Universidad
Nacional del
Altiplano de Puno

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

TRABAJO ENCARGADO

CURSO: Programación Numérica

TEMA: Interpolación: Métodos y Fórmulas de Alta Precisión

DOCENTE: Ing. Fred Torres Cruz

PRESENTADO POR: Sadith Lina Apaza Huayta

SEMESTRE: cuarto

SECCIÓN: “A”

PUNO – PERÚ

2025 – II

Interpolación: Métodos y Fórmulas de Alta Precisión

1. Interpolación Lineal:

Ejercicio 1.1— Temperatura de CPU
En un servidor de bases de datos se registró la temperatura de la CPU: a los 10 minutos del inicio de un proceso intensivo la temperatura es 65 °C; a los 30 minutos sube a 85 °C. Como ingeniero de infraestructura quieres estimar rápidamente la temperatura en el punto medio (20 min) para activar un control de ventilación si supera 75 °C.

Código:
```
x <- c(10, 30)
y <- c(65, 85)
x_int <- 20
y_int <- y[1] + (y[2] - y[1]) * (x_int - x[1]) / (x[2] - x[1])
cat("Temperatura estimada:", y_int, "°C\n")
```

Ejercicio 1.2 — Pérdida de paquetes en red
El equipo de redes observa que a 5 Mbps la pérdida de paquetes promedio es 2% y a 15 Mbps sube a 5%. Debes estimar la pérdida en 10 Mbps para decidir si cambiar la configuración QoS antes de un despliegue crítico.

Código:
```
x <- c(5, 15)
y <- c(2, 5)
x_int <- 10
y_int <- y[1] + (y[2] - y[1]) * (x_int - x[1]) / (x[2] - x[1])
cat("Pérdida estimada:", y_int, "%\n")
```

Ejercicio 1.3 — Crecimiento de usuarios
Una app universitaria tenía 500 usuarios activos en enero (mes 1) y 800 en marzo (mes 3). Para planificar escalado de base de datos quieres estimar usuarios en febrero (mes 2) y con eso dimensionar memoria temporal.

Código:
```
x <- c(1, 3)
y <- c(500, 800)
x_int <- 2
y_int <- y[1] + (y[2] - y[1]) * (x_int - x[1]) / (x[2] - x[1])
cat("Usuarios estimados en febrero:", y_int, "\n")
```

2. Interpolación de Lagrange:

Ejercicio 2.1 — Latencia de red

Durante pruebas mediste latencias: a 1 Mbps la latencia media es 80 ms, a 5 Mbps 50 ms y a 10 Mbps 30 ms. Necesitas estimar la latencia a 7 Mbps para modelar la experiencia de usuario al seleccionar un servidor en la nube.

Código:

```
x <- c(1, 5, 10)
y <- c(80, 50, 30)
x_int <- 7
P <- function(x_int, x, y){
  n <- length(x)
  L <- rep(1, n)
  for(i in 1:n){
    for(j in 1:n){
      if(j != i){
        L[i] <- L[i] * (x_int - x[j]) / (x[i] - x[j])
      }
    }
  }
  sum(y * L)
}
cat("Latencia estimada:", P(x_int, x, y), "ms\n")
```

Ejercicio 2.2 — Fallos de software
En pruebas de estrés de una API, el número de fallos detectados fue: tras 1 h = 2 fallos, 3 h = 6 fallos, 5 h = 20 fallos. Quieres predecir fallos en la hora 4 para estimar recursos de rollback y plan de pruebas adicionales.

Código:

```
x <- c(1, 3, 5)
y <- c(2, 6, 20)
cat("Fallos estimados:", P(4, x, y), "\n")
```

Ejercicio 2.3 — Uso de CPU en simulación
En una simulación de cargas, uso de CPU registrado: 2 s → 20%, 4 s → 40%, 6 s → 85%. Debes estimar uso a 5 s para calibrar el número de hilos en producción.

Código:

```
x <- c(2, 4, 6)
y <- c(20, 40, 85)
cat("Uso estimado:", P(5, x, y), "%\n")
```

3. Diferencias Divididas de Newton

Ejercicio 3.1 — Rendimiento de algoritmo
Mides tiempo de un algoritmo: 10 datos → 0.5 s; 20 → 1.5 s; 30 → 3.2 s. Quieres estimar el tiempo para 25 datos para decidir si el algoritmo es viable en producción. Usa Newton para interpolar.

Código:

```
x <- c(10, 20, 30)
```

```
y <- c(0.5, 1.5, 3.2)
newton_interp <- function(x, y, x_int){
  n <- length(x)
  dd <- matrix(0, n, n)
  dd[,1] <- y
  for(j in 2:n){
    for(i in 1:(n-j+1)){
      dd[i,j] <- (dd[i+1,j-1] - dd[i,j-1]) / (x[i+j-1] - x[i])
    }
  }
  P <- y[1]
  prod_term <- 1
  for(i in 1:(n-1)){
    prod_term <- prod_term * (x_int - x[i])
    P <- P + dd[1,i+1] * prod_term
  }
  P
}
cat("Tiempo estimado:", newton_interp(x, y, 25), "s\n")
```

Ejercicio 3.2 — Densidad de datos faltantes
El equipo de ETL observó: archivo 1 GB → 50k registros; 3 GB → 120k; 5 GB → 210k. Para planificar particionado quieres estimar registros a 4 GB.

Código:
```
x <- c(1,3,5)
y <- c(50,120,210)
cat("Densidad estimada:", newton_interp(x, y, 4), "mil datos\n")
```

Ejercicio 3.3 — Velocidad de conexión
Monitoreaste velocidad: 8 h → 10 Mbps, 10 h → 20 Mbps, 12 h → 40 Mbps. Para balanceo de carga quieres estimar la velocidad a las 11 h usando Newton.

Código:
```
x <- c(8,10,12)
y <- c(10,20,40)
cat("Velocidad estimada:", newton_interp(x, y, 11), "Mbps\n")
```

4. Interpolación Cuadrática

Ejercicio 4.1 — Carga del sistema
Durante una ventana pico medir: 1 h → 15% de carga, 2 h → 40%, 3 h → 90%. Encuentra el polinomio cuadrático que ajuste estos puntos y estima carga a 2.5 h para planificar autoscaling.

Código:
```
x <- c(1,2,3)
y <- c(15,40,90)
modelo <- lm(y ~ poly(x, 2, raw=TRUE))
pred <- predict(modelo, data.frame(x=2.5))
```

```
cat("Carga estimada:", pred, "%\n")
```

Ejercicio 4.2 — Tiempo de ejecución
Perfilas una función: tamaño 2 → 1 s, 3 → 3 s, 5 → 9 s. Estimar tiempo a 4 usando ajuste cuadrático (útil para estimar costos de Job).

Código:
```
x <- c(2,3,5)
y <- c(1,3,9)
modelo <- lm(y ~ poly(x,2,raw=TRUE))
cat("Tiempo estimado:", predict(modelo, data.frame(x=4)), "s\n")
```

Ejercicio 4.3 — Consumo energético
Mediste consumo: 1 V → 10 mA, 2 V → 25 mA, 3 V → 55 mA. Ajusta un polinomio cuadrático y estima consumo a 2.5 V para dimensionar fuente de alimentación.

Código:
```
x <- c(1,2,3)
y <- c(10,25,55)
modelo <- lm(y ~ poly(x,2,raw=TRUE))
cat("Consumo estimado:", predict(modelo, data.frame(x=2.5)), "mA\n")
```

5. Splines Cúbicos

Ejercicio 5.1 — Datos de sensores
Un experimento IoT registra humedad: t=0h→20%, 2h→35%, 4h→50%, 6h→65%. Hay ruido; quieres una curva suave para estimar 3.5 h (para alimentar un modelo predictivo). Usa spline cúbico natural.

Código:
```
library(splines)
x <- c(0,2,4,6)
y <- c(20,35,50,65)
modelo <- splinefun(x,y,method="natural")
cat("Humedad estimada:", modelo(3.5), "%\n")
```

Ejercicio 5.2 — Temperatura ambiente
Estaciones: 8h→12°C, 10h→18°C, 12h→27°C, 14h→25°C. Para optimizar ventilación estimas temperatura en 11h con spline cúbico.

Código:
```
x <- c(8,10,12,14)
y <- c(12,18,27,25)
modelo <- splinefun(x,y,method="natural")
cat("Temperatura estimada:", modelo(11), "°C\n")
```

Ejercicio 5.3 — Carga de red
En un monitoreo ves tráfico acumulado: 0h→100MB, 2h→200MB, 5h→800MB, 7h→1000MB. Necesitas estimar consumo a 3h y trazar la curva suave para presentación.

Código:

```
x <- c(0,2,5,7)
y <- c(100,200,800,1000)
modelo <- splinefun(x,y,method="natural")
cat("Tráfico estimado:", modelo(3), "MB\n")
```

6. Error de Interpolación

Ejercicio 6.1 — Estimación de error en CPU

La relación real entre carga y tiempo es $f(x)=x^2$. Si por simplicidad usas interpolación lineal con puntos (1,1) y (3,9) para estimar en x=2, calcula el error absoluto y concluye si la aproximación es aceptable para control rápido.

Código:

```
f <- function(x) x^2
x <- c(1,3); y <- f(x)
interp <- y[1] + (y[2]-y[1])*(2-x[1])/(x[2]-x[1])
error <- abs(f(2)-interp)
cat("Error estimado:", error, "\n")
```

Ejercicio 6.2 — Error en medición de temperatura

Para una calibración rápida usas interpolación lineal entre (0,0) y (π/2,1) para predecir sin(π/4). Calcula el error real y discute si es suficiente según tolerancia 0.01 para un sensor.

Código:

```
f <- function(x) sin(x)
x <- c(0,pi/2); y <- f(x)
interp <- y[1] + (y[2]-y[1])*(pi/4 - x[1])/(x[2]-x[1])
error <- abs(f(pi/4)-interp)
cat("Error estimado:", error, "\n")
```

Ejercicio 6.3 — Error en datos de sensores

Para $f(x)=\ln(1+x)$ en [0,0.5] quieres que la cota máxima del error del polinomio interpolante sea ≤ 1e-4. Usando nodos equiespaciados $x_k = k\cdot(0.5/n)$, estima el menor n necesario (búsqueda numérica) y muestra la cota aproximada. Esto ayuda a decidir cuántos nodos medir en un muestreo de logs.

Código:

```
f <- function(x) log(x+1)
x <- c(0,2); y <- f(x)
interp <- y[1] + (y[2]-y[1])*(1-x[1])/(x[2]-x[1])
error <- abs(f(1)-interp)
cat("Error estimado:", error, "\n")
```