JMOSS Documentation: V4.0

Juan Jurado and Clark McGehee

25 May 2018

1 Introduction

This document describes the usage and syntax for the Jurado-McGehee Online Self Survey (JMOSS) V4.0 algorithm. Figure 1 illustrates the inputs, outputs, and overall flow of information throughout the algorithm. For further information on the JMOSS algorithm, please see:

Jurado, J.D., and McGehee, C.C., "A Complete Online Algorithm for Air Data System Calibration", AIAA Journal of Aircraft [DRAFT], March 2018.

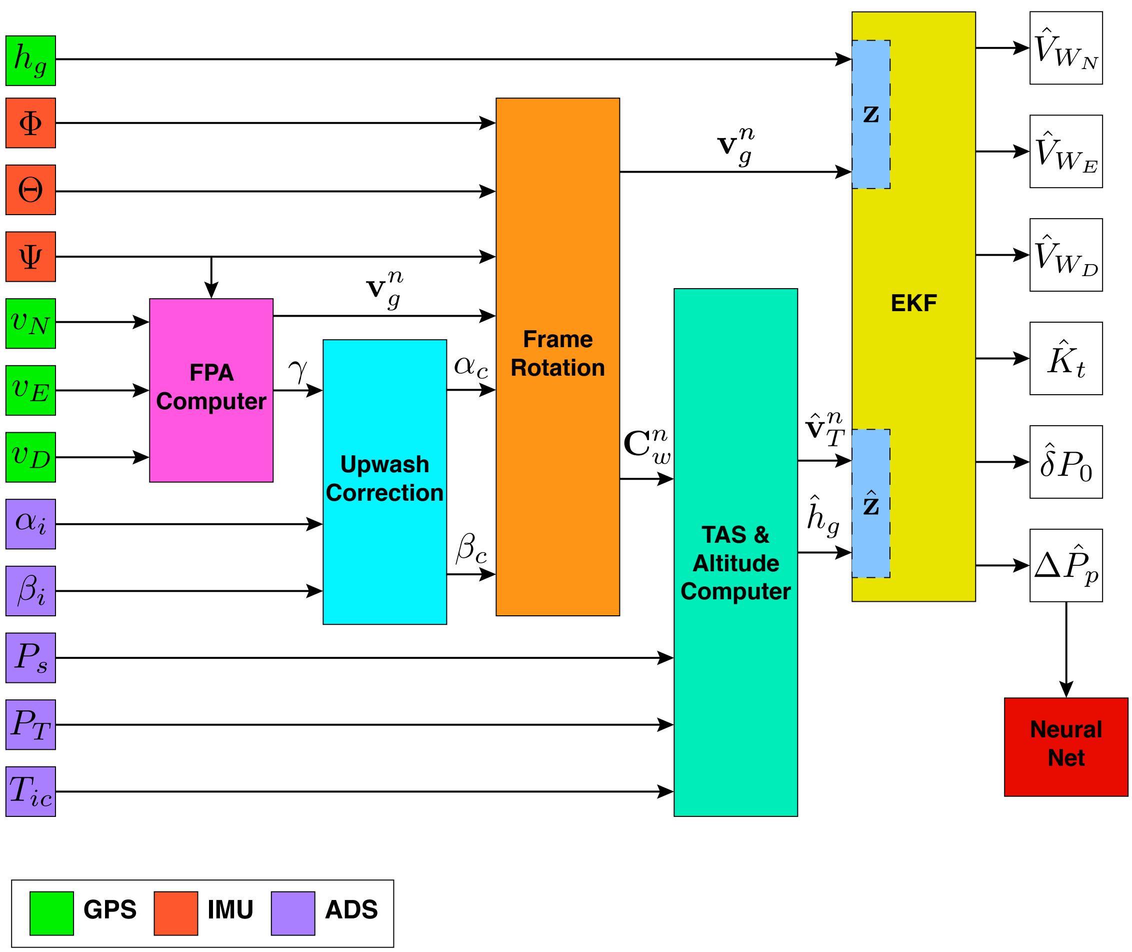

Figure 1: Overview of JMOSS Data Processing

2 Flight Technique

As a summary of the article referenced above, the flight technique needed to execute this algorithm is two phased:

1. A level acceleration or deceleration to cover the desired Mach domain.
2. A level, constant Mach, 360-degree turn somewhere practical within the desired Mach domain.

The provided sample data were collected using a level deceleration from 1.1 to 0.55 Mach, followed by a level turn at 0.65 Mach for Test Point 1 and 0.75 Mach for Test Point 2. Experimentally, it was found that adhering to the constant altitude tolerance of ±100 feet was critical to ensuring accurate results. Finally, remember to use a calibrated source of altitude control during test execution (e.g., GPS) since altimeter errors are expected in an uncalibrated Pitot-static system.

3 Syntax

3.1 Inputs

To run JMOSS V4.0, use the following syntax:

$$[\text{MODEL, EKF}] = \text{JMOSSV4(TestPoints)}$$

where the K-dimensional structure, TestPoints, contains K unique test points, each with N observations of the following required data:

TestPoints(k).Pt	-	Total pressure	[psi]
TestPoints(k).Ps	-	Static pressure	[psi]
TestPoints(k).Tic	-	Total temperature	[K]
TestPoints(k).alphai	-	Indicated angle of attack	[rad]
TestPoints(k).betai	-	Indicated angle of sideslip	[rad]
TestPoints(k).Vn	-	North GPS speed	[fps]
TestPoints(k).Ve	-	East GPS speed	[fps]
TestPoints(k).Vd	-	Down GPS speed	[fps]
TestPoints(k).hg	-	Total Pressure	[psi]
TestPoints(k).roll	-	Roll angle	[rad]
TestPoints(k).pitch	-	Pitch angle	[rad]
TestPoints(k).yaw	-	True heading angle	[rad]
TestPoints(k).time	-	Absolute or relative time vector	[s]

3.2 Outputs

The algorithm produces two objects: MODEL, and EKF. MODEL is a Gaussian Regression Process (GRP) object containing smooth M_{ic} and $\Delta P_p/P_s$ results as well as model statistics:

MODEL.mach	-	1000×1 smooth vector spanning observed M_{ic} domain
MODEL.dPp_Ps	-	1000×1 of corresponding $\Delta P_p/P_s$ results

MODEL.PredictionBand	-	1000×2 prediction band for $\Delta P_p/P_s$
MODEL.maxPB	-	Maximum full width of Prediction Band
MODEL.fullModel	-	Full MATLAB GRP object
MODEL.predictionFun	-	The final $\Delta P_p/P_s$ as a function of input M_{ic}. This function outputs
	-	the best estimate for $\Delta P_p/P_s$ for any given input M_{ic} number(s).

EKF is a structure containing various Extended Kalman Filter (EKF) time histories for the forward and backward pass, including estimates of 3D wind, variable temperature recovery factor, r, and pressure linearization point P_0:

EKF(k).mach	-	$N \times 1$ raw EKF observations of M_{ic} for k-th test point
EKF(k).machIC	-	$N \times 1$ raw EKF estimates of M_{pc} for k-th test point
EKF(k).dPp_Ps	-	$N \times 1$ raw EKF estimates of $\Delta P_p/P_s$ for k-th test point
EKF(k).fwdPass	-	$N \times 6$ array containing time histories of the 6 EKF states
	-	for the first (forward) pass on the k-th test point
EKF(k).bkdPass	-	$N \times 6$ array containing time histories of the 6 EKF states
	-	for the second (backward) pass on the k-th test point
EKF(k).deweightedPts	-	$N \times 1$ boolean indexing vector indicating which samples were
	-	included (true) or excluded (false) from the GRP smoothing
	-	process based on rollLimit for the k-th test point.
EKF(k).subSamp	-	$N \times 1$ vector containing the indices
	-	of the samples that were used in the EKF process
	-	after decimating the input data to remove repeated measurements.

3.3 Optional Argument Pairs

JMOSS V4.0 also supports an additional two input pairs:

'alpha'	-	Significance level for statistical inferences. Default is 0.05, which results in 95% inferences.
'rollLimit'	-	The Angle of Bank (AOB) limit for excluding turn data from $\Delta P_p/P_s$ modeling/smoothing.
	-	Default is 10 [deg] AOB.

4 Example MATLAB Usage and Output

```
% JMOSS V4 Demo File
% This script demonstrates how to use the basic features of the JMOSS Air
% Data Calibration algorithm.
%
% Written by Juan Jurado, Air Force Institute of Technology, 2018

clear; close all; clc;

%% Load sample data and feed to algorithm
% In this example, we will load two test points simultaneously to exercise
% the neural net training features with more than a single test point.
load sampleData.mat;
%#ok<*SAGROW>
for ii = 1:length(data)
    TestPoints(ii).Pt = data(ii).Pt; % [Psi]
    TestPoints(ii).Ps = data(ii).Ps; % [Psi]
    TestPoints(ii).Tic = data(ii).Tic; % [K]
    TestPoints(ii).alphai = data(ii).alphai; % [rad]
    TestPoints(ii).betai = data(ii).betai; % [rad]
    TestPoints(ii).Vn = data(ii).Vn; % [ft/s]
    TestPoints(ii).Ve= data(ii).Ve;% [ft/s]
    TestPoints(ii).Vd = data(ii).Vd; % [ft/s]
    TestPoints(ii).hg = data(ii).hg; % [ft MSL]
    TestPoints(ii).roll = data(ii).roll; % [rad]
    TestPoints(ii).pitch = data(ii).pitch; % [rad]
    TestPoints(ii).yaw = data(ii).yaw; % [rad]
    TestPoints(ii).time = data(ii).time; % [seconds]
end

%% Feed TestPoints to JMOSSV4
[MODEL,EKF] = JMOSSV4(TestPoints);
% Other example usage:
% [MODEL,EKF] = JMOSSV4(TestPoints(1)); % Only feed one of the TestPoints at a time
% [MODEL,EKF] = JMOSSV4(TestPoints,'alpha',0.1); % Specify a confidence level of 90% (default 95%)
% [MODEL,EKF] = JMOSSV4(TestPoints,'rollLimit',15); % Specify a AOB limit for excluding deltaP results (default is AOB>10 deg)

%% Process results for plotting
% Raw EKF results
```

```
ekfmach = cell2mat({EKF.mach}'); % Raw Extended Kalman Filter (EKF) mach vector (all test points)
ekfdPp_Ps = cell2mat({EKF.dPp_Ps}');% Raw EKF dPp_Ps vector (all test points)
turnPts = cell2mat({EKF.deweightedPts}');  % Index of excluded points due to AOB

% GRP Neural Net (Smoothed Output)
grpMach = MODEL.mach; % Smoothed GRP mach vector (all test points)
grpdPp_Ps = MODEL.dPp_Ps; % Smoothed GRP dPp_Ps curve (all test points)
pb = MODEL.PredictionBand; % 95% Prediction Band

%% Plot main results
% Display additional data from TestPoint 1.
TP = 1;
fontSize = 16;
WnHat = mean(EKF(TP).bkdPass(:,2)); % North wind [fps]
WeHat = mean(EKF(TP).bkdPass(:,3)); % East wind [fps]
WdHat = mean(EKF(TP).bkdPass(:,4)); % Down wind [fps]
rRange = EKF(TP).bkdPass([end 1],5); % Range of Temperature recovery factor values [unitless]

stateStrs = {sprintf('\\bf{Additional Parameters (Test Point %0.0f)}',TP)
             sprintf('$\\hat{V}_{W_n}$ = %0.5f ft/s',WnHat);
             sprintf('$\\hat{V}_{W_e}$ = %0.5f ft/s',WeHat);
             sprintf('$\\hat{V}_{W_d}$ = %0.5f ft/s',WdHat);
             sprintf('$\\hat{r} \\in [%0.3f,%0.3f]$',rRange)};

figure;
h(1) = plot(ekfmach(~turnPts),ekfdPp_Ps(~turnPts),'bo','MarkerFaceColor','b',...
    'MarkerSize',3); hold on;
h(2) = plot(ekfmach(turnPts),ekfdPp_Ps(turnPts),'go','MarkerFaceColor','g',...
    'MarkerSize',3);
h(3) = plot(grpMach,grpdPp_Ps,'r-','LineWidth',3);
h(4:5) = plot(grpMach,pb,'r--','LineWidth',2);
set(gca,'FontSize',fontSize,'FontName','helvetica');
t = text(0,0,stateStrs,'Interpreter','latex','EdgeColor','k',...
    'BackgroundColor',[1 1 1],'FontSize',fontSize,'Units','normalized');
set(t,'Position',[0.01,0.77,0])

xlabel('Instrument Corrected Mach, $M_{ic}$','Interpreter','latex'...
    ,'FontSize',fontSize);
```

```
ylabel('SPE, $\Delta P_p/P_s$','Interpreter','latex',...
    'FontSize',fontSize);
axis tight;
l = legend(h([1 2 3 4]),'Raw EKF Ouput','Excluded Turn Data','GRP','$95$\% Pred. Band',...
    'Location','NorthWest');
set(l,'Interpreter','latex','FontSize',fontSize);
grid minor;
set(gcf,'Position',[0 0 1.5*800 800]);
title('\textbf{Static Position Error, JMOSS V4 Demo}',...
    'Interpreter','latex','FontSize',fontSize+2)

%% AoA Effects
% We can look at each test point individually to see how they might differ
% from run to run. Be on the lookout for dPp_Ps vs. machCI changes due to
% chanes in w/delta, which can be diagnosed by plotting against AoA. In
% this plot, we'll go ahead and not plot the turn data so it doesn't clutter
% the plot. We can animate the 3D plot so get a good look at how AoA affects
% dPp_Ps vs. MachIC.
animate = true;
figure; hold on;
nPoints = length(data);
colors = jet(nPoints);
names = cell(nPoints,1);
for ii = 1:nPoints
    subSamp = EKF(ii).subSample; % Get the indices of decimated points
    c = colors(ii,:);
    mach = EKF(ii).mach;
    dPpPs = EKF(ii).dPp_Ps;
    alpha = TestPoints(ii).alphai(subSamp);
    turn = EKF(ii).deweightedPts;
    h1(ii) = plot3(mach(~turn),dPpPs(~turn),alpha(~turn),'o','Color',c,'MarkerFaceColor',c,'MarkerSize',3);
    names{ii} = sprintf('TestPoint %0.0f',ii);
end
xlabel('Instrument Corrected Mach, $M_{ic}$','Interpreter','latex'...
    ,'FontSize',fontSize);
ylabel('SPE, $\Delta P_p/P_s$','Interpreter','latex',...
    'FontSize',fontSize);
zlabel('AoA [rad]');
axis tight;
```

```
legend(h1,names)
grid minor;
set(gcf,'Position',[0 0 1.5*800 800]);
title('\textbf{Static Position Error, JMOSS V4 Demo}',...
    'Interpreter','latex','FontSize',fontSize+2)
view(3);
if animate
    M = 200;
    az = linspace(40,-20,M);
    el = 30;
    for ii = 1:M
        view(az(ii),el);
        drawnow;
    end
end
%% State Diagnostics
% Here we can look at the EKF state history on the first and second (final)
% pass. This can be used to look at EKF estimates of 3D wind, total temp
% recovery factor (r) and Pa linearization point (P0). We will focus on the
% first TestPoint just to see an example.
TP = 1;
stateTitles = {'$\Delta P_p/P_s$','$V_{W_N}$','$V_{W_E}$','$V_{W_D}$','$r$','$P_0$'};
N = length(stateTitles);
figure;
for ii = N:-1:1
    subplot(N,1,ii);
    plot(EKF(TP).mach,EKF(TP).fwdPass(:,ii),'b-','LineWidth',2); hold on;
    set(gca,'FontSize',fontSize,'FontName','helvetica');
    grid minor;
    axis tight;
    ylabel(stateTitles{ii},'Interpreter','latex','FontSize',fontSize);
    if ii == 6
        xlabel('Instrument Corrected Mach, $M_{ic}$',...
            'Interpreter','latex','FontSize',fontSize);
    end
end
titleStr = sprintf('\\bf{JMOSS EKF: Forward Pass (Test Point %0.0f)}',TP);
title(titleStr,'Interpreter','latex','FontSize',fontSize+2);
set(gcf,'Position',[0 0 800 1600]);

figure;
```

```
for ii = N:-1:1
    subplot(N,1,ii);
    plot(EKF(TP).mach,EKF(TP).bkdPass(:,ii),'b-','LineWidth',2); hold on;
    set(gca,'FontSize',fontSize,'FontName','helvetica');
    grid minor;
    axis tight;
    ylabel(stateTitles{ii},'Interpreter','latex','FontSize',fontSize);
    if ii == 6
        xlabel('Instrument Corrected Mach, $M_{ic}$',...
            'Interpreter','latex','FontSize',fontSize);
    end
end
titleStr = sprintf('\\bf{JMOSS EKF: Backward Pass (Test Point %0.0f)}',TP);
title(titleStr,'Interpreter','latex','FontSize',fontSize+2);
set(gcf,'Position',[0 0 800 1600]);
```

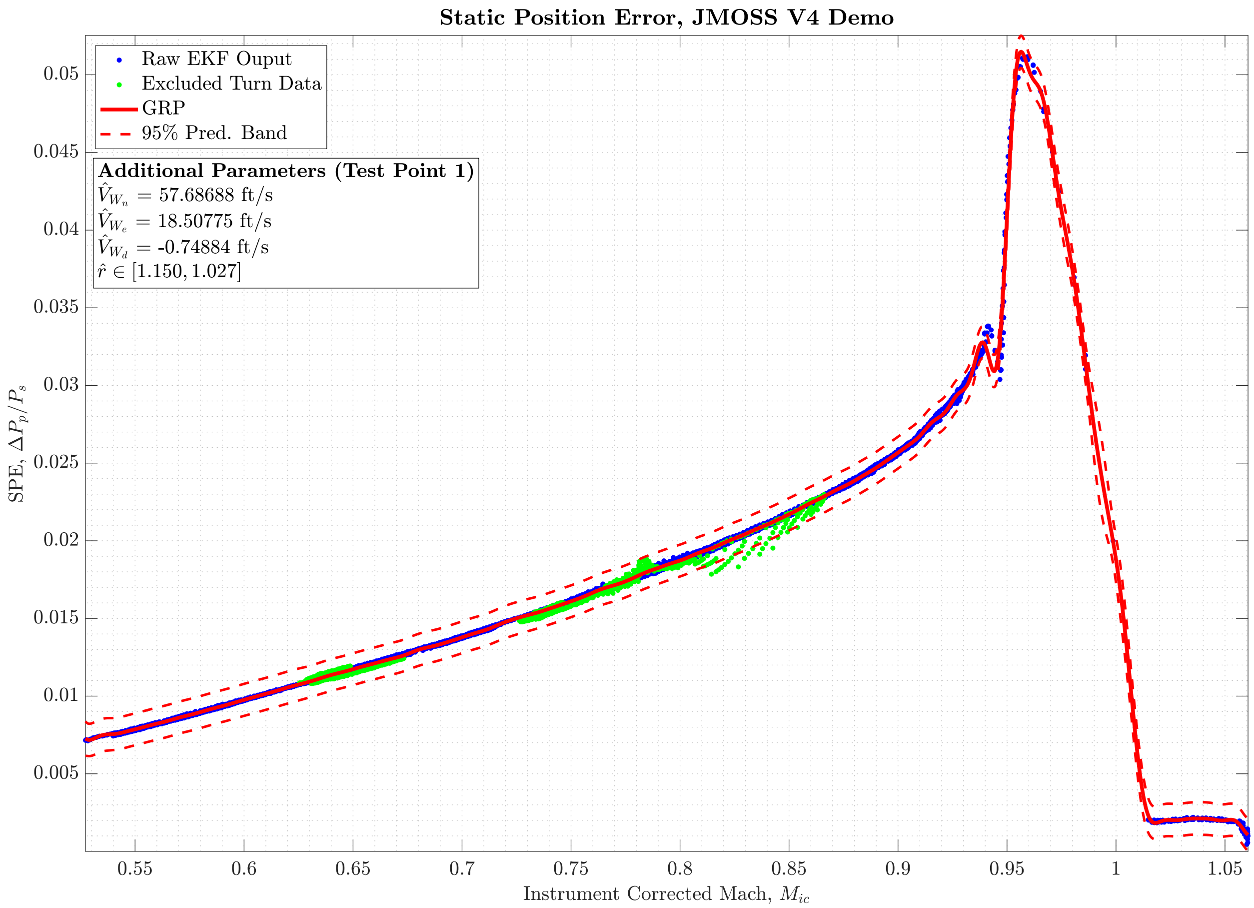

Figure 2: EKF and ASM Output

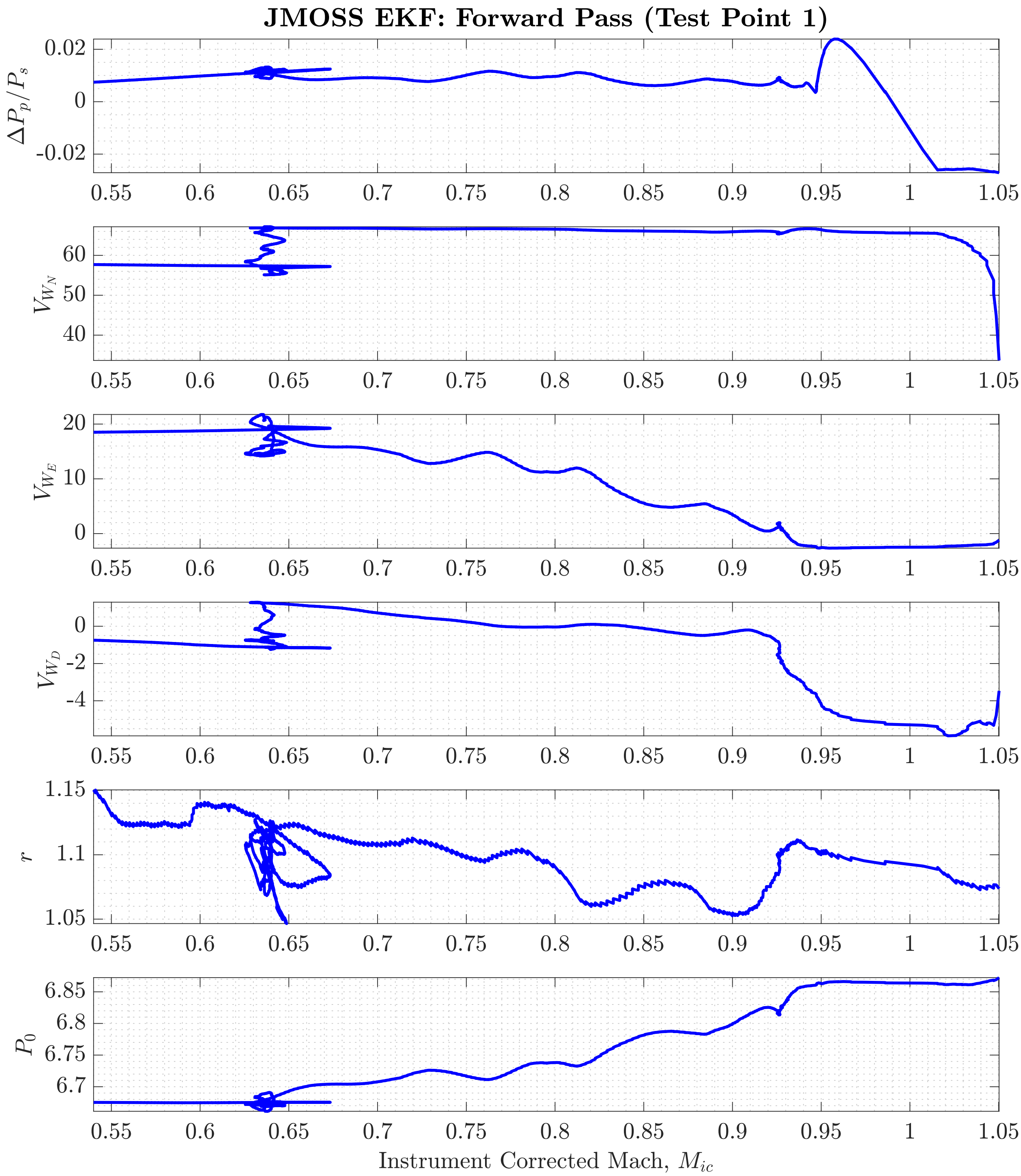

Figure 3: EKF Output on Forward Pass

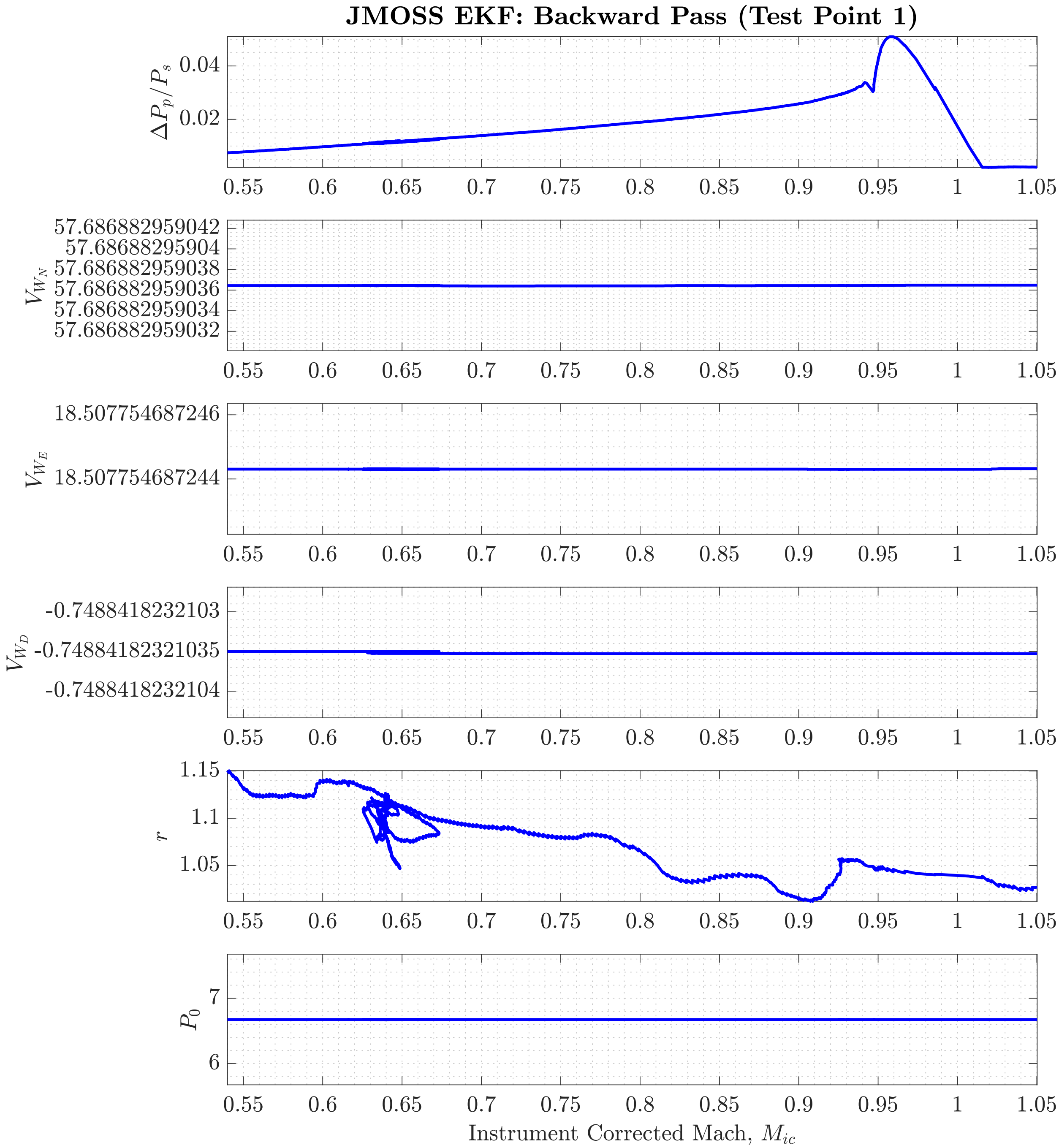

Figure 4: EKF Output on Backward Pass