Fixed Wing Aircraft Optimization: Real-Time Optimal Flight-Path Computation on an Android Tablet Using Reduced-Order Models

Sunil Deolalikar, Stanford University and Fazzel Gurrola, New Mexico State University
Mentor: David Amsallem, Charbel Farhat
research group

Abstract
Military aircraft flying at high velocities and low altitudes can encounter flutter, which can damage the aircraft. The flutter behavior of an aircraft depends on its structural properties and the speed and altitude it flies at. In practice, it restricts the flight conditions at which the aircraft can operate. The goal of this project is to include such flight restrictions in an optimal flight path computational setting and implement such a procedure on a portable device.

Introduction
Flutter is an unwanted aeroelastic phenomenon that can damage or even destroy an aircraft. Predicting flutter is therefore essential. Recent advances in parametric model reduction [1] have enabled the prediction of flutter in real-time in a database setting: computationally intensive calculations are done off-line for a few aircraft flight configurations and the resulting reduced-order models stored in a database. When flutter predictions are needed for other configurations online, a real-time framework is then capable of providing the required predictions [1].

This real-time framework is here applied to predict a flutter-free optimal path for an aircraft. In the present work, the aircraft considered is an F-22 Raptor in a configuration where two external fuel tanks are attached underneath its wings. It has recently been demonstrated that sloshing of fuel inside an aircraft fuel tanks can affect its flutter behavior [2]. As a result, the level of fuel inside the tanks will be taken into account in this work.

Dynamic programming is applied to compute a flutter-free optimal flight path for the F-22. The algorithm takes into account the aircraft flight envelope, its engine specifications and fuel consumption. An Android application is then implemented to demonstrate that the proposed computational framework can operate in real-time.

Aircraft Specifications

Engine Specification and Fuel Consumption
The specifications for the F119-PW-100 engine that equips the F-22 aircraft are available in the literature [3]. The thrust (T) and thrust specific fuel consumption (TSFC) data specifications can then be used to predict the instantaneous fuel consumption of the engine flying at a given speed and altitude using

$$\frac{dW_{fuel}}{dt} = -T \cdot TSFC$$  \hspace{1cm} (Equation 1)

Figure 1 reports the instantaneous fuel consumption behavior in function of aircraft speed and altitude. The data tabulated in [3] is here interpolated using piecewise-bilinear polynomials to obtain the missing data.

Fuel Tanks Configuration: The F-22 has internal fuel reservoirs with a capacity of 18000 lbs. In addition, in the present study, the aircraft is equipped with two external fuel tanks with a capacity of 4000 lbs. each.

Flight Envelope: The F-22 flight envelope is tabulated in [3]. Only feasible flight conditions where stall does not occur are considered in the present work.

Flutter Behavior: The flutter characteristics of the aircraft with fuel tanks are here modeled using the aeroelastic behavior of a representative wing system with a fuel tank attached to it. The fuel sloshing inside the tank is also modeled, resulting in flutter predictions that depend on the fuel level inside the tank as well as the aircraft speed and flight altitude. As previously indicated in the introduction, the reduced-order model framework developed in [1] is here applied to obtain...
flutter predictions for all fill levels and flight conditions. Figure 2 reports the minimal altitude the aircraft has to fly to avoid flutter in function of the Mach number and fuel fill level. As expected, altitude limitations only occur in the transonic regime and depend on the fuel level in the tank.

**Figure 2: Critical Flutter Altitude**

**Optimal Flight Path**

**Problem:** The goal is to minimize the flight time under the following three constraints: (1) the aircraft evolves in its flight envelope; (2) the aircraft does not run out of fuel; (3) the aircraft does not flutter. In addition, the initial flight condition and the final flight conditions (landing on an airfield) are prescribed.

The pilot can here act on two control variables: the speed of the aircraft and the altitude it flies at. The state of the system is then completely defined by adding the fuel fill level to those two variables. In the present work, the aircraft has a maximum speed of Mach 1.1 and maximum cruise altitude of 35,000 ft.

**Dynamic Programming:** The optimal flight path is computed in the present work using dynamic programming. The state space is discretized using 8835 points. The flight path is discretized using 21 points. Starting the from the initial state, the trajectories are evolved step by step and at a given time only the trajectory from the initial state to any point in the state space at the current time is considered. At the last step, only the state relative to the landing conditions (final flight condition) is considered. The whole optimal flight path is then recovered by stepping backward in time.

**Optimal Flight Example:** In Figure 3, a representative optimal flight path computed by the algorithm is reported. The initial flight conditions are here Mach: .8, Altitude: 25000 ft., and Distance to airfield: 100 mi. To minimize its flight time, the aircraft accelerates to reach its maximum speed and decreases its altitude before decelerating as it approaches its final destination. Interestingly, as the aircraft reaches the transonic regime during its approach, it gains altitude to avoid flutter.

**Creating an Android Program**

The initial implementation of the algorithm was done in MATLAB for development. Then it was ported to Java in order to create an Android mobile application for real-time predictions.

**User Interface:** The program written in Java performs the entire computation and is interactive so that the user can chose specific initial and final flight positions, as well as the initial aircraft flight speed and altitude. After the optimization is done, the user can observe the optimal safe flight trajectory for the aircraft as well as the corresponding flight time. A snapshot of the application interface is shown in Figure 4.

**Performance:** Table 1 reports the performance of the algorithm on three different platforms. It can see that the CPU time decreases dramatically when porting the algorithm from MATLAB to Java. However the Android app is then slower on the mobile device but is still capable to predict the optimal path in real-time.

<table>
<thead>
<tr>
<th>Platform</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>7</td>
</tr>
<tr>
<td>Java</td>
<td>1.5</td>
</tr>
<tr>
<td>Android</td>
<td>17</td>
</tr>
</tbody>
</table>

*Table 1: Computation times*
Further Developments
Algorithmic improvements can be done by including more realistic flight maneuvers for the aircraft and making the implementation even more efficient. Porting the tablet on board and enabling it to read live data such as tracking position using GPS may also be a future goal.

Real World Uses
The present work is the first of its kind to include CFD-based flutter predictions in an optimal flight path framework. Beyond the flutter problem, the current work shows that the future of scientific computing lies in the ability to deliver real-time predictions to users on the field. The future will likely see Army personnel deployed with advanced computational tool on mobile devices that will help them fulfill their mission. Reduced-order models provide an ideal tool to obtain accurate predictions in that context.

Conclusion
This work has demonstrated the possibility of implementing flutter constrained flight path optimizers on a mobile device. The methodology relies on the possibility of obtaining real-time flutter predictions using a database of reduced-order models.

References