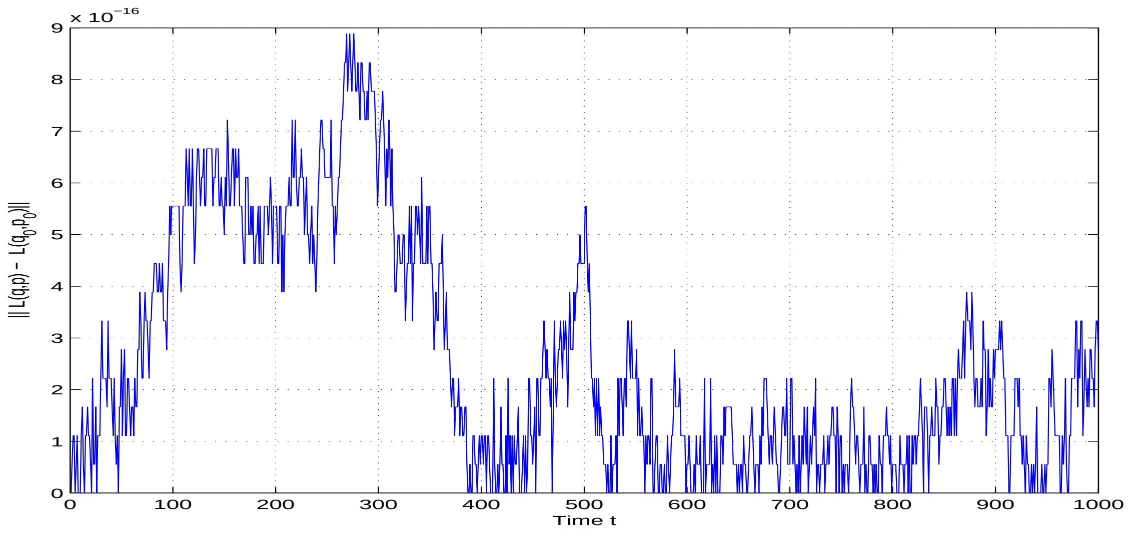

Figure 5.11. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for the Symplectic Euler.

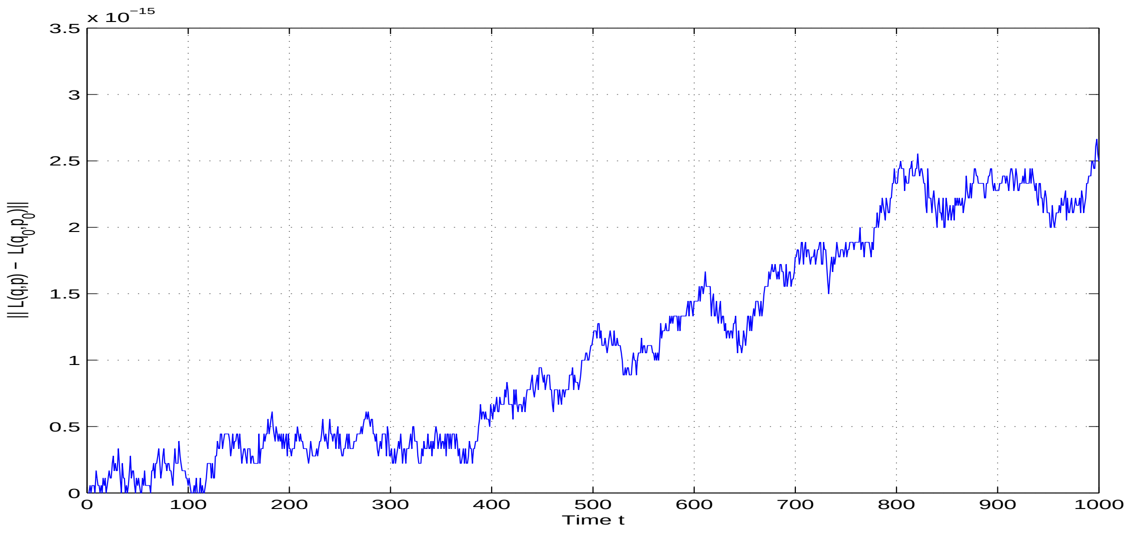

Figure 5.12. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for the Adjoint of Symplectic Euler.

5.1.4. Conservation of Runge-Lenz Vector

Conservation of Runge-Lenz vector is presented in Figure(5.13, 5.14, 5.15, 5.16) by explicit Euler, implicit Euler, symplectic Euler and adjoint of symplectic Euler, respectively. It can be seen in these figures that symplectic methods preserve this quantity better than explicit and implicit Euler method.

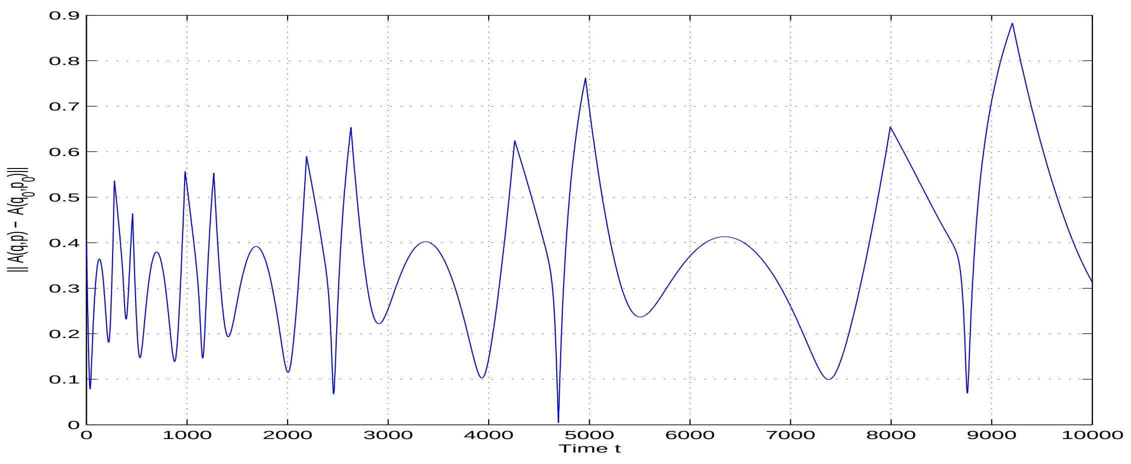

Figure 5.13. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for the Explicit Euler.

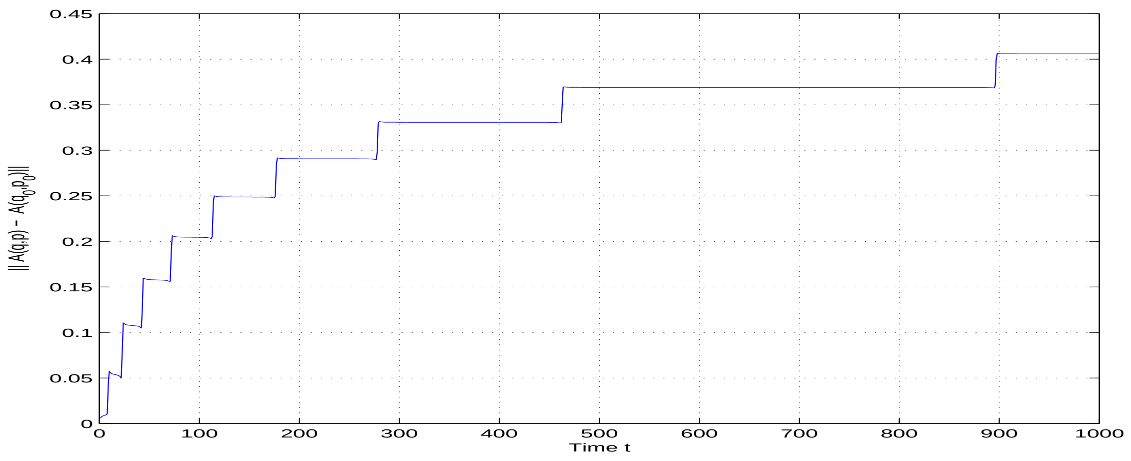

Figure 5.14. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for the Implicit Euler .

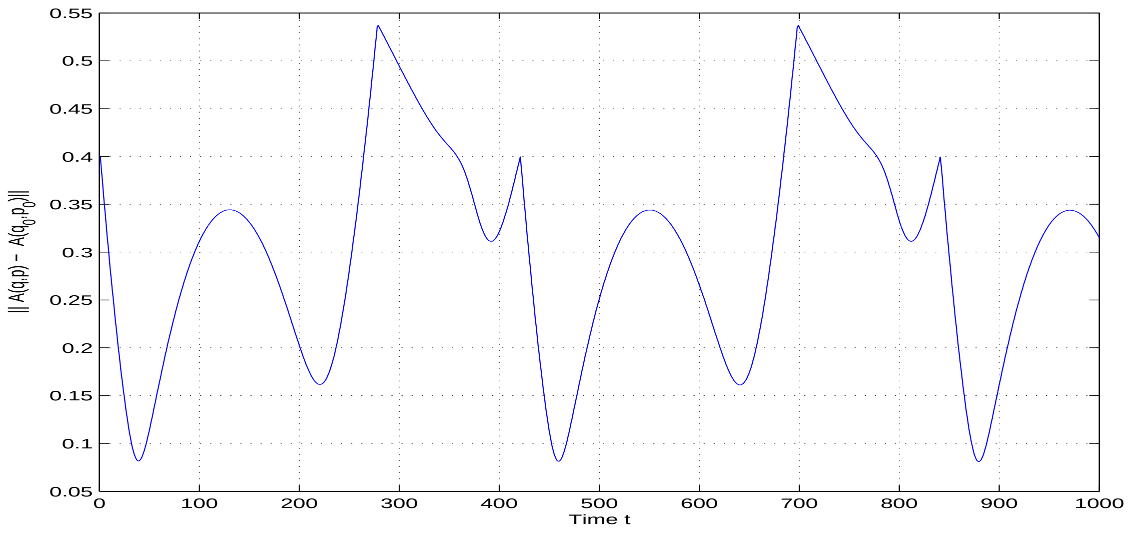

Figure 5.15. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for the Symplectic Euler.

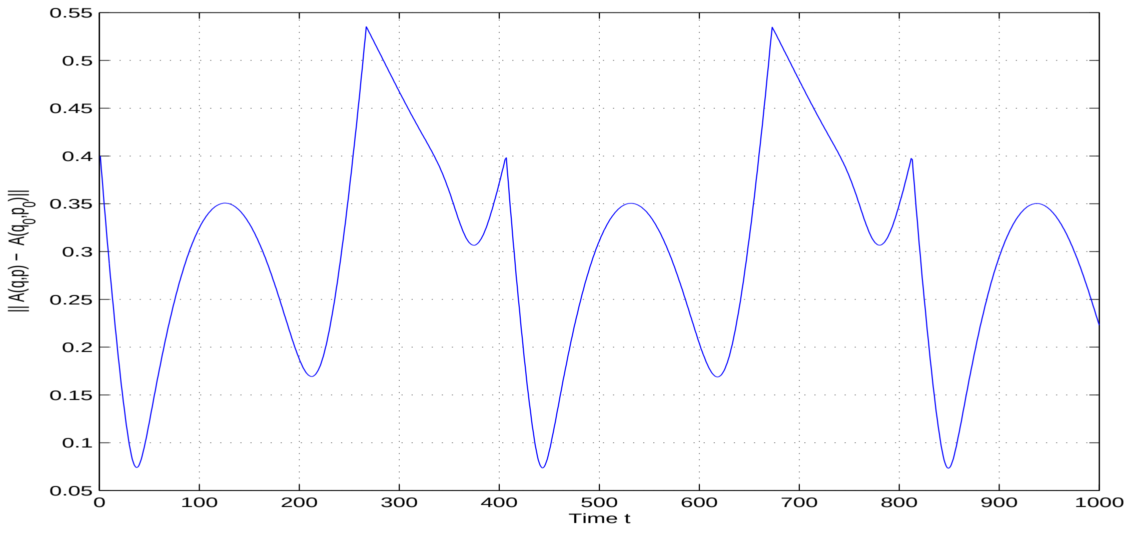

Figure 5.16. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for the Adjoint of Symplectic Euler.

5.2. Second Order Methods

In this section we consider second order methods, namely, trapezoidal rule, implicit midpoint rule and Störmer-Verlet method.

5.2.1. Phase Spaces

In this subsection we obtain trajectory of motion by using trapezoidal rule, implicit midpoint rule and Störmer-Verlet method. Figure(5.17) presents the trajectory of motion obtained by trapezoidal rule. It can be easily seen that the trajectory of motion gets deformed. Since trapezoidal rule is a non-geometric integrator we could not obtain closed curves. Figure(5.18) and Figure(5.19) show the phase space of the Kepler problem obtained by implicit midpoint and Störmer-Verlet method, respectively. Since both of these methods are symplectic geometric integrators the shape of the trajectory preserved by both these methods.

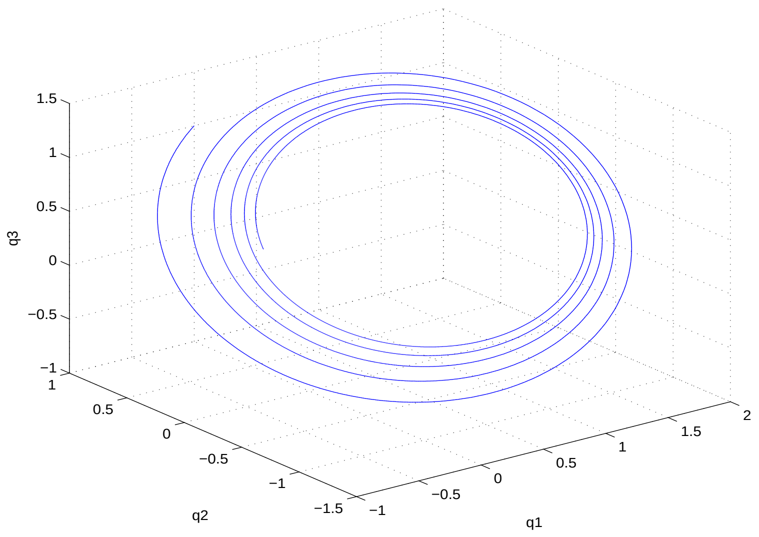

Figure 5.17. Trajectory of motion by Trapezoidal Rule.

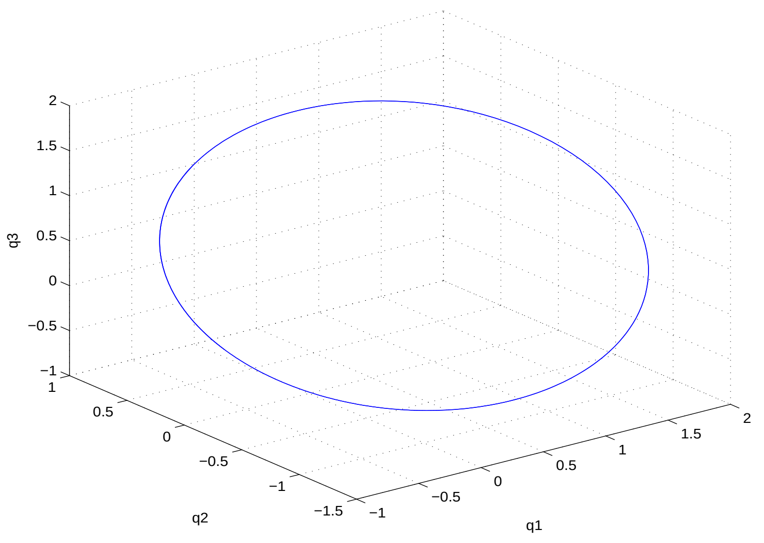

Figure 5.18. Trajectory of motion by Midpoint Rule.

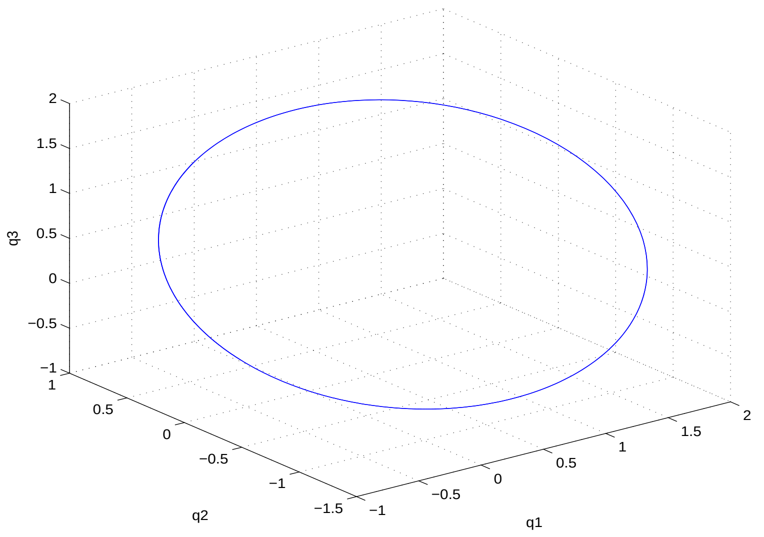

Figure 5.19. Trajectory of motion by Störmer-Verlet scheme.

5.2.2. Conservation of Energy

Figure(5.20, 5.21, 5.22) present conservation of the energy by trapezoidal rule, implicit midpoint rule and Störmer-Verlet method, respectively. We observed the following: implicit midpoint rule and Störmer-Verlet method conserved the energy since they are symplectic geometric integration methods. Conservation of the energy by these methods is better than trapezoidal rule.

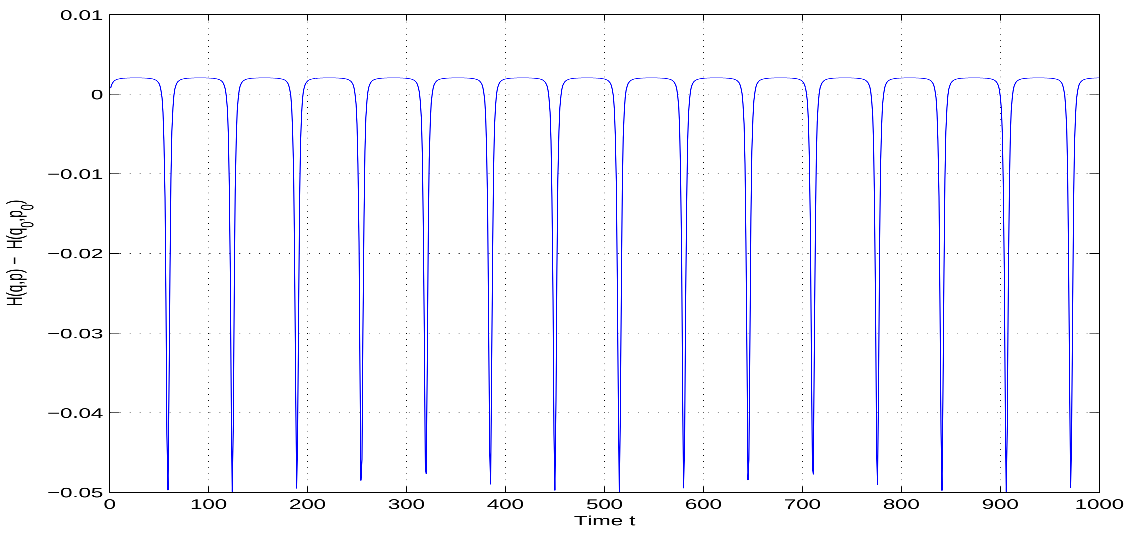

Figure 5.20. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for the Trapezoidal Rule.

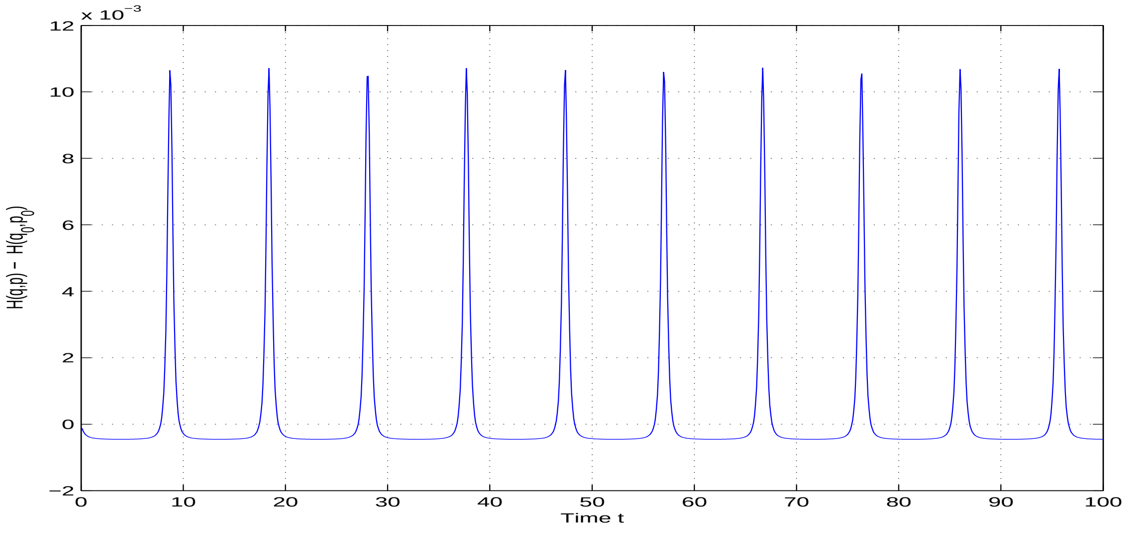

Figure 5.21. The errors in Hamiltonian $H(q, p) - H(q_0, p_0)$ for Midpoint Rule.

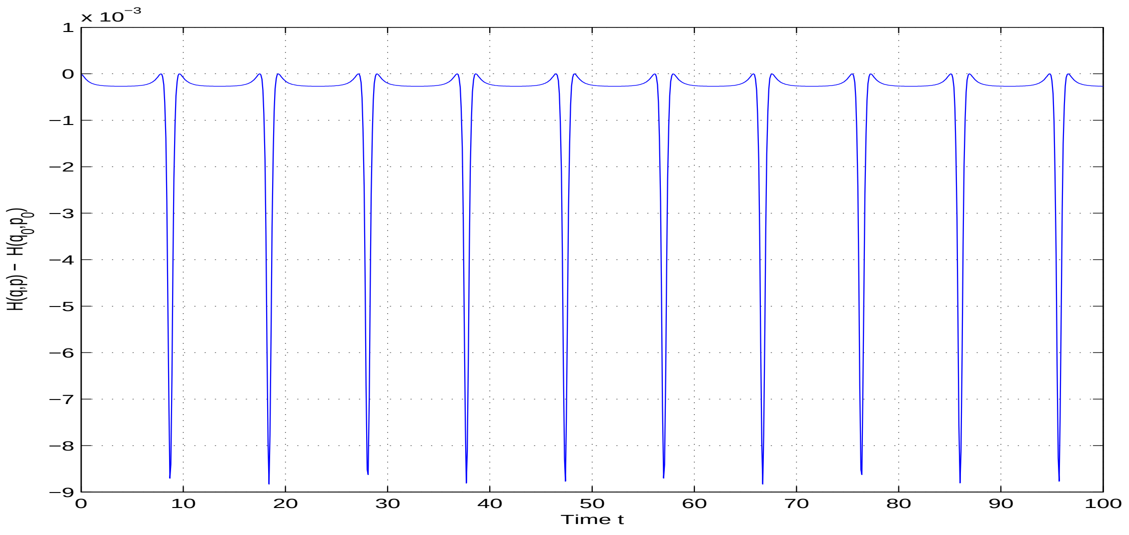

Figure 5.22. The errors in Hamiltonian $H(q, p) - H(q_0, p_0)$ for Störmer-Verlet scheme.

5.2.3. Conservation of Angular Momentum

Error in angular momentum is shown in Figure(5.23, 5.24, 5.25) by trapezoidal rule, implicit midpoint rule and Störmer-Verlet method, respectively. Angular momentum is conserved by midpoint rule and Störmer-Verlet method up to the computer accuracy. Implicit midpoint rule and Störmer-Verlet method conserve this quantity better than trapezoidal rule.

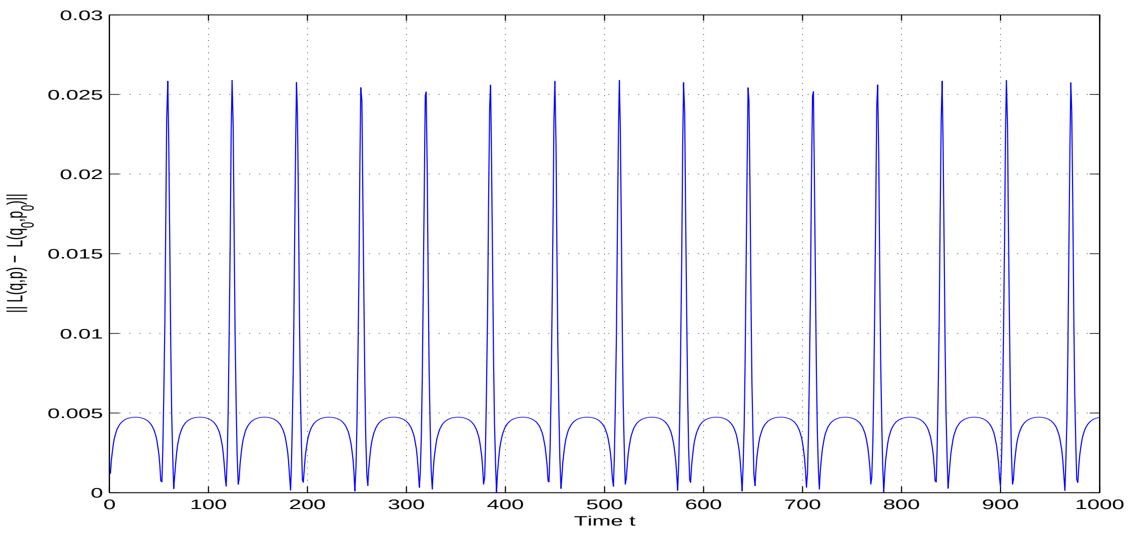

Figure 5.23. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for Trapezoidal rule.

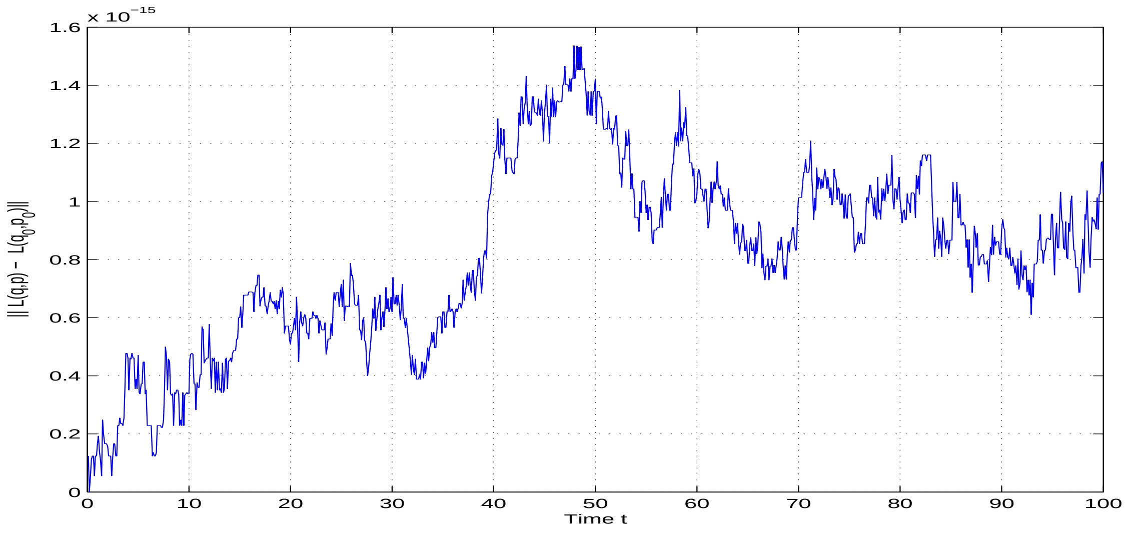

Figure 5.24. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ Midpoint Rule.

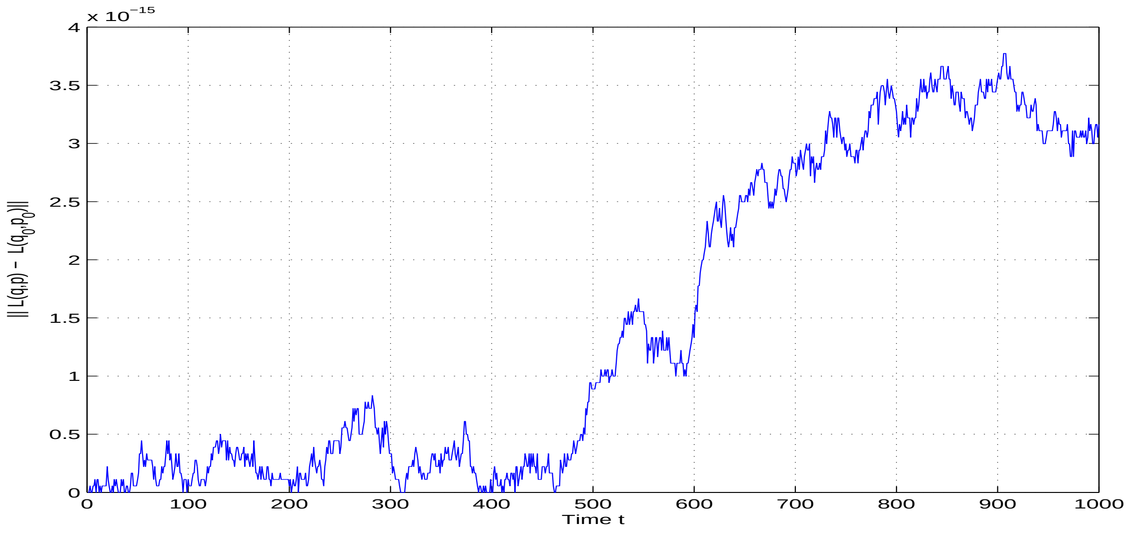

Figure 5.25. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for Störmer-Verlet scheme.

5.2.4. Conservation of Runge-Lenz Vector

Conservation of Runge-Lenz vector is shown in Figure(5.26, 5.27, 5.28) by trapezoidal rule, implicit midpoint rule and Störmer-Verlet method, respectively. As can be seen in these figures that implicit midpoint rule and Störmer-Verlet method preserve the Runge-Lenz vector better than trapezoidal rule.

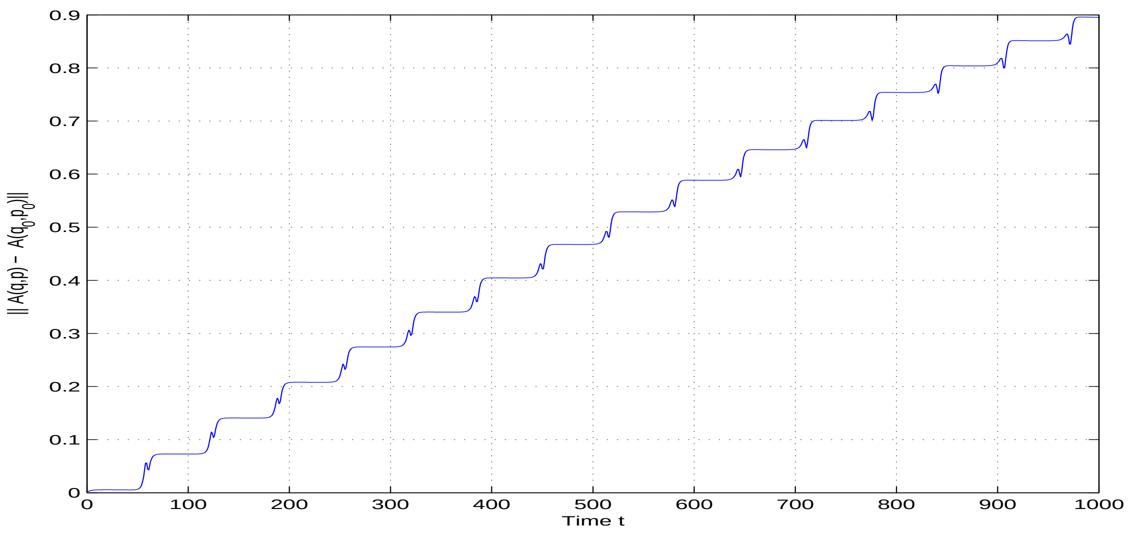

Figure 5.26. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for the Trapezoidal rule.

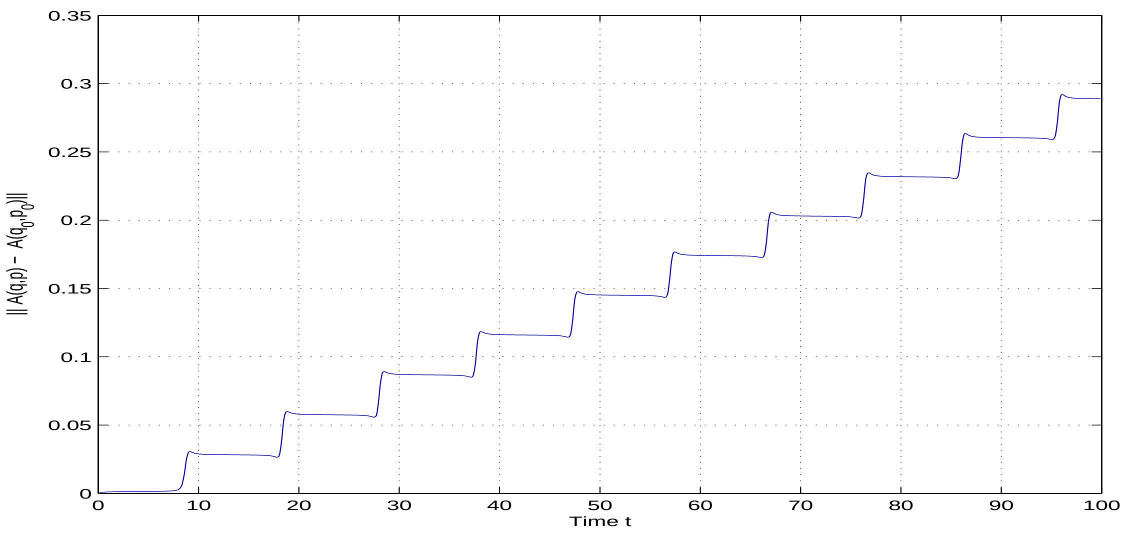

Figure 5.27. The errors in Runge-Lenz vector $A(q, p) - A(q_0, p_0)$ for Midpoint Rule.

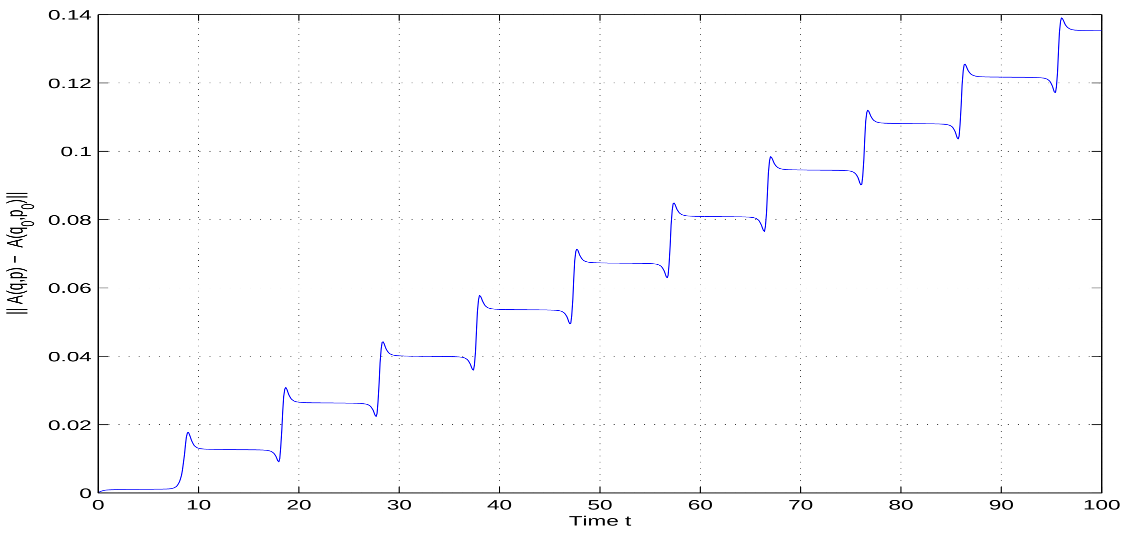

Figure 5.28. The errors in Runge-Lenz vector $A(q, p) - A(q_0, p_0)$ for Störmer-Verlet scheme.

5.3. Third Order Methods

In this section classic 3rd order Runge-Kutta method as we mention in section 3.4 and composition methods of order 3 are considered as third order methods. In composition methods we use midpoint rule and Störmer-Verlet method as the base methods such that let Φ_h be our base method and $\gamma_1 = \gamma_3 = \frac{1}{2-2^{(1/3)}}$, $\gamma_2 = -\frac{2^{(1/3)}}{2-2^{(1/3)}}$ real numbers. Its composition with step sizes $\gamma_1 h, \gamma_2 h, \gamma_3 h$ i.e.,

$$\Psi_h = \Phi_{\gamma_3 h} \, o \, \Phi_{\gamma_2 h} \, o \, \Phi_{\gamma_1 h}$$

is the corresponding composition method (Hairer, et al. 2001).

5.3.1. Phase Spaces

In this subsection trajectory of motion obtained by classic 3rd order Runge-Kutta method can be seen in Figure(5.29). Since 3rd order Runge-Kutta method is a non-geometric integrator we could not obtain closed curves. Figure(5.30) and Figure(5.31) present the trajectory of motion obtained by composition method of order 3 (midpoint rule as base method) and composition method of order 3 (Störmer-Verlet as base method), respectively. Since composition of symplectic method is symplectic method the shape of the trajectory preserved by this method.

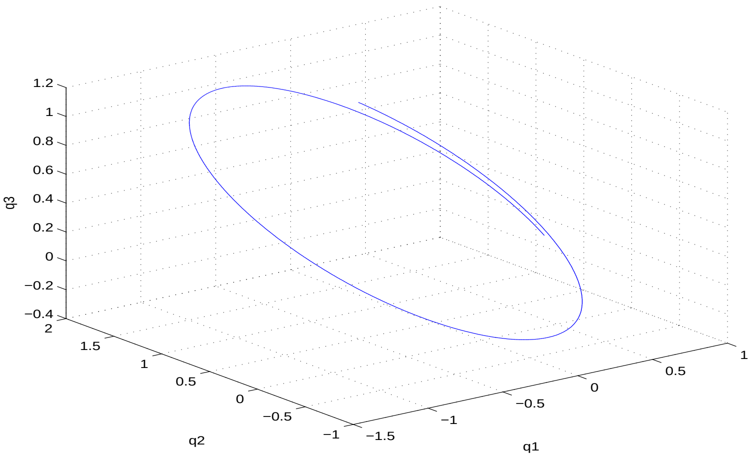

Figure 5.29. Trajectory of motion by 3rd order Runge-Kutta.

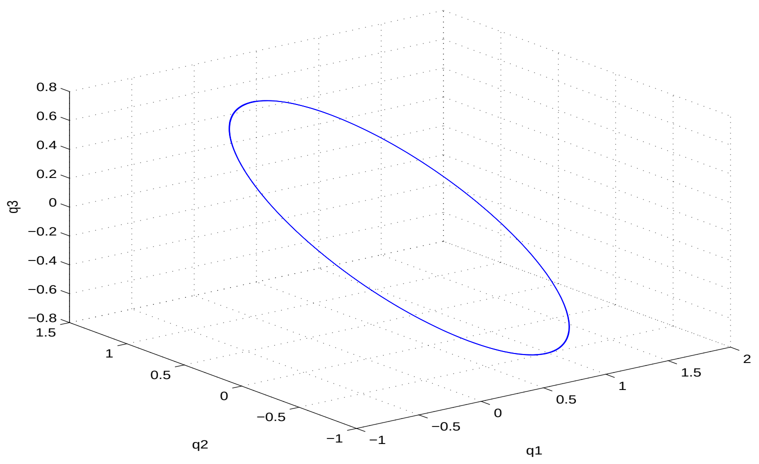

Figure 5.30. Trajectory of motion by Composition method of order 3 (midpoint rule as base method).

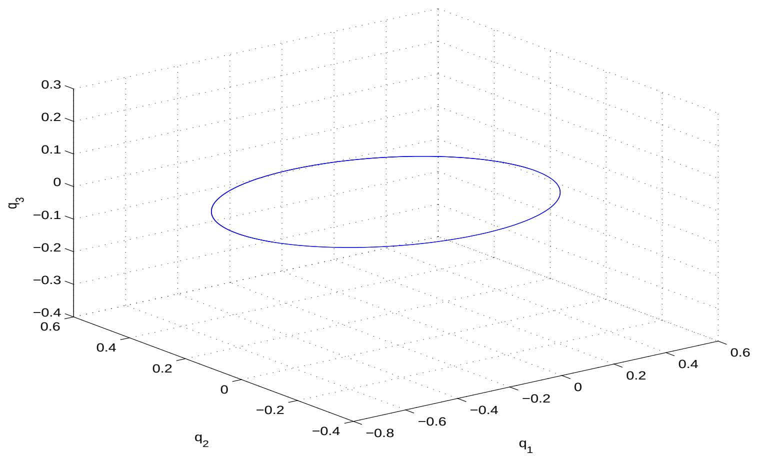

Figure 5.31. Trajectory of motion by Composition method of order 3 (Störmer-Verlet as base method).

5.3.2. Conservation of Energy

Conservation of the energy is presented in Figure(5.32, 5.33, 5.34) by 3rd order Runge-Kutta, composition method of order 3 (midpoint rule as base method) and composition method of order 3 (Störmer-Verlet as base method), respectively. We observed following: composition method of order 3 preserve the energy better than 3rd order Runge-Kutta method.

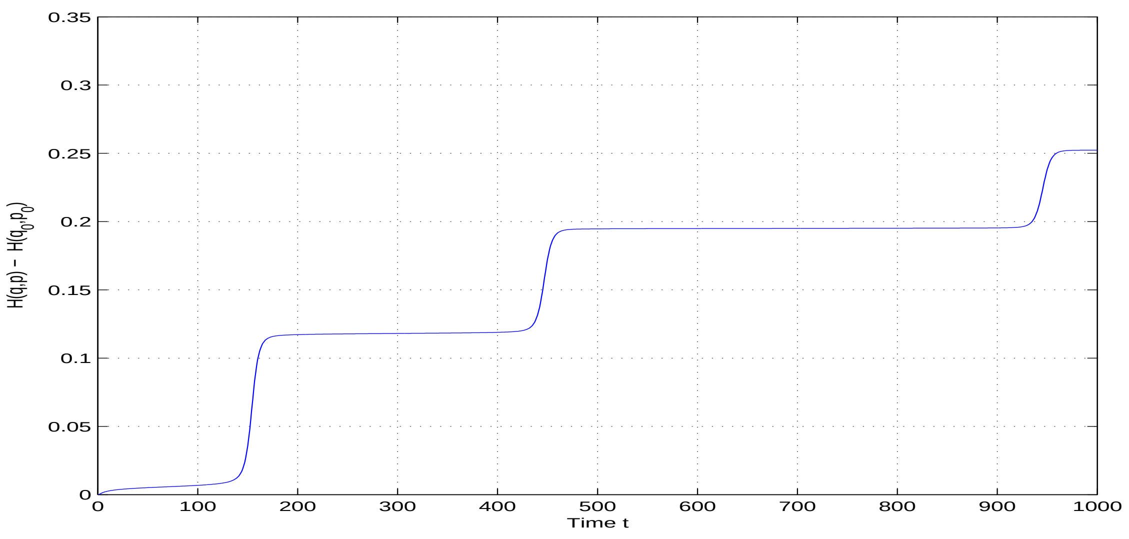

Figure 5.32. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for 3rd order Runge-Kutta.

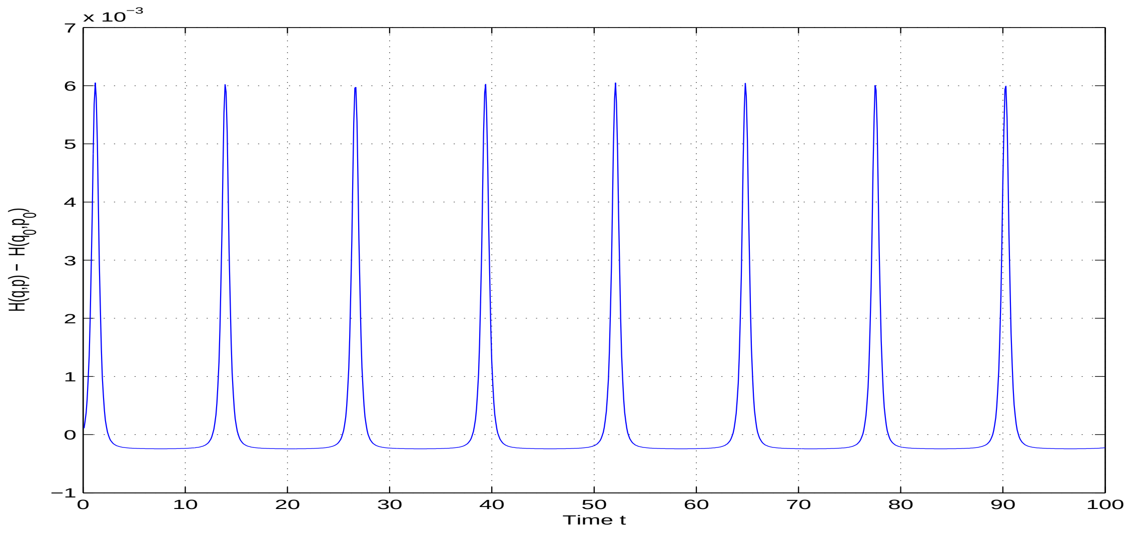

Figure 5.33. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for Composition method of order 3 (midpoint rule as base method).

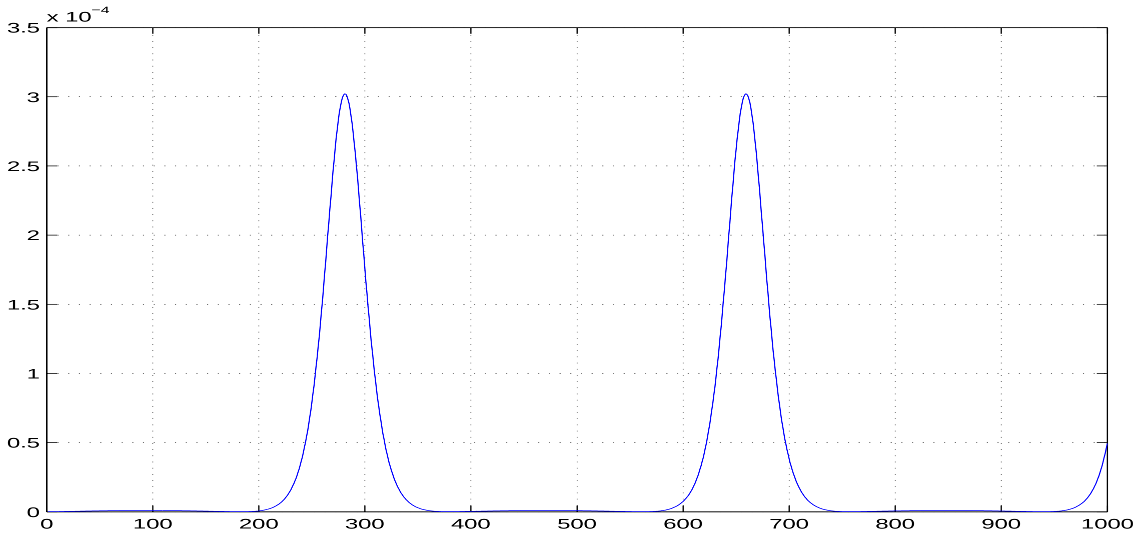

Figure 5.34. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for Composition method of order 3 (Störmer-Verlet as base method).

5.3.3. Conservation of Angular Momentum

In this subsection Figure(5.35, 5.36, 5.37) show conservation of angular momentum by 3rd order Runge-Kutta, composition method of order 3 (midpoint rule as base method) and composition method of order 3 (Störmer-Verlet as base method), respectively. Composition method of order 3 (midpoint rule as base method) and composition method of order 3 (Störmer-Verlet as base method) conserved this quantity up to the computer accuracy.

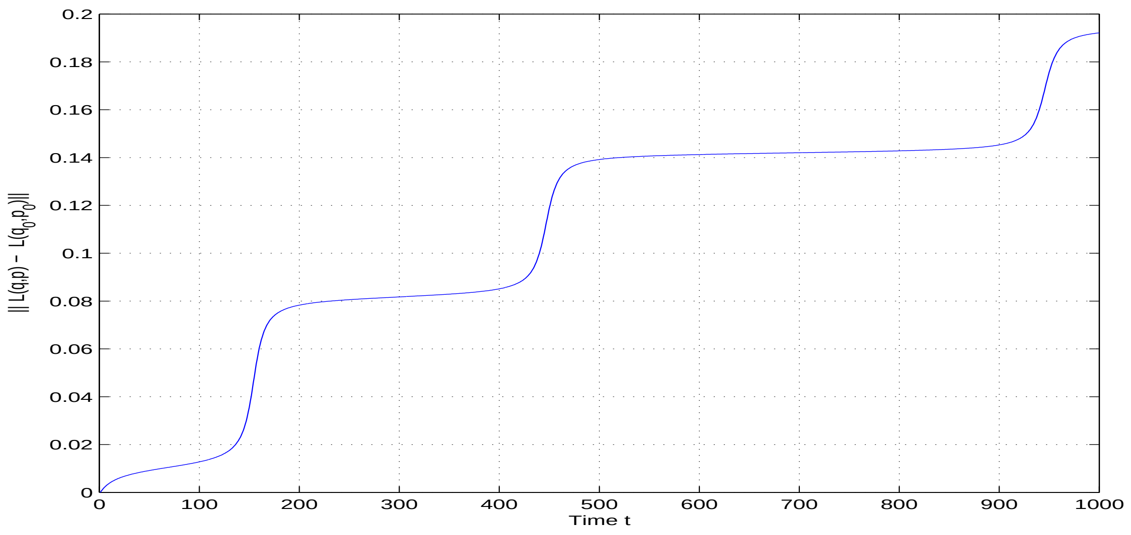

Figure 5.35. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for 3rd order Runge-Kutta.

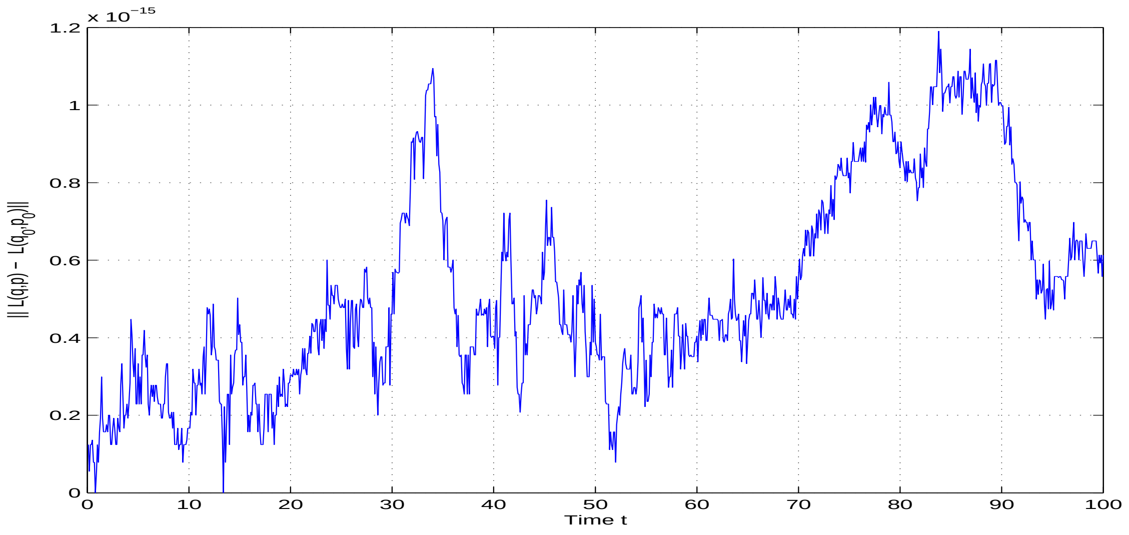

Figure 5.36. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ Composition method of order 3 (midpoint rule as base method).

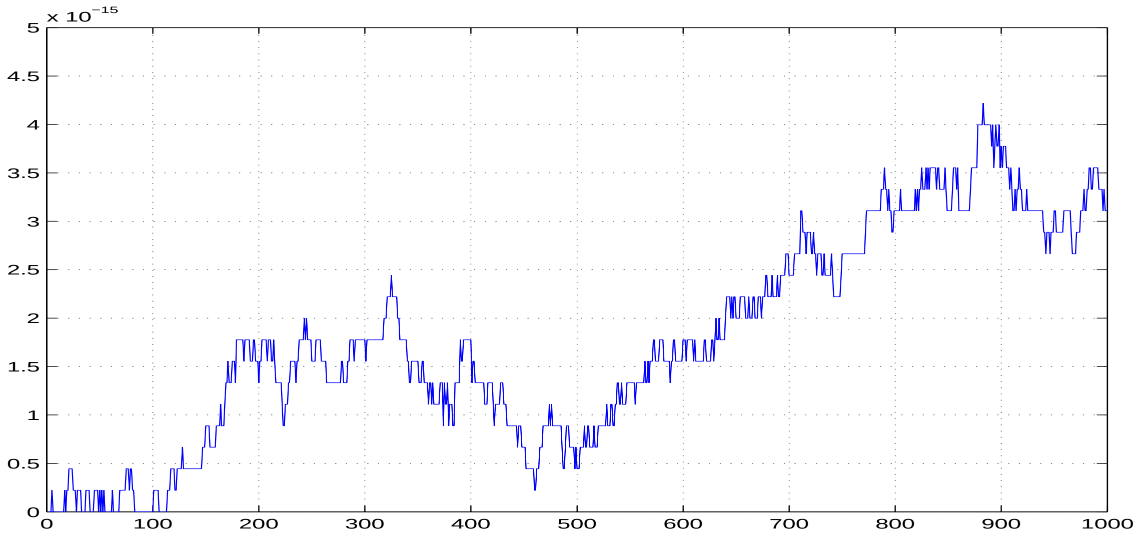

Figure 5.37. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ Composition method of order 3 (Störmer-Verlet as base method).

5.3.4. Conservation of Runge-Lenz Vector

Conservation of Runge-Lenz vector is shown in Figure(5.38, 5.39, 5.40) by 3rd order Runge-Kutta, composition method of order 3 (midpoint rule as base method) and composition method of order 3 (Störmer-Verlet as base method) respectively. It can be seen in below figures that composition method of order 3 preserve the Runge-Lenz vector better than 3rd order Runge-Kutta method.

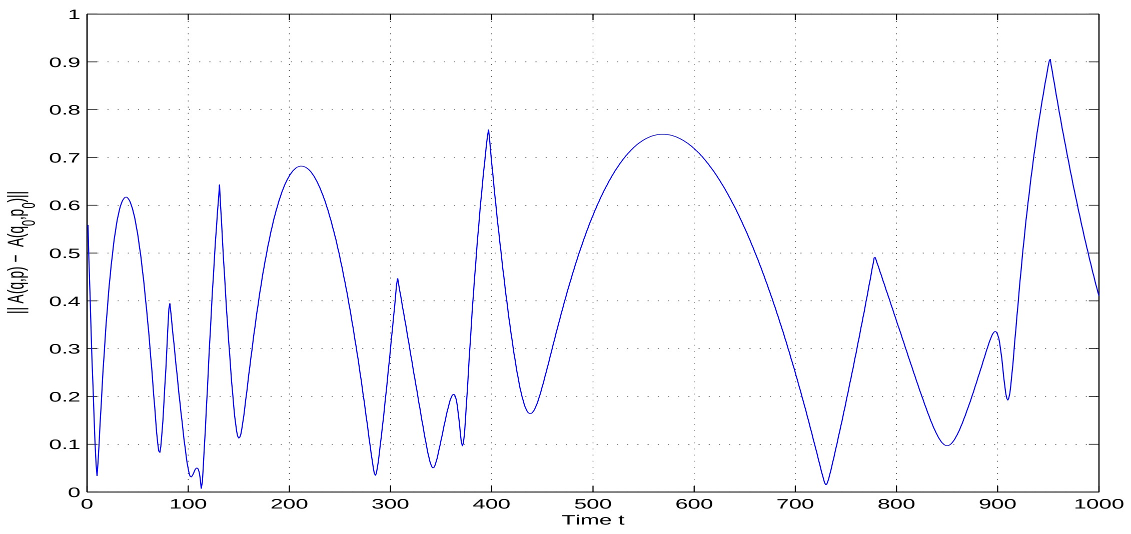

Figure 5.38. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for 3rd order Runge-Kutta.

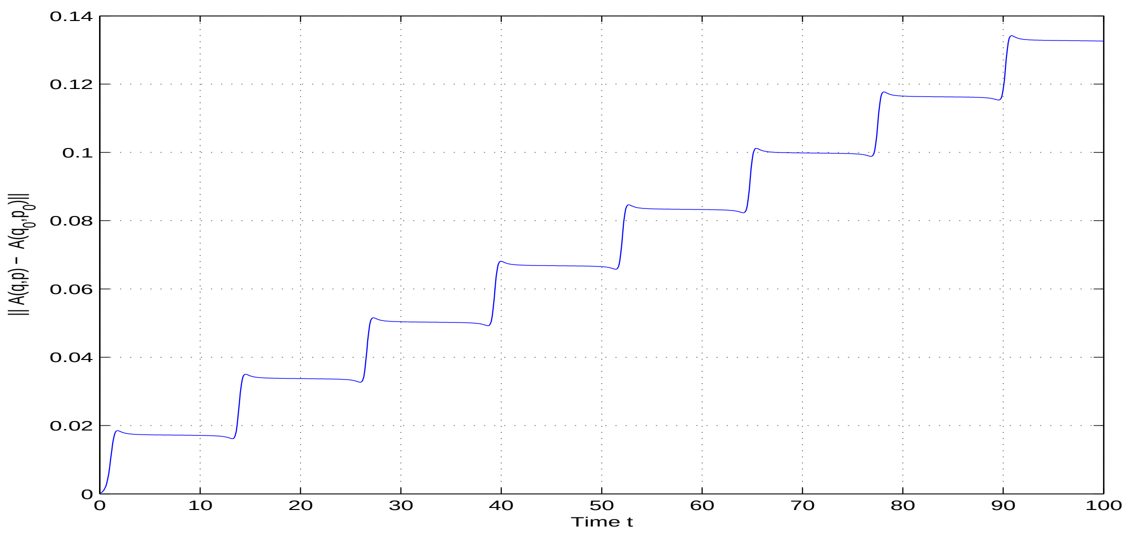

Figure 5.39. The errors in Runge-Lenz vector $A(q, p) - A(q_0, p_0)$ for Composition method of order 3 (midpoint rule as base method).

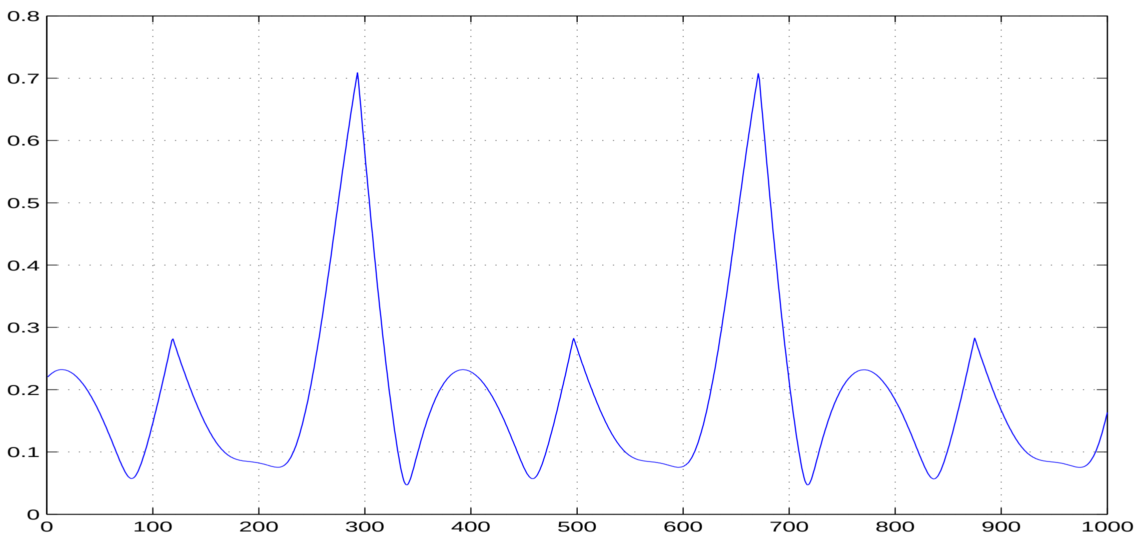

Figure 5.40. The errors in Runge-Lenz vector $A(q, p) - A(q_0, p_0)$ for Composition method of order 3 (Störmer-Verlet as base method).

5.4. Fourth and Fifth Order Methods

In this section we consider fourth and fifth order methods, namely, classic 4th order Runge-Kutta method that we mentioned in chapter 3.4 and composition methods of order 5. In composition methods we use midpoint rule and Störmer-Verlet method as base methods. In composition methods we use step sizes $\gamma_1 = \gamma_2 = \gamma_4 = \gamma_5 = \gamma_1 = \frac{1}{4-4^(1/5)}$ *and* $\gamma_3 = -\frac{4^(1/5)}{4-4^(1/5)}$.

5.4.1. Phase Spaces

In this subsection trajectory of motion obtained by classic 4th order Runge-Kutta method is shown in Figure(5.41). Since 4th order Runge-Kutta is a non-geometric integrator we could not obtain closed curves. Figure(5.42) and Figure(5.43) present trajectory of motion obtained by composition method of order 5 (midpoint rule as base method) and composition method of order 5 (Störmer-Verlet as base method), respectively. Since composition method of order 5 are symplectic geometric integrator the shape of the trajectory preserved by this method.

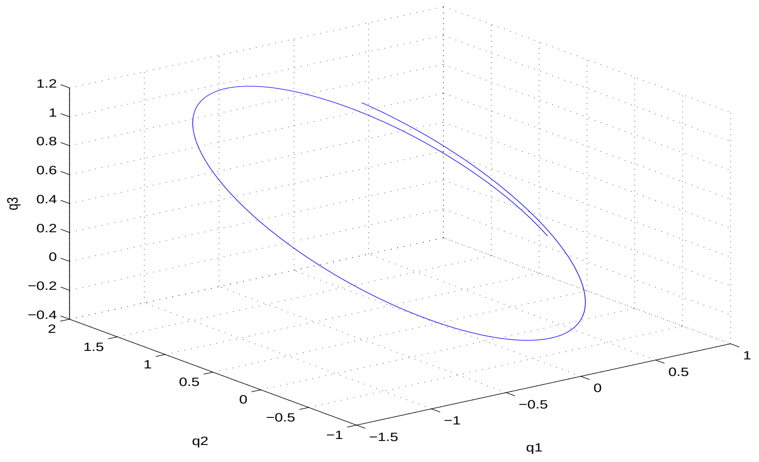

Figure 5.41. Trajectory of motion by 4th order Runge-Kutta.

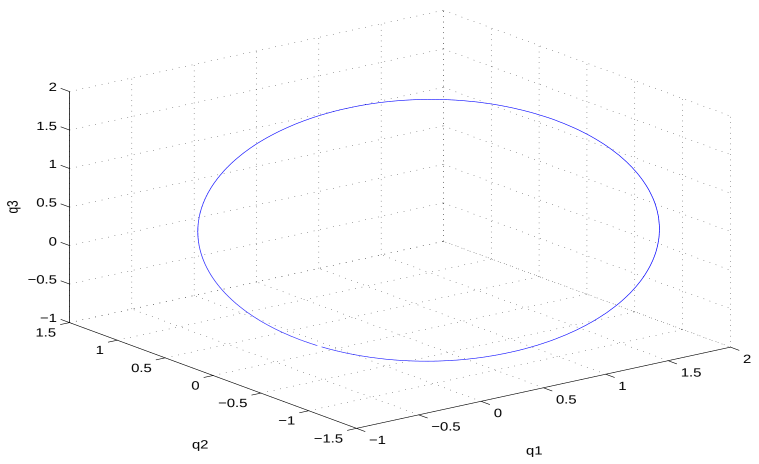

Figure 5.42. Trajectory of motion by Composition method of order 5 (midpoint rule as base method).

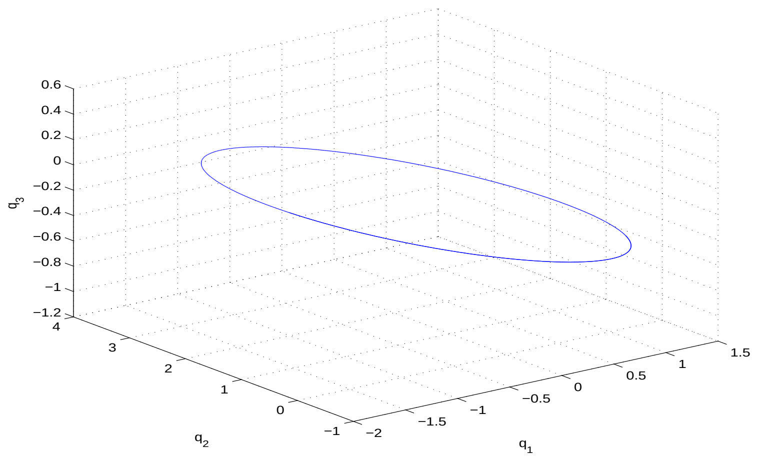

Figure 5.43. Trajectory of motion by Composition method of order 5 (Störmer-Verlet as base method).

5.4.2. Conservation of Energy

Conservation of the energy is presented in Figure(5.44, 5.45, 5.46) by classic 4th order Runge-Kutta method, composition method of order 5 (midpoint rule as base method) and composition method of order 5 (Störmer-Verlet as base method), respectively. We observed following: composition method of order 5 preserve the energy better than 4th order Runge-Kutta.

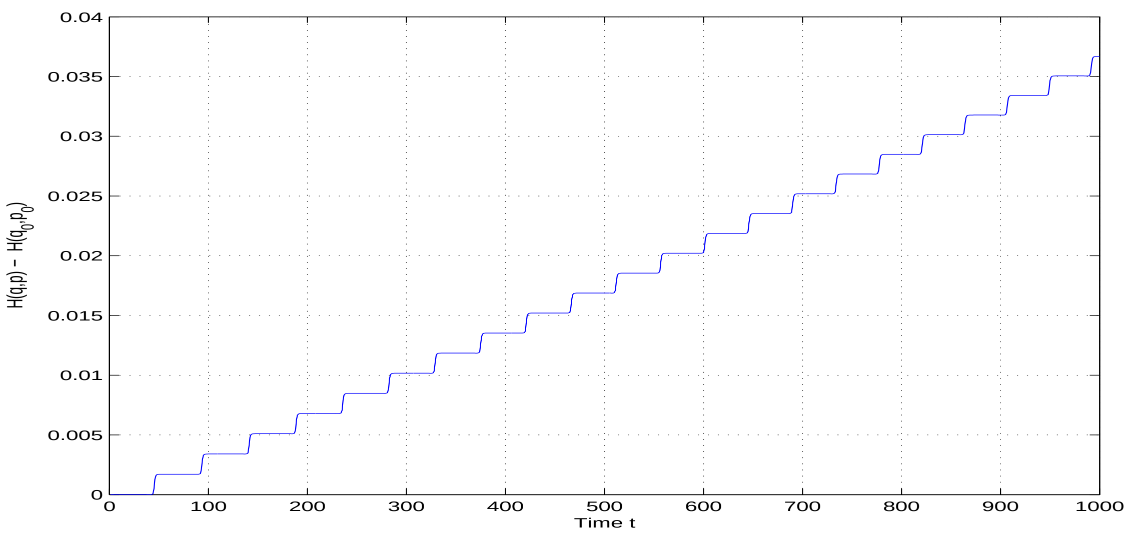

Figure 5.44. The errors in Hamiltonian $H(q,p) - H(q_0,p_0)$ for 4th order Runge-Kutta.

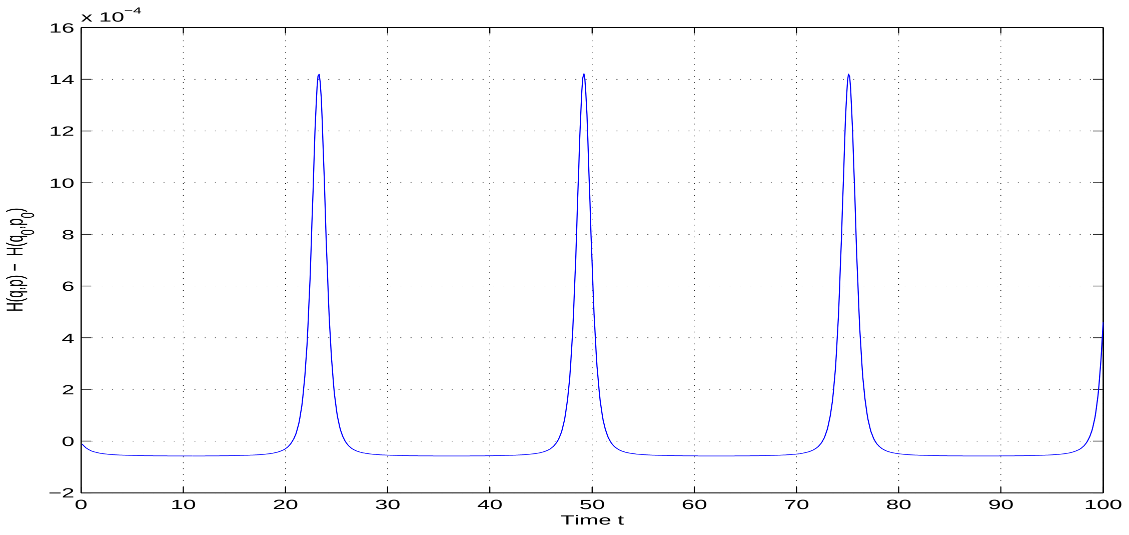

Figure 5.45. The errors in Hamiltonian $H(q,p)-H(q_0,p_0)$ for Composition method of order 5 (midpoint rule as base method).

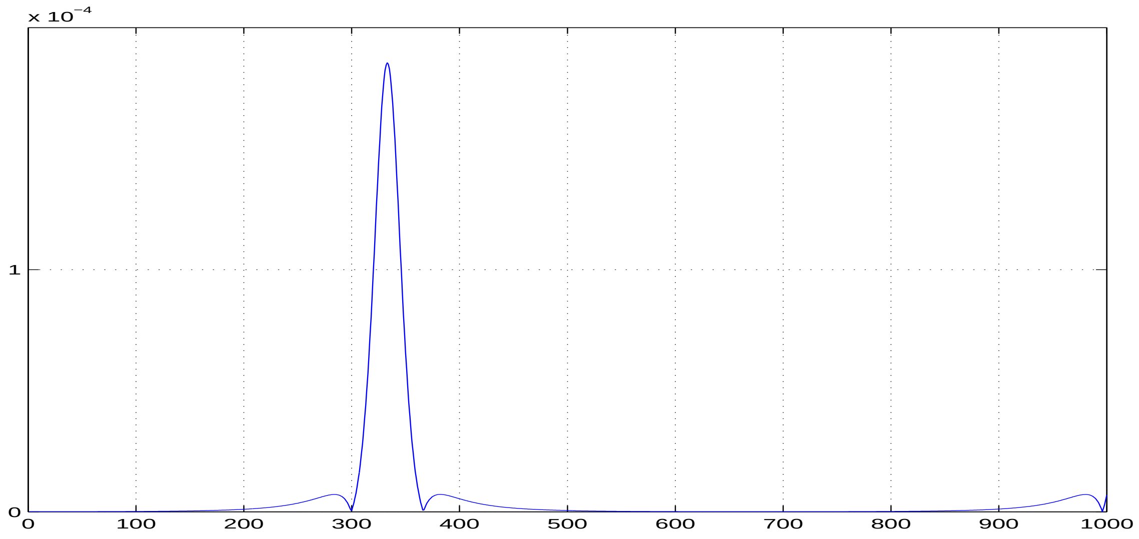

Figure 5.46. The errors in Hamiltonian $H(q,p)-H(q_0,p_0)$ for Composition method of order 5 (Störmer-Verlet as base method).

5.4.3. Conservation of Angular Momentum

In this subsection conservation of angular momentum is shown in Figure(5.47, 5.48, 5.49) by 4th order Runge-Kutta, composition method of order 5 (midpoint rule as base method) and composition method of order 5 (Störmer-Verlet as base method), respectively. Composition method of order 5 conserved the angular momentum up to the computer accuracy. Conservation of angular momentum by composition method of order 5 is better than 4th order Runge-Kutta method.

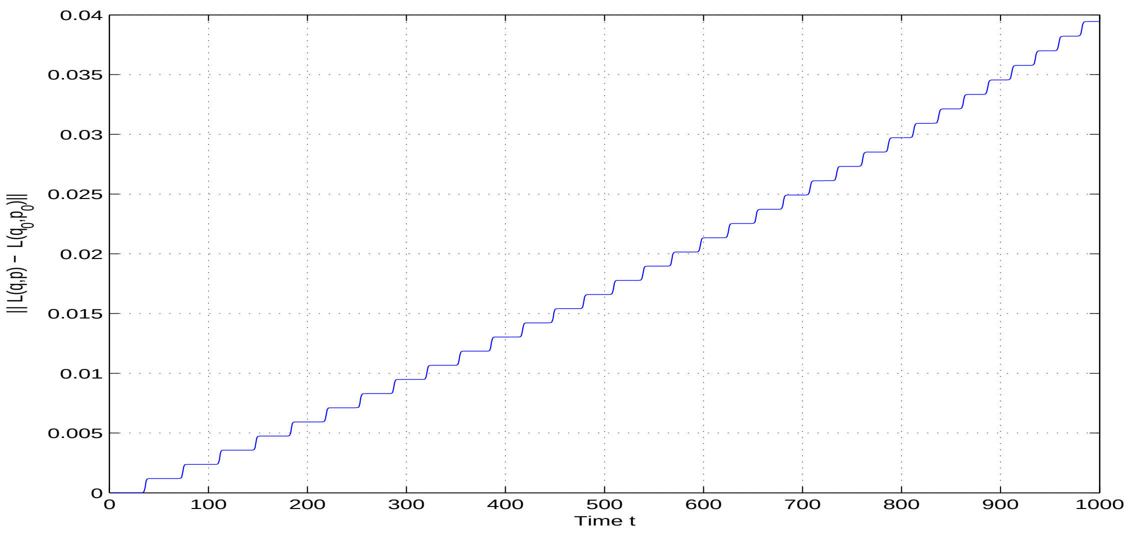

Figure 5.47. The errors in angular momentum $L(q,p) - L(q_0,p_0)$ for 4th order Runge-Kutta.

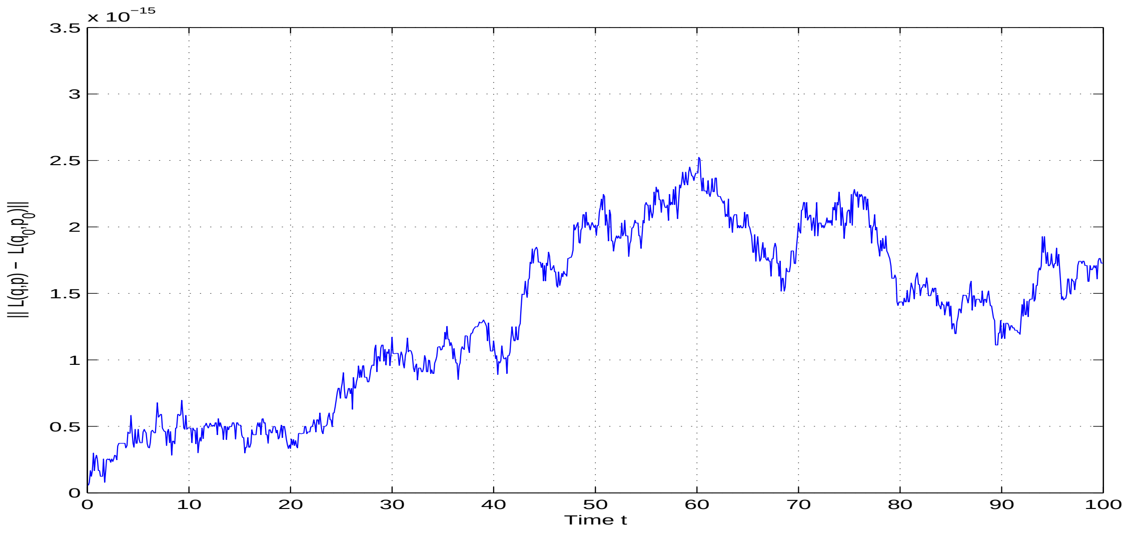

Figure 5.48. The errors in angular momentum $L(q, p) - L(q_0, p_0)$ Composition method of order 5 (midpoint rule as base method).

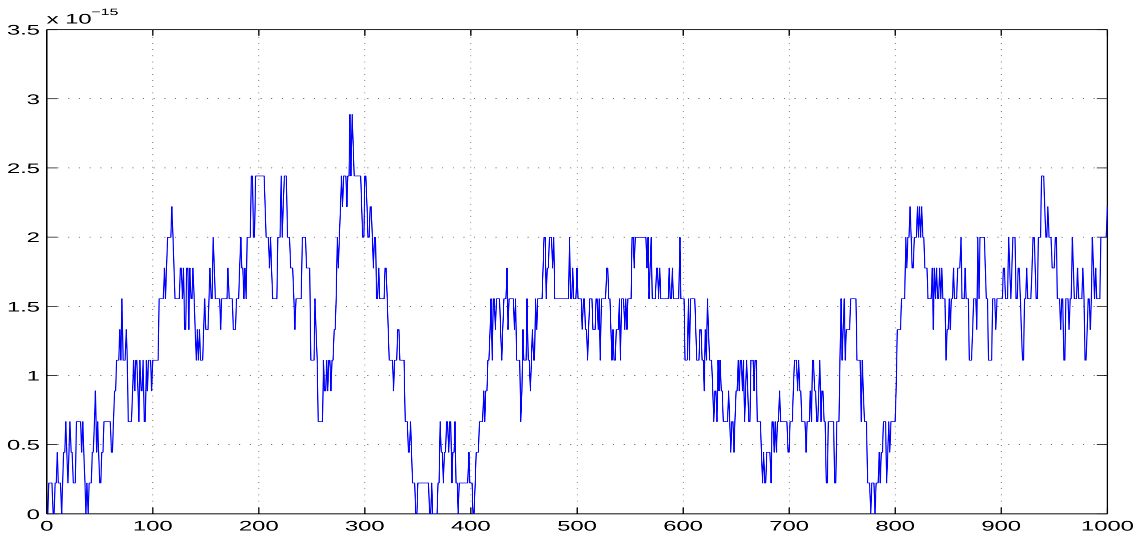

Figure 5.49. The errors in angular momentum $L(q, p) - L(q_0, p_0)$ Composition method of order 5 (Störmer-Verlet as base method).

5.4.4. Conservation of Runge-Lenz Vector

Conservation of Runge-Lenz vector is shown in Figure(5.50, 5.51, 5.52) by 4th order Runge-Kutta, composition method of order 5 (midpoint rule as base method) and composition method of order 5 (Störmer-Verlet as base method), respectively. We observe the following: composition method of order 5 preserve the Runge-Lenz vector better than 4th order Runge-Kutta method.

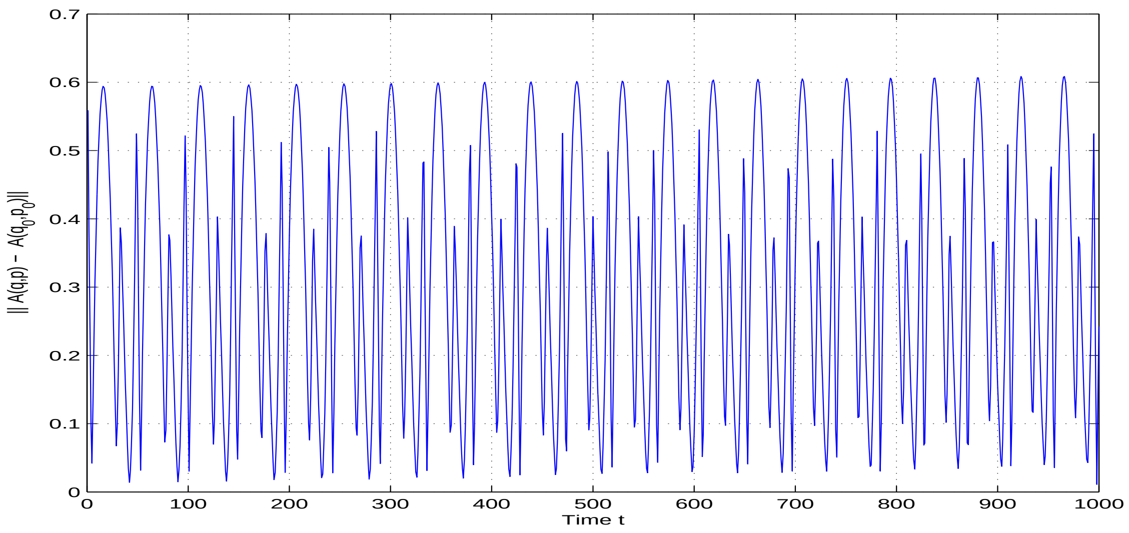

Figure 5.50. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for 4th order Runge-Kutta.

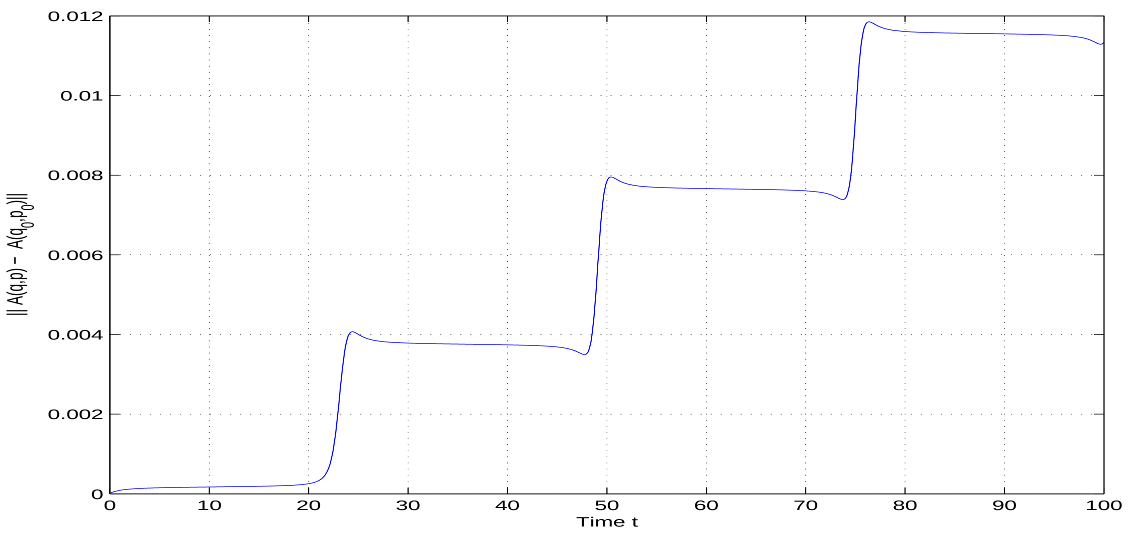

Figure 5.51. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for Composition method of order 5 (midpoint rule as base method).

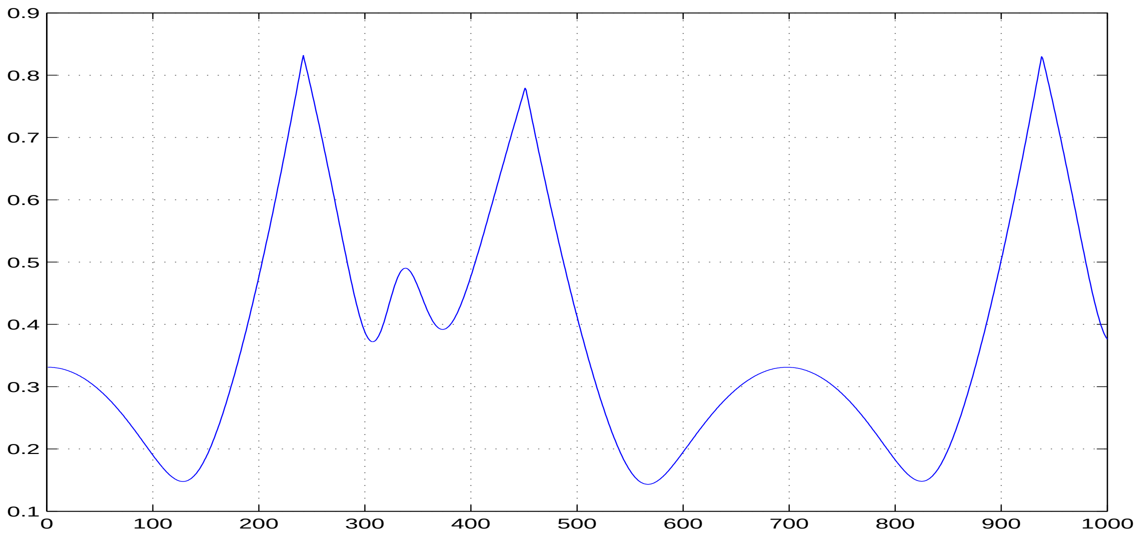

Figure 5.52. The errors in Runge-Lenz vector $A(q,p) - A(q_0,p_0)$ for Composition method of order 5 (Störmer-Verlet as base method).

CHAPTER 6

SUMMARY AND CONCLUSION

In this thesis both geometric and non-geometric integration methods were applied to the Kepler problem in order to determine which of its conserved quantities were preserved better. For this purpose explicit Euler, implicit Euler, symplectic Euler, adjoint of symplectic Euler, trapezoidal rule, midpoint rule, Störmer-Verlet method, 3rd and 4th order Runge-Kutta methods were chosen. In addition we constructed higher order symplectic method by composition techniques considering midpoint rule and Störmer-Verlet method as base methods. All algorithms of the methods were written in MATLAB.

The results obtained from computations are summarized in Table (1). In this table norm of errors in energy, angular momentum, Runge-Lenz vector and CPU times of the methods that we mentioned above are presented. From the table the following results are relived:

• Implicit methods are more expensive than explicit methods.

• Symplectic methods conserved three quantities of the Kepler problem better.

• When the order of the methods increase we get better conservation of the quantities.

• The best conservation of the quantities is obtained by composition method of order 5 (midpoint as the base method) however it's the most expensive method.

• In general the Runge-Lenz vector is not conserved by any method in a higher accuracy.

Table 1: Comparison of both Geometric and Non-Geometric Integration Methods with respect to norm of errors in energy, angular momentum, Runge-Lenz vector and CPU time.

		Norm of H	Norm of L	Norm of A	CPU time
First	EE	8.08	5.41	11.43	1.09
Order	IE	10.17	11.24	10.56	4.76
Methods	SE	1.19	4.05e-014	2.797	0.31
	ASE	1.07	3.35e-014	3.525	0.23
Second	TR	0.33	0.22	17.05	6.06
Order	IMR	0.07	2.76e-014	5.17	0.26
Methods	SV Method	0.05	3.16e-014	2.41	0.25
Third	RK	0.44	0.26	11.05	0.32
Order	CM (MR)	0.04	2.66e-014	2.58	17.48
Methods	CM (SV)	0.002	2.52e-014	7.33	12.80
Fourth and Fifth	RK	0.64	0.15	11.45	0.31
Order	CCM (MR)	0.008	2.57e-014	0.22	28.93
Methods	CMM (SV)	0.0008	2.45e-014	3.43	14.60

Abbreviations in the table:

EE	Explicit Euler Method
IE	Implicit Euler Method
SE	Symplectic Euler Method
ASE	Adjoint of Symplectic Euler Method
TR	Trapezoidal Rule
IMR	Implicit Midpoint Rule
SV	Störmer-Verlet Method
RK	Runge-Kutta Method
CM	Composition Method of Order 3
CCM	Composition Method of Order 5

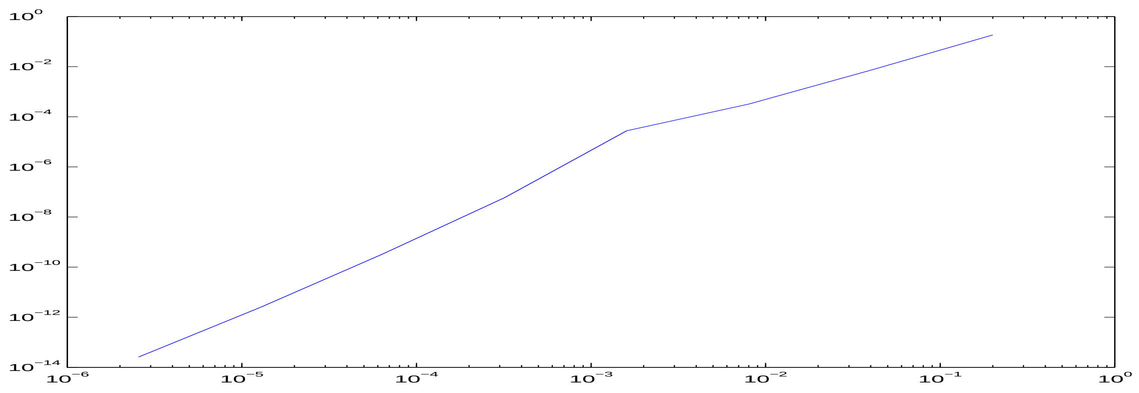

Figure 6.1. Errors for energy versus time step h in logarithmic scale for composition method of order 3 (midpoint rule as the base method).

Finally, errors for energy versus time step h in logarithmic scale for composition method of order 3 (midpoint rule as the base method) is presented in Figure (6.1).

As can be seen in this figure, when the step-size diminish the error in energy is also decreases linearly. This confirms the stability and consistency of this method.

REFERENCES

Budd C.J. and Iserles A., 1999. Geometric Integration: Numerical Solutions of Differential Equations. *Phil. Trans. Roy. Soc. A 357*: 943-1133.

Budd C.J. and Piggott M.D., 2000. Geometric Integration and Its Applications. *In handbook of Numerical Analysis*: 35-139.

Butcher J.C., 1987. *The Numerical Analysis of Ordinary Differential Equations, RungeKutta and General Linear Methods.* A Wiley- Interscience Publication. John Wiley and Sons Ltd.

Hairer E., 2005. Important Aspects of Geometric Numerical Integration. *Journal of Scientific Computing*: 67-81.

Hairer E., Lubich C.,Wanner G., eds. 2001. *Geometric Numerical Integration.* Berlin: Springer Series in Computational Mathematics.

Hairer E., Lubich C.,Wanner G., eds. 2006. *Geometric Numerical Integration. Structure-preserving algorithms for ordinary differential equations.* Springer-Verlag, Berlin : Second edition.

Kozlov R., 2007. Conservative Discritizations of the Kepler Problem. *Journal of Physics A: Mathematical and Theoretical*: 4529-4539.

Leimkuhler B. and Reich S., 2004. *Simulating Hamiltonian Dynamics.* USA : Cambridge University Press.

Rama Mohana Rao M., 1979. *Ordinary Differential Equations.Theory and Applications.* Indiana: Adword Arnold.

McLachlan R., 2004. *Geometric Integrators for ODE's.* Australia: Australian Research Council.

McLachlan R.,Quispel W., 2005. Geometric integrators for ODEs. *J. Phys. A: Math. Gen. 39*: 5251-5285.

McLachlan R. and Quispel W., 1999. Six lectures on the geometric integration of ODEs. *In Foundations of Computational Mathematics (Oxford, 1999)*: 155-210.

Sanz Serna J.M. 1994. *Numerical Hamiltonian Problems*. London: Applied mathematics and mathematical computation.

Stuart A.M., Humphries A. R., 1996. *Dynamical Systems and Numerical Analysis.* USA : Cambridge University Press.

Quispel G.R.W. and Dyt C., 1997. Solving ODE's Numerically while Preserving Symmetries, Hamiltonian Structure, Phase Space Volume or First Integrals. *Proc. 15th IMACS World Congress:* 601-607.

APPENDIX A

MATLAB CODES

A.1. First Order Methods

A.1.1. Explicit Euler

```
     N = 1000;
h = 100/N ;
K = 1;
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
r0 = sqrt( q0'*q0) ;

q1 = q0 ;
p1 = p0 ;
r1 = sqrt( q1'*q1) ;

%first energy
H0 =  (  ( p1'*p1/2 ) - ( K /r1 ) );
%for angular momentum
m1 = q1(2)*p1(3) - q1(3)*p1(2);
m2 = q1(3)*p1(1) - q1(1)*p1(3);
m3 = q1(1)*p1(2) - q1(2)*p1(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q1 *(H0 + p1'*p1/2 ) - p1 *(q1'*p1) ;

% computational unit
```

```
for i = 1:N

    q2 = q1+ h * p1 ;
    r = sqrt(q1'*q1) ;
    p2 = p1 - (h/2)*q1/(r^3) ;

    q1=q2;
    p1=p2;

    Q(:,i) = q1 ;
    P(:,i) = p1;
    T(i) = i*h ;

H(i)  =  (  ( p1'*p1/2 ) - ( K / sqrt(q1'*q1) ) ) ;
error_H(i)  =  H(i)   - H0  ;
%for angular momentum
mm1 = q1(2)*p1(3) - q1(3)*p1(2);
mm2 = q1(3)*p1(1) - q1(1)*p1(3);
mm3 = q1(1)*p1(2) - q1(2)*p1(1);

L(:,i) =  [mm1 , mm2 , mm3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;



RL(:, i) = q1 *(H(i) + p1'*p1/2 ) - p1 *(q1'*p1)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
plot3 (Q(1,:),Q(2,:),Q(3,:) ) ; grid on;
  xlabel('q_1');
  ylabel('q_2');
  zlabel('q_3');
```

```
plot (T,error_H) ; grid on;
 xlabel('Time t'); ylabel('H(q,p) -  H(q_0,p_0)');
plot (T,error_L) ; grid on;
 xlabel('Time t');  ylabel('|| L(q,p) -  L(q_0,p_0)||');
plot (T,error_RL) ;grid on;
  xlabel('Time t'); ylabel('|| A(q,p) -  A(q_0,p_0)||');
```

A.1.2. Implicit Euler

```
      N = 1000;
h = 100/N ;
K = 1;
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
r0 = sqrt( q0'*q0) ;

q = q0 ;
p = p0 ;
r = sqrt( q'*q) ;

%first energy
H0 =  (  ( p'*p/2 ) - ( K /r ) );
%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q *(H0 + p'*p/2 ) - p *(q'*p) ;
```

```
% computational unit

for i = 1:N

    r = sqrt(q'*q) ;
    p = p - h * q/(r^3) ;
    q = q + h * p;

    Q(:,i) = q ;
    P(:,i) = p;
    T(i) = i*h ;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(q'*q) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);

L(:,i) =  [m1 , m2 , m3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;

 RL(:, i) = q *(H(i) + p'*p/2 ) - p *(q'*p)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;
end;
```

A.1.3. Symplectic Euler

```
      N = 1000;
h = 100/N ;
K = 1;
```

```
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
r0 = sqrt( q0'*q0) ;

q = q0 ;
p = p0 ;
r = sqrt( q'*q) ;

%first energy
H0 =  (  ( p'*p/2 ) - ( K /r ) );
%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q *(H0 + p'*p/2 ) - p *(q'*p) ;

% computational unit

for i = 1:N
q = q + h * p;
    r = sqrt(q'*q) ;
    p = p - h * q/(r^3) ;

    Q(:,i) = q ;
    P(:,i) = p;
    T(i) = i*h ;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(q'*q) ) ) ;
```

```
error_H(i)  =  H(i)   - H0  ;

%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);

L(:,i) =  [m1 , m2 , m3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;

RL(:, i) = q *(H(i) + p'*p/2 ) - p *(q'*p)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
```

A.1.4. Adjoint of Symplectic Euler

```
      N = 1000;
h = 100/N ;
K = 1;
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
r0 = sqrt( q0'*q0) ;

q = q0 ;
p = p0 ;
r = sqrt( q'*q) ;

%first energy
H0 =  (  ( p'*p/2 ) - ( K /r ) );
%for angular momentum
```

```
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q *(H0 + p'*p/2 ) - p *(q'*p) ;

% computational unit

for i = 1:N

    r = sqrt(q'*q) ;
    p = p - h * q/(r^3) ;
    q = q + h * p;

    Q(:,i) = q ;
    P(:,i) = p;
    T(i) = i*h ;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(q'*q) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);

L(:,i) =  [m1 , m2 , m3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;
```

```
RL(:, i) = q *(H(i) + p'*p/2 ) - p *(q'*p)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
```

A.2. Second Order Methods

A.2.1. Trapezoidal Rule

```
      N = 1000 % 752;
h = 0.15 ;
K = 1;
x0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
t0 = 0 ;
r0 = sqrt( x0'*x0) ;

x = x0 ;
p = p0 ;
r = sqrt( x'*x) ;

H0 =  (  ( p'*p/2 ) - ( K /r ) )

m12 = x(1)*p(2) - x(2)*p(1);
m13 = x(1)*p(3) - x(3)*p(1);
m23 = x(2)*p(3) - x(3)*p(2);

M0  = [ m12 , m13 , m23 ]' ;

RL0 = x *(H0 + p'*p/2 ) - p *(x'*p) ;
% computational unit
```

```
for i = 1:N

x1 = x ;
p1 = p;

x2 = x1 ;
p2 = p1;

for k = 1:100

p_mid  = (p+p1)/2;
x2 =  x + h * p_mid ;

r1 = sqrt( x1'*x1) ;
r2= sqrt( x'*x) ;
p2 =  p - h/2*K*(x1/(r1^3) + x/(r2^3) ) ;

x1= x2;
p1 = p2;

end;

 X(:,i) = x1 ;
 P(:,i) = p1;
T(i) = i*h ;

x = x1;
p = p1;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(x'*x) ) ) ;
error_H(i)  =  H(i)   - H0  ;
m12 = x(1)*p(2) - x(2)*p(1);
```

```
m13 = x(1)*p(3) - x(3)*p(1);
m23 = x(2)*p(3) - x(3)*p(2);
M(:,i) =  [m12 , m13 , m23 ]'   ;
error_M(:,i) = sqrt( (M(:,i) - M0)'*(M(:,i)-M0)) ;
RL(:, i) = x *(H(i) + p'*p/2 ) - p *(x'*p)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;
end;
```

A.2.2. Midpoint Rule

```
      N = 1000;
h = 100/N ;
K = 1;
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;

r0 = sqrt( q0'*q0) ;
q = q0 ;
p = p0 ;
r = sqrt( q'*q) ;
%first energy
H0 =  (  ( p'*p/2 ) - ( K /r ) );
%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q *(H0 + p'*p/2 ) - p *(q'*p) ;

% computational unit
```

```
for i = 1:N

q1 = q ;
p1 = p;

q2 = q1 ;
p2 = p1;

for k = 1:100

p_mid  = (p+p1)/2;
q2 =  q + h * p_mid ;
q_mid  = (q+q1)/2;
r = sqrt( q_mid'*q_mid) ;
p2 =  p - h*K*q_mid/(r^3) ;

q1= q2;
p1 = p2;

end;

Q(:,i) = q1 ;
P(:,i) = p1;
T(i) = i*h ;

q = q1;
p = p1;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(q'*q) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%for angular momentum
```

```
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);

L(:,i) =  [m1 , m2 , m3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;



RL(:, i) = q *(H(i) + p'*p/2 ) - p *(q'*p)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;
end;
```

A.2.3. Störmer-Verlet Method

```
      N = 1000;
h = 100/N ;
K = 1;
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;
r0 = sqrt( q0'*q0) ;

q = q0 ;
p = p0 ;
r = sqrt( q'*q) ;

%first energy
H0 =  (  ( p'*p/2 ) - ( K /r ) );
%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);
```

```
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q *(H0 + p'*p/2 ) - p *(q'*p) ;

% computational unit

for i = 1:N

    r = sqrt(q'*q) ;
    x = p - (h/2)*q/(r^3) ;
    q = q + h * x ;
    r = sqrt(q'*q) ;
    p = x - (h/2)*q/(r^3) ;

    Q(:,i) = q ;
    P(:,i) = p;
    T(i) = i*h ;

H(i)  =  (  ( p'*p/2 ) - ( K / sqrt(q'*q) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%for angular momentum
m1 = q(2)*p(3) - q(3)*p(2);
m2 = q(3)*p(1) - q(1)*p(3);
m3 = q(1)*p(2) - q(2)*p(1);

L(:,i) =  [m1 , m2 , m3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;



RL(:, i) = q *(H(i) + p'*p/2 ) - p *(q'*p)  ;
```

```
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
```

A.3. Third Order Methods

A.3.1. Third Order Runge-Kutta Method

```
      Y=[0;1;0.5;0.8;0.6;0];
h=0.01;
n=1000;
for i=1:n
    p1(i)=Y(1,1);
    p2(i)=Y(2,1);
    p3(i)=Y(3,1);
    q1(i)=Y(4,1);
    q2(i)=Y(5,1);
    q3(i)=Y(6,1);
    w=(sqrt(q1(i)^2+q2(i)^2+q3(i)^2))^3;
    A=[0 0 0 -1/w 0 0;0 0 0 0 -1/w 0;0 0 0 0 0 -1/w;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k1=A*Y;
    Y1=Y+(h/2)*k1;
    w1=(sqrt(Y1(4,1)^2+Y1(5,1)^2+Y1(6,1)^2))^3;
    A1=[0 0 0 -1/w1 0 0;0 0 0 0 -1/w1 0;0 0 0 0 0 -1/w1;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k2=A1*Y1;
    Y2=Y+(h/2)*k2;
    w2=(sqrt(Y2(4,1)^2+Y2(5,1)^2+Y2(6,1)^2))^3;
    A2=[0 0 0 -1/w2 0 0;0 0 0 0 -1/w2 0;0 0 0 0 0 -1/w2;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k3=A2*Y2;
    Y=Y+(1/6)*h*(k1+4*k2+2*k3);
    L(i)=((q1(i)*p2(i)+q2(i)*p3(i)+q3(i)*p1(i))-
    (q2(i)*p1(i)+q3(i)*p2(i)+q1(i)*p3(i)));
    r(i)=abs(L(i)-L(1));
    H(i)=0.5*((p1(i)).^2+(p2(i)).^2+(p3(i)).^2)-1/
    (sqrt((q1(i)).^2+(q2(i)).^2+(q3(i)).^2));
    e(i)=abs(H(i)-H(1));
    H(1)=0.5*((p1(1)).^2+(p2(1)).^2+(p3(1)).^2)-1/
    (sqrt((q1(1)).^2+(q2(1)).^2+(q3(1)).^2));
a=p1(1)*q1(1)+p2(1)*q2(1)+p3(1)*q3(1);
b=H(1)+(1/2)*sqrt((p1(1).^2)+(p2(1).^2)+(p3(1).^2));
a1=(b*q1(1)-a*p1(1));
a2=(b*q2(1)-a*p2(1));
a3=(b*q3(1)-a*p3(1));
%Runge-Lenz vector
    c=p1(i)*q1(i)+p2(i)*q2(i)+p3(i)*q3(i);
    d=H(i)+(1/2)*sqrt((p1(i).^2)+(p1(i).^2)+(p1(i).^2));
    b1(i)=(d*q1(i)-c*p1(i));
    b2(i)=(d*q2(i)-c*p2(i));
    b3(i)=(d*q3(i)-c*p3(i));
    R(i)=sqrt(((b1(i)-a1).^2)+((b2(i)-a2).^2)+((b3(i)-a3).^2));
end
```

A.3.2. Composition Method of Order 3 (Midpoint as base method)

```
     N = 1000;
h = 100/N ;
K = 1;

gama1=1/2-(2^(1/3));
gama2=-((2^(1/3))/2-(2^(1/3)));
gama3=1/2-(2^(1/3));
```

```
%initial conditions
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;

r0 = sqrt( q0'*q0) ;
q1 = q0 ;
p1 = p0 ;
r1 = sqrt( q1'*q1) ;
%first energy
H0 =  (  ( p1'*p1/2 ) - ( K /r1 ) );
%for angular momentum
m1 = q1(2)*p1(3) - q1(3)*p1(2);
m2 = q1(3)*p1(1) - q1(1)*p1(3);
m3 = q1(1)*p1(2) - q1(2)*p1(1);
%first angular momentum
L0  = [ m1 , m2 , m3 ]' ;
%first Runge-Lenz vector
RL0 = q1 *(H0 + p1'*p1/2 ) - p1 *(q1'*p1) ;

% ----------------computational unit 1---------------

for i = 1:N

q11 = q1 ;
p11 = p1;

q21 = q11 ;
p21 = p11;

for k = 1:100

p_mid1  = (p1+p11)/2;
```

```
q21 =  q1 + h *gama1* p_mid1 ;
q_mid1  = (q1+q11)/2;
r1 = sqrt( q_mid1'*q_mid1) ;
p21 =  p1 - h*K*gama1*q_mid1/(r1^3) ;

q11= q21;
p11 = p21;

end;

x1 = q11;
y1 = p11;

end;

%----------------computational unit 2------------------

q2 = x1 ;
p2 = y1 ;
r2 = sqrt( q2'*q2) ;


for i = 1:N

q12 = q2 ;
p12 = p2;

q22 = q12 ;
p22 = p12;

for k = 1:100
```

```
p_mid2  = (p2+p12)/2;
q22 =  q2 + h *gama2* p_mid2 ;
q_mid2  = (q2+q12)/2;
r2 = sqrt( q_mid2'*q_mid2) ;
p22 =  p2 - h*K*gama2*q_mid2/(r2^3) ;

q12= q22;
p12 = p22;
end;

x2 = q12;
y2 = p12;

end;

%----------------computational unit 3------------------

q3 = x2 ;
p3 = y2 ;
r3 = sqrt( q3'*q3) ;


for i = 1:N

q13 = q3 ;
p13 = p3;

q23 = q13 ;
p23 = p13;

for k = 1:100
```

```
p_mid3  = (p3+p13)/2;
q23 =  q3 + h *gama3* p_mid3 ;
q_mid3  = (q3+q13)/2;
r3 = sqrt( q_mid3'*q_mid3) ;
p23 =  p3 - h*K*gama3*q_mid3/(r3^3) ;

q13= q23;
p13 = p23;

end;

Q(:,i) = q13 ;
P(:,i) = p13;
T(i) = i*h ;

q3 = q13;
p3 = p13;

% energy

H(i)  =  (  ( p3'*p3/2 ) - ( K / sqrt(q3'*q3) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%angular momentum
mm1 = q3(2)*p3(3) - q3(3)*p3(2);
mm2 = q3(3)*p3(1) - q3(1)*p3(3);
mm3 = q3(1)*p3(2) - q3(2)*p3(1);

L(:,i) =  [mm1 , mm2 , mm3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;

%runge-lenz vector
```

```
RL(:, i) = q3 *(H(i) + p3'*p3/2 ) - p3 *(q3'*p3)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
```

A.3.3. Composition Method of Order 3 (Störmer-Verlet as base method)

```
     q11(1)=0.8;
q21(1)=0.6;
p11(1)=0;
p21(1)=1;
q31(1)=0;
p31(1)=0.5;

h=0.01;
gama1=1/2-(2^(1/3));
gama2=-((2^(1/3))/2-(2^(1/3)));
gama3=1/2-(2^(1/3));h=0.007;

for i=1:1000;

    k=p11(i)-(h/2)*gama1*(q11(i)/(q11(i).^2+
    q21(i).^2+q31(i).^2).^(1.5));
    m=p21(i)-(h/2)*gama1*(q21(i)/(q11(i).^2+
    q21(i).^2+q31(i).^2).^(1.5));
    t=p31(i)-(h/2)*gama1*(q31(i)/(q11(i).^2+
    q21(i).^2+q31(i).^2).^(1.5));
    q11(i+1)=q11(i)+h*gama1*k;
    q21(i+1)=q21(i)+h*gama1*m;
    q31(i+1)=q31(i)+h*gama1*t;
    p11(i+1)=k-(h/2)*gama1*(q11(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
    p21(i+1)=m-(h/2)*gama1*(q21(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
    p31(i+1)=t-(h/2)*gama1*(q31(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
end

p12(1)=p11(1001);
p22(1)=p21(1001);
q12(1)=q11(1001);
q22(1)=q21(1001);
p32(1)=p31(1001);
q32(1)=q31(1001);

for i=1:1000;
    k=p12(i)-(h/2)*gama2*(q12(i)/(q11(i).^2+
    q22(i).^2+q32(i).^2).^(1.5));
    m=p22(i)-(h/2)*gama2*(q22(i)/(q12(i).^2+
    q22(i).^2+q32(i).^2).^(1.5));
    t=p32(i)-(h/2)*gama2*(q32(i)/(q12(i).^2+
    q22(i).^2+q32(i).^2).^(1.5));
    q12(i+1)=q12(i)+h*gama2*k;
    q22(i+1)=q22(i)+h*gama2*m;
    q32(i+1)=q32(i)+h*gama2*t;
    p12(i+1)=k-(h/2)*gama2*(q12(i+1)/(q12(i+1).^2+q22(i+1).^2+
    q32(i+1).^2).^(1.5));
    p22(i+1)=m-(h/2)*gama2*(q12(i+1)/(q12(i+1).^2+
    q22(i+1).^2+q32(i+1).^2).^(1.5));
    p32(i+1)=t-(h/2)*gama2*(q32(i+1)/(q12(i+1).^2+
    q22(i+1).^2+q32(i+1).^2).^(1.5));
end
```

```
p1(1)=p12(1001);
p2(1)=p22(1001);
q1(1)=q12(1001);
q2(1)=q22(1001);
p3(1)=p32(1001);
q3(1)=q32(1001);

H(1)=0.5*((p1(1)).^2+(p2(1)).^2+(p3(1)).^2)-1/
(sqrt((q1(1)).^2+(q2(1)).^2+(q3(1)).^2));
a=p1(1)*q1(1)+p2(1)*q2(1)+p3(1)*q3(1);
b=H(1)+(1/2)*sqrt((p1(1).^2)+(p2(1).^2)+(p3(1).^2));
a1=(b*q1(1)-a*p1(1));
a2=(b*q2(1)-a*p2(1));
a3=(b*q3(1)-a*p3(1));

for i=1:1000;
    % Energy
    H(i)=0.5*((p1(i)).^2+(p2(i)).^2+(p3(i)).^2)-1/
(sqrt((q1(i)).^2+(q2(i)).^2+(q3(i)).^2));
    e(i)=abs(H(i)-H(1));
    %angular momentum
    L(i)=((q1(i)*p2(i)+q2(i)*p3(i)+q3(i)*p1(i))-
    (q2(i)*p1(i)+q3(i)*p2(i)+q1(i)*p3(i)));
    r(i)=abs(L(i)-L(1));
    %Runge-Lenz vector
    c=p1(i)*q1(i)+p2(i)*q2(i)+p3(i)*q3(i);
    d=H(i)+(1/2)*sqrt((p1(i).^2)+(p1(i).^2)+(p1(i).^2));
    b1(i)=(d*q1(i)-c*p1(i));
    b2(i)=(d*q2(i)-c*p2(i));
    b3(i)=(d*q3(i)-c*p3(i));
    R(i)=sqrt(((b1(i)-a1).^2)+((b2(i)-a2).^2)+((b3(i)-a3).^2));
    k=p1(i)-(h/2)*gama3*(q1(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    m=p2(i)-(h/2)*gama3*(q2(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    t=p3(i)-(h/2)*gama3*(q3(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    q1(i+1)=q1(i)+h*gama3*k;
    q2(i+1)=q2(i)+h*gama3*m;
    q3(i+1)=q3(i)+h*gama3*t;
    p1(i+1)=k-(h/2)*gama3*(q1(i+1)/(q1(i+1).^2+
    q2(i+1).^2+q3(i+1).^2).^(1.5));
    p2(i+1)=m-(h/2)*gama3*(q2(i+1)/(q1(i+1).^2+
    q2(i+1).^2+q3(i+1).^2).^(1.5));
    p3(i+1)=t-(h/2)*gama3*(q3(i+1)/(q1(i+1).^2+
    q2(i+1).^2+q3(i+1).^2).^(1.5));
    end
```

A.4. Fourth and Fifth Order Methods

A.4.1. Fourth Order Runge-Kutta Method

```
      Y=[0;1;0.5;0.8;0.6;0];
h=0.2;
n=1000;
for i=1:n
    p1(i)=Y(1,1);
    p2(i)=Y(2,1);
    p3(i)=Y(3,1);
    q1(i)=Y(4,1);
    q2(i)=Y(5,1);
    q3(i)=Y(6,1);
    w=(sqrt(q1(i)^2+q2(i)^2+q3(i)^2))^3;
    A=[0 0 0 -1/w 0 0;0 0 0 0 -1/w 0;0 0 0 0 0 -1/w;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k1=A*Y;
    Y1=Y+(h/2)*k1;
    w1=(sqrt(Y1(4,1)^2+Y1(5,1)^2+Y1(6,1)^2))^3;
    A1=[0 0 0 -1/w1 0 0;0 0 0 0 -1/w1 0;0 0 0 0 0 -1/w1;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k2=A1*Y1;
    Y2=Y+(h/2)*k2;
    w2=(sqrt(Y2(4,1)^2+Y2(5,1)^2+Y2(6,1)^2))^3;
    A2=[0 0 0 -1/w2 0 0;0 0 0 0 -1/w2 0;0 0 0 0 0 -1/w2;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k3=A2*Y2;
    Y3=Y+h*k3;
    w3=(sqrt(Y3(4,1)^2+Y3(5,1)^2+Y3(6,1)^2))^3;
    A3=[0 0 0 -1/w3 0 0;0 0 0 0 -1/w3 0;0 0 0 0 0 -1/w3;
    1 0 0 0 0 0;0 1 0 0 0 0;0 0 1 0 0 0];
    k4=A3*Y3;
    Y=Y+(1/6)*h*(k1+2*k2+2*k3+k4);
    L(i)=((q1(i)*p2(i)+q2(i)*p3(i)+q3(i)*p1(i))-
    (q2(i)*p1(i)+q3(i)*p2(i)+q1(i)*p3(i)));
    r(i)=abs(L(i)-L(1));
    H(i)=0.5*((p1(i)).^2+(p2(i)).^2+(p3(i)).^2)-1/
    (sqrt((q1(i)).^2+(q2(i)).^2+(q3(i)).^2));
    e(i)=abs(H(i)-H(1));
    H(1)=0.5*((p1(1)).^2+(p2(1)).^2+(p3(1)).^2)-1/
    (sqrt((q1(1)).^2+(q2(1)).^2+(q3(1)).^2));
a=p1(1)*q1(1)+p2(1)*q2(1)+p3(1)*q3(1);
b=H(1)+(1/2)*sqrt((p1(1).^2)+(p2(1).^2)+(p3(1).^2));
a1=(b*q1(1)-a*p1(1));
a2=(b*q2(1)-a*p2(1));
a3=(b*q3(1)-a*p3(1));
%Runge-Lenz vector
    c=p1(i)*q1(i)+p2(i)*q2(i)+p3(i)*q3(i);
    d=H(i)+(1/2)*sqrt((p1(i).^2)+(p1(i).^2)+(p1(i).^2));
    b1(i)=(d*q1(i)-c*p1(i));
    b2(i)=(d*q2(i)-c*p2(i));
    b3(i)=(d*q3(i)-c*p3(i));
    R(i)=sqrt(((b1(i)-a1).^2)+((b2(i)-a2).^2)+((b3(i)-a3).^2));
end
```

A.4.2. Composition Method of Order 5 (Midpoint as base method)

```
     N = 1000;
h = 100/N ;
K = 1;

gama1=1/(4-(4^(1/5)));
gama2=1/(4-(4^(1/5)));
gama3=-((4^(1/5))/(4-(4^(1/5))));
gama4=1/(4-(4^(1/5)));
gama5=1/(4-(4^(1/5)));



%ilk deerler
q0  =  [ 0.8 , 0.6 , 0 ]' ;
p0  =  [ 0 , 1 , 0.5 ]' ;

r0 = sqrt( q0'*q0) ;
q1 = q0 ;
p1 = p0 ;
r1 = sqrt( q1'*q1) ;
%energy nin ilk deeri
H0 =  (  ( p1'*p1/2 ) - ( K /r1 ) );
%angular momentum iin
m1 = q1(2)*p1(3) - q1(3)*p1(2);
m2 = q1(3)*p1(1) - q1(1)*p1(3);
m3 = q1(1)*p1(2) - q1(2)*p1(1);
%angular momentum un ilk deeri
```

```
L0  = [ m1 , m2 , m3 ]' ;
%Runge-Lenz vector un ilk deeri
RL0 = q1 *(H0 + p1'*p1/2 ) - p1 *(q1'*p1) ;

% ----------------computational unit 1---------------

for i = 1:N

q11 = q1 ;
p11 = p1;

q21 = q11 ;
p21 = p11;

for k = 1:100

p_mid1  = (p1+p11)/2;
q21 =  q1 + h *gama1* p_mid1 ;
q_mid1  = (q1+q11)/2;
r1 = sqrt( q_mid1'*q_mid1) ;
p21 =  p1 - h*K*gama1*q_mid1/(r1^3) ;

q11= q21;
p11 = p21;

end;

x1 = q11;
y1 = p11;

end;
```

```
%----------------computational unit 2------------------

q2 = x1 ;
p2 = y1 ;
r2 = sqrt( q2'*q2) ;



for i = 1:N

q12 = q2 ;
p12 = p2;

q22 = q12 ;
p22 = p12;

for k = 1:100

p_mid2  = (p2+p12)/2;
q22 =  q2 + h *gama2* p_mid2 ;
q_mid2  = (q2+q12)/2;
r2 = sqrt( q_mid2'*q_mid2) ;
p22 =  p2 - h*K*gama2*q_mid2/(r2^3) ;

q12= q22;
p12 = p22;

end;

x2 = q12;
y2 = p12;

end;
```

```
%----------------computational unit 3------------------

q3 = x2 ;
p3 = y2 ;
r3 = sqrt( q3'*q3) ;


for i = 1:N

q13 = q3 ;
p13 = p3;

q23 = q13 ;
p23 = p13;

for k = 1:100

p_mid3  = (p3+p13)/2;
q23 =  q3 + h *gama3* p_mid3 ;
q_mid3  = (q3+q13)/2;
r3 = sqrt( q_mid3'*q_mid3) ;
p23 =  p3 - h*K*gama3*q_mid3/(r3^3) ;

q13= q23;
p13 = p23;

end;

x3 = q13 ;
y3 = p13;

end;
```

```
%----------------computational unit 4------------------



q4 = x3 ;

p4 = y3 ;

r4 = sqrt( q4'*q4) ;



for i = 1:N


q14 = q4 ;

p14 = p4;


q24 = q14 ;

p24 = p14;


for k = 1:100


p_mid4  = (p4+p14)/2;

q24 =  q4 + h *gama4* p_mid4 ;

q_mid4  = (q4+q14)/2;

r4 = sqrt( q_mid4'*q_mid4) ;

p24 =  p4 - h*K*gama4*q_mid4/(r4^3) ;


q14= q24;

p14 = p24;


end;


x4 = q14 ;

y4 = p14;
```

```
end;

%----------------computational unit 5------------------
q5 = x4 ;
p5 = y4 ;
r5 = sqrt( q5'*q5) ;


for i = 1:N

q15 = q5 ;
p15 = p5;

q25 = q15 ;
p25 = p15;

for k = 1:100

p_mid5  = (p5+p15)/2;
q25 =  q5 + h *gama5* p_mid5 ;
q_mid5  = (q5+q15)/2;
r5 = sqrt( q_mid5'*q_mid5) ;
p25 =  p5 - h*K*gama5*q_mid5/(r5^3) ;

q15= q25;
p15 = p25;

end;

Q(:,i) = q15 ;
P(:,i) = p15;
T(i) = i*h ;
```

```
q5 = q15;
p5 = p15;

% energy

H(i)  =  (  ( p5'*p5/2 ) - ( K / sqrt(q5'*q5) ) ) ;
error_H(i)  =  H(i)   - H0  ;

%angular momentum
mm1 = q5(2)*p5(3) - q5(3)*p5(2);
mm2 = q5(3)*p5(1) - q5(1)*p5(3);
mm3 = q5(1)*p5(2) - q5(2)*p5(1);
L(:,i) =  [mm1 , mm2 , mm3 ]'   ;
error_L(:,i) = sqrt( (L(:,i) - L0)'*(L(:,i)-L0)) ;

%runge-lenz vector

RL(:, i) = q5 *(H(i) + p5'*p5/2 ) - p5 *(q5'*p5)  ;
error_RL(:,i) = sqrt( (RL(:,i)- RL0)'*(RL(:,i)- RL0)) ;

end;
```

A.4.3. Composition Method of Order 5 (Störmer-Verlet as base method)

```
      q11(1)=0.8;
q21(1)=0.6;
p11(1)=0;
p21(1)=1;
q31(1)=0;
p31(1)=0.5;
```

```
%h=0.01;

gama1=1/(4-(4^(1/5)));
gama2=1/(4-(4^(1/5)));
gama4=1/(4-(4^(1/5)));
gama5=1/(4-(4^(1/5)));h=0.08;
gama3=-((4^(1/5))/(4-(4^(1/5))));
% for s=1:8;
%     h=(1/5)^s;
%     x(s)=h;
for i=1:1000;

    k=p11(i)-(h/2)*gama1*(q11(i)/(q11(i).^2+q21(i).^2+
    q31(i).^2).^(1.5));
    m=p21(i)-(h/2)*gama1*(q21(i)/(q11(i).^2+
    q21(i).^2+q31(i).^2).^(1.5));
    t=p31(i)-(h/2)*gama1*(q31(i)/(q11(i).^2+
    q21(i).^2+q31(i).^2).^(1.5));
    q11(i+1)=q11(i)+h*gama1*k;
    q21(i+1)=q21(i)+h*gama1*m;
    q31(i+1)=q31(i)+h*gama1*t;
    p11(i+1)=k-(h/2)*gama1*(q11(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
    p21(i+1)=m-(h/2)*gama1*(q21(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
    p31(i+1)=t-(h/2)*gama1*(q31(i+1)/(q11(i+1).^2+
    q21(i+1).^2+q31(i+1).^2).^(1.5));
end

p12(1)=p11(1001);
p22(1)=p21(1001);
q12(1)=q11(1001);
```

```
q22(1)=q21(1001);
p32(1)=p31(1001);
q32(1)=q31(1001);

for i=1:1000;
    k=p12(i)-(h/2)*gama2*(q12(i)/(q11(i).^2+q22(i).^2+
    q32(i).^2).^(1.5));
    m=p22(i)-(h/2)*gama2*(q22(i)/(q12(i).^2+q22(i).^2+
    q32(i).^2).^(1.5));
    t=p32(i)-(h/2)*gama2*(q32(i)/(q12(i).^2+q22(i).^2+
    q32(i).^2).^(1.5));
    q12(i+1)=q12(i)+h*gama2*k;
    q22(i+1)=q22(i)+h*gama2*m;
    q32(i+1)=q32(i)+h*gama2*t;
    p12(i+1)=k-(h/2)*gama2*(q12(i+1)/(q12(i+1).^2+
    q22(i+1).^2+q32(i+1).^2).^(1.5));
    p22(i+1)=m-(h/2)*gama2*(q12(i+1)/(q12(i+1).^2+
    q22(i+1).^2+q32(i+1).^2).^(1.5));
    p32(i+1)=t-(h/2)*gama2*(q32(i+1)/(q12(i+1).^2+
    q22(i+1).^2+q32(i+1).^2).^(1.5));
end
p13(1)=p12(1001);
p23(1)=p22(1001);
q13(1)=q12(1001);
q23(1)=q22(1001);
p33(1)=p32(1001);
q33(1)=q32(1001);

for i=1:1000;
    k=p13(i)-(h/2)*gama3*(q13(i)/(q13(i).^2+q23(i).^2+
    q33(i).^2).^(1.5));
    m=p23(i)-(h/2)*gama3*(q23(i)/(q13(i).^2+q23(i).^2+
    q33(i).^2).^(1.5));
    t=p33(i)-(h/2)*gama3*(q33(i)/(q13(i).^2+q23(i).^2+
    q33(i).^2).^(1.5));
    q13(i+1)=q13(i)+h*gama3*k;
    q23(i+1)=q23(i)+h*gama3*m;
    q33(i+1)=q33(i)+h*gama3*t;
    p13(i+1)=k-(h/2)*gama3*(q13(i+1)/(q13(i+1).^2+
    q23(i+1).^2+q33(i+1).^2).^(1.5));
    p23(i+1)=m-(h/2)*gama3*(q13(i+1)/(q13(i+1).^2+
    q23(i+1).^2+q33(i+1).^2).^(1.5));
    p33(i+1)=t-(h/2)*gama3*(q33(i+1)/(q13(i+1).^2+
    q23(i+1).^2+q33(i+1).^2).^(1.5));
end
p14(1)=p13(1001);
p24(1)=p23(1001);
q14(1)=q13(1001);
q24(1)=q23(1001);
p34(1)=p33(1001);
q34(1)=q33(1001);

for i=1:1000;
    k=p14(i)-(h/2)*gama4*(q14(i)/(q14(i).^2+q24(i).^2+
    q34(i).^2).^(1.5));
    m=p24(i)-(h/2)*gama4*(q24(i)/(q14(i).^2+q24(i).^2+
    q34(i).^2).^(1.5));
    t=p34(i)-(h/2)*gama4*(q34(i)/(q14(i).^2+q24(i).^2+
    q34(i).^2).^(1.5));
    q14(i+1)=q14(i)+h*gama4*k;
    q24(i+1)=q24(i)+h*gama4*m;
    q34(i+1)=q34(i)+h*gama4*t;
    p14(i+1)=k-(h/2)*gama4*(q14(i+1)/(q14(i+1).^2+
    q24(i+1).^2+q34(i+1).^2).^(1.5));
    p24(i+1)=m-(h/2)*gama4*(q14(i+1)/(q14(i+1).^2+
    q24(i+1).^2+q34(i+1).^2).^(1.5));
    p34(i+1)=t-(h/2)*gama4*(q34(i+1)/(q14(i+1).^2+
    q24(i+1).^2+q34(i+1).^2).^(1.5));
end

p1(1)=p14(1001);
p2(1)=p24(1001);
q1(1)=q14(1001);
q2(1)=q24(1001);
p3(1)=p34(1001);
q3(1)=q34(1001);

H(1)=0.5*((p1(1)).^2+(p2(1)).^2+(p3(1)).^2)-1/
(sqrt((q1(1)).^2+(q2(1)).^2+(q3(1)).^2));
a=p1(1)*q1(1)+p2(1)*q2(1)+p3(1)*q3(1);
b=H(1)+(1/2)*sqrt((p1(1).^2)+(p2(1).^2)+(p3(1).^2));
a1=(b*q1(1)-a*p1(1));
a2=(b*q2(1)-a*p2(1));
a3=(b*q3(1)-a*p3(1));

for i=1:1000;
    % Energy
    H(i)=0.5*((p1(i)).^2+(p2(i)).^2+(p3(i)).^2)-1/
    (sqrt((q1(i)).^2+(q2(i)).^2+(q3(i)).^2));
    e(i)=abs(H(i)-H(1));
    %angular momentum
    L(i)=((q1(i)*p2(i)+q2(i)*p3(i)+q3(i)*p1(i))-
    (q2(i)*p1(i)+q3(i)*p2(i)+q1(i)*p3(i)));
    r(i)=abs(L(i)-L(1));
    %Runge-Lenz vector
    c=p1(i)*q1(i)+p2(i)*q2(i)+p3(i)*q3(i);
    d=H(i)+(1/2)*sqrt((p1(i).^2)+(p1(i).^2)+(p1(i).^2));
    b1(i)=(d*q1(i)-c*p1(i));
    b2(i)=(d*q2(i)-c*p2(i));
    b3(i)=(d*q3(i)-c*p3(i));
    R(i)=sqrt(((b1(i)-a1).^2)+((b2(i)-a2).^2)+((b3(i)-a3).^2));
    k=p1(i)-(h/2)*gama5*(q1(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    m=p2(i)-(h/2)*gama5*(q2(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    t=p3(i)-(h/2)*gama5*(q3(i)/(q1(i).^2+q2(i).^2+q3(i).^2).^(1.5));
    q1(i+1)=q1(i)+h*gama5*k;
    q2(i+1)=q2(i)+h*gama5*m;
    q3(i+1)=q3(i)+h*gama5*t;
    p1(i+1)=k-(h/2)*gama5*(q1(i+1)/(q1(i+1).^2+q2(i+1).^2+
    q3(i+1).^2).^(1.5));
    p2(i+1)=m-(h/2)*gama5*(q2(i+1)/(q1(i+1).^2+q2(i+1).^2+
    q3(i+1).^2).^(1.5));
    p3(i+1)=t-(h/2)*gama5*(q3(i+1)/(q1(i+1).^2+q2(i+1).^2+
    q3(i+1).^2).^(1.5));
end
```