Linea 1: Definizione della funzione

Matlab

```
function L = leb_con(z, x)
```

Dichiara una funzione MATLAB che:

- **Prende in input:** `z` (nodi di interpolazione) e `x` (punti di valutazione)
- **Restituisce:** `L` (la costante di Lebesgue massima)

Linee 2-9: Commenti documentazione

Matlab

```
% LEB_CON Approssimazione della costante di Lebesgue sui punti x
%   L = leb_con(z, x)
%   INPUT:
%     z : vettore riga dei nodi dell'interpolante (length n = d+1)
%     x : vettore colonna dei punti in [-1,1] dove valutare...
%   OUTPUT:
%     L : scalare reale, max_{x} lambda_n(x)
```

Spiegano come usare la funzione (INPUT/OUTPUT).

Linea 11: Trasformazione di z in vettore riga

Matlab

```
z = z(:).';
```

- `z(:)` – converte z in vettore colonna
- `.'` – lo traspone in vettore riga (per facilitare i calcoli)

Linea 12: Trasformazione di x in vettore colonna

Matlab

```
x = x(:);
```

Assicura che `x` sia un vettore colonna (ogni punto su una riga diversa).

Linea 13: Calcolo del numero di nodi

Matlab

```
n = numel(z);
```

`numel()` conta quanti elementi ha il vettore `z`.

Linee 15-20: Calcolo dei pesi baricentrici

Matlab

```
W = ones(1, n);             % inizializza W a 1
for i = 1:n
    diffs = z(i) - z;       % differenza tra z(i) e tutti gli altri nodi
    diffs(i) = 1;           % evita divisione per zero
    W(i) = 1 / prod(diffs); % W(i) = 1 / prodotto delle differenze
end
```

Calcola i **pesi baricentrici**, fondamentali per l'interpolazione. Sono fattori che pesano l'importanza di ogni nodo.

Linea 22: Replica di x

Matlab

```
X = x * ones(1, n);
```

Crea una matrice dove ogni riga è il vettore `x`. Utile per fare operazioni vettoriali.

Linea 23: Replica di z

Matlab

```
Z = ones(numel(x), 1) * z;
```

Crea una matrice dove ogni riga è il vettore `z`.

Esempio: Se `x` ha 3 elementi e `z` ha 2 elementi:

$$X = \begin{bmatrix} x_1 & x_1 \\ x_2 & x_2 \\ x_3 & x_3 \end{bmatrix} \quad Z = \begin{bmatrix} z_1 & z_2 \\ z_1 & z_2 \\ z_1 & z_2 \end{bmatrix}$$

Linea 24: Calcolo della matrice A

Matlab

```
A = W ./ (X - Z);
```

Formula baricentrica: divide i pesi `W` per le differenze `(x - z)`. Ogni elemento = $w_j/(x_i - z_j)$.

Linea 26: Tolleranza per confronti

Matlab

```
tol = 1e-14;
```

Valore molto piccolo (0.00000000000001) usato per identificare quando un punto coincide con un nodo.

Linea 27: Identifica i nodi

Matlab

```
is_node = abs(X - Z) < tol;
```

Crea una matrice logica: `true` se la distanza tra `x(i)` e `z(j)` è quasi zero (stesso punto).

Linea 29: Somma delle righe di A

Matlab

```
S = sum(A, 2);
```

Somma ogni riga di `A`. $S(i)$ = somma degli elementi della riga i.

Linea 30: Calcolo della funzione di Lebesgue

Matlab

```
lambda = sum(abs(A), 2) ./ abs(S);
```

La funzione di Lebesgue: $\lambda_n(x) = \frac{\sum_j |w_j/(x-z_j)|}{|\sum_j w_j/(x-z_j)|}$

Misura l'instabilità dell'interpolazione in ogni punto `x`.

Linee 32-33: Correzione per i nodi

Matlab

```
any_node = any(is_node, 2);    % true se la riga contiene almeno un true
lambda(any_node) = 1;          % nei nodi, la funzione di Lebesgue vale 1
```

Nei punti di interpolazione, la costante di Lebesgue è sempre 1 per definizione.

Linea 35: Valore massimo

Matlab

```
L = max(lambda);
```

Restituisce il massimo della funzione di Lebesgue su tutti i punti `x`. È la "costante di Lebesgue" finale.

Riassunto

Questo codice implementa l'interpolazione **baricentrica** per calcolare la costante di Lebesgue, un indicatore di qualità della scelta dei nodi di interpolazione.

Valori bassi = interpolazione stabile, **Valori alti** = instabile.