Ejercicios Tema 3

Juan Rubio Cobeta

November 15, 2025

Contents

1 Ejercicio 1:

1. Consideremos el modelo de espacio de estados definido en la Sección 1 del Tema 3:

$$x(k+1) = \Phi(k+1,k)x(k) + \Gamma(k+1,k)w(k), \quad k \geq 0; \quad x(0) = x_0$$
$$z(k) = H(k)x(k) + v(k), \quad k \geq 0$$

verificando las siguientes hipotesis:

- El estado inicial, x_0, es un vector aleatorio n-dimensional gaussiano con media cero y matriz de covarianzas $E[x_0 x_0^T] = P_0$.
- El proceso $\{w(k); k \geq 0\}$ es una sucesión ruido blanco gaussiana, centrada, con matrices de covarianzas $E[w(k)w^T(k)] = Q(k), k \geq 0$.
- El proceso $\{v(k); k \geq 0\}$ es un ruido blanco gaussiano, centrado y con covarianzas $E[v(k)v^T(k)] = R(k), k \geq 0$, siendo $R(k)$ una matriz definida positiva.
- El estado inicial, x_0, y los ruidos aditivos, $\{w(k); k \geq 0\}$ y $\{v(k); k \geq 0\}$, son mutuamente independientes.

Solución:

1.1 (a) Demostraciones de propiedades de esperanza cruzada

Para estas demostraciones, utilizaremos las ecuaciónes del modelo y las hipótesis 1-4.

Primero, establecemos que el estado $x(k)$ es un proceso centrado (media cero). $E[x(0)] = E[x_0] = 0$ (por Hipótesis 1). Para $k \geq 0$, $E[x(k+1)] = E[\Phi(k+1,k)x(k) + \Gamma(k+1,k)w(k)]$. Usando la linealidad de la esperanza: $E[x(k+1)] = \Phi(k+1,k)E[x(k)] + \Gamma(k+1,k)E[w(k)]$. Dado que $E[w(k)] = 0$ (Hipótesis 2), tenemos la recursión $E[x(k+1)] = \Phi(k+1,k)E[x(k)]$. Por inducción, partiendo de $E[x(0)] = 0$, se concluye que $E[x(j)] = 0$ para todo $j \geq 0$.

(a.1) Demostrar $E[x(j)w^T(k)] = 0, \quad j \leq k$

La solución a la ecuación de estado es:

$$x(j) = \Phi(j,0)x(0) + \sum_{i=1}^{j} \Phi(j,i)\Gamma(i,i-1)w(i-1)$$

El estado $x(j)$ es, por tanto, una función lineal del estado inicial x_0 y de los ruidos del proceso $\{w(0), \ldots, w(j-1)\}$.

Calculamos la esperanza $E[x(j)w^T(k)]$:

$$E[x(j)w^T(k)] = E\left[\left(\Phi(j,0)x_0 + \sum_{i=1}^{j} \Phi(j,i)\Gamma(i,i-1)w(i-1)\right) w^T(k)\right]$$

$$E[x(j)w^T(k)] = \Phi(j,0)E[x_0 w^T(k)] + \sum_{i=1}^{j} \Phi(j,i)\Gamma(i,i-1)E[w(i-1)w^T(k)]$$

Analizamos los términos de la esperanza:

1. $E[x_0 w^T(k)]$: Por la Hipótesis 4, x_0 y el proceso $\{w(k)\}$ son independientes.

$$E[x_0 w^T(k)] = E[x_0]E[w^T(k)]$$

Por Hipótesis 1, $E[x_0] = 0$, y por Hipótesis 2, $E[w(k)] = 0$.

$$E[x_0 w^T(k)] = 0 \cdot 0^T = 0$$

2. $E[w(i-1)w^T(k)]$: Por la Hipótesis 2, $\{w(k)\}$ es una sucesión ruido blanco. Esto implica que $E[w(m)w^T(n)] = 0$ si $m \neq n$. En nuestra suma, i toma valores de 1 a j. Por tanto, $i-1$ toma valores de 0 a $j-1$. La condición del problema es $j \leq k$.

   - Si $j < k$, entonces $i - 1 < k$ para todos los i en la suma. Así, $E[w(i-1)w^T(k)] = 0$.
   - Si $j = k$, la suma va hasta $i - 1 = k - 1$. De nuevo, $i - 1 \neq k$ para todos los i. Así, $E[w(i-1)w^T(k)] = 0$.

   En ambos casos ($j \leq k$), el término de la esperanza es cero.

Sustituyendo estos resultados:

$$E[x(j)w^T(k)] = 0 + \sum_{i=1}^{j} \Phi(j,i)\Gamma(i,i-1)(0) = 0$$

Queda demostrado (a.1).

(a.2) Demostrar $E[z(j)w^T(k)] = 0, \quad j \leq k$

Usamos la ecuación de observación $z(j) = H(j)x(j) + v(j)$.

$$E[z(j)w^T(k)] = E[(H(j)x(j) + v(j))w^T(k)]$$

$$E[z(j)w^T(k)] = H(j)E[x(j)w^T(k)] + E[v(j)w^T(k)]$$

Analizamos los dos términos:

1. $H(j)E[x(j)w^T(k)]$: Por (a.1), que acabamos de demostrar, $E[x(j)w^T(k)] = 0$ para $j \leq k$. Este término es 0.
2. $E[v(j)w^T(k)]$: Por la Hipótesis 4, los procesos $\{v(k)\}$ y $\{w(k)\}$ son mutuamente independientes.

   $$E[v(j)w^T(k)] = E[v(j)]E[w^T(k)]$$

   Por Hipótesis 3, $E[v(j)] = 0$, y por Hipótesis 2, $E[w(k)] = 0$.

   $$E[v(j)w^T(k)] = 0 \cdot 0^T = 0$$

Sustituyendo estos resultados:

$$E[z(j)w^T(k)] = 0 + 0 = 0$$

Queda demostrado (a.2).

(a.3) Demostrar $E[x(j)v^T(k)] = 0, \quad j, k \geq 0$

Como en (a.1), $x(j)$ depende de x_0 y $\{w(0), \ldots, w(j-1)\}$. El vector $v(k)$ pertenece al proceso $\{v(k)\}$. Por Hipótesis 4, el proceso $\{v(k)\}$ es independiente tanto de x_0 como del proceso $\{w(k)\}$. Por lo tanto, $v(k)$ es independiente de x_0 y de $\{w(0), \ldots, w(j-1)\}$, y en consecuencia, $v(k)$ es independiente de $x(j)$ para cualesquiera j, k.

$$E[x(j)v^T(k)] = E[x(j)]E[v^T(k)]$$

Como establecimos al inicio, $E[x(j)] = 0$. Por Hipótesis 3, $E[v(k)] = 0$.

$$E[x(j)v^T(k)] = 0 \cdot 0^T = 0$$

Queda demostrado (a.3).

(a.4) Demostrar $E[z(j)v^T(k)] = 0, \quad j \neq k$

Usamos de nuevo la ecuación de observación $z(j) = H(j)x(j) + v(j)$.

$$E[z(j)v^T(k)] = E[(H(j)x(j) + v(j))v^T(k)]$$

$$E[z(j)v^T(k)] = H(j)E[x(j)v^T(k)] + E[v(j)v^T(k)]$$

Analizamos los dos términos:

1. $H(j)E[x(j)v^T(k)]$: Por (a.3), que acabamos de demostrar, $E[x(j)v^T(k)] = 0$ para todos j, k. Este término es 0.
2. $E[v(j)v^T(k)]$: Por la Hipótesis 3, $\{v(k)\}$ es un proceso ruido blanco. Por definición de ruido blanco, sus valores en instantes distintos no están correlacionados. Dado que $j \neq k$,

$$E[v(j)v^T(k)] = 0$$

Sustituyendo estos resultados:

$$E[z(j)v^T(k)] = 0 + 0 = 0$$

Queda demostrado (a.4).

1.2 (b) Demostración de la recursividad del estimador

Queremos demostrar la fórmula recursiva para el estimador $\hat{x}(k/j) = E[x(k)|Z_j]$, donde $Z_j = \{z(0), \ldots, z(j)\}$ es el conjunto de observaciones.

La demostración se basa en el Lema de Proyección Ortogonal (LPO), que es el fundamento de la estimación lineal mínimo-cuadrática.
1. Definimos el estimador en el instante $j-1$ como $\hat{x}(k/j-1) = E[x(k)|Z_{j-1}]$.
2. Definimos la innovación en el instante j como $\tilde{z}(j/j-1) = z(j) - \hat{z}(j/j-1)$, donde $\hat{z}(j/j-1) = E[z(j)|Z_{j-1}]$. Esta innovación representa la "nueva" información contenida en $z(j)$ que no está presente en Z_{j-1}.
3. Por el LPO, la innovación $\tilde{z}(j/j-1)$ es ortogonal a todo el espacio vectorial generado por las observaciones anteriores, Z_{j-1}.
4. El espacio de observaciones Z_j puede descomponerse en la suma de dos subespacios ortogonales: el espacio Z_{j-1} y el espacio generado por la innovación $\{\tilde{z}(j/j-1)\}$.

$$Z_j = Z_{j-1} \oplus \{\tilde{z}(j/j-1)\}$$

5. Dado que $\hat{x}(k/j)$ es la proyección de $x(k)$ sobre Z_j, esta puede calcularse como la suma de las proyecciones de $x(k)$ sobre estos dos subespacios ortogonales:

$$\hat{x}(k/j) = \text{Proj}(x(k)|Z_j) = \text{Proj}(x(k)|Z_{j-1}) + \text{Proj}(x(k)|\tilde{z}(j/j-1))$$

6. Identificamos los términos de la ecuación anterior:

- $\text{Proj}(x(k)|Z_{j-1}) = \hat{x}(k/j-1)$ (por definicion).
- La proyección de un vector $x(k)$ sobre otro vector $\tilde{z}$ es una transformación lineal de $\tilde{z}$, es decir, $K \cdot \tilde{z}$.

7. Por lo tanto, existe una matriz de ganancia $K(k,j)$ tal que:

$$\text{Proj}(x(k)|\tilde{z}(j/j-1)) = K(k,j)\tilde{z}(j/j-1)$$

(Esta ganancia $K(k,j)$ sería $E[x(k)\tilde{z}^T(j/j-1)](E[\tilde{z}(j/j-1)\tilde{z}^T(j/j-1)])^{-1}$).
8. Sustituyendo los términos en la ecuación de proyección:

$$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)\tilde{z}(j/j-1)$$

9. Finalmente, sustituyendo la definición de la innovación $\tilde{z}(j/j-1)$:

$$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)[z(j) - \hat{z}(j/j-1)], \quad j > 0$$

Queda demostrado.

1.3 (c) Determinación y propiedades del proceso innovación

El proceso innovación se define como $\tilde{z}(k/k-1) = z(k) - \hat{z}(k/k-1)$ para $k \geq 0$.

1. **Gaussiano:** El estado inicial x_0 y los ruidos $w(k)$ y $v(k)$ son gaussianos (Hipótesis 1-3). El estado $x(k)$ es una combinación lineal de x_0 y $\{w(i)\}_{i=0}^{k-1}$, por lo que es gaussiano. La observación $z(k) = H(k)x(k) + v(k)$ es una suma de procesos gaussianos, por lo que es gaussiana. El conjunto $Z_{k-1} = \{z(0), \ldots, z(k-1)\}$ es un conjunto de vectores gaussianos. El estimador $\hat{z}(k/k-1) = E[z(k)|Z_{k-1}]$, al ser una esperanza condiciónada en un sistema gaussiano, es una combinacion lineal de Z_{k-1} y, por tanto, es gaussiano. $\tilde{z}(k/k-1)$ es la diferencia de dos vectores gaussianos ($z(k)$ y $\hat{z}(k/k-1)$), por lo que $\tilde{z}(k/k-1)$ es un proceso gaussiano.

2. **Centrado (Media Cero):** $E[\tilde{z}(k/k-1)] = E[z(k) - \hat{z}(k/k-1)] = E[z(k)] - E[\hat{z}(k/k-1)]$. Como se demostró en (a), $E[x(k)] = 0$. Entonces $E[z(k)] = E[H(k)x(k) + v(k)] = H(k)E[x(k)] + E[v(k)] = 0 + 0 = 0$. Por la propiedad de la esperanza iterada, $E[\hat{z}(k/k-1)] = E[E[z(k)|Z_{k-1}]] = E[z(k)] = 0$. Por tanto, $E[\tilde{z}(k/k-1)] = 0 - 0 = 0$. El proceso es centrado.

3. **Blanco:** Debemos demostrar que $E[\tilde{z}(k/k-1)\tilde{z}^T(j/j-1)] = 0$ para $j \neq k$. Supongamos $j < k$. Por definición, $\tilde{z}(j/j-1) = z(j) - E[z(j)|Z_{j-1}]$. Este vector es una función (lineal) de las observaciones $Z_j = \{z(0), \ldots, z(j)\}$. Por el LPO, la innovación $\tilde{z}(k/k-1)$ es ortogonal al espacio Z_{k-1}. Dado que $j < k$, se cumple que $j \leq k-1$, y por tanto $Z_j \subseteq Z_{k-1}$. Como $\tilde{z}(j/j-1)$ pertenece a Z_{k-1} (es una combinación de sus elementos), y $\tilde{z}(k/k-1)$ es ortogonal a Z_{k-1}, entonces $\tilde{z}(k/k-1)$ es ortogonal a $\tilde{z}(j/j-1)$.

$$E[\tilde{z}(k/k-1)\tilde{z}^T(j/j-1)] = 0, \quad \text{para } j < k$$

   Por simetría (transponiendo la matriz), $E[\tilde{z}(j/j-1)\tilde{z}^T(k/k-1)] = 0$ para $j > k$. El proceso es, por tanto, un ruido blanco.

4. **Matriz de Covarianzas $\Pi(k)$:** $\Pi(k) = E[\tilde{z}(k/k-1)\tilde{z}^T(k/k-1)]$. Usamos la expresión de la innovación $\tilde{z}(k/k-1) = z(k) - \hat{z}(k/k-1)$. Sustituimos $z(k) = H(k)x(k) + v(k)$ y $\hat{z}(k/k-1) = H(k)\hat{x}(k/k-1)$:

$$\tilde{z}(k/k-1) = (H(k)x(k) + v(k)) - H(k)\hat{x}(k/k-1)$$

$$\tilde{z}(k/k-1) = H(k)(x(k) - \hat{x}(k/k-1)) + v(k)$$

$$\tilde{z}(k/k-1) = H(k)\tilde{x}(k/k-1) + v(k)$$

   donde $\tilde{x}(k/k-1)$ es el error de predicción. Calculamos la covarianza:

$$\Pi(k) = E[(H(k)\tilde{x}(k/k-1) + v(k))(H(k)\tilde{x}(k/k-1) + v(k))^T]$$

$$\Pi(k) = E[H(k)\tilde{x}(k/k-1)\tilde{x}^T(k/k-1)H^T(k) + H(k)\tilde{x}(k/k-1)v^T(k) + v(k)\tilde{x}^T(k/k-1)H^T(k) + v(k)v^T(k)]$$

   El error de predicción $\tilde{x}(k/k-1)$ es función de x_0 y $\{w(i), v(i)\}_{i=0}^{k-1}$. Por la Hipótesis 4, $v(k)$ es independiente de todos ellos. Al ser centrados, los términos cruzados $E[H(k)\tilde{x}(k/k-1)v^T(k)]$ y su transpuesto son nulos.

$$\Pi(k) = H(k)E[\tilde{x}(k/k-1)\tilde{x}^T(k/k-1)]H^T(k) + E[v(k)v^T(k)]$$

   Usando la definición $P(k/k-1) = E[\tilde{x}(k/k-1)\tilde{x}^T(k/k-1)]$ y $R(k) = E[v(k)v^T(k)]$:

$$\Pi(k) = H(k)P(k/k-1)H^T(k) + R(k), \quad k > 0$$

Caso $k = 0$: $\tilde{z}(0/-1) = z(0) - \hat{z}(0/-1)$. La estimación sin medidas $\hat{z}(0/-1) = E[z(0)] = 0$.

$$\tilde{z}(0/-1) = z(0) = H(0)x_0 + v(0)$$

$$\Pi(0) = E[(H(0)x_0 + v(0))(H(0)x_0 + v(0))^T]$$

$$\Pi(0) = E[H(0)x_0x_0^TH^T(0) + H(0)x_0v^T(0) + v(0)x_0^TH^T(0) + v(0)v^T(0)]$$

Por Hipótesis 4, x_0 y $v(0)$ son independientes y centrados, los términos cruzados son nulos.

$$\Pi(0) = H(0)E[x_0x_0^T]H^T(0) + E[v(0)v^T(0)]$$

$$\Pi(0) = H(0)P_0H^T(0) + R(0)$$

(Esto coincide con la fórmula general, ya que $P(0/-1) = P_0$).

1.4 (d) Demostración del algoritmo de filtrado de Kalman

El algoritmo de filtrado de Kalman es un conjunto de ecuaciones recursivas que proporcionan el estimador óptimo $\hat{x}(k/k)$ y su matriz de covarianzas del error $P(k/k)$. La demostración se articula en dos fases: una predicción temporal (de $k-1$ a k) y una actualización de medida (en k).

1. Ecuaciones del Ciclo de Tiempo (Predicción)

- **Predicción del estado $\hat{x}(k/k-1)$:** Por definición, $\hat{x}(k/k-1) = E[x(k)|Z_{k-1}]$. Usando la ecuación de estado:

$$\hat{x}(k/k-1) = E[\Phi(k,k-1)x(k-1) + \Gamma(k,k-1)w(k-1)|Z_{k-1}]$$

$$= \Phi(k,k-1)E[x(k-1)|Z_{k-1}] + \Gamma(k,k-1)E[w(k-1)|Z_{k-1}]$$

$E[w(k-1)|Z_{k-1}] = 0$ (Hipótesis 2 y 4). Por tanto:

$$\hat{x}(k/k-1) = \Phi(k,k-1)\hat{x}(k-1/k-1)$$

- **Predicción de la covarianza $P(k/k-1)$:** Definimos el error de predicción $\tilde{x}(k/k-1) = x(k) - \hat{x}(k/k-1)$.

$$\tilde{x}(k/k-1) = (\Phi(k,k-1)x(k-1) + \Gamma(\dots)w(k-1)) - \Phi(k,k-1)\hat{x}(k-1/k-1)$$

$$\tilde{x}(k/k-1) = \Phi(k,k-1)\tilde{x}(k-1/k-1) + \Gamma(k,k-1)w(k-1)$$

$P(k/k-1) = E[\tilde{x}(k/k-1)\tilde{x}^T(k/k-1)]$. El error $\tilde{x}(k-1/k-1)$ es ortogonal a $w(k-1)$ (depende de Z_{k-1}). Los terminos cruzados son nulos.

$$P(k/k-1) = \Phi(\dots)E[\tilde{x}(k-1/k-1)\tilde{x}^T(k-1/k-1)]\Phi^T(\dots) + \Gamma(\dots)E[w(k-1)w^T(k-1)]\Gamma^T(\dots)$$

$$P(k/k-1) = \Phi(k,k-1)P(k-1/k-1)\Phi^T(k,k-1) + \Gamma(k,k-1)Q(k-1)\Gamma^T(k,k-1)$$

Esto coincide con la ecuación (2) del Teorema 2.

2. Ecuaciones del Ciclo de Medida

- **Ecuación del filtro $\hat{x}(k/k)$:** Como se demostró en (b) y se indica en los apuntes (p. 6), el filtro tiene la forma recursiva:

$$\hat{x}(k/k) = \hat{x}(k/k-1) + K(k)[z(k) - \hat{z}(k/k-1)]$$

  Donde $\hat{z}(k/k-1) = E[z(k)|Z_{k-1}] = E[H(k)x(k) + v(k)|Z_{k-1}] = H(k)\hat{x}(k/k-1)$. Sustituyendo $\hat{x}(k/k-1)$ de la predicción, obtenemos la forma del Teorema 2:

$$\hat{x}(k/k) = \Phi(k,k-1)\hat{x}(k-1/k-1) + K(k)[z(k) - H(k)\Phi(k,k-1)\hat{x}(k-1/k-1)]$$

- **Ganancia de Kalman $K(k)$:** La ganancia óptima $K(k)$ minimiza la varianza del error y viene dada por $K(k) = E[x(k)\tilde{z}^T(k/k-1)](E[\tilde{z}(k/k-1)\tilde{z}^T(k/k-1)])^{-1}$.
  - Denominador: Es la covarianza de la innovación $\Pi(k)$, demostrada en (c): $\Pi(k) = H(k)P(k/k-1)H^T(k) + R(k)$.
  - Numerador: $E[x(k)\tilde{z}^T(k/k-1)] = E[x(k)(H(k)\tilde{x}(k/k-1) + v(k))^T]$. $E[x(k)v^T(k)] = 0$ por (a.3). $E[x(k)\tilde{z}^T(k/k-1)] = E[x(k)\tilde{x}^T(k/k-1)]H^T(k)$. $E[x(k)\tilde{x}^T(k/k-1)] = E[(\hat{x}(k/k-1) + \tilde{x}(k/k-1))\tilde{x}^T(k/k-1)]$. $E[\hat{x}(k/k-1)\tilde{x}^T(k/k-1)] = 0$ por ortogonalidad (LPO). $E[x(k)\tilde{x}^T(k/k-1)] = E[\tilde{x}(k/k-1)\tilde{x}^T(k/k-1)] = P(k/k-1)$.

  Juntando ambos:

$$K(k) = P(k/k-1)H^T(k)[H(k)P(k/k-1)H^T(k) + R(k)]^{-1}$$

  Esto coincide con la ecuación (1) del Teorema 2.

- **Actualización de la covarianza $P(k/k)$:** Definimos el error de filtrado $\tilde{x}(k/k) = x(k) - \hat{x}(k/k)$.

$$\tilde{x}(k/k) = x(k) - (\hat{x}(k/k-1) + K(k)\tilde{z}(k/k-1)) = \tilde{x}(k/k-1) - K(k)\tilde{z}(k/k-1)$$

$$\tilde{x}(k/k) = \tilde{x}(k/k-1) - K(k)(H(k)\tilde{x}(k/k-1) + v(k))$$

$$\tilde{x}(k/k) = [I - K(k)H(k)]\tilde{x}(k/k-1) - K(k)v(k)$$

  $P(k/k) = E[\tilde{x}(k/k)\tilde{x}^T(k/k)]$. $\tilde{x}(k/k-1)$ y $v(k)$ son ortogonales.

$$P(k/k) = [I - K(k)H(k)]P(k/k-1)[I - K(k)H(k)]^T + K(k)R(k)K^T(k)$$

  Expandiendo:

$$P(k/k) = P(k/k-1) - K(k)H(k)P(k/k-1) - P(k/k-1)H^T(k)K^T(k) + K(k)[H(k)P(k/k-1)H^T(k) + R(k)]K^T(k)$$

  Sustituyendo $K(k)[\dots] = P(k/k-1)H^T(k)$ de la ecuación de ganancia:

$$P(k/k) = P(k/k-1) - K(k)H(k)P(k/k-1) - P(k/k-1)H^T(k)K^T(k) + P(k/k-1)H^T(k)K^T(k)$$

$$P(k/k) = P(k/k-1) - K(k)H(k)P(k/k-1) = [I - K(k)H(k)]P(k/k-1)$$

  Esto coincide con la ecuación (3) del Teorema 2.

3. Condiciones Iniciales

- $\hat{x}(0/-1) = E[x(0)|\text{sin medidas}] = E[x_0] = 0$.
- $P(0/-1) = E[(x_0 - \hat{x}(0/-1))(x_0 - \hat{x}(0/-1))^T] = E[x_0 x_0^T] = P_0$.

Queda demostrado el algoritmo.

1.5 (e) Demostración del algoritmo de suavizamiento punto fijo

El objetivo es encontrar una recursión para $\hat{x}(k/j)$ con k fijo y $j = k+1, k+2, \dots$ La recursión se basa en la del apartado (b):

$$\hat{x}(k/j) = \hat{x}(k/j-1) + K(k,j)[z(j) - \hat{z}(j/j-1)], \quad j > k$$

La condición inicial es $\hat{x}(k/k)$. Necesitamos obtener la ganancia $K(k,j)$ y la covarianza del error $P(k/j)$.

1. Ganancia del Suavizador $K(k,j)$

La ganancia óptima es $K(k,j) = E[x(k)\tilde{z}^T(j/j-1)]\Pi(j)^{-1}$, donde $\Pi(j) = E[\tilde{z}(j/j-1)\tilde{z}^T(j/j-1)]$. $\Pi(j) = H(j)P(j/j-1)H^T(j) + R(j)$ (demostrado en (c)).

Analizamos el numerador $E[x(k)\tilde{z}^T(j/j-1)]$:

$$\tilde{z}(j/j-1) = H(j)\tilde{x}(j/j-1) + v(j)$$

$$E[x(k)\tilde{z}^T(j/j-1)] = E[x(k)(H(j)\tilde{x}(j/j-1) + v(j))^T]$$

$E[x(k)v^T(j)] = 0$ por (a.3), ya que $k < j$.

$$E[x(k)\tilde{z}^T(j/j-1)] = E[x(k)\tilde{x}^T(j/j-1)]H^T(j)$$

Definamos $L(k,j) = E[x(k)\tilde{x}^T(j/j)]$.

$$\tilde{x}(j/j-1) = \Phi(j,j-1)\tilde{x}(j-1/j-1) + \Gamma(\dots)w(j-1)$$

$$E[x(k)\tilde{x}^T(j/j-1)] = E[x(k)(\Phi(\dots)\tilde{x}(j-1/j-1) + \Gamma(\dots)w(j-1))^T]$$

$E[x(k)w^T(j-1)] = 0$ por (a.1), ya que $k \leq j-1$.

$$E[x(k)\tilde{x}^T(j/j-1)] = E[x(k)\tilde{x}^T(j-1/j-1)]\Phi^T(j,j-1) = L(k,j-1)\Phi^T(j,j-1)$$

Sustituyendo esto en la ganancia:

$$K(k,j) = L(k,j-1)\Phi^T(j,j-1)H^T(j)\Pi(j)^{-1}$$

Esto coincide con la ecuación de la ganancia.

2. Recursión de $L(k,j)$

Necesitamos una recursión para $L(k,j) = E[x(k)\tilde{x}^T(j/j)]$.

$$\tilde{x}(j/j) = [I - K(j)H(j)]\tilde{x}(j/j-1) - K(j)v(j)$$

$$L(k,j) = E[x(k)([I - K(j)H(j)]\tilde{x}(j/j-1) - K(j)v(j))^T]$$

$E[x(k)v^T(j)] = 0$ por (a.3).

$$L(k,j) = E[x(k)\tilde{x}^T(j/j-1)][I - K(j)H(j)]^T$$

Usando $E[x(k)\tilde{x}^T(j/j-1)] = L(k,j-1)\Phi^T(j,j-1)$ del paso anterior:

$$L(k,j) = L(k,j-1)\Phi^T(j,j-1)[I - K(j)H(j)]^T$$

Condición inicial $L(k,k)$: $L(k,k) = E[x(k)\tilde{x}^T(k/k)] = E[(\hat{x}(k/k) + \tilde{x}(k/k))\tilde{x}^T(k/k)]$
Por ortogonalidad, $E[\hat{x}(k/k)\tilde{x}^T(k/k)] = 0$. $L(k,k) = E[\tilde{x}(k/k)\tilde{x}^T(k/k)] = P(k/k)$.

3. Matriz de Covarianzas $P(k/j)$

$P(k/j) = E[\tilde{x}(k/j)\tilde{x}^T(k/j)]$.

$$\tilde{x}(k/j) = x(k) - \hat{x}(k/j) = x(k) - (\hat{x}(k/j-1) + K(k,j)\tilde{z}(j/j-1))$$

$$\tilde{x}(k/j) = \tilde{x}(k/j-1) - K(k,j)\tilde{z}(j/j-1)$$

$P(k/j) = E[(\tilde{x}(k/j-1) - K(k,j)\tilde{z}(j/j-1))(\dots)^T]$

$P(k/j) = P(k/j-1) - E[\tilde{x}(k/j-1)\tilde{z}^T(j/j-1)]K^T(k,j) - K(k,j)E[\tilde{z}(j/j-1)\tilde{x}^T(k/j-1)] + K(k,j)\Pi(j)K^T(k,j)$

$E[\tilde{x}(k/j-1)\tilde{z}^T(j/j-1)] = E[x(k)\tilde{z}^T(j/j-1)]$ (ya que $\hat{x}(k/j-1)$ es ortogonal a $\tilde{z}(j/j-1)$). De la ecuación de ganancia, $E[x(k)\tilde{z}^T(j/j-1)] = K(k,j)\Pi(j)$.

$$P(k/j) = P(k/j-1) - (K(k,j)\Pi(j))K^T(k,j) - K(k,j)(K(k,j)\Pi(j))^T + K(k,j)\Pi(j)K^T(k,j)$$

$$P(k/j) = P(k/j-1) - K(k,j)\Pi(j)K^T(k,j)$$

Sustituyendo $\Pi(j)$:

$$P(k/j) = P(k/j-1) - K(k,j)[H(j)P(j/j-1)H^T(j) + R(j)]K^T(k,j)$$

Queda demostrado el algoritmo.

1.6 (f) Ciclo computacional y programa en R del suavizador punto fijo

1. Ciclo Computacional

El objetivo es obtener $\hat{x}(k/j)$ para un k fijo, a medida que j avanza ($j = k+1, \dots, k+N$). Esto requiere una recursión en j. Las ecuaciones de esta recursión se encuentran en la Seccion 4.1 de los apuntes.

El ciclo computacional completo requiere dos fases:

Fase 1: Ejecución del Filtro de Kalman (Hacia Adelante)

Antes de poder suavizar en el punto k, es necesario procesar todas las observaciones hasta el instante final $k+N$.

1. Ejecutar el ciclo completo del Filtro de Kalman (Teorema 2) desde el instante $i = 0$ hasta $i = k+N$.
2. Durante esta ejecución, almacenar todos los valores relevantes que se necesitarán después:
   - Las innovaciones: $\tilde{z}(i/i-1) = z(i) - H(i)\hat{x}(i/i-1)$, para $i = k+1, \dots, k+N$.
   - Las covarianzas de la innovación: $\Pi(i) = H(i)P(i/i-1)H^T(i) + R(i)$, para $i = k+1, \dots, k+N$.
   - Las ganancias del filtro: $K(i)$, para $i = k+1, \dots, k+N$.
   - El filtro y su covarianza en k: $\hat{x}(k/k)$ y $P(k/k)$.
   - Las matrices del sistema $\Phi(i, i-1)$ y $H(i)$ para $i = k+1, \dots, k+N$.

Fase 2: Ejecución del Suavizador Punto Fijo (recursión en j)

Esta fase implementa las ecuaciones de la Seccion 4.1.

1. Inicialización (en $j = k$):
   - Establecer el estimador inicial: $\hat{x}_{smooth}(k/k) = \hat{x}(k/k)$ (obtenido del filtro).
   - Establecer la covarianza inicial: $P_{smooth}(k/k) = P(k/k)$ (obtenida del filtro).
   - Inicializar la matriz $L(k,k) = P(k/k)$.
2. Bucle Recursivo: Iniciar un bucle para j desde $k+1$ hasta $k+N$.

- a. Calcular la ganancia del suavizador $K(k, j)$:

$$K(k, j) = L(k, j-1) \cdot \Phi^T(j, j-1) \cdot H^T(j) \cdot \Pi(j)^{-1}$$

  (donde $\Pi(j)$ se calculó en la Fase 1).
- b. Actualizar el estimador de suavizamiento $\hat{x}(k/j)$:

$$\hat{x}_{smooth}(k/j) = \hat{x}_{smooth}(k/j-1) + K(k, j) \cdot \tilde{z}(j/j-1)$$

  (donde $\tilde{z}(j/j-1)$ se calculo en la Fase 1).
- c. Actualizar la covarianza de suavizamiento $P(k/j)$:

$$P_{smooth}(k/j) = P_{smooth}(k/j-1) - K(k, j) \cdot \Pi(j) \cdot K^T(k, j)$$

- d. Actualizar la matriz L para la siguiente iteracion (para que sea $L(k, j)$):

$$L(k, j) = L(k, j-1) \cdot \Phi^T(j, j-1) \cdot [I - K(j)H(j)]^T$$

  (donde $K(j)$ se calculo en la Fase 1).

3. Fin del Bucle. El resultado es la secuencia de estimadores $\hat{x}(k/k), \hat{x}(k/k+1), \ldots, \hat{x}(k/k+N)$, que refinan la estimacion en el punto k a medida que hay mas datos disponibles.

2. Programa en R

El siguiente codigo R implementa la simulación del modelo, el Filtro de Kalman (Fase 1) y el Suavizador Punto Fijo (Fase 2) para el sistema escalar especificado.

```
# 0. Definicion de Parametros y simulación
Phi <- 0.95
H   <- 1
Q   <- 0.1
R   <- 0.5
P0  <- 1
x0_media <- 0
k_fixed   <- 20
N_horizon <- 30
N_total   <- k_fixed + N_horizon
N_steps <- N_total + 1

# 1. Simulación del Sistema
x_true <- numeric(N_steps)
z_obs  <- numeric(N_steps)
x_true[1] <- x0_media + sqrt(P0) * rnorm(1)
z_obs[1]  <- H * x_true[1] + sqrt(R) * rnorm(1)
for (i in 2:N_steps) {
  x_true[i] <- Phi * x_true[i-1] + sqrt(Q) * rnorm(1)
  z_obs[i]  <- H * x_true[i] + sqrt(R) * rnorm(1)
}

# 2. Fase 1: Ejecucion del Filtro de Kalman
x_pred <- numeric(N_steps)
P_pred <- numeric(N_steps)
x_fil  <- numeric(N_steps)
```

```
P_fil  <- numeric(N_steps)
K_gain <- numeric(N_steps)
innov  <- numeric(N_steps)
Pi_cov <- numeric(N_steps)
x_pred[1] <- x0_media
P_pred[1] <- P0
Pi_cov[1] <- H * P_pred[1] * H + R
K_gain[1] <- P_pred[1] * H / Pi_cov[1]
innov[1]  <- z_obs[1] - H * x_pred[1]
x_fil[1]  <- x_pred[1] + K_gain[1] * innov[1]
P_fil[1]  <- (1 - K_gain[1] * H) * P_pred[1]
for (i in 2:N_steps) {
  x_pred[i] <- Phi * x_fil[i-1]
  P_pred[i] <- Phi * P_fil[i-1] * Phi + Q
  Pi_cov[i] <- H * P_pred[i] * H + R
  K_gain[i] <- P_pred[i] * H / Pi_cov[i]
  innov[i]  <- z_obs[i] - H * x_pred[i]
  x_fil[i]  <- x_pred[i] + K_gain[i] * innov[i]
  P_fil[i]  <- (1 - K_gain[i] * H) * P_pred[i]
}

# 3. Fase 2: Suavizador Punto Fijo
i_fixed <- k_fixed + 1
j_start_idx <- i_fixed + 1
j_end_idx   <- N_steps
n_smooth_steps <- N_horizon + 1
x_smooth <- numeric(n_smooth_steps)
P_smooth <- numeric(n_smooth_steps)
x_smooth[1] <- x_fil[i_fixed]
P_smooth[1] <- P_fil[i_fixed]
L <- P_fil[i_fixed]
s_idx <- 1
for (j_idx in j_start_idx:j_end_idx) {
  s_idx <- s_idx + 1
  K_smooth_j <- L * Phi * H / Pi_cov[j_idx]
  x_smooth[s_idx] <- x_smooth[s_idx-1] + K_smooth_j * innov[j_idx]
  P_smooth[s_idx] <- P_smooth[s_idx-1] - K_smooth_j * Pi_cov[j_idx] * K_smooth_j
  L <- L * Phi * (1 - K_gain[j_idx] * H)
}

# 4. Impresion de Resultados
print(paste("Estado real en k =", k_fixed, ":", x_true[i_fixed]))
print(paste("Estimacion filtrada x_hat(k/k):", x_smooth[1]))
print(paste("Estimacion suavizada x_hat(k/k+N):", x_smooth[n_smooth_steps]))
print("Varianza del error de filtrado P(k/k):")
print(P_smooth[1])
print("Varianza del error de suavizado P(k/k+N):")
print(P_smooth[n_smooth_steps])
```

```
> print(paste("Estado real en k =", k_fixed, ":", x_true[i_fixed]))
[1] "Estado real en k = 20 : 1.33685223632401"
> print(paste("Estimacion filtrada x_hat(k/k):", x_smooth[1]))
[1] "Estimacion filtrada x_hat(k/k): 1.10895711658274"
> print(paste("Estimacion suavizada x_hat(k/k+N):", x_smooth[n_smooth_steps]))
[1] "Estimacion suavizada x_hat(k/k+N): 1.46659730582783"
> print("Varianza del error de filtrado P(k/k):")
[1] "Varianza del error de filtrado P(k/k):"
> print(P_smooth[1])
[1] 0.1669754
> print("Varianza del error de suavizado P(k/k+N):")
[1] "Varianza del error de suavizado P(k/k+N):"
> print(P_smooth[n_smooth_steps])
[1] 0.1110764
>
```

Figure 1: Resultados para el suavizador punto fijo.

Comentario sobre los Resultados de R

Los resultados obtenidos de forma práctica el correcto funcionamiento de los algoritmos de filtrado y suavizado.

1. Calidad de las Estimaciones (Valores de $\hat{x}$)

El objetivo es que la estimación ($\hat{x}$) se aproxime lo máximo posible al estado real (x_{true}).

- **Estado Real (x_{true}):** 1.3368
- **Estimación Filtrada ($\hat{x}(k/k)$):** 1.1089
- **Estimación Suavizada ($\hat{x}(k/k+N)$):** 1.4665

Análisis: El filtro (con informacion hasta $k = 20$) tuvo un error de ≈ 0.228. El suavizador, tras procesar las $N = 30$ observaciones futuras, corrigió la estimación para $k = 20$, resultando en 1.4665. Esta nueva estimación tiene un error de ≈ 0.130, casi la mitad que el error del filtro.

Conclusión: La estimación suavizada es significativamente mejor (más cercana al valor real). Esto demuestra el concepto clave: el suavizador usa la información del "futuro" para corregir y refinar la estimación del "pasado".

2. Reducción de la Incertidumbre (Varianzas P)

Las varianzas (P) miden la incertidumbre teórica del algoritmo sobre sus propias estimaciones. Un valor más bajo implica mayor confianza.

- **Varianza del Filtro ($P(k/k)$):** 0.1669
- **Varianza del Suavizador ($P(k/k+N)$):** 0.1110

Análisis: El hecho de que $P(k/k+N) < P(k/k)$ (es decir, $0.1110 < 0.1669$) es la demostración teórica de que el suavizador es superior.

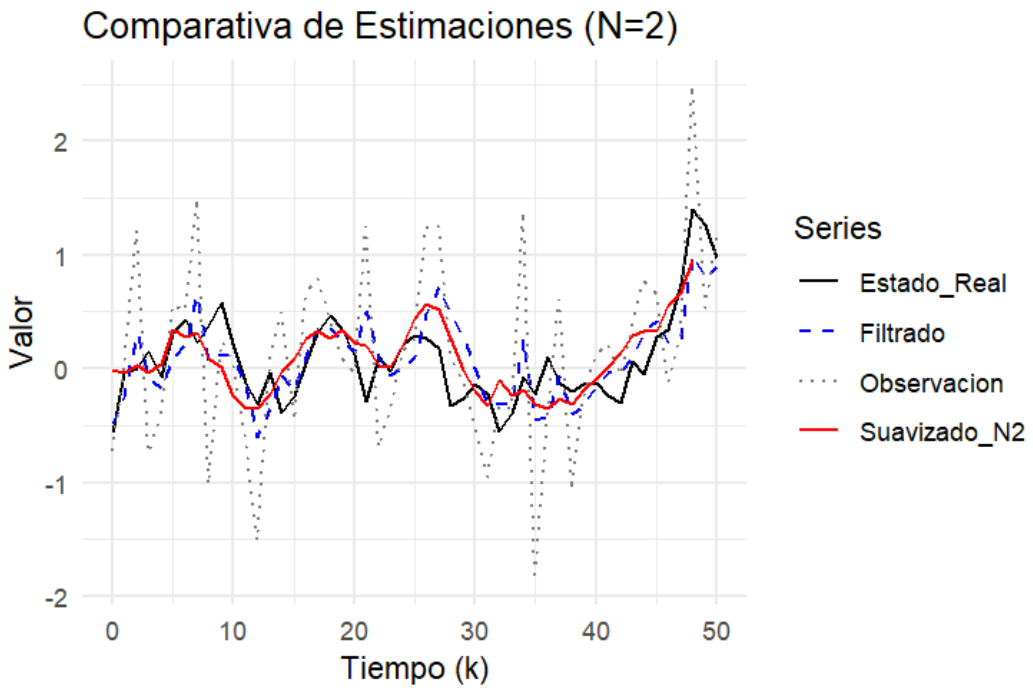
Conclusión: Al incorporar más información (las observaciones de $j = k+1$ a $k+N$), la incertidumbre sobre el estado en $k = 20$ disminuye. Los resultados teóricos (las varianzas) concuerdan perfectamente con los resultados prácticos (las estimaciones).

1.7 (g) Gráficas del Filtro y Suavizador Punto Fijo

```
library(ggplot2)
library(tidyr)

# 0. Definicion de parametros y simulación
Phi <- 0.95; H <- 1; Q <- 0.1; R <- 0.5; P0 <- 1; x0_media <- 0
N_iter <- 50; N_steps <- N_iter + 1
N_horizons <- c(1, 2, 4); N_max <- max(N_horizons)
time <- 0:N_iter
set.seed(123)
x_true <- numeric(N_steps); z_obs <- numeric(N_steps)
x_true[1] <- x0_media + sqrt(P0) * rnorm(1)
z_obs[1]  <- H * x_true[1] + sqrt(R) * rnorm(1)
for (i in 2:N_steps) {
  x_true[i] <- Phi * x_true[i-1] + sqrt(Q) * rnorm(1)
  z_obs[i]  <- H * x_true[i] + sqrt(R) * rnorm(1)
}

# 1. Ejecucion del Filtro de Kalman
x_pred <- numeric(N_steps); P_pred <- numeric(N_steps)
x_fil  <- numeric(N_steps); P_fil  <- numeric(N_steps)
K_gain <- numeric(N_steps); innov  <- numeric(N_steps)
Pi_cov <- numeric(N_steps)
x_pred[1] <- x0_media; P_pred[1] <- P0
Pi_cov[1] <- H * P_pred[1] * H + R
K_gain[1] <- P_pred[1] * H / Pi_cov[1]
innov[1]  <- z_obs[1] - H * x_pred[1]
x_fil[1]  <- x_pred[1] + K_gain[1] * innov[1]
P_fil[1]  <- (1 - K_gain[1] * H) * P_pred[1]
for (i in 2:N_steps) {
  x_pred[i] <- Phi * x_fil[i-1]
  P_pred[i] <- Phi * P_fil[i-1] * Phi + Q
  Pi_cov[i] <- H * P_pred[i] * H + R
  K_gain[i] <- P_pred[i] * H / Pi_cov[i]
  innov[i]  <- z_obs[i] - H * x_pred[i]
  x_fil[i]  <- x_pred[i] + K_gain[i] * innov[i]
  P_fil[i]  <- (1 - K_gain[i] * H) * P_pred[i]
}

# 2. Ejecucion del Suavizador Punto Fijo
x_smooth <- matrix(NA, nrow = N_steps, ncol = length(N_horizons))
P_smooth <- matrix(NA, nrow = N_steps, ncol = length(N_horizons))
colnames(x_smooth) <- paste0("N=", N_horizons)
colnames(P_smooth) <- paste0("N=", N_horizons)
for (i in 1:N_steps) {
  x_smooth_current <- x_fil[i]; P_smooth_current <- P_fil[i]; L_current <- P_fil[i]
  for (j_offset in 1:N_max) {
    j_idx <- i + j_offset
    if (j_idx > N_steps) { break }
    K_smooth_j <- L_current * Phi * H / Pi_cov[j_idx]
```

```
    x_smooth_current <- x_smooth_current + K_smooth_j * innov[j_idx]
    P_smooth_current <- P_smooth_current - K_smooth_j * Pi_cov[j_idx] * K_smooth_j
    L_current <- L_current * Phi * (1 - K_gain[j_idx] * H)
    if (j_offset %in% N_horizons) {
      col_idx <- which(N_horizons == j_offset)
      x_smooth[i, col_idx] <- x_smooth_current
      P_smooth[i, col_idx] <- P_smooth_current
    }
  }
}

# 3. Preparacion de datos y graficacion
data_g1 <- data.frame(Tiempo = time, Estado_Real = x_true, Observacion = z_obs,
                      Filtrado = x_fil, Suavizado_N2 = x_smooth[, "N=2"])
data_g1_long <- tidyr::pivot_longer(data_g1, cols = -Tiempo,
                                    names_to = "Serie", values_to = "Valor")
g1 <- ggplot(data_g1_long, aes(x = Tiempo, y = Valor, color = Serie)) +
  geom_line(aes(linetype = Serie)) +
  scale_linetype_manual(values = c("Estado_Real" = "solid", "Observacion" = "dotted", "Filtrado" =
  "dashed", "Suavizado_N2" = "solid")) + scale_color_manual(values = c("Estado_Real" = "black",
  "Observacion" = "grey50", "Filtrado" = "blue", "Suavizado_N2" = "red")) +
  labs(title = "Comparativa de Estimaciones (N=2)", x = "Tiempo (k)", y = "Valor", color = "Series",
  linetype = "Series") +  theme_minimal()
data_g2 <- data.frame(Tiempo = time, Filtrado = P_fil, Suavizado_N1 = P_smooth[, "N=1"],
                      Suavizado_N2 = P_smooth[, "N=2"], Suavizado_N4 = P_smooth[, "N=4"])
data_g2_long <- tidyr::pivot_longer(data_g2, cols = -Tiempo,
                                    names_to = "Serie", values_to = "Varianza")
g2 <- ggplot(data_g2_long, aes(x = Tiempo, y = Varianza, color = Serie)) +
  geom_line() +
  labs(title = "Comparativa de Varianzas del Error", x = "Tiempo (k)", y = "Varianza del Error (P)",
  color = "Estimador") + coord_cartesian(ylim = c(0, NA)) +
  theme_minimal()
print(g1)
print(g2)
```

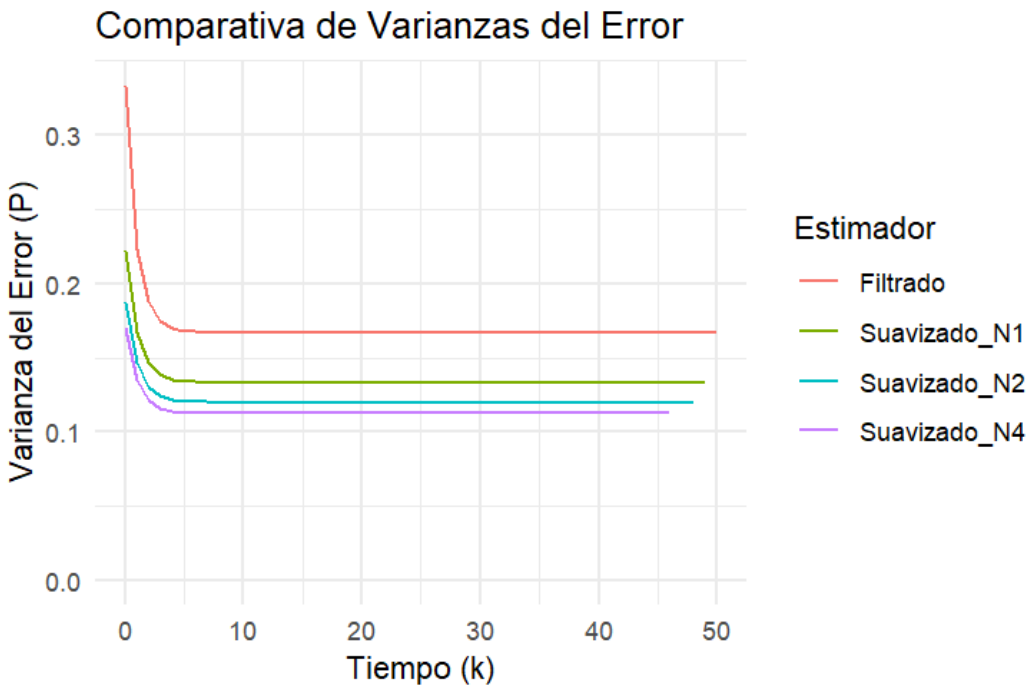
A continuación se comentan las dos gráficas generadas por el código R.



Esta gráfica compara las diferentes series temporales a lo largo de 50 iteraciones:

- **Observación** ($z(k)$): La línea gris punteada. Representa las mediciones reales, que están visiblemente afectadas por un alto nivel de ruido (varianza $R = 0.5$).
- **Estado_Real** ($x(k)$): La línea negra sólida. Es la trayectoria "verdadera" del sistema, la cual queremos estimar. Es un proceso más suave, pero está oculto por el ruido de la observación.
- **Filtrado** ($\hat{x}(k/k)$): La línea azul discontinua. Es la estimación del Filtro de Kalman. Se puede ver como filtra exitosamente el ruido de la observación, siguiendo la tendencia general del estado real. Sin embargo, muestra un ligero retraso (lag) en los cambios bruscos (p.ej., en la subida final cerca de $k = 50$).
- **Suavizado_N2** ($\hat{x}(k/k+2)$): La línea roja sólida. Es la estimación del suavizador punto fijo con un horizonte de $N = 2$. Esta estimación es notablemente superior a la del filtro; al utilizar dos observaciones futuras, puede "anticipar" los cambios, eliminando el retraso del filtro y ajustándose de forma más precisa a la trayectoria del estado real.

En resumen, la gráfica ilustra visualmente como el suavizador (rojo) corrige y refina la estimación del filtro (azul), proporcionando una reconstrucción más fiel del estado real (negro).



Esta gráfica muestra la evolución de la varianza teórica del error de estimación (P) para los distintos estimadores.

- Se observa que, tras un breve periodo transitorio inicial (aprox. $k = 0$ a $k = 5$), donde la incertidumbre disminuye drásticamente desde $P_0 = 1$, todos los estimadores alcanzan un **régimen estacionario**.
- Jerarquía de Varianza: La gráfica confirma perfectamente la teoría. La varianza del Filtrado (línea roja, $P(k/k)$) es la más alta, representando la mayor incertidumbre.
- A medida que se añaden observaciones futuras (aumenta N), la varianza del suavizador disminuye:

$$P(k/k) > P(k/k+1) > P(k/k+2) > P(k/k+4)$$

- Conclusión: Esto demuestra matemáticamente que cuanta más información "futura" se utiliza (mayor N), menor es la incertidumbre (varianza) sobre la estimación del estado en el instante k.
- Rendimiento Decreciente: Se aprecia que la mayor ganancia de precisión (reducción de varianza) se obtiene al pasar del filtro a $N = 1$. Las mejoras al aumentar a $N = 2$ y $N = 4$ son progresivamente más pequeñas, indicando un rendimiento decreciente.

2 Ejercicio 2:

Sea $\{x(k); k \geq 0\}$ un proceso estocástico escalar definido mediante la relación

$$x(k+1) = (-1)^{2^{k+1}} x(k), \quad k \geq 0$$

donde x_0 es una variable gaussiana con media cero y varianza $E[x_0^2] = P_0$. Supongamos que disponemos de observaciones de este proceso de la forma

$$z(k) = x(k) + v(k), \quad k \geq 0$$

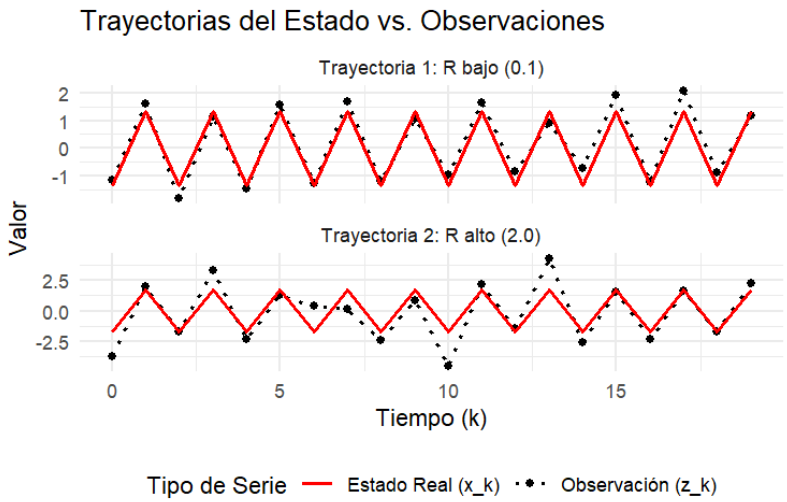
donde $\{v(k); k \geq 0\}$ es una sucesión blanca gaussiana, centrada y con varianzas $E[v^2(k)] = R, \quad k \geq 0$.

2.1 (a) Simulación de dos trayectorias del sistema con diferentes niveles de ruido

```
# 0. Cargar Librerias
library(ggplot2)
library(tidyr)

# 1. Definicion de Parametros
N_steps <- 20
k_values <- 0:(N_steps - 1)
P0_1 <- 1.0
R_1  <- 0.1
P0_2 <- 1.0
R_2  <- 2.0
set.seed(456)

# 2. Simulacion de las Dos Trayectorias
# Dinamica real: x(k+1) = -x(k)
x_true_1 <- numeric(N_steps)
z_obs_1  <- numeric(N_steps)
x_true_2 <- numeric(N_steps)
z_obs_2  <- numeric(N_steps)
x0_1 <- rnorm(1, mean = 0, sd = sqrt(P0_1))
x_true_1[1] <- x0_1
v_1 <- rnorm(N_steps, mean = 0, sd = sqrt(R_1))
z_obs_1[1] <- x_true_1[1] + v_1[1]
x0_2 <- rnorm(1, mean = 0, sd = sqrt(P0_2))
x_true_2[1] <- x0_2
v_2 <- rnorm(N_steps, mean = 0, sd = sqrt(R_2))
z_obs_2[1] <- x_true_2[1] + v_2[1]
for (i in 2:N_steps) {
  x_true_1[i] <- -x_true_1[i-1]
  z_obs_1[i]  <- x_true_1[i] + v_1[i]

  x_true_2[i] <- -x_true_2[i-1]
  z_obs_2[i]  <- x_true_2[i] + v_2[i]
}
df_1 <- data.frame(k = k_values,
                   x_true = x_true_1,
                   z_obs = z_obs_1,
                   Escenario = "Trayectoria 1: R bajo (0.1)")
```

```
df_2 <- data.frame(k = k_values,
                   x_true = x_true_2,
                   z_obs = z_obs_2,
                   Escenario = "Trayectoria 2: R alto (2.0)")
df_total <- rbind(df_1, df_2)

# 3. Preparacion de Datos y Grafica
df_long <- tidyr::pivot_longer(df_total,
                               cols = c(x_true, z_obs),
                               names_to = "Serie",
                               values_to = "Valor")
df_long$Serie <- factor(df_long$Serie,
                        levels = c("x_true", "z_obs"),
                        labels = c("Estado Real (x_k)", "Observacion (z_k)"))
g <- ggplot(df_long, aes(x = k, y = Valor, color = Serie)) +
  geom_line(aes(linetype = Serie), size = 0.8) +
  scale_linetype_manual(values = c("Estado Real (x_k)" = "solid",
                                   "Observacion (z_k)" = "dotted")) +
  scale_color_manual(values = c("Estado Real (x_k)" = "red",
                                "Observacion (z_k)" = "black")) +
  geom_point(data = subset(df_long, Serie == "Observacion (z_k)"), size = 1.5) +
  geom_line(data = subset(df_long, Serie == "Estado Real (x_k)"), size = 0.8) +
  facet_wrap(~ Escenario, ncol = 1, scales = "free_y") +
  labs(title = "Trayectorias del Estado vs. Observaciones",
       x = "Tiempo (k)",
       y = "Valor",
       color = "Tipo de Serie",
       linetype = "Tipo de Serie") +
  theme_minimal() +
  theme(legend.position = "bottom")
print(g)
```

Comentario de los Resultados de la simulación (Corregido)



Esta gráfica refleja la verdadera dinámica del sistema descrita en el enunciado.

- **Análisis del Estado Real (línea Roja):** La ecuación de estado es $x(k+1) = (-1)^{2k+1}x(k)$. Dado que $2k+1$ es siempre un número impar para $k \geq 0$ (ej. $1, 3, 5, \dots$), el término $(-1)^{2k+1}$ es siempre -1. Por lo tanto, la dinámica real del sistema es:
$$x(k+1) = -x(k)$$
El estado oscila en cada instante de tiempo, alternando su signo. La gráfica refleja esto perfectamente, mostrando una trayectoria en "dientes de sierra" que salta entre x_0 y $-x_0$.
- **Análisis de las Observaciones (Puntos Negros):** La ecuación $z(k) = x(k) + v(k)$ implica que las mediciones son el estado real oscilante más el ruido. Los puntos negros de la gráfica (Observación) siguen la trayectoria roja (Estado Real), pero con desviaciones aleatorias.
- **Comparación de Escenarios:**
  - **Trayectoria 1 (R bajo = 0.1):** Con una varianza de ruido baja, las observaciones $z(k)$ son muy precisas y siguen fielmente al estado real oscilante.
  - **Trayectoria 2 (R alto = 2.0):** Con una varianza de ruido alta, las observaciones $z(k)$ son muy ruidosas, y aunque siguen la tendencia oscilante, sus valores se desvían considerablemente del estado real.

2.2 (b) Escritura de algoritmos

1. Algoritmo de Filtrado de Kalman

El algoritmo de filtrado se aplica para obtener la estimación óptima $\hat{x}(k/k)$ y la varianza del error $P(k/k)$. Para este modelo escalar, las ecuaciónes son:

Condiciones Iniciales ($k = 0$):

- $\hat{x}(0/-1) = E[x_0] = 0$
- $P(0/-1) = E[x_0^2] = P_0$

Bucle Recursivo (para $k \geq 0$):

1. **Predicción (Ciclo de Tiempo):**
$$\begin{aligned}\hat{x}(k/k-1) &= \Phi\hat{x}(k-1/k-1) = -\hat{x}(k-1/k-1) \quad (\text{para } k>0)\\ P(k/k-1) &= \Phi P(k-1/k-1)\Phi^T + Q\\ &= (-1)P(k-1/k-1)(-1) + 0 = P(k-1/k-1) \quad (\text{para } k>0)\end{aligned}$$
2. **Actualización (Ciclo de Medida):**
$$\begin{aligned}\Pi(k) &= HP(k/k-1)H^T + R = P(k/k-1) + R\\ K(k) &= P(k/k-1)H^T\Pi(k)^{-1} = \frac{P(k/k-1)}{P(k/k-1)+R}\\ \tilde{z}(k/k-1) &= z(k) - H\hat{x}(k/k-1) = z(k) - \hat{x}(k/k-1)\\ \hat{x}(k/k) &= \hat{x}(k/k-1) + K(k)\tilde{z}(k/k-1)\\ P(k/k) &= (1-K(k)H)P(k/k-1) = (1-K(k))P(k/k-1)\end{aligned}$$

2. Algoritmo de Suavizamiento Punto Fijo

El algoritmo de suavizamiento punto fijo se utiliza para refinar la estimacion $\hat{x}(k/k)$ usando observaciones futuras, $j = k+1, \ldots, N$. Se calculan $\hat{x}(k/j)$ y $P(k/j)$ para un k fijo.

Condiciones Iniciales (en $j = k$):

- $\hat{x}(k/k)$ (obtenido del filtro)
- $P(k/k)$ (obtenido del filtro)
- $L(k,k) = P(k/k)$

Bucle Recursivo (para $j = k+1, k+2, \ldots$):

1. Calcular la ganancia $K(k,j)$ y la matriz $L(k,j)$:

$$\begin{aligned} L(k,j) &= L(k,j-1)\Phi^T(1-K(j)H)^T \\ &= L(k,j-1)\cdot(-1)\cdot(1-K(j)) = (K(j)-1)L(k,j-1) \\ K(k,j) &= L(k,j-1)\Phi^T H^T \Pi(j)^{-1} \\ &= L(k,j-1)\cdot(-1)\cdot 1\cdot\Pi(j)^{-1} = \frac{-L(k,j-1)}{\Pi(j)} \end{aligned}$$

   (donde $K(j)$ y $\Pi(j)$ son la ganancia del filtro y la covarianza de la innovación en el instante j, calculadas en la fase de filtrado).

2. Actualizar el estimador y la varianza:

$$\begin{aligned} \hat{x}(k/j) &= \hat{x}(k/j-1) + K(k,j)\tilde{z}(j/j-1) \\ P(k/j) &= P(k/j-1) - K(k,j)\Pi(j)K(k,j)^T \\ &= P(k/j-1) - K(k,j)^2\Pi(j) \end{aligned}$$

A continuación se implementa el código en R para ambos algoritmos.

```
library(ggplot2)
library(tidyr)

# 1. Definicion de parametros y simulación
Phi <- -1; H <- 1; Q <- 0; R <- 0.5; P0 <- 1; x0_media <- 0
N_iter <- 50; N_steps <- N_iter + 1
N_horizon <- 2; time <- 0:N_iter
set.seed(123)
x_true <- numeric(N_steps); z_obs <- numeric(N_steps)
v <- rnorm(N_steps, 0, sqrt(R))
x_true[1] <- rnorm(1, x0_media, sqrt(P0))
z_obs[1] <- H * x_true[1] + v[1]
for (i in 2:N_steps) {
  x_true[i] <- Phi * x_true[i-1]
  z_obs[i] <- H * x_true[i] + v[i]
}

# 2. Algoritmo de Filtrado de Kalman
x_pred <- numeric(N_steps); P_pred <- numeric(N_steps)
x_fil <- numeric(N_steps); P_fil <- numeric(N_steps)
K_gain <- numeric(N_steps); innov <- numeric(N_steps)
```

```
Pi_cov <- numeric(N_steps)
x_pred[1] <- x0_media; P_pred[1] <- P0
Pi_cov[1] <- H * P_pred[1] * H + R
K_gain[1] <- P_pred[1] * H / Pi_cov[1]
innov[1] <- z_obs[1] - H * x_pred[1]
x_fil[1] <- x_pred[1] + K_gain[1] * innov[1]
P_fil[1] <- (1 - K_gain[1] * H) * P_pred[1]
for (i in 2:N_steps) {
  # Bucle del Filtro de Kalman
  x_pred[i] <- Phi * x_fil[i-1]
  P_pred[i] <- P_fil[i-1] + Q # P(k/k-1) = Phi*P*Phi' + Q = (-1)^2*P + 0 = P

  Pi_cov[i] <- H * P_pred[i] * H + R
  K_gain[i] <- P_pred[i] * H / Pi_cov[i]
  innov[i] <- z_obs[i] - H * x_pred[i]
  x_fil[i] <- x_pred[i] + K_gain[i] * innov[i]
  P_fil[i] <- (1 - K_gain[i] * H) * P_pred[i]
}

# 3. Algoritmo de Suavizamiento Punto Fijo (N=2)
x_smooth <- numeric(N_steps); P_smooth <- numeric(N_steps)
x_smooth[] <- NA; P_smooth[] <- NA
for (i in 1:N_steps) {
  # Bucle del Suavizador (para cada k, mira N pasos adelante)
  x_smooth_current <- x_fil[i]; P_smooth_current <- P_fil[i]; L_current <- P_fil[i]
  j_offset <- 0
  for (j in (i + 1):(i + N_horizon)) {
    j_offset <- j_offset + 1
    if (j > N_steps) { break }
    K_smooth_j <- L_current * Phi * H / Pi_cov[j]
    x_smooth_current <- x_smooth_current + K_smooth_j * innov[j]
    P_smooth_current <- P_smooth_current - K_smooth_j * Pi_cov[j] * K_smooth_j
    L_current <- L_current * Phi * (1 - K_gain[j] * H)
  }
  if (j_offset == N_horizon || (i + N_horizon > N_steps && i <= N_steps)) {
    x_smooth[i] <- x_smooth_current
    P_smooth[i] <- P_smooth_current
  }
}
x_smooth[is.na(x_smooth)] <- x_fil[is.na(x_smooth)]
P_smooth[is.na(P_smooth)] <- P_fil[is.na(P_smooth)]

# 4. Impresion de resultados
k_sample <- 20
i_sample <- k_sample + 1
print(paste("Resultados en k =", k_sample))
print(paste("  Estado Real x(k):", round(x_true[i_sample], 4)))
print(paste("  Observacion z(k):", round(z_obs[i_sample], 4)))
print(paste("  Filtro x_hat(k/k):", round(x_fil[i_sample], 4)))
print(paste("  Suavizado x_hat(k/k+N):", round(x_smooth[i_sample], 4)))
print("---")
print("Varianzas del Error (Teoricas):")
```

```
print(paste("  Filtro P(k/k):", round(P_fil[i_sample], 4)))
print(paste("  Suavizado P(k/k+N):", round(P_smooth[i_sample], 4)))
```

```
> print(paste("Resultados en k =", k_sample))
[1] "Resultados en k = 20"
> print(paste("  Estado Real x(k):", round(x_true[i_sample], 4)))
[1] "  Estado Real x(k): -0.0285"
> print(paste("  Observacion z(k):", round(z_obs[i_sample], 4)))
[1] "  Observacion z(k): -0.7836"
> print(paste("  Filtro x_hat(k/k):", round(x_fil[i_sample], 4)))
[1] "  Filtro x_hat(k/k): 0.0523"
> print(paste("  Suavizado x_hat(k/k+N):", round(x_smooth[i_sample], 4)))
[1] "  Suavizado x_hat(k/k+N): 0.0211"
> print("---")
[1] "---"
> print("Varianzas del Error (Teoricas):")
[1] "Varianzas del Error (Teoricas):"
> print(paste("  Filtro P(k/k):", round(P_fil[i_sample], 4)))
[1] "  Filtro P(k/k): 0.0233"
> print(paste("  Suavizado P(k/k+N):", round(P_smooth[i_sample], 4)))
[1] "  Suavizado P(k/k+N): 0.0213"
>
```

2.3 (c) Gráficas del Filtro y Suavizador Punto Fijo y Análisis de Resultados

```
# 0. Cargar Librerias
library(ggplot2)
library(tidyr)

# 1. Definicion de Parametros del Modelo (Ejercicio 2)
# x(k+1) = -1*x(k) + 0*w(k)
# z(k)   = 1*x(k) + v(k)
Phi <- -1
H   <- 1
Q   <- 0    # No hay ruido de proceso
R   <- 0.5  # Varianza de v(k)
P0  <- 1    # Varianza de x(0)
x0_media <- 0

# 2. Parametros de simulacion y Suavizado
N_iter    <- 20 # 20 iteraciones (k=0 a 20)
N_steps   <- N_iter + 1
N_horizons <- c(1, 2, 4) # Horizontes N para suavizado
N_max     <- max(N_horizons)
time <- 0:N_iter
set.seed(123) # Para reproducibilidad

# 3. Simulacion del Sistema (x(k+1) = -x(k))
x_true <- numeric(N_steps)
z_obs  <- numeric(N_steps)
v <- rnorm(N_steps, 0, sqrt(R)) # Ruido de observacion
x_true[1] <- rnorm(1, x0_media, sqrt(P0))
z_obs[1]  <- H * x_true[1] + v[1]
for (i in 2:N_steps) {
  x_true[i] <- Phi * x_true[i-1]
  z_obs[i]  <- H * x_true[i] + v[i]
```

```
}

# 4. Fase 1: Algoritmo de Filtrado de Kalman
x_pred <- numeric(N_steps) # x_hat(k/k-1)
P_pred <- numeric(N_steps) # P(k/k-1)
x_fil  <- numeric(N_steps) # x_hat(k/k)
P_fil  <- numeric(N_steps) # P(k/k)
K_gain <- numeric(N_steps) # K(k)
innov  <- numeric(N_steps) # z_tilde(k/k-1)
Pi_cov <- numeric(N_steps) # Cov(innov)
x_pred[1] <- x0_media
P_pred[1] <- P0
Pi_cov[1] <- H * P_pred[1] * H + R
K_gain[1] <- P_pred[1] * H / Pi_cov[1]
innov[1]  <- z_obs[1] - H * x_pred[1]
x_fil[1]  <- x_pred[1] + K_gain[1] * innov[1]
P_fil[1]  <- (1 - K_gain[1] * H) * P_pred[1]
for (i in 2:N_steps) {
  x_pred[i] <- Phi * x_fil[i-1]
  P_pred[i] <- P_fil[i-1] + Q
  Pi_cov[i] <- H * P_pred[i] * H + R
  K_gain[i] <- P_pred[i] * H / Pi_cov[i]
  innov[i]  <- z_obs[i] - H * x_pred[i]
  x_fil[i]  <- x_pred[i] + K_gain[i] * innov[i]
  P_fil[i]  <- (1 - K_gain[i] * H) * P_pred[i]
}

# 5. Fase 2: Algoritmo de Suavizamiento Punto Fijo
x_smooth <- matrix(NA, nrow = N_steps, ncol = length(N_horizons))
P_smooth <- matrix(NA, nrow = N_steps, ncol = length(N_horizons))
colnames(x_smooth) <- paste0("N=", N_horizons)
colnames(P_smooth) <- paste0("N=", N_horizons)
for (i in 1:N_steps) {
  x_smooth_current <- x_fil[i]
  P_smooth_current <- P_fil[i]
  L_current <- P_fil[i]
  for (j_offset in 1:N_max) {
    j_idx <- i + j_offset
    if (j_idx > N_steps) { break }
    K_smooth_j <- L_current * Phi * H / Pi_cov[j_idx]
    x_smooth_current <- x_smooth_current + K_smooth_j * innov[j_idx]
    P_smooth_current <- P_smooth_current - K_smooth_j * Pi_cov[j_idx] * K_smooth_j
    L_current <- L_current * Phi * (1 - K_gain[j_idx] * H)
    if (j_offset %in% N_horizons) {
      col_idx <- which(N_horizons == j_offset)
      x_smooth[i, col_idx] <- x_smooth_current
      P_smooth[i, col_idx] <- P_smooth_current
    }
  }
}

# 6. Preparacion de Datos y Graficas
```

```
# Grafica 1: Trayectorias
data_g1 <- data.frame(
  Tiempo = time,
  Estado_Real = x_true,
  Observacion = z_obs,
  Filtrado = x_fil,
  Suavizado_N2 = x_smooth[, "N=2"]
)
data_g1$Suavizado_N2[is.na(data_g1$Suavizado_N2)] <- data_g1$Filtrado[is.na(data_g1$Suavizado_N2)]
data_g1_long <- tidyr::pivot_longer(
  data_g1,
  cols = -Tiempo,
  names_to = "Serie",
  values_to = "Valor"
)
g1 <- ggplot(data_g1_long, aes(x = Tiempo, y = Valor, color = Serie)) +
  geom_line(aes(linetype = Serie), size = 0.8) +
  geom_point(data = subset(data_g1_long, Serie == "Observacion"), size = 1.5, shape = 1) +
  scale_linetype_manual(values = c(
    "Estado_Real" = "solid",
    "Observacion" = "blank",
    "Filtrado" = "dashed",
    "Suavizado_N2" = "solid"
  )) +
  scale_color_manual(values = c(
    "Estado_Real" = "black",
    "Observacion" = "grey50",
    "Filtrado" = "blue",
    "Suavizado_N2" = "red"
  )) +
  labs(
    title = "Comparativa de Estimaciones (N=2)",
    x = "Tiempo (k)",
    y = "Valor",
    color = "Series",
    linetype = "Series"
  ) +
  theme_minimal() +
  theme(legend.position = "bottom")

# Grafica 2: Varianzas
data_g2 <- data.frame(
  Tiempo = time,
  Filtrado = P_fil,
  Suavizado_N1 = P_smooth[, "N=1"],
  Suavizado_N2 = P_smooth[, "N=2"],
  Suavizado_N4 = P_smooth[, "N=4"]
)
data_g2[is.na(data_g2)] <- data_g2$Filtrado[is.na(data_g2)]
data_g2_long <- tidyr::pivot_longer(
  data_g2,
```

```
  cols = -Tiempo,
  names_to = "Serie",
  values_to = "Varianza"
)
data_g2_long$Serie <- factor(data_g2_long$Serie, levels = c("Filtrado", "Suavizado_N1", "Suavizado_N2", "S
g2 <- ggplot(data_g2_long, aes(x = Tiempo, y = Varianza, color = Serie)) +
  geom_line(size = 0.8) +
  labs(
    title = "Comparativa de Varianzas del Error",
    x = "Tiempo (k)",
    y = "Varianza del Error (P)",
    color = "Estimador"
  ) +
  coord_cartesian(ylim = c(0, NA)) +
  theme_minimal() +
  theme(legend.position = "bottom")
print(g1)
print(g2)
```

(c) Comentario de los Resultados Gráficos

Comentario de: Comparativa de Estimaciones (N=2)

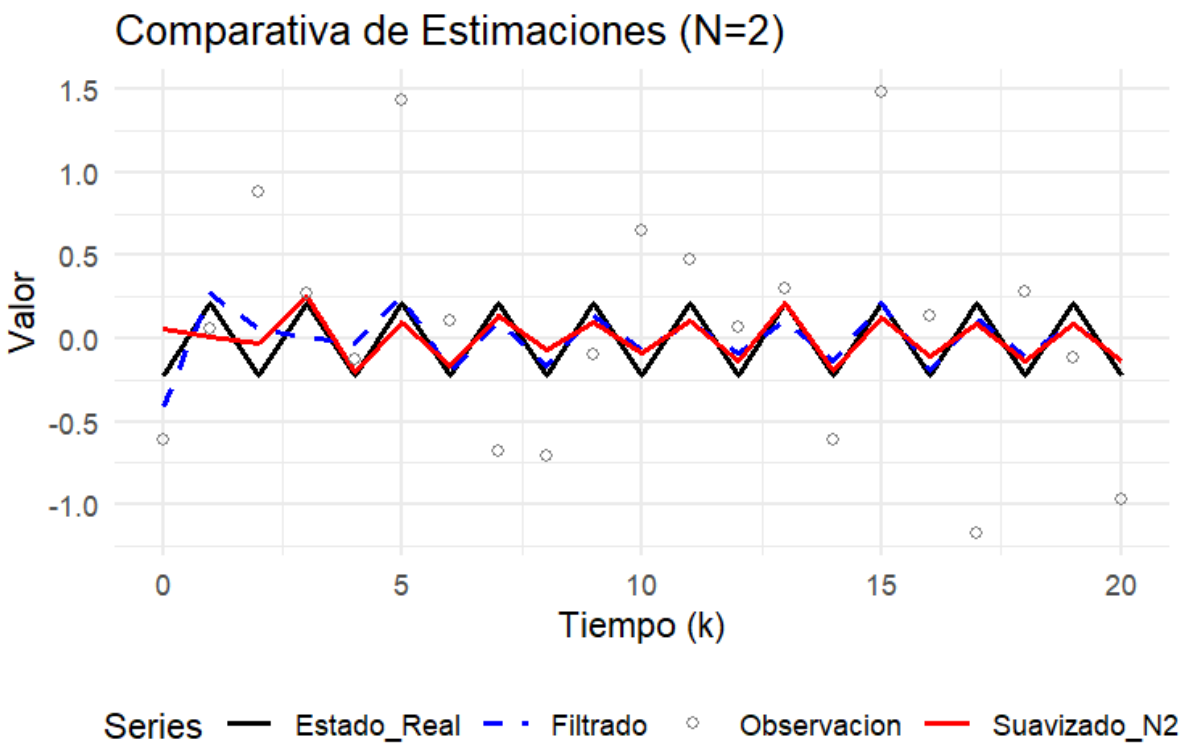


Figure 2: Trayectorias del estado, observacion, filtro y suavizador (N=2).

Esta gráfica muestra la superioridad del suavizador sobre el filtro.

- **Estado Real (Negro):** Muestra la trayectoria oscilante $x(k+1) = -x(k)$, tal como se dedujo.
- **Observación (Círculos):** Muestra los valores $z(k) = x(k) + v(k)$. Los puntos están dispersos aleatoriamente alrededor de la línea negra, como corresponde a un ruido con $R = 0.5$.
- **Filtrado (Azul discontínuo):** Es la estimación $\hat{x}(k/k)$. Sigue la trayectoria real, pero es "ruidosa" y reacciona bruscamente a observaciones atípicas (ver $k = 0$, $k = 4$, $k = 9$). El filtro solo usa el pasado, por lo que una mala observación lo desvía temporalmente.
- **Suavizado N=2 (Rojo):** Es la estimación $\hat{x}(k/k+2)$. Esta trayectoria es visiblemente más suave y más precisa que la del filtro. Al usar $N = 2$ observaciones futuras, el suavizador puede "ignorar" mejor las observaciones atípicas. Por ejemplo, en $k = 4$, la observación es muy alta (≈ 1.5), el filtro reacciona, pero el suavizador (que ya ha "visto" $k = 5$ y $k = 6$) sabe que el estado debe volver a bajar y corrige la estimación, manteniéndose mucho más cerca del estado real.

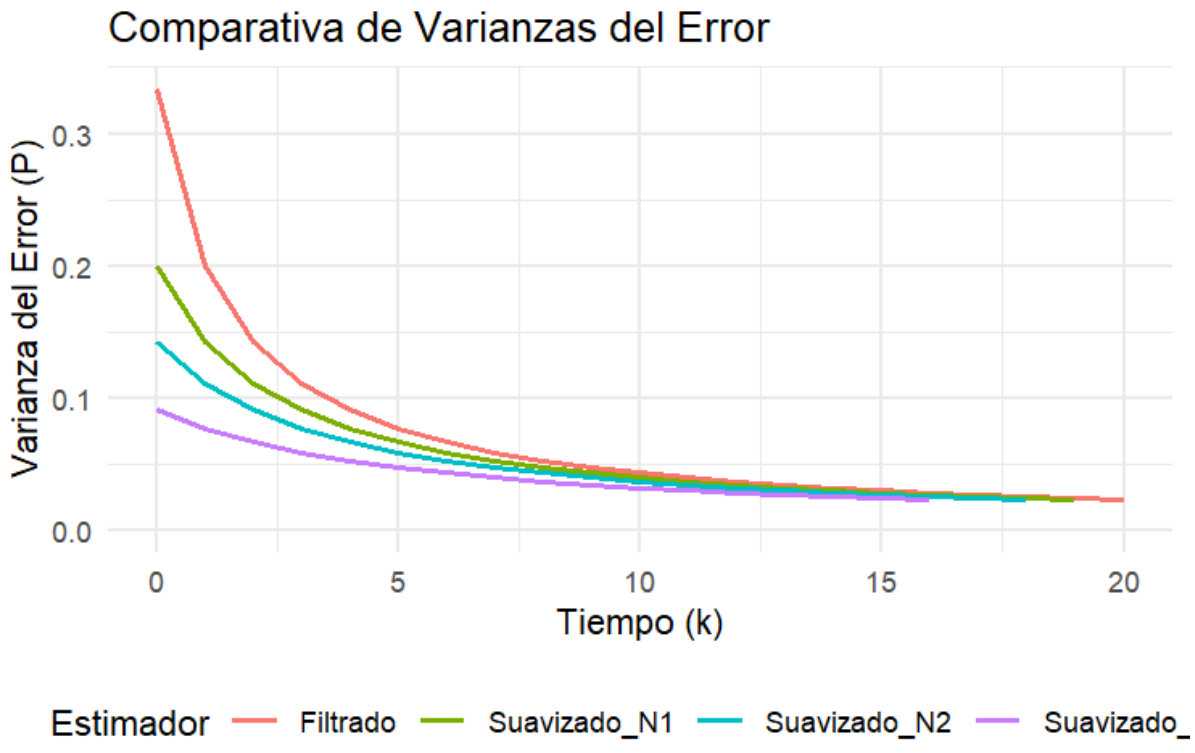


Figure 3: Varianzas del error (P) para el filtro y suavizadores (N=1, 2, 4).

Esta gráfica confirma teóricamente lo que se observó en la gráfica anterior.

- **Régimen Transitorio y Estacionario:** Todas las varianzas comienzan altas (la incertidumbre inicial $P_0 = 1$ es alta) y disminuyen rápidamente a medida que el filtro procesa observaciones. Después de $k \approx 15$, el sistema alcanza un régimen estacionario donde la varianza del error se vuelve constante.
- **Jerarquía de Varianza:** La gráfica muestra una jerarquía clara de precisión:

$$P(\text{Filtrado}) > P(\text{N=1}) > P(\text{N=2}) > P(\text{N=4})$$

La varianza del Filtro (línea roja) es siempre la más alta, lo que indica la mayor incertidumbre. A medida que aumenta el horizonte de suavizado N, la varianza del error $P(k/k+N)$ disminuye.

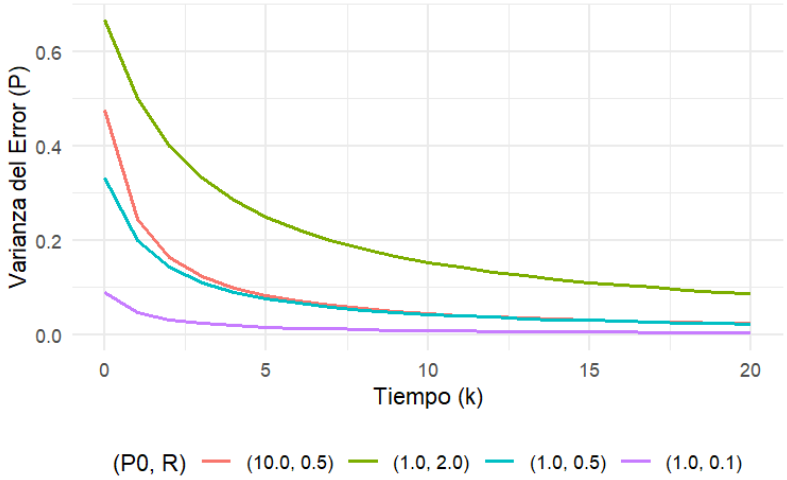
- **Conclusión Teórica:** Esto demuestra que usar observaciones futuras reduce la incertidumbre teórica (la varianza) de la estimación. Un N más grande produce una estimación teóricamente mejor.
- **Rendimiento Decreciente:** La mayor ganancia de precisión (la mayor "bajada" de varianza) se obtiene al pasar de Filtro a Suavizado N=1. Las mejoras al aumentar N (de 1 a 2, de 2 a 4) son cada vez más pequeñas.

2.4 (d) Representación de las varianzas de los errores de filtrado

```
# 0. Cargar Librerias
library(ggplot2)
library(tidyr)

# 1. Definicion de Parametros del Modelo
Phi <- -1
H <- 1
Q <- 0
N_iter <- 20
N_steps <- N_iter + 1
time <- 0:N_iter

# 2. Funcion para calcular la Varianza del Filtro
calcular_varianza_filtro <- function(P0_val, R_val, Phi, H, Q, N_steps) {
  P_pred <- numeric(N_steps)
  P_fil  <- numeric(N_steps)
  P_pred[1] <- P0_val
  Pi_cov_k0 <- H * P_pred[1] * H + R_val
  K_gain_k0 <- P_pred[1] * H / Pi_cov_k0
  P_fil[1]  <- (1 - K_gain_k0 * H) * P_pred[1]
  for (i in 2:N_steps) {
    P_pred[i] <- P_fil[i-1] + Q
    Pi_cov_i <- H * P_pred[i] * H + R_val
    K_gain_i <- P_pred[i] * H / Pi_cov_i
    P_fil[i]  <- (1 - K_gain_i * H) * P_pred[i]
  }
  return(P_fil)
}

# 3. Definicion y calculo de Escenarios
P0_base <- 1.0; R_base <- 0.5
P0_alt <- 10.0; R_alt_P0 <- 0.5
P0_alt_R <- 1.0; R_alt <- 2.0
P0_baj_R <- 1.0; R_baj <- 0.1
P_fil_base <- calcular_varianza_filtro(P0_base, R_base, Phi, H, Q, N_steps)
P_fil_P0_alt <- calcular_varianza_filtro(P0_alt, R_alt_P0, Phi, H, Q, N_steps)
P_fil_R_alt <- calcular_varianza_filtro(P0_alt_R, R_alt, Phi, H, Q, N_steps)
P_fil_R_baj <- calcular_varianza_filtro(P0_baj_R, R_baj, Phi, H, Q, N_steps)

# 4. Preparacion de Datos para Grafica
data_g3 <- data.frame(
```

```
  Tiempo = time,
  Varianza_Base = P_fil_base,
  Varianza_P0_Alto = P_fil_P0_alt,
  Varianza_R_Alto = P_fil_R_alt,
  Varianza_R_Bajo = P_fil_R_baj
)
data_g3_long <- tidyr::pivot_longer(
  data_g3,
  cols = -Tiempo,
  names_to = "Escenario",
  values_to = "Varianza"
)
data_g3_long$Escenario <- factor(data_g3_long$Escenario,
                                 levels = c("Varianza_P0_Alto", "Varianza_R_Alto", "Varianza_Base", "Varia
                                 labels = c("P0=10.0, R=0.5 (P0 Alto)",
                                            "P0=1.0, R=2.0 (R Alto)",
                                            "P0=1.0, R=0.5 (Base)",
                                            "P0=1.0, R=0.1 (R Bajo)"))

# 5. Representacion
g3 <- ggplot(data_g3_long, aes(x = Tiempo, y = Varianza, color = Escenario)) +
  geom_line(size = 0.8) +
  labs(
    x = "Tiempo (k)",
    y = "Varianza del Error (P)",
    color = "Escenario (P0, R)"
  ) +
  coord_cartesian(ylim = c(0, NA)) +
  theme_minimal() +
  theme(legend.position = "bottom")
print(g3)
```

Comentario de la Gráfica de Varianzas



Esta gráfica muestra cómo evoluciona la "confianza" del filtro (la varianza del error P) bajo diferentes condiciones de incertidumbre inicial (P_0) y ruido de medición (R).

1. El Efecto de P_0 (Incertidumbre Inicial)

Comparamos la línea "Base" (azul, $P_0 = 1.0$) con la línea "P0 Alto" (roja, $P_0 = 10.0$).

- **Al principio (Transitorio):** La línea "P0 Alto" (roja) empieza con una varianza mucho mayor. Esto es lógico, ya que el filtro arranca con una confianza inicial mucho menor.
- **Al final (Estacionario):** Ambas curvas convergen exactamente al mismo valor (la línea roja y la azul se juntan después de $k \approx 10$).
- **Conclusión:** La incertidumbre inicial (P_0) solo afecta al principio. A largo plazo, el filtro acumula suficientes mediciones y "olvida" su suposición inicial.

2. El Efecto de R (Calidad de la Medición)

Comparamos las tres curvas con $P_0 = 1.0$: "R Bajo" (morada, $R = 0.1$), "Base" (azul, $R = 0.5$) y "R Alto" (verde, $R = 2.0$).

- **Convergencia:** A diferencia del caso anterior, estas tres curvas convergen a niveles estacionarios distintos.
- **Interpretación:**
  - **R Bajo (Morada):** Un sensor preciso ($R = 0.1$) permite al filtro alcanzar una confianza muy alta (varianza muy baja).
  - **R Alto (Verde):** Un sensor ruidoso ($R = 2.0$) limita al filtro. Aunque reciba muchas mediciones, nunca puede estar muy seguro, y su varianza estacionaria es alta.
- **Conclusión:** La calidad del sensor (R) es un factor crítico que determina la precisión máxima que el filtro puede alcanzar a largo plazo.