Implementación del Método de Análisis en el Dominio de las Frecuencias.

Introducción al método:

La idea de este método es pasar al dominio de las frecuencias, aplicando la Transformada de Fourier a la ecuación dinámica clásica, para luego una vez resuelta, volver al espacio del tiempo, usando la Transformada Inversa. Recordar que:

$$\mathcal{F}(\{v(t)\}) = \{V(\bar{\omega})\} \qquad \mathcal{F}(\{\dot{v}(t)\}) = i\bar{\omega} \cdot \{V(\bar{\omega})\} \qquad \mathcal{F}(\{\ddot{v}(t)\}) = -\bar{\omega}^2 \cdot \{V(\bar{\omega})\}$$

$$\mathcal{F}(\{p(t)\}) = \{P_\omega(\bar{\omega})\}$$

A partir de la ecuación dinámica, se tiene entonces:

$$\{V(\bar{\omega})\} = \left[-[M] \cdot \bar{\omega}^2 + [C] \cdot i\bar{\omega} + [K]\right]^{-1} \cdot \{P_\omega(\bar{\omega})\}$$

Notar que esta es una ecuación matricial. El problema queda entonces sujeto a encontrar la matriz FRF:

$$[H(\bar{\omega})] = \left[-[M] \cdot \bar{\omega}^2 + [C] \cdot i\bar{\omega} + [K]\right]^{-1}$$

Esto se puede hacer directamente (sin necesariamente tener C clásica), lo que puede ser costoso computacionalmente, o bien aprovechando la ortogonalidad de las matrices M, K y C (asumiéndola clásica para que cumpla con ortogonalidad) con respecto a las formas modales. De esta forma se tiene:

$$[H(\bar{\omega})]_{nxn} = \sum_{j=1}^{m} \{\phi_j\}_{nx1} \cdot \{\phi_j\}^T_{1xn} \cdot \underbrace{\frac{1}{K_j \cdot \left(1 - \frac{\bar{\omega}^2}{\omega_j^2} + 2 \cdot \beta_j \cdot i \cdot \frac{\bar{\omega}}{\omega_j}\right)}}_{h_j(\bar{\omega})}$$

Donde ω_j corresponde a la frecuencia angular del modo j-ésimo de la estructura.

Notar que n corresponde a la cantidad de grados de libertad del sistema (misma cantidad de modos), pero la sumatoria podría ser hasta un número m<n, truncando así la solución a ciertos modos solamente.

Zero Padding:

Al hacer la transformada de Fourier de una función real $f(t)$ (se asume que el vector {P(t)} de carga es real) se genera un muestreo de la función continua $F(\bar{\omega})$ resultante de la transformada. Este muestreo se hace, como se mencionó anteriormente, a partir de un vector discreto de frecuencias angulares ω dado por el largo total de la función $f(t)$ (tiempo total). El espaciamiento de estos valores de frecuencia está dado por[1]:

$$\Delta\bar{\omega} = \frac{2 \cdot \pi}{T_p}$$

Donde T_p : Tiempo total de la función f(t). Teóricamente, para la transformada de Fourier este valor es el período de la función que se repite desde $-\infty$ hasta ∞.

Así entonces, para resolver la ecuación dinámica usando el resultado de la ecuación anterior, se calcula para cada frecuencia angular ω, el valor de $\{P_\omega(\bar{\omega})\}$ (y su respectivo $[H(\bar{\omega})]$) para luego obtener el valor de $\{V(\bar{\omega})\}$ y aplicar la antitransformada de Fourier para obtener las respuestas en el espacio del tiempo. A partir de esto, la cantidad de valores de frecuencias angulares se puede aumentar disminuyendo el espaciamiento $\Delta\bar{\omega}$, con lo que se mejora la resolución de la solución.

Para lograr esta mejora de resolución, se utiliza el método llamado Zero Padding, el que consiste, básicamente, en agregar ceros al final de la función con tal de aumentar su longitud, aumentando T_p, disminuyendo el espaciamiento.

Cabe destacar que el rango de valores de frecuencias (no angulares) dentro del cual se tiene este espaciamiento es aproximadamente (dependiendo igualmente de si el número de muestras es par o impar) desde $-Fs/2$ hasta $Fs/2$, lo que en frecuencias angulares ($\omega = 2\pi f$) se transforma a un rango entre $-\pi \cdot Fs$ y $\pi \cdot Fs$. Este rango, como se puede ver, no depende del largo de la función, por lo tanto al agregar ceros, se mantiene constante. Por esta razón, al disminuir el espaciamiento y mantenerse el rango, la resolución aumenta considerablemente.

Matlab en su función fft ("Fast Fourier Transform") da la opción de entregar el largo hasta el cual se quiere alargar la función 'N', donde si N es menor al largo de la función (sin zero padding), usa el largo de la función (es decir, no "acorta" la función). Valores recomendables de 'N' son las potencias de 2. Por ejemplo, se puede tomar la 4 siguiente potencia de 2 usando el código que sigue:

```
N_tot = 2^(nextpow2(N)+4);
```

Cabe destacar finalmente, que al volver del espacio de las frecuencias al espacio del tiempo, se generan números complejos con su parte imaginaria despreciable, por lo que se obtiene solo la parte real como solución. Es recomendable comprobar que efectivamente la parte compleja es despreciable, verificando el mayor de sus valores absolutos.

[1] Apuntes de Clases - Dinámica Avanzada de Estructuras. Página 75.

Detalles sobre implementación en Matlab:

La implementación en Matlab es relativamente sencilla, una vez entendido el procedimiento que usa el programa al hacer la función fft. La función fft(P,N) entrega una matriz en que CADA COLUMNA contiene la Transformada de Fourier de CADA COLUMNA de P. Además hace zero padding usando N como largo total. Si N es menor al largo de la señal, entonces no hace zero padding, mientras que si es mayor, sí lo hace.

Esta Transformada de Fourier son números complejos, y se puede **graficar el módulo de estos para comprobar su forma.** Esta forma no es la forma real de la Transformada, y hay que re-ordenarla para que tenga la forma que ustedes conocen. Antes de hacer ifft (Transformada Inversa) deben "re-desordenarla" a la forma inicial, ya que esta es la forma que maneja Matlab.

Para definir el eje de las frecuencias angulares, se debe usar el siguiente código:

```
T = Ntot/Fs;    %T: tiempo total del registro
dw = 2*pi/T;    %dw: espaciamiento en las frecuencias angulares
if mod(Ntot,2)==0  %N par
    nn = -Ntot/2:Ntot/2-1;
else               %N impar
    nn = -(Ntot-1)/2:(Ntot-1)/2;
end
w = dw*nn;       %vector de frecuencias angulares
```

Luego para que calce la Transformada de Fourier con el dominio de las frecuencias recién formado, se usa la función fftshift. Recordar que fft actúa sobre las columnas, por lo tanto se debe trasponer el vector P para que quede en cada columna, la carga en el tiempo de cada grado de libertad hacia abajo por las filas.

```
Pw = fft(P',Ntot);
Pw = fftshift(Pw,1);    %Se ordena para calzar con dominio de frecuencias

%Se vuelve a trasponer para dejar tiempo en filas y gdl en columnas
Pw = Pw.';
```

Notar que para trasponer la matriz Pw se debe usar .' en vez de ' ya que si se usa este último el resultado es la matriz transpuesta conjugada, mientras que .' solo transpone. Además, el "1" de fftshift es importante, ya que algunas versiones de Matlab más nuevas, si no se agrega tal dimensión, además de hacer el shift que queremos, re-ordena las columnas, cosa que no queremos en este caso.

Luego de calculado el vector de desplazamiento en el espacio de las frecuencias (y a partir de este, el de velocidad y aceleración usando las ecuaciones mostradas arriba), se debe "re-desordenar" para hacer la Transformada Inversa, usando ifftshift, y pasar al dominio del tiempo nuevamente:

```
Vw = ifftshift(Vw.',1);    %Se deja frecuencias en filas, gdl en columnas
v = ifft(Vw)';      %Se vuelve a trasponer para dejar tiempo en columnas
```

Notar que Vw se construyó de tal manera que en cada fila se tuviera la transformada de un grado de libertad hacia el lado en el dominio de las frecuencias a través de las columnas.

Se comprueba que efectivamente los imaginarios son despreciables:

```
disp(max(max(imag(v)))) %Se muestra que el máximo de los imaginarios es
despreciable
v = real(v);
```

Finalmente la señal se debe truncar, ya que si se grafica tal como está, se verá el Zero Padding después de la señal:

```
v = v(:,1:N);
```

El mismo procedimiento se hace con la velocidad y la aceleración, para pasar al dominio del tiempo.

Espero que este documento sea de ayuda para su examen. Cualquier duda, no duden en preguntarme vía mail o ucursos.