Preamble

```
In [ ]: # @title Preamble
import numpy as np
import math
import plotly.graph_objects as go

import plotly.io as pio
pio.renderers.default = "notebook_connected"
```

Functions

```
In [ ]: # System of equations
def f(trap, blood):

    # constants
    a = 4
    b = 0.05
    c = 3.8
    d = 0.11

    # FODEs
    dTdt = -a*(1/WEIGHT)*trap + b*BODYFAT*blood
    dBdt = c*(1/WEIGHT)*trap - 0.11*BODYFAT*blood
    dMdt = ((1/WEIGHT) - BODYFAT*(1/WEIGHT)) * c * trap - (BODYFAT-(BODYFAT ** 2))* d * blood

    return dTdt, dBdt, dMdt

def runge_kutta_4(f, trap0, blood0, muscle0, t0, t_end, dt):

    # Number of steps
    n = math.ceil((t_end - t0) / dt)
    # Fill arrays of n+1 zeroes
    t, trap, blood, muscle = np.zeros(n+1), np.zeros(n+1), np.zeros(n+1), np.zeros(n+1)
    # Initial conditions
    trap[0], blood[0], muscle[0], t[0] = trap0 , blood0, muscle0, t0
```

```
    # Runge-Kutta 4th order integration
    for i in range(n):
        k1T, k1B, k1M = f(
            trap[i],
            blood[i]
        )
        k2T, k2B, k2M = f(
            trap[i] + 0.5*dt*k1T,
            blood[i] + 0.5*dt*k1B
        )
        k3T, k3B, k3M = f(
            trap[i] + 0.5*dt*k2T,
            blood[i] + 0.5*dt*k2B
        )
        k4T, k4B, k4M = f(
            trap[i] + dt*k3T,
            blood[i] + dt*k3B
        )

        trap[i+1] = trap[i] + (dt/6)*(k1T + 2*k2T + 2*k3T + k4T)
        blood[i+1] = blood[i] + (dt/6)*(k1B + 2*k2B + 2*k3B + k4B)
        muscle[i+1] = muscle[i] + (dt/6)*(k1M + 2*k2M + 2*k3M + k4M)
        t[i+1] = t[i] + dt

    return t, trap, blood, muscle

# Plot the results
def plot(t, trap, blood, muscle, maximum_allowable, t_end):
    fig = go.Figure()
    fig.add_traces([
        go.Scatter(x=t, y=trap, mode='lines', marker = {'color' : 'red'}, name="Trapezius"),
        go.Scatter(x=t, y=blood, mode='lines', marker = {'color' : 'blue'}, name="Bloodstream"),
        go.Scatter(x=t, y=muscle, mode='lines', marker = {'color' : 'green'}, name="Other muscles"),
        go.Scatter(x=[i for i in range(0, t_end+1, 1)], y=[maximum_allowable for _ in range(0, t_end+1, 1)], line_dash ='dash', marker = {'
    ])
    fig.update_layout(
        title_text='Testosterone presence in body over time',
        xaxis_title='Time (hours)',
        yaxis_title='Concentration (mcg/L)',
```

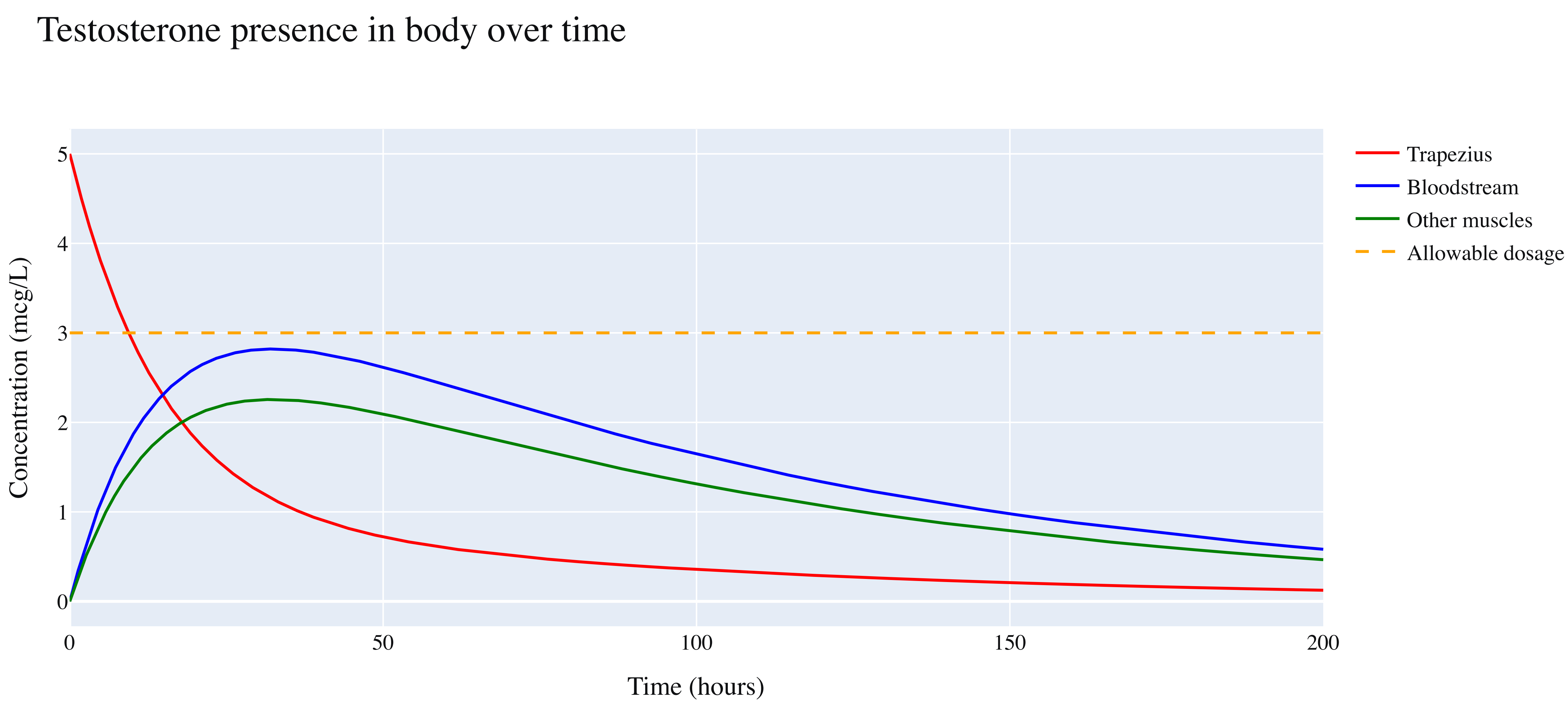
```
        height=1080*0.5,
        width=1920*0.6,
        font_family="CMU Serif",
        font_size=15,
        title_font_size=25,
        font_color="#0e0f11",
        margin=dict(t=120, b=80)
    )
    fig.show()
```

Evaluation

In [ ]:

```
# Initial conditions
trap0, blood0, muscle0 = 5, 0.0, 0.0
# From t = 0 to 200 hours with step size 0.01
t0, t_end, dt = 0, 200, 0.01
maximum_allowable = 3.0
WEIGHT, BODYFAT = 70, 0.20

# Solve the system using the Runge-Kutta method
t, trap, blood, muscle =  runge_kutta_4(f, trap0, blood0, muscle0, t0, t_end, dt)
plot(t, trap, blood, muscle, maximum_allowable, t_end)
```

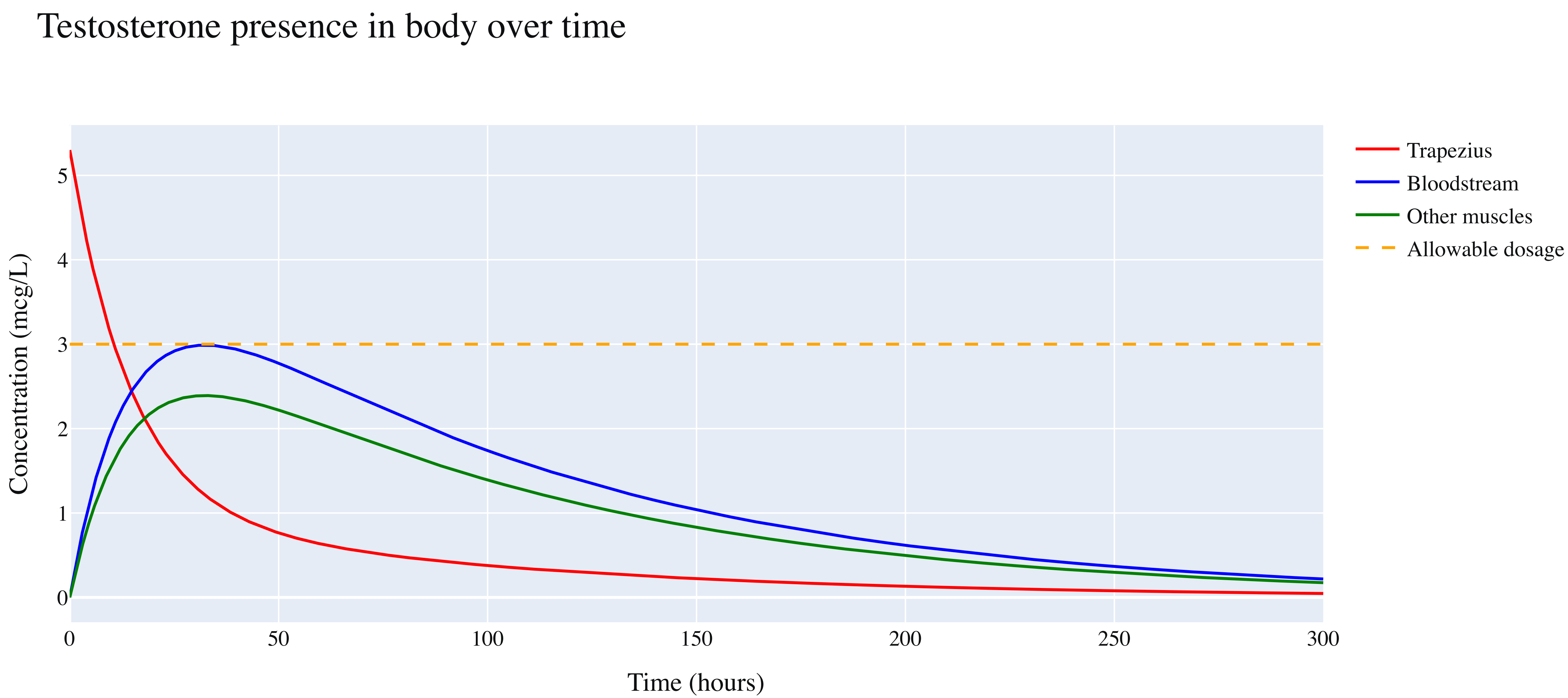


In [ ]:

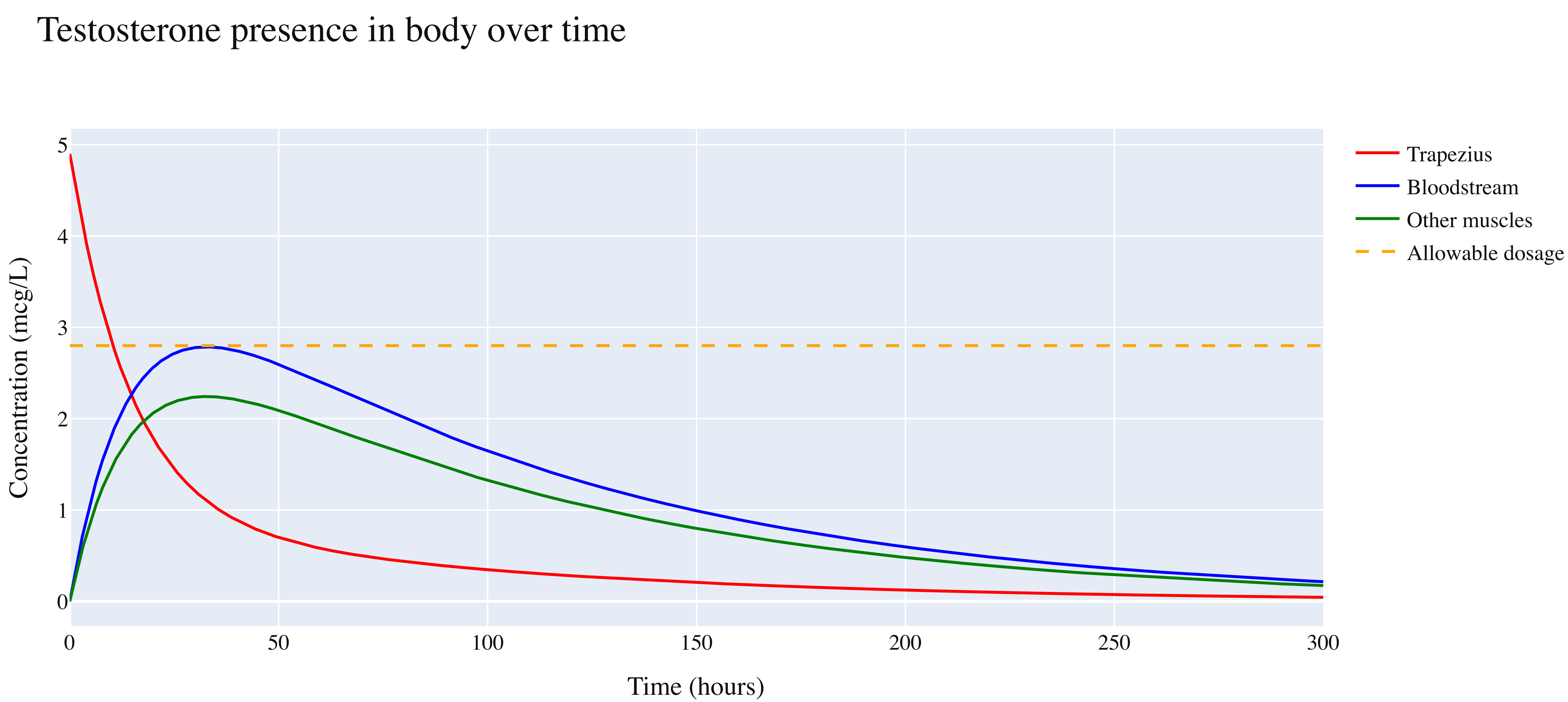
```
WEIGHT, BODYFAT = 70, 0.20
maximum_allowable = 3.0
t0, t_end, dt = 0, 300, 0.01

BODYFAT_GOAL = BODYFAT - 0.015
maximum_dosage = 0
time = 0
```

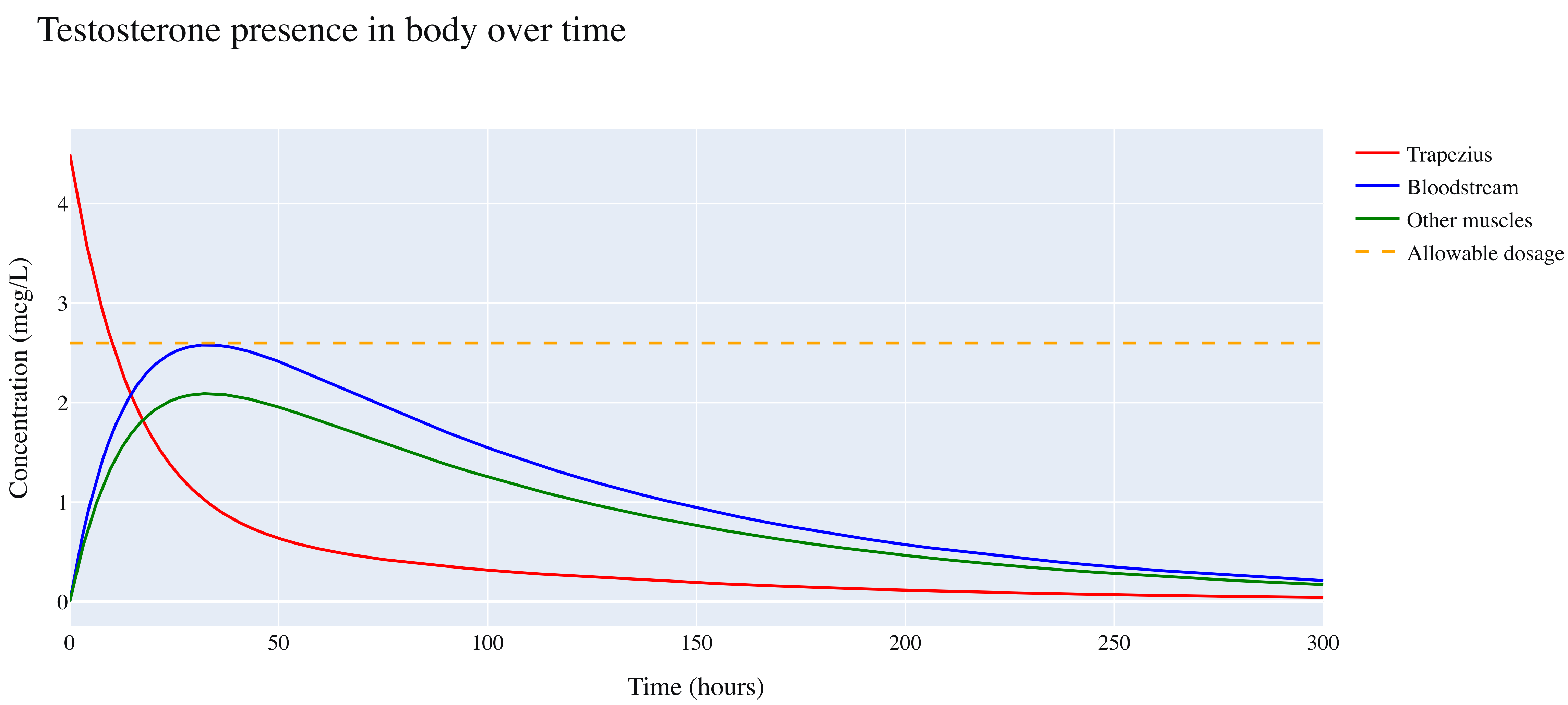
```
for inject in range(1, 10):
    for dosage in range(1, 1000, 1):
        t, trap, blood, muscle = runge_kutta_4(f, dosage/10, blood0, muscle0, t0, t_end, dt)
        if max(blood) < maximum_allowable:
            maximum_dosage = dosage/10
        else:
            break
    t, trap, blood, muscle = runge_kutta_4(f, maximum_dosage, blood0, muscle0, t0, t_end, dt)
    time += t[list(blood).index(min(blood, key=lambda x:abs(x-0.5)))]
    plot(t, trap, blood, muscle, maximum_allowable, t_end)
    maximum_allowable -= 0.2
    BODYFAT -= 0.005
    print(f"Bodyfat: {BODYFAT}")
    print(f"Time: {time} hours")
    if BODYFAT <= BODYFAT_GOAL: break
```




```
Bodyfat: 0.195
Time: 220.26999999994464 hours
```




```
Bodyfat: 0.19
Time: 437.579999999892 hours
```




```
Bodyfat: 0.185
Time: 651.1799999998427 hours
```