**[2Guidance Algorithms**

**Guidance stages:**

1. **The entry guidance stage**
2. **Parachute Descent Guidance**
3. **Powered Descent Guidance**
4. **Sky-Crane Guidance**

**The mathematical model for Mars entry**

**The guidance algorithms:**

* Numerical Predictor-Corrector (NPC)
* Analytical Predictor-Corrector (APC)
* EDL theoretical

**Numerical Predictor-Corrector (NPC) in details**

**—----------------------------------------------------------------------------------------------------------**

The stages of guidance in Mars missions can vary depending on the mission design and architecture, but a general overview is as follows:

**Entry Guidance**: This stage involves controlling the angle of attack and the bank angle of the spacecraft to achieve the desired range, cross-range, and energy management during the hypersonic flight through the atmosphere2. The entry guidance can use different methods, such as bank angle steering, direct force control, or flap-based control, to modulate the lift and drag forces and steer the spacecraft towards the target landing site2.[2]

**Parachute Descent Guidance**: This stage involves deploying a supersonic parachute to slow down the spacecraft and stabilize its descent rate3. The parachute descent guidance can use different methods, such as range trigger, constant altitude trigger, or velocity trigger, to determine the optimal time and altitude to deploy the parachute based on the current state and the desired conditions. .[72] [3]

**Powered Descent Guidance**: This stage involves separating the powered descent vehicle (PDV) from the backshell and the parachute and using thrusters to control the velocity and attitude of the PDV during the final approach to the surface4. The powered descent guidance can use different methods, such as numerical predictor-corrector, quadratic programming, or artificial intelligence, to compute the optimal thrust profile and attitude commands for the PDV to achieve the minimum fuel consumption and maximum landing accuracy4.

**Sky-Crane Guidance**: This stage involves lowering the rover or the lander from the PDV using a tether and a sky-crane mechanism and flying away from the PDV after the touchdown. The sky-crane guidance can use different methods, such as proportional-integral-derivative, linear-quadratic regulator, or adaptive control, to regulate the descent rate and lateral position of the rover or the lander relative to the PDV, and to detect the touchdown and cut the tether.

1. **The entry guidance stage** is the phase of the entry, descent, and landing (EDL) process that involves controlling the angle of attack and the bank angle of the spacecraft to achieve the desired range, cross-range, and energy management during the hypersonic flight through the atmosphere1. Different algorithms and equations can be used for entry guidance, depending on the vehicle model, the mission objectives, and the constraints. Some examples of entry guidance algorithms and equations are:

* **Bank angle steering**: This algorithm modulates the bank angle to control the lift and drag forces and steer the spacecraft towards the target landing site2. The equation for the bank angle command is:

where σ is the bank angle, CL​ and CD​ are the coefficients of lift and drag, g is the gravitational acceleration, γ is the flight path angle, V is the velocity, and R is the radius of curvature of the trajectory.

The bank angle steering algorithm uses a reference trajectory that is computed offline and stored onboard the spacecraft. The reference trajectory specifies the desired altitude, velocity, flight path angle, heading angle, and range as functions of time. The algorithm compares the current state of the spacecraft with the reference trajectory and computes the bank angle command that minimizes the error. The algorithm also performs periodic bank reversals to limit the error in the lateral direction.

* **Direct force control**: This algorithm directly controls the lift and drag forces by adjusting the angle of attack and the bank angle. The equations for the lift and drag commands are:

where L and D are the lift and drag forces, m is the vehicle mass, and V˙ is the velocity rate.

* **Flap-based control**: This algorithm uses aerodynamic flaps to control the lift and drag forces by varying the flap deflection angle4. The equations for the lift and drag coefficients are:

where CL0​ and CD0​ are the zero-lift and zero-drag coefficients, CLα​ and CDα​ are the lift and drag slope coefficients, CLδ​ and CDδ​ are the flap effectiveness coefficients, α is the angle of attack, and δ is the flap deflection angle.

**The bank angle steering algorithm was successful in guiding the Mars 2020 spacecraft to a precise landing site.**

1. **Parachute Descent Guidance:** uses Guidance algorithms (Numerical Predictor-Corrector (NPC), Analytical Predictor-Corrector (APC), EDL theoretical) which are described in more detail in the guidance algorithms section.

**In Mars 2020** The algorithm used for this stage is the **Range Trigger**

**The Range Trigger** is an algorithm that uses the Terrain Relative Navigation (TRN) estimate of the altitude to determine the optimal time to deploy the supersonic parachute. The equation for the Range Trigger is:

where td​ is the time of parachute deployment, t0​ is the time of TRN activation, hd​ is the desired altitude of parachute deployment, h0​ is the initial altitude estimate from TRN, and h˙0​ is the initial altitude rate estimate from TRN.

The Range Trigger reduces the variability in the parachute deployment conditions and enables a more precise landing.

1. **The Powered Descent Vehicle (PDV) Guidance** is the system that controls the descent and landing of the Mars 2020 Perseverance rover.

It’s an improved version of the MSL PDV guidance that uses a numerical predictor-corrector algorithm to compute the optimal thrust profile and attitude commands for the PDV. It also incorporates the TRN position and velocity estimates to correct for any errors in the entry phase.

The four main components of the Powered Descent Vehicle (PDV) Guidance are:

* **Trajectory Commander**: This component computes the optimal trajectory for the PDV to follow, based on the current state, the desired landing site, and the constraints on the vehicle dynamics and performance. It uses a quadratic programming method to solve a convex optimization problem that minimizes the fuel consumption and maximizes the landing accuracy. The optimization problem can be formulated as:

Subject to:

where x(t) is the state vector, u(t) is the control vector, f(x,u) is the system dynamics, x0​ and xf​ are the initial and final states, and u min​ and u max​ are the control limits.

* **Trajectory Control**: This component implements a numerical predictor-corrector algorithm to generate the thrust commands for the eight Mars Lander Engines (MLEs) that steer the PDV along the desired trajectory. It also incorporates the Terrain Relative Navigation (TRN) estimates of the position and velocity to correct for any errors in the entry phase. The algorithm is based on the following equation:

where h is the step size, f(x,y) is the function that defines the differential equation, and y~​n+1​ is an initial guess for yn+1​ obtained by using a predictor method, such as the forward Euler method. The algorithm iteratively updates y~​n+1​ until it converges to the desired accuracy.

* **Attitude Command**: This component computes the desired attitude of the PDV, based on the thrust commands, the aerodynamic torques, and the attitude constraints3. It uses a quaternion-based method to represent the orientation of the PDV and to avoid the singularity problem of the Euler angles. The desired attitude quaternion can be computed as:

where q0​ is the initial attitude quaternion, qΔ​ is the attitude change quaternion, and ⊗ is the quaternion multiplication operator.

* **Attitude Control**: This component implements a proportional-integral-derivative (PID) controller to generate the attitude commands for the MLEs that stabilize the PDV orientation. It uses a quaternion feedback method to compute the attitude error and the control law. The attitude error quaternion can be computed as:

where qd​ is the desired attitude quaternion, q is the current attitude quaternion, and ∗ denotes the quaternion conjugate. The control law can be computed as:

where ua​ is the attitude control vector, Kp​, Kd​, and Ki​ are the proportional, derivative, and integral gains, ω is the angular velocity vector, and qi​ is the integral of the attitude error quaternion.

The algorithm used in the Powered Descent Guidance stage for **Mars 2020** is the **numerical predictor-corrector** algorithm.

An example for the numerical predictor-corrector algorithm that is used in PDV Guidance in python code:

# Define the function f(x,y) = y'

def f(x,y):

return -y

# Define the initial conditions

x0 = 0 # initial time

y0 = 1 # initial value

h = 0.1 # step size

N = 10 # number of steps

# Initialize the arrays for storing the results

x = np.zeros(N+1) # time array

y = np.zeros(N+1) # solution array

x[0] = x0 # initial time

y[0] = y0 # initial condition

# Define the tolerance and the maximum number of iterations

tol = 1e-6 # tolerance for convergence

max\_iter = 10 # maximum number of iterations

# Implement the predictor-corrector algorithm

for n in range(N):

# Predictor step: use forward Euler to get an initial guess

y\_pred = y[n] + h\*f(x[n], y[n])

# Corrector step: use the predictor-corrector formula and iterate until convergence

iter = 0 # iteration counter

error = tol + 1 # initial error

while error > tol and iter < max\_iter:

y\_corr = y[n] + h/2\*(f(x[n], y[n]) + f(x[n+1], y\_pred))

error = abs(y\_corr - y\_pred) # compute the error

y\_pred = y\_corr # update the guess

iter += 1 # increment the iteration counter

# Store the results

x[n+1] = x[n] + h # update the time

y[n+1] = y\_corr # store the solution

1. **The Sky-Crane Guidance** stage is the final phase of the powered descent of a spacecraft to land on a planetary surface, such as Mars. In this stage, the spacecraft uses a set of thrusters to control its attitude and position, while lowering a rover or a lander on a tether. The goal is to safely and accurately deliver the payload to the desired landing site, while avoiding obstacles and hazards.

One of the algorithms used for the Sky-Crane Guidance stage is the Numerical Predictor-Corrector (NPC) guidance algorithm, which is based on solving a set of equations that describe the desired terminal conditions, such as altitude, velocity, and downrange.

The algorithm used in The Sky-Crane Guidance stage for **Mars 2020** is a proportional-integral-derivative (PID) controller.

**The proportional-integral-derivative (PID)** controller algorithm is a method that uses three terms to adjust the control output based on the error between the desired setpoint and the measured process variable. The equations used in the PID controller algorithm are:

The error equation, which calculates the difference between the setpoint and the process variable:

where e(t) is the error, SP(t) is the setpoint, and PV(t) is the process variable.

The proportional term, which is proportional to the current error:

where P is the proportional term, Kp​ is the proportional gain, and e(t) is the error.

The integral term, which is proportional to the sum of the past errors:

where I is the integral term, Ki​ is the integral gain, and e(τ) is the error at time τ.

The derivative term, which is proportional to the rate of change of the error:

where D is the derivative term, Kd​ is the derivative gain, and e(t) is the error.

The control output, which is the sum of the three terms:

where u(t) is the control output, P is the proportional term, I is the integral term, and D is the derivative term.

The PID controller algorithm can be tuned by adjusting the values of the gains Kp​, Ki​, and Kd​ to achieve the desired performance and stability of the system.

**The mathematical model for Mars entry** is based on the equations of motion of a vehicle in a rotating spherical planet atmosphere. The state vector x(t) consists of the following variables:

* r: the radial distance from the center of the planet
* θ: the latitude angle
* ϕ: the longitude angle
* V: the velocity magnitude
* γ: the flight path angle
* ψ: the heading angle
* α: the angle of attack
* β: the bank angle

The control vector u(t) consists of the following variables:

* α˙: the angle of attack rate
* β˙​: the bank angle rate

The dynamic model f(x(t),u(t),t) is a function of the state vector, the control vector, and the time, and it includes the effects of gravity, aerodynamics, and planet rotation. The dynamic model can be written as a system of ordinary differential equations (ODEs) as follows:

where D and L are the drag and lift forces, m is the vehicle mass, g is the gravitational acceleration, and ω is the planet angular velocity.

The desired terminal state xf​ consists of the following variables:

* rf​: the final radial distance from the center of the planet
* Vf​: the final velocity magnitude
* γf​: the final flight path angle
* sf​: the final downrange distance

The terminal state can be expressed as a set of algebraic equations as follows:

r(tf​)V(tf​)γ(tf​)s(tf​)​=rf​=Vf​=γf​=sf​​

where s(t) is the downrange distance, which can be computed as:

s(t)=rcos−1(sinθ0​sinθ+cosθ0​cosθcos(ϕ−ϕ0​))

where θ0​ and ϕ0​ are the initial latitude and longitude angles.

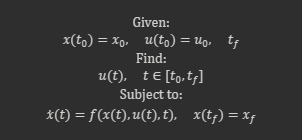
**The guidance algorithms:** is responsible for finding the optimal control vector u(t) that satisfies the terminal state equations, while minimizing some performance criteria, such as fuel consumption, peak heating, or peak deceleration. Different guidance algorithms use different methods to solve this optimal control problem, such as numerical integration, analytical solutions, or semi-analytical solutions. Some examples of guidance algorithms are:

• **Numerical Predictor-Corrector (NPC) guidance**: This algorithm uses a numerical integration method to predict the future trajectory of the vehicle, and then corrects the control commands to steer the vehicle towards the target1.

• **Analytical Predictor-Corrector (APC) guidance**: This algorithm uses analytical solutions to the trajectory equations, which can reduce the computational burden and improve the robustness of the guidance system12.

• **EDL theoretical guidance**: This algorithm uses semi-analytical solutions to the trajectory equations, which can account for some nonlinear effects and uncertainties in the model parameters

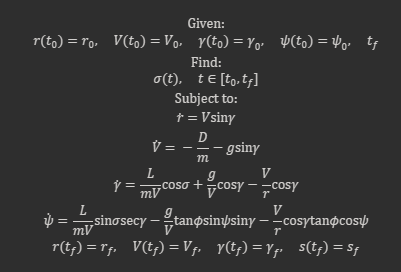
**Numerical Predictor-Corrector (NPC) guidance**: This algorithm uses a numerical integration method to predict the future trajectory of the vehicle, and then corrects the control commands to steer the vehicle towards the target1. The mathematical model of the NPC guidance algorithm can be expressed as follows:



where x(t) is the state vector, u(t) is the control vector, f(x(t),u(t),t) is the dynamic model, and xf​ is the desired terminal state. The NPC guidance algorithm iteratively updates the control vector u(t) until the predicted state vector x(tf​) matches the desired terminal state xf​ within a specified tolerance1.

Some of the advantages of the NPC guidance algorithm are that it can handle nonlinear dynamics, multiple constraints, and uncertainties in the model parameters. However, it also has some drawbacks, such as high computational cost, sensitivity to initial guesses, and possible convergence issues.

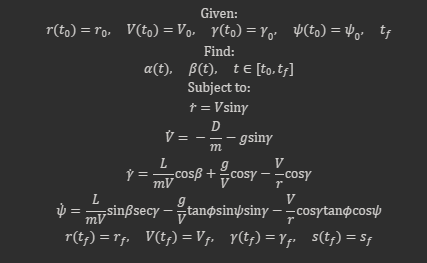
**Analytical Predictor-Corrector (APC) guidance**: This algorithm uses analytical solutions to the trajectory equations, which can reduce the computational burden and improve the robustness of the guidance system. The analytical solutions are based on some simplifying assumptions, such as constant drag coefficient, constant lift-to-drag ratio, and exponential atmosphere mode. The mathematical model of the APC guidance algorithm can be expressed as follows:



where r is the radial distance from the center of the planet, V is the velocity magnitude, γ is the flight path angle, ψ is the heading angle, σ is the bank angle, D and L are the drag and lift forces, m is the vehicle mass, g is the gravitational acceleration, s is the downrange distance, and the subscript f denotes the final values. The APC guidance algorithm consists of two phases: a predictor phase and a corrector phase. In the predictor phase, the analytical solutions are used to estimate the final state vector and the range error. In the corrector phase, the bank angle command is updated by a feedback law to reduce the range error.

Some of the advantages of the APC guidance algorithm are that it has low computational requirements, fast convergence, and good robustness to uncertainties. However, it also has some limitations, such as the need for accurate aerodynamic and atmospheric models, the sensitivity to initial conditions, and the possible violation of physical constraints.

**EDL theoretical guidance**: This algorithm uses semi-analytical solutions to the trajectory equations, which can account for some nonlinear effects and uncertainties in the model parameters. The semi-analytical solutions are based on some approximations, such as linearizing the drag and lift coefficients, neglecting the Coriolis and centrifugal forces, and using a polynomial atmosphere model. The mathematical model of the EDL theoretical guidance algorithm can be expressed as follows:



where α is the angle of attack, β is the bank angle, and the other variables are the same as before. The EDL theoretical guidance algorithm also consists of two phases: a predictor phase and a corrector phase. In the predictor phase, the semi-analytical solutions are used to estimate the final state vector and the range error. In the corrector phase, the angle of attack and bank angle commands are updated by feedback laws to reduce the range error and the crossrange error.

Some of the advantages of the EDL theoretical guidance algorithm are that it can handle nonlinear aerodynamics, variable lift-to-drag ratio, and atmospheric uncertainties. However, it also has some drawbacks, such as the need for accurate initial guesses, the sensitivity to vehicle mass, and the possible violation of physical constraints.

**the guidance algorithm used for the Mars 2020 mission** was **the Numerical Predictor-Corrector (NPC) guidance**, which is based on solving a set of equations that describe the desired terminal conditions, such as altitude, velocity, and downrange12. The NPC guidance algorithm uses a numerical integration method to predict the future trajectory of the vehicle, and then corrects the control commands to steer the vehicle towards the target1.

**TRN and Range Trigger was the new feathers added in mars 2020 mission.**

Range Trigger improves landing accuracy Terrain Relative Navigation enables landing in rougher terrain.

**The numerical predictor-corrector (NPC)** in details:

The numerical predictor-corrector (NPC) guidance algorithm is a method for determining the optimal control inputs for a vehicle during an atmospheric entry or aeroassist maneuver. The algorithm works by predicting the future state of the vehicle based on the current state and control inputs, and then correcting the control inputs based on the difference between the predicted and desired end states. The algorithm iterates until the error is minimized or a specified tolerance is reached.

The NPC guidance algorithm can be formulated as follows:

1. Initialize the control inputs, such as angle of attack and bank angle, and set the iteration counter to zero.
2. Integrate the equations of motion of the vehicle from the current state to the end of the entry phase, using the control inputs as inputs. This is the prediction step.
3. Compare the predicted end state with the desired end state, such as the target energy and inclination, and compute the error vector. This is the correction step.
4. Update the control inputs using a feedback law that depends on the error vector and the sensitivity matrix, which relates the changes in the control inputs to the changes in the end state.
5. Increment the iteration counter and check if the error is below a specified tolerance or the maximum number of iterations is reached. If not, go back to step 2. If yes, output the final control inputs.

The feedback law can be chosen based on different criteria, such as minimizing the control effort, the error, or a combination of both. One common choice is the Newton-Raphson method, which uses the inverse of the sensitivity matrix to update the control inputs. Another choice is the steepest descent method, which uses a scalar step size to update the control inputs in the direction of the negative gradient of the error.

The equations of motion of the vehicle depend on the aerodynamic forces and moments, the gravity, and the propulsion (if any). The aerodynamic forces and moments are usually expressed as functions of the aerodynamic coefficients, which depend on the vehicle geometry, the Mach number, the angle of attack, and the sideslip angle. The gravity is usually modeled as a central force that varies with the altitude and latitude. The propulsion can be modeled as a constant or variable thrust that acts along a specified direction.

The NPC guidance algorithm can be applied to various entry scenarios, such as ballistic, lifting, or skip entry, and to different planetary bodies, such as Earth, Mars, or Titan. The algorithm can also be extended to handle multiple phases, such as aero-gravity assist or aero-breaking, by concatenating the solutions of each phase.

**The equations used in the numerical predictor-corrector guidance algorithm are**:

* **The equations of motion** of the vehicle, which describe how the state vector changes over time as a function of the control inputs and the external forces and moments. The state vector typically includes the position, velocity, attitude, and mass of the vehicle. The control inputs are the angle of attack, sideslip angle, and bank angle for a lifting vehicle, or the thrust magnitude and direction for a propulsion vehicle. The external forces and moments are the aerodynamic, gravity, and propulsion forces and moments. The equations of motion can be written as:

where x is the state vector, u is the control input vector, and f is a nonlinear function that depends on the vehicle dynamics and the environment.

* **The aerodynamic coefficients**, which relate the aerodynamic forces and moments to the dynamic pressure and the reference area of the vehicle. The aerodynamic coefficients depend on the vehicle geometry, the Mach number, the angle of attack, and the sideslip angle. The aerodynamic coefficients can be written as:

CD​=CD​ (M, α, β)

CL​=CL​ (M, α, β)

Cm​=Cm​ (M, α, β)

Cn​=Cn ​(M, α, β)

Cl​=Cl​ (M, α, β)

where CD​ is the drag coefficient, CL​ is the lift coefficient, Cm​ is the pitching moment coefficient, Cn​ is the yawing moment coefficient, Cl​ is the rolling moment coefficient, M is the Mach number, α is the angle of attack, and β is the sideslip angle.

* **The sensitivity matrix**, which relates the changes in the control inputs to the changes in the end state. The sensitivity matrix is computed by linearizing the equations of motion around the nominal trajectory and integrating the resulting state transition matrix. The sensitivity matrix can be written as:

where S is the sensitivity matrix, xf​ is the end state vector, and u is the control input vector.

* **The feedback law**, which updates the control inputs based on the error vector and the sensitivity matrix. The feedback law can be chosen based on different criteria, such as minimizing the control effort, the error, or a combination of both. One common choice is the Newton-Raphson method, which uses the inverse of the sensitivity matrix to update the control inputs. Another choice is the steepest descent method, which uses a scalar step size to update the control inputs in the direction of the negative gradient of the error. The feedback law can be written as:

or

where uk​ is the control input vector at the k-th iteration, K is a tuning parameter, S−1 is the inverse of the sensitivity matrix, e is the error vector, and λ is the step size.

“**NASA AAS 21-201 paper have lots of data about NPC algorithms and was one of my important ref I will attach it in the new ref folder on drive**

**Here is a sample pseudocode for the NPC guidance algorithm:**

# Define the initial state, the desired end state, the control inputs, and the tolerance

x0 = [position, velocity, attitude, etc.] # initial state vector

xf = [energy, inclination, etc.] # desired end state vector

u = [alpha, beta, etc.] # control input vector

tol = 1e-6 # error tolerance

# Define the equations of motion and the aerodynamic coefficients

def eom(x, u):

# Calculate the aerodynamic forces and moments

F\_aero = ...

M\_aero = ...

# Calculate the gravity force

F\_grav = ...

# Calculate the propulsion force and moment (if any)

F\_prop = ...

M\_prop = ...

# Calculate the state derivatives

x\_dot = ...

return x\_dot

def aero\_coeff(M, alpha, beta):

# Calculate the aerodynamic coefficients based on the vehicle geometry and the Mach number, angle of attack, and sideslip angle

C\_D = ...

C\_L = ...

C\_m = ...

C\_n = ...

C\_l = ...

return [C\_D, C\_L, C\_m, C\_n, C\_l]

# Initialize the iteration counter and the error

iter = 0

error = 1

# Loop until the error is below the tolerance or the maximum number of iterations is reached

while error > tol and iter < max\_iter:

# Predict the end state by integrating the equations of motion using the control inputs

x\_pred = integrate(eom, x0, u)

# Correct the control inputs by comparing the predicted end state with the desired end state and using a feedback law

error = x\_pred - xf # error vector

S = sensitivity(eom, x0, u) # sensitivity matrix

u = u - inv(S) \* error # Newton-Raphson method

# u = u - step\_size \* error # steepest descent method

# Increment the iteration counter

iter = iter + 1

# Output the final control inputs

print(u)