

# **UNIVERSIDAD DE GUANAJUATO**

DIVSIÓN DE CIENCIAS E INGENIERÍAS

MÉTODOS NUMÉTICOS

GRUPO A

CLAVE IILI0505

CATEDRÁTICO: ALMA XÓCHTL

TAREA FINAL:

**“APLICACIÓN DE MÉTODOS RUNGE-KUTTA A MODELOS CINÉTICOS LOGÍSTICOS EN  
INGENIERÍA QUÍMICA”**

ALUMNA:

ANA ISABEL ESQUIVEL CASTRO

FECHA DE ENTREGA: 09/12/2025



## Tarea Final

# “Aplicación de Métodos Runge-Kutta a Modelos Cinéticos Logísticos en Ingeniería Química”

## 1. Código de Runge-Kutta de 4º Orden

```
#include <stdio.h>

void calcular_derivada(double t, double C, double *dCdt) { //función que calcula la derivada de dc/dt
    double k = 0.45;//cte cinética
    double Cmax = 1.0;//concentración máxima

    // Ecuación logística: dC/dt = k * C * (Cmax - C)
    *dCdt = k * C * (Cmax - C);
}

int main() {
    double tiempo = 0.0;//definimos el tiempo inicial
    double paso = 0.1;//nuestro delta t es Δt = 0.1
    double concentracion = 0.02;// la concentración en 0 es C(0) = 0.02, usando litros por mol
    double k1, k2, k3, k4; //tenemos las variables para las 4 pendientes de RK4

    printf("Resultados\n");
    printf("dC/dt= 0.45*C*(1-C)\n");
    printf("C(0)= 0.02 mol/L, dt= 0.1 min\n\n");
    printf("t(min)\tC(mol/L)\n");
    printf("%.1f\t%.6f\n", tiempo, concentracion); //mostramos la condición inicial
}
```

```
int num_pasos = (int)(25.0 / paso);//calculamos hasta t=25 min

for(int i = 0; i < num_pasos; i++) {
    //ciclo para calcular la pendiente k1, k2, k3 y k4
    calcular_derivada(tiempo, concentracion, &k1);
    calcular_derivada(tiempo + paso/2, concentracion +
    paso*k1/2,
    &k2);
    calcular_derivada(tiempo + paso/2,
    concentracion +
    paso*k2/2,
    &k3);
    calcular_derivada(tiempo + paso,
    concentracion +
    paso*k3,
    &k4);

    concentracion = concentracion +
    (paso/6.0)*(k1 + 2*k2 + 2*k3 + k4);
    //actualizamos la concentración usando la fórmula de RK4
    tiempo = tiempo + paso;
    if (i % 25 == 0) { //Resultado cada 2.5 min
        printf("%.1f\t%.6f\n", tiempo,
        concentracion);
    }
}
return 0;
}
```

**Resultado**

```

Resultados
dC/dt= 0.45*C*(1-C)
C(0)= 0.02 mol/L, dt= 0.1 min

t(min) C(mol/L)
0.0 0.020000
0.1 0.020901
2.6 0.061698
5.1 0.168426
7.6 0.384185
10.1 0.657727
12.6 0.855472
15.1 0.948003
17.6 0.982505
20.1 0.994252
22.6 0.998127

```

## 2. Código de Range-Kutta de 2º Orden

```

#include <stdio.h>

void calcular_derivada(double t, double C,
double *dCdt) { // Función que calcula la
derivada dC/dt

    double k = 0.45;//cte cinética

    double Cmax = 1.0;// concentración
máxima

    // Ecuación logística: dC/dt = k * C *
(Cmax - C)

    *dCdt = k * C * (Cmax - C);
}

int main() {
    double tiempo = 0.0;//tiempo 0

    double paso = 0.1;// saltos o pasos con
delta Δt = 0.1

```

```

        double concentracion = 0.02; // C(0) =
0.02 en mol por litro

        double k1, k2;//variables de las
pendientes

        printf("Resultados\n");
        printf("Ecuacion: dC/dt = 0.45 *C*(1-
C)\n");
        printf("C(0) = 0.02 mol/L, dt = 0.1
min\n\n");

        printf("t(min)\tC(mol/L)\n");
        printf("%.1f\t%.6f\n", tiempo,
concentracion);//condición inicial

        int num_pasos = (int)(25.0 / paso);
//calcular hasta t=25 min

        for(int i = 0; i < num_pasos; i++) { //ciclo
para calcular k1 y k2

            calcular_derivada(tiempo,
concentracion, &k1);

            calcular_derivada(tiempo + paso,
concentracion + paso*k1,
&k2);

            concentracion = concentracion +
(paso/2.0)*(k1 + k2);

            tiempo = tiempo + paso;

            if (i % 25 == 0) { //resultado cada 2.5
min

                printf("%.1f\t%.6f\n", tiempo,
concentracion);

```

```

        }
    }

return 0;
}

```

## Resultado

```

Resultados
Ecuacion: dC/dt = 0.45 *C*(1-C)
C(0) = 0.02 mol/L, dt = 0.1 min

t(min)  C(mol/L)
0.0      0.020000
0.1      0.020901
2.6      0.061678
5.1      0.168342
7.6      0.384005
10.1     0.657532
12.6     0.855342
15.1     0.947937
17.6     0.982476
20.1     0.994240
22.6     0.998122

```

### 3. Código comparativo de Range-Kuta entre 2º y 4º Orden

```

#include <stdio.h>
#include <math.h>

// Función que calcula la derivada
void derivada(double t, double C, double
*dCdCdt) {
    *dCdCdt = 0.45 * C * (1.0 - C);
}

// Función para calcular un paso con
RK4
double paso_rk4(double t, double C,
double paso) {
    double k1, k2, k3, k4;

```

```

        derivada(t, C, &k1);
        derivada(t + paso/2, C + paso*k1/2,
&k2);
        derivada(t + paso/2, C + paso*k2/2,
&k3);
        derivada(t + paso, C + paso*k3, &k4);

        return C + (paso/6.0)*(k1 + 2*k2 +
2*k3 + k4);
    }

    // Función para calcular un paso con
    RK2
    double paso_rk2(double t, double C,
    double paso) {
        double k1, k2;

        derivada(t, C, &k1);
        derivada(t + paso, C + paso*k1, &k2);

        return C + (paso/2.0)*(k1 + k2);
    }

    int main() {
        double t = 0.0;
        double paso_rk4_valor = 0.1;
        double C_rk4 = 0.02, C_rk2 = 0.02;

        printf("RK4 usa dt = 0.1\n");
        printf("RK2 prueba diferentes
dt\n\n");

        printf("Resultado RK4 (dt=0.1) en
t=25 min:\n"); //primero calculamos
RK4 con dt=0.1
        double tiempo_rk4 = 0.0;
        double C_rk4_final = 0.02;
        int pasos_rk4 = (int)(25.0 /
paso_rk4_valor);
    }
}

```

```

for(int i = 0; i < pasos_rk4; i++) {
    C_rk4_final = 
paso_rk4(tiempo_rk4,      C_rk4_final,
paso_rk4_valor);
    tiempo_rk4 += paso_rk4_valor;
}
printf("C(25) = %.8f mol/L\n\n",
C_rk4_final);
printf("dt\tC(25) RK2\tDiferencia
con    RK4\n");//ahora probamos
diferentes dt para RK2
double dt_prueba[] = {0.1, 0.05, 0.02,
0.01, 0.005};

for(int j = 0; j < 5; j++) {
    double dt = dt_prueba[j];
    double t_rk2 = 0.0;
    double C_rk2_temp = 0.02;
    int pasos_rk2 = (int)(25.0 / dt);

    for(int i = 0; i < pasos_rk2; i++) {
        C_rk2_temp = paso_rk2(t_rk2,
C_rk2_temp, dt);
        t_rk2 += dt;
    }

    double diferencia =
fabs(C_rk2_temp - C_rk4_final);
    printf("%.4f\t%.8f\t%.8f\n", dt,
C_rk2_temp, diferencia);
}

double dt_prueba_fino = 0.01;
double diferencia_minima = 1.0;
double dt_optimo = 0.1;

for(int intento = 0; intento < 10;
intento++) {

```

```

    double t_rk2 = 0.0;
    double C_rk2_temp = 0.02;
    int pasos_rk2 = (int)(25.0 /
dt_prueba_fino);

    for(int i = 0; i < pasos_rk2; i++) {
        C_rk2_temp = paso_rk2(t_rk2,
C_rk2_temp, dt_prueba_fino);
        t_rk2 += dt_prueba_fino;
    }

    double diferencia =
fabs(C_rk2_temp - C_rk4_final);
    printf("dt = %.6f -> C(25) = %.8f,
diferencia = %.8f\n",
dt_prueba_fino, C_rk2_temp,
diferencia);

    if (diferencia < diferencia_minima)
{
        diferencia_minima = diferencia;
        dt_optimo = dt_prueba_fino;
    }

    dt_prueba_fino /= 2.0; // Probamos
    con dt más pequeño
}

printf("Para obtener el mismo
resultado que RK4 con dt=0.1,\n");
printf("RK2 necesita dt ≈ %.6f min\n",
dt_optimo);

return 0;
}

```

**Resultado**

```

RK4 usa dt = 0.1
RK2 prueba diferentes dt

Resultado RK4 (dt=0.1) en t=25 min:
C(25) = 0.99936305 mol/L

dt      C(25) RK2      Diferencia con RK4
0.1000  0.99936123  0.00000181
0.0500  0.99936260  0.00000045
0.0200  0.99936298  0.00000007
0.0100  0.99936303  0.00000002
0.0050  0.99936304  0.00000000
dt = 0.010000 -> C(25) = 0.99936303, diferencia = 0.00000002
dt = 0.005000 -> C(25) = 0.99936304, diferencia = 0.00000000
dt = 0.002500 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.001250 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000625 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000313 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000156 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000078 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000039 -> C(25) = 0.99936305, diferencia = 0.00000000
dt = 0.000020 -> C(25) = 0.99936305, diferencia = 0.00000000
Para obtener el mismo resultado que RK4 con dt=0.1,
RK2 necesita dt ≈ 0.000625 min

```

#### 4. Conclusión

En el presente trabajo se resolvió la ecuación diferencial logística que modela la cinética de la reacción  $A+B \rightarrow 2B$ , utilizando métodos numéricos de Runge-Kutta.

Con los parámetros proporcionados ( $k=0.45 \text{ L}\cdot\text{mol}^{-1}\cdot\text{min}^{-1}$ ,  $C_{\max}=1 \text{ mol/L}$ ,  $C(0)=0.02 \text{ mol/L}$ ) y un intervalo de tiempo de 0 a 25 minutos, se implementó el método de cuarto orden (RK4) con un paso de tiempo  $\Delta t=0.1 \text{ min}$ . Los resultados muestran que la concentración del producto aumenta de manera sigmoide, partiendo de 0.02 mol/L y aproximándose asintóticamente al valor máximo de 1 mol/L. En el tiempo final  $t=25 \text{ min}$ , la concentración calculada con RK4 fue de 0.99936305 mol/L, lo que confirma que el sistema alcanza prácticamente la concentración máxima esperada.

Al comparar con el método de Runge-Kutta de segundo orden (RK2) utilizando el mismo  $\Delta t=0.1 \text{ min}$ , se obtuvo un valor de  $C(25)=0.99936123 \text{ mol/L}$ . La diferencia entre ambos métodos es pequeña, del orden de  $1.81\times 10^{-6} \text{ mol/L}$ , lo que indica que, para este problema, ambos métodos convergen a la solución, pero RK4 ofrece mayor precisión con el mismo tamaño de paso. Esto era de esperarse debido al mayor orden del método, que reduce el error de truncamiento.

Para determinar el tamaño de paso necesario en RK2 que proporcione la misma precisión que RK4 con  $\Delta t=0.1 \text{ min}$ , se realizaron varias pruebas con diferentes valores de  $\Delta t$ . Se observó que, al reducir  $\Delta t$  en RK2, la diferencia con el resultado de RK4 disminuye. Con  $\Delta t=0.05 \text{ min}$ , la diferencia es de  $4.5\times 10^{-7} \text{ mol/L}$ ; con  $\Delta t=0.01 \text{ min}$ , la diferencia es de  $2\times 10^{-8} \text{ mol/L}$ ; y con  $\Delta t=0.005 \text{ min}$ , la diferencia se hace prácticamente nula. Por lo tanto, se concluye que RK2 requiere un paso aproximadamente de

0.005 min para igualar la precisión de RK4 con  
paso 0.1 min.

Esto implica que el método de cuarto orden es más eficiente, ya que con menos evaluaciones de la función se logra una precisión comparable a la del método de segundo orden con un paso mucho más pequeño.