MA 880 Project: Lie Group Based Numerical Analysis of a System of Linear ODEs Evolving on $SO(3)$

Julian Christopher

November 30, 2020

1 Introduction

Suppose that we are performing a numerical analysis on a matrix ODE of the form

$$\begin{cases} y' = a(t) \cdot y \\ y(0) = y_0 \end{cases}, \tag{1}$$

with $a, y \in GL(n)$, and that we know that y is confined to a Lie group $G \subseteq GL(n)$; it may be useful to retain this quality in our numerical results. It is in general not the case that a numerical solution will stay constrained to the proper submanifold, e.g. a solution that should be constrained to $SO(n)$ may have a determinant that is close to, but not exactly 1, due to truncation error. An approach that would restrain the spproximate solution to the required Lie group, proposed by Magnus in [3] in 1954, is to find an $\Omega(t) \in \mathfrak{g}$ such that

$$y = \exp(\Omega(t))y_0; \tag{2}$$

in general, one can use any $f : \mathfrak{g} \to G$ that is injective and attempt to find a $v(t) : I \to \mathfrak{g}$ such that

$$y = f(v(t)) \cdot y_0;$$

exp is not the only option, though it is the most general. In this paper we will use the Cayley transform

$$\begin{aligned} \operatorname{cay} : \mathfrak{o}_J(n) &\longrightarrow O_J(n) \\ v &\mapsto \left(I - \frac{1}{2}v\right)^{-1}\left(I + \frac{1}{2}v\right). \end{aligned} \tag{3}$$

as our f. The Cayley transform is bijective as long as $(I + {}^{v}/_{2})$ is invertible, i.e. when -1 is not an eigenvalue of $v(t)$; it is also much easier to work with than the Magnus solution using exp, as $\exp(v)$ must be approximated using a series containing nested commutators, while $\operatorname{cay}(v)$ can be computed directly. The trade off is that the Cayley transform is only

a mapping between a Lie algebra and it's corresponding Lie group for J-orthogonal matrix groups defined by

$$O_J(n) = \{A \in GL(n) : A^T JA = J\},$$

with corresponding Lie algebra

$$\mathfrak{o}_J(n) = \{B \in \mathfrak{gl}_n(\mathbb{R}) : B^T J + JB = 0\}.$$

We will derive a numeric algorithm for solving $J-$orthogonal systems and apply it to a text example.

2 General Theory

Considering equation (1), we think of $a(t)y$ as a vector field on G, i.e. $a(t)y \in T_yG$.

$$\begin{array}{c} TG \\ {\scriptstyle a(t)}\uparrow \\ G \end{array}$$

We a priori will chose a bijective mapping $\phi : \mathfrak{g} \to G$ allowing us to create a correspondence between $\mathfrak{g}$ and G; the goal being to frame (1) as a problem on the vector space $\mathfrak{g}$ rather than the manifold G.

$$\begin{array}{ccc} & & TG \\ & & {\scriptstyle a(t)}\uparrow \\ \mathfrak{g} & \longleftrightarrow & G \end{array}$$

We establish this connection using the Lie algebra action defined by

$$\begin{aligned} \lambda : \mathfrak{g} \times G &\to G \\ (v, y) &\to \ \phi(v) \cdot y; \end{aligned}$$

where $\cdot : G \times G \to G$ is a left translation Lie group action. For a fixed $p \in G$, we define the function

$$\begin{aligned} \lambda_p : \mathfrak{g} &\to G \\ v &\mapsto \lambda(v, p); \end{aligned}$$

establishing the correspondence depicted in the diagram:

$$\begin{array}{ccc} & & TG \\ & & {\scriptstyle a(t)}\uparrow \\ \mathfrak{g} & \overset{\lambda_p}{\longleftrightarrow} & G \end{array}$$

We can take the derivative of the map λ_p to obtain a mapping from $T\mathfrak{g} \to TG$. Note that for a fixed base-point $v \in \mathfrak{g}$, we have the derivative of λ_p as $T_v\lambda_p : T_v\mathfrak{g} \to T_{\lambda_p(v)}G \simeq \mathfrak{g}$.

$$\begin{array}{ccc} T\mathfrak{g} & \xrightarrow{T\lambda_p} & TG \\ \tilde{f} \uparrow & & \uparrow a(t) \\ \mathfrak{g} & \xleftarrow{\lambda_p} & G \end{array}$$

It is clear that there is a function $\tilde{f}(t, v)$ that closes the diagram above, and so

$$T\lambda_p \circ \tilde{f}(v, t) = a(t) \circ \lambda_p(v)\,.$$

Theorem 4 from [4] specifies that

$$\tilde{f}(t, v) = d\phi^{-1}(a(t) \cdot \lambda_p(v)) \tag{4}$$

and so, if we write (1) as $y'(t) = a(t) \circ \lambda_{y_0}(v)$ we see that $y(t) = \lambda_{y_0}(v(t))$ and

$$\begin{aligned} v'(t) &= \tilde{f}(t, v) \\ &= d\phi^{-1}(a(t) \cdot \lambda_{y_0}(v))\,. \end{aligned} \tag{5}$$

3 The Cayley Transform

If G is a J-orthogonal matrix Lie group, we let $\phi(v) = cay(v)$; then, by (5) solving (1) is equivalent to solving

$$v'(t) = d\mathrm{cay}^{-1}(a(t) \cdot \mathrm{cay}(v(t)))$$

Equation (3) can be written in a more explicit form, the following proof uses a computation from [2].

Theorem 3.1. *For $y \in O_J(n)$, differential equations of the form*

$$y' = a(t) \cdot y$$

can be solved by

$$y(t) = cay(v(t))y_0 \tag{6}$$

where

$$v'(t) = dcay^{-1}(a(t) \circ cay(v)) \tag{7}$$

$$= a - \frac{1}{2}[v, a] - \frac{1}{4}vav. \tag{8}$$

Proof. Equations (6) and (7) were proved in the previous section. We can calculate (8) by

$$
\begin{aligned}
y &= \operatorname{cay}(v) y_0 \\
y &= \left(I - \frac{v}{2}\right)^{-1} \left(I + \frac{v}{2}\right) y_0 \\
y' &= \left(I - \frac{v}{2}\right)^{-1} \frac{v'}{2} \left(I - \frac{v}{2}\right)^{-1} \left(I + \frac{v}{2}\right) + \left(I - \frac{v}{2}\right)^{-1} \frac{v'}{2} \\
\left(I - \frac{v}{2}\right) y' &= \frac{v'}{2} \left(I - \frac{v}{2}\right)^{-1} \left(I + \frac{v}{2}\right) + \frac{v'}{2} \\
\left(I - \frac{v}{2}\right) y' &= \frac{v'}{2} \left[\left(I - \frac{v}{2}\right)^{-1} \left(I + \frac{v}{2}\right) + I\right] \\
\left(I - \frac{v}{2}\right) y' &= \frac{v'}{2} \left(I - \frac{v}{2}\right)^{-1} \left[\left(I + \frac{v}{2}\right) + \left(I - \frac{v}{2}\right)\right] \\
v' &= \left(I - \frac{v}{2}\right) y' \left(I - \frac{v}{2}\right) \\
v' &= \left(I - \frac{v}{2}\right) a\operatorname{cay}(v) \left(I - \frac{v}{2}\right) \\
v' &= \left(I - \frac{v}{2}\right) a \left(I - \frac{v}{2}\right)^{-1} \left(I + \frac{v}{2}\right) \left(I - \frac{v}{2}\right) \\
v' &= \left(I - \frac{v}{2}\right) a \left(I + \frac{v}{2}\right) \\
v' &= a - \frac{1}{2}[v, a] - \frac{1}{4} vav \,.
\end{aligned}
$$

□

4 From Theory to Numerics

We will apply an implicit 4th and 6th order Runge-Kutta schemes, based on the Gauss-Legendre polynomials, to (8).For a general ODE written as

$$Y = f(t, Y) \tag{9}$$

the s-stage Runge-Kutta scheme for a numerical approximation of Y is given in equation 228 of [1] as

$$
\begin{aligned}
K_i &= Y_n + h \sum_{j=1}^{s} a_{ij} f(t_n + c_j h, K_j) \\
Y_{n+1} &= Y_n + h \sum_{i=1}^{s} b_i f(t_n + c_i h, K_i). \qquad (10)
\end{aligned}
$$

Where a_{ij}, b_i and c_j are given by the the Butcher Tableau. The Butcher Tableaus for the 4th and 6th order schemes are given in equation 237 of [1] as

$$\begin{array}{c|cc} \frac{3-\sqrt{3}}{6} & \frac{1}{4} & \frac{3-2\sqrt{3}}{12} \\ \frac{3+\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}; \tag{11}$$

and

$$\begin{array}{c|ccc} \frac{5-\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}+\frac{\sqrt{15}}{24} \\ \frac{5+\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}. \tag{12}$$

Solving the the differential equation (8) with the scheme (10) gives us equation 8 from [4] which we will write in our notation as

$$K_i = v_n + h\sum_{j=1}^{s} a_{ij}\left(A(t_n+c_ih) - \frac{1}{2}[K_j, A(t_n+c_ih)] - \frac{1}{4}K_iA(t_n+c_ih)K_i\right) \tag{13}$$

$$v_{n+1} = v_n + h\sum_{i=1}^{s} b_i\left(A(t_n+c_ih) - \frac{1}{2}[K_i, A(t_n+c_ih)] - \frac{1}{4}K_iA(t_n+c_ih)K_i\right). \tag{14}$$

We will solve equation (13) iteratively using.

$$\mathbf{K}^{[\ell]} = v_n + hG(h, \mathbf{K}^{[\ell-1]}), \tag{15}$$

with

$$G_n(h,\mathbf{K}) = \sum_{j=1}^{s} a_{ij}\left(A(nh+c_ih) - \frac{1}{2}[K_j, A(nh+c_ih)] - \frac{1}{4}K_iA(nh+c_ih)K_i\right),$$

where

$$\mathbf{K} = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_s \end{bmatrix}.$$

For an error estimate we will compare the results we obtain using the Lie group method with a standard 7-th order Runge Kutta Scheme, the Butcher tableau for this method is given in equation 236 of [1].

5 Results of Test Problem

The test problem we will use is of the form of problem 1 from [4]. We will look at the system evolving on $SO(3)$ so that the computational times are reasonable, and so we can visualize

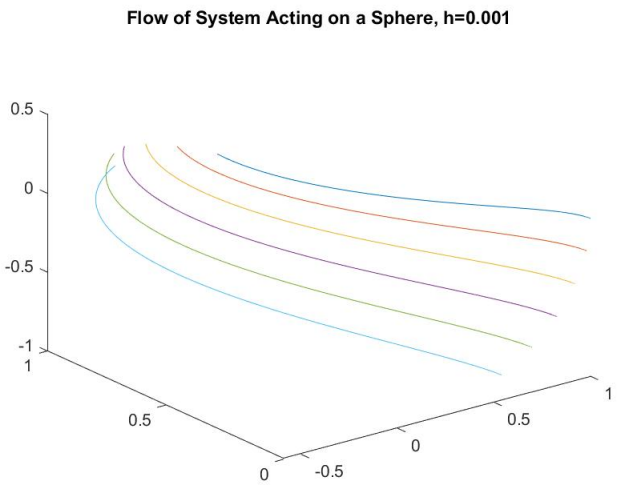


Figure 1: Flow of 6^{th} order Lie group based numerical solution from $t = 0$ to $t = 2.5$

the results in 3 dimensions. Define the upper triangular entries of $a(t)$ as

$$a(t)_{ij} = (-1)^{i+j}\frac{i}{j+1}t^{j-i}, \qquad 1 \leq i < j \leq n,$$

and the lower triangular entries as

$$a_{ji} = -a_{ij}, \qquad 1 \leq i < j \leq n.$$

The figures 1 to 3 all evolve from $t = 0$ to $t = 2.5$ and use $h = 0.001$; due to nonlinearity in time, as t increases the timestep size must be decreased in order for the algorithm to remain stable. We will perform tests for higher values of t later in this section. Analysing a system on $SO(3)$ allows us to visualize the flows of the system by having our solution act on $\mathbb{R}^3$ as seen in figure 1. Figure 2 shows the error for both the 4^{th} and 6^{th} order Lie based Gauss-Legendre methods when compared to a 7^th order Runge-Kutta method. Figure 3 shows the distance from the solution to the manifold, easily calculated as $|1 - ||y||_2|$; the distance of the standard 7^{th} order Runge-Kutta scheme from the manifold is on the same order as the expected error, while the Lie group methods remain very close and display an oscillatory behaviour, this is likely due to machine rounding as theoretically they should not leave the manifold at all.

In figures 4 and 5 we look at the error and distance to manifold for the same situation with time step $h = 0.00001$. We see that the error improves but that the distance to the manifold stays the same with the exception of the standard Runge-Kutta approximation which follows expected error.

Next, we can calculate the numerical solution for $t = 2.5$ to $t = 3$ with step size $h = 0.00001$, The flow, error, and distance to the manifold for this situation are shown in figures 6, 7, and 8. We see that error is much higher in this situation.

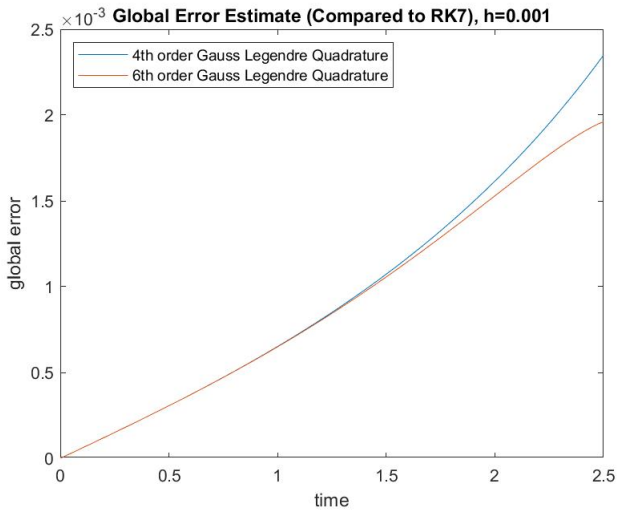


Figure 2: Error of 4^{th} and 6^th order Lie group based numerical solution

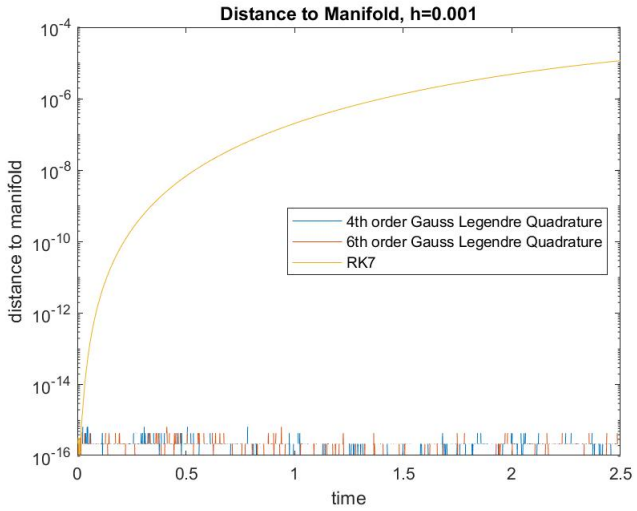


Figure 3: Distance between the manifold and the 4^{th} and 6^th order Lie group based, and 7^{th} order non Lie group based numerical solutions

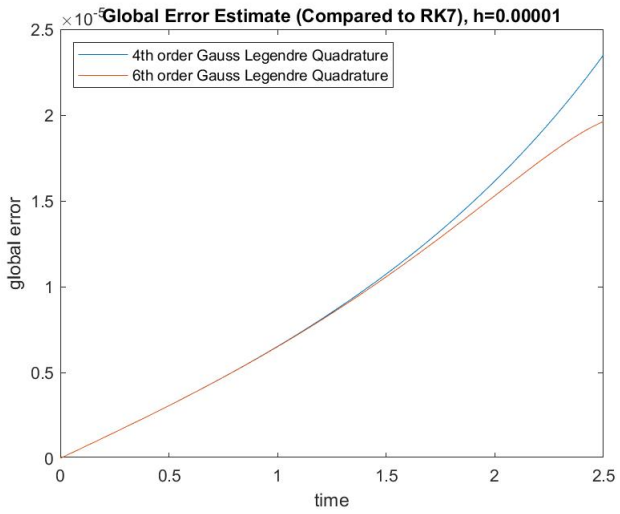


Figure 4: Error of 4^{th} and 6^th order Lie group based numerical solution

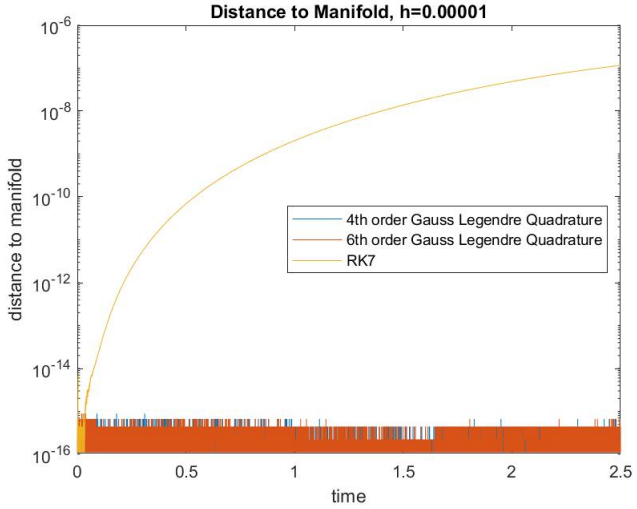


Figure 5: Distance between the manifold and the 4^{th} and 6^th order Lie group based, and 7^{th} order non Lie group based numerical solutions

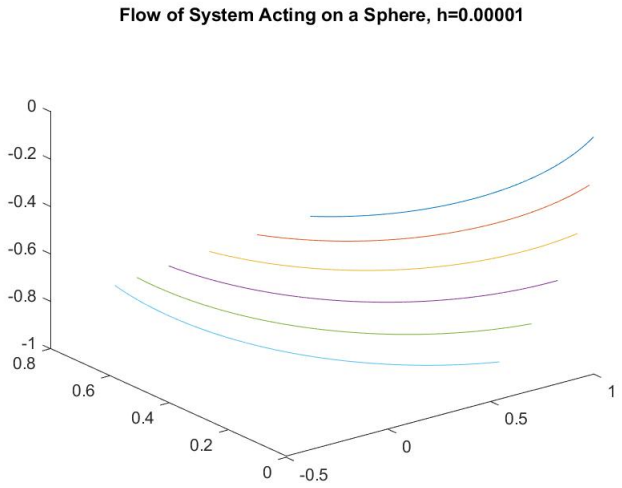


Figure 6: Flow of 6^{th} order Lie group based numerical solution from $t = 2.5$ to $t = 3$

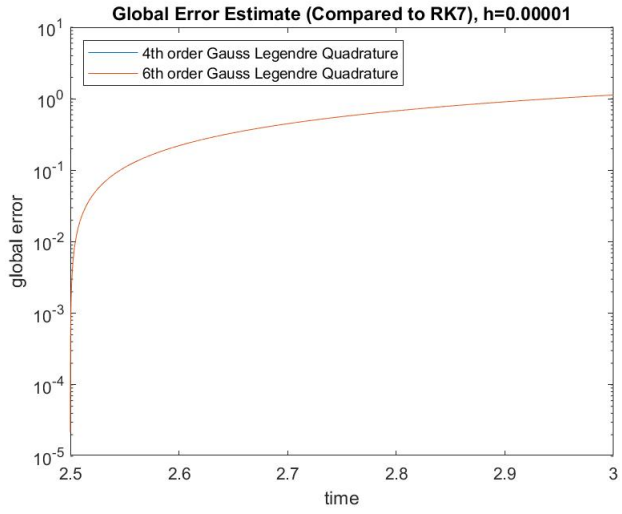


Figure 7: Error of 4^{th} and 6^th order Lie group based numerical solution

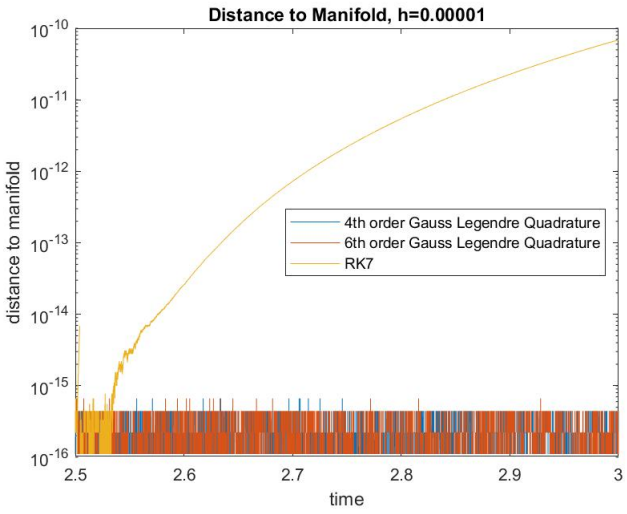


Figure 8: Distance between the manifold and the 4^{th} and 6^th order Lie group based, and 7^{th} order non Lie group based numerical solutions

6 Concluding Remarks

We see from the experiments in the previous section that, for our test cases, the Lie group based methods stay very close to the manifold while a standard Runge-Kutta scheme will move away from the manifold over time according to the order of it's error. The Lie group methods have inherent error but remain confined to the Lie group on which they are known to evolve.

References

[1] Sergio Blanes, Fernando Casas, Jose-Angel Oteo, and José Ros. The magnus expansion and some of its applications. *Physics reports*, 470(5-6):151–238, 2009.

[2] Arieh Iserles. On cayley-transform methods for the discretization of lie-group equations. *Foundations of Computational Mathematics*, 1(2):129–160, 2001.

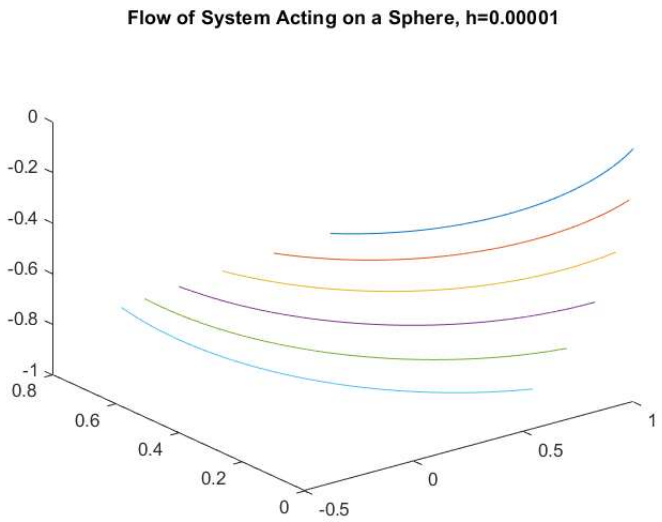
[3] Wilhelm Magnus. On the exponential solution of differential equations for a linear operator. *Communications on pure and applied mathematics*, 7(4):649–673, 1954.

[4] Arne Marthinsen and Brynjulf Owren. Quadrature methods based on the cayley transform. *Applied numerical mathematics*, 39(3-4):403–413, 2001.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solution using Cayley transform and 6-th order implicit Runge-Kutta %
%Methods                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ButcherTableaux = QGSix;    %Butcher tableaux to be used for RK with Cayley tranform
s = size(ButcherTableaux,2)-1; %number of RK stages
n = 3;                        %size of matrix
h = 0.00001;                     %timestep size
epsilon = 0.00001;              % Distance between iterations of RK method
tInitial = 2.5;
tFinal = 3;                       %final time
ts1 = round(tInitial/h);
ts = round((tFinal-tInitial)/h);              %number of timesteps
v6 = zeros(n,n,s,ts);              %Lie algebra element at time t (matrix,s copies of matrix,timestep number)
y6 = zeros(n,n,ts);                %Lie group element time t
y0 = eye(n);                      %initial condition on Lie group
Knew = zeros(n,n,s);              %RK intermediate stages current result (matrix,stage number)
diff = zeros(1,s);                %Difference between iterations (by stages)
c = zeros(1,ts);                  %Iteration counter
f = waitbar(0);

for t = 1:(ts-1)    %Calculate v for each timestep
 b = true;
 Kold = zeros(n,n,s);
 while b        %Solve nonlinear matrix equation by iteration
   Knew = v6(:,:,:,t) + h*G(Kold,t+ts1-1,h,s,n,ButcherTableaux);
   for i = 1:s
        diff(i) = norm(Knew(:,:,i)-Kold(:,:,i));
   end
   if max(diff) < epsilon
       b = false;
   end
   Kold = Knew;
   c(t) = c(t)+1;
 end
 for m = 1:s
     v6(:,:,m,t+1) = v6(:,:,m,t) + h*C(Knew,t+ts1-1,h,s,n,ButcherTableaux);
 end
 waitbar(t/ts,f);
end
close(f);

for t = 1:ts
    y6(:,:,t) = cay(v6(:,:,1,t),n)*y0;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Act on R3 and graph flow lines for visualization purposes %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

p1 = zeros(3,ts);                 %curve on manifold(R3)
p2 = zeros(3,ts);
p3 = zeros(3,ts);
p4 = zeros(3,ts);
p5 = zeros(3,ts);
p6 = zeros(3,ts);
p1i = transpose([1,0,0]);     %Initial condition on manifold
%establish several initial conditions

p2i = rot1*p1i;
p3i = rot2*p1i;
p4i = rot3*p1i;
p5i = rot4*p1i;
p6i = rot5*p1i;

%solutions to 6th order Lie method act on R3
for t = 1:ts
    p1(:,t) = y6(:,:,t)*p1i;
end

for t = 1:ts
    p2(:,t) = y6(:,:,t)*p2i;
end

for t = 1:ts
```

```
    p3(:,t) = y6(:,:,t)*p3i;
end

for t = 1:ts
    p4(:,t) = y6(:,:,t)*p4i;
end

for t = 1:ts
    p5(:,t) = y6(:,:,t)*p5i;
end

for t = 1:ts
    p6(:,t) = y6(:,:,t)*p6i;
end
%Plot flow curves
plot3(p1(1,:),p1(2,:),p1(3,:),p2(1,:),p2(2,:),p2(3,:),p3(1,:),p3(2,:),p3(3,:),p4(1,:),p4(2,:),p4(3,:),p5(1,:),p5(2,:),p5(3,:),p6(1,:),p6(2
title('Flow of System Acting on a Sphere, h=0.00001')
saveas(gcf,'D:\Documents\Modelling Project\flow2.jpg')
```

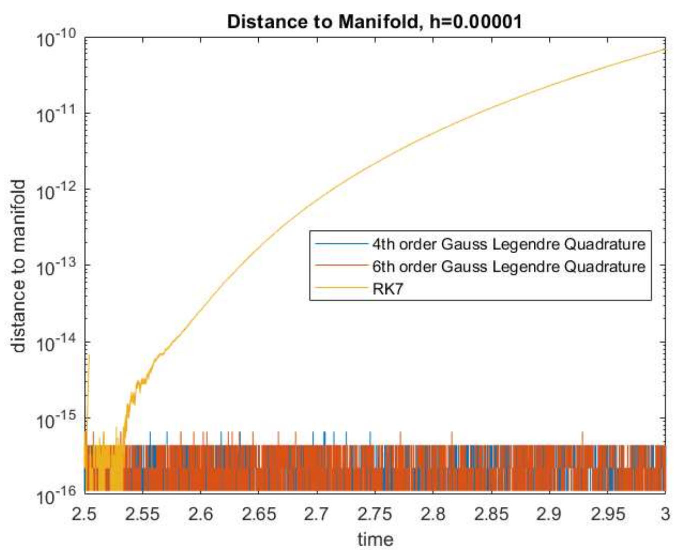



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solution using Cayley transform and 6-th order implicit Runge-Kutta %
%Methods                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ButcherTableaux = QGFour;    %Butcher tableaux to be used for RK
s = size(ButcherTableaux,2)-1; %number of RK stages
v4 = zeros(n,n,s,ts);             %Lie algebra element at time t (matrix,s copies of matrix,timestep number)
y4 = zeros(n,n,ts);               %Lie group element time t
Knew = zeros(n,n,s);             %RK intermediate stages current result (matrix,stage number)
diff = zeros(1,s);               %Difference between iterations (by stages)
c = zeros(1,ts);                 %Iteration counter
f = waitbar(0);

for t = 1:(ts-1)    %Calculate v for each timestep
 b = true;
 Kold = zeros(n,n,s);
 while b        %Solve nonlinear matrix equation by iteration
   Knew = v4(:,:,:,t) + h*G(Kold,t+ts1-1,h,s,n,ButcherTableaux);
   for i = 1:s
        diff(i) = norm(Knew(:,:,i)-Kold(:,:,i));
   end
   if max(diff) < epsilon
       b = false;
   end
   Kold = Knew;
   c(t) = c(t)+1;
 end
 for m = 1:s
     v4(:,:,m,t+1) = v4(:,:,m,t) + h*C(Knew,t+ts1-1,h,s,n,ButcherTableaux);
 end
 waitbar(t/ts,f);
end
close(f);

for t = 1:ts
    y4(:,:,t) = cay(v4(:,:,1,t),n)*y0;
end
```

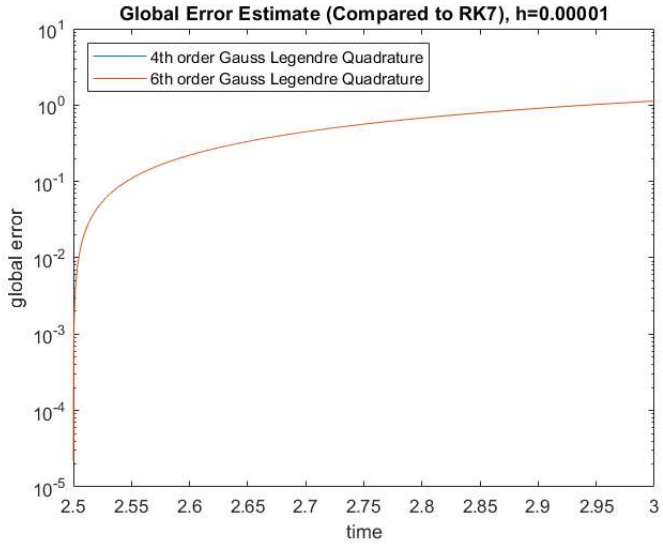
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 7-th Order non Cayley Approximation for comparison %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RK = RK7;
YK = zeros(n,n,length(RK));
RKx = zeros(n,n,ts);
L = length(RK);
RKx(:,:,1) = y0;
for t = 1:ts-1
    for i = 1:L
        YK(:,:,i) = RKx(:,:,t)+h*D(YK,t,i,h,n,RK);
    end
    RKx(:,:,t+1) = RKx(:,:,t)+h*E(YK,t,h,n,RK);
end
```

```
time = zeros(1,ts);
for t = 1:ts
    time(t) = t*h +2.5;
end
```

```
%Distance from manifold
distMan4 = zeros(ts,1);
distMan6 = zeros(ts,1);
distMan7 = zeros(ts,1);
for t = 1:ts
    distMan4(t) = abs(1 - norm(y4(:,:,t)));
    distMan6(t) = abs(1 - norm(y6(:,:,t)));
    distMan7(t) = abs(1 - norm(RKx(:,:,t)));
end
semilogy(time,distMan4,time,distMan6,time,distMan7)
title('Distance to Manifold, h=0.00001')
xlabel('time')
ylabel('distance to manifold')
legend({'4th order Gauss Legendre Quadrature','6th order Gauss Legendre Quadrature','RK7'},'location','east')
saveas(gcf,'D:\Documents\Modelling Project\distance2.jpg')
```




```
%Error when compared to RK7 solution
error4 = zeros(1,ts);
error6 = zeros(1,ts);
for t = 1:ts
    error4(t) = norm(RKx(:,:,t)-y4(:,:,t));
end
for t = 1:ts
    error6(t) = norm(RKx(:,:,t)-y6(:,:,t));
end

semilogy(time,error4,time,error6)
title('Global Error Estimate (Compared to RK7), h=0.00001')
xlabel('time')
ylabel('global error')
legend({'4th order Gauss Legendre Quadrature','6th order Gauss Legendre Quadrature'},'location','northwest')
saveas(gcf,'D:\Documents\Modelling Project\error2.jpg')
```




```
%%%%%%%%%%%%%%%%%%%%
% Special Functions %
%%%%%%%%%%%%%%%%%%%%
```

```
function v = G(K,t,h,s,n,Butcher)
% Lie RK intermediate steps for timestep t
% input (Matrix (nXn), timestep, timestep size, number of RK stages, matrix size, Butcher Matrix)
v = zeros(n,n,s);
L = length(Butcher);
for i = 1:s
    for k = 1:s
        A = B(t+Butcher(k,L)*h,h,n);
        v(:,:,i) = v(:,:,i) + Butcher(i,k)*(A-(1/2)*(K(:,:,k)*A-A*K(:,:,k))-(1/4)*(K(:,:,k)*A*K(:,:,k)));
    end
end
end

function a = B(t,h,n)%populate A in SO(n)
% input(timestep,timestep size, matrix size)
a=zeros(n);
%populate the matrix A
for l = 1:(n-1)
    for j = (l+1):n
        a(l,j) = (-1)^(l+j)*(l)/(j+1)*(t*h)^(j-l);
    end
end
for l = 1:(n-1)
    for j = (l+1):n
        a(j,l) = (-1)*a(l,j);
    end
end
end

function y = D(yold,t,i,h,n,Butcher)
%Regular RK intermediate step
y = zeros(n);
L = length(Butcher(:,1));
for k = 1:i
    y = y+Butcher(i,k)*B(t+Butcher(k,L)*h,h,n)*yold(:,:,k);
end
end

function x = E(Y,t,h,n,Butcher)
%regular RK final
x = zeros(n);
s = length(Butcher(:,1))-1;
L = length(Butcher(:,1));
for k = 1:s
    x = x + Butcher(L,k)*B(t+Butcher(k,L)*h,h,n)*Y(:,:,k);
end
end

function v = C(K,t,h,s,n,Butcher)
%Lie RK for timestep t
% input (Matrix (nXn), timestep, timestep size, number of RK stages, matrix size, Butcher Matrix)
v = zeros(n);
```

```
L = length(Butcher(:,1));
for k = 1:s
    A = B(t+Butcher(k,L)*h,h,n);
    v(:,:) = v(:,:) + Butcher(L,k)*(A-(1/2)*(K(:,:,k)*A-A*K(:,:,k))-(1/4)*(K(:,:,k)*A*K(:,:,k)));
end
end

function c = cay(v,n)
%Cayley transform
c = (eye(n)-(1/2)*v)\(eye(n)+(1/2)*v);
end
```