Modelling Biological Systems

BIOS13 **Student:** Chandrashekar CR, ch1131ch-s@student.lu.se

Lecturer: Mikael Pnotarp, mikael.pontarp@biol.lu.se

Problem 4: The dynamics of a chemical reaction (4p)

A mixture has two chemical compounds, X and Y. Two units of X can combine and form a single unit of Y. The compound Y is, however, unstable and spontaneously disintegrates into two X. The chemical reactions can be written

$$2X \leftrightarrow Y$$

The dynamics of the corresponding concentrations, denoted x and y respectively, follow -

$$\frac{dx}{dt} = -2kx^2 + 2y\mu$$

$$\frac{dy}{dt} = kx^2 - y\mu$$

where k and μ are positive constants

a) Show that the system has a whole suite of equilibrium states (which depend on the initial conditions, i.e. x(0) and y(0)) (1p)

4a) We know that at equilibrium condition, the concentration x and y do not change over time, we can now write the differential equation as follows -:

$$\frac{dx}{dt} = -2kx^2 + 2y\mu = 0 \tag{1}$$

$$\frac{dy}{dt} = kx^2 - y\mu = 0 \tag{2}$$

$$\text{From the first equation:} \quad -2kx^2 + 2y\mu = 0 \Rightarrow kx^2 = y\mu \tag{3}$$

$$\text{From the second equation:} \quad kx^2 - y\mu = 0 \Rightarrow kx^2 = y\mu \tag{4}$$

Both the equations reduce to same condition. By rearranging we get, $y = \frac{k}{\mu}x^2$. This equilibrium condition means that any pair of (x,y) satisfying this relation is an equilibrium state. Importantly:

1. The equilibrium states depend on the initial conditions x(0) and y(0) because the system evolves dynamically to reach a state where $y = \frac{k}{\mu}x^2$.
2. For any non-negative value of x, there exists a corresponding value of y such that the condition $y = \frac{k}{\mu}x^2$ holds true.
3. Therefore, the system has a continuous suite of equilibrium states determined by the initial concentrations x(0) and y(0).

4. The concentration of x and y must be non-negative (since they represent chemical concentrations.)

$$x \geq 0 \text{ and } y = \frac{k}{\mu}x^2 \geq 0 \tag{5}$$

This further confirms that the equilibrium states are valid for all non-negative x.

> **Problem 4: The dynamics of a chemical reaction (4p)**
>
> b) Write an R script that plots the possible equilibrium in the xy phase plane (1p) (Any positive values of k and μ will do.)

4b) This R script visualizes the equilibrium states of a chemical system using a phase plane, where axes represent species concentrations and plotted lines indicate equilibrium points. The equilibrium relationship between two species, X and Y, derived from a previous calculation (question 4a), is given by $y = (k/\mu)x^2$, where k and μ are rate constants.

The script first prepares the environment by clearing variables and loading the ggplot2 library. It then defines the rate constants k and μ. To generate the equilibrium curve, a sequence of x values (representing species X concentration) from 0 to 10 in 0.1 increments is created, and corresponding y values (species Y concentration) are calculated using the equilibrium equation. These x and y values are stored in a data frame for use with ggplot2.

Using ggplot2, the script creates a plot mapping the data frame's x and y columns to the graph's axes, adding a blue line to represent the equilibrium curve. The plot is enhanced with a title, axis labels, a clean theme (removing grid lines), and increased font sizes for improved readability. Finally, the generated plot is displayed and saved as a high-resolution PNG image with specific dimensions and DPI for publication-quality output, particularly for LaTeX integration.

In summary, the script visualizes a known equilibrium relationship by generating data and using ggplot2 to create a well-formatted phase plane plot.

```
# This script plots the possible equilibrium in the xy phase plane.

# Clear the environment
rm(list = ls())

# Importing Libraries
if(!require(ggplot2)){install.packages("ggplot2")}
library(ggplot2)

# Define parameters
k = 0.5 # Rate constant
mu = 0.2 # Rate constant

# Generate a sequence of x values (non-negative as the chemical concentration cannot be less
    than zero.)
x_values = seq(0, 10, by = 0.1)

# Calculate corresponding y values using the equilibrium condition: y = (k/mu)*x^2
# Solve mathematically from previous question 4a.
y_values = (k / mu) * x_values^2
```

```r
# Create a data frame for plotting
# ggplot2 requires the data to be in data frame format.
eqb_data = data.frame(x = x_values, y = y_values)

# Plot the equilibrium curve in the xy phase plane
eqb_curve = ggplot(data = eqb_data, aes(x, y)) +
  geom_line(aes(color = "Equilibrium States"), linewidth = 1) +
  scale_color_manual(
    name = "Legend",
    values = c(
      "Equilibrium States" = "blue"
    )
  ) +
  labs(
    title = "Equilibrium States in the XY Phase Plane",
    x = "Concentration of X (x)",
    y = "Concentration of Y (y)"
  ) +
  theme_classic() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 18, face = "bold"), # Centered and scaled
        title
    axis.title = element_text(size = 16, face = "bold"),        # Larger and bold axis titles
    axis.text = element_text(size = 14),                        # Larger axis text
    panel.grid.major = element_blank(),                          # Adjust grid line thickness
    panel.grid.minor = element_blank()                           # Hide minor grid lines
  )

# Display the plot
print(eqb_curve)

# Save the plot as a high-resolution PNG
ggsave(
  filename = "./scripts/final_exam/exam_plots/eqb_phase_plane.png",
  plot = eqb_curve,
  width = 200, # Width in mm for LaTeX integration
  units = "mm",
  dpi = 600
)
```

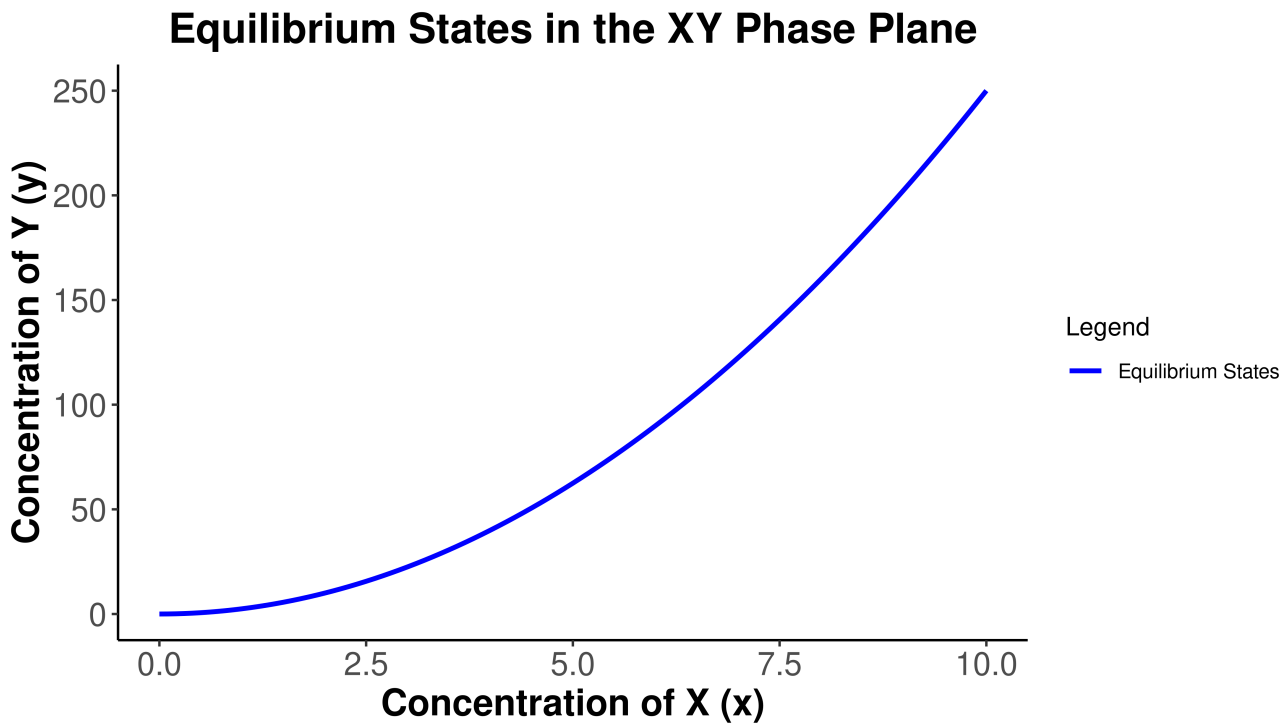

Figure 1: Equilibrium states in the xy Phase Plane

Problem 4: The dynamics of a chemical reaction (4p)

c) Write another script that simulates the differential equations above, using a few different initial conditions, and plots the results in two ways: i) as x and y vs. time and ii) in the phase plane together with solutions in b). (2p)

4c) This R script simulates a system of differential equations describing the dynamics of two reacting chemicals, X and Y. It explores how the initial concentrations of X and Y affect their behavior over time. The script generates two key plots:

1. **Concentration of X and Y over Time:** This plot shows the changing concentrations of both X and Y for three different initial concentration combinations.
2. **Phase Plane with Trajectories:** This plot depicts the system's behavior in a phase plane, where each point represents the concentrations of X and Y at a given time. It visualizes the trajectories of the system for each initial condition, along with the equilibrium curve.

The script works in the following way:

1. **Setting Up the Environment**
   The script starts by clearing the workspace (`rm(list = ls())`) to ensure a clean working environment. Necessary libraries are loaded: `deSolve` for solving the system of differential equations and `ggplot2` for creating visualizations.
2. **Defining the Reaction Dynamics**
   A function named `reaction_dynamics` defines the system of differential equations representing the reaction rates of X and Y. It takes three arguments:
   (a) **t:** Represents time.
   (b) **state:** A vector containing the current concentrations of X and Y.
   (c) **parameters:** A list containing the reaction rate constants (`k` and `mu`).

   The function calculates the rate of change for X (`dxdt`) and Y (`dydt`) based on the current concentrations and parameters. It returns a list in a specific format required by the `deSolve` package.
3. **Defining Parameters and Initial Conditions**
   (a) A list named `params` holds the reaction rate constants (`k` and `mu`).
   (b) Three sets of initial concentrations for X and Y are defined in the `initial_conditions` list. These represent different starting points for the simulations.
   (c) A time sequence (`time`) is created, specifying the time points for which the concentrations will be calculated (0 to 10 with increments of 0.1).
4. **Solving the Differential Equations**
   A function named `solve_reaction_dynamics` takes the initial conditions, parameters, and time sequence as arguments. It uses the `ode` function from the `deSolve` package to solve the system of differential equations for the given initial conditions and parameters over the specified time points. The result is a data frame containing the concentrations of X and Y at each time point. The function adds a label (`"Condition"` with a number) to each data frame to identify the corresponding initial condition.

5. **Simulating Dynamics for Each Condition**
   The `lapply` function iterates through each set of initial conditions. Inside the loop, the `solve_reaction_dynamics` function is called for each initial condition, generating a data frame of concentration values. The resulting data frames are combined into a single data frame (`results_df`) using `do.call(rbind)`. This allows for easier plotting using `ggplot2`.

6. **Generating Plot Captions**
   A text string (`initial_conditions_caption`) is created, listing the initial X and Y values for each condition. This will be used as a caption in the plots for clarity.

7. **Plotting Concentration Dynamics over Time (4c i)**
   A `ggplot2` plot named `time_plot` is created. It uses the `results_df` data frame, where each row represents a data point for a specific time, condition, and concentration (X or Y). The plot shows two lines:

   - Red line: Concentration of X.
   - Blue line: Concentration of Y.

   The plot is facetted by "Condition" using `facet_wrap` to display the results for each initial condition side-by-side. Axis labels, title, color legend, and caption are added using `labs` and other formatting options. The plot is customized for better readability using `scale_color_manual` and `theme_classic`.

8. **Plotting Phase Plane with Trajectories (4c ii)**
   A function named `plot_phase_plane` is defined to create the phase plane plot:

   (a) It generates points for the equilibrium curve by calculating corresponding Y values for a range of X values based on the equilibrium condition: $Y = \frac{k}{\mu} \cdot X^2$.

   (b) It combines the simulation data from all conditions into a single data frame (`combined_data`). Each data point includes concentration values (X and Y), time, and a label indicating the condition.

   (c) The plot adds:
   - The equilibrium curve as a blue dashed line (`geom_line`).
   - Trajectories for each initial condition as colored lines (`geom_path`).
   - Initial points (time = 0) for each condition marked with larger circles (`geom_point`).

   (d) Axis labels, title, and color legend are added using `labs`, and plot limits are set using `coord_cartesian`.

   (e) The plot is customized using `theme_classic`.

9. **Generating and Saving Plots**

   - The script calls `print(time_plot)` to display the concentration dynamics over time plot. It saves this plot as `"time_plot.png"` using `ggsave` with a resolution of 600 dpi.
   - Similarly, `print(phase_plane_plot)` displays the phase plane plot, saved as `"phase_plane_traj.png"` with a width of 200 mm and 600 dpi resolution.

This script demonstrates how to simulate a system of differential equations for different initial conditions and visualize the results using time series and phase plane plots with `ggplot2`.

```r
# This script simulates the differential equations metioned in the previous questions 4b and
    4a, using a set of different initial conditions.
# i) The first plot is the concentration of X and Y with respect to time.
# ii) The second plot is in the phase plane together with the solutions in 4b.


# Clear the environment
rm(list = ls())

# Import Required Libraries
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(deSolve)){install.packages("deSolve")}
library(deSolve) # For solving the ODE
library(ggplot2)

# Define the ODE system for reaction dynamics
reaction_dynamics = function(t, state, parameters) {
  x = state[1] # Initial concentration of X (reactant)
  y = state[2] # Initial concentration of Y (product)

  k = parameters$k # Reaction rate constant (positive)
  mu = parameters$mu # Reaction rate constant (positive)

  dxdt = -(2 * k * x^2) + (2 * mu * y) # Rate of change of concentration of X (ODE 1)
  dydt = (k * x^2) - (mu * y) # Rate of change of concentration of Y (ODE 2)

  return(list(c(dxdt, dydt))) # deSolve package requires this particular format
}

# Define parameters and initial conditions
params = list(k = 0.5, mu = 0.2) # Any value can be given here
initial_conditions = list(
  c(x = 0.5, y = 0.1),
  c(x = 2, y = 1),
  c(x = 1, y = 2)
)
time = seq(0, 10, by = 0.1) # Sequence of time points from 0 to 10 with increments of 0.1

# Solve the ODEs for each initial condition
solve_reaction_dynamics = function(initial_cond, params, time) {
  as.data.frame(ode(
    y = initial_cond,
    times = time,
    func = reaction_dynamics, # Pass the function defined previously here.
    parms = params
  ))
}

# Simulate dynamics for each initial condition and add labels
simulation_results = lapply(seq_along(initial_conditions), function(i) {
  # Solve the ODEs for the i-th initial condition
  result = solve_reaction_dynamics(initial_conditions[[i]], params, time)
  # Add a label indicating the initial condition for clarity
```

```
  result$Condition = paste("Condition", i)
  return(result)
})

# Combine simulation results from each of the initial conditions into a single data frame
    for easier plotting using ggplot2.
results_df = do.call(rbind, simulation_results)

# Generate caption text listing the initial conditions used
initial_conditions_caption = paste(
  "Initial Conditions:",
  paste(
    lapply(seq_along(initial_conditions), function(i) {
      # Format the initial conditions for each condition (X and Y values)
      sprintf("Condition %d: (x = %.1f, y = %.1f)", i, initial_conditions[[i]]["x"],
          initial_conditions[[i]]["y"])
    }),
    collapse = "; " # Separate conditions with a semicolon and space
  )
)

# 4c i) Plot dynamics of X and Y over time
time_plot = ggplot(results_df, aes(x = time)) +
  geom_line(aes(y = x, color = "Concentration of X"), linewidth = 1) +
  geom_line(aes(y = y, color = "Concentration of Y"), linewidth = 1) +
  facet_wrap(~Condition, scales = "free_y") + # Doing a facet_wrap allows to plot all the
      three defined conditions side-by-side.
  labs(
    title = "Dynamics of X and Y Over Time",
    x = "Time",
    y = "Concentration",
    color = "Legend",
    caption = initial_conditions_caption # To know the initial values of each of the
        conditions, from the plot.
  ) +
  scale_color_manual(
    values = c(
      "Concentration of X" = "red",
      "Concentration of Y" = "blue"
    )
  ) +
  theme_classic() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 18, face = "bold"), # Centered and scaled
        title
    axis.title = element_text(size = 16, face = "bold"),        # Larger and bold axis titles
    axis.text = element_text(size = 14),                        # Larger axis text
    panel.grid.major = element_blank(),                         # Adjust grid line thickness
    panel.grid.minor = element_blank(),                         # Hide minor grid lines
    legend.text = element_text(size=12),
    legend.position = "bottom",
    plot.caption = element_text(hjust = 0, face = "italic", size = 12) # Caption styling
  )
```

```r
# 4c ii) Plot 2: Phase plane with trajectories and equilibrium curve
plot_phase_plane <- function(simulations, params) {
  # Generate points for the equilibrium curve
  x_vals <- seq(0, 10, by = 0.1)
  y_vals <- (params$k / params$mu) * x_vals^2
  equilibrium_curve <- data.frame(x = x_vals, y = y_vals, label = "Equilibrium Line") # Add
      label for the legend

  # Combine simulation data from all conditions
  combined_data <- do.call(rbind, lapply(seq_along(simulations), function(i) {
    df <- simulations[[i]]
    df$Simulation <- paste("Condition", i)
    return(df)
  }))

  # Create the phase plane plot
  ggplot() +
    # Add the equilibrium curve with explicit color and label
    geom_line(data = equilibrium_curve, aes(x = x, y = y, color = label),
              linetype = "dashed", linewidth = 1) +
    # Add trajectory paths for each initial concentration condition
    geom_path(data = combined_data, aes(x = x, y = y, color = Simulation),
              linewidth = 1) +
    # Mark initial points
    geom_point(data = combined_data[combined_data$time == 0, ],
               aes(x = x, y = y, color = Simulation), size = 3) +
    # Customize color scale to include equilibrium line and conditions
    scale_color_manual(
      name = "Legend",
      values = c(
        "Equilibrium Line" = "blue",
        "Condition 1" = "red",
        "Condition 2" = "green",
        "Condition 3" = "purple"
      )
    ) +
    labs(
      title = "Phase Plane with Trajectories and Equilibrium Curve",
      x = "Concentration of X",
      y = "Concentration of Y"
    ) +
    coord_cartesian(xlim = c(0, 2.5), ylim = c(0, 5)) +
    theme_classic() +
    theme(
      plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
      axis.title = element_text(size = 16, face = "bold"),
      axis.text = element_text(size = 14),
      legend.position = "bottom", # Position the legend at the bottom
      legend.title = element_text(size = 14, face = "bold"),
      legend.text = element_text(size = 12)
    )
}
```

```r
# Generate the phase plane plot
phase_plane_plot = plot_phase_plane(simulation_results, params)

# Display plots
print(time_plot)
ggsave(plot = time_plot, filename = "./scripts/final_exam/exam_plots/time_plot.png", dpi =
    600)

print(phase_plane_plot)
ggsave(plot = phase_plane_plot, filename =
    "./scripts/final_exam/exam_plots/phase_plane_traj.png", width = 200, units = "mm", dpi =
    600)
```

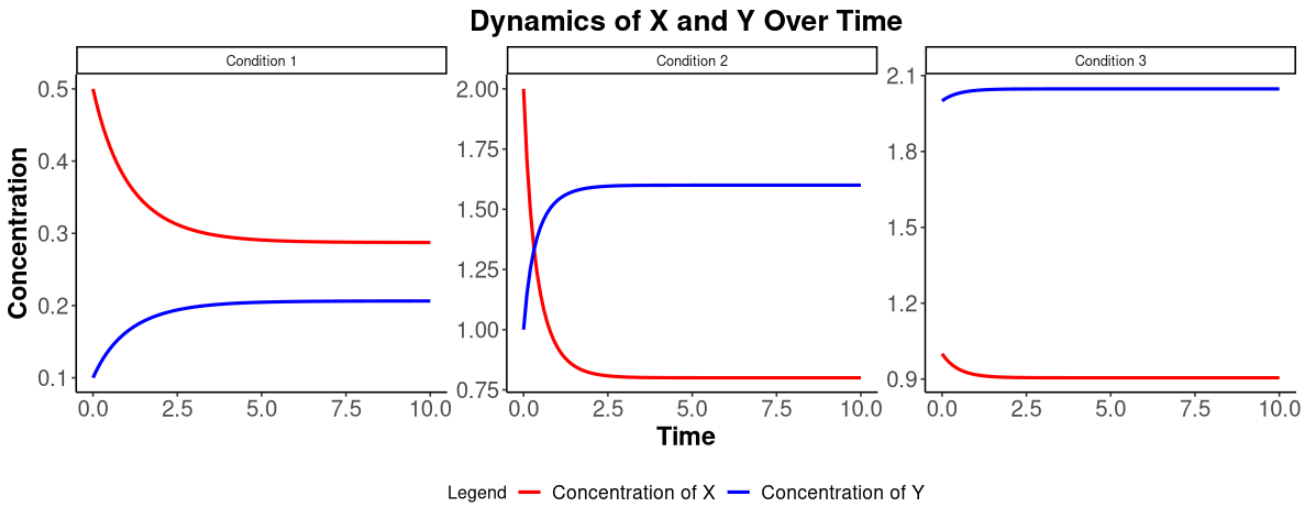

Figure 2: (i) Dynamics of X and Y over time.

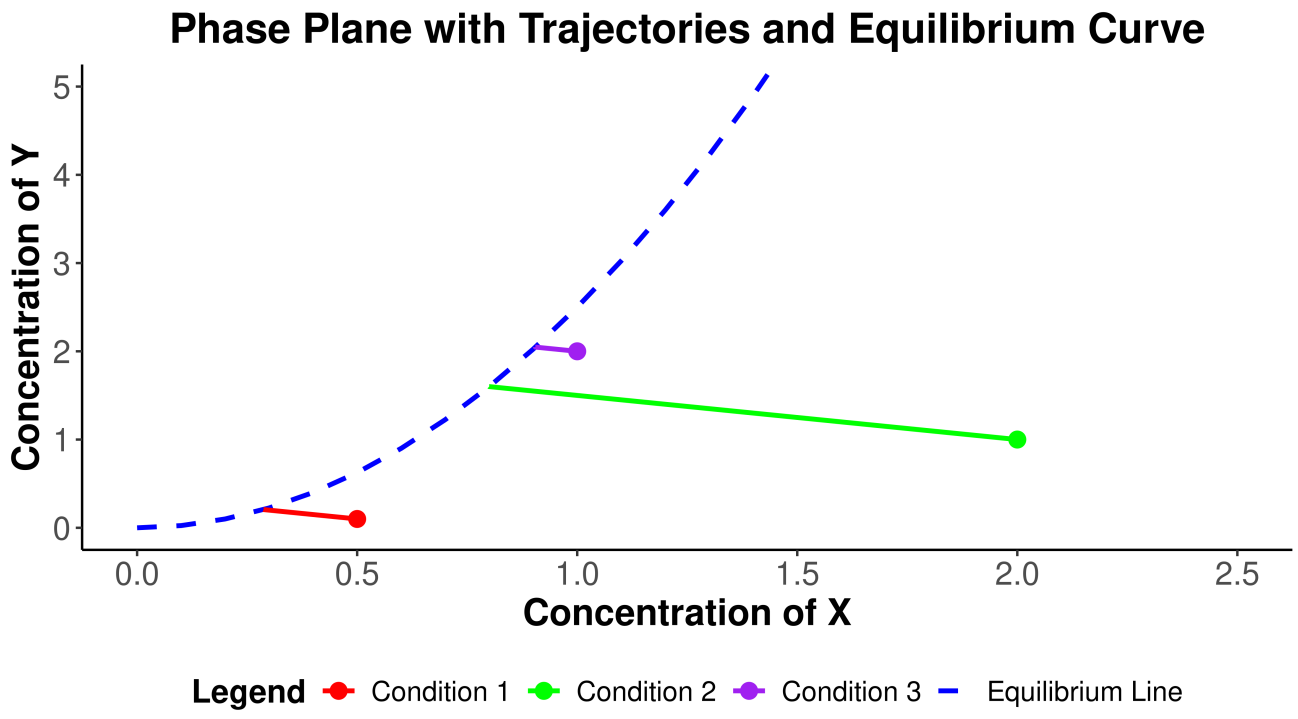

Figure 3: (ii) Phase Plane with trajectories and equilibrium curve.