# MPC控制器的基本原理

对于离散LTI系统，其系统方程为：





其中，下标*k*表示系统当前所处时刻，*k*+1为系统的下一时刻。下标时刻间隔是离散化的，与实际系统的步长无关。采用不同的计算步长，系统的状态矩阵也不同。

MPC假设系统满足马尔可夫假设，即系统未来的状态与系统过去的状态无关，系统在每一个时刻的状态都只受到前一刻状态和控制输入的影响。因此对于一个确定的控制输入序列，系统的状态轨迹可以通过式递推得到。

在实际应用中，控制输入往往受到执行器特性的限制，不能在短时间内发生较大的变化。因此，实际中往往采用输入增量控制，以每一时刻的控制量增量作为考察控制器性能的指标之一。即系统变形为：



递推至观测窗末尾，有：



以系统状态轨迹与设定状态轨迹的跟踪误差定义系统的代价函数*J*：



其中，称为预测步长，称为控制步长。为节省计算量，有时取，即控制步长以外至预测步长之间的窗口内，控制量取控制窗口的最后一个量不变。*S*、*Q*、*R*为认为给定的对角权重矩阵，分别确定终端误差、累计轨迹误差和累计控制增量对控制性能的影响程度。

将系统方程代入代价函数，即得到代价函数*J*关于控制轨迹的函数：



通过求取代价函数的极小值，即可得到未来一段时间内系统的一个较优控制序列。MPC控制器将保留作为系统当前时刻的实际控制量，发出控制指令。在下一时刻，根据系统新的状态量进行新一轮的计算。

代价函数可以采用不同的范数，最终形成不同的优化问题。采用一阶范数时，求解问题为线性规划（LP）问题，采用二阶范数时，求解问题为二次规划（QP）问题。考虑实际控制问题的边界要求、执行器饱和等问题，还可以对状态参数*x*、控制量*u*添加一系列等式/不等式限制条件（硬约束），通过求解带约束的规划问题，获取期望的控制器特性。

标准形式的硬约束MPC优化问题可表示为状态量与控制量的任意线性组合满足给定不等式，即：



# 代价函数的推导

MPC控制器的一大特点是可以根据实际问题的需求，对系统模型和代价函数进行改进。通过、、，求解优化问题，从而满足不同的控制需求。

一般情况下，代价函数采取二次型，故代价函数可按如下步骤化简：

1. 记：



即系统有*n*个状态变量，*r*个输出，*m*个控制输入。

1. 记：



将以进行整理，将代入，有：



记：



有：



将代入，（[计算过程在此](#化简J)）有：



去除不含的项（因为过去不可改变，所以解算优化问题时只保留对未来的影响），有：



进一步定义：



代价函数最终简化为：



至此，在系统的、、矩阵已知的情况下，给出预测步长、控制步长，即可得到代价函数关于控制量增量的函数。

# 基于卡尔曼滤波的控制校正

MPC算法本身基于模型完全精确的假设。因此，如果实际过程中计算采用的的状态量包含噪声，MPC给出的控制量也会受到噪声影响。一般需要在算法中对状态量进行一定形式的反馈，以消除噪声。

对进行改动，考虑一个带有过程噪声的LTI系统：







其中，为时刻的测量量，为过程噪声，为观测噪声。其正态协方差矩阵分别为：





在各变量噪声独立的情况下，有：





采用卡尔曼滤波时，根据，可以通过上一个时刻测量到的测量量得到一个观测估计量：



根据系统LTI方程，若已知上一时刻的状态估计量，可以计算出一个先验状态估计:



先验状态估计包含系统模型的信息，观测估计量包含实际系统的噪声信息。若将二者通过线性组合方式进行融合，即可得到一个既包含模型、又包含噪声的估计量：



式中，称为卡尔曼增益矩阵，称为残差。

显然，由于后验估计是先验估计和观测估计的线性组合，它的满足的分布也是先验估计和观测估计的线性组合。进一步，后验估计也应当满足正态分布。同时，由于采用的构造方法，不难看出，先验估计、观测估计、后验估计的期望均和状态真实值一致，后验估计的分布同时受到*Q*、*R*、LTI系数矩阵和时刻状态值的影响。不妨设时刻后验估计的方差阵为，先验估计的方差阵为则：



将代入，有：



简单变形后，使两侧的分布都去除均值：



其中，,。但由于二者都与状态转移过程有关，二者并不独立。

由，有：



后验估计方差阵的迹最小时，后验估计的估计最准确。因此，对的迹关于求偏导，得到迹最小时，的取值，即为最佳卡尔曼增益：



将代入，有：



注意到，，因此初始估计导致的后验估计方差会不断减小，因此，时间越久，理论上估计将越精确。

# MPC整体流程

1. 建立系统模型

确定向量，及其边界条件，取得矩阵。

1. 设定初始参数与优化

设定步长，设定干扰的分布。

1. 预先计算中间量

计算存入内存，方便计算。

1. 初始化算法

采用当前测量值初始化先验和后验估计、卡尔曼增益：







采用初始化的值，进行一次预测更新：



计算出下一个时刻的控制输出用于控制。

1. 开始算法

根据，更新先验估计：



根据推理，可更新先验协方差矩阵：



于是根据可计算卡尔曼增益：



根据可更新后验估计：



根据更新后验协方差阵：



根据、，由得到矩阵:



最后，采用进行预测更新：



计算出下一个时刻的控制输出用于控制，如此重复。

化简J

import sympy as sp

n,r,m,Np,Nc = sp.symbols('n,r,m,Np,Nc',type='int')

n,r,m,Np,Nc = 5 , 4, 3, 15, 5

Phi = sp.MatrixSymbol(r'\Phi\_{[k]}',r\*Np,n)

Phiminus = sp.MatrixSymbol(r'\Phi\_{[k-1]}',r\*Np,n)

Gamma = sp.MatrixSymbol(r'\Gamma',r\*Np,m\*Nc)

Yk = sp.MatrixSymbol('Y\_{[k]}',r\*Np,1)

xk = sp.MatrixSymbol('x\_{[k|k]}',n,1)

xkminus = sp.MatrixSymbol('x\_{[k-1|k]}',n,1)

yk = sp.MatrixSymbol('y\_{[k|k]}',r,1)

ykminus = sp.MatrixSymbol('y\_{[k-1|k]}',r,1)

Q = sp.MatrixSymbol('Q',r,r)

Omega = sp.MatrixSymbol(r'\Omega',r\*Np,r\*Np)

Psi = sp.MatrixSymbol(r'\Omega',m\*Nc,m\*Nc)

dUk = sp.MatrixSymbol(r'\Delta U\_{[k]}',m\*Nc,1 )

Yk = Phi\*xk -Phiminus\*xkminus+Gamma\*dUk

J = yk.T\*Q\*yk/2+Yk.T\*Omega\*Yk/2+dUk.T\*Psi\*dUk/2

J

#n,r,m,Np,Nc分别采用了不同的值，可以确保推导中矩阵和向量的维数是完全正确的。