**Implicit ODE (Ordinary Differential Equation) Solvers**

1. **Problem Statement and Useful Formula**
   1. **Problem definition**
2.  (1)
3. Or
4.  (2)
   1. **Richardson’s extrapolation method for Numerical Differetation**

* **Taylor series expansion of f(x+kh) for a small value h around a given point x (k is an integer)**



When , where  is a constant and is a step size.

 (3)

With  and , we can get

 (4)

* **Higher order Finite Difference formula**

Therefore, we can get two different expressions for the first derivative  using Eq. (3) and (4) as

 (5)

 (6)

For the convenience of derivations, let’s define the following function of .

 (7)

The above finite difference (FD) formula have the 2nd order accuracy. By removing the 2nd order term, we can get the 4th order FD formula through the operation of  as

 (8)

Or

 (9)

By substituting  with two different values of  and  into Eq. (8), we can have an opportunity to get a 6th order approximation of the first derivatives by removing the 4th order terms in the following equations.

 (10)

The Richardson’s recursive extrapolation method is originally derived by applying . However, the FD formula with  can be more straightforwardly derived when a uniform node is used. Followings are examples of such derivations.

2nd order : 

4th order: 

6th order



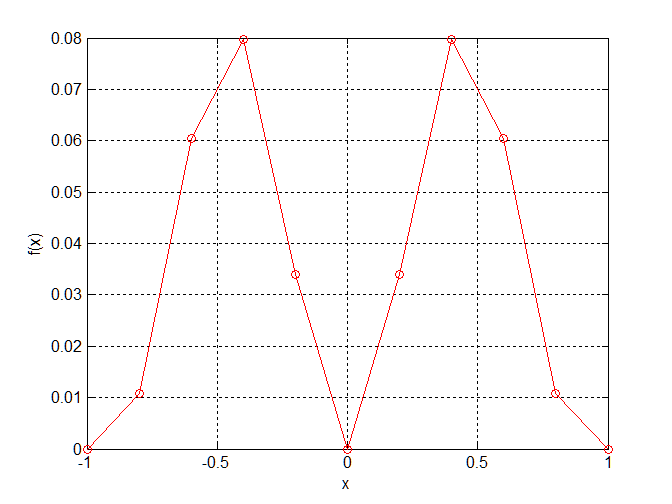
🡪

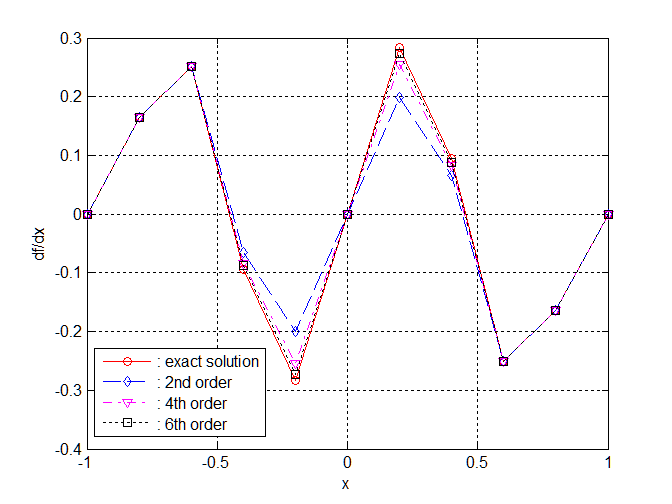
🡪



* **Example Application**

 with h = 0.2





* 1. **Richardson’s extrapolation for Backward Difference formula**
* **Backward difference formula: Similar to the Richardson’s extrapolation method**



Let’s define function , then



Therefore, we can get

(1)

By combining the above results,

 (2)

We can continue to remove the -term in Eq(2) as

 (3)

Or

 (4)

The further removal of the -term in Eq.(4) results in

 (5)

Or

 (6)

The results of above derivation can be summarized as

1st order FD formula using (1):  (7)

2nd order FD formula using (2):  (8)

3rd order FD formula using (4):

 (9)

4th order FD formula using (6):

 (10)

* 1. **Richardson’s extrapolation for Cnetral Difference formula**
* **Central difference formula: Similar to the Richardson’s extrapolation method**

Let’s define function , then





By combining the above results,

 (11)

Removal of of the -term

 (12)

Removal of of the -term



Or

 (13)

As a result

2nd order FD formula using (11):  (14)

4th order FD formula using (12):

 (15)

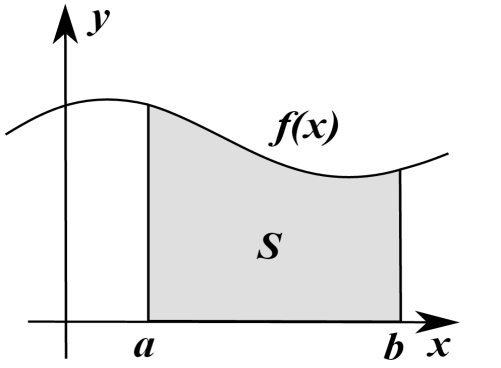
6th order FD formula using (13):



* 1. **Numerical Integration formula**

**(1-5-1) Newton-Cotes Integration Formula**

 where is the approximating function



**Trapezoidal Formula** 

**Simpson’s 1/3 Rule** 

**Simpson’s 3/8 Rule** 

**(1-5-2) Gauss-Quadrature Integration Formula**



* **Transform into the Standard Integration Using Affine Transformation**

🡪

Using Affine transformation, we can redefine above integration as



* **If we use the standard form, the integration can be estimated by integrating the following form**

 (15)

* **General form of n-point Gauss Quadrature Formula**

 (16)

where

: Gauss Quadrature points

: Gauss Quadrature weights

* **Origination of the Gauss-Quadrature formula**

Using the data set , we can interpolate the function  using the Lagrange interpolating polynomials as

 (17)

where



For 

By substituting Eq. (17) into Eq.(15) we can get

 (18)

Therefore, we can conclude that the quadrature weight for each node can be obtained by integrating the corresponding Lagrange interpolating polynomial. Also, it should be recalled that the quadrature nodes for the Legendre-Gauss interpolation is determined by the roots of the Legendre polynomial.

 (19)

* **Partial Integration formula for the Gauss-Quadrature Rule**

Using the data set , let’s consider the partial integration upto.

 (20)

If we define  as

 (21)

Then, the partial integration shown in Eq. (19) can be represented by

If we define  as

 (22)

Therefore, once we have the integration matrix , the partial integration can be accurately estimated using Eq. (21)

* **Computing Quadrature Nodes, Weights and Integration Matrix**

The Legendre-Gauss (LG) quadrature formula uses the nodes defined by the roots of the Legendre polynomials. The N-th order Legendre polynomial can be obtained from the Eigen functions of the sigular Sturm-Lioville problem

 with 

As previously mentioned, there exist a recursive formula which allows us to get the higher order Legendre polynomials.

* + Three term recurrence formula



* + Recurrence relation for derivatives









We can build the quadrature nodes by solving 

The solution of  is straightforward when we use the orthogonality of the Legendre polynomials. The following procedure can be adopted for this purpose.

1. Set the solution of  with 
2. Two roots of  are located at  and . Therefore, we can guess the roots as



Then, we can use the Newton-Raphson method to solve  and . The extremely fast convergence is guaranteed using the initial guesses. And, the required gradients can be estimated using



1. For general case of  , we can use the roots of  to guess the initial solution of the roots since

The first root located at 🡪 

The last root located at 🡪 

The intermediate root located at 🡪 

Therefore, N-point quadrature nodes can be built with the solution of . Then, we can define the Lagrange interpolation polynomials  and they can be expanded in the form of standard polynomial expression as

 (23)

The integration and differentiation of Eq. (23) can be expressed as

 (24)

Therefore, the quadrature weights  shown in Eq. (19) can be obtained using the following formula.

 (25)

Likewise, the partial integration matrix shown in Eq. (21) can be built by

 (26)

1. **Two Different Approaches to the Solution of the System of ODE**
2.  (1)
3. Or
4.  (2)

To solve the problem (1) or (2) using the numerical methods, the time horizon should be divided by multiple time intervals using time nodes such as . After approximating the derivative in Eq. (1) or the integration part of Eq. (2), the solution at each node can be obtained by applying suitable numerical methods. In this sense, the time node can be classified as a collocation point, where we want to get a solution. In a case when the derivative in Eq. (1) is approximated first, the corresponding method is called by the differentiation method. Whereas, the integration method first approximates the integration part in Eq. (2).

**(2-1) Differentiation method**

In this method, the first derivative in Eq. (1) is typically approximated using the finite difference formula. In such a case, the uniform interval with is conveniently adopted. At , Eq. (1) can be represented by

 (27)

The first time-derivative can be approximated by the backward difference formula, which is reasonable for the initial value problem since the later states are affected by the previous system states. As an example, if we use the first order backward difference formula shown in Eq. (7), Eq. (27) can be approximated by

 (28)

The state  is already known but Eq. (28) have the unknown state  in both sides. Therefore, we can iteratively solve the following NAE using the Newton-like method.

 (29)

When we adopt the standard Newton-Raphson method, the solution can be updated with the initial guess of solution by

 (30)

Therefore, the above iterative procedure requires the time-consuming evaluation of the Jacobian matrix  and the LAE solution.

Since Eq.(28) is a fixed-point problem like , we can adopt the fixed-point iterative solver. In a case when the Piccard method is adopted, the solution can be iteratively updated by

 (31)

When 2nd- and 3rd-order backward difference formula are used as shown in Eq. (8) and Eq. (9), The corresponding fixed-point iterations can be defined by Eq. (32) and Eq. (33).

 (32)

 (33)

**(2-2) Integration method**

In this method, the integration part of Eq.(2) is first approximated using a suitable integration formula. When the trapezoidal formula is applied, Eq.(2) can be approximated at  as

1.  (2)

 (34)

Eq. (34) is also an NAE but it has a form of the fixed-point problem, Therefore, we can use the following update formula.

 (35)

The integration part shown in Eq. (2) also can be approximated using the Gauss quadrature integration formula. For this purpose, the integration part should be represented by the standard form using the affine transformation over  as

 (36)

Then, the partial integration up to  can be approximated by using the integration matrix and Eq. (36) can be approximated by

 (37)

The solution at  can be obtained using the quadrature formula.

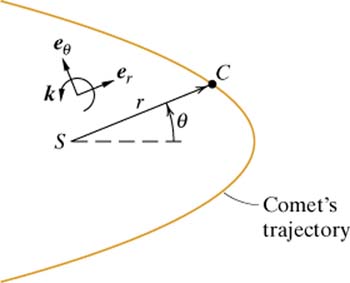
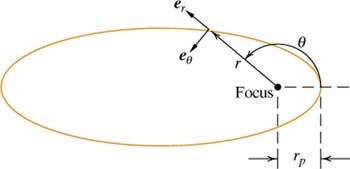
 (38)

Eq. (37) needs the iterative solution process but  in Eq. (38) can be simply updated with the fully converged solution of Eq. (37) at intermediate nodes. Therefore,

 (39)

1. **Comparative Study on Numerical Accuracy of ODE solvers**

**(6-1) Problem Statement: Elliptic Orbit**

** **

The satellite motion equation in 2D space can be described by

 (6-1)

Let

🡪  (6-2)

Then, we can obtain

 with 

With eccentricity , the conditions at the perigee can be defined by

When , we can define the initial condition as



**(6-2) Exact solution**

The conic section equation can be represented by

 🡨 

Orbital parameters can be derived as

* Semi major axis 
* Semi minor axis 
* Perigee radius 
* Apogee radius 
* Period 
* Mean anomaly 
* Eccentric anomaly 

 and 

Computational Procedure of the exact solution

1. With given time , compute 
2. Compute  by iteratively solving 
3. Compute  using 
4. Compute  using 
5. Compute 
6. Compute 

**(6-3) Summary**

1. **Motion equation and initial condition**

** with **

If we use the following non-dimensional time-variable,



Then,

 with ****

1. **Exact solution**
2. With given time , compute  (Mean anomaly)

Where  , , 

1. Compute  by iteratively solving  (Eccentric anomaly)
2. Compute  using 
3. Compute  using 
4. Compute 
5. Compute 

**(6-4) ODE Solvers**

1.  (1)
2. Explicit Euler Method



1. Explicit Heun’s Method



1. Explicit 4-stage Runge-Kutta Method



1. Implicit 1st order backward difference method with the Piccard iterative algorithm



1. Implicit 2nd order backward difference method with the Piccard iterative algorithm



1. Implicit 3rd order backward difference method with the Piccard iterative algorithm



1. Implicit 4th order backward difference method with the Piccard iterative algorithm



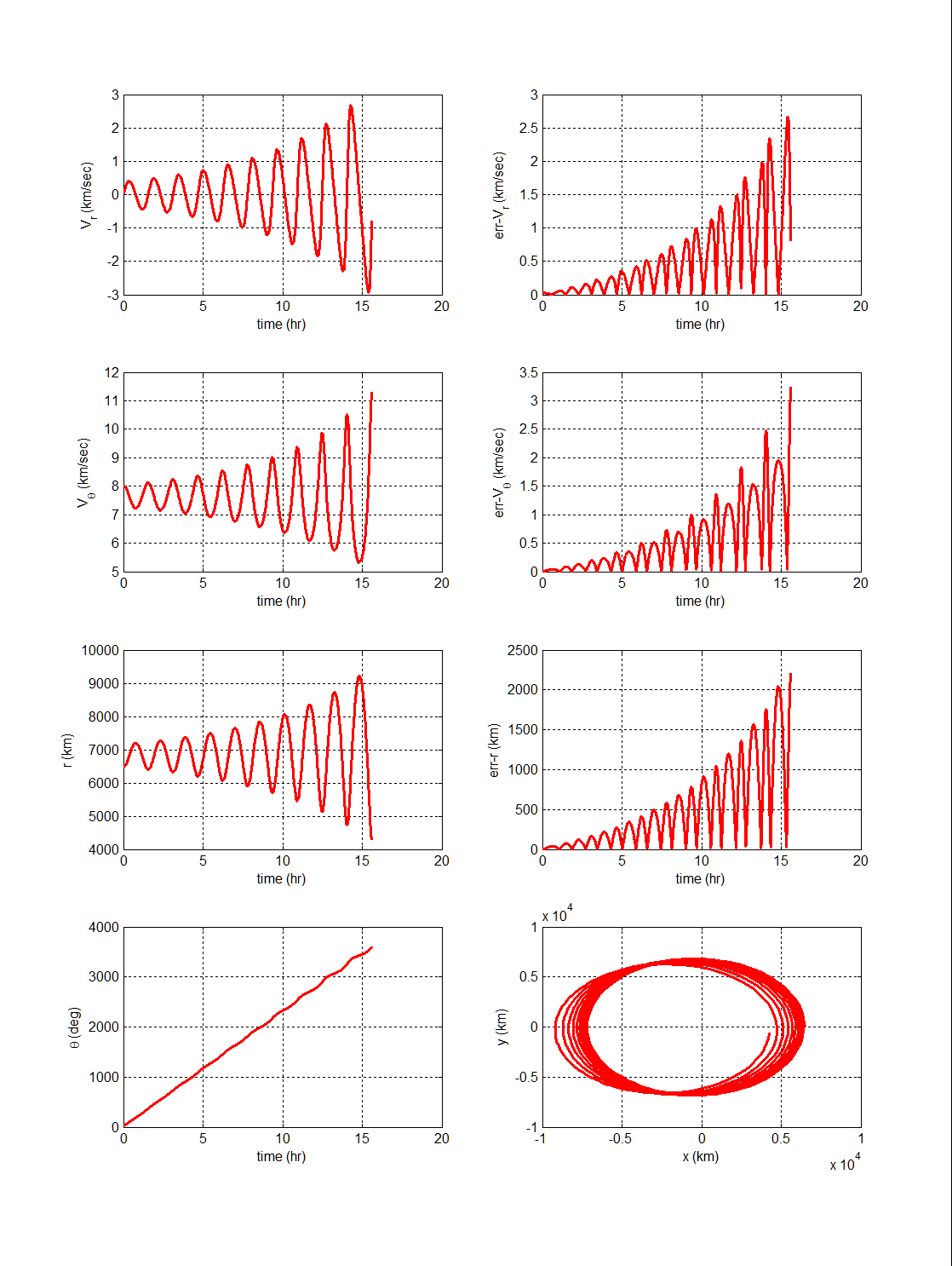
1. Pseudospectral Method with the Piccard iterative algorithm



**(6-5) Results**

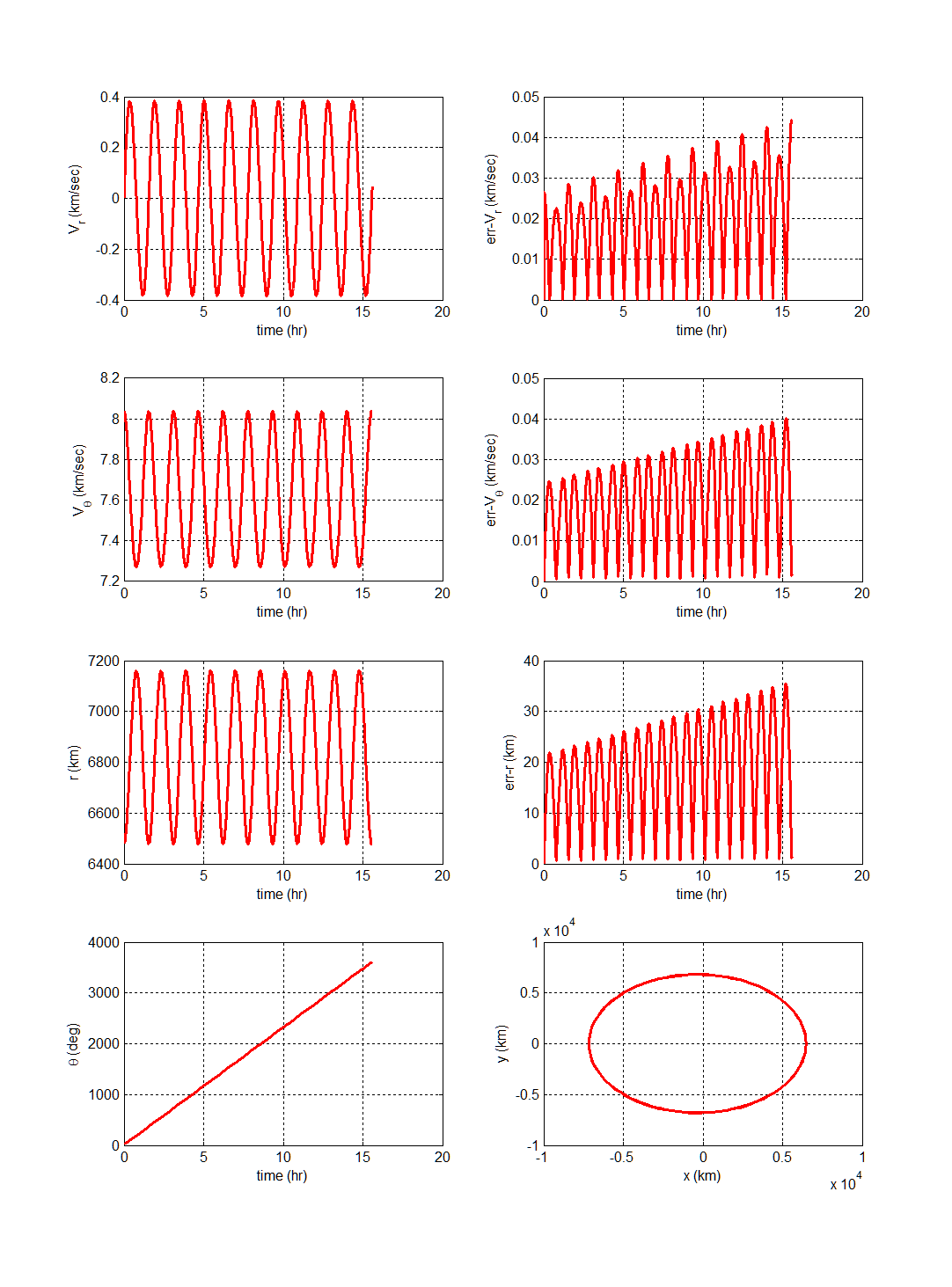
1. **Explicit Euler: e= 0.05 No\_rev=10 Node/rev=100**





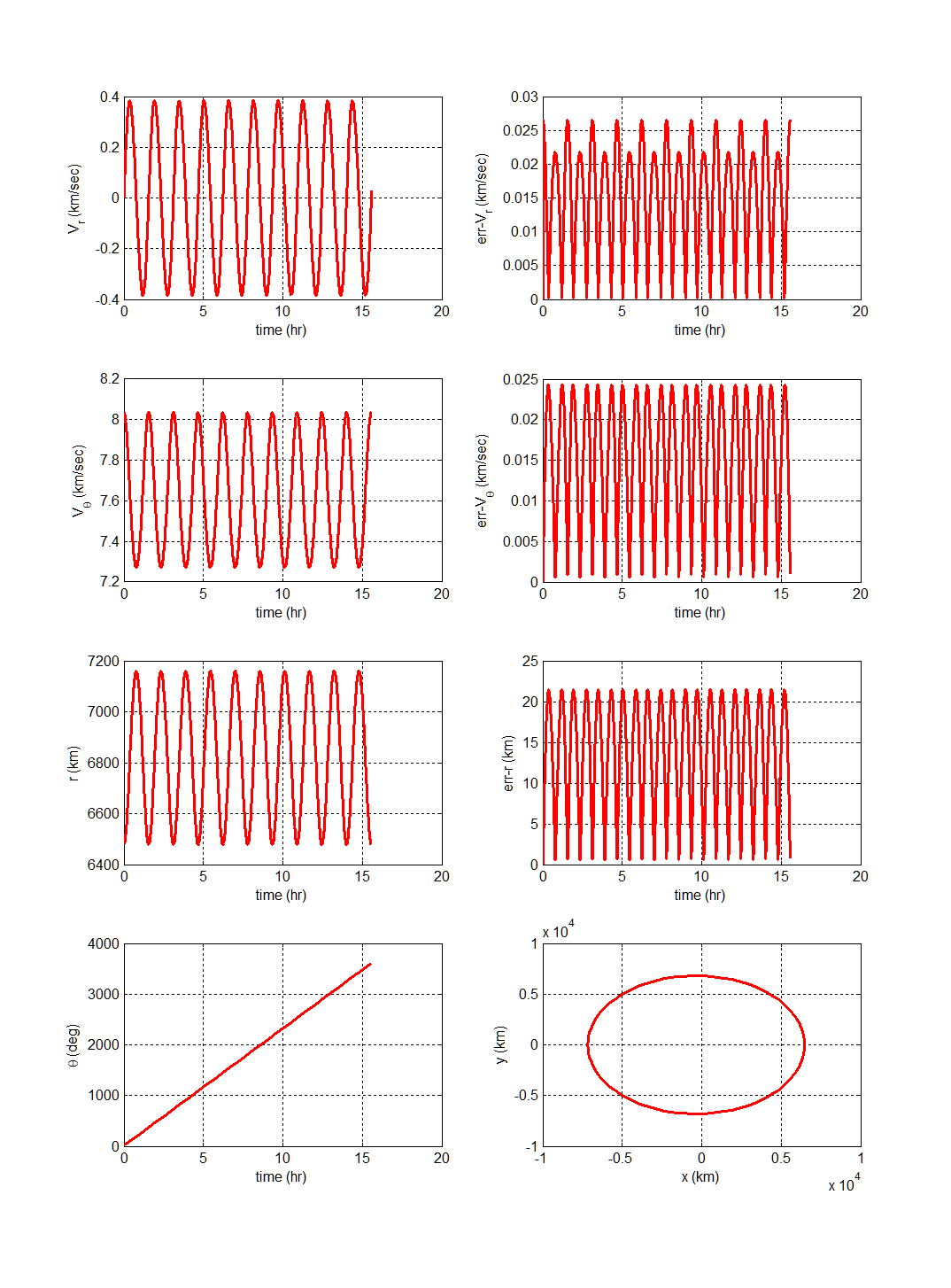
1. **Explicit Heun : e= 0.05 No\_rev=10 Node/rev=100**





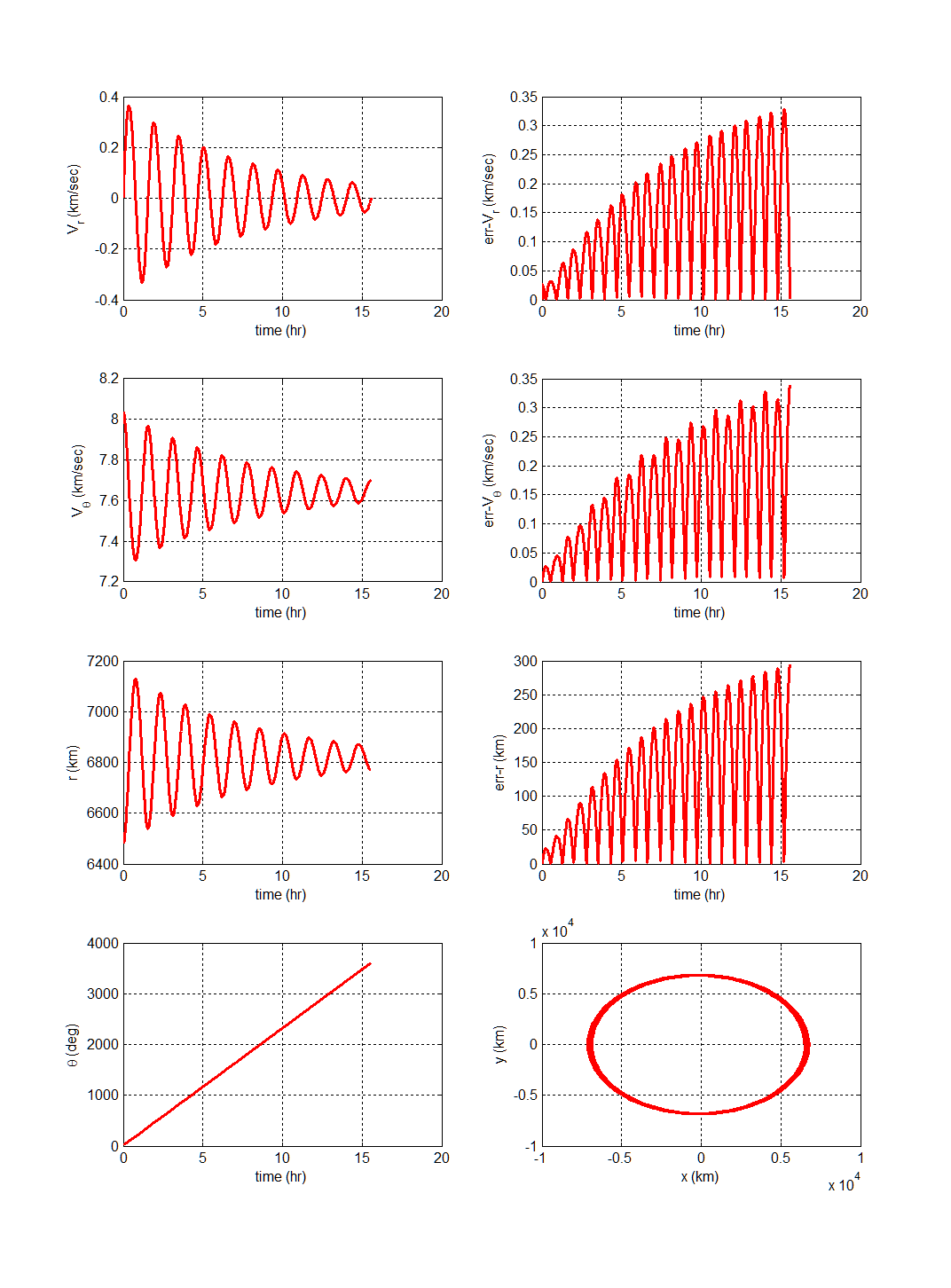
1. **Explicit 4-stage Runge-Kutta: e= 0.05 No\_rev=10 Node/rev=100**

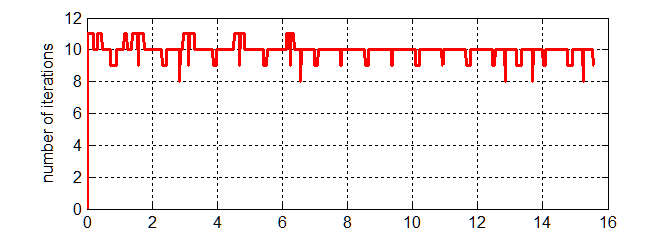




1. **Implicit 1st-order Backward Difference: e= 0.05, No\_rev=10, Node/rev=100**

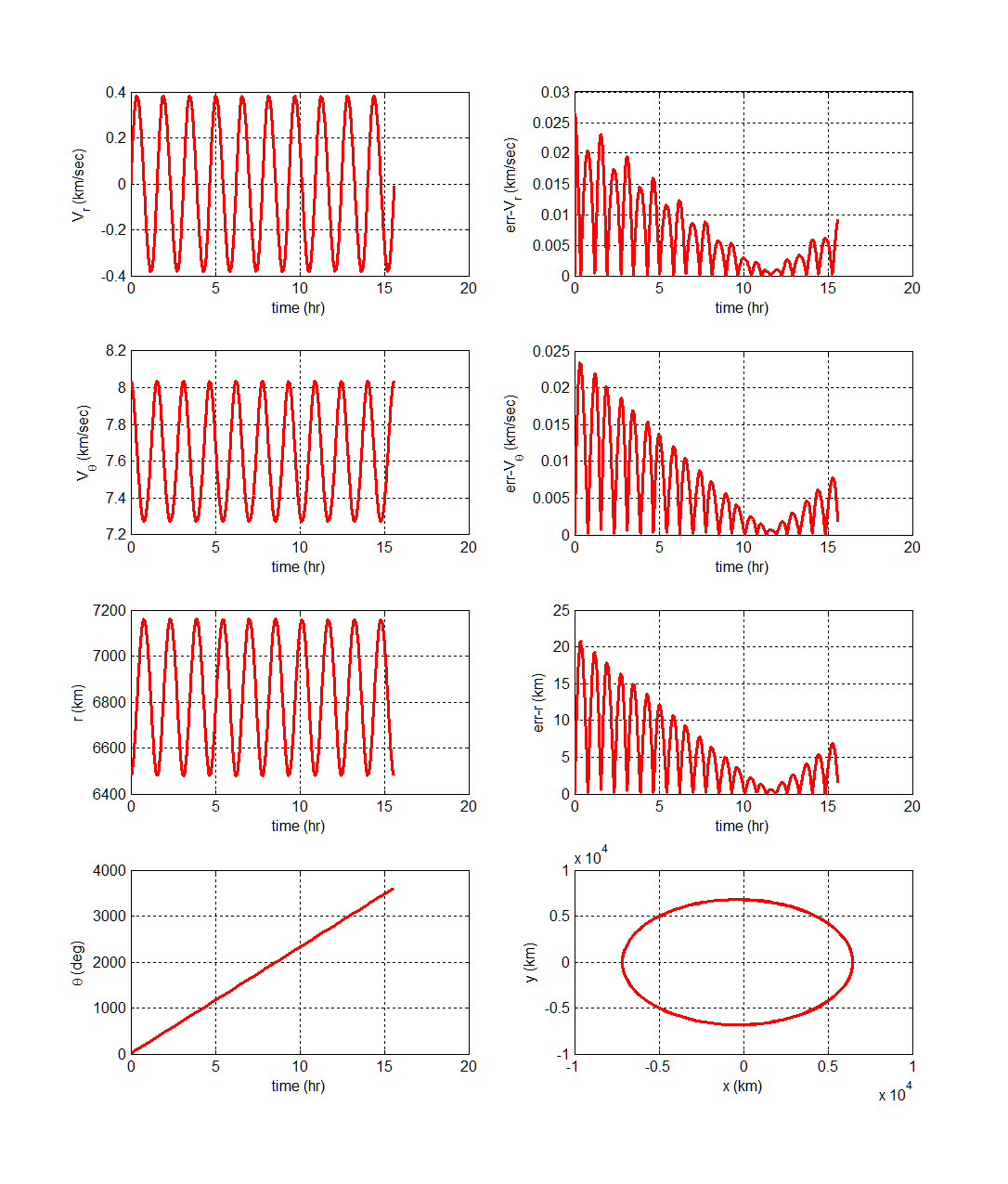


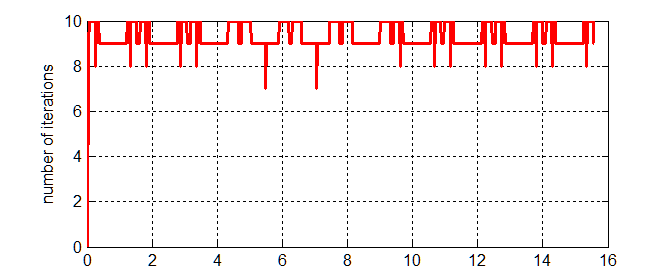




1. **Implicit 2nd -order Backward Difference: e= 0.05, No\_rev=10, Node/rev=100**

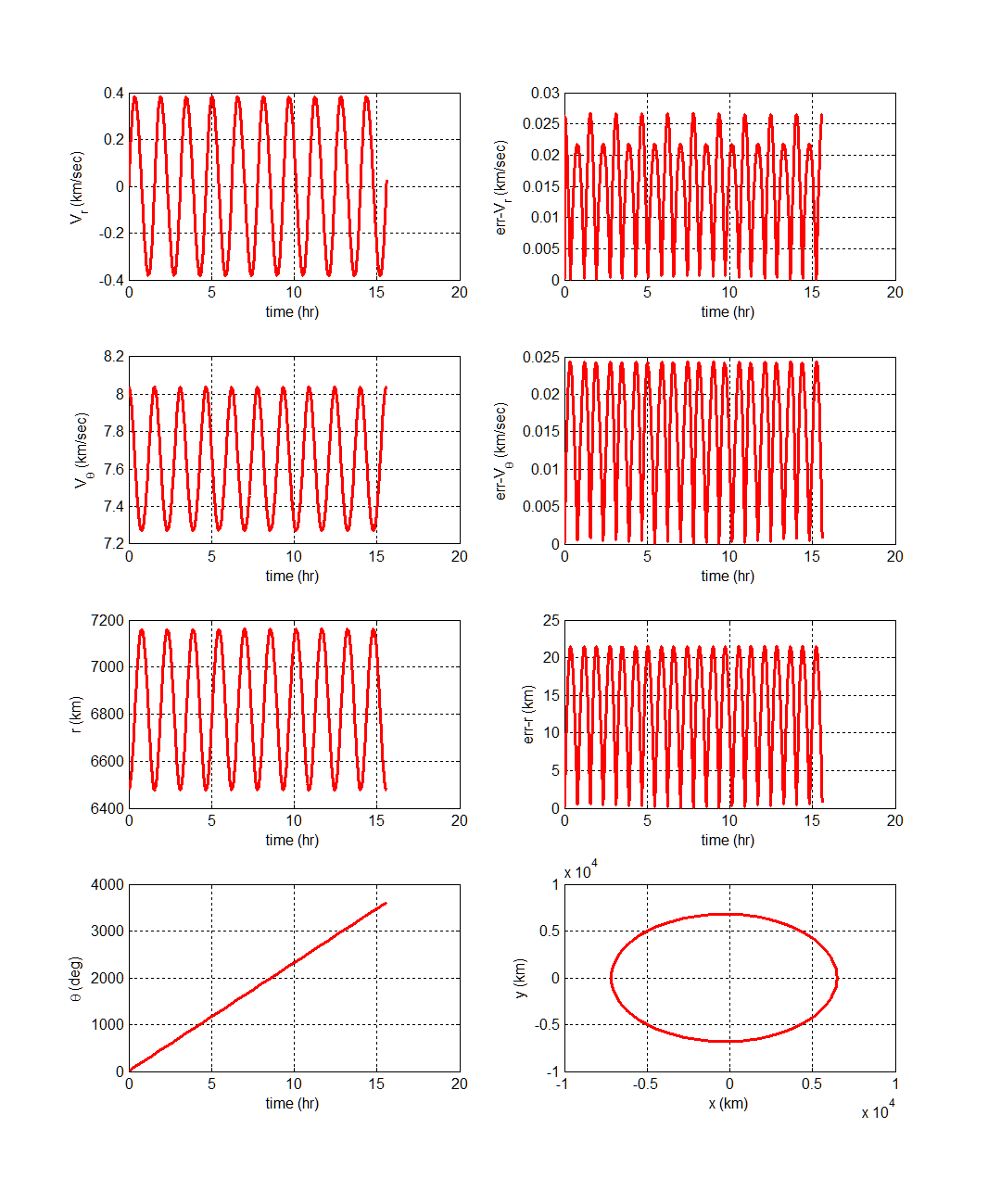


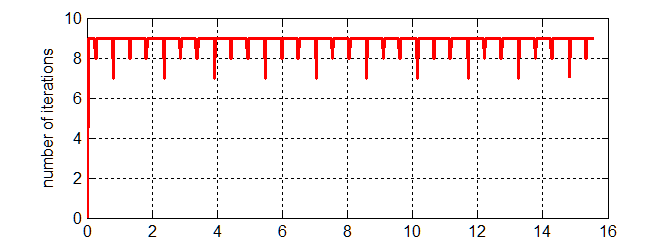




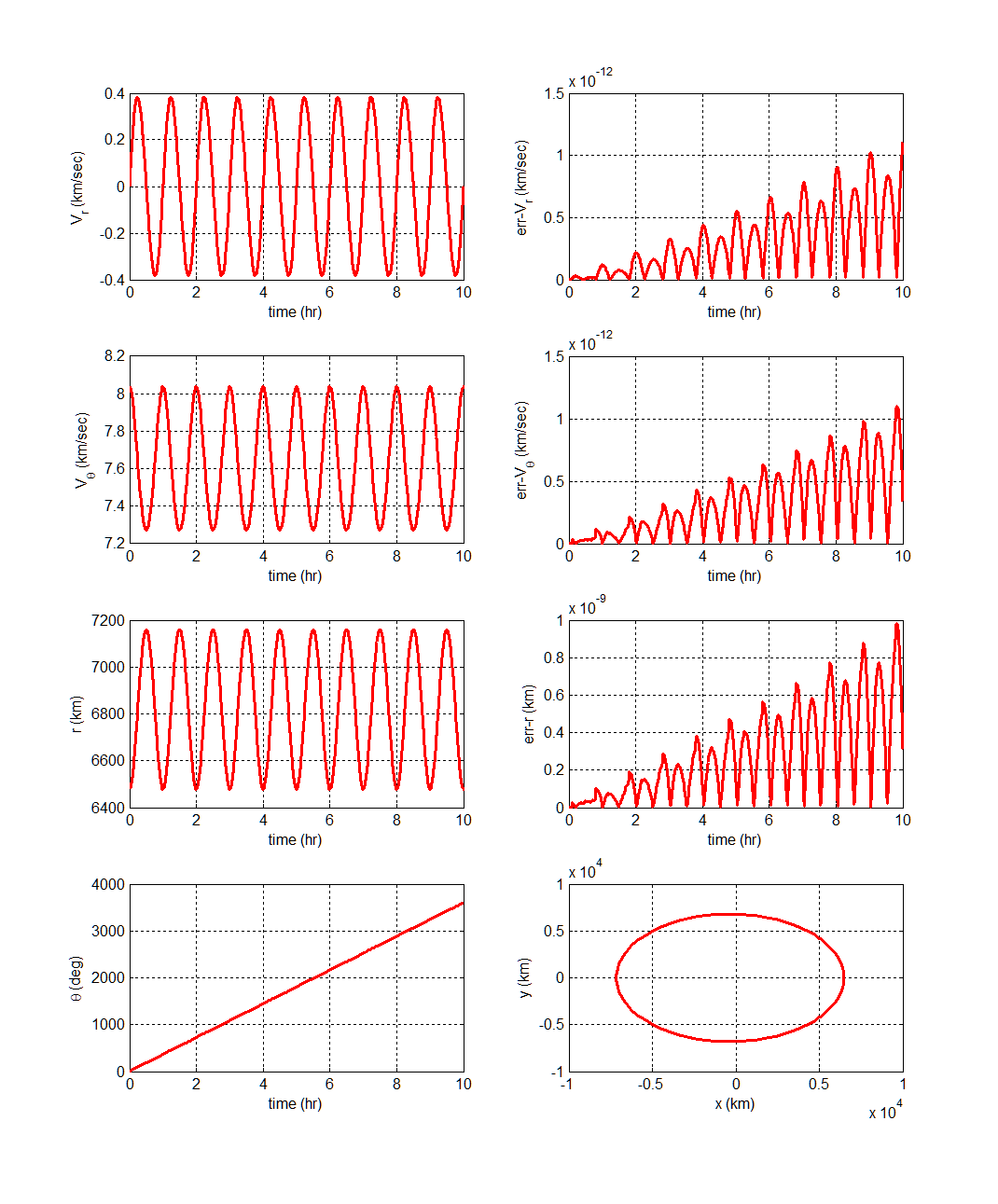
1. **Implicit 3rd-order Backward Difference: e= 0.05, No\_rev=10, Node/rev=100**

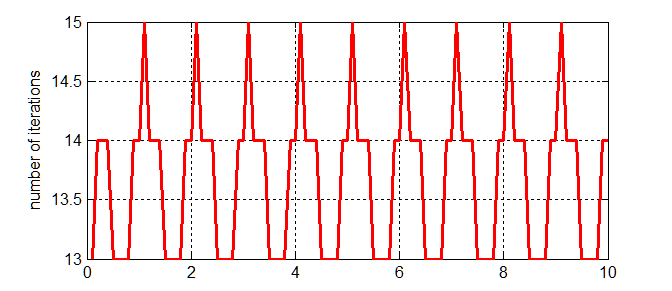






1. **Pseudo-Spectral Integrator : e= 0.05, No\_rev=10, Node/rev=10, No\_Qnode=11**





**Appendix A: Stiff ODE**

1. Problem #1: 

  Exact solution 

1. Problem #2: 

 Exact solution 

1. Problem #3: 

 Exact solution 

1. Problem #4: 

Exact solution 

