usage of 2-factor algorithms:
[W,H,iterdone,costhistory] = nmf(V,m,cost,iter_allowed,time_allowed)
[W,H,iterdone,costhistory] = nsnmf(V,m,ns,cost,iter_allowed,time_allowed)
[W,H,iterdone,costhistory,kl_and_sums] = lnmf(V,m,a,b,iter_allowed,time_allowed)
[W,H,iterdone,costhistory] = nmfsc(V,sW,sH,m,iter_allowed,time_allowed)
[W,H,iterdone,costhistory] = snmf(V,a,m,iter_allowed,time_allowed)

parameters:

- V: the data matrix
- m: inner dimension
- *(nsnmf only:)* ns: the degree of nonsmoothing of both W and H, in [0,1]; use 0 for standard NMF and larger values for smoother result matrices
- *(lnmf only:)* a, b: a controls smoothness of W and b controls sparsity of H
- *(nmfsc only:)* sW, sH: sparseness of W, H; both in [0,1]; give [] if no constraint
- *(snmf only:)* a: positive sparsity parameter for H; use 0 for standard NMF
- cost: cost function; either 'eucl' (Euclidean squared error) or 'kl' (extended Kullback-Leibler divergence)
- iter_allowed: maximum number of iterations i.e. how many times the result matrices can be updated before the simulation stops
- time_allowed: maximum running time in seconds; however, the updates of the current iteration are completed before stopping the simulation

output:

- W, H: matrices whose matrix product WH estimates V
- iterdone: number of iterations performed
- costhistory: a vector containing values of the cost function after each iteration
- *(lnmf only:)* kl_and_sums: a 3-column matrix; use *cost = lnmf_costs(kl_and_sums,a,b)* to calculate values of the lnmf cost functions with different sparsity parameters a and b

usage of 3-factor algorithms:
[W,H,P,iterdone,costhistory] = nmf3(V,m1,m2,normd,cost,iter_allowed,time_allowed)
[W,H,P,iterdone,costhistory] =
nmf3_nonNaN(V,m1,m2,normd,cost,iter_allowed,time_allowed)
[W,H,P,iterdone,costhistory] = snmf3(V,alpha,m1,m2,iter_allowed,time_allowed)

parameters:

- V: the data matrix
- *(snmf3 only:)* alpha: positive sparsity parameter for P; use 0 for standard NMF3
- m1, m2: inner dimensions
- *(nmf3 and nmf3_nonNaN only:)* normd: use 'W' or 'H' to normalize W (column-wise) or H (row-wise); in either case, P is row-normalized
- cost: cost function; either 'eucl' (Euclidean squared error) or 'kl' (extended Kullback-Leibler divergence)

- iter_allowed: maximum number of iterations i.e. how many times the result matrices can be updated before the simulation stops
- time_allowed: maximum running time in seconds; however, the updates of the current iteration are completed before stopping the simulation

output:

- W, H, P: matrices whose matrix product WHP estimates V
- iterdone: number of iterations performed
- costhistory: a vector containing values of the cost function after each iteration

usage of other algorithms:

cost = lnmf_costs(kl_and_sums,a,b)

- calculates values of the lnmf cost functions with different sparsity parameters a and b; parameter kl_and_sums is a matrix the lnmf algorithm outputs

s = sparsity(A)

- calculates the sparsity of the matrix A as defined by Hoyer (2004)