JMOSS V3.1 Documentation

Juan Jurado and Clark McGehee

March 2018

1 Introduction

This document describes the usage and syntax for the Jurado-McGehee Online Self Survey (JMOSS) V3.1 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.

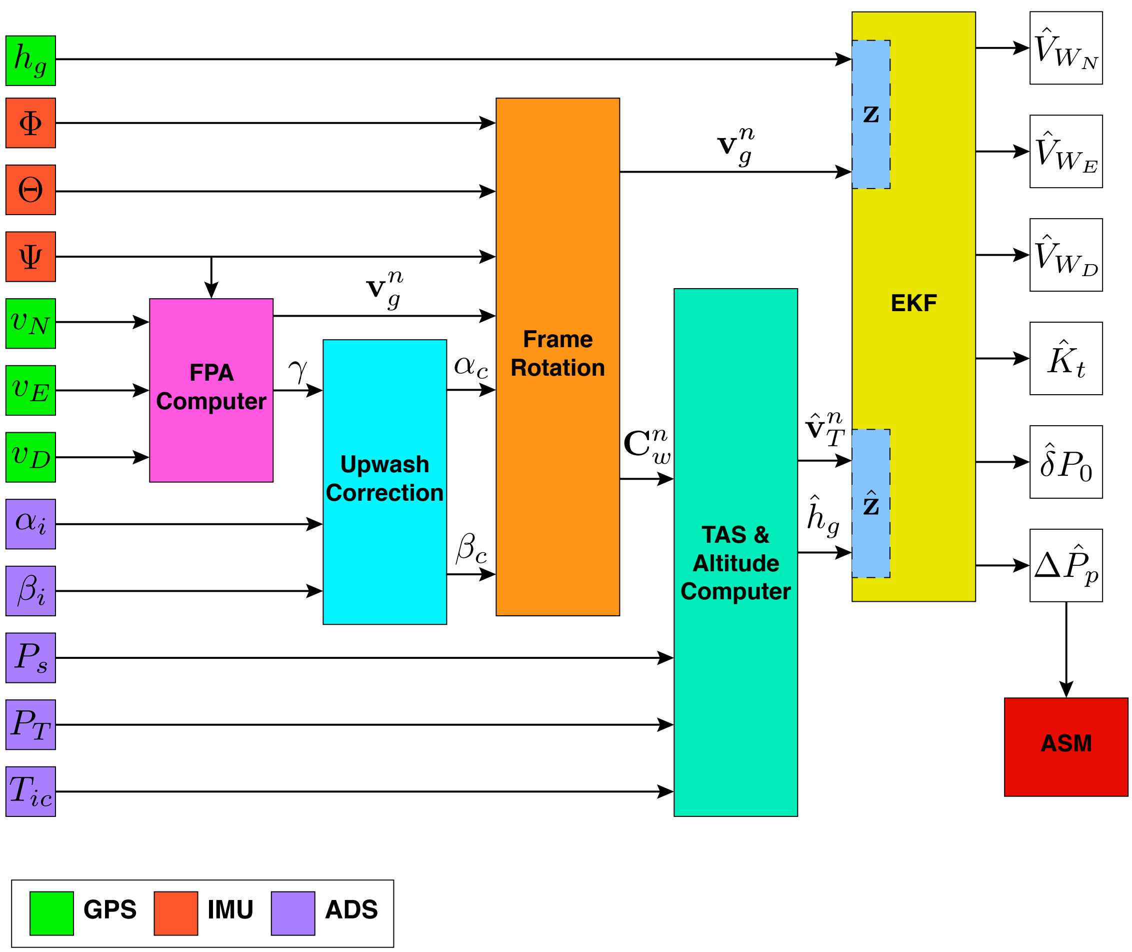

Figure 1: Overview of JMOSS Data Processing

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 was collected using a level deceleration from 1.1 to 0.55 Mach, followed by a level turn at 0.55 Mach. 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 V3.1, use the following syntax:

$$[\text{ASM, ekfResults}] = \text{JMOSSV31(TestPoints)}$$

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: ASM, and ekfResults. ASM is an Akaike Spline Model object containing smooth M_{ic} and $\Delta P_p/P_s$ results as well as model statistics:

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

ASM.PredictionBand	-	1000×2 prediction band for $\Delta P_p/P_s$
ASM.PredictionInterval	-	1000×2 prediction interval for $\Delta P_p/P_s$
ASM.ConfidenceBand	-	1000×2 confidence band for $\Delta P_p/P_s$
ASM.ConfidenceInterval	-	1000×2 confidence interval for $\Delta P_p/P_s$
ASM.maxPB	-	Maximum full width of Prediction Band
ASM.maxPI	-	Maximum full width of Prediction Interval
ASM.maxCB	-	Maximum full width of Confidence Band
ASM.maxCI	-	Maximum full width of Confidence Interval
ASM.splines	-	M_{ic} values of any spline knots used in smoothing
ASM.statModel	-	MATLAB statistical model object for ASM
ASM.PredictionFun	-	Function that produces $\Delta P_p/P_s$ estimates at desired M_{ic} values

ekfResults is a structure containing various Extended Kalman Filter (EKF) time histories for the forward and backward pass, including estimates of 3D wind, K_t, and non-standard pressure correction:

ekfResults(k).mach	-	$N \times 1$ raw EKF observations of M_{ic} for k-th test point
ekfResults(k).dPp_Ps	-	$N \times 1$ raw EKF estimates of $\Delta P_p/P_s$ for k-th test point
ekfResults(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
ekfResults(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

3.3 Optional Argument Pairs

JMOSS V3.1 also supports an additional four input pairs:

'alpha'	-	Significance level for statistical inferences. Default is 0.05, which results in 95% inferences.
'maxKnots'	-	Maximum number of ASM smoothing spline knots allowed. Default is 20.
	-	Supersonic data will have a minimum of 7.
'knots'	-	A $P \times 1$ vector containing specific knot locations defined by the user.
	-	This overrides the ASM automatic knot optimization algorithm and forces P knots
	-	at the specified M_{ic} values.
'freezeStates'	-	A true/false boolean indicating whether or not the estimates for 3D wind,
	-	K_t, and non-standard pressure correction should be held constant during the
	-	backward EKF pass. Default is `true`.

4 Example MATLAB Usage and Output

```matlab
% Example use of JMOSS V3.1 Pitot-static calibration algorithm
%
% Authors: Juan Jurado and Clark McGehee, Copyright 2018
clc; clear; close all;

%% Load Data and Construct Test Point Structure
% Note: You can load multiple test points into the JMOSS algorithm by
% adding to the TestPoint structure (i.e. TestPoint(1).Pt, TestPoint(2).Pt,
% etc...). The final smoothing will occur on the combined results from all
% test points loaded in this manner.

data = load('SampleData.mat'); % UNITS:
TestPoints(1).Pt = data.Pt; % [Psi]
TestPoints(1).Ps = data.Ps; % [Psi]
TestPoints(1).Tic = data.Tic; % [K]
TestPoints(1).alphai = data.alphai; % [rad]
TestPoints(1).betai = data.betai; % [rad]
TestPoints(1).Vn = data.Vn; % [ft/s]
TestPoints(1).Ve= data.Ve;% [ft/s]
TestPoints(1).Vd = data.Vd; % [ft/s]
TestPoints(1).hg = data.hg; % [ft MSL]
TestPoints(1).roll = data.roll; % [rad]
TestPoints(1).pitch = data.pitch; % [rad]
TestPoints(1).yaw = data.yaw; % [rad]
TestPoints(1).time = data.time; % [seconds]
%% Feed TestPoint structure to JMOSSV31
[ASM,EKFOutput] = JMOSSV31(TestPoints); % Uses all default settings
% Other example usage:
% [ASM,EKFOutput] = JMOSSV31(TestPoints,'alpha',0.1);
% [ASM,EKFOutput] = JMOSSV31(TestPoints,'freezeStates',false);
% [ASM,EKFOutput] = JMOSSV31(TestPoints,'knots',.9:.01:1.0,'alpha',0.01,'freezeStates',false);
% [ASM,EKFOutput] = JMOSSV31(TestPoints,'maxKnots',10);

%% Process results for plotting
ekfmach = cell2mat({EKFOutput.mach}'); % Raw EKF mach vector (all test points)
ekfdPp_Ps = cell2mat({EKFOutput.dPp_Ps}');% Raw EKF dPp_Ps vector (all test points)
asmmach = ASM.mach; % Smoothed ASM mach vector (all test points)
asmdPp_Ps = ASM.dPp_Ps; % Smoothed ASM dPp_Ps curve (all test points)
```

```matlab
pb = ASM.PredictionBand; % 95% Prediciton Band (i.e. Tolerance Interval)

%% Plot Results Including Wind and Kt estiamtes
fontSize = 16;
WnHat = mean(EKFOutput(1).bkdPass(:,2));
WeHat = mean(EKFOutput(1).bkdPass(:,3));
WdHat = mean(EKFOutput(1).bkdPass(:,4));
KtHat = mean(EKFOutput(1).bkdPass(:,5));

stateStrs = {'\bf{Additional Parameters:}';
             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('$\\hat{K}_t$ = %0.5f',KtHat)};

figure;
h(1) = plot(ekfmach,ekfdPp_Ps,'bo','MarkerFaceColor','b',...
    'MarkerSize',3); hold on;
h(2) = plot(asmmach,asmdPp_Ps,'r-','LineWidth',3);
h(3:4) = plot(asmmach,pb,'r--','LineWidth',2);

set(gca,'FontSize',fontSize,'FontName','helvetica');
text(0.55,0.04,stateStrs,'Interpreter','latex','EdgeColor','k',...
    'BackgroundColor',[1 1 1],'FontSize',fontSize);

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]),'Raw EKF Ouput','ASM','$95$\% 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 Algorithm Demo}',...
    'Interpreter','latex','FontSize',fontSize+2)

%% 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, Kt, and
% non-standard pressure correction.
stateTitles = {'$\Delta P_p/P_s$','$V_{W_N}$','$V_{W_E}$',...
    '$V_{W_D}$','$K_t$','$\delta P_0$'};
```

```
figure;
for ii = 6:-1:1
    subplot(6,1,ii);
    plot(EKFOutput(1).mach,EKFOutput(1).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

title('\bf{EKF: First Pass}','Interpreter','latex','FontSize',fontSize+2);
set(gcf,'Position',[0 0 800 1600]);

figure;
for ii = 6:-1:1
    subplot(6,1,ii);
    plot(EKFOutput(1).mach,EKFOutput(1).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

title('\bf{EKF: Second Pass}','Interpreter','latex','FontSize',fontSize+2);
set(gcf,'Position',[0 0 800 1600]);
```

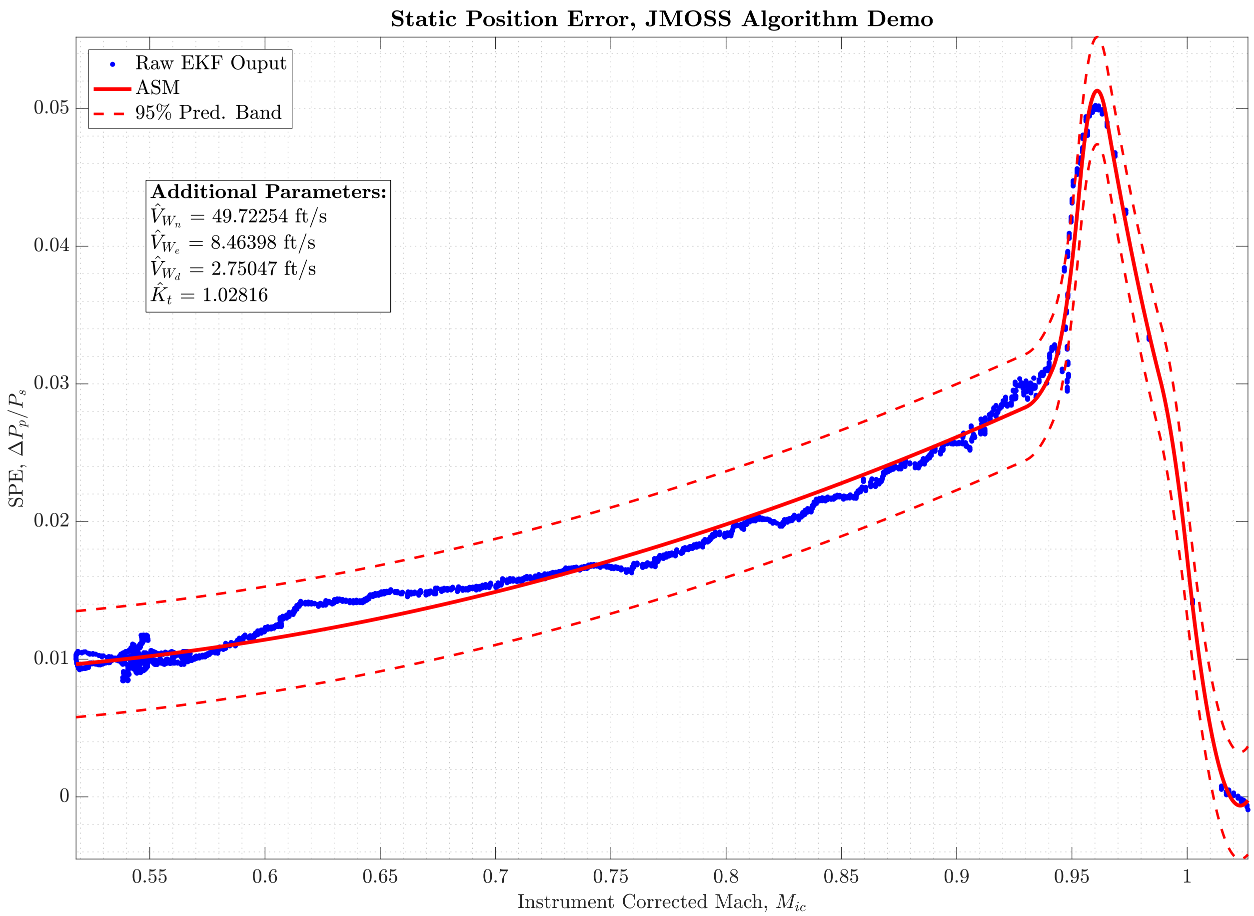

Figure 2: EKF and ASM Output

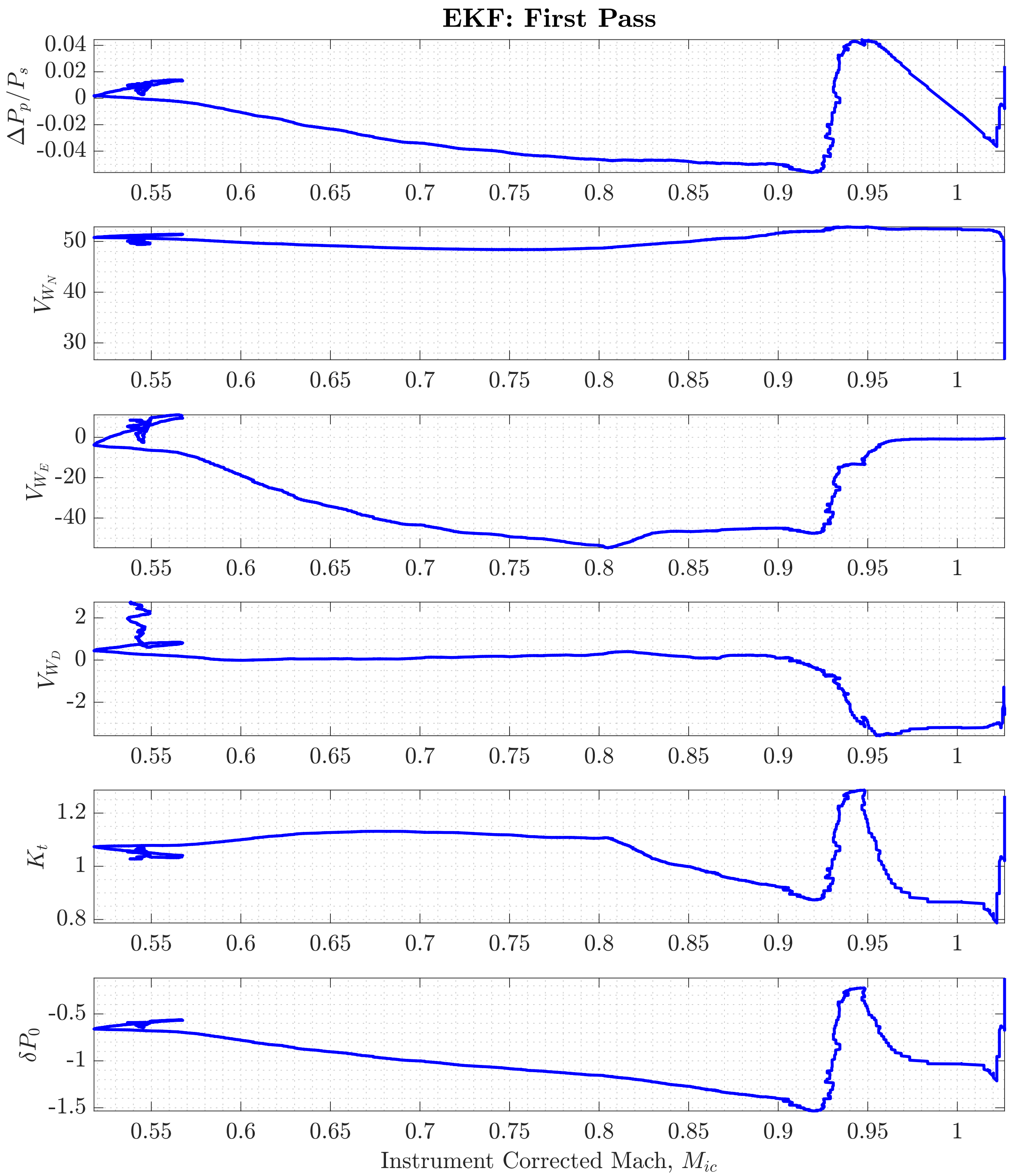

Figure 3: EKF Output on Forward Pass

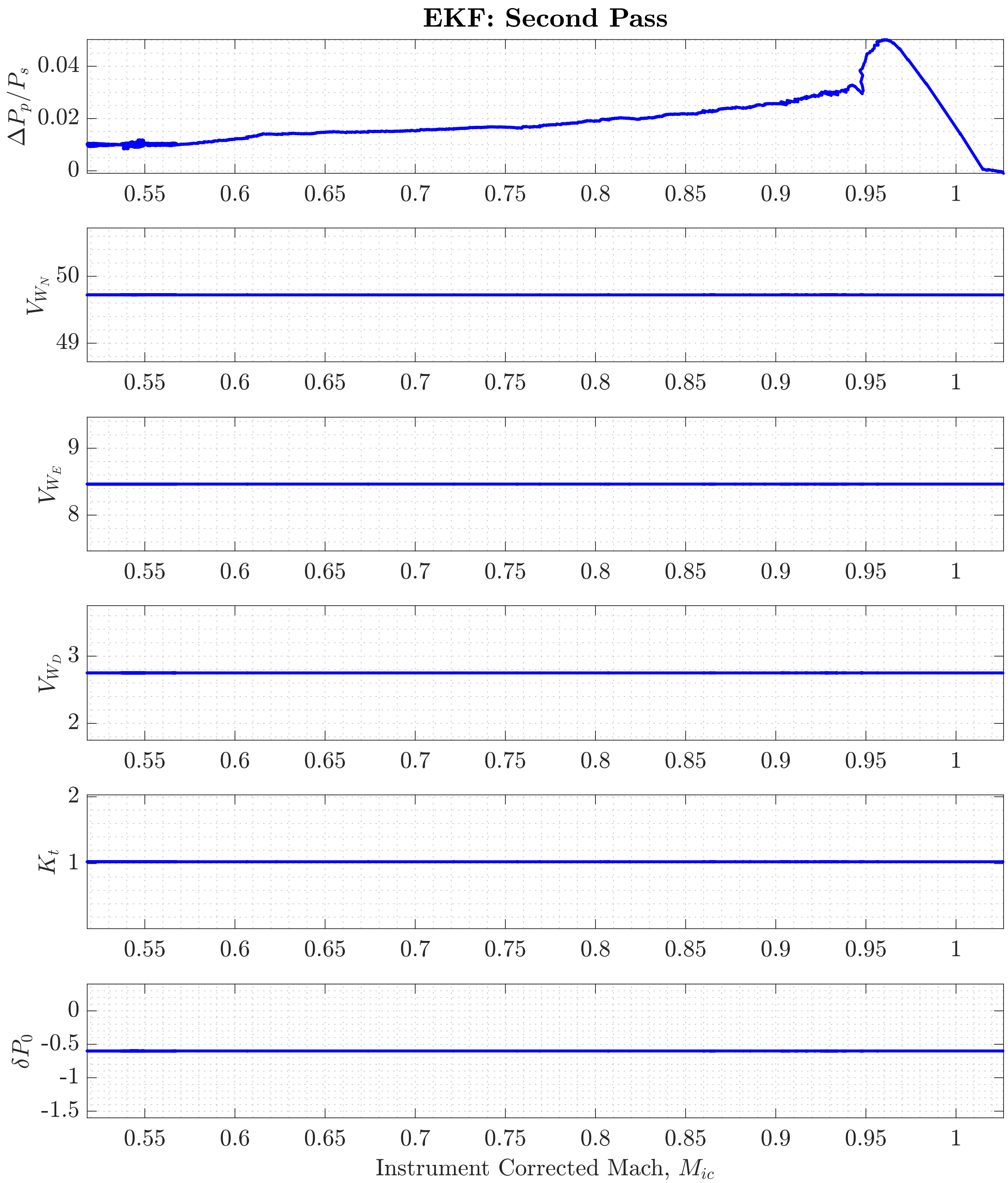

Figure 4: EKF Output on Backward Pass