QFCAPX

Approximates QFC with triangle inequation, norming is pulled out to input stage kernel is feeded with norm vectors or scalars instead of matrices.

Syntax

```
K = QFCAPX(ax, bx, ay, by, theta)
```

Description

K = QFCAPX(ax, bx, ay, by, theta) computes quadratic distances bewtween data points and parametrize it with height and length scales. Computes distance with quadratic euclidian norm.

Input Argurments

ax vector of cosine simulation components.

bx vector of cosine simulation components.

ay vector of sine simulation components.

by vector of sine simulation components.

theta vector of kernel parameters.

Output Argurments

K noise free covarianc matrix.

Requirements

- Other m-files required: None
- Subfunctions: None
- MAT-files required: None

See Also

- initQFCAPX
- meanPolyQFCAPX


```
function K = QFCAPX(ax, bx, ay, by, theta)
     arguments
        % validate data as real vector of same size
        ax (:,:,:) double {mustBeReal}
        bx (:,:,:) double {mustBeReal, mustBeFitSize(ax,bx)}
        ay (:,:,:) double {mustBeReal, mustBeFitSize(ax,ay)}
        by (:,:,:) double {mustBeReal, mustBeFitSize(ax,by)}
        % validate kernel parameters as 1x2 vector
        theta (1,2) double {mustBeReal}
    end

    % get number of observations for each dataset, cosine and sine
    M = length(ax);
    N = length(bx);
```

```
    % expand covariance parameters, variance and lengthscale
    c2 = 2 * theta(2)^2; % 2*sl^2
    c1 = theta(1) * c2;   % s2f * c

    % allocate memory for K
    K = zeros(M, N);

    % loop through observation points and compute the covariance for each
    % observation against another
    for m = 1:M
        for n = 1:N
            % get distance between m-th and n-th observation
            % compute distance with quadratic frobenius normed vectors
            r2 = (ax(m) - bx(n))^2 + (ay(m) - by(n))^2;

            % engage lengthscale and variance on distance
            K(m,n) = c1 / (c2 + r2);

        end
    end
end

function mustBeFitSize(a, b)
    % Test for equal size
    if ~isequal(size(a,1,2), size(b,1,2))
        eid = 'Size:notEqual';
        msg = 'Sizes of  are not fitting.';
        throwAsCaller(MException(eid,msg))
    end
end
```

Published with MATLAB® R2020b