# initKernelParameters

Init GPR model on current kernel parameters, computes covariance matrix and depending kernel values means, mean coefficients, regression weights and likelihoods.

## Syntax

```
Mdl = initKernelParameters(Mdl)
```

## Description

**Mdl = initKernelParameters(Mdl)** initializes the regresion model in final.

## Input Argurments

**Mdl** model struct.

## Output Argurments

**Mdl** initialized regression model.

## Requirements

- Other m-files required: basicMathFunctions, kernelQFC, kernelQFCAPX
- Subfunctions: None
- MAT-files required: None

## See Also

- basicMathFunctions
- kernelQFC
- kernelQFCAPX
- initGPR

Created on November 06. 2019 by Klaus Jünemann. Copyright Klaus Jünemann 2019.

```
function Mdl = initKernelParameters(Mdl)

    % compute noise free covariance matrix
    Mdl.Ky = Mdl.kernelFun(Mdl.Xcos, Mdl.Xcos, Mdl.Xsin, Mdl.Xsin, Mdl.theta);

    % add noise to covariance matrix along its diagonal, which is noise on
    % trainig observations with itself
    Mdl.Ky = addNoise2Covariance(Mdl.Ky, Mdl.s2n);

    % compute the cholesky decomposition of the covariance matrix and the log
    % determinate of the covariance matrix, computes lower triangle matrix
    [Mdl.L, Mdl.logDet] = decomposeChol(Mdl.Ky);

    % compute beta coefficients to fit H matrices of cosine and sine mean
    % function if none zero mean is set as model mean, for zero mean all beta
    % and related means are zero.
    switch Mdl.mean
        case 'zero'
            % set not needed kernel parameters to zero,
            % beta is not used in zero mean GPR and so all related means are
            % zero, as name lets expect
            Mdl.BetaCos = 0;
            Mdl.BetaSin = 0;
            Mdl.meanFunCos = @(X) 0;
            Mdl.meanFunSin = @(X) 0;

        case 'poly'
            % estimate beta for none zero H matrices
            Mdl.BetaCos = estimateBeta(Mdl.basisFun(Mdl.Xcos), Mdl.L, Mdl.Ycos);
            Mdl.BetaSin = estimateBeta(Mdl.basisFun(Mdl.Xsin), Mdl.L, Mdl.Ysin);

            % mean function for polynom approximated mean H' * beta
            Mdl.meanFunCos = @(X) Mdl.basisFun(X)' * Mdl.BetaCos;
            Mdl.meanFunSin = @(X) Mdl.basisFun(X)' * Mdl.BetaSin;

        otherwise
            error('Unsupported mean function %s in beta estimation.', Mdl.mean);
    end

    % compute weights for cosine and sine, angles in rads and radius
    Mdl.AlphaCos = computeAlphaWeights(Mdl.L, Mdl.Ycos, ...
        Mdl.meanFunCos(Mdl.Xcos));
    Mdl.AlphaSin = computeAlphaWeights(Mdl.L, Mdl.Ysin, ...
        Mdl.meanFunSin(Mdl.Xsin));

    % compute log marginal likelihoods for each cosine and sine weights
    Mdl.LMLcos = computeLogLikelihood(Mdl.Ycos, Mdl.meanFunCos(Mdl.Xcos), ...
        Mdl.AlphaCos, Mdl.logDet, Mdl.N);
    Mdl.LMLsin = computeLogLikelihood(Mdl.Ysin, Mdl.meanFunSin(Mdl.Xsin), ...
        Mdl.AlphaSin, Mdl.logDet, Mdl.N);
end
```

*Published with MATLAB® R2020b*