Universidad de Valladolid
Máster en Matemáticas

Curso 2021/2022

Computación paralela y cálculo distribuido

PROYECTO FINAL

Estimación de parámetros de modelos FMM de una componente

Canedo Ortega, Christian

Índice

1. Planteamiento del problema.

El modelo FMM es un modelo no lineal específico para señales oscilatorias o circulares, esto es, que después de un cierto periodo de tiempo la señal se repite delineando una forma similar salvo por procesos aleatorios y artefactos externos. Un ejemplo claro es el electrocardiograma.

La formulación del modelo FMM de una única componente, que también es del tipo señal más ruido, es el siguiente:

Modelo $\mathbf{FMM_1}$.

$$X(t_i) = \mu(t_i) + \epsilon(t_i) = M + Acos(\phi(t_i; \alpha, \beta, \omega)) + \epsilon(t_i),\ i = 1, ..., n,$$
$$(\epsilon(t_1), ..., \epsilon(t_n))' \sim N_n(0, \sigma^2 I_n)$$

donde $M \in \mathbb{R}$, A $\in \mathbb{R}^+$ y $\phi(t; \alpha, \beta, \omega) = \beta + 2\, arctan(\omega\, tan(\frac{t-\alpha}{2})))$; $\alpha, \beta \in [0, 2\pi)$, $\omega \in [0, 1]$.

Descripción de los parámetros

Los parámetros α, β y ω definen la fase del modelo, mientras que M y A son parámetros de intercept y escala.

α es un parámetro de localización de la fase. En la figura 1 se muestran cuatro señales FMM estándar ($M = 0$ y amplitud unidad) con distintos valores para α manteniendo fijos el resto de parámetros.

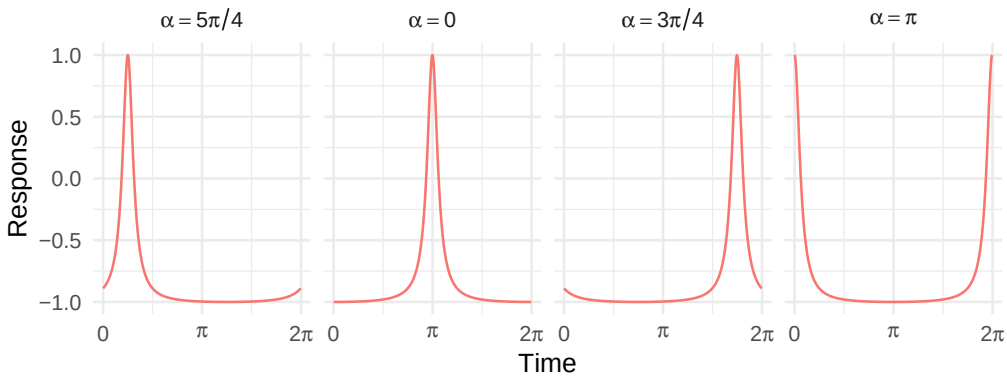

Figura 1. Modelos FMM. $M = 0$, $A = 1$, $\omega = 0{,}1$, $\beta = \pi$.

Por otro lado, ω y β son parámetros de forma. El parámetro β describe la asimetría de la onda y ω el apuntamiento. En la Figura 2 se ilustran algunos patrones asociados a valores específicos de los parámetros.

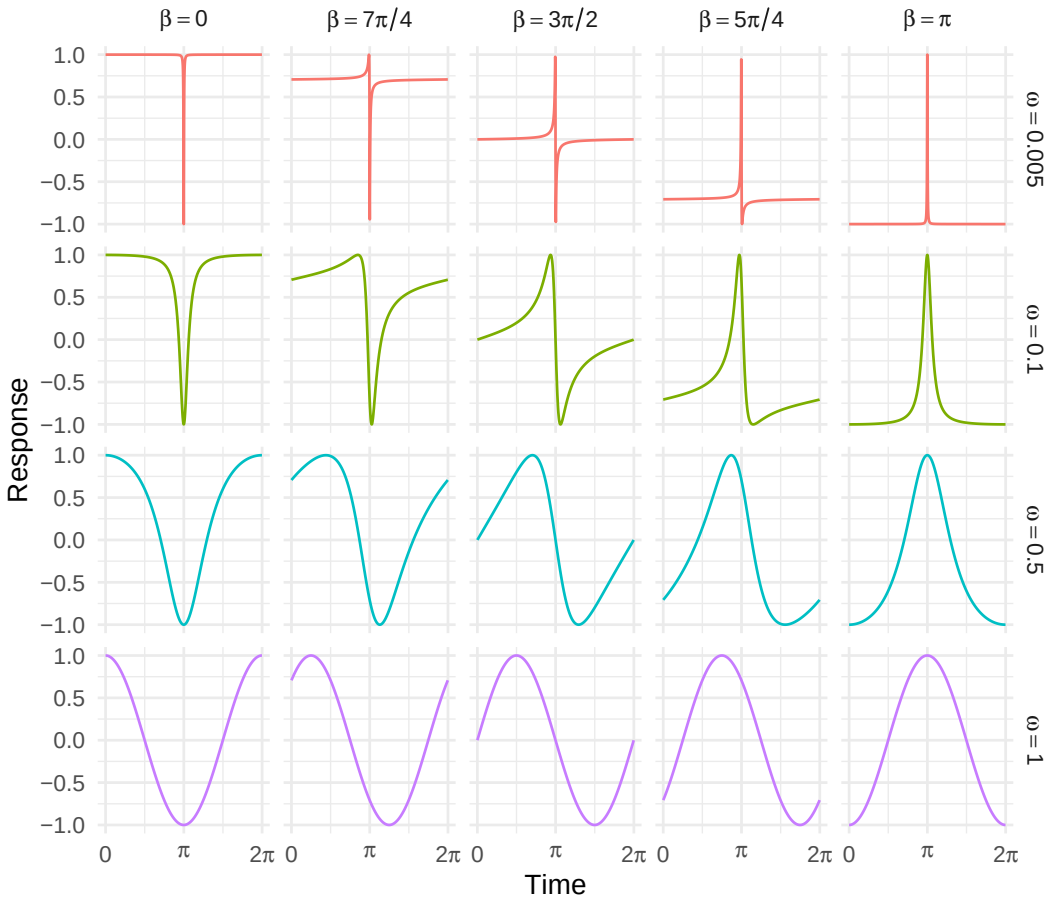

Figura 2. Modelos FMM. $M = 0$, $A = 1$, $\alpha = 0$.

La onda es totalmente simétrica cuando $\beta = 0$ y $\beta = \pi$, mientras que en sus valores intermedios se dan formas asimétricas y ω mide el apuntamiento en la onda FMM. Aquellas ondas con un valor de ω cercano a 0 tienen un apuntamiento más pronunciado, mientras que valores de ω próximos a 1 describen formas más suaves, similares a ondas sinusoidales.

Estimación de los parámetros del modelo.

El problema de calcular el estimador máximo verosímil, en este caso, se reduce a un problema de mínimos cuadrados (puesto que el error es normal):

$$\hat{\theta} = \underset{\theta \in \Theta}{argmin} \sum_{i=1}^{n} (X(t_i) - \mu(t_i, \theta))^2 \tag{1}$$

donde Θ es el espacio paramétrico para $\theta = (M, A, \alpha, \beta, \omega)$, $\Theta = \mathbb{R} \times \mathbb{R}^+ \times [0, 2\pi) \times [0, 2\pi) \times [0, 1]$.

Para la estimación es conveniente una reformulación del modelo, como sigue:

$$\mu(t_i, \theta) = M + Acos(t_i^* + \varphi), i = 1, ..., n \tag{2}$$

donde $t_i^* = \alpha + arctan(\omega tan(\frac{t_i - \alpha}{2}))$ y $\varphi = \beta - \alpha$.

Pudiendo llegar a nueva expresión del modelo en forma lineal:

$$\mu(t_i, \theta) = M + \delta z_i + \gamma \omega_i, i = 1, ..., n \tag{3}$$

donde $\delta = Acos(\varphi)$, $\gamma = -Asin(\varphi)$, $z_i = cos(t_i^*)$, $\omega = sin(t_i^*)$, para $i = 1, ..., n$.

Para valores fijos y conocidos de α y ω, lo que supone valores conocidos de t_i^*, la estimación de los parámetros se reduce a un problema de mínimos cuadrados ordinarios. De esta forma, la estimación del resto de parámetros (M, A y β) queda como sigue:

$$\hat{M} = \bar{X} - \hat{\delta} \sum_{i=1}^{n} z_i - \hat{\gamma} \sum_{i=1}^{n} \omega_i \tag{4}$$

$$\hat{A} = \sqrt{\hat{\delta}^2 + \hat{\gamma}^2} \tag{5}$$

$$\hat{\beta} = \hat{\alpha} + \hat{\varphi} \tag{6}$$

Teniendo en cuenta el desarrollo anterior, el algoritmo de estimación es el siguiente:

1. Se define un grid de α y ω, $\{\alpha_1, ..., \alpha_N\} \times \{\omega_1, ..., \omega_M\}$. El grid para α es equiespaciado, puesto que la estimación es igual de sensible en todos los valores. Sin embargo, el grid de ω se define de manera logarítmica, explorando de manera más exhaustiva los valores más cercanos a 0. **Esto se lleva a cabo en el main haciendo uso de la función `seq` en el proyecto.**

2. Para cada punto se obtienen los estimadores definidos en las ecuaciones anteriores 4, 5 y 6. **Función `step1FMM` en el proyecto.**

3. Se calcula la predicción para cada valor ($\hat{y}_i$) y se computa la suma de cuadrados de los errores $RSS = \sum_{i=1}^{n} (y - \hat{y}_i)^2$.

4. Seleccionamos la el conjunto de estimadores que minimiza el RSS tal como muestra el problema propuesto en la ecuación 1. **Desde las llamadas a `step1FMM` en el paso 2 hasta este punto se realiza en `fitFMM`.**

2. Procedimiento de paralelización.

El algoritmo de estimación anterior es susceptible de paralelizarse puesto que cada una de las llamadas a `step1FMM` es independiente del resto, teniendo que computarse para una rejilla de 350 valores de α y 350 valores de ω se resuelven un total de 122500 sistemas lineales.

```
/* ---------------------- En la función fitFMM -------------------- */
// Definidos nOmega, nAlpha, seqAlphas, seqOmegas.
...
#pragma omp parallel for
for(int i = 0; i < nOmega; i++){
    for(int j = 0; j < nAlpha; j++){
        step1FMM(vData,timePoints,n,seqAlphas[j],seqOmegas[i],params);
        RSSs[i*nAlpha + j] = params[5];
    }
}
...
```

Otra estrategia que se ha descartado debido a que se ha considerado un nivel de grano demasiado fino es paralelizar el bucle que calcula la matriz de varianzas-covarianzas (función `covMatrix`). Esto supondría inicializar los mecanismos de paralelización en cada llamada de `step1FMM`.

Tener también en cuenta que el crecimiento del número de operaciones en el caso de cualquiera de los parámetros (*nAlpha* nodos para α, *nOmega* nodos para ω, n datos de la señal) supone un incremento lineal en las operaciones. En el caso de la señal afecta al cálculo de la matriz de varianzas covarianzas, de orden lineal con respecto al parámetro mencionado.

3. Análisis de los resultados de la paralelización.

En las siguientes gráficas se muestra el tiempo medio de ejecución por cada número de cores usado en el paralelizado, además de el speedup (proporción de tiempo con respecto a la ejecución en serie)

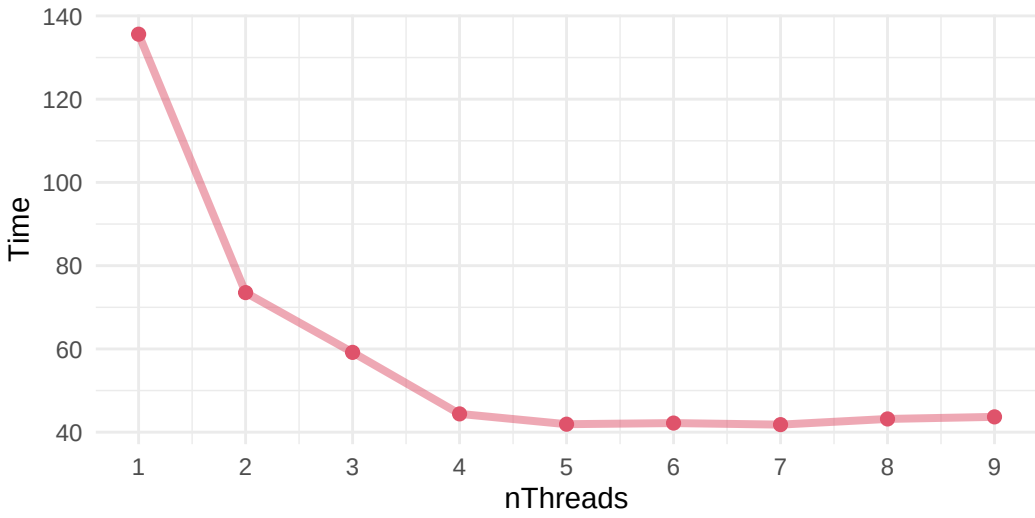

Figura 3. Tiempos de cómputo para nAlpha, nOmega = 350, n = 3500 (nThreads de 1 a 9).

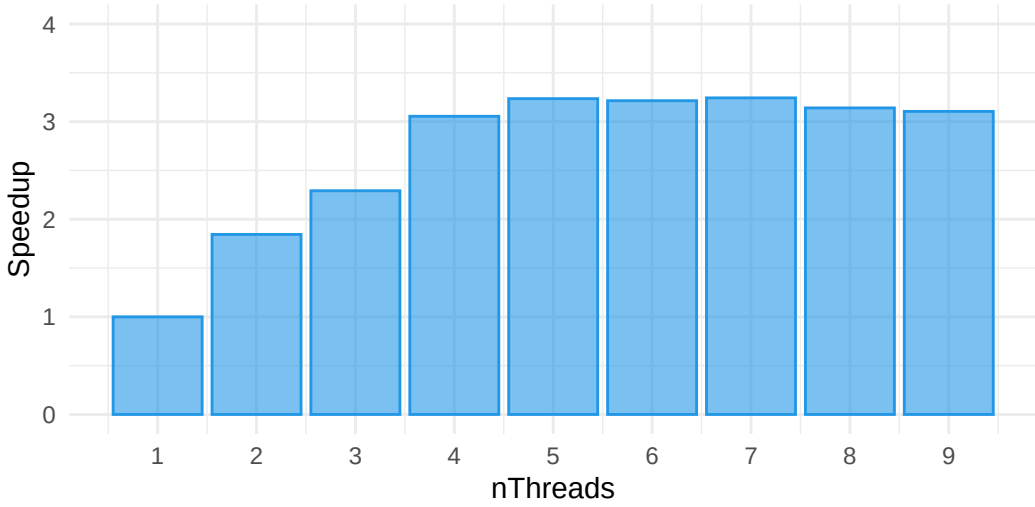

Figura 4. Speedup para nAlpha, nOmega = 350, n = 3500 (nThreads de 1 a 9).

Puede observarse en las Figuras 3 y 4 que el tiempo de cómputo no se reduce bajo una ley de proporcionalidad. A partir de los 4 cores en la máquina **Heracles** no se observa ningún tipo de mejora en los tiempos de cómputo, incluso un peor tiempo.

De forma análoga, el speedup muestra que con dos cores la ganancia es prácticamente proporcional pero que decrece muy rápidamente en cuanto aumenta el número de threads. Para este caso concreto, no se recomendaría usar más de cuatro threads en ningún caso.

4. Conclusiones.

La carga computacional que supone este problema para C y las máquinas usadas no parece ser suficiente como para aprovechar un paralelizado intensivo. El tamaño de señal que se ha usado (n = 3500) es razonablemente más grande que los casos que se usan en la práctica con este modelo (la mayoría ronda de los 500 a los 1000 datos muestreados).

Aunque es prácticamente obvio, C supone una mejoría con respecto a R que es el lenguaje donde actualmente se realizan los cómputos de la estimación de parámetros del modelo.