Comparisons of Interpolation Methods for the uncertain case

Yihang Ding and J.A Rossiter

University of Sheffield
Department of Automatic Control and Systems Engineering
E-Mail : cop04yd@sheffield.ac.uk

Abstract—A recent paper [13] proposed a new interpolation method which can enlarge feasible regions but requires fewer degrees of freedom than many existing approaches. This paper shows that one can use some, by now, common techniques to extend this to the uncertain case. Moreover, the paper gives a proper critique by providing a comparison with many existing interpolation methods. In addition, the paper gives a proper critique by providing a comparison with many existing interpolation methods. In addition, the example section makes use of some higher order examples whereas earlier papers have tended to focus on 2nd order models only. These examples demonstrate that the proposed approach has a useful place in the portfolio of algorithms available to a designer.

Keywords: constraints, interpolation, feasibility, computational efficiency, uncertain, LPV

I. INTRODUCTION

Model Predictive Control (MPC) [3], [10], is one of the most important advanced control techniques to have had a significant and widespread impact on industrial process control. A common objective in the MPC community is guaranteeing asymptotic stability and recursive constraint satisfaction for a set of initial states that is as large as possible and with both a minimal control cost and computational load. Interpolation techniques [9], [1] provide a favorable trade off between these different aspects. Typically there is a conflict existing between the computational efficiency, which depends on the number of d.o.f., the volume of the feasible region and the performance. To this end, a recent paper [13] proposed an interpolation method that makes use of the best facets of the existing ones, that is it combines the feasibility advantages of interpolation with a systematic selection of just two d.o.f.

Unfortunately, MPC algorithms are often specified for a linear or nominal case and it is necessary to assume that either, the inherent robustness of the approach or some form of backoff, will negate the effects of uncertainty. Hence, there is much interest in how to extend MPC to cater explicitly for parameter uncertainty. This paper focuses on uncertainty modeled as a linear parameter varying (LPV) system. The predominant number of papers in the literature use ellipsoidal invariance as a key tool in establishing the stability of LPV system. This is probably because one can use linear matrix inequalities (LMI) to set up conditions for feasibility, stability and convergence and LMIs give rise to convex optimizations. The flip side however is that the optimizations can be significantly more demanding than quadratic programming (QP) and feasible regions are restricted to ellipsoids, which in general can be far smaller that polyhedrals.

Thus, authors [7] worked on finding the maximal admissible set (MAS) [4] for the uncertain case. Given this, it is straightforward to extend MPC techniques and interpolation based constraint handling [11] to cater for LPV systems. Thus, one minor contribution of this paper is to establish that the interpolation proposed in [14] can indeed use recent insights to be extended to the uncertain case. Having established this, the next key questions is whether the proposed interpolation has any value, that is, how does it compare to existing techniques, including a conventional MPC algorithm? It is demonstrated through examples that it does indeed give a middle ground between computational complexity and flexibility. Moreover, this paper uses higher order examples than typically adopted in the literature, thus testing algorithms in a tougher scenario.

Section 2 will give some background to MPC, the modeling assumptions and a quick review how we extend the original algorithms to LPV system. Section 3 proposed how to combine the new interpolation method with LPV system. Section 4 gives three simulation examples after which the paper finishes with conclusions and suggestions for future work.

II. BACKGROUND

This section introduces standard material from the existing literature on MPC, invariant sets and some basic interpolation schemes for uncertain systems.
A. Model and objective

This paper considers uncertain systems of the form

$$x_{k+1} = A(k)x_k + B(k)u_k, \quad k = 0, \ldots, \infty$$ (1)

where \((A(k), B(k)) \in C_o(A_1, B_1), \ldots, (A_m, B_m)\) and subject to constraints:

$$u(k) \in U \equiv \{ u: u \leq \mu \leq \Pi \}, \quad k = 0, \ldots, \infty, \quad (2a)$$

$$x(k) \in X \equiv \{ x: \underline{x} \leq x \leq \overline{x} \}, \quad k = 0, \ldots, \infty. \quad (2b)$$

\(x(k) \in \mathbb{R}^{n_x}\) and \(u(k) \in \mathbb{R}^{n_u}\) denote state and input vectors at discrete time \(k\) with \(n_x\) and \(n_u\) respectively denoting the number of states and inputs of the system.

An underlying aim is to minimise an upper bound on a predicted cost of the form:

$$J = \sum_{k=0}^{\infty} (x(k)^T Q x(k) + u(k)^T R u(k)) \quad (3)$$

with \(Q \in \mathbb{R}^{n_x \times n_x}\) and \(R \in \mathbb{R}^{n_u \times n_u}\) positive definite state and input cost weighting matrices. Assume that one can choose from \(r\) different state feedback gains \(K_j, j = 1, \ldots, r\) (one of these, say \(K_1\), would logically be the unconstrained optimal minimising \(J\)) with which there are associated closed-loop state matrices:

$$u = -K_j x; \Phi_{ij} = A_i - B_i K_j, (j = 1, \ldots, r, i = 1, \ldots, m) \quad (4)$$

All the algorithms in this paper seek to minimise \(J\), subject to predicted constraint satisfaction (2). The key design parameter is how one sets up the flexibility in the class of predictions. Interpolation uses a class defined around different closed-loop behaviours (4) as opposed to varying the input predictions directly, as in more conventional MPC algorithms.

For the nominal case and a control law \(u = -Kx\), one can express an upper bound on (3) as \(J(x) = x^T V_0 x\) (for a suitable \(V_0\)); thus this could serve as a Lyapunov function in the unconstrained case. In general, one requirement of interpolation methods is a very similar quadratic stabilisability condition, that is for any of the allowable feedbacks \(K_i\), there must exist a quadratic Lyapunov function which applies irrespective of the variation in the process allowed in (1). Hence there must exist \(V_j\) such that:

$$V_j \Phi_{ij} V_j \Phi_{ij} \leq 0, \forall i; \quad (5)$$

These \(V_j\) will not match \(V_0\) in general. Use will be made of these functions in a later section, where the \(V_j\) is also chosen to give the largest feasible region, e.g. [5].

B. Polyhedral Invariant Sets for LPV systems

Under mild conditions, the maximum volume feasible (and invariant) region MAS [4] for a stable linear system with linear constraints is polyhedral (under mild conditions). Recently [7] it has been shown that so long as an LPV system is quadratically stabilisable, then the same statement holds.

Definition 2.1 (Invariant sets): Define the robust MAS for a given feedback \(K_1\) as \(S_1 = \{ x : M_i x \leq d_i \}\).

Remark 2.1: A MAS is invariant, so \(x(k) \in S \Rightarrow x(k+i) \in S, \forall i > 0\), irrespective of the variation of \((A(k), B(k))\). Moreover, the trajectories satisfy (2) and, from quadratic stabilisability (5), converge to the origin.

C. Polyhedron based GIMPC for the uncertain case

This brief section summarises the GIMPC algorithm [1] with later extensions to make use of polyhedral sets in the uncertain case [11]. A key in this algorithm is the decomposition of state \(x\) into several components \((x_i - these are vectors not axis values) which are then treated independently through different closed-loop dynamics.

Definition 2.2 (GIMPC cost function for uncertain case): An upper bound on the cost function \(J\) for the uncertain case is given as follows:

$$\hat{J} = \hat{x}^T P \hat{x} = \sum_{k=0}^{\infty} (x(k+1)^T Q x(k+1) + u(k)^T R u(k)) \quad (6)$$

$$\hat{J} = [x_1^T \ldots x_r^T] \Gamma_u + \Psi^T \Gamma \Psi + \Psi^T \Psi$$

Algorithm 2.1: [Polyhedral GIMPC for the uncertain case]

1) Define the MAS \(S_j, j = 1, \ldots, r\) for the \(r\) different feedbacks \(K_j, j = 1, \ldots, r\) corresponding to the uncertain system/constraints (1,2).

2) Define an appropriate least upper bound \(J = \hat{x}^T P \hat{x}\) for LPV system made up from (6).

3) Using these \(S_j\) and cost \(J\), at each sample instant, perform the optimisation:

$$\min_{x_j, \lambda_j} \hat{x}^T P \hat{x}, \quad \text{subject to (8)}, \quad (7)$$

$$x(0) = \sum_{j=1}^{r} x_j, \quad \text{with} \quad \sum_{j=1}^{r} \lambda_j = 1, \lambda_j \geq 0, \quad (8)$$

Theorem 2.1: Algorithm 2.1 has a guarantee of recursive feasibility and a guarantee of convergence when applied to system (1) [1].

D. Defining constraint based on GIMPC2 in the uncertain case

The GIMPC algorithm gives relatively conservative feasibility due to the constraints in (8). This was tackled in GIMPC2 by defining invariance for the entire vector \(\hat{x}\) as opposed to the individual components \(x_i\), thus removing the constraint on \(\lambda_i\). The offline computations, given next, are a little more demanding, but the improvements in feasibility can be substantial.

Given control law (9) and state decomposition:

$$u(k) = -\sum_{j=1}^{r} K_j x_j, \quad \lambda_r = x - x_1 - x_2 - \ldots - x_{r-1} \quad (9)$$

\footnote{More general linear state, input and mixed state/input constraints can also be considered.}
define an augmented state

\[ X = \begin{bmatrix} x \\ x_1 \\ \vdots \\ x_{r-1} \end{bmatrix} \] 

(10)

and hence an augmented uncertain but controlled system as

\[ X(k+1) = \Psi(k)X(k), \quad \Psi(k) \in \text{Co}(\Psi_1, \ldots, \Psi_m) \]

(11)

Rewrite constraints (2) in terms of \( X \) and then, find a corresponding robust MAS in the form

\[ S_a = \{ X : M_a X \leq d_a \} \] 

(12)

It is possible to project this back to \( x \)-space, but only (12) is needed for the online algorithm. By making use of (6) and (9), the cost \( J \) can also be readily expressed in terms of \( X \) thus giving a standard optimisation where the undefined part of \( X \) (i.e. not \( x \)) constitutes the d.o.f. for optimisation.

III. GIMPC2\( \beta \): REDUCING THE NUMBER OF DEGREES OF FREEDOM

A main weakness of GIMPC/GIMPC2 is that they give computational advantages over conventional MPC only for low order systems. This is because the number of d.o.f. (essential all the components of \( x_i, \forall i > 1 \)) required is linked explicitly to the state dimension. Thus, although they can be very powerful within that context, they do not tackle the issue of whether interpolation can be useful for higher order systems. Earlier proposals using just one d.o.f. gave limited feasibility gains or even non-convex feasibility regions [9] and thus, although useful at times, it was difficult to make any a priori statements.

This paper develops an idea in [13] which considered how one could make more selective search directions within the decomposition of \( X \) or \( \tilde{x} \) while not giving up the feasibility benefits. Clearly a reduction to just one search direction is insufficient and well studied [9], so a logical next step was to consider interpolations with just two d.o.f. [6] in more detail. If this can be shown to be effective, it would be a significant step forward compared to GIMPC & GIMPC2 methods for applications to higher order systems.

For completeness, this section implicitly includes algorithm development for the uncertain case.

A. Proposal summary

The proposal is to allow two search directions in the decomposition of \( x \), and thus make use of two d.o.f. Hence, define the search directions (or decomposition) as given by the actual state measurement \( x \) and a new direction (to be defined) \( \omega \), thus:

\[ x = x_1 + x_2 \]
\[ x_1 = (1 - \alpha)x + \beta \omega; \] 
\[ x_2 = \alpha x - \beta \omega; \] 

Thus, the aim within an MPC algorithm is to choose the d.o.f. \( \alpha, \beta \) such that the associated cost of (6) (where \( \tilde{x} \) has the same definition as in (6) but with \( x_1, x_2 \) defined by (13)) is minimised subject to constraint satisfaction.

Remark 3.1: It is straightforward [13] to show that the constraints and performance index can be computed in the form:

\[ [M_\alpha \quad M_\beta][\begin{array}{c} \alpha \\ \beta \end{array}] \leq d_{\alpha \beta} \] 

(14)

\[ J = [\alpha \quad \beta]S_{\alpha \beta}[\begin{array}{c} \alpha \\ \beta \end{array}] + [\alpha \quad \beta]P_{\alpha \beta} \]

(15)

for appropriate \( S_{\alpha \beta}, P_{\alpha \beta} \). The details are included in the appendix.

Given linear inequalities and a quadratic cost, the optimisation reduces to a two dimensional QP:

Algorithm 3.1 (GIMPC2\( \beta \) overview):

1. Define \( \omega \).
2. Update the constraint and cost from (14,15).
3. Minimise, w.r.t. \( \alpha, \beta \) the cost function subject to constraints:

\[ \min_{\alpha, \beta} J = [\alpha \quad \beta]S_{\alpha \beta}[\begin{array}{c} \alpha \\ \beta \end{array}] + [\alpha \quad \beta]P_{\alpha \beta} \]

s.t. \[ [M_\alpha \quad M_\beta][\begin{array}{c} \alpha \\ \beta \end{array}] \leq d_{\alpha \beta} \] 

(16)

4. Implement the control law as

\[ u = -K_1((1 - \alpha)x + \beta \omega) - K_2(\alpha x - \beta \omega) \] 

(17)

Remark 3.2: Due to the restriction to just two d.o.f., a simple proof of recursive feasibility, and thus convergence, is not straightforward for reasons similar to those given in [9]. However, it is accepted that, for the nominal case, one can always ride on the tail [6], so one simple option is to adopt the following rule. If the new optimum does not give a lower cost than the tail from the previous sample, use the tail which by definition must give a reduction in cost. Simple extensions of this principle for the uncertain case constitute work in progress as in that case Lypanuov arguments can break down.

B. Selecting the best search direction

The previous section avoided details such as how to chose the search direction and how to incorporate the uncertain case. In fact the latter is automatic if one builds constraint matrices and cost functions in the same way as described in the previous section, hence this section focuses on how to choose \( \omega \).

The intuitive argument is to choose a direction that is most likely to pull state trajectories away from constraints. However, an analytic solution is required so that the potentially high (for high order systems) dimensional optimisation of GIMPC is avoided when dealing directly with (12). Fortunately, invariant ellipsoids, although conservative by way of constraint handling, do nevertheless give insight into which directions move trajectories away from constraints. Thus, the aim is to choose a search direction \( \omega \) which is
best by way of ellipsoids, as this gives a simple analytic
answer, and then to use this in a stage two QP for further
improve the solution.

Algorithm 3.2 (The search direction $\omega$):
1) Define the maximum volume invariant ellipsoids
$V_1, V_2$ [5] for the given feedbacks $K_1$ and constraints
(2) as:
$$V_i = \{ x : x^T P_i x \leq 1 \}; \quad i = 1, 2 \quad (18)$$
2) Constraint satisfaction is ensured by (8) if $V \leq 1$, where
$$V = (x_1^T P_1 x_1)^{1/2} + (x_2^T P_2 x_2)^{1/2} \quad (19)$$
3) In general, distance from constraints is maximised by
minimising $V$. Thus suppose $x_1 = x + \omega$; $x_2 = x - \omega$,
then $V$ is minimized by:
$$\omega = 2 P_2 x / (x^T P_2 x)^{1/2} - 2 P_1 x / (x^T P_1 x)^{1/2} \quad (20)$$

Remark 3.3: One should add into the above that it is
intuitively logical to restrict the optimisation of $\omega$ to directions
which are orthogonal to $x$, as the decomposition of
(13) already allows variation in the $x$-direction. Nevertheless,
the author’s experience is that this may not make a
large difference in practice, although it can affect numerical
conditioning in (16).

IV. EXAMPLES

This section seeks to give a good testbed for the proposed
interpolation by utilising three examples. These have two,
three and four states respectively. In particular this section
gives comparisons of feasibility, performance and computa-
tional complexity for the three algorithms GIMPC, GIMPC2,
GIMPC2b and a conventional MPC algorithm. It will be
shown that, perhaps unsurprisingly, one can get the best
performance by using a conventional algorithm, but at the
cost of large numbers of d.o.f. Also, as previously shown,
GIMPC2 gives very good feasibility, especially compared
to GIMPC. Perhaps what is surprising however is that
GIMPC2b can give feasibility quite close to GIMPC, despite
the restriction to two search directions. In general, it easily
outperforms GIMPC and thus may be a candidate method
where small numbers of d.o.f are required.

A. Two state-space example

The first example is the popular double integrator:
$$A_1 = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (21)$$
$$A_2 = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0 \\ 1.5 \end{bmatrix}$$

$\pi = 1, \quad \bar{u} = -1, \quad \bar{x} = [10, \ 10]^T \quad (22)$

Two robustly stabilising control laws (optimal control with
$Q = \text{eye}(2), R_1 = 0.1, R_2 = 1$) are:
$$K_1 = [-0.3 \ -0.1]; \quad K_2 = [-0.5 \ -0.3] \quad (24)$$

Figure 1 gives the underlying MAS $S_1, S_2$, the feasible
region for GIMPC2, the feasible region for GIMPC2b and the
feasible regions for a conventional robust MPC (OMPC)
with 3, 5 d.o.f. (denoted as $n_c$). The figure clearly demon-
strates that GIMPC2b has good feasibility, in this case very
similar to GIMPC2 and far better than OMPC with many
more d.o.f.

In terms of control performance, closed-loop simulations
are taken from points where all three algorithms are
feasible, but near the boundary. The associated runtime costs
are added, normalised, averaged and then tabulated in table
1. It is clear that the performance of GIMPC2b is very close
to optimal, but perhaps, given the low order of the system,
this is not surprising.

<table>
<thead>
<tr>
<th>GIMPC2b</th>
<th>GIMPC2</th>
<th>OMPC ($n_c = 5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2949</td>
<td>1.1489</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE I
NORMALIZED AVERAGE COSTS

B. Three state-space example

Next, an illustration is given for a three state example:

$$A_1 = \begin{bmatrix} 0.97 & 0.15 & 0.18 \\ 0.07 & 0.597 & 0.001 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0.005 \\ -0.020 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} 1 & 0 & 0.25 \\ 0 & 1 & 0.20 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.005 \\ -0.022 \end{bmatrix}$$

$\pi = 1, \quad \bar{u} = -1, \quad \bar{z} = [-10, \ -10]^T \quad (26)$

Two robustly stabilising control laws, corresponding to an
LQR-optimal controller with $Q = \text{eye}(3), R_1 = 0.1, R_2 = 1,$
investigation as it may be a serious limiting factor in general. d.o.f interpolations [9]) and this perhaps merits further non-convexity in the feasible regions (earlier noted for one two-dimensional planes at the origin and table II contrasts the complexity of the associated inequalities. It is clear that once again GIMPC2b has given good feasibility, far far the complexity of the associated inequalities. It is clear that once again GIMPC2b has given good feasibility, far better than GIMPC. However, it is interesting to note the non-convexity in the feasible regions (earlier noted for one d.o.f. interpolations [9]) and this perhaps merits further investigation as it may be a serious limiting factor in general.

C. Four state-space example

The final example has 4 states and is given by:

\[ A_1 = \begin{bmatrix} -1.80 & 0 & 0 & 0.9 \\ 0.386 & -0.406 & 0 & 0.9 \\ 0 & 0 & -0.60 & 0 \\ 0.81 & -0.770 & 0.405 & -0.406 \\ -2.06 & 0 & 0 & 1.0 \\ 0.402 & -0.446 & 0 & 1.0 \\ 0 & 0 & -0.66 & 0 \\ 0.83 & -0.798 & 0.421 & -0.446 \end{bmatrix}, \]

\[ A_2 = \begin{bmatrix} 1.251 & 0 & 0 \\ 0 & 1.31 & 0 \\ 0 & 0 & 1.37 \end{bmatrix}, \]

\[ B_1 = \begin{bmatrix} 1.297 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \]

\[ B_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \]

Two robustly stabilising control laws are

\[ K_1 = \begin{bmatrix} -1.6174 & 0.3630 & -0.1936 & 0.5206 \end{bmatrix}, \]

\[ K_2 = \begin{bmatrix} -1.7761 & 0.3828 & -0.1952 & 0.6041 \end{bmatrix} \]

The figures here show a similar pattern to the three state example, that is a larger feasible region than GIMPC, despite using only 2 d.o.f whereas GIMPC has 5! (GIMPC has the largest feasible region, but uses 4 d.o.f.) Moreover, there is more evidence of non-convexity.

Table III summarises the comparison of the numbers of d.o.f utilised by each algorithm in four state-space system.

V. CONCLUSION AND FUTURE WORK

This paper develops the GIMPC2β algorithm and demonstrates it can easily be extended to the uncertain case. It is shown through examples that this interpolation method can give feasible regions quite close in volume to MPC techniques requiring far larger numbers of d.o.f., with little cost to performance, and thus if could be a good method to consider. This is particularly the case for higher order systems where earlier interpolations may not be efficient.

Nevertheless the paper exposes some issues that came to light in the early work on interpolation. Although combining closed-loop trajectories to capture the best characteristics with just a few parameters seems intuitively a good idea, it can also introduce some theoretical challenges: (i) standard stability proofs (if you consider these important) do not apply [9] and although other proofs can be used, they are not as generic and may require a posteriori analysis or an additional logic within the algorithm; (ii) moreover, perhaps of more interest to the theorist, is that the feasible regions do not appear to be always convex. A further investigation of this aspect constitutes ongoing work.

VI. ACKNOWLEDGMENTS

This principal investigator is funded by ORS, the University of Sheffield and the ACSE department.

REFERENCES

[3] Camacho, E. and C. Bordons, Model predictive Control, Springer Verlag
Fig. 3. Feasible region comparison of OMPC, GIMPC2 and GIMPC2 β for LPV system

Fig. 4. Feasible region comparison of OMPC, GIMPC2 and GIMPC2 β for uncertain system


APPENDIX

Constraint calculation for GIMPC2β in LPV case

Based on a similar the method used to construct state $X$ and constraints in the GIMPC2, define a new augmented state as:

$$\hat{x} = \begin{bmatrix} (1 - \alpha_i)x + \beta_i \omega \\ \alpha_i x - \beta_i \omega \end{bmatrix}$$  \hspace{1cm} (33)

Then substitution into an equivalent to (12) gives:

$$M_a \begin{bmatrix} (1 - \alpha_i)x + \beta_i \omega \\ \alpha_i x - \beta_i \omega \end{bmatrix} \leq d_a$$  \hspace{1cm} (34)

Where $M_a = [M_1, M_2]$, which clearly can be reduced to (14) as following, hence:

$$[M_{\alpha}, M_{\beta}] \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \leq d_{\alpha\beta}$$  \hspace{1cm} (35)

Clearly, $M_{\alpha} = [M_2 - M_1]x$, $M_{\beta} = [M_1 - M_2]\omega$, $d_{\alpha\beta} = d - M_1 x$.

Remark 6.1: Note $M_{\alpha}$, $M_{\beta}$ depend on $x$, $\omega$, so must be recomputed every sample.

Cost function for GIMPC2β in LPV case

In a similar manner to (6), the cost function can be represented as $\hat{x}^T P \hat{x}$. Then, substitution from (33) gives

$$J = \left[ (1 - \alpha_i)x + \beta_i \omega \right]^T P \left[ (1 - \alpha_i)x + \beta_i \omega \right]$$  \hspace{1cm} (36)

Expansion of this gives (15) as:

$$J = [\alpha_i \beta_i] S_{\alpha\beta} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} + [\alpha_i \beta_i] P_{\alpha\beta}$$  \hspace{1cm} (37)

where

$$S_{\alpha\beta} = \begin{bmatrix} x^T Y x & -x^T Y \omega \\ -x^T Y \omega & \omega^T Y \omega \end{bmatrix}, \quad P_{\alpha\beta} = \begin{bmatrix} -2x^T Y x - x^T Z x \\ 2x^T Y \omega + \omega^T Z x \end{bmatrix}$$

$$Y = ((P_{11} + P_{22} - P_{12} - P_{12}^T) + (P_{11} + P_{22} - P_{12} - P_{12}^T)^T)/2,$n

$$Z = 2P_{12} - 2P_{22}$$  \hspace{1cm} (38)

after $P$ has been split into $[P_{11} \quad P_{12}^T \\ P_{12} \quad P_{22}]$.

Remark 6.2: Again, notable that $S_{\alpha\beta}$, $P_{\alpha\beta}$ depend on $x$, $\omega$, so change each sample.