Control of Nonlinear Spacecraft Attitude Motion

August 24, 2020

1 Stability of Nonlinear Dynamic Systems.

Lyapunov methods can be applied to show stability of a non necessarily linear system of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$. The first method is given by linearization of the system about an equillibrium point.

An equillibrium point of the system is a state $\mathbf{x}_e$ such that $\mathbf{f}(\mathbf{x}_e) = 0$.

Definition 1. *An equilibrium point is said to be Lyapunov stable (or simply stable) if for every $\epsilon > 0$ there exists $\delta > 0$ such that for all $\mathbf{x}$ solution of the system, if $\boldsymbol{x}(0) \in B_\delta(\boldsymbol{x}_e)$ then $\boldsymbol{x}(t) \in B_\epsilon(\boldsymbol{x}_e)$.*

In the above definition $B_\delta(\mathbf{x})$ denotes a ball in the metric space of states centered at $\mathbf{x}$ with radius δ. Rougly speaking, Lyapunov stability means that: initial conditions near the equillibrium implies solution near the equillibrium. An equillibrium point is said to be asymptotically stable if: solutions starting near the equillibrium coverge to the equillibrium, i.e:

Definition 2. *An equilibrium point is said to be asymptotically stable if it is stable and if there exists $\delta > 0$ such that for all $\mathbf{x}$ solution of the system,*

$$\textit{if } \mathbf{x}(0) \in B_\delta(\mathbf{x}_e) \textit{ then } \lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_e$$

Let DF denotes the Jacobian matrix of $\mathbf{f}(\mathbf{x})$ evaluated at an equillibrium $\mathbf{x}_e$. Lyapunov's linearization method gives the following stability criteria

1. The equillibrium point is asymptotically stable if the linearized system is strictly stable, with all eigenvalues of DF strictly in the left-side plane
2. The equillibrium point is unstable if the linearized system is strictly unstable, with at least one eigenvalue strictly on the right-side plane.
3. Nothing can be concluded if the linearized system if the linearized system has eigenvalues lying in the imaginary axis (i.e having real part equal to 0)

Remark: Lyapunov's linearization method gives only local stability. It provides a powerful approach to help qualify the stability of a system if a control (or estimation) scheme is designed to remain within a linear region, but does not give a thorough understanding of the nonlinear system in many cases.

Lyapunov's direct method (second method) gives a global stability condition for the general nonlinear system. This concept is related to the energy of a system, which is a scalar function.

Let $\mathbf{x}_e$ be an equillibrium point and let B a neighborhood of the equillibrium, then a function $V : B \to \mathbb{R}$ is called a Lyapunov function if it has continuous derivatives in B and if it satisfies the following conditions: 1. $V(\mathbf{x}_e) = 0$ 2. $V(\mathbf{x}) > 0$ and $V\dot{(}\mathbf{x}) \leq 0$ for all $\mathbf{x} \in B - \{\mathbf{x}_e\}$.

Theorem 1. *The equillibrium point is stable if there exists a Lyapunov function. Furthermore, if the inequality for the derivative is strict, i. e $\dot{V(\mathbf{x})} < 0$ for all $\mathbf{x} \in B - \{\mathbf{x}_e\}$ then the equillibrium point is asymptotically stable.*

Theorem 2 (Asymptotic stability)**.** *Assume $V(x)$ isa Lyapunov function about an equillibrium point of a nonlinear system. Then this equillibrium is asymptotically stable if $\dot{V(x)} < 0$ for all $\mathbf{x}$ in a neighborhood of the equillibrium, except obviously at the equillibrium. Sometimes $\dot{V}$ is said to be definite negative*

Example: Consider the following spring-mass-damper system with nonlinear spring and damper components:

$$m\ddot{x} + c\dot{x}|\dot{x}| + k_1 x + k_2 x^3 = 0$$

where m, c, k_1, k_2 are poitive constants. The system can be represented in first order form by defining the state vector to be: $\mathbf{x} = (x, \dot{x})^T$, then

$$\begin{aligned}\dot{x}_1 =& x_2\\ \dot{x_2} =& -\frac{k_1}{m}x_1 - \frac{k_2}{m}x_1^3 - \frac{c}{m}x_2|x_2|\end{aligned}$$

Linearization of this system about the equillibrium $(0,0)^T$ produces a matrix whose eigenvalues are pure imaginary, therefore Lyapunov's first method is inconclusive. Now, we can define a Lyapunov function for this system by:

$$V(\mathbf{x}) = \frac{1}{2}mx_2^2 + \int_0^{x_1}(k_1x + k_2x^3)dx = \frac{1}{2}mx_2^2 + \frac{1}{2}k_1x_1^2 + \frac{1}{4}k_2x_1^4$$

Then this function satisfies that:

$$V(x_1, x_2) > 0 \text{ for all } (x_1, x_2) \in \mathbb{R}^2 - \{\mathbf{0}\},$$

$$\dot{V}(x_1, x_2) = m\dot{x_2}x_2 + (k_1x_1 + k_2x_1^3)x_2 = -c|x_2|^3$$

then $\dot{V}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \in \mathbb{R}^2 - \{\mathbf{0}\}$. Note that in this case we only obtained local stability, but this equillibrium is in fact asymptotically stable. To see this we can use the following theorem, due to J.P. LaSalle.

first recall that a set G is invariant for the dynamic system if every integral curve of the system starting at some point in G remains in G for all future time.

Theorem 3. *Assume $\dot{V}(\mathbf{x}) =\leq 0$ for all $\mathbf{x} \in \mathbb{R}^n$ and that $|V(\mathbf{x})| \to +\infty$ as $||\mathbf{x}|| \to +\infty$ and let $\Omega = \{\mathbf{x} \in \mathbb{R}^n\ :\ \dot{V}(\mathbf{x}) = 0\}$ and M the largest invariant set in Ω. Then all solutions globally asymptotically converge to M as time approaches to infinity.*

In the example of the mass spring system, we have $\dot{V}(x_1, x_2) = -c|x_2|^3$, then $\Omega = \mathbb{R} \times 0$ and we would like to show that M in this case consists of just one point $(0,0)$ which is the equillibrium point. In fact, if $\mathbf{x} = (x_1(t), x_2(t))$ is a solution starting at some point in Ω, then $\dot{x}_1(t) = x_2 = 0$ so x_1 is constant, and this constant has to be 0, otherwise by the differential equation we would have that $\dot{x_2} \neq 0$, but $x_2 = 0$, then $M = \{(0,0)\}$ and by Lasalle theorem all solutions globally asymptotically converge to $(0,0)$.

Another usefull result for asymptotic stability is the following corollary of Lasalle's Theorem:

Corollary 1 (See [2]). *Assume there exists a Lyapunov function $V(\mathbf{x})$ of a dynamical system about an equillibrium point. Let $\Omega = \{\mathbf{x}\ :\dot{V}(\mathbf{x}) = 0\}$. Let k and integer such that:*

$$\frac{d^i}{dt^i}V(\mathbf{x}) = 0 \quad \forall \mathbf{x} \in \Omega, \;\; i = 1, 2, \ldots, k-1$$

and the k-th derivative is negative definite on the set Ω, i.e $\frac{d^k}{dt^k}V(\mathbf{x}) < 0$ for all $\mathbf{x} \in \Omega$. Then the equillibirum is asymptotically stable if k is an odd number.

1.0.1 Concept Check 2 - Lyapunov Functions

2. Consider the modified dynamical system $\ddot{x} + kx^3 = 0$. Use the Lyapunov function

   $$V(x, \dot{x}) = \frac{\dot{x}^2}{2} + \frac{k}{4}x^4$$

   to study the stability about the equillibrium $x_e = 0$. What is $\dot{V}$ in this case?

   Ans:
   $$\dot{V}(x, \dot{x}) = \dot{x}\ddot{x} + kx^3\dot{x} = \dot{x}(-kx^3) + kx^3\dot{x} = 0 \quad \text{for all} x, \dot{x}.$$

3. What is the definitness of $\dot{V}$ computed above?

   Ans: It is negative semi-definite

4. What can be said about the stability of this equillibrium?

   Ans: This equillibrium is Lyapunov stable as it has a Lyapunov function about it.

5. In the above dynamical system, what is a damping is added to it, let's say $\ddot{x} + c\dot{x} + kx^3 = 0$ where $c > 0$. what is $\dot{V}$ in this case?

   Ans:
   $$\dot{V}(x, \dot{x}) = \dot{x}\ddot{x} + kx^3\dot{x} = -c\dot{x}^2$$

For the rest of the questionaire consider this modified dynamical system and analize the stability of the equillibrium $x_e = 0$

6. Whit this damping, what is the definitness of $\dot{V}$?

   Ans: this $\dot{V}$ is still negaive semi-definite as $\dot{V} \leq 0$, i.e the inequality is not strict: for states of the form $(x, 0)$ $\dot{V}(x, 0) = 0$ no matter the value of x.

7. With this damping, does this Lyapunov function allow us to prove asymptotically stability?

   Ans: No, since $\dot{V}$ is only negative semi-definite. We have to use Lasalle's corollary to prove that the stability is asymptotic.

8. To study if the system is asymptotic, what is the set where $\dot{V}$ is equal to zero?

   Ans: The set $\Omega = \{(x, 0) : \;\; x \in \mathbb{R}\}$, i.e the set of all states with $\dot{x} = 0$.

9. What is $\ddot{V}$ for this system?

   Ans:
   $$\ddot{V}(x, \dot{x}) = -2c\dot{x}\ddot{x} = -2c\dot{x}(-c\dot{x} - kx^3)$$

Then in the set Ω we have $\ddot{V} = 0$ since $\dot{x} = 0$.

10. In the set Ω what is $\dddot{V}$?
    Ans:
$$\dddot{V}(x, \dot{x}) = 4c^2\dot{x}\ddot{x} + 6ckx^2\dot{x} + 2ckx^3\ddot{x}$$

then $\dddot{V}(x, 0) = 2ckx^3(-c\dot{x} - kx^3) = -2ckx^6$.

Theorem (Lyapunov method for LTI's): An autonomous linear system $\dot{\mathbf{x}} = A\mathbf{x}$ is stable if and only if for any symmetric positive definite matrix Q there exists a symmetric positive definitive matrix P such that
$$A^T P + PA = -Q \text{ Algebraic Lyapunov Equation}$$

1.0.2 Example (8.8 from Junkins-Schaub book)

Let us conider the spacecraft mission scenario where we wish to arrest any rotational motion using thrusters producing an external torque Q (this will be the control mechanism). So the differential equation is:
$$I\dot{\omega} = -[\tilde{\omega}]I\omega + Q$$
with I the inertia tensor and $[\tilde{\omega}]$ the matrix representing the linear transformation $\omega \times -$. Note that the final spacecraft orientation is not a concern, i.e the only relevant dynamics is the one for the spacecraft's angular velocity. Using the Lyapunov function based on the rigid body rotational kinetic energy expressions
$$V(\omega) = \frac{1}{2}\omega^T I\omega$$
whose time derivative is:
$$\dot{V} = \frac{1}{2}\dot{\omega}^T I\omega + \frac{1}{2}\omega^T I\dot{\omega} = \frac{1}{2}\omega^T I\dot{\omega} + \frac{1}{2}\omega^T I\dot{\omega} = \omega^T I\dot{\omega} = \omega^T(-[\tilde{\omega}]I\omega + Q) = \omega^T Q$$

Let us choose the simple control law $Q = -P\omega$ with P some symmetric positive-definite feedback gain matrix. With this choice the Lyapunov derivative is $\dot{V} = -\omega^T P\omega < 0$, which guarantees globally asymptotic stabilizing maneuver. Consider that this control law, alogn with the corresponding stability result, does not require any knowledge of the inertia matrix I itself. This makes this control very robust to inertial modeling errors. For example let $\hat{\omega}$ be the measured body angular velocity vector, and let $P = diag(P_1, P_2, P_3)$ be a diagonal matrix with $P_i > 0$. The rate feedback control becomes
$$Q = -^B(P_1\hat{\omega}_1, P_2\hat{\omega}_2, P_3\hat{\omega}_3)^T$$
the Lyapunov derivative is then expressed as
$$\dot{V} = -\sum_{i=1}^{3} P_i\omega_i\hat{\omega}_i$$

No knowledge is required to implement the rate feedback control law, and the stability argument does not contain inertia expressions. In fact, the measured angular velocity vector components $\hat{\omega}_i$ do not need to be precise as well. As long as the measured angula velocity components have the same sign as the actual ω_i, then the Lyapunov rate is negative-definite and the rate feedback control is globally asymptotically stabilizing.

1.0.3 Concept Check 6 - Rigid Body Elemental Rate Lyapunov Function

1. Consider the controlled rotational motion of a rigid body. The goal is to drive $\omega = \omega_{B/N}$ to zero. Use the function $V(\omega) = \frac{1}{2}\omega^T\omega$ to derive a detumble control law where the derivative of V becomes $\dot{V} = -p\omega^T\omega$ for some parameter tune $p > 0$.
   Ans: If we compute $\dot{V}$ and use the equation of motion $I\dot{\omega} = -[\tilde{\omega}]I\omega + \mathbf{u}$ where $\mathbf{u}$ is the control torque, we get:

$$\dot{V}(\omega) = \omega^T\dot{\omega} = \omega^T(-I^{-1}[\tilde{\omega}]I\omega + I^{-1}\mathbf{u})$$

   A suitable control law is:

$$\mathbf{u} = [\tilde{\omega}]I\omega - pI\omega$$

   thus,

$$\dot{V} = \omega^T\dot{\omega} = \omega^T(-I^{-1}[\tilde{\omega}]I\omega + I^{-1}([\tilde{\omega}]I\omega - pI\omega)) = -\omega^T\omega$$

Another approach is to treat the rate $\dot{\omega}$ as a steering control to control the angular velocity ω. In this case, we can take the following control law:

$$\mathbf{u} = \dot{\omega} = p\omega$$

with p a positive tune parameter. therefore we have the rate $\dot{V}(\omega) = \omega^T\dot{\omega} = -p\omega^T\omega$ which also implies that the system is asymptotically stable about $\omega = \mathbf{0}$. This control is a very robust one, since it does not depend on the inertia tensor of the spacecraft (robustness to the mass distribution uncertainties). Also it is a linear feedback control.

3. Consider the angular velocity vector tracking problem where the control goal is $\omega_{B/N} \to \omega_{R/N}$, i.e, to drive the angular velocity of the body relative to some innertial frame N to some refence angular velocity. The control solution is:

$$\mathbf{u} = -P\delta\omega + [\tilde{\omega}_{B/N}]I\omega_{B/N} + I(\dot{\omega}_{R/N} - [\tilde{\omega}_{B/N}]\omega_{R/N})$$

   Use the Lyapunov function

$$V(\delta\omega) = \frac{1}{2}\delta\omega^T I\delta\omega, \quad \delta\omega := \omega_{B/N} - \omega_{R/N}$$

   what is the corresponding Lyapunove derivative expression?

Ans: First to take the time derivative of V we'll use the transport theorem:

$$\dot{V} = \frac{1}{2}\frac{{}^B d}{dt}(\delta\omega^T I\delta\omega) = \frac{{}^B d\delta\omega^T}{dt}I\delta\omega + \frac{1}{2}\delta\omega^T I\frac{{}^B d\delta\omega}{dt} = \delta\omega^T I\frac{{}^B d\delta\omega}{dt} = \delta\omega^T I(\delta\dot{\omega} + \omega \times \omega_{R/N})$$

where in the last equality we've used that

$$\delta\dot{\omega} = \frac{{}^B d\delta\omega}{dt} + \omega \times \delta\omega = \frac{{}^B d\delta\omega}{dt} - \omega \times \omega_{R/N}$$

now using the differential equation of the system:

$$\dot{V}(\delta\omega) = \delta\omega^T I(\dot{\omega} - \dot{\omega}_{R/N} + \omega \times \omega_{R/N}) = \delta\omega^T(-[\tilde{\omega}]I\omega + \mathbf{u} - I\dot{\omega}_{R/N} + I(\omega \times \omega_{R/N})) = -\delta\omega^T P\delta\omega$$

where in the last step we replaced the chosen law control. This control law is asymptotically stabilizing as $\dot{V}$ is negative definite.

1.1 Position-Based Lyapunov Functions

This section provides elemental position-based Lyapunov functions that allow us to control de position of a body. Analogous to the velocity-based (kinetic enrgy-like) Lyapunov functions given above, the state space of interest here is simply $\mathbf{q}$ (this is a generalized position variable, it could be Euler angles, CRP's, quaternions, MRP's. Now $\dot{\mathbf{q}}$ is treated as the control variable. This type of control law is called steering law. This control law will determine a desired $\dot{\mathbf{q}}(t)$ trajectory that must be followed to stabilize the system about a desired position. A standard V function is of the type of potential energy:

$$V(\mathbf{q}) = \frac{1}{2}\mathbf{q}^T K \mathbf{q}$$

where the position vector $\mathbf{q}$ is assumed to be measured relative to the target state. The symmetric matrix K must be positive definite to guarantee that V is in fact a Lyapunov function.. When using these Lyapunov functions to create feedback control laws, the matrix K assumes the role of a position feedback gain matrix and also has the perfect analog to a system of linear springs. The time derivative of V is:

$$\dot{V}(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^T K \mathbf{q}.$$

To create a steering law for $\mathbf{q}$, one can define the Lyapunov function to be $V(\mathbf{q}) = \frac{1}{2}\mathbf{q}^T\mathbf{q}$, so the time derivative is: $\dot{V}(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{q}$. Then defining the steering $\dot{\mathbf{q}}$ to be:

$$\dot{\mathbf{q}} = -K\mathbf{q} \quad \text{steering control law}$$

with K symmetric positive definite matrix gain. This steering law would bring $\mathbf{q}$ asymptotically to zero. Note that in all steering laws(controlling $\mathbf{q}$ and treating $\dot{\mathbf{q}}$ as a control variable), the system dynamics do not appear.

In rigid body dynamics, the kinetic energy is typically not expressed in terms of position coordinate derivatives (i.e., $\dot{\mathbf{q}}_i$) but rather in terms of the body angular velocity vector ω. Therefore, velocity expressions in $\dot{V}$ for rigid bodies will need to be written in terms of ω, too. There is a multitude of attitude coordinates available to describe rigid body orientations. Convenient Lyapunov fucntions for a selected subset of the attitude coordinates are present in the following. A popular set of attitude coordinates is the Euler angles `, where this vector θ could be the 3-2-1 yaw, pitch roll angles, the 3-1-2 Euler angles or any other set of sequential rotational coordinates. Assume θ measures the current ridig body attitude relative to some target orientation, then the candidate Lypaunov function

$$V(\theta) = \frac{1}{2}\boldsymbol{\theta}^T K \boldsymbol{\theta}$$

provides a positive definite measure of the attitude error. Using the kinematic differential equation of $\boldsymbol{\theta}$, i.e $\dot{\boldsymbol{\theta}} = B(\boldsymbol{\theta})\omega$, with $B(\boldsymbol{\theta})$ given by:

$$B(\boldsymbol{\theta}) = \frac{1}{\cos\theta}\begin{bmatrix} 0 & \sin\phi & \cos\phi \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ \cos\theta & \sin\phi\sin\theta & \cos\phi\sin\theta \end{bmatrix} \quad \text{for the 3-2-1 Euler angles, the vector } \boldsymbol{\theta} = (\Psi, \theta, \phi)^T$$

$$B(\boldsymbol{\theta}) = \frac{1}{\sin\theta_2}\begin{bmatrix} \sin\theta_3 & \cos\theta_3 & 0 \\ \cos\theta_3\sin\theta_2 & -\sin\theta_3\sin\theta_2 & 0 \\ -\sin\theta_3\cos\theta_2 & -\cos\theta_3\cos\theta_2 & \sin\theta_2 \end{bmatrix} \quad \text{for the 3-1-3 Euler angles, the vector } \boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^T$$

The time derivative of V is

$$\dot{V}(\boldsymbol{\theta}) = \boldsymbol{\omega}^T B(\boldsymbol{\theta})^T K\boldsymbol{\theta}$$

In this case, this Lyapunov function will not lead to a linear feedback control law in terms of the orientation coordinates $\boldsymbol{\theta}$ because of the non linear nature of the $B(\boldsymbol{\theta})$ matrix. Also if the target attitude is nonstationary, but defined thorugh the reference body angular velocity $\boldsymbol{\omega}_r := \omega_{R/N}$, then the relative attitude error rate $\dot{\boldsymbol{\theta}}$ is given by

$$\dot{\boldsymbol{\theta}} = B(\boldsymbol{\theta})\delta\boldsymbol{\omega}$$

where $\boldsymbol{\theta}$ is the attitude vector from the reference frame to the body frame. The corresponding Lyapunov function time derivative is:

$$\dot{V}(\boldsymbol{\theta}) = \delta\boldsymbol{\omega}^T B(\boldsymbol{\theta})^T K\boldsymbol{\theta}.$$

For the remainder of this section, unless noted otherwise, it will always be assumed that the attitude vector is measured relative to the target state and not relative to some inertial frame. Therefore, no distinction will be made if this reference state is stationary or not because the corresponding angular velocities $\boldsymbol{\omega}, \delta\boldsymbol{\omega}$ can be interchanged trivially as shown above in the two expressions for $\dot{V}$.

1.1.1 Using other attitude coordinates.

Gibbs or CRP (Classical Rodrigues Parameter) vector $\boldsymbol{q}$. The Lyapunov function is expressed as:

$$V(\boldsymbol{q}) = \boldsymbol{q}^T K\boldsymbol{q} \quad K > 0, \;\; K^T = K$$

with time derivative:

$$\dot{V} = 2\dot{\boldsymbol{q}}^T K\boldsymbol{q} = 2(B(\boldsymbol{q})\boldsymbol{\omega})^T K\boldsymbol{q} = \boldsymbol{\omega}^T(\mathbb{I} - [\tilde{\boldsymbol{q}}] + \boldsymbol{q}\boldsymbol{q}^T))K\boldsymbol{q}$$

where we've used that $\dot{\boldsymbol{q}} = B(\boldsymbol{q})\boldsymbol{\omega}$ with $B(\boldsymbol{q}) = \frac{1}{2}(\mathbb{I} + [\tilde{\boldsymbol{q}}] + \boldsymbol{q}\boldsymbol{q}^T)$.

The case where K is a positve scalar (not a matrix) we have that

$$\dot{V} = \boldsymbol{\omega}^T(K(1+q^2)\boldsymbol{q})$$

which again will lead to a nonlinear feedback control law in $\boldsymbol{q}$. This nonlinear scaling term $(1+q^2)$ can be avoided by choosing a different Lyapunov function. Instead of using the potential energy like function we take the logarithmic function

$$V(\boldsymbol{q}) = K\ln(1+q^2)$$

then

$$V(\dot{\boldsymbol{q}}) = \frac{2}{1+q^2}\dot{\boldsymbol{q}}^T\boldsymbol{q} = \boldsymbol{\omega}^T\boldsymbol{q}$$

then the steering law

$$\boldsymbol{\omega} = -K\boldsymbol{q}, \;\; K > 0, \;\; K^T = K$$

would asymptotically drive the CRP attitude error $\boldsymbol{q}$ to zero.

MRP's attitude coordinates. Attitude Lyapunov functions in terms of MRPs are generated in a very simi- lar way. For a symmetric positive definite matrix K the Lyapunov function

$$V(\boldsymbol{\sigma}) = 2\boldsymbol{\sigma}^T K\boldsymbol{\sigma}$$

whose time derivative is:

$$\dot{V} = 4\dot{\boldsymbol{\sigma}}^T K\boldsymbol{\sigma} = 4(B(\boldsymbol{\sigma})\boldsymbol{\omega})^T K\boldsymbol{\sigma} = \boldsymbol{\omega}^T((1-|\sigma|^2)I - 2[\tilde{\boldsymbol{\sigma}}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T)K\boldsymbol{\sigma}$$

where we used that:

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}((1-|\sigma|^2)I + 2[\tilde{\boldsymbol{\sigma}}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T)\boldsymbol{\omega} = \frac{1}{4}B(\boldsymbol{\sigma})\boldsymbol{\omega}$$

If the gain K is a scalar we will have $\dot{V} = \boldsymbol{\omega}^T K(1+\sigma^2)\boldsymbol{\sigma}$ which will also lead to a nonlinear feedback control law in terms of $\boldsymbol{\sigma}$. To obtain a simpler $\dot{V}$ we agin use the logarithmic Lyapunov function

$$V(\boldsymbol{\sigma}) = 2\ln(1+\sigma^2)$$

Note that by switching the MRPs to their alternate shadow set whenever $\sigma^2 > 1$, this Lyapunov function can describe any orientation error without encountering a singularity. Further, it provides a bounded measure of the atti- tude error. This is convenient, because two orientations can differ only by a finite rotation. Therefore, having a bounded Lyapunov function describing the attitude error inherently reflects this fact and will have some important conse- quences when designing attitude feedback control laws in terms of the MRPs.

Then after substituting the kinematic differential equation for the MRP's we get

$$\dot{V} = \boldsymbol{\omega}^T\boldsymbol{\sigma}$$

then the steering control law is

$$\boldsymbol{\omega} = -K\boldsymbol{\sigma},\ \ K > 0,\ \ K^T = K$$

which is asymptotically stabilizing.

1.2 Nonlinear Feedback Control laws (combining Lyapunov position and velocity-based functions)

1.2.1 Unconstrained Control Law

The modified Rodrigues parameter vector $\boldsymbol{\sigma}$ is very well suited for describ- ing attitude errors in a feedback control law setting. Particularly when very large attitude errors are present, the MRPs are extremely attractive. By switching between the original and shadow MRP set, they are able to describe any arbitrary orientation without encountering singularities by using only three parameters instead of four as do the Euler parameters. Adopting the switching surface $\sigma^2 = 1$ bounds the attitude error vector norm within the unit ball $|\boldsymbol{\sigma}| \leq 1$ where the $\boldsymbol{\sigma}$ motion is approximately linear with respect to $\boldsymbol{\omega}$. This bounded attitude error property is very useful because it will make designing the attitude feedback gain much easier. Choosing the $\sigma^2 = 1$ switching surface also has a big benefit when trying to bring a tumbling rigid body to rest. Conventional attitude parameters such as the Euler angles have no explicit means of determining the shortest rotational distance back to the reference attitude. Let I be the rigid body inertia matrix, $\boldsymbol{\omega}(t)$ the body angular velocity vector and $\boldsymbol{u}(t)$ be some unconstrained external torque vector. The vector $\boldsymbol{L}$ is som known external torquea acting on the body. Euler's rotational equations of motion for a rigid body are given by

$$I\dot{\boldsymbol{\omega}} = -[\tilde{\boldsymbol{\omega}}]I\boldsymbol{\omega} + \boldsymbol{u} + L$$

The vector $\boldsymbol{\sigma}(t)$ measures the attitude error of this rigid body relative to some reference trajectory that is itself defined through the reference angular velocity vector $\boldsymbol{\omega}_r(t)$. The error $\delta\boldsymbol{\omega}(t)$ in angular velocities is defined as

$$\delta\boldsymbol{\omega}(t) = \boldsymbol{\omega}(t) - \boldsymbol{\omega}_r(t)$$

The MRP rate vector $\dot{\boldsymbol{\sigma}}$ and the body angular velocity error vector $\delta\boldsymbol{\omega}$ are related through

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}((1-\sigma^2)\mathbb{I} + 2[\tilde{\boldsymbol{\sigma}}] + 2\boldsymbol{\sigma}\boldsymbol{\sigma}^T)\delta\boldsymbol{\omega}$$

The chosen Lyapunov function for this system is:

$$V(\delta\boldsymbol{\omega},\boldsymbol{\sigma}) = \frac{1}{2}\delta\boldsymbol{\omega}^T I\delta\boldsymbol{\omega} + 2K\ln(1+\boldsymbol{\sigma}^T\boldsymbol{\sigma})$$

which is a positive definite function, radially unbounded measure of the rigid body state error relative to the reference trajectory. The parameter K is a positive scalar attitude feedback gain. The time derivative of V is:

$$\dot{V} = \delta\boldsymbol{\omega}^T(I\frac{{}^B d}{dt}(\delta\boldsymbol{\omega}) + K\boldsymbol{\sigma})$$

To guarantee stability, we force $\dot{V}$ to be negative semidefinite by setting it equal to

$$\dot{V} = -\delta\boldsymbol{\omega}^T P\delta\boldsymbol{\omega}$$

where P is the positive definite angular velocity feedback gain matrix. This leads to the following stability constraint:

$$I\frac{{}^B d}{dt}(\delta\boldsymbol{\omega}) + P\delta\boldsymbol{\omega} + K\boldsymbol{\sigma} = 0$$

which will give us a control law after replacing the derivative in the B frame of $\delta\boldsymbol{\omega}$ and using the rigid body dynamics:

$$I(\dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\omega}}_r + \boldsymbol{\omega}\times\boldsymbol{\omega}) + P\delta\boldsymbol{\omega} + K\boldsymbol{\sigma} = -[\tilde{\boldsymbol{\omega}}]I\boldsymbol{\omega} + \boldsymbol{u} + L - I\dot{\boldsymbol{\omega}}_r + I\boldsymbol{\omega}\times\boldsymbol{\omega} + P\delta\boldsymbol{\omega} + K\boldsymbol{\sigma} = 0$$

then

$$\boldsymbol{u} = -K\boldsymbol{\sigma} - P\delta\boldsymbol{\omega} + I(\dot{\boldsymbol{\omega}}_r - [\tilde{\boldsymbol{\omega}}]\boldsymbol{\omega}_r) + [\tilde{\boldsymbol{\omega}}]I\boldsymbol{\omega} - L$$

Since the Lyapunov function is radially unbounded and the MRP's with the $\sigma^2 = 1$ switching surface are nonsingular, the feedbac control law $\boldsymbol{u}$ is qguaranteed to be globally stabilizing. The cross-coupling term $\boldsymbol{\omega} \times \boldsymbol{\omega}_r$ is sometimes neglected in this feedback control law. For typical spacecraft maneuvers, both $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_r$ are relatively small, and this cross product is not scaled by any feedback gains. Therefore, this product usually has a negligible impact on the control law performance. However, to rigorously guarantee stability or to include the possibility of large v and vr vectors, this cross-coupling term must be included.

1.2.2 Concept Check 1 - General 3-Axis Attitude Control

4. Write software code to simulate the general spacecraft motion where a control torque $\boldsymbol{u}$ is being applied. The principal inertias are $I_1 = 100kgm^2$, $I_2 = 75kgm^2$ and $I_3 = 80kgm^2$. The initial states are

$$\boldsymbol{\sigma}_{B/N}(t_0) = (0.1, 0.2, -0.1), \ \ \boldsymbol{\omega}_{B/N}(t_0) = (30, 10, -20) \deg /s$$

Use the globally asymptotically stabilizing control solution

$$\boldsymbol{u} = -K\boldsymbol{\sigma}_{B/R} - P\boldsymbol{\omega}_{B/R} + I(\dot{\boldsymbol{\omega}}_{R/N} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{R/N}) + [\tilde{\boldsymbol{\omega}}_{B/N}]I\boldsymbol{\omega}_{B/N} - L$$

with gains $K = 5Nm$ and $p = 10Nms$ where $P = p\mathbb{I}_{3\times3}$. For the following case assume $L = 0$.

a. Simulate a regulator case (i.e. reference frame is equal to inertial frame) for 120 seconds where $\boldsymbol{\sigma}_{R/N}$ is the zero orientation. Finally compute the euclidean norm of $\boldsymbol{\sigma}(30)$

b. Simulate an attitude tracking case for 120 seconds where $\boldsymbol{\sigma}_{R/N} = (0.2\sin(ft), 0.3\cos(ft), -0.3\sin(ft))$ with $f = 0.05 rad/sec$. Compute the tracking error norm $|\boldsymbol{\sigma}_{B/R}|$ at 40 seconds into the simulation.

[1]:
```python
import numpy as np
import scipy
import math
from scipy.linalg import *
import scipy.integrate as integrate
from DCMmatrix import *
from Euler_Integrator import *
import matplotlib.pyplot as plt
```

[2]:
```python
def sigma_b_r(s_BN, s_RN, tracking):
    """
    This function computes the orientation of the body relative to the reference frame R:

    Parameters
    ----------
    s_BN: 1-d np.array: orientation sigma_B/N (orientation of body frame B relative to inertial N)
    s_RN: 1-d np.array: orientation sigma_R/N (orientation of reference frame R relative to inertial N)

    Returns
    -------
    s_BR : the orienatation of the body relative to reference
    """
    BN = MRP_to_DCM(s_BN)
    if tracking:
        RN = MRP_to_DCM(s_RN)
        BR = BN@(RN.T)
        s_BR = DCM_to_MRP(BR)
    else :
        RN = np.eye(3)
        BR = BN
        s_BR = s_BN

    return BR,s_BR



def s_RN(t,f, tracking):
    """
    computes for each t sigma_R/N
    """
    if tracking:
        return np.array([0.2*np.sin(f*t), 0.3*np.cos(f*t), -0.3*np.sin(f*t)])
    else:
        return np.zeros(3)



def s_RN_dot(t,f, tracking):
    """
    computes for each t the time derivative of sigma_R/N
    """
    if tracking:
        return f*np.array([0.2*np.cos(f*t), -0.3*np.sin(f*t), -0.3*np.cos(f*t)])
    else:
        return np.zeros(3)

def Rw_RN(t, f, tracking):
    """
    computes the angular velocity w_R/N in the R frame
    """
    s_rn = s_RN(t, f, tracking)
    s_rndot = s_RN_dot(t, f,  tracking)
    B_s_RNT = (1-norm(s_rn,2)**2)*np.eye(3) - 2*Matrix_tilde(s_rn) + 2*np.outer(s_rn,s_rn)  # B(s_RN)^T
    R_w_RN = 4/((1 + norm(s_rn,2)**2)**2)*B_s_RNT@s_rndot   #angular velocity of R relative to N: w_R/N in the R frame

    return R_w_RN

def w_RN(t, BR, f,tracking):
    """
    computes the angular velocity w_R/N in the B frame
    """

    return BR@Rw_RN(t,f, tracking)

def w_RN_dot(t, dt, BR, f,tracking):
    """
    computes the time derivative of ^Bw_R/N in the B frame using centered␣
 ↪differences
    """

    return (w_RN(t+dt, BR, f, tracking) - w_RN(t-dt, BR, f, tracking))/(2*dt)


def w_BN_dot(t,w, u, L, DeltaL):
    """
    computes the r.hs of the diff equ. for w_b/N_dot as a function of the␣
 ↪control and external torque
    """
    rhs = inv(I)@((-Matrix_tilde(w) @ I) @ w + u + L + DeltaL)
    return rhs

def MRP_dot(t, sigma0, w):
    """
    computes r.h.s f the differential eq for sigma_B/N

    Parameters
    ----------
    t: double. time t
    sigma0: numpy array. initial condition for sigma_B/N
    w: numpy array. tha angular velocity w_B/N at time t

    Returns
    --------
    rhs: numpy array of size= w.shape. The r.h.s of the kinematic differential␣
 ↪equation of the MRP's
    """
    B_sigma = 0.25*((1- norm(sigma0,2)**2)*np.eye(3) +2*Matrix_tilde(sigma0)␣
 ↪+2*np.outer(sigma0,sigma0))
    rhs = B_sigma @ w
    return rhs


def order4(function, tval, x, step, *param):
    """
    this function computes the fourth order approximation of a solution of a␣
 ↪differential equation whose r.h.s is given by function
```

```
    Parameters
    ----------
    function: a function that returns the r.h.s of the differential equation
    tval: double. time at which function is evaluated
    x: numpy array. Initial condition
    step: float. The delta t of the grid in the time interval.
    """

    k1 = function(tval, x, *param)
    k2 = function(tval + 0.5*step, x + 0.5*step*k1, *param)
    k3 = function(tval + 0.5*step, x + 0.5*step*k2, *param)
    k4 = function(tval + step, x + step*k3, *param)

    x = x +step*(k1 + 2*k2 +2*k3 + k4)/6

    return x

def simulation(tspan, sigma_0, w_0, dt,f, I, L, k, P,tracking, control_law, K_I,␣
 ↪DeltaL):

    t0, tf = tspan[0], tspan[1]
    t = np.arange(t0, tf, dt)
    nt = len(t)

    #history matrices
    Sigma = np.zeros((len(sigma_0), nt)) # time history of the attitude variable
    W = np.zeros((len(w_0), nt))  # time history of angular velocity w_B/N
    U = np.zeros((len(sigma_0), nt)) #time history of the control law

    # initial values:
    s_bn = sigma_0  # initialized at time 0
    w_bn = w_0     # initialized at time 0
    s_rn = s_RN(0,f, tracking)
    BR, s_br = sigma_b_r(s_bn, s_rn, tracking)
    w_rn = w_RN(0, BR, f, tracking)
    w_rndot = w_RN_dot(0, dt, BR, f,tracking)
    del_w = w_bn - w_rn
    # control1
    if control_law == 1 :
        u = control1(s_br,del_w, w_bn, w_rn, w_rndot, L)
    # control2
    if control_law == 2 :
        u = control2(s_br, del_w)
    #control3
    if control_law == 3 :
        u = control3(s_br, del_w, w_bn, w_rn, w_rndot)
    # control PID
    if control_law == 'PID':
        del_w0 = del_w
        u = controlPID(s_br, np.array([0,0,0]), del_w, del_w0, w_bn, w_rn,␣
↪w_rndot, L, K_I)



    Sigma[:,0] = s_bn
    W[:,0] = w_0
    U[:,0] = u

    # history matrices for the tracking problem
    Sigma_BR = np.zeros(Sigma.shape)
    W_BR = np.zeros(W.shape)
    Sigma_BR[:,0] = s_br
    W_BR[:,0] = del_w

    # history matrix of state z for the case of PID control
    Z = np.zeros(Sigma.shape)

    for i in range(1,nt):
        w_bn = order4(w_BN_dot, t[i]-1, w_bn, dt, u, L, DeltaL)
        s_bn = order4(MRP_dot, t[i]-1, s_bn, dt, w_bn)
        if norm(s_bn, 2) >= 1:
            s_bn = -s_bn/(norm(s_bn,2)**2)

        s_rn = s_RN(t[i], f, tracking)
        BR, s_br = sigma_b_r(s_bn, s_rn, tracking)
        w_rn = w_RN(t[i], BR, f, tracking)
        w_rndot = w_RN_dot(t[i], dt, BR, f, tracking)
        del_w = w_bn - w_rn

        Sigma[:,i] = s_bn
        W[:, i] = w_bn
        Sigma_BR[:,i] = s_br
        W_BR[:,i] = del_w

        # control1:
        if control_law == 1 :
            u = control1(s_br,del_w, w_bn, w_rn, w_rndot, L)

        # control2:
        if control_law == 2 :
            u = control2(s_br, del_w)

        # control3:
```

```
        if control_law == 3 :
            u = control3(s_br, del_w, w_bn, w_rn, w_rndot)

        # control PID
        if control_law == 'PID':
            s_br_integral = integrate.trapz(Sigma_BR[:,0:i], dx = dt, axis =1)
            u = controlPID(s_br, s_br_integral, del_w, del_w0, w_bn, w_rn,␣
↪w_rndot, L, K_I)
            z = k*s_br_integral +I@(del_w -del_w0)
            Z[:,i] = z

        U[:,i] = u




    return Sigma, W, U, Sigma_BR, W_BR, Z
```

1.2.3 Wrapping the simulation into a function according to the kind of problem

For the Tracking problem, i.e $\sigma_{R/N}$ is not the zero orientation. Therefore we are interested in the error orientation $\boldsymbol{\sigma}_{B/R}$

[3]:
```
def Tracking(tspan, sigma_0, w_0, dt, f, I, L, k, P, control_law, K_I = 0,␣
↪DeltaL = 0) :
    """
    This function simulates and plots the problem of tracking a spacecraft

    Parameters
    ----------
    tspan: tuple --> It contains the initial and final time for the simulation
    sigma_0: numpy array --> the initial condition of the body attitude relative␣
↪to inertia frame N, sigma_B/N
    w_0: numpy array --> initial angular velocity of body relative to inertial␣
↪frame (in the B frame)
    f : float --> the frequency parameter in rad/s for the attitude of some␣
↪reference frame R relative to inertial frame N
    I : 2d numpy array --> inertia tensor usually represented by a matrix in the␣
↪B frame
    L : numpy array --> external torque
    k : float --> feedback gain corresponding to the attitude vector
    P : 2d numpy array --> The feedback gain corresponding to the angular␣
↪velocity
    control_law : int or string --> indicates the control_law to use
    K_I : 2d numpy array --> feedback gain corresponding to the integral term in␣
↪the PID control. Default value is 0
```

```
    Delta_L : numpy array --> unknown external torque. The total torque on the␣
↪spacecraft is expressed as L + DeltaL

    Returns
    -------
    Sigma_BR : 2d numpy array of dim = (3,number of time points) --> history␣
↪matrix of the attitude error sigma_B/R
    W_BR : 2d numpy array of dim = (3,number of time points) --> history matrix␣
↪of the angular velocity error del_w = w_B/R
    Z : 2d numpy array of dim = (3,number of time points) --> history matrix of␣
↪the of the new state z (only for the PID control case)
    """

    tracking = True
    _, _, U, Sigma_BR, W_BR, Z = simulation(tspan,sigma_0, w_0, dt, f, I, L, k,␣
↪P, tracking, control_law, K_I, DeltaL)
    sbr1 = Sigma_BR[0,:]
    sbr2 = Sigma_BR[1,:]
    sbr3 = Sigma_BR[2,:]
    wbr1 = W_BR[0,:]
    wbr2 = W_BR[1,:]
    wbr3 = W_BR[2,:]
    u1 = U[0,:]
    u2 = U[1,:]
    u3 = U[2,:]
    t = np.arange(tspan[0],tspan[1], dt)

    plt.figure(figsize = (20,6))
    plt.subplot(121)
    plt.plot(t,sbr1, label = '$\sigma_1$')
    plt.plot(t,sbr2, label =' $\sigma_2$')
    plt.plot(t, sbr3, label = '$\sigma_3$')
    plt.xlabel('time [s]')
    plt.ylabel('Error attitude vector')
    plt.legend()
    plt.title('Attitude vector $\sigma_{B/R}$')
    plt.subplot(122)
    plt.plot(t,wbr1,label='$\omega_1$')
    plt.plot(t,wbr2, label='$\omega_2$')
    plt.plot(t,wbr3, label ='$\omega_3$')
    plt.xlabel('time [s]')
    plt.ylabel('Angular velocities [rad/s]')
    plt.legend()
    plt.title('Angular velocity vector $\omega_{B/R}$')

    plt.figure(figsize = (12,6))
    plt.plot(t, u1, label ='u1')
    plt.plot(t, u2, label ='u2')
    plt.plot(t, u3, label ='u3')
    plt.xlabel('time [s]')
    plt.ylabel('control vector [N-m]')
    plt.legend()
    plt.title('control law')

    plt.show()
    if control_law == 'PID':
        return Sigma_BR, W_BR, Z
    else :
        return Sigma_BR, W_BR
```

1.2.4 For the regulator problem, i.e, the reference frame R is the same as the inertial frame N.

[4]:

```
def Regulator(tspan, sigma_0, w_0, dt, I, L, k, P, control_law, K_I = 0, DeltaL␣
 ↪= 0) :
    """
    This function simulates and plots the problem of the regulator of a␣
 ↪spacecraft


    Parameters
    ----------
    tspan: tuple --> It contains the initial and final time for the simulation
    sigma_0: numpy array --> the initial condition of the body attitude relative␣
 ↪to inertia frame N, sigma_B/N
    w_0: numpy array --> initial angular velocity of body relative to inertial␣
 ↪frame (in the B frame)
    I : 2d numpy array --> inertia tensor usually represented by a matrix in the␣
 ↪B frame
    L : numpy array --> external torque
    k : float --> feedback gain corresponding to the attitude vector
    P : 2d numpy array --> The feedback gain corresponding to the angular␣
 ↪velocity
    control_law : int or string --> indicates the control_law to use
    K_I : 2d numpy array --> feedback gain corresponding to the integral term in␣
 ↪the PID control. Default value is 0
    Delta_L : numpy array --> unknown external torque. The total torque on the␣
 ↪spacecraft is expressed as L + DeltaL


    Returns
    -------
    Sigma : 2d numpy array of dim = (3,number of time points) --> history matrix␣
 ↪of the attitude  sigma_B/N
```

```
    W : 2d numpy array of dim = (3,number of time points) --> history matrix of␣
↪the angular velocity  w_B/N
    Z : 2d numpy array of dim = (3,number of time points) --> history matrix of␣
↪the of the new state z (only for the PID control case)
    """

    tracking = False
    f = 0
    Sigma, W, U, _, _, Z = simulation(tspan,sigma_0, w_0, dt, f, I, L, k, P,␣
↪tracking, control_law, K_I, DeltaL)
    s1 = Sigma[0,:]
    s2 = Sigma[1,:]
    s3 = Sigma[2,:]
    w1 = W[0,:]
    w2 = W[1,:]
    w3 = W[2,:]
    u1 = U[0,:]
    u2 = U[1,:]
    u3 = U[2,:]
    t = np.arange(tspan[0],tspan[1], dt)

    plt.figure(figsize = (20,6))
    plt.subplot(121)
    plt.plot(t,s1, label = '$\sigma_1$')
    plt.plot(t,s2, label =' $\sigma_2$')
    plt.plot(t, s3, label = '$\sigma_3$')
    plt.xlabel('time [s]')
    plt.ylabel('Attitude')
    plt.legend()
    plt.title('Attitude vector $\sigma_{B/N}$')
    plt.subplot(122)
    plt.plot(t,w1,label='$\omega_1$')
    plt.plot(t,w2, label='$\omega_2$')
    plt.plot(t,w3, label ='$\omega_3$')
    plt.xlabel('time [s]')
    plt.ylabel('Angular velocities [rad/s]')
    plt.legend()
    plt.title('Angular velocity vector $\omega_{B/N}$')

    plt.figure(figsize = (12,6))
    plt.plot(t, u1, label ='u1')
    plt.plot(t, u2, label ='u2')
    plt.plot(t, u3, label ='u3')
    plt.xlabel('time [s]')
    plt.ylabel('control vector [N-m]')
    plt.legend()
    plt.title('control law')
    plt.show()

    if control_law == 'PID':
        return Sigma, W, Z
    else :
        return Sigma, W
```

[5]:

```
def control1(s_br,delw, w_bn, w_rn, w_rndot, L):

    u = -k*s_br - P@delw + I@(w_rndot - np.cross(w_bn, w_rn)) +␣
 ↪(Matrix_tilde(w_bn)@I)@w_bn - L

    return u

def control2(s_br, delw):

    u = -k*s_br - P@delw

    return u

def control3(s_br, delw, w_bn, w_rn, w_rndot):

    u = control1(s_br,delw, w_bn, w_rn, w_rndot, 0)

    return u

def controlPID(s_br, s_br_integral, delw, delw0, w_bn, w_rn, w_rndot, L, K_I) :

    u = -k*s_br -(P + P@K_I@I)@delw - k*(P@K_I)@s_br_integral + (P@K_I@I)@delw0␣
 ↪+ I@(w_rndot - np.cross(w_bn, w_rn)) + (Matrix_tilde(w_bn)@I)@w_bn - L

    return u
```

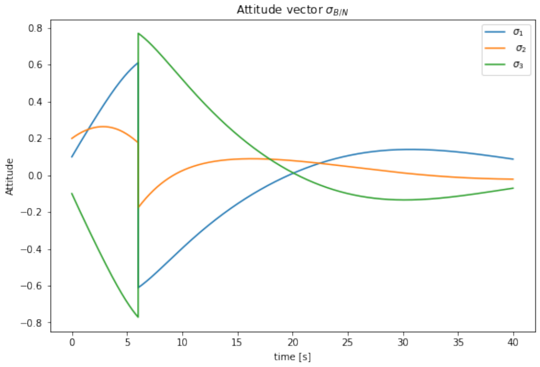
[6]:

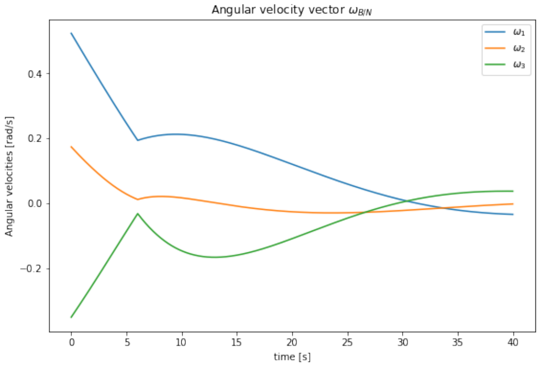
```
I = np.diag([100,75,80])
k = 5
P = 10*np.eye(3)
sigma_0 = np.array([0.1, 0.2, -0.1])
w_0 = np.array([30, 10, -20])  # deg/s
w_0 = (np.pi/180)*w_0
```

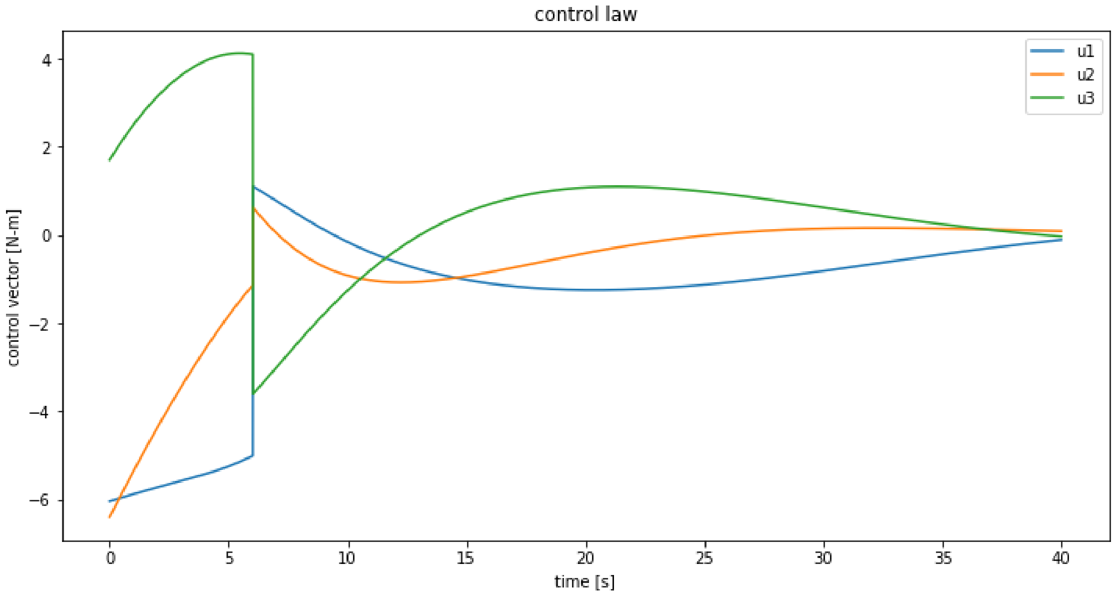
[7]:

```
# for the Regulator problem, compute the norm of the attitude at 30 seconds:

S, W = Regulator((0,40), sigma_0, w_0, 0.01, I, 0, k, P, control_law=1)
```







[8]:

```
print(' The orientation $\sigma$ at 30 seconds and its norm are: ', S[:,3000],␣
  ↪norm(S[:,3000],2))
```

```
 The orientation $\sigma$ at 30 seconds and its norm are:  [ 0.13980378
0.01252705 -0.13411701] 0.19413757280378702
```

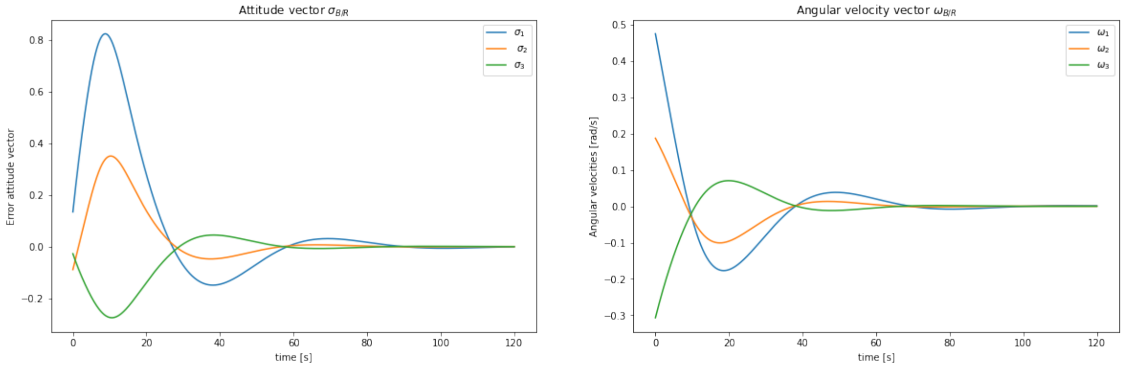
[61]:

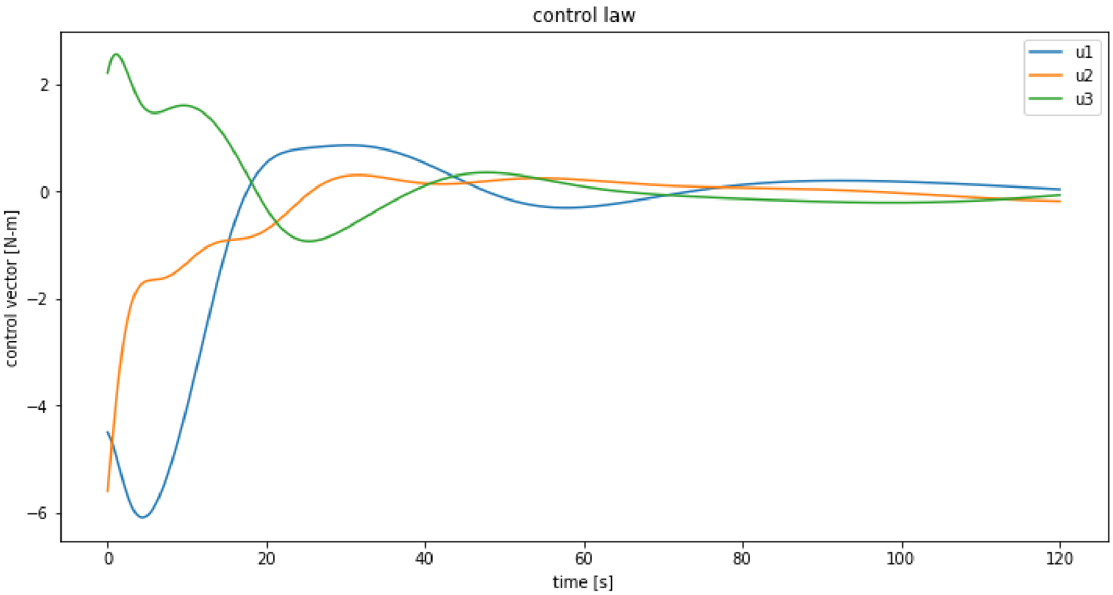
```
# for the tracking problem, compute the norm of the error attitude $\sigma_B/R$␣
  ↪at 40 seconds:

I = np.diag([100,75,80])
k = 5
P = 10*np.eye(3)
sigma_0 = np.array([0.1, 0.2, -0.1])
w_0 = np.array([30, 10, -20])  # deg/s
```

```
w_0 = (np.pi/180)*w_0
```

[62]:
```
S_BR, W_BR = Tracking((0,120), sigma_0, w_0, 0.01, 0.05, I, 0, k, P, control_law␣
↪= 1)
```

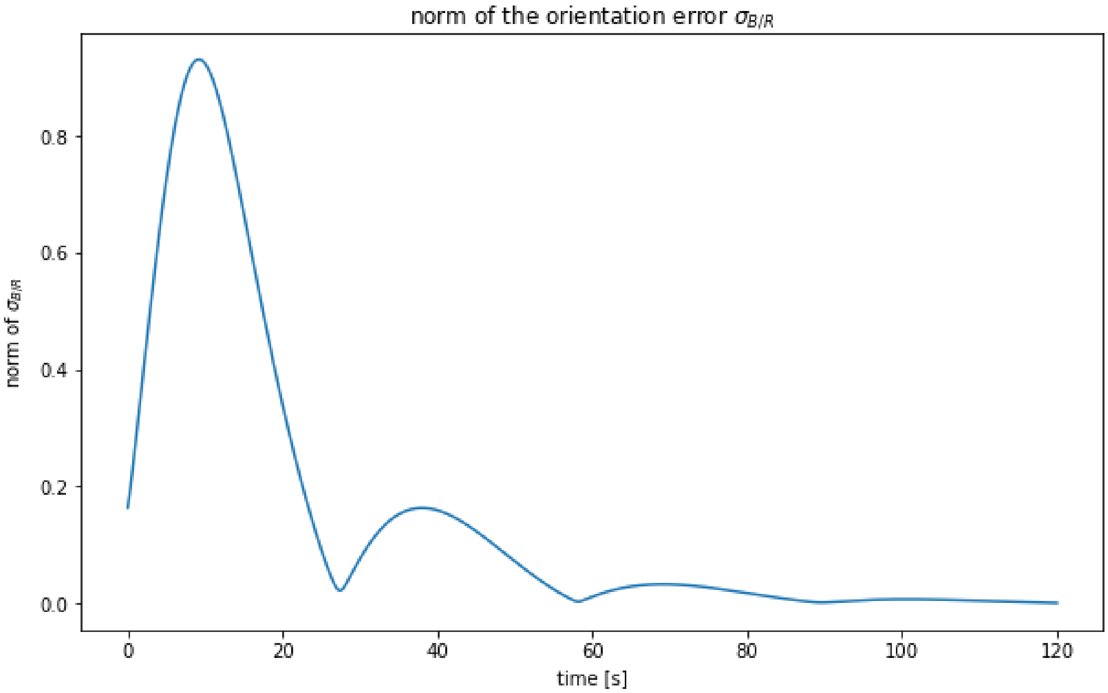




[63]:
```
print('the error orientation at time 40 sec and its norm are: ', S_BR[:,3000], '␣
↪', norm(S_BR[:,3000],2))
```

```
the error orientation at time 40 sec and its norm are:   [-0.07267975 -0.02026086
0.01024423]   0.07614323489198148
```

[11]:
```
plt.figure(figsize = (10,6))
plt.plot(np.arange(0,120,0.01), norm(S_BR, 2, 0))
plt.xlabel('time [s]')
plt.ylabel('norm of $\sigma_{B/R}$')
plt.title('norm of the orientation error $\sigma_{B/R}$')
plt.show()
```



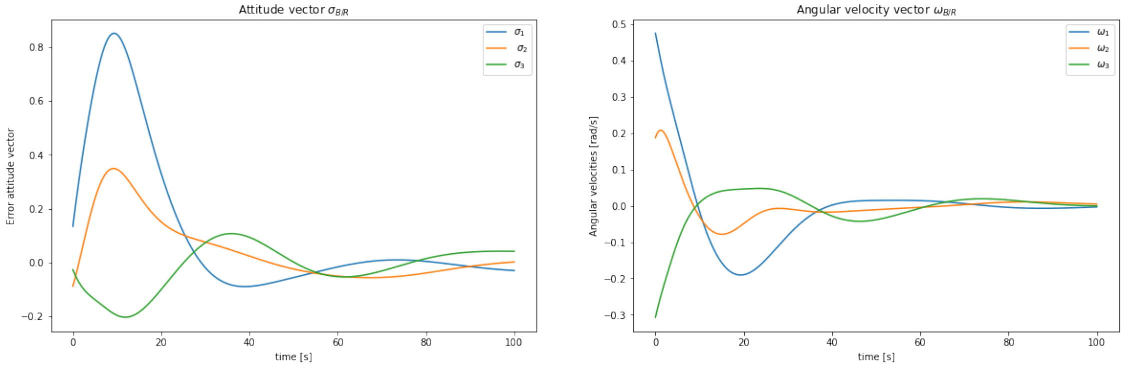
1.3 Concept Check 2 - Asymptotic Stability

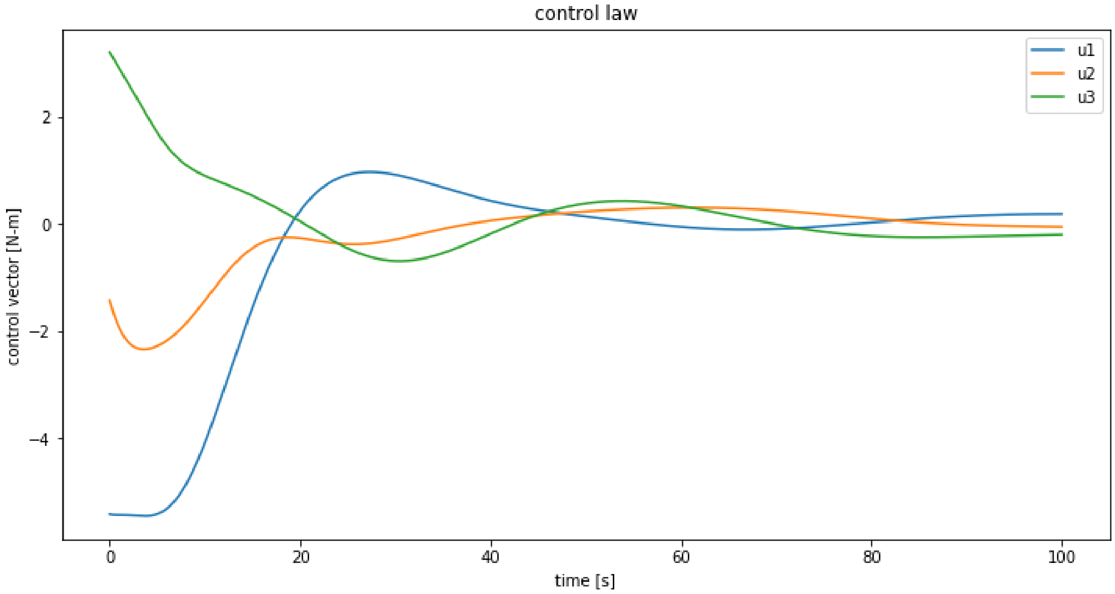
5. Same problem as above, i,e, a Tracking problem with the same parameters and initial conditions but now we use the following control law:

$$\boldsymbol{u} = -k\boldsymbol{\sigma}_{B/R} - P\delta\boldsymbol{\omega}$$

Recall that $\boldsymbol{\sigma}_{B/R}$ is the error attitude and $\delta\boldsymbol{\omega}$ is simply the angular velocity of the body frame relative to the reference frame R ${}^{B}\boldsymbol{\omega}_{B/R}$. This control law is implemented above as the function control2.

```
[12]: S_BR2, W_BR2 = Tracking((0,100), sigma_0, w_0, 0.01, 0.05, I, 0, k, P,␣
      ↪control_law = 2)
```

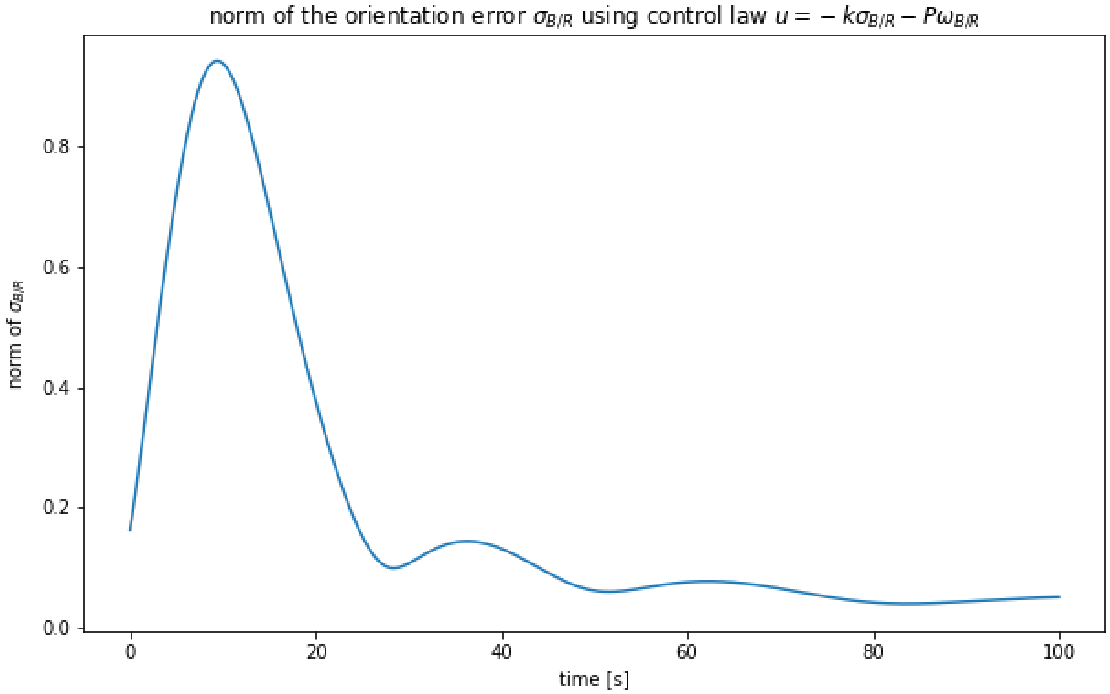




```
[13]: print('the error orientation at time 20 sec and its norm are: ', S_BR2[:,2000],␣
       ↪' ', norm(S_BR2[:,2000],2))
```

```
the error orientation at time 20 sec and its norm are:  [ 0.33090697  0.15073047
-0.10127221]  0.37745882012867704
```

```
[14]: plt.figure(figsize = (10,6))
      plt.plot(np.arange(0,100,0.01), norm(S_BR2, 2, 0))
      plt.xlabel('time [s]')
      plt.ylabel('norm of $\sigma_{B/R}$')
      plt.title('norm of the orientation error $\sigma_{B/R}$ using control law $u =␣
       ↪-k\sigma_{B/R} - P\omega_{B/R}$')
      plt.show()
```



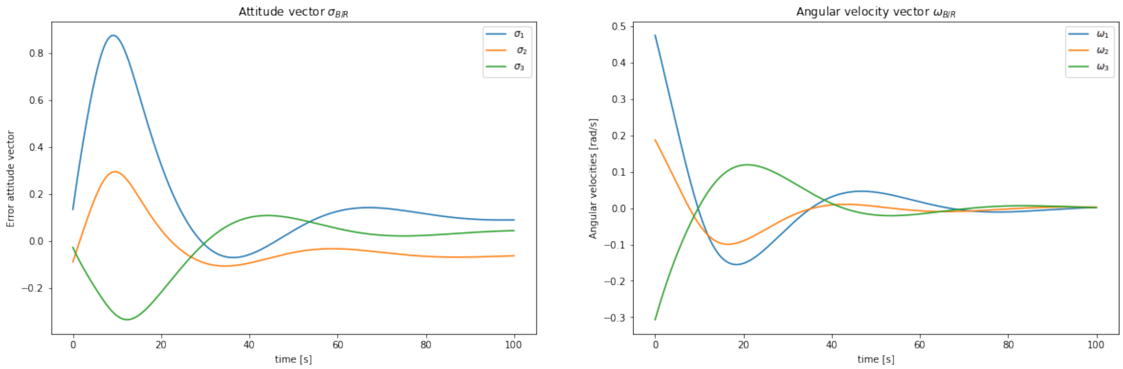
6. Simulate the attitude respone if the control is

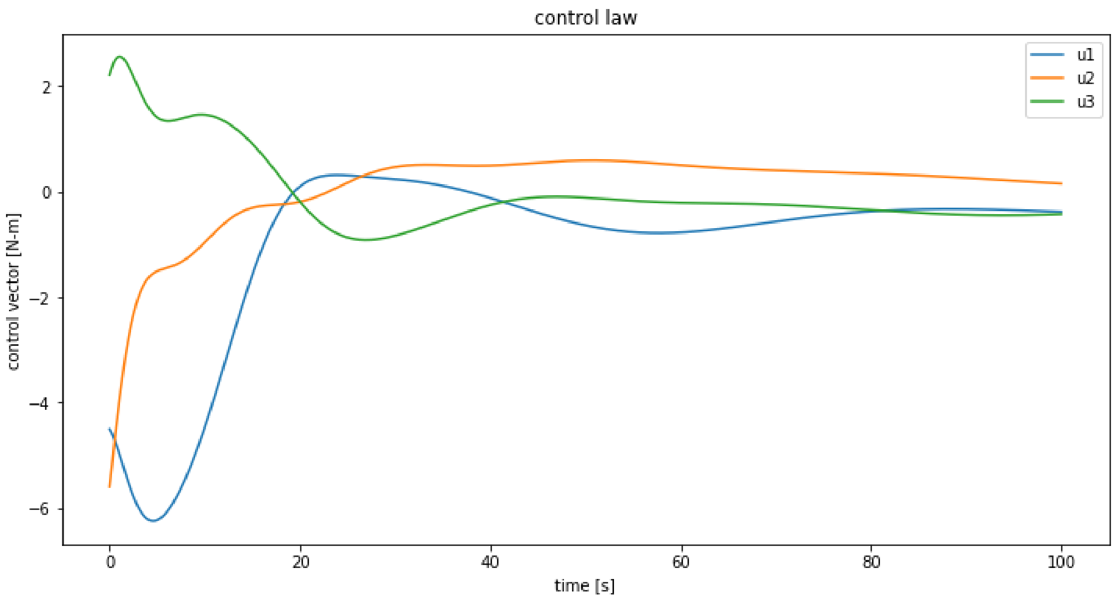
$$\boldsymbol{u} = -k\boldsymbol{\sigma}_{B/R} - P\boldsymbol{\omega}_{B/R} + I(\dot{\boldsymbol{\omega}}_{R/N} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{R/N}) + [\tilde{\boldsymbol{\omega}}_{B/N}]I\boldsymbol{\omega}_{B/N}$$

   Model an external torque of $\boldsymbol{L} = (0.5, -0.3, 0.2)Nm$. What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/R}|$ at 80 seconds into the simulation?

For this problem we use control_law = 3 which uses the control3 function implemented above.

[15]:
```
L = np.array([0.5, -0.3, +0.2])
S_BR3, W_BR3 = Tracking((0,100), sigma_0, w_0, 0.01, 0.05, I, L, k, P, control_law = 3)
```

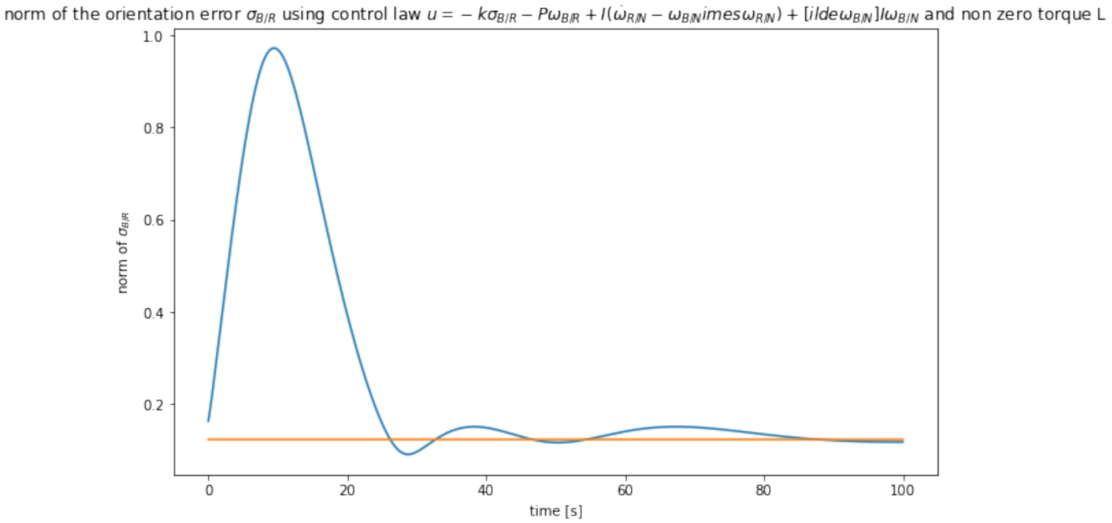




```
[16]: print('the orientation error at time 80 sec and its norm are: ', S_BR3[:,8000],␣
       ↪' ', norm(S_BR3[:,8000],2))
```

```
the orientation error at time 80 sec and its norm are:  [ 0.11505072 -0.06533818
0.02373139]   0.13442070005770307
```

```
[17]: plt.figure(figsize = (10,6))
      plt.plot(np.arange(0,100,0.01), norm(S_BR3, 2, 0))
      plt.plot(np.arange(0,100,0.01), norm(L/k,2)*np.ones(10000))
      plt.xlabel('time [s]')
      plt.ylabel('norm of $\sigma_{B/R}$')
      plt.title('norm of the orientation error $\sigma_{B/R}$ using control law $u␣
       ↪=-k\sigma_{B/R} - P\omega_{B/R} + I(\dot{\omega}_{R/N} -\omega_{B/
       ↪N}\times\omega_{R/N}) +[\tilde{\omega}_{B/N}]I\omega_{B/N}$ and non zero␣
       ↪torque L')
      plt.show()
```



```
[18]: print('norm of the torque L divided by the gain k is: ', norm(L/k,2))
```

```
norm of the torque L divided by the gain k is:  0.12328828005937953
```

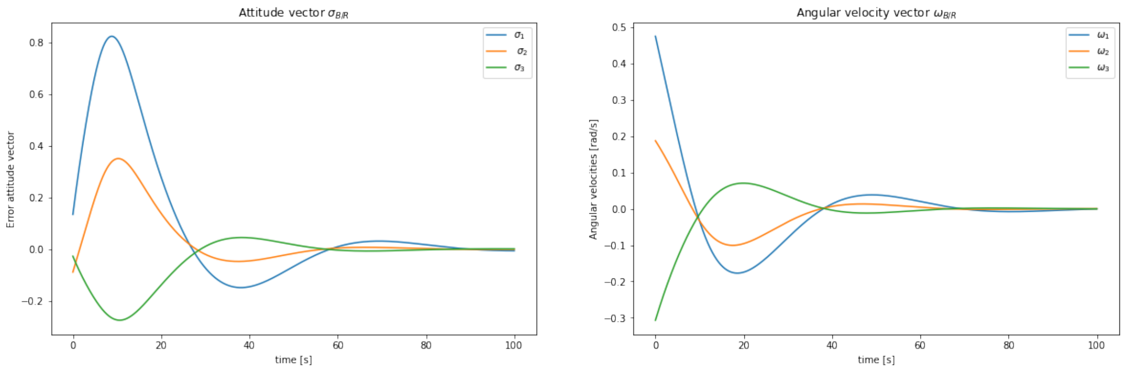
7. Simulate the attitude respone if the control is

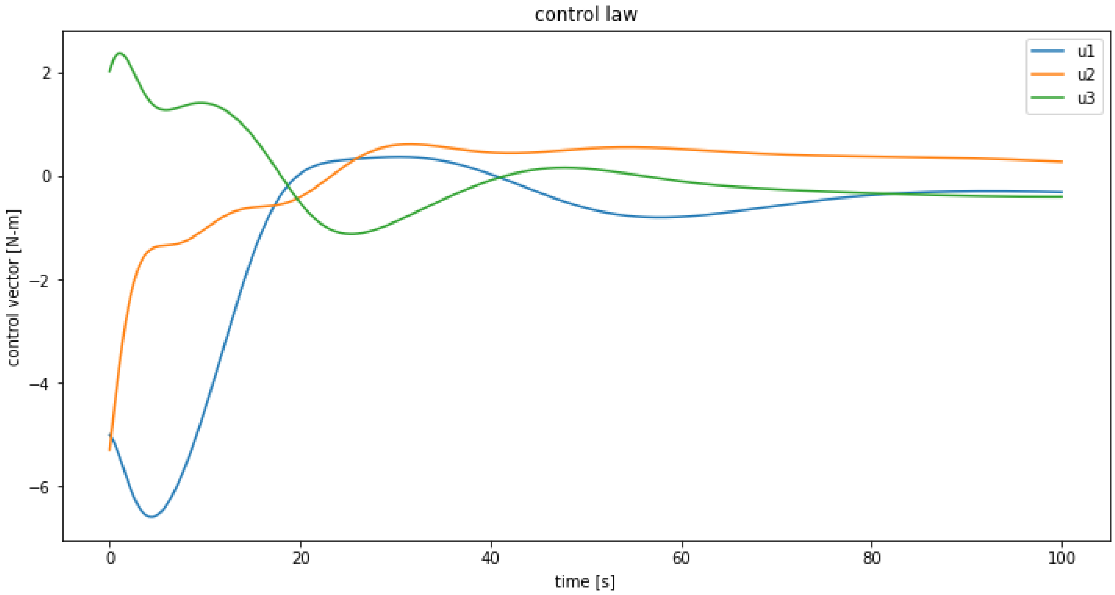
$$\boldsymbol{u} = -k\boldsymbol{\sigma}_{B/R} - P\boldsymbol{\omega}_{B/R} + I(\dot{\boldsymbol{\omega}}_{R/N} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{R/N}) + [\tilde{\boldsymbol{\omega}}_{B/N}]I\boldsymbol{\omega}_{B/N} - \boldsymbol{L}$$

Model an external torque of $\boldsymbol{L} = (0.5, -0.3, 0.2)Nm$. What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/R}|$ at 70 seconds into the simulation?

For this problem we use control_law = 1 which uses the control1 function implemented above. This will be identical to the case of $L = 0$ since in the dynamic equation the torque will cancel.

```
[19]: I = np.diag([100,75,80])
      k = 5
      P = 10*np.eye(3)
      sigma_0 = np.array([0.1, 0.2, -0.1])
      w_0 = np.array([30, 10, -20])  # deg/s
      w_0 = (np.pi/180)*w_0
      L = np.array([0.5, -0.3, 0.2])
      S_BR1, W_BR1 = Tracking((0,100), sigma_0, w_0, 0.01, 0.05, I, L, k, P,␣
       ↪control_law = 1)
```

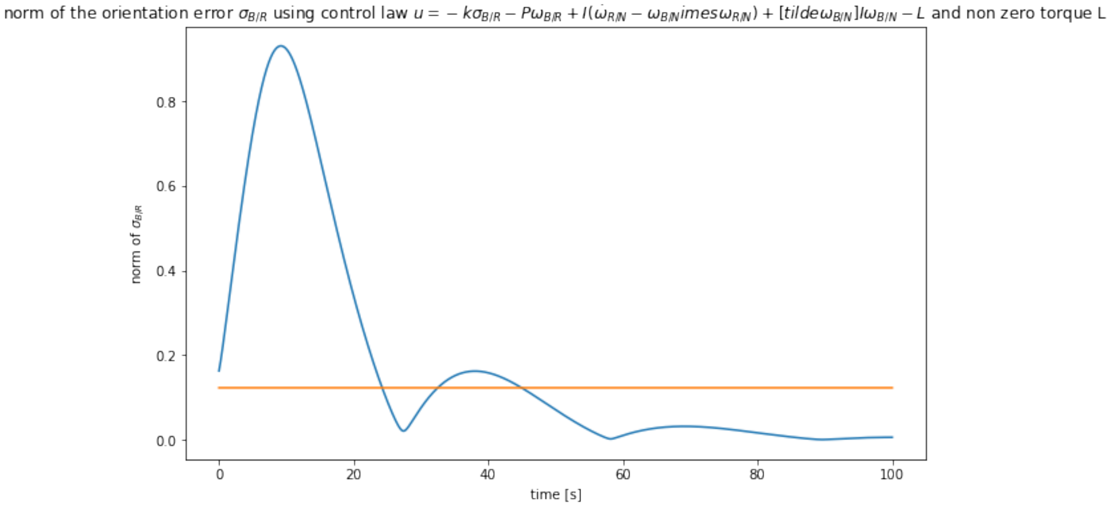




```
[20]: print('the orientation error at time 70 sec and its norm are: ', S_BR1[:,7000],␣
        ↪' ', norm(S_BR1[:,7000],2))
```

```
the orientation error at time 70 sec and its norm are:  [ 0.03079537  0.00626207
-0.00687998]  0.032169898516934956
```

```
[21]: plt.figure(figsize = (10,6))
      plt.plot(np.arange(0,100,0.01), norm(S_BR1, 2, 0))
      plt.plot(np.arange(0,100,0.01), norm(L/k,2)*np.ones(10000))
      plt.xlabel('time [s]')
      plt.ylabel('norm of $\sigma_{B/R}$')
      plt.title('norm of the orientation error $\sigma_{B/R}$ using control law $u␣
       ↪=-k\sigma_{B/R} - P\omega_{B/R} + I(\dot{\omega}_{R/N} -\omega_{B/
       ↪N}\times\omega_{R/N}) +[tilde{\omega}_{B/N}]I\omega_{B/N} - L$ and non zero␣
       ↪torque L')
      plt.show()
```



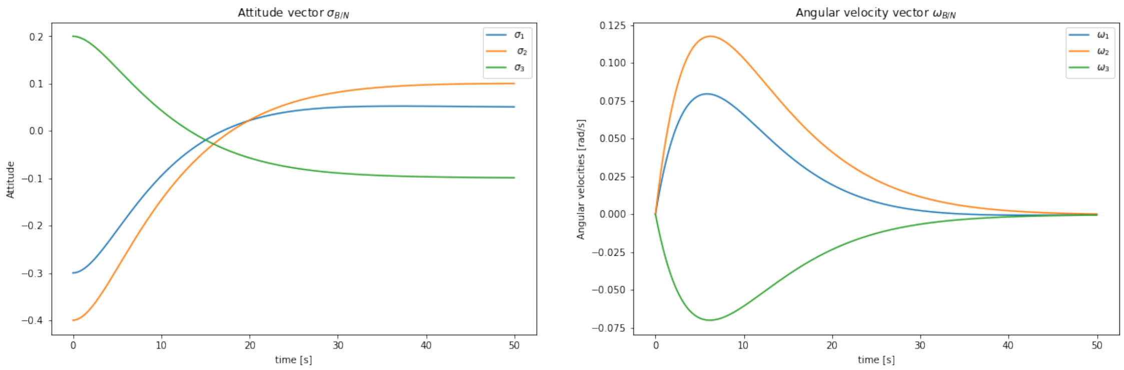
Example 8.10 (from Schaub - Junkins Book) A rogid body with a large initial attitude error is to be brought to rest to the zero reference attitude. All three principal inertias are $10kgm^2$. The rigid body is initially at rest with an MRP attitude vector $\boldsymbol{\sigma}(t_0) = (-0.3, -0.4, 0.2)^T$. The control law used to stabilize the body is of the simple PD form

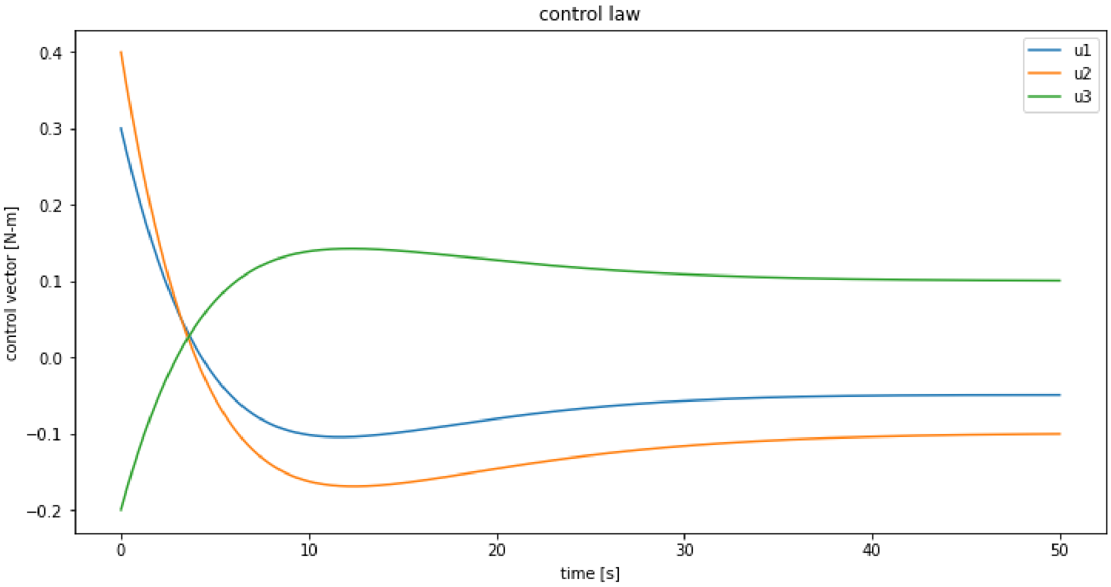
$$\boldsymbol{u} = -k\boldsymbol{\sigma} - P\boldsymbol{\omega}.$$

The scalar feedback gains k, P are chosen to be $1kgm^2/s^2$ and $3kgm^2/s$ respectively.

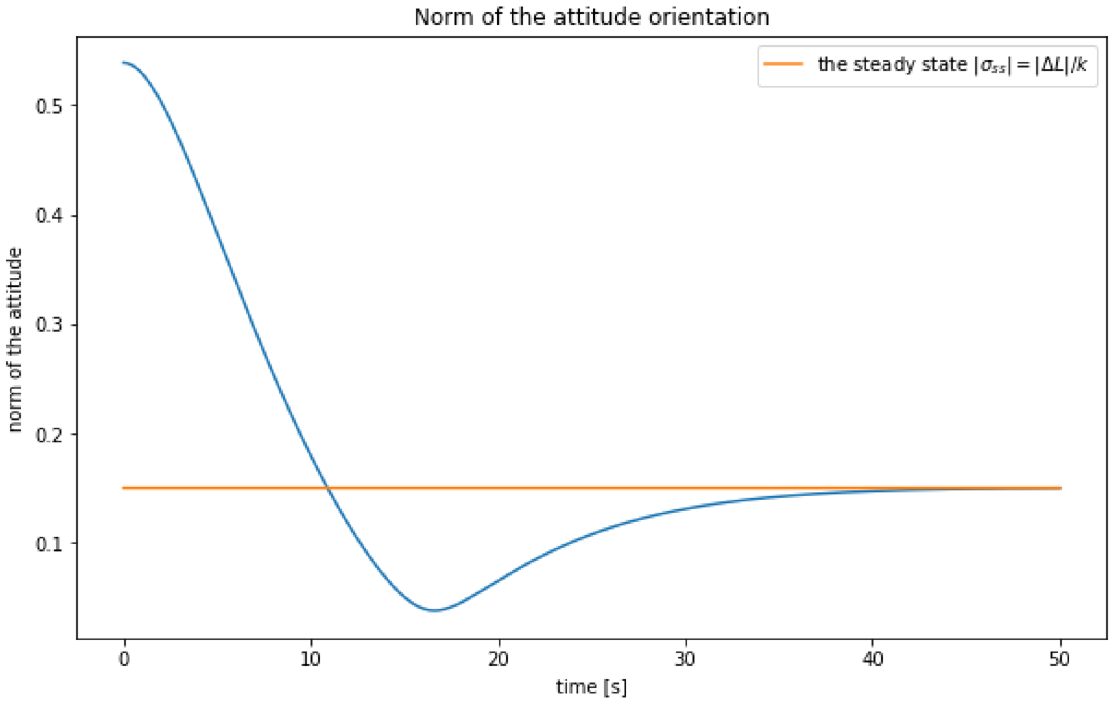
[22]:
```
I = 10*np.eye(3)
s_0 = np.array([-0.3,-0.4,0.2])
w_0 = np.zeros(3)
k = 1
P = 3*np.eye(3)
DeltaL = np.array([0.05, 0.10, -0.10])
```

[23]:
```
S, W = Regulator((0,50), s_0, w_0, 0.01, I, 0, k, P, control_law = 2, K_I = 0,␣
 ↪DeltaL =DeltaL)
```





```
[24]: plt.figure(figsize=(10,6))
plt.plot(np.arange(0,50,0.01), norm(S, 2, 0))
plt.plot(np.arange(0,50,0.01), norm(DeltaL/k)*np.ones(5000), label = 'the steady state $|\sigma_{ss}| = |\Delta L|/k$')
plt.xlabel('time [s]')
plt.ylabel('norm of the attitude')
plt.title('Norm of the attitude orientation')
plt.legend()
plt.show()
```



1.3.1 Concept Check 3 - Unknown External Torques

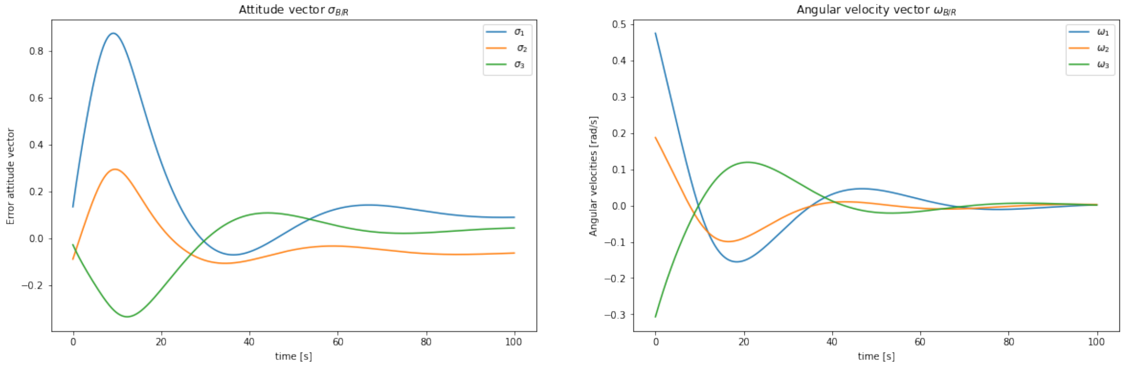
4. Simulate the control response and validate that the spacecraft attitude settles to this predicted steady state pointing error. The control law is given by control1, i.e:

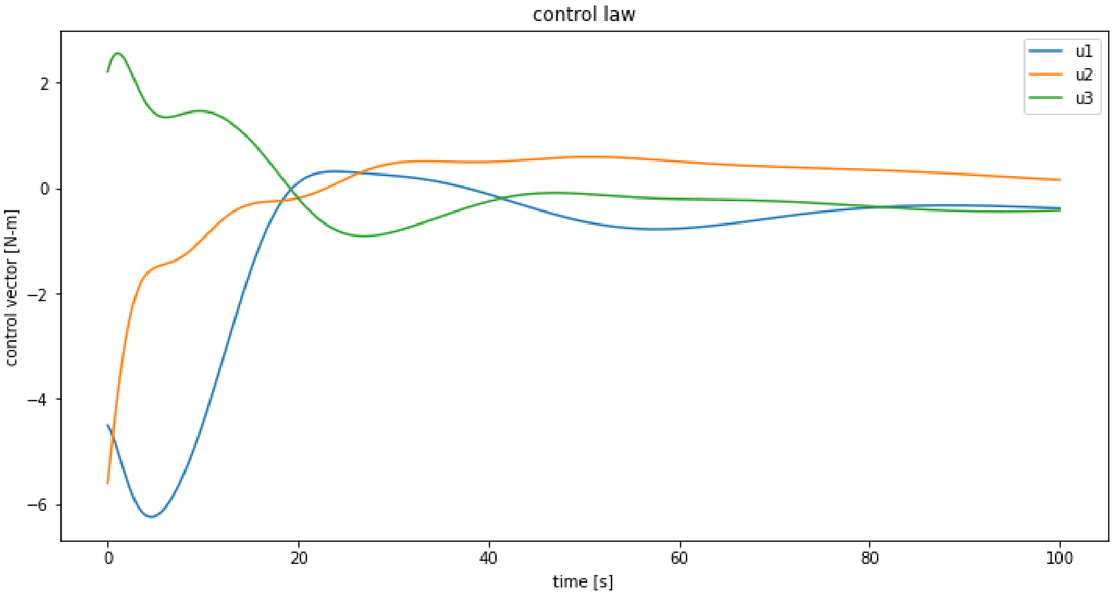
$$\boldsymbol{u} = -k\boldsymbol{\sigma}_{B/R} - P\boldsymbol{\omega}_{B/R} + I(\dot{\boldsymbol{\omega}}_{R/N} - \boldsymbol{\omega}_{B/N} \times \boldsymbol{\omega}_{R/N}) + [\tilde{\boldsymbol{\omega}}_{B/N}]I\boldsymbol{\omega}_{B/N} - \boldsymbol{L}$$

but there is an external unknown torque given by: $\Delta L = (0.5, -0.3, 0.2$ What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/R}|$ at 35 seconds into the simulation?

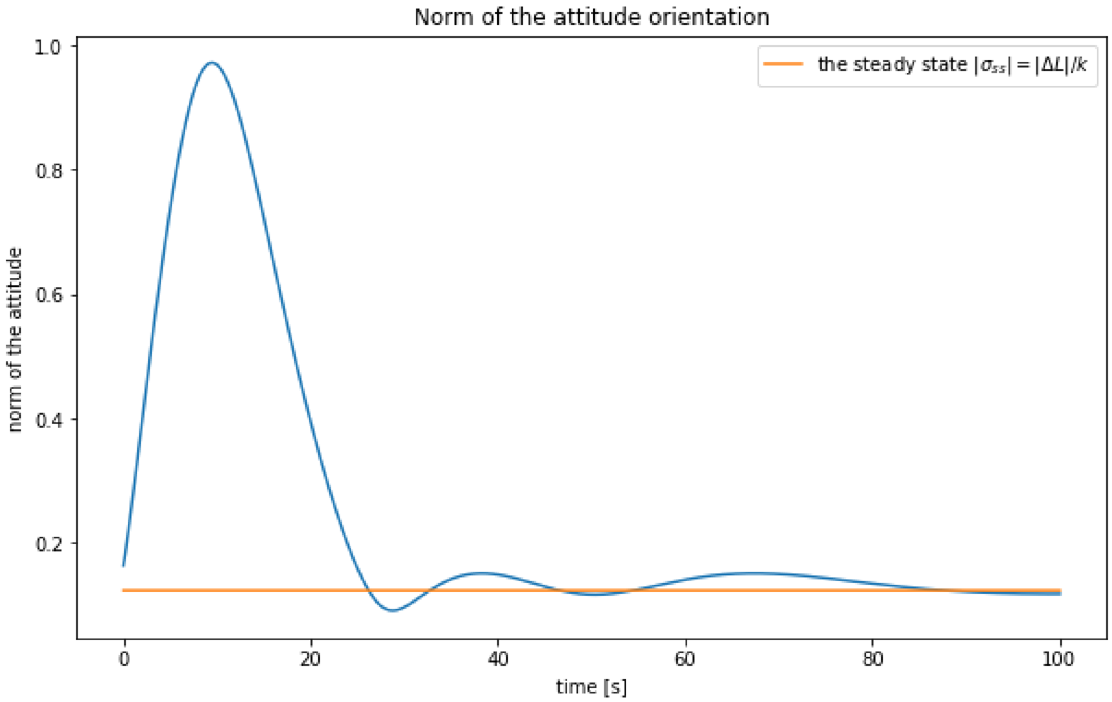
[25]:

```
I = np.diag([100,75,80])
k = 5
P = 10*np.eye(3)
sigma_0 = np.array([0.1, 0.2, -0.1])
w_0 = np.array([30, 10, -20])  # deg/s
w_0 = (np.pi/180)*w_0
deltaL = np.array([0.5, -0.3, 0.2])
S_BR4, W_BR4 = Tracking((0,100), sigma_0, w_0, 0.01, 0.05, I, deltaL, k, P, control_law = 3)
```





[26]:
```
plt.figure(figsize=(10,6))
plt.plot(np.arange(0,100,0.01), norm(S_BR4, 2, 0))
plt.plot(np.arange(0,100,0.01), norm(deltaL/k)*np.ones(10000), label = 'the␣
 ↪steady state $|\sigma_{ss}| = |\Delta L|/k$')
plt.xlabel('time [s]')
plt.ylabel('norm of the attitude')
plt.title('Norm of the attitude orientation')
plt.legend()
plt.show()
```



[27]:

```
print('The tracking error MRP $|\simga_{B/R}$ at 35 seconds is : ', norm(S_BR4[:
↪,3500],2))
```

```
The tracking error MRP $|\simga_{B/R}$ at 35 seconds is :  0.14156469328495994
```

1.3.2 Integral Feedback Control

Example 8.11 of PID control (from Schaub - Junkins Book) A rigid body with a large initial attitude error is to be brought to rest to the zero reference attitude. All three principal inertias are $10kgm^2$. The initial angular velocity $\boldsymbol{\omega}(t_0) = (0.2, 0.2, 0.2)rad/s$ with an MRP attitude vector $\boldsymbol{\sigma}(t_0) = (-0.3, -0.4, 0.2)^T$. The control law used to stabilize the body is of the simple PID form

$$\boldsymbol{u} = -k\boldsymbol{\sigma} - P\boldsymbol{\omega} - PK_I\boldsymbol{z} + I(\dot{\boldsymbol{\omega}}_r - [\tilde{\boldsymbol{\omega}}]\boldsymbol{\omega}_r) + [\tilde{\boldsymbol{\omega}}]I\boldsymbol{\omega} - L$$

with

$$\boldsymbol{z}(t) = \int_0^t (K\boldsymbol{\sigma} + I\delta\dot{\boldsymbol{\omega}})dt = K\int_0^t \boldsymbol{\sigma}dt + I(\delta\boldsymbol{\omega} - \delta\boldsymbol{\omega}_0)$$

and $\delta\boldsymbol{\omega}_0$ is the initial body angular velocity error vector. The control law can be re expressed in the following way:

$$\boldsymbol{u} = -K\boldsymbol{\sigma} - P\delta\boldsymbol{\omega} - PK_I I\delta\boldsymbol{\omega} - KPK_I\int_0^t \boldsymbol{\sigma}dt + PK_I I\delta\boldsymbol{\omega}_0 + I(\dot{\boldsymbol{\omega}}_r - [\tilde{\boldsymbol{\omega}}]\boldsymbol{\omega}_r) + [\tilde{\boldsymbol{\omega}}]I\boldsymbol{\omega} - L$$

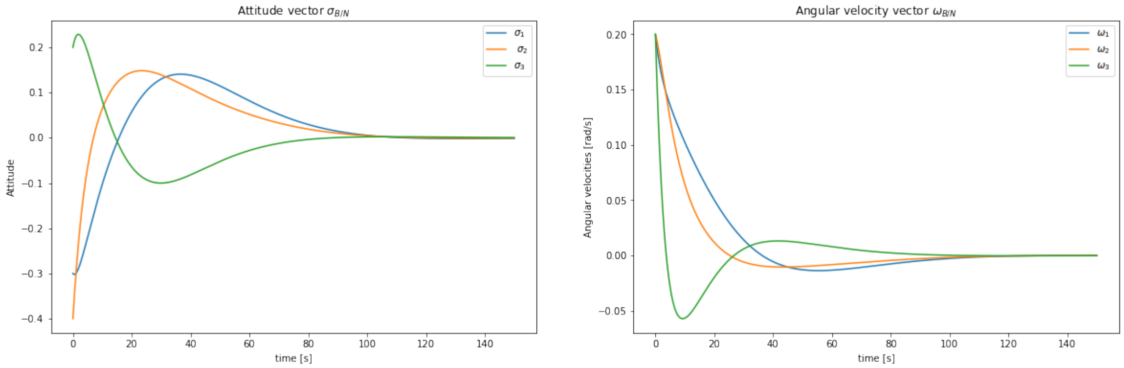
The scalar feedback gains k, p are chosen to be $1kgm^2/s^2$ and $3kgm^2/s$ respectively. The feedback gain corresponding to the integral term is $K_I = 0.01 * \mathbb{I}$

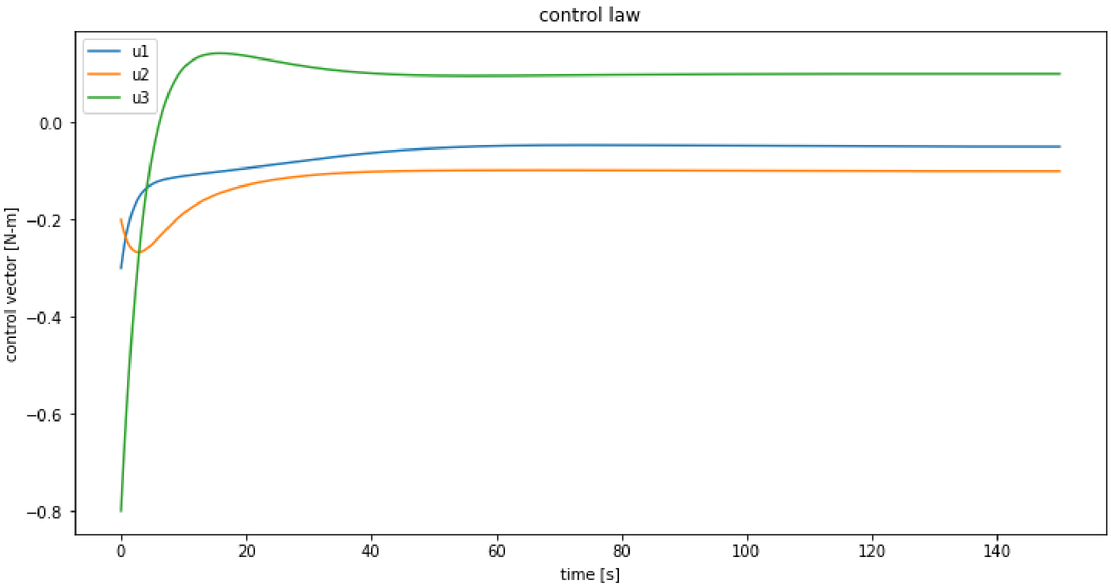
[28]:

```python
I = 10*np.eye(3)
s_0 = np.array([-0.3,-0.4,0.2])
w_0 = np.array([0.2, 0.2, 0.2])
k = 1
P = 3*np.eye(3)
DeltaL = np.array([0.05, 0.10, -0.10])
K_I = 0.01*np.eye(3)
```

[29]:

```python
S_pid, W_pid, Z = Regulator((0,150), s_0, w_0, 0.01, I, 0, k, P, control_law = 'PID', K_I= K_I, DeltaL =DeltaL)
```

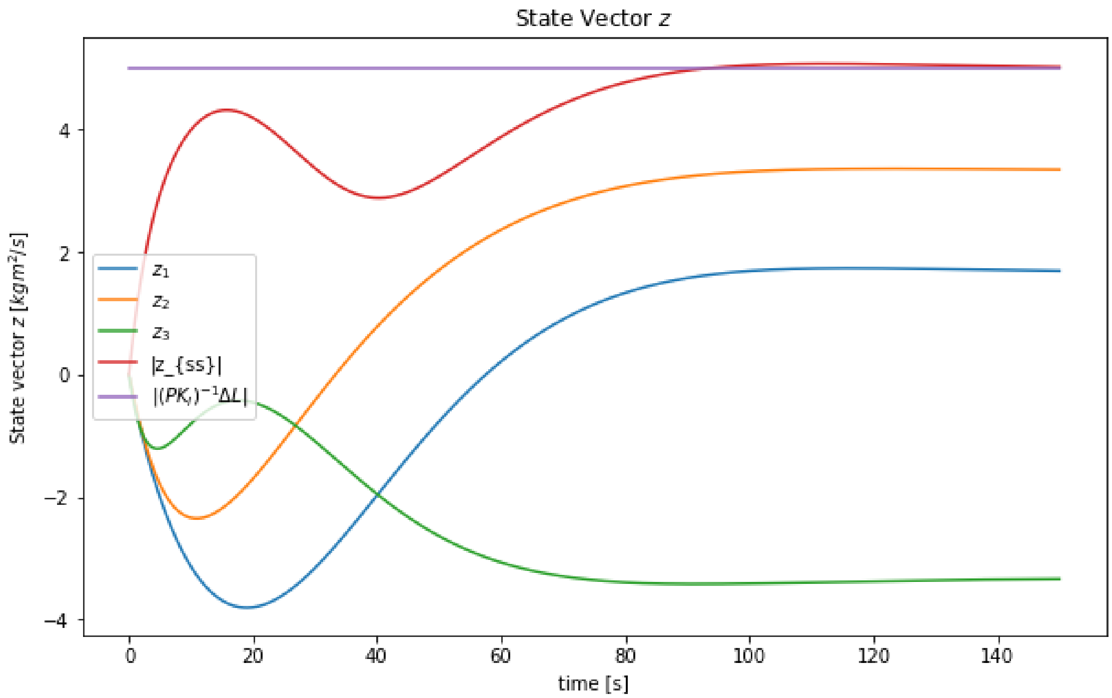




[30]:

```python
z1 = Z[0,:]
z2 = Z[1,:]
z3 = Z[2,:]
```

```
plt.figure(figsize = (10,6))
plt.plot(np.arange(0,150,0.01), z1, label ='$z_1$')
plt.plot(np.arange(0,150,0.01), z2, label ='$z_2$')
plt.plot(np.arange(0,150,0.01), z3, label ='$z_3$')
plt.plot(np.arange(0,150,0.01), norm(Z, 2, 0), label = '|z_{ss}|' )
plt.plot(np.arange(0,150,0.01), norm(inv(P@K_I)@DeltaL,2)*np.ones(15000), label␣
 ↪='$|(PK_I)^{-1}\Delta L|$')
plt.xlabel('time [s]')
plt.ylabel('State vector $z$ $[kgm^2/s]$')
plt.title('State Vector $z$')
plt.legend()
plt.show()
```



[31]:
```
z_ss = inv(P@K_I)@DeltaL
print('The steady state for z is $z_{ss} =lim_{t \to 0}z = (PK_I)^{-1}\Delta L =␣
 ↪$', z_ss, '$kgm^2/s')
```

```
The steady state for z is $z_{ss} =lim_{t       o 0}z = (PK_I)^{-1}\Delta L = $
[ 1.66666667  3.33333333 -3.33333333] $kgm^2/s
```

Concept Check 4 - Integral Feedback

2. Write software code to simulate the general spacecraft motion where a control torque $\boldsymbol{u}$ is being applied. The principal inertias are $I = diag(100, 75, 80)kgm^2$. The initial states are $\boldsymbol{\sigma}_{B/N}(t_0) = (0.1, 0.2, -0.1)$ and $\boldsymbol{\omega}_{B/N}(t_0) = (3, 1, -2)deg/s$. Use the gains $K = 5Nm$, $P =$

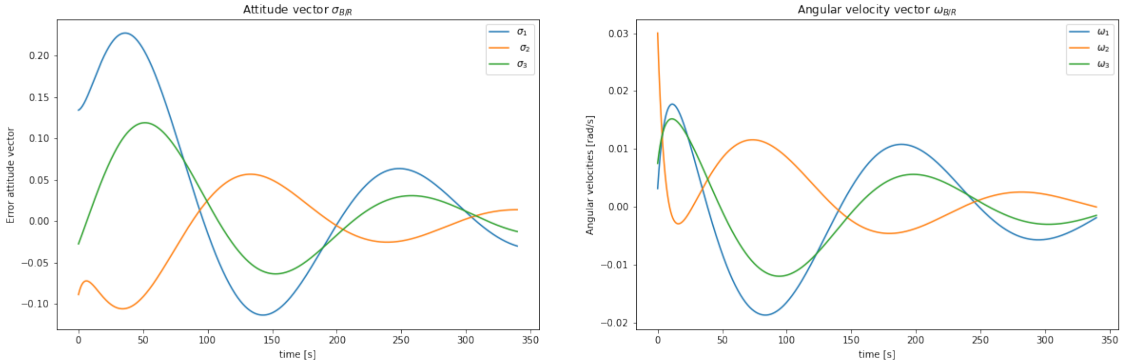
$10 * \mathbb{I} Nm$, $K_I = 0.005$. The reference attitude is $\boldsymbol{\sigma}_{R/N} = (0.2\sin(ft), 0.3\cos(ft), -0.3\sin(ft))$ (the same reference is being implemented along this week of the course) with $f = 0.05 rad/s$. The unknown external torque is $\delta L = (0.5, -0.3, 0.2) Nm$, while $\boldsymbol{L} = 0$. First simulate the nonlinear integral control (control PID) for 4 minutes. What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/R}|$ at 45 seconds into the simulation.

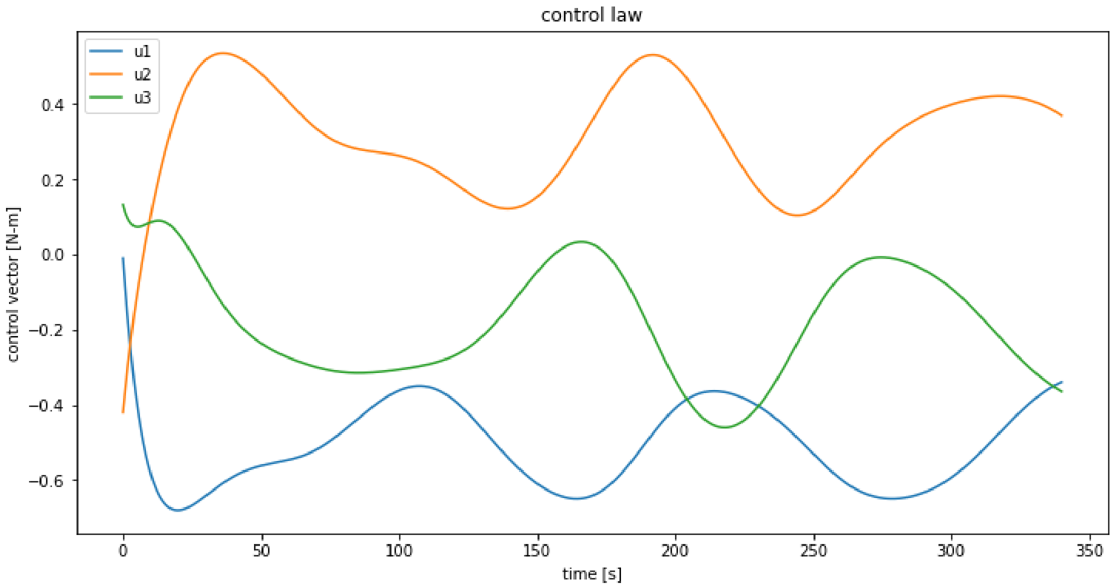
[32]:

```
I = np.diag([100, 75, 80])
sigma_0 = np.array([0.1,0.2, -0.1])
w_0 = (np.pi/180)*np.array([3,1,-2]) # rad/sec
K = 5 # Nm
P = 10*np.eye(3)
K_I = 0.005*np.eye(3)
f = 0.05 # rad/sec
DeltaL = np.array([0.5, -0.3, 0.2]) #Nm
L = np.zeros(3)
dt = 0.1
```

[33]:

```
S_BR5, W_BR5, Z5 = Tracking((0,340), sigma_0, w_0, dt, f, I, L, K, P, 'PID',␣
↪K_I,  DeltaL)
```

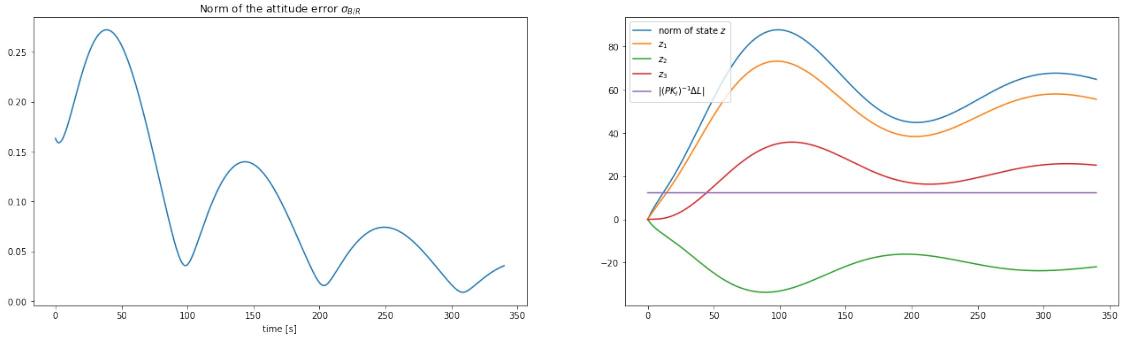




[34]:

```
plt.figure(figsize = (22,6))

plt.subplot(121)
plt.plot(np.arange(0,340,dt), norm(S_BR5,2,0))
plt.xlabel('time [s]')
plt.title('Norm of the attitude error $\sigma_{B/R}$')
plt.subplot(122)
plt.plot(np.arange(0,340,dt), norm(Z5,2,0), label = 'norm of state $z$')
plt.plot(np.arange(0,340,dt), Z5[0,:], label = '$z_1$')
plt.plot(np.arange(0,340,dt), Z5[1,:], label = '$z_2$')
plt.plot(np.arange(0,340,dt), Z5[2,:], label = '$z_3$')
plt.plot(np.arange(0,340,dt), norm(inv(P@K_I)@DeltaL,2)*np.ones(3400), label
 ↪='$|(PK_I)^{-1}\Delta L|$')
plt.legend()
plt.show()
```



[35]:
```
print('The norm attitude error $\sigma_{B/R} at 45 seconds in to the simulation␣
 ↪is', S_BR5[:,450], norm(S_BR5[:,450], 2))
```

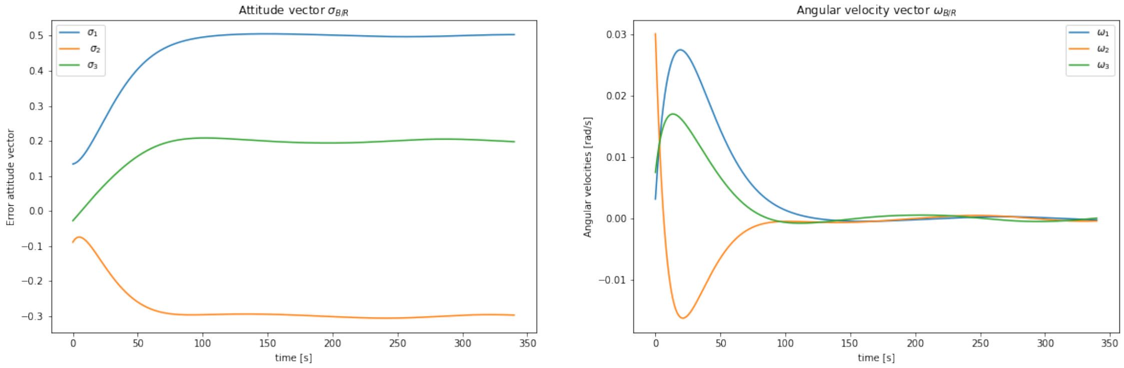
```
The norm attitude error $\sigma_{B/R} at 45 seconds in to the simulation is [
0.21933445 -0.09898844  0.11623972] 0.26724143790638094
```

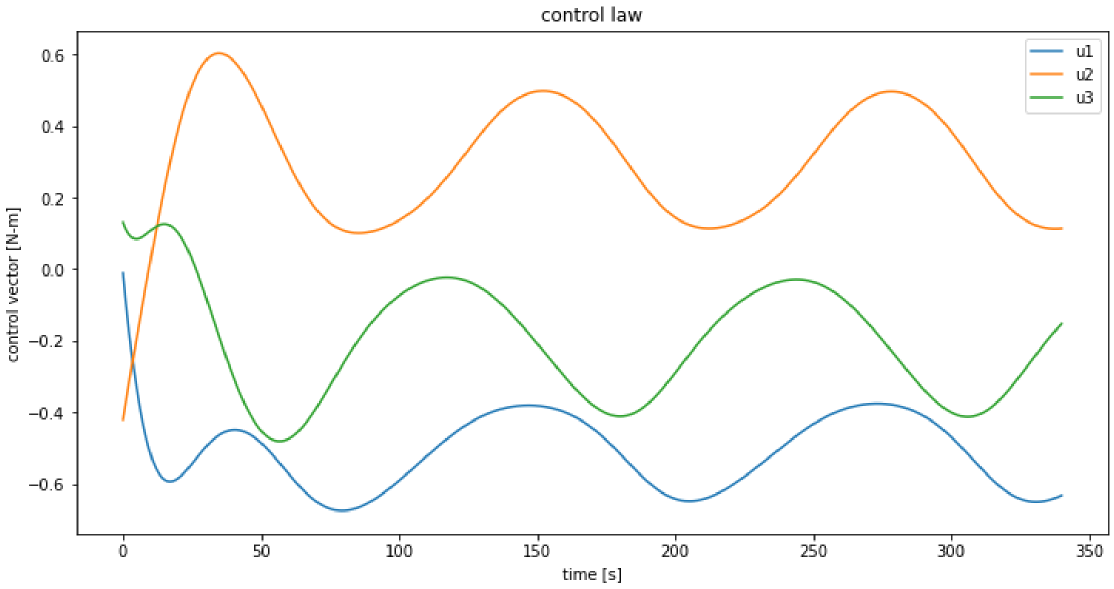
[36]:
```
inv(P@K_I)@DeltaL
```

[36]:
```
array([10., -6.,  4.])
```

4. For comparison, simulate the same control for 4 minutes with $K_I = 0\;0$ to turn off the integral gain. What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/R}|$ at 35 seconds in to the simulation?

[37]:
```
S_BR6, W_BR6, Z6 = Tracking((0,340), sigma_0, w_0, dt, f, I, L, K, P, 'PID',␣
 ↪K_I=0*np.eye(3),  DeltaL= DeltaL)
```



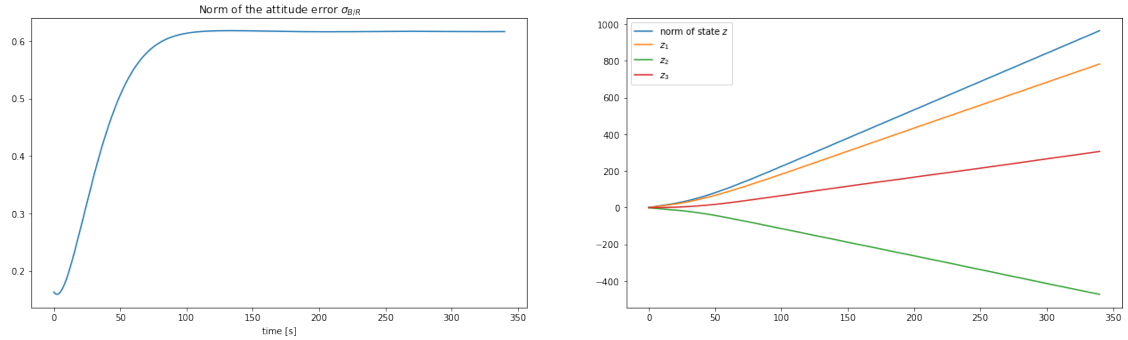


```
[38]: print('The norm attitude error $\sigma_{B/R} at 35 seconds in to the simulation␣
       ↪is', S_BR6[:,350], norm(S_BR6[:,350], 2))
```

```
The norm attitude error $\sigma_{B/R} at 35 seconds in to the simulation is [
0.32988367 -0.20876877  0.11257356] 0.40630092294331827
```

```
[39]: plt.figure(figsize = (22,6))

      plt.subplot(121)
      plt.plot(np.arange(0,340,dt), norm(S_BR6,2,0))
      plt.xlabel('time [s]')
      plt.title('Norm of the attitude error $\sigma_{B/R}$')
      plt.subplot(122)
      plt.plot(np.arange(0,340,dt), norm(Z6,2,0), label = 'norm of state $z$')
      plt.plot(np.arange(0,340,dt), Z6[0,:], label = '$z_1$')
      plt.plot(np.arange(0,340,dt), Z6[1,:], label = '$z_2$')
      plt.plot(np.arange(0,340,dt), Z6[2,:], label = '$z_3$')
      plt.legend()
      plt.show()
```



```
[40]: print('since $K_I = 0$ in this case, the tate vector $z$ blows up to infinity!')
```

```
since $K_I = 0$ in this case, the tate vector $z$ blows up to infinity!
```

1.3.3 Feedback Gain Selection

Concept Check 5 - Feedback Gain Selection

1. Consider the regulation problem where the unperturbed spacecraft is driven to the zero orientation. The control law is:

$$\boldsymbol{u} = -k\boldsymbol{\sigma}_{B/N} - P\boldsymbol{\omega}_{B/N} + [\tilde{\boldsymbol{\omega}}_{B/N}]I\boldsymbol{\omega}_{B/N}$$

The principal inertias are $I_1 = 100kgm^2$, $I_2 = 75kgm^2$ and $I_3 = 80kgm^2$. Use the gain $k = 5Nm$. Choose a diagonal gain matrix $P = diag(P_1, P_2, P_3)$ such that the linearized closed loop response critically damped with $\xi_i = 1$.

Ans: The linearized closed loop consists of the system of linear differential equations obtained by the constrain of the Lyapunov function $V(\boldsymbol{\sigma}, \delta\boldsymbol{\omega}) = \frac{1}{2}\delta\boldsymbol{\omega}^T I \delta\boldsymbol{\omega} + 2k\ln(1+\boldsymbol{\sigma}^T\boldsymbol{\sigma}$. Then the closed loop dynamics is:

$$I\delta\dot{\boldsymbol{\omega}} + P\delta\boldsymbol{\omega} + k\boldsymbol{\sigma} = 0$$

together with the kinematic differential equation of the MRP's

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}B(\boldsymbol{\sigma})\delta\boldsymbol{\omega} \approx \frac{1}{4}\delta\boldsymbol{\omega}$$

or equivalently:

$$\begin{pmatrix} \dot{\boldsymbol{\sigma}} \\ \delta\dot{\boldsymbol{\omega}} \end{pmatrix} = \begin{bmatrix} 0 & \frac{1}{4}\mathbb{I}_{3\times 3} \\ -kI^{-1} & -I^{-1}P \end{bmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \delta\boldsymbol{\omega} \end{pmatrix}$$

In the case of diagonal matrices I and P, the above linear system gives rise to three 2×2 linear system, one for each coordinate functions $\sigma_i, \delta\omega_i$, $i = 1, 2, 3$:

$$\begin{pmatrix} \dot{\sigma_i} \\ \delta\dot{\omega_i} \end{pmatrix} = \begin{bmatrix} 0 & \frac{1}{4} \\ -\frac{k}{I_i} & -\frac{P_i}{I_i} \end{bmatrix} \begin{pmatrix} \sigma_i \\ \delta\omega_i \end{pmatrix}$$

with corresponding natural frequencies ω_{n_i} and damping ratios ξ_i given by

$$\omega_{n_i} = \frac{1}{2I_i}\sqrt{kI_i}, \quad \xi_i = \frac{P_i}{\sqrt{kI_i}}$$

and with decay time constants (which indicate how long it would take for the state errors to decay to $1/e$ of their respective initial values)

$$T_i = \frac{2I_i}{P_i}.$$

For this problem we want critical damping ratios, i.e $\xi_i = 1$ for $i = 1, 2, 3$ therefore with the value of $k = 5$ and the inertias, we get

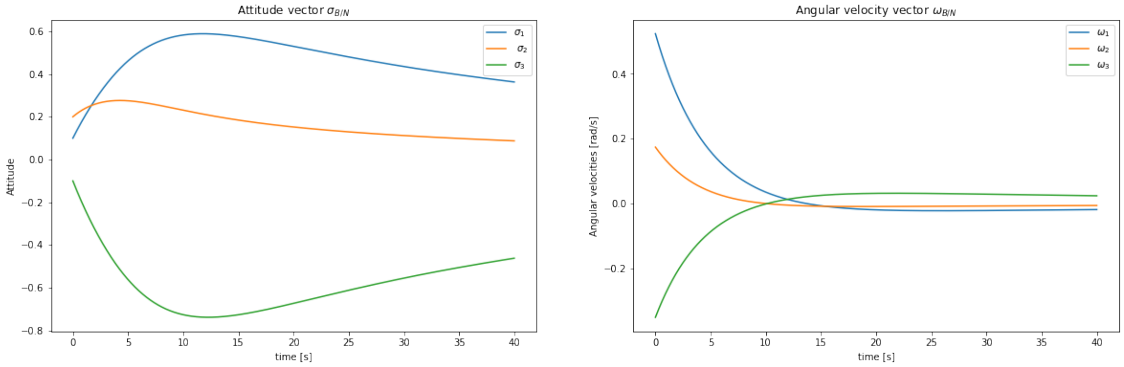
$$P = diag(\sqrt{kI_1}, \sqrt{kI_2}, \sqrt{kI_3}) = diag(22.36067977, 19.36491673, 20)$$

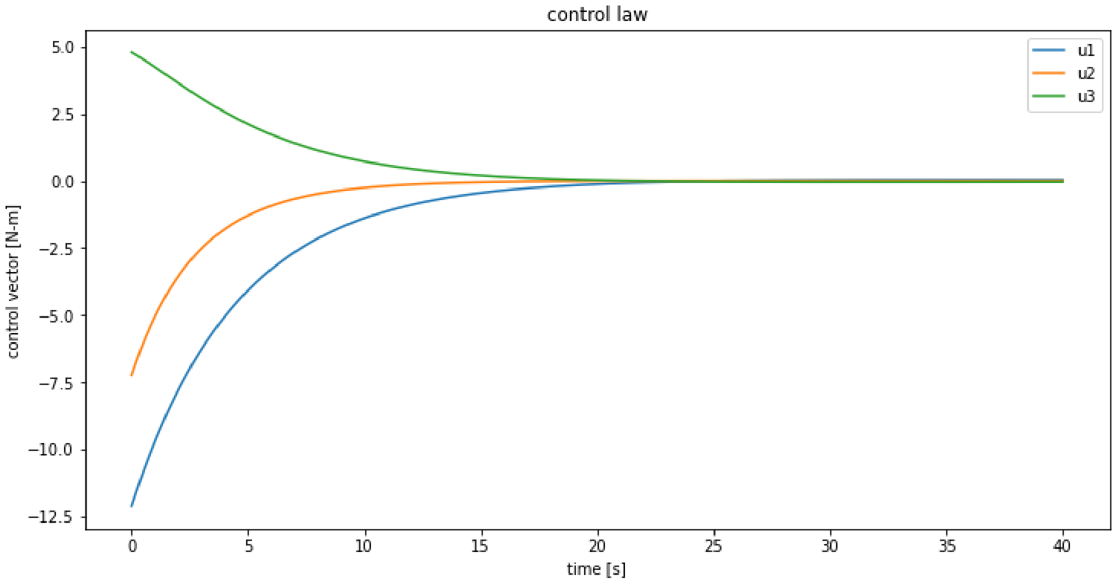
2. Simulate the closed loop response with the initial states $\boldsymbol{\sigma}_{B/N}(t_0) = (0.1, 0.2, -0.1)$ and $\boldsymbol{\omega}_{B/N}(t_0) = (30, 10, -20)$ deg $/s$. What is the tracking error MRP norm $|\boldsymbol{\sigma}_{B/N}|$ at 30 seconds into the simulation?

[41]:

```
I1, I2, I3 = 100, 75, 80
I = np.diag([I1,I2,I3])
K = 5
P= np.diag([np.sqrt(K*I1), np.sqrt(K*I2), np.sqrt(K*I3)])
s_bn_0 = np.array([0.1, 0.2, -0.1])
w_bn_0 = (np.pi/180)*np.array([30,10, -20])
L = np.zeros(3)
dt = 0.01
```

```
S7, W7 = Regulator((0,40), s_bn_0, w_bn_0, dt, I, L, K, P, 1)
```

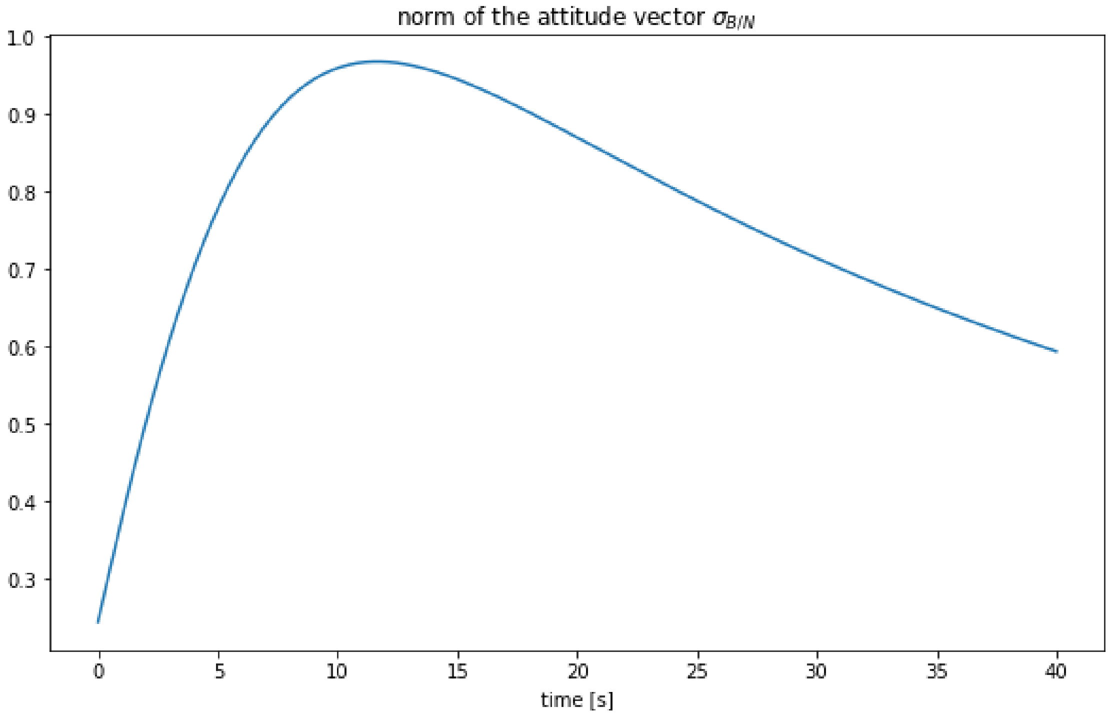




[42]:
```
print('the norm of the orientation vector at 30 seconds in to the simulation is:␣
  ↪', norm(S7[:,3000],2))
```

```
the norm of the orientation vector at 30 seconds in to the simulation is:
0.7137313745984792
```

[43]:
```
plt.figure(figsize = (10,6))
plt.plot(np.arange(0,40,dt), norm(S7,2,0))
plt.xlabel('time [s]')
plt.title('norm of the attitude vector $\sigma_{B/N}$')
plt.show()
```



3. What are the corresponding decay times T_i for each axis?

Using the formulas for the T_i we get that

$$(T_1, T_2, T_3) = 2IP^{-1}$$

[44]:
```
print('The corresponding decay times $T_1,T_2, T_3$ are : \n', np.
 ↪diag(2*I@inv(P)), 's')
```

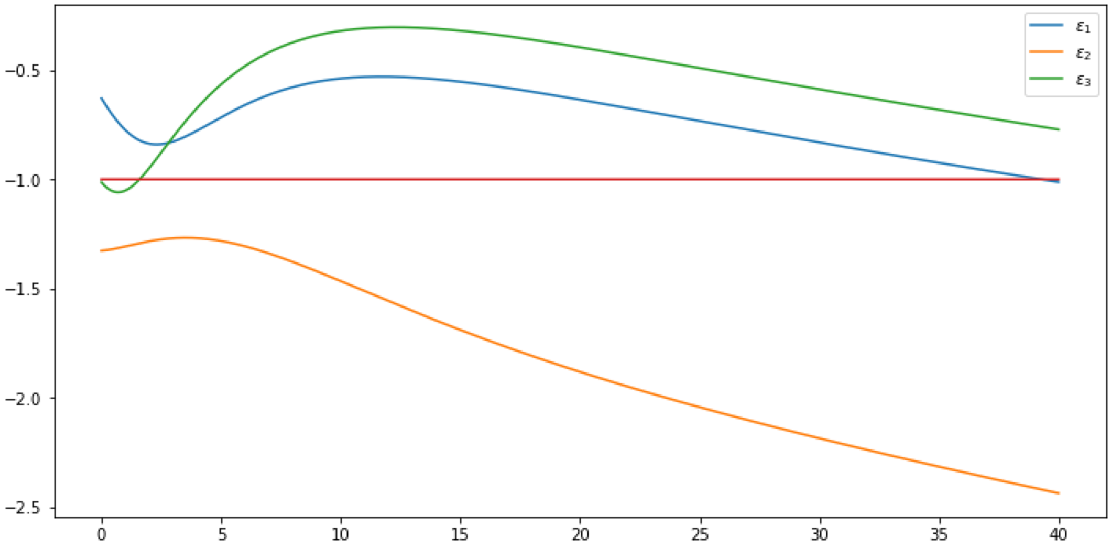
```
The corresponding decay times $T_1,T_2, T_3$ are :
 [8.94427191 7.74596669 8.        ] s
```

Let $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \epsilon_3)$ be the state error, whos components are given by:

$$\epsilon_i = \sqrt{\sigma_i^2 + \delta\omega_i^2}$$

[45]:
```
Eps = S7**2 + W7**2
eps1 = np.sqrt(Eps[0,:])
eps2 = Eps[1,:]**0.5
eps3 = Eps[2,:]**0.5
t = np.arange(0,40, dt)
plt.figure(figsize = (12,6))
plt.plot(t, np.log(eps1), label = '$\epsilon_1$')
plt.plot(t, np.log(eps2), label = '$\epsilon_2$')
plt.plot(t, np.log(eps3), label = '$\epsilon_3$')
plt.plot(t, -np.ones(len(t)))
plt.legend()
```

```
[45]: <matplotlib.legend.Legend at 0x7fd08c5653d0>
```



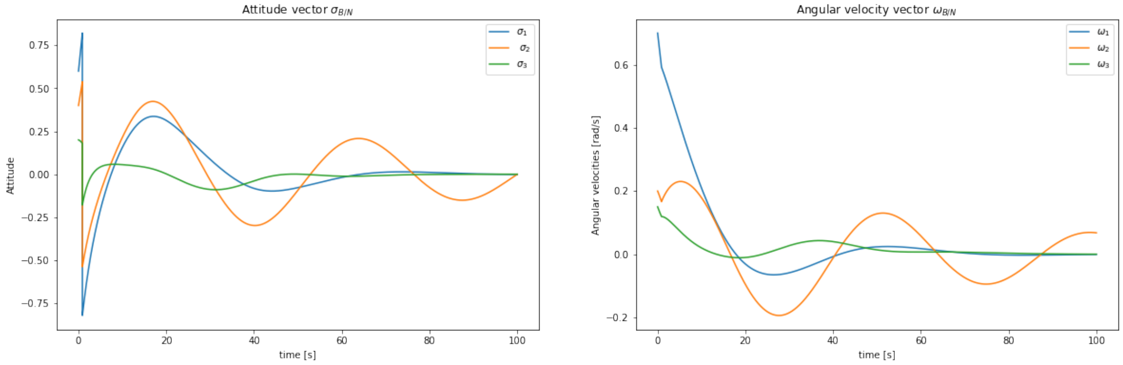
Example 8.12 (from the book)

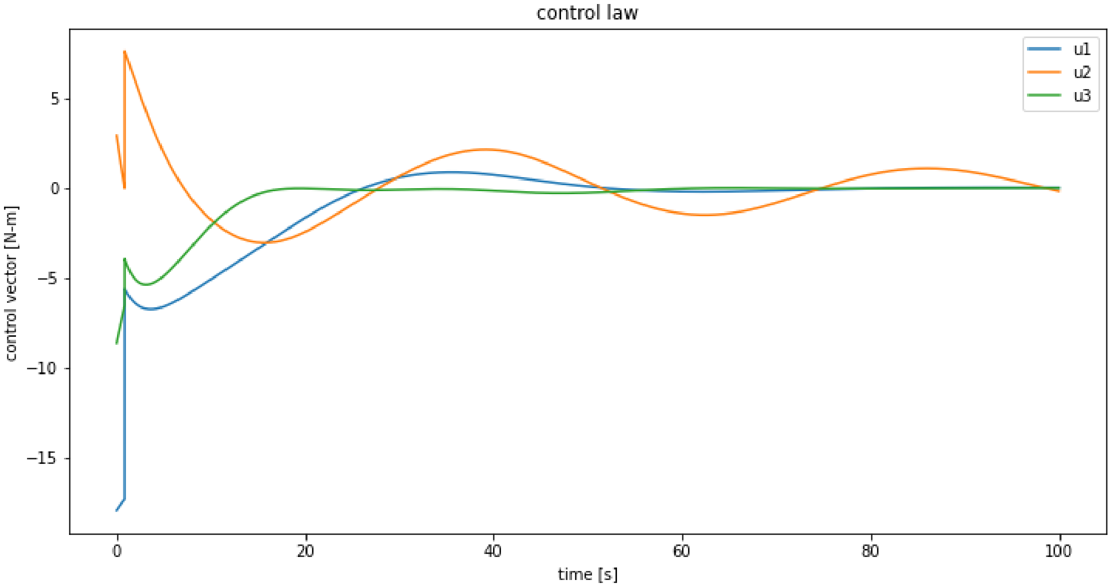
$$I_1 = 140, \;\; I_2 = 100, \;\; I_3 = 80 \quad kgm^2$$

$$\boldsymbol{\sigma}(t_0) = (0.60, 0.40, 0.20), \;\; \boldsymbol{\omega}(t_0) = (0.70, 0.20, 0.15) rad/s$$

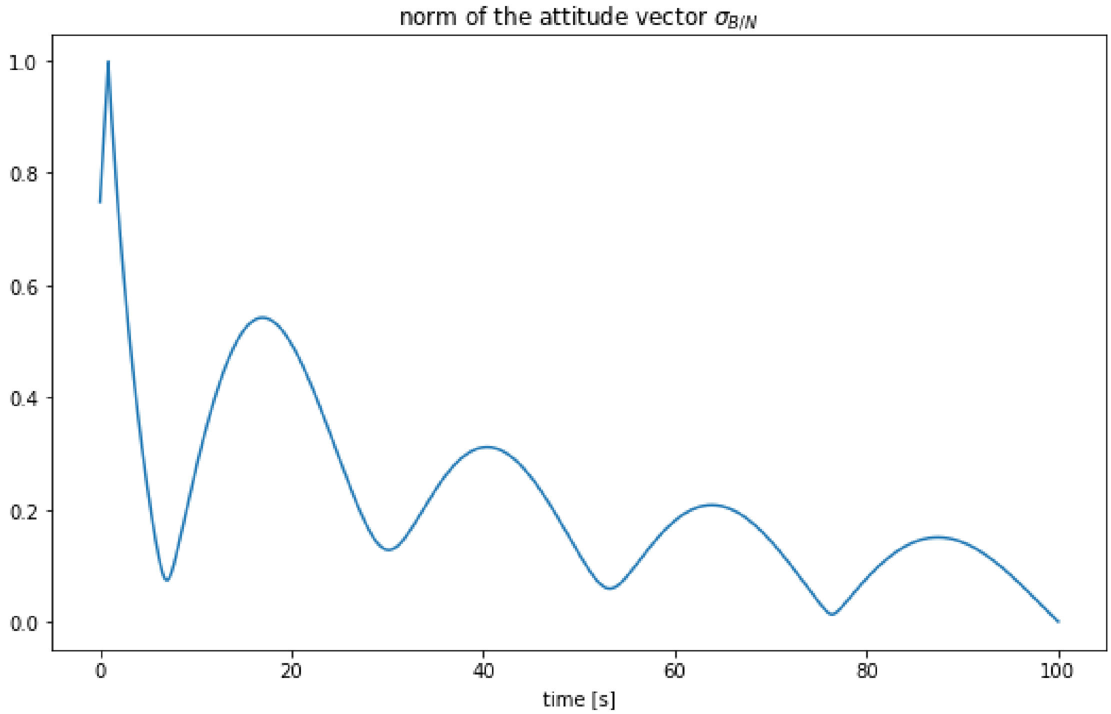
$$P = diag(18.67, 2.67, 10.67) kgm^2/s, \;\; k = 7.11 kgm^2/s^2$$

```
[46]: I = np.diag([140, 100, 80])
      s_0 = np.array([0.60, 0.40, 0.20])
      w_0 = np.array([0.70, 0.20, 0.15])
      P = np.diag([18.67, 2.67, 10.67])
      k = 7.11
      L = np.zeros(3)
      dt = 0.01
      S8, W8 = Regulator((0,100), s_0, w_0, dt, I, L, k, P, 1)
```





```
[47]: plt.figure(figsize = (10,6))
      plt.plot(np.arange(0,100,dt), norm(S8,2,0))
      plt.xlabel('time [s]')
      plt.title('norm of the attitude vector $\sigma_{B/N}$')
      plt.show()
```

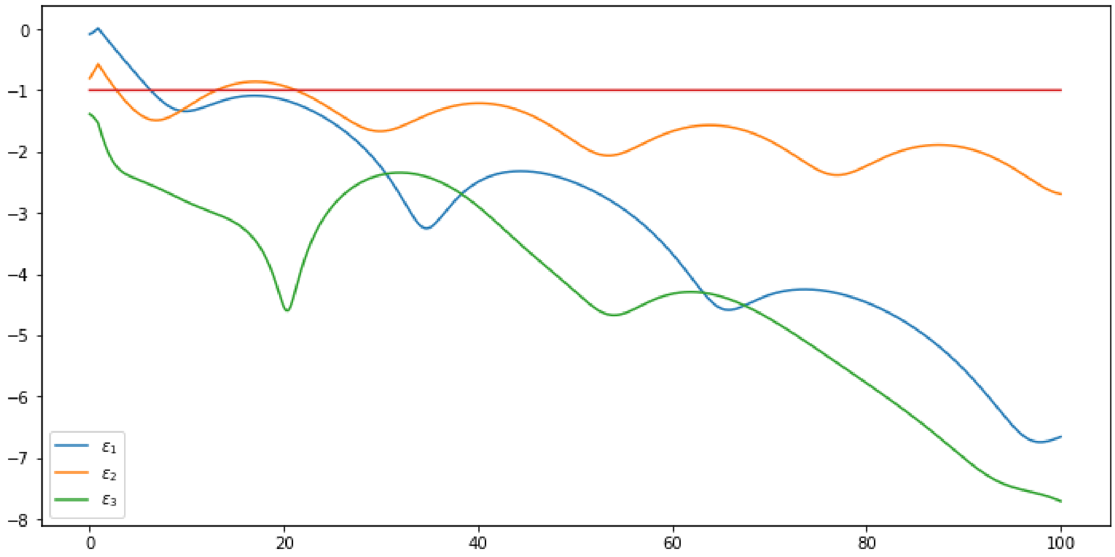


[48]:

```
Eps = S8**2 + W8**2
eps1 = np.sqrt(Eps[0,:])
eps2 = np.sqrt(Eps[1,:])
eps3 = np.sqrt(Eps[2,:])
t = np.arange(0,100, dt)
plt.figure(figsize = (12,6))
plt.plot(t, np.log(eps1), label = '$\epsilon_1$')
plt.plot(t, np.log(eps2), label = '$\epsilon_2$')
plt.plot(t, np.log(eps3), label = '$\epsilon_3$')
plt.plot(t, -np.ones(len(t)))
plt.legend()
```

[48]:

```
<matplotlib.legend.Legend at 0x7fd08b0a52e0>
```



```
[51]: type(0.05)
```

References

[1] J. Junkins and H. Schaub “Analytical Mechanics of Space Systems”. 2009. AIAA Education Series.

[2] Mukherjee, R., and Chen, D., “Asymptotic Stability Theorem for Autonomous Systems, ” Journal of Guidance, Control, and Dynamics, Vol. 16, Sept. Oct. 1993, pp. 961 963.