Test funzione pagerank.mlx

Casi d'uso della funzione *pagerank.mlx* .

mathwork200.mat

Calcolo del pagerank con la funzione pagerank.

Caso d'uso con la matrice presente nel file *mathwork200.mat*. Utilizziamo la funzione *pagerank.mlx* che restituisce i rank delle varie pagine e verifichiamo che la somma sia pari a 1 e che tutti gli elementi siano non negativi.

```
load mathwork200.mat;
R = pagerank(G)
```

```
R = 200×1
     1.759217077077456e-03
     5.536967249760527e-03
     4.787988545059624e-03
     4.798261903167869e-03
     4.798261903167869e-03
     4.787988545059624e-03
     4.787988545059624e-03
     4.787988545059624e-03
     4.787988545059624e-03
     4.787988545059624e-03
     ⋮
```

```
somma = sum(R)
```

```
somma =
     1.000000000000001e+00
```

```
elementi_negativi = R(R<0)
```

```
elementi_negativi =

  0×1 empty double column vector
```

Calcolo accuratezza con il confronto con *centrality*.

```
gs = digraph(G','omitselfloops');
Pk = centrality(gs,'pagerank','FollowProbability',0.85, 'Tolerance',10^-7)
```

```
Pk = 200×1
     1.759218218273203e-03
     5.536987754565612e-03
     4.788004697167616e-03
     4.798278107809245e-03
     4.798278107809245e-03
     4.788004697167616e-03
```

```
     4.788004697167616e-03
     4.788004697167616e-03
     4.788004697167616e-03
     4.788004697167616e-03
     ⋮
```

```
err_rel = norm(R - Pk)/norm(Pk)
```

```
err_rel =
     2.255144023869366e-06
```

repubblica.mat

Calcolo del pagerank con la funzione pagerank.

Caso d'uso con la matrice presente nel file *repubblica.mat*. Utilizziamo la funzione *pagerank.mlx* che restituisce i rank delle varie pagine e verifichiamo che la somma sia pari a 1 e che tutti gli elementi siano non negativi.

```
load repubblica.mat;
R = pagerank(G)
```

```
R = 100×1
     3.664699397456729e-03
     1.029684187083375e-02
     1.116717668417418e-02
     1.366765578072275e-02
     1.127516818154505e-02
     1.180565271229337e-02
     2.263673517866172e-02
     1.180565271229337e-02
     1.180565271229337e-02
     9.317240410089991e-02
     ⋮
```

```
somma = sum(R)
```

```
somma =
     1.000000000000001e+00
```

```
elementi_negativi = R(R<0)
```

```
elementi_negativi =

  0×1 empty double column vector
```

Calcolo accuratezza con il confronto con *centrality*.

```
gs = digraph(G','omitselfloops');
Pk = centrality(gs,'pagerank','FollowProbability',0.85, 'Tolerance',10^-7)
```

```
Pk = 100×1
```

```
      3.664699631657786e-03
      1.029684172841195e-02
      1.116717613261781e-02
      1.366765616199609e-02
      1.127516762994737e-02
      1.180565199946794e-02
      2.263673331531058e-02
      1.180565199946794e-02
      1.180565199946794e-02
      9.317241228041624e-02
      ⋮
```

```
err_rel = norm(R - Pk)/norm(Pk)
```

```
err_rel =
      1.328518307259801e-07
```

ilsole24.mat - Generato attraverso surfer.m

Calcolo del pagerank con la funzione pagerank.

Caso d'uso con la matrice presente nel file *ilsole24.mat*. Utilizziamo la funzione *pagerank.mlx* che restituisce i rank delle varie pagine e verifichiamo che la somma sia pari a 1 e che tutti gli elementi siano non negativi.

```
load ilsole24.mat;
R = pagerank(G)
```

```
R = 50×1
      1.754064806994633e-02
      5.501631556589102e-02
      3.832664334518363e-02
      2.163697112447623e-02
      1.784492461948087e-02
      1.784492461948087e-02
      1.784492461948087e-02
      1.784492461948087e-02
      1.784492461948087e-02
      1.784492461948087e-02
      ⋮
```

```
somma = sum(R)
```

```
somma =
      1.000000000000000e+00
```

```
elementi_negativi = R(R<0)
```

```
elementi_negativi =

  0×1 empty double column vector
```

Calcolo accuratezza con il confronto con *centrality*.

```
gs = digraph(G','omitselfloops');
Pk = centrality(gs,'pagerank','FollowProbability',0.85, 'Tolerance',10^-7)
```

```
Pk = 50×1
     1.754064989155613e-02
     5.501627210757631e-02
     3.832662297948022e-02
     2.163697385138413e-02
     1.784492650363190e-02
     1.784492650363190e-02
     1.784492650363190e-02
     1.784492650363190e-02
     1.784492650363190e-02
     1.784492650363190e-02
     ⋮
```

```
err_rel = norm(R - Pk)/norm(Pk)
```

```
err_rel =
     3.651360140139452e-07
```

galeazzi.mat - Generato attraverso surfer.m

Calcolo del pagerank con la funzione pagerank.

Caso d'uso con la matrice presente nel file *galeazzi.mat*. Utilizziamo la funzione *pagerank.mlx* che restituisce i rank delle varie pagine e verifichiamo che la somma sia pari a 1 e che tutti gli elementi siano non negativi.

```
load galeazzi.mat;
R = pagerank(G)
```

```
R = 100×1
     5.757299647526484e-03
     1.445745499257526e-02
     2.310100886932469e-02
     6.369012748855036e-03
     6.369012748855036e-03
     6.369012748855036e-03
     6.369012748855036e-03
     6.369012748855036e-03
     6.369012748855036e-03
     9.603584402562067e-03
     ⋮
```

```
somma = sum(R)
```

```
somma =
     1.000000000000005e+00
```

```
elementi_negativi = R(R<0)
```

```
elementi_negativi =
```

```
0×1 empty double column vector
```

Calcolo accuratezza con il confronto con *centrality*.

```
gs = digraph(G','omitselfloops');
Pk = centrality(gs,'pagerank','FollowProbability',0.85, 'Tolerance',10^-7)
```

```
Pk = 100×1
     5.757300357651519e-03
     1.445745708004059e-02
     2.310101420449277e-02
     6.369013550566284e-03
     6.369013550566284e-03
     6.369013550566284e-03
     6.369013550566284e-03
     6.369013550566284e-03
     6.369013550566284e-03
     9.603585701942361e-03
     ⋮
```

```
err_rel = norm(R - Pk)/norm(Pk)
```

```
err_rel =
     6.283601285943538e-07
```

Casi di errore della funzione *pagerank.mlx* .

Caso in cui la matrice in ingresso è vuota:

```
G = [];
R = pagerank(G);
```

```
Error using pagerank (line 8)
La matrice di input è vuota.
```

Caso in cui la matrice in ingresso non è sparsa:

```
G = [1 2 4; 1 5 6; 4 5 6];
R = pagerank(G);
```

```
Error using pagerank (line 12)
La matrice di input non è sparsa.
```

Caso in cui la matrice non è di tipo sparse logic:

```
G = sparse([1 0 0; 0 1 0; 0 0 1]);
R = pagerank(G);
```

```
Error using pagerank (line 16)
La matrice di input non contiene solo elementi logici.
```

Caso in cui la matrice in ingresso non è quadrata:

```
G = sparse(logical([1 0 0; 0 1 0]));
R = pagerank(G);
```

```
Error using pagerank (line 22)
La matrice di input non è quadrata.
```

Caso in cui l'ingresso è un elemento logico:

```
G = sparse(logical(1));
R = pagerank(G)
```

```
Error using pagerank (line 26)
Dimensioni della matrice di input non corrette.
```