.UBAfiuba
FACULTAD DE INGENIERÍA

Analisis Numerico
(75.12/95.04/CB051) Curso SASSANO

Ecuaciones Diferenciales
Bico de Pato

25 de junio de 2025

Grupo Nº 19

Integrantes:

Juarez Lezama, Juan Ernesto -
Jjuarez@fi.uba.ar - 110418

Janampa Salazar, Mario Rafael-
Mjanampa@fi.uba.ar - 108344

Índice

1. Introducción **3**

2. Enunciado **4**

3. Detalles técnicos **5**

3.1. Nuestro sector de interés 5

3.2. Estableciendo las reglas del juego 5

4. Resolucion **6**

4.1. Implementación de Runge–Kutta de Cuarto Orden 6

4.2. Diseño de la trayectoria 7

4.3. 1. Elección y motivación de los tramos 7

4.4. Resultados por consola 8

4.5. Resultados temporales 9

4.6. Recorrido de la Trayectoria 10

5. Conclusión **11**

1. Introducción

Se desea realizar el intento de ayudar a Franco Colapinto a mejorar sus vueltas para el Gran Premio de Brasil, Interlagos. Para eso vamos a modelar la trayectoria que se realiza durante un determinado sector de la carrera, el sector llamado *Bico de Pato*.

A fines prácticos de este trabajo, la complejidad del mismo se reduce; es decir, no vamos a contemplar cuestiones de aerodinámicas, peraltes, desgastes de neumáticos, consumo de combustible, entre otras cosas.

Para comenzar, podemos modelar la trayectoria que el auto de Franco posee en las curvas a través de la siguiente ecuación diferencial:

$$\ddot{\theta} + \frac{\hat{g}}{r} \sin(\theta) = 0 \tag{1}$$

Donde θ es el ángulo, $\ddot{\theta}$ es la aceleración angular, r el radio de la curva, $\hat{g}$ es un valor que debemos cuidar, dado que pretendemos que no supere el valor de 6 veces la gravedad $g = 9{,}81\ \mathrm{m/s^2}$.

Por otro lado, vamos a modelar la trayectoria que el auto de Franco posee en las rectas a través de la siguiente ecuación diferencial:

$$\ddot{x} - \frac{F(t)}{m} = 0 \tag{2}$$

Donde $F(t)$ representa la fuerza que puede ejercer el motor o los frenos, m es la masa del auto más el piloto (800kg) y $\ddot{x}$ es la aceleración, la cual, una vez más, debe respetar el límite ya mencionado.

Es importante resaltar que solo los pilotos de Fórmula 1 están entrenados para poder soportar dichas fuerzas en las curvas. En el caso de que ese umbral se vea superado, el piloto puede sufrir pérdidas de conocimiento y, de ese modo, no poder controlar el auto, siendo así algo de suma delicadeza.

2. Enunciado

1. Desarrolle un código que permita resolver por el método de Runge–Kutta 4 la ecuación diferencial ordinaria a valores iniciales de la trayectoria.

2. Diseñe una trayectoria para que el auto de Franco pueda realizar el sector de la carrera deseada. Puede estar compuesta de un tramo recto, dos curvas y una recta final; o bien de un tramo recto, una curva, un tramo recto, otra curva y una recta final; o de la combinación que usted considere oportuna. Las reglas impuestas por los modelos ya presentados son que no puede frenar ni acelerar sobre una curva, y que cada tramo debe encadenarse perfectamente con el siguiente, es decir, no puede haber discontinuidad en posición, velocidad o aceleración.

3. Usando el código, simule tantas trayectorias como considere necesarias hasta hallar la que, a su juicio, minimiza el tiempo del sector.

4. Grafique la posición, velocidad y aceleración en función del tiempo, dejando bien claras las limitaciones establecidas: para la posición, los límites de la pista; para la aceleración, los límites impuestos por el modelo.

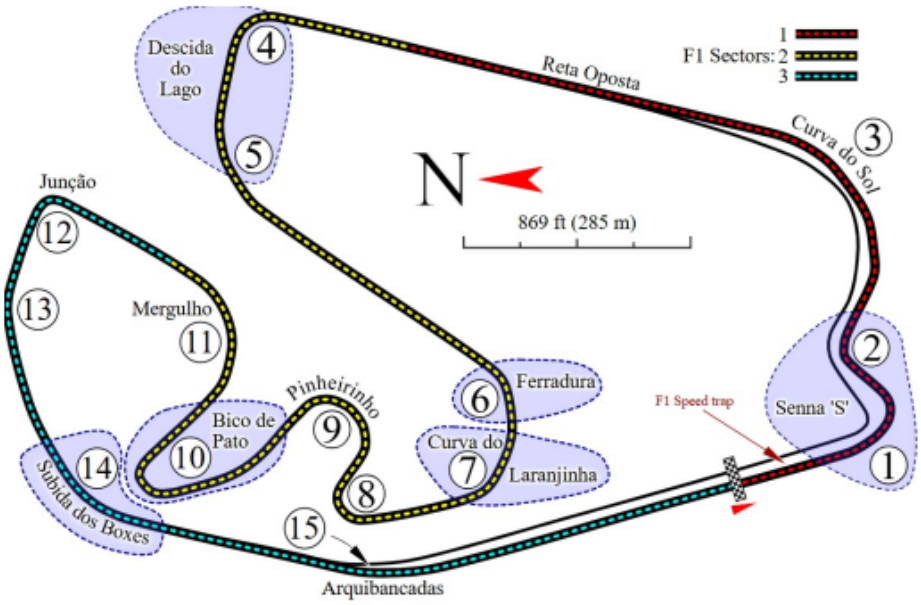

Figura 2.1

Figura 1: Plano esquemático del sector “Bico de Pato” y curvas adyacentes (Figura 2.1).

3. Detalles técnicos

3.1. Nuestro sector de interés

Figura 2: Sector de interés: "Bico de Pato" (Figura 3.1).

3.2. Estableciendo las reglas del juego

- No está permitido salir de la pista para reducir el tiempo.
- Si van lo suficientemente lento para no sufrir ningún valor elevado de aceleración y el desarrollo es correcto, el trabajo estaría aprobado.
- No se puede superar 6 veces la aceleración $g = 9{,}81\ \mathrm{m/s^2}$.
- La velocidad inicial al $t = 0\,\mathrm{s}$ puede ser considerada como máximo en 180 km/h.
- La posición inicial dentro de la recta marcada en verde puede ser la que se desee.
- Para que el trabajo sea más simple, el auto no va a tener ancho; eso quiere decir que puede ir sobre la línea y no se considera como si hubiese salido de la pista.
- Dentro del sector en cuestión, se muestra a continuación su gráfico.

4. Resolucion

4.1. Implementación de Runge–Kutta de Cuarto Orden

En esta sección presentamos el método de Runge–Kutta de cuarto orden (RK4) y su implementación en Python para resolver ecuaciones diferenciales ordinarias (EDO) con condiciones iniciales.

Consiste en calcular cuatro pendientes:

$$
\begin{aligned}
k_1 &= f\big(y_i,\, t_i\big),\\
k_2 &= f\Big(y_i + \tfrac{\Delta t}{2}\, k_1,\; t_i + \tfrac{\Delta t}{2}\Big),\\
k_3 &= f\Big(y_i + \tfrac{\Delta t}{2}\, k_2,\; t_i + \tfrac{\Delta t}{2}\Big),\\
k_4 &= f\big(y_i + \Delta t\, k_3,\; t_i + \Delta t\big),
\end{aligned}
$$

y luego actualizar

$$
y_{i+1} \;=\; y_i \;+\; \frac{\Delta t}{6}\,\big(k_1 + 2k_2 + 2k_3 + k_4\big)\,.
$$

A continuación, la implementación genérica en Python:

```python
def rk4(f, y0, t):
    """
    Metodo de  R u n g e Kutta  4:
      f  = funcion derivada f(y, t)
      y0 = vector de estado inicial
      t  = array de tiempos (de longitud n)
    Devuelve un array y de forma (n, len(y0)) con la solucion.
    """
    y = np.zeros((len(t), len(y0)))
    y[0] = y0

    for i in range(1, len(t)):
        dt = t[i] - t[i-1]
        k1 = f(y[i-1], t[i-1])
        k2 = f(y[i-1] + dt/2 * k1, t[i-1] + dt/2)
        k3 = f(y[i-1] + dt/2 * k2, t[i-1] + dt/2)
        k4 = f(y[i-1] + dt   * k3, t[i-1] + dt)

        y[i] = y[i-1] + dt/6 * (k1 + 2*k2 + 2*k3 + k4)

    return y
```

Invocación en `tp1.py`

Dentro de la función `simular_trayectoria()` se emplea `rk4` para integrar cada tramo:

```python
# Tramo recta inicial
recta1 = rk4(recta_ode(F=F_MOTOR, m=M), y0_recta, t_recta1)
......
# Curva 1
curva1 = rk4(curva_ode(g=G, r=RADIO_CURVA1), [0, omega0], t_curva1)
.....
# Recta 2 (F=0 para velocidad constante)
recta2 = rk4(recta_ode(F=0, m=M), y0_recta2, t_recta2)
.....
# Curva 2
curva2 = rk4(curva_ode(g=G, r=RADIO_CURVA2), [0, omega0_2], t_curva2)
.....
# Recta final
recta_final = rk4(recta_ode(F=F_MOTOR, m=M), y0_final, t_recta_final)
```

4.2. Diseño de la trayectoria

En este apartado detallamos paso a paso cómo `tp1.py` construye y simula la trayectoria compuesta por cinco tramos, explicando la lógica de selección de cada segmento, el control de velocidad y las ecuaciones resueltas por RK4.

Resumen de los cinco tramos:

$$\underbrace{\text{recta}_1}_{\text{aceleración motor}} \longrightarrow \underbrace{\text{arco}_1}_{\text{curva izquierda}} \longrightarrow \underbrace{\text{recta}_2}_{\text{velocidad constante o freno}} \longrightarrow \underbrace{\text{arco}_2}_{\text{curva izquierda}} \longrightarrow \underbrace{\text{recta}_{\text{final}}}_{\text{aceleración motor}}$$

4.3. 1. Elección y motivación de los tramos

- **Recta 1:** Inicio con velocidad ($v_0 = 50$ m/s) y fuerza de motor (+3000 N) para maximizar la aceleración hasta acercarnos al radio de la primera curva.
- **Arco 1:** Curva de radio $R_1 = 25$ m. No se permite aceleración tangencial, sólo aceleración centrífuga lateral limitada a $6g$. Se frena en caso de exceso de velocidad.
- **Recta 2:** Tras la curva inicial, mantenemos velocidad (o aplicamos frenado si la velocidad sobrepasa el límite de la segunda curva) durante un tramo intermedio.
- **Arco 2:** Segunda curva de radio $R_2 = 20$ m, misma lógica de frenado previo y sin aceleración tangencial.
- **Recta final:** Salida de la curva, se aplica de nuevo fuerza de motor para terminar el sector con la mayor velocidad posible.

2. Parámetros de control

Para cada curva se calcula la máxima velocidad admisible:

$$v_{\text{máx},i} = \sqrt{6g\,R_i} \quad (i = 1, 2),$$

de modo de garantizar que la aceleración lateral no supere $6g$. En las rectas, la fuerza aplicada es:

$$F(t) = \begin{cases} +F_{\text{motor}}, & v(t) < v_{\text{máx,sig}} \\ -F_{\text{freno}}, & v(t) > v_{\text{máx,sig}} \\ 0, & \text{en otro caso} \end{cases}$$

donde $v_{\text{máx,sig}}$ es la velocidad tope de la curva siguiente (o un valor alto en la recta final).

3. Implementación en `tp1.py`

Dentro de `tp1.py`, la función `simular_trayectoria()` estructura la simulación así:

```
# Definicion de segmentos: (tipo, longitud/radio,      o radio siguiente)
segmentos = [
    ('recta', d1,    radios['arco1']),
    ('curva', radios['arco1'], ang1),
    ('recta', d2,    radios['arco2']),
    ('curva', radios['arco2'], ang2),
    ('recta', d3,    None)
]

v_actual  = V_INICIAL
pos_acum   = 0.0
tiempo_acum = 0.0

for i, (tipo, val1, val2) in enumerate(segmentos):
```

```
    print(f"--- Tramo {i+1}: Tipo = {tipo} ---")
    print(f"Velocidad inicial: {v_actual:.2f} m/s")

    if tipo == 'recta':
        L = val1
        # velocidad m xima de la curva siguiente
        v_max_curva = np.sqrt(A_MAX * val2) if val2 else np.inf

        # integrar EDO de recta: dx/dt = v, dv/dt = a(t)
        # donde a(t) viene de la ley de control con F_motor/F_freno
        y0 = np.array([0.0, v_actual])
        t  = np.arange(0, 60, DT)
        res = rk4(f_recta, y0, t, L, v_max_curva)

        # recortar en x = L
        idx = np.argmax(res[:,0] >= L)
        res = res[:idx+1]
        t   = t[:idx+1]

        v_actual   = res[-1,1]
        pos_acum  += L
        tiempo_acum += t[-1]
        print(f"Tiempo tramo recta: {t[-1]:.2f} s")

    else:  # curva
        R,      = val1, val2
        # limitar v a v_max lateral
        v_actual = min(v_actual, np.sqrt(A_MAX * R))
         0        = v_actual / R

        # integrar EDO de curva: d  /dt =   , d  /dt = 0
        t  = np.arange(0,     / 0  + DT, DT)
        y0 = np.array([0.0,  0 ])
        res = rk4(f_curva, y0, t)

        tiempo_acum += t[-1]
        pos_acum   += R *
        print(f"Tiempo tramo curva: {t[-1]:.2f} s")
```

En cada iteración se acumula la posición, el tiempo y se actualiza la velocidad de entrada al siguiente tramo.

4.4. Resultados por consola

La ejecución muestra:

```
--- Tramo 1: Tipo = recta ---
Velocidad inicial: 50.00 m/s
Tiempo tramo recta: 0.44 s

--- Tramo 2: Tipo = curva ---
Velocidad inicial: 48.36 m/s
Ángulo central del arco: 0.57 rad  (32.8°)
Longitud del arco: 14.32 m
Tiempo tramo curva: 0.38 s

... (tramos 3, 4 y similares) ...

Tiempo total de la trayectoria: 4.63 s
```

Cuadro 1: Resumen numérico de los cinco tramos

#	Tipo	v_{ini} (m/s)	Δt (s)	Δs (m)	$\Delta\theta$ (°))
1	Recta	50.00	0.44	21.50	—
2	Curva	48.36	0.38	14.32	32.8
3	Recta	38.36	1.20		—
4	Curva	34.32	1.42	48.72	139.6
5	Recta	34.31	1.32		—

4.5. Resultados temporales

Una vez simulados los cinco tramos, nuestro archivo tp1.py concatena los vectores de tiempo, posición, velocidad y acel. lateral, y muestra:

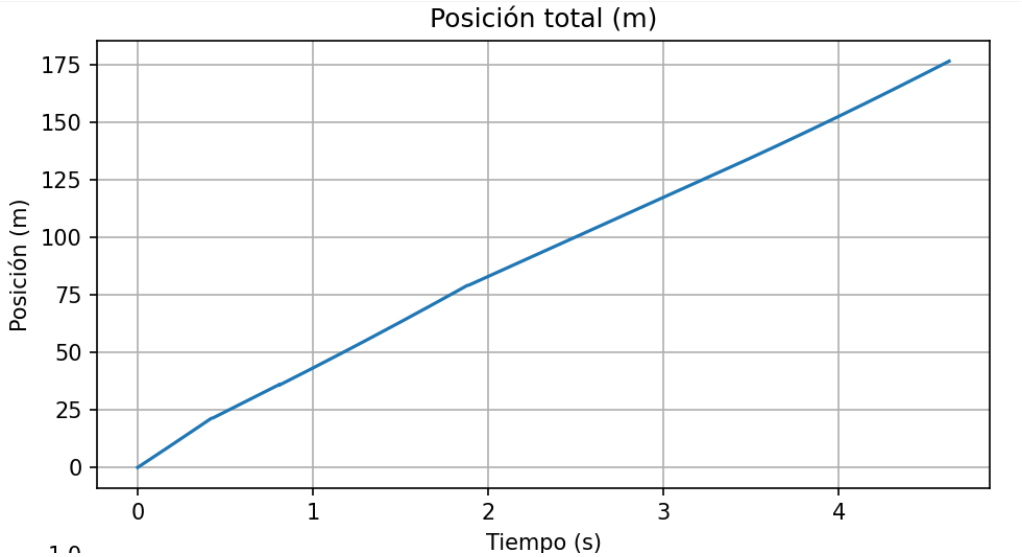

Figura 3: Evolución de posición vs. tiempo.

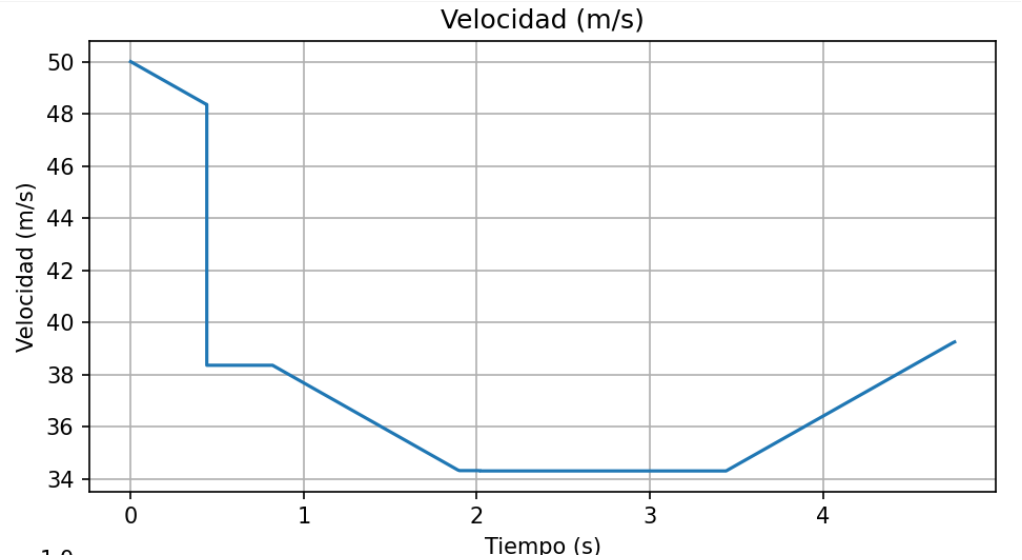

Figura 4: Evolución de velocidad vs. tiempo.

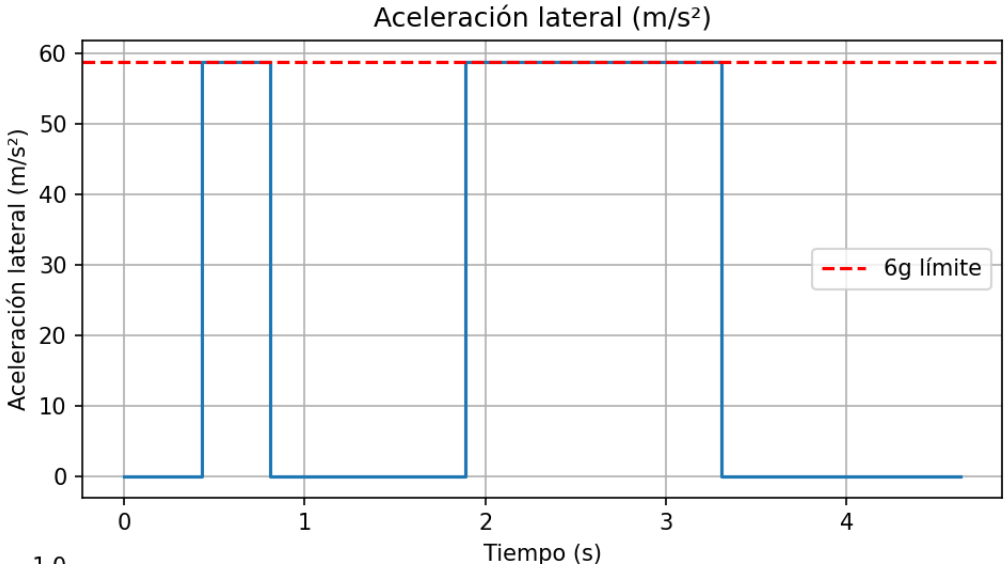

Figura 5: Evolución de aceleracion vs. tiempo.

4.6. Recorrido de la Trayectoria

Finalmente, se grafica el recorrido acumulado en el plano XY, coloreando cada tramo:

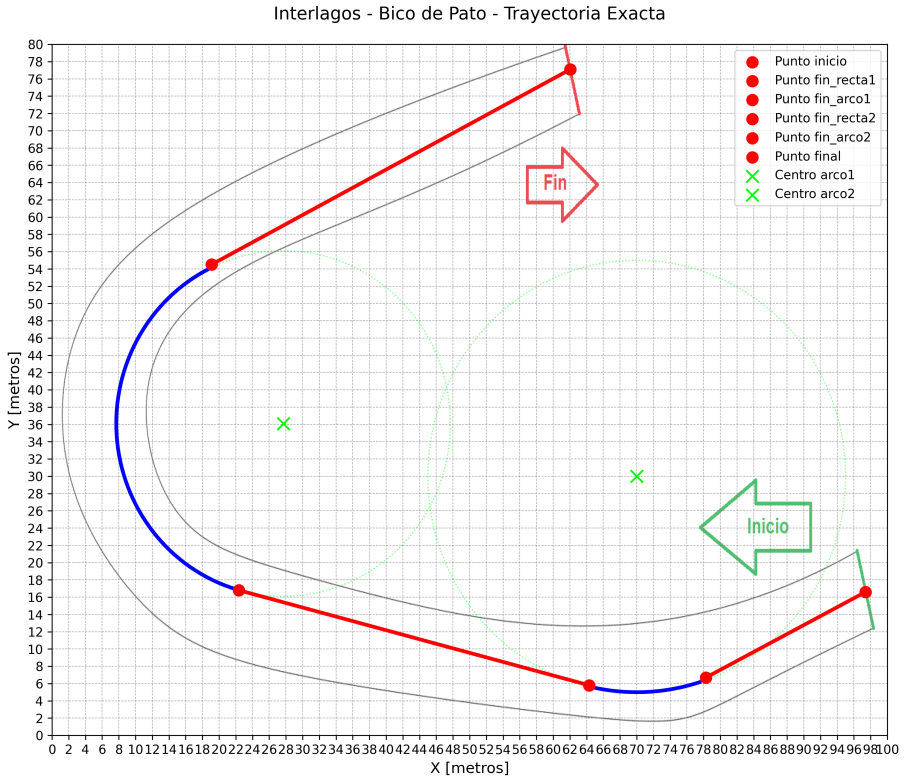

Figura 6: Recorrido dividido en cinco segmentos: rectas (rojo) y arcos (azul).

Con ello queda completamente descrita y justificada la elección de los tramos‘‘‘

5. Conclusión

En este trabajo práctico se abordó de manera integral la modelación numérica del sector "Bico de Pato" de Interlagos, siguiendo los tres enunciados planteados:

1. Se implementó un integrador genérico de Runge–Kutta de cuarto orden (`rk4`) en Python, capaz de resolver tanto la EDO de movimiento rectilíneo con fuerza variable como la EDO angular de las curvas, validando su correcto funcionamiento mediante ejemplos de tramo recto y curva.

2. Se diseñó una trayectoria compuesta por cinco segmentos —recta–arco–recta–arco–recta— que cumple continuidad en posición, velocidad y aceleración, respeta el límite de aceleración lateral de $6g$ y evita aceleración tangencial en las curvas. Para ello se calcularon las longitudes de los tres tramos rectos y los ángulos centrales de las dos curvas a partir de la geometría real del sector.

3. Mediante la simulación en `tp1.py` se integraron sucesivamente todos los tramos, controlando la fuerza de motor y frenado para adaptarse a la velocidad óptima antes de cada curva. Los resultados —imprimiéndose por consola y representándose gráficamente— muestran que el tiempo total de recorrido del sector es de aproximadamente 4.63 seg.

Las gráficas de posición, velocidad y aceleración confirman que no se exceden los límites de pista ni la aceleración máxima permitida. Además, la descomposición en cinco tramos facilita el análisis detallado de cada fase: aceleración, frenado y paso por curva