# Towards Real-Time Aircraft Simulation with the MPI Motion Simulator

Marta Niccolini[*], Lorenzo Pollini[†] and Mario Innocenti[‡]
*University of Pisa, Pisa, Italy*

*and*

Paolo Robuffo Giordano[§], Harald J. Teufel[**] and Heinrich H. Bülthoff[††]
*Max Planck Institute for Biological Cybernetics, Tübingen, Germany*

**The paper describes the recent advancements gained on the MPI motion simulator project. The aim of this project is the use of an anthropomorphic robot as actuation system for a motion platform intended for real time flight simulation. Almost all commercially available motion platforms rely on the so called Stewart platform, that is a 6-DOF platform that can bear high payloads and can achieve high accelerations. On the other hand an anthropomorphic manipulator offers a larger range of motion and higher dexterity, that let envisage this novel motion simulator as a viable and superior alternative [1,2]. The paper addresses the use of a new inverse kinematics algorithm capable of keeping joint velocities and accelerations within their limits. Preliminary experimental results performed using the proposed algorithm along with possible further improvements are discussed.**

## I. Motion Platform and Washout Filters

The MPI Motion Simulator is based on the industrial robot Robocoaster, manufactured by KUKA Roboter GmbH, which has been modified for use as a real-time motion simulator [2]. The logical structure of the whole system is shown in Figure 1 where the main function blocks are highlighted.

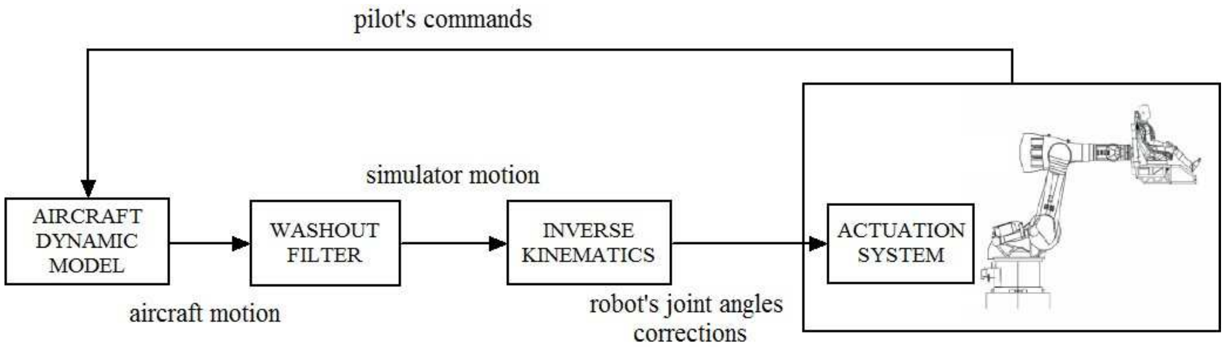

**Figure 1. Block structure of the Simulation and Motion system.**

The goal of the motion system, along with the visualization system, is to provide a virtual environment to the pilot so that he/she can experience cues similar to the ones of a real vehicle. Due to the limited workspace of the

[*] PhD student, Dept of Electrical Systems and Automation, Via Diotisalvi, 2, Pisa, Italy
[†] Assistant Professor, Dept of Electrical Systems and Automation, Via Diotisalvi, 2, Pisa, Italy, AIAA Member
[‡] Professor, Dept of Electrical Systems and Automation, Via Diotisalvi, 2, Pisa, Italy, AIAA Associated Fellow
[§] Research Scientist, Max Planck Institute for Biological Cybernetics, P.O. Box 2169, 72012 Tübingen, Germany.
[**] Research Assistant, Max Planck Institute for Biological Cybernetics, P.O. Box 2169, 72012 Tübingen, Germany.
[††] Director, Max Planck Institute for Biological Cybernetics, P.O. Box 2169, 72012 Tübingen Germany, Member AIAA.

platform, a one-to-one motion mapping between the trajectory of the simulated vehicle and the trajectory of the cabin cannot be obtained. It is thus of paramount importance to characterize and attempt to reproduce the motion information that are important for the task and that could affect pilot's behaviors. In order to ascertain and evaluate the motion cues that the platform can supply, the Classical Washout Filter (CWF) motion cueing algorithm is employed in this preliminary implementation: Figure 2. The motion cues that are considered as output of the aircraft dynamic model are the linear specific forces, $f_B$, and the angular rates, $\omega_B$. The platform attempts to reproduce a specific force vector and an angular velocity vector at the pilot's location in the simulator that approximate the stimulus that the pilot would experience in an actual aircraft.

The Classical Washout Filter makes use of the tilt coordination mechanism which exploits the gravity vector to recover for low-frequency specific forces that, due to the limited workspace of the platform, have to be washed out. The filter can be seen as composed of two connected channels: the high frequency components of the specific forces and angular rates are reproduced by direct motion of the platform, while the low frequencies components are mainly reproduced by tilting the cabin on the platform and by the visualization system [4].

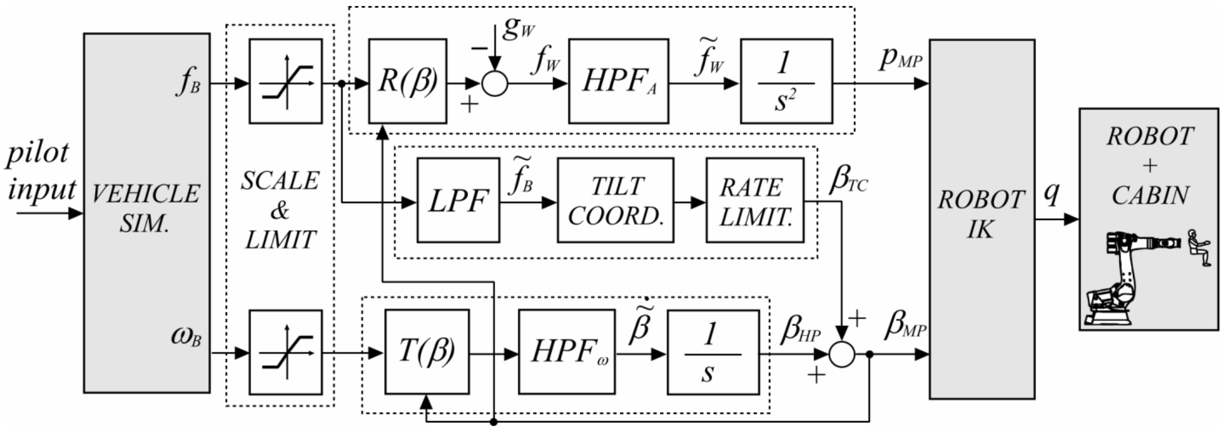

**Figure 2. The Classical Washout Filter including the Simulator and the Platform IK.**

Since the outputs of the washout filter are the desired displacement and the desired attitude of the cabin, this can be seen as a 6-DOF trajectory generator for the cabin actuation system. The problem of reproducing linear specific forces and angular velocities is then turned into the problem of real-time trajectory following.

## A. The Platform Inverse Kinematics Problem

Differently from a Stewart platform, the inverse kinematics problem (IK), that is finding the needed joint angles of the anthropomorphic robot given the posture (position $p_{mp}$ and attitude $\beta_{mp}$) of the cabin, may present some problems. Past work has shown that it is possible to use an exact IK algorithm to generate robot joint angles [1]. The algorithm described in the above mentioned article does not take into consideration the presence of constraints to maximum joint velocities and accelerations that may be imposed by a robot control system. The MPI motion platform is built over the Robocoaster system which is equipped with a robot controller that guarantees a range of safety countermeasures that allow a human operator to ride the robot. For safety reasons it is not possible to use the Robocoaster without its controller and safety rules. These safety rules include constraints to maximum joint velocities and accelerations that guarantee that the controller is always able to perform a safety stop using accelerations which are tolerable by a human onboard, and by the robot structure itself.

By this motivation, any IK algorithm to be used on the MPI motion platform must deal with the velocity/acceleration constraints of the controlled robot, which has been studied by the authors in [2], and, in particular produce joint trajectories that respect the given constraints.

As anticipated, analytic solutions of the IK problem are available only for simple manipulation structures and suffer from not being able to handle joint rate and acceleration limits, and lack robustness to singularities. The differential kinematics, which maps joint rates to the velocity of the cabin in the Cartesian space, can be used to iteratively solve for the inverse kinematics problem. The inverse differential kinematics was first introduced in [7] under the name of resolved rate control. This technique derives the joint velocities that result in the desired motion trajectory by inversion of the Jacobian matrix, then integrate them to compute the desired joint angles. Therefore the reconstruction of the joint angles by numerical integration leads to drift phenomena of the solution; as a consequence, the achieved end-effector position and orientation might differ from the desired ones. A feedback correction term was introduced to recover for the errors in the Cartesian space and to avoid numerical drift instabilities ([8],[9]). For the goal of this project, a classical first order Closed Loop Inverse Kinematic (CLIK) [6] algorithm was considered preliminarily. Figure 3 highlights this concept: the reference signal $x_d$ contains both position and attitude references for the robot end-effector (generated by the washout filters), the integrator between $\dot{q}$ and $q$ represents an ideal robot (which integrates joint velocity commands into joint angles), $k(\cdot)$ represents the robot forward kinematics (which transforms joint angles into robot end-effector positions) $K$ is a gain matrix (which weights position and attitude errors) and generates, together with the velocity feed-forward $\dot{x}_d$, a Cartesian velocity reference signal $v$ which is translated into desired joint velocities by the matrix $J^{-1}(q)$. The approach clearly requires the inversion of the Jacobian matrix $J(q)$, which may not be possible or numerically tractable at and near kinematics singularities [1]. Close to kinematics singularities the Jacobian (pseudo) inverse kinematics algorithm becomes ill-conditioned and results in very high joint velocities and control deviations ([10], [11]). Based on this framework, alternative solutions to avoid the matrix inversion problem have been proposed in literature, such as the Jacobian transpose CLIK algorithm ([8]).

Another of these alternative solutions is the Damped Least Square (DLS) approach that exploits the power of CLIK for real-time IK solution, and is capable to deal with joint velocity limits [3]. This paper presents a modified Damped Least Square resolved rate control (mDLS), that is capable to handle both joint rate and acceleration limits.

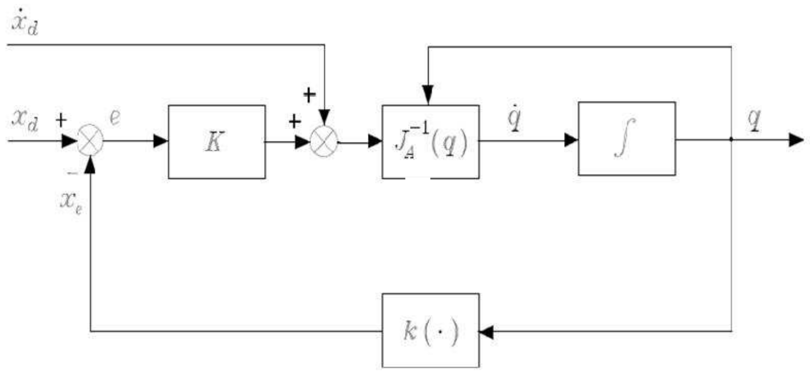

**Figure 3. The Closed Loop Inverse Kinematics.**

### B. A Modified DLS Algorithm for Joint Rates and Acceleration Constraints

The goal of the family of DLS algorithms is to minimize the Jacobian matrix inversion error trading off with limited joint velocities: if the Jacobian matrix can be exactly inverted, then the norm $\|v - J(q)\dot{q}\|^2 = 0$; this is not possible near to singularities, thus the DLS algorithm searches for the joint rates $\dot{q}$ that minimize the Jacobian matrix inversion error $\|v - J(q)\dot{q}\|^2$ and keep $\|\dot{q}\|^2$ as low as possible. The original DLS algorithm ([3], [12]) derives the joint rates $\dot{q}$ from the solution of the optimization problem:

$$\begin{cases} \min\limits_{\dot{q}} \left( \|v - J(q)\dot{q}\|^2 + \lambda^2 \|\dot{q}\|^2 \right) \\ \qquad v = Ke + \dot{x}_d \end{cases}$$

where the parameter λ is used to trade off between accuracy and feasibility of the joint velocity required to generate the given end-effector velocity ν. The algorithm can be seen as a position control scheme which tries to bring the error $e(t) = x_d(t) - x_e(t)$ to zero, where $x_d(t)$ is the desired cabin position and attitude, as generated by the washout filter, and $x_e(t)$ is the actual cabin posture; finally $\dot{x}_d(t)$ is a velocity feed-forward signal generated by the washout filter as well. The problem described above has a closed form solution:

$$\dot{q} = \left( J^T(q)J(q) + \lambda^2 I \right)^{-1} J^T(q)v$$

where the parameter λ is generally varied as a function of the minimum and maximum singular values of the Jacobian matrix.

In order to handle joint acceleration limits as well, the optimization problem of the original DLS algorithm was modified using a joint acceleration term in place of the joint velocity/rate term. Furthermore, in order to proceed to a real-time implementation, the modified DLS (mDLS) optimization problem is formulated directly in discrete-time:

$$\dot{q}(t_k) \cong \frac{q(t_k) - q(t_{k-1})}{t_k - t_{k-1}} = \arg\min \left( \left\| v(t_k) - J\dot{q}(t_k) \right\|^2 + \lambda^2 \left\| \ddot{q}(t_k) \right\|^2 \right)$$

By using a first order approximation of the joint acceleration:

$$\ddot{q}(t_k) \cong \frac{\dot{q}(t_k) - \dot{q}(t_{k-1})}{t_k - t_{k-1}} = \frac{\dot{q}(t_k) - \dot{q}(t_{k-1})}{\Delta t}$$

the optimization problem can be rewritten as:

$$\dot{q}(t_k) = \arg\min \left( \frac{1}{2} \left\| v(t_k) - J\dot{q}(t_k) \right\|^2 + \frac{1}{2} \bar{\lambda}^2 \left\| \dot{q}(t_k) - \dot{q}(t_{k-1}) \right\|^2 \right)$$

where the parameter $\bar{\lambda}$ trades off between smaller Jacobian inversion errors and larger joint accelerations. Note that, at each time step, the term $\dot{q}(t_{k-1})$, that is the joint velocity vector at the previous time step, is known and comes from the solution of the inverse kinematics problem at the previous time step.

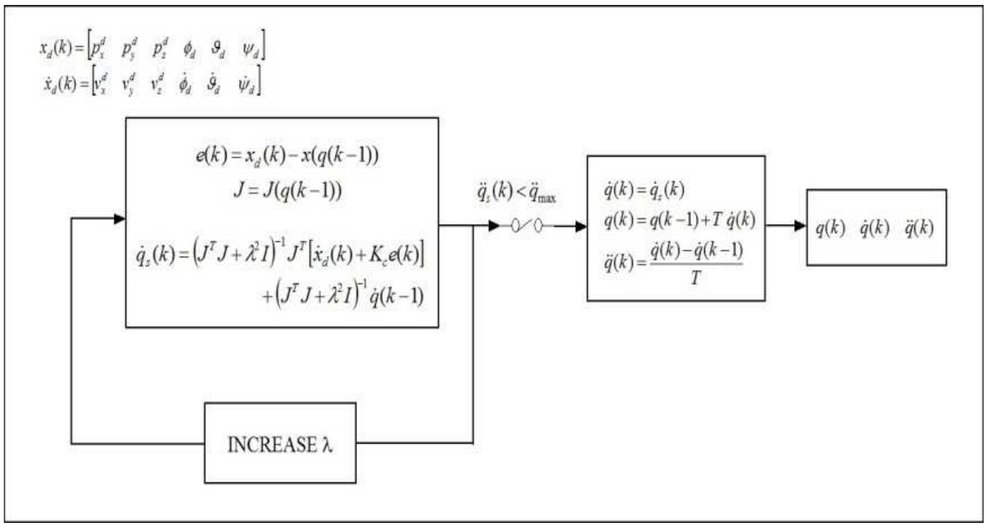
$x_d(k) = [p_x^d \;\; p_y^d \;\; p_z^d \;\; \phi_d \;\; \vartheta_d \;\; \psi_d]$
$\dot{x}_d(k) = [v_x^d \;\; v_y^d \;\; v_z^d \;\; \dot{\phi}_d \;\; \dot{\vartheta}_d \;\; \dot{\psi}_d]$
$e(k) = x_d(k) - x(q(k-1))$
$J = J(q(k-1))$
$\dot{q}_s(k) = (J^T J + \lambda^2 I)^{-1} J^T [\dot{x}_d(k) + K_c e(k)] + (J^T J + \lambda^2 I)^{-1} \dot{q}(k-1)$
$\ddot{q}_s(k) < \ddot{q}_{\max}$
$\dot{q}(k) = \dot{q}_s(k)$
$q(k) = q(k-1) + T\dot{q}(k)$
$\ddot{q}(k) = \frac{\dot{q}(k) - \dot{q}(k-1)}{T}$
$q(k) \;\; \dot{q}(k) \;\; \ddot{q}(k)$
INCREASE λ

**Figure 4. The mDLS algorithm flow chart.**

The mDLS problem has a closed form solution as well:

$$\dot{q}(t_k) = \left( J^T J + \bar{\lambda}^2 I \right)^{-1} J^T v(t_k) + \bar{\lambda}^2 \left( J^T J + \bar{\lambda}^2 I \right)^{-1} \dot{q}(t_{k-1})$$

It can be easily understood that a constant weighting factor $\bar{\lambda}$ may not work well for all situations: when exact IK is possible (i.e. far from kinematics singularities) it would be best to relax completely the joint acceleration constraint (this is possible using $\bar{\lambda} = 0$), while when Jacobian inversion becomes badly conditioned $\bar{\lambda}$ should have larger values. Adaptively changing $\bar{\lambda}$ according to distance from singularities appears a viable solution. In particular an iterative adaptation algorithm was developed which tries to keep $\bar{\lambda} = 0$ and increases $\bar{\lambda}$ when necessary only, i.e. when an acceleration limit is hit by any of the joints, until the new mDLS solution $\dot{q}(t_k)$ guarantees that all the components of the joint acceleration vector $\ddot{q}(t_k)$ are inside their respective limits. $\bar{\lambda}$ is thus adapted at each time step to guarantee that all joints respect their limits. This algorithm is capable of limiting joint rates as well by bringing accelerations to zero (by increasing $\bar{\lambda}$) until all joint rates respect their limits. Figure 1 shows a flowchart of the IK algorithm. A formal proof of the stability of this algorithm is under development and is out of the scope of this paper. Its capabilities have been ascertained during simulations and online tests.

## II. Simulations and Tests

Experimental tests have been performed with a nonlinear dynamic model of a two-seater light-weight aircraft. A joystick was used to control the aircraft inside a three-dimensional virtual environment. The dynamic model of the aircraft, the washout filter and the inverse kinematics algorithm were implemented using Simulink, while the visualization system was based on DynaWorlds [5]; RealTimeWorkshop is used to implement the IK algorithms in real-time. Figure 5 shows the MPI Motion Simulator inside the Cyberneum at the Max Plank Institute in Tubingen (Germany) during a simulation and a snapshot of the display screen as seen by the pilot onboard the cabin. The out of the window view is reproduced using a panoramic screen and a distortion corrected video projector. An Inertial Mesaurement System (INS) composed by a triad of accelerometers and gyroscopes was placed on the end-effector of the robot in order to record linear accelerations and angular rates actually produced on the cabin.

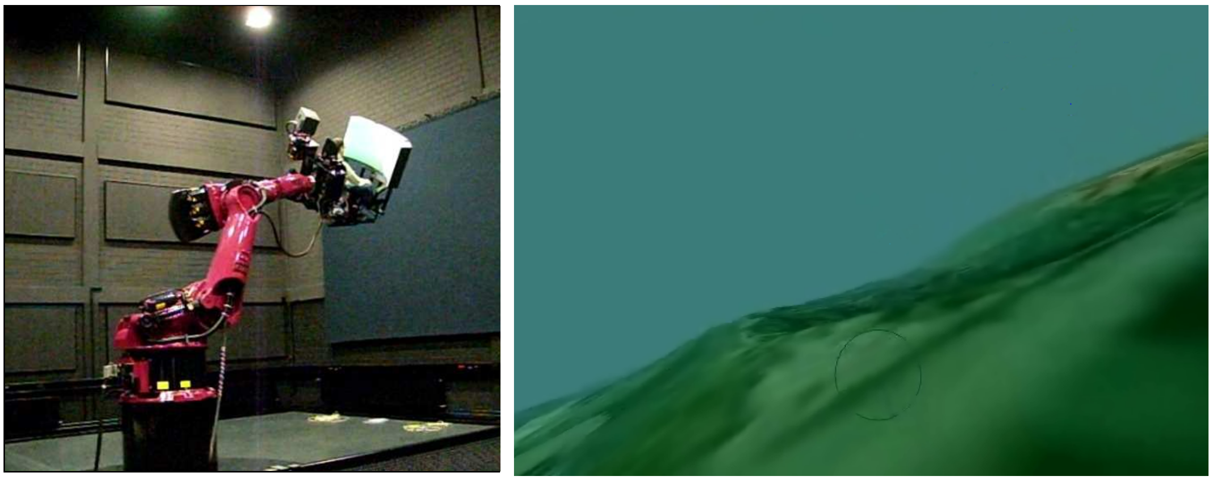
**Figure 5. The MPI Motion Simulator and a snapshot of the graphical environment.**

A complete analysis of the system would require a comparison between the accelerations produced by the aircraft simulator and the accelerations measured on the end-effector, that is of the complete simulator/washout/Inverse Kinematics/Actuator Dynamics/Platform Kinematics chain. Moreover, the evaluation of how well the CWF can be adapted or modified for the robot kinematics is out of the scope of this paper and will be part of future work. In order to evaluate the inverse kinematics algorithm only, the accelerations and the angular rates filtered by the washout and the ones produced and recorded on the end-effector will be compared. Several experimental tests were performed; the following sections present and discuss three of them.

### A. Flight 1

Figures 6, 7 and 8 show the results of a simulation run obtained with the described setup. Figure 6 shows the trajectory errors of the cabin with respect to the desired trajectory computed using the washout filter. These errors, which remain limited to few millimeters, and less than 1 degree in attitude, are mainly due to the saturation on the accelerations of the joints of the spherical wrist.

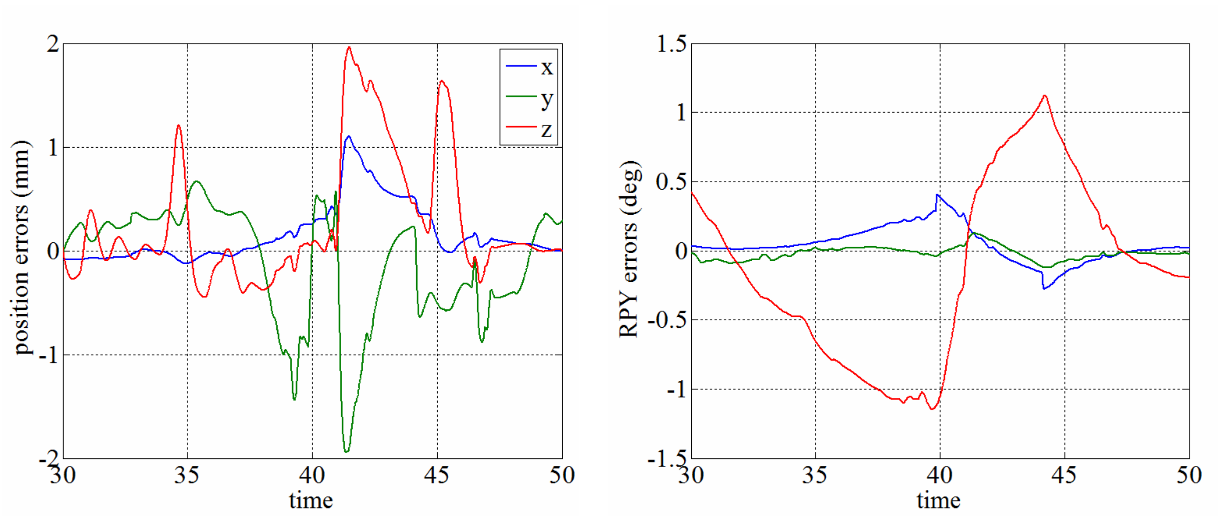

**Figure 6. Position and Attitude errors .**

Figure 2 shows the acceleration and angular velocity errors, i.e. the difference between the desired accelerations and angular velocities filtered by the washout and the ones computed from the trajectory of the end-effector.

Figure 3 shows, in the three columns, joint angles, rates and accelerations respectively. Saturation of joint accelerations of two joints of the spherical wrist (q4 and q6) can be noticed around time 40s and 45s, the same sample time at which large errors appear in the yaw rates (see Figure 7).

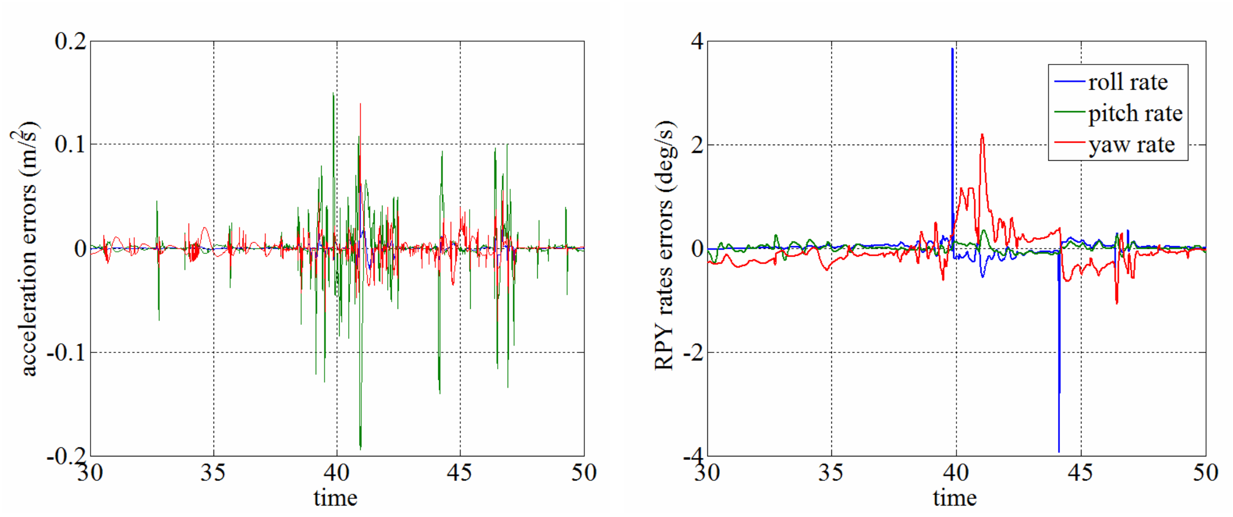

**Figure 7. Acceleration and Angular rate errors .**

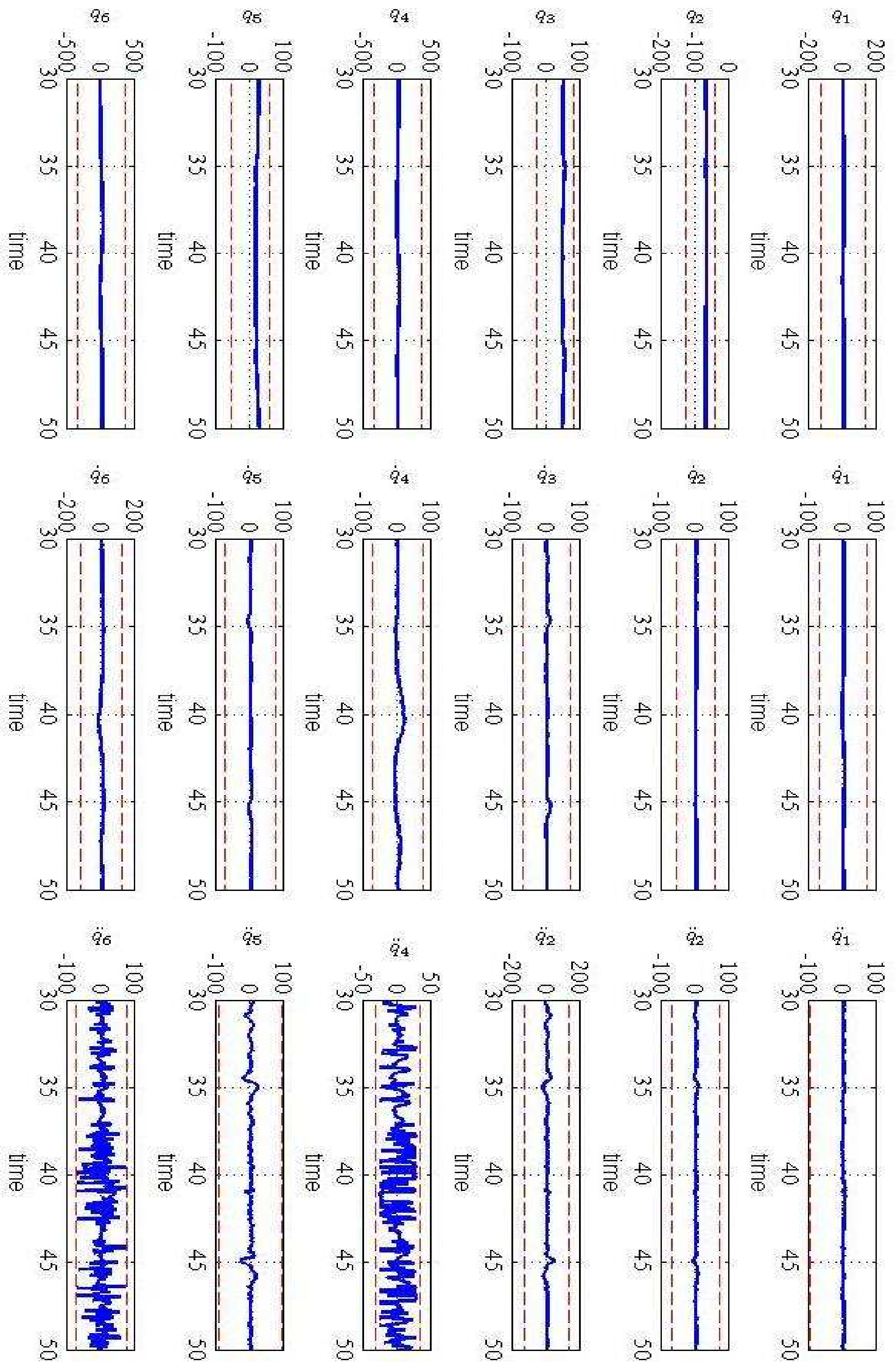

**Figure 8. Joint angles, velocities and accelerations.**

## B. Flight 2

Results from this second flight, which was performed with maneuvers intended to excite strongly the IK, are used to show the effect of the weighting parameter $\lambda$ of the mDLS algorithm. Figure 9 compares the desired cabin accelerations (as produced by the washout filters) and the actual cabin acceleration produced through the IK on the cabin. A portion of the flight was selected where the robot works very close to its wrist singularity and the mDLS does a hard work to keep joint accelerations inside their limits. Several acceleration artifacts can be noted and they are all in correspondence to the sample times at which $\lambda \neq 0$. It is currently believed that the heights of these spikes could be reduced by enhancing the algorithm which varies $\lambda$ in order to produce smoother variations and to reduce the quantum of $\lambda$ adaptation.

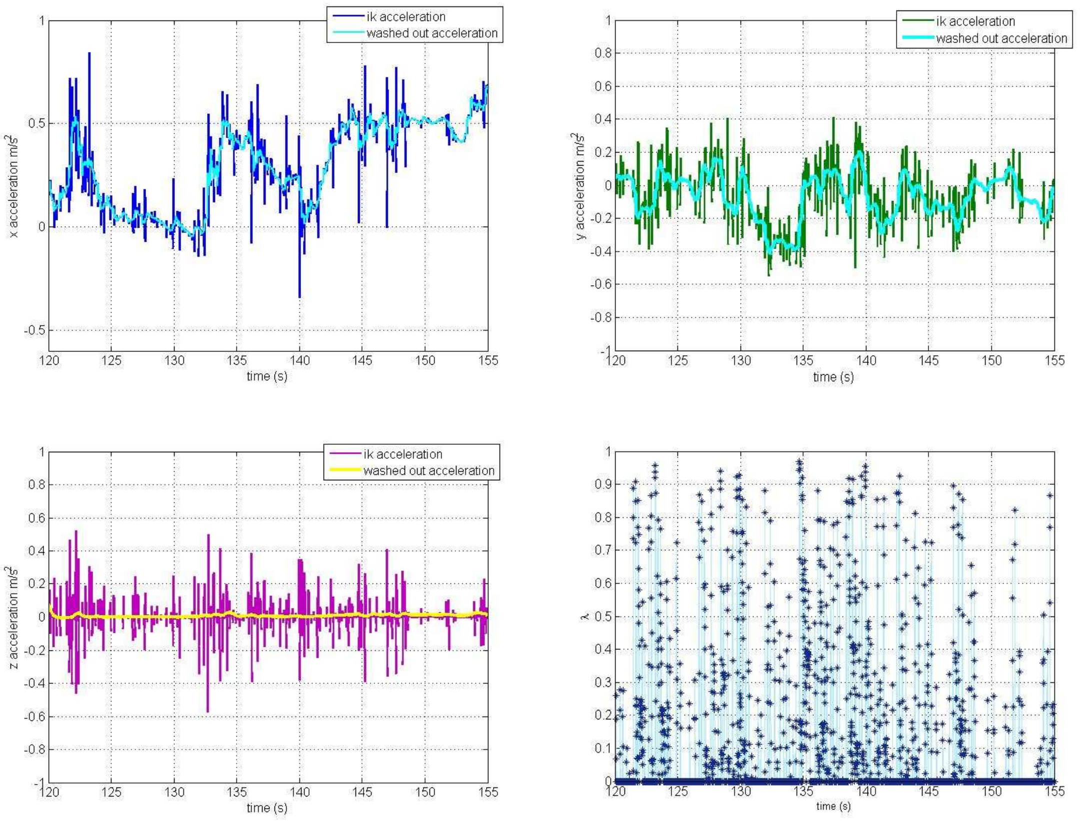

**Figure 9. Desired vs Commanded cabin accelerations and effect of the $\lambda$ parameter.**

## C. Flight 3

Results from the third  flight, are used to compare the desired accelerations and angular velocities filtered by the washout and the ones actually recorded by the INS on the cabin: Figure 10. Since accelerations and angular rates were recorded outside of the system where the washout filter and the IK algorithm were run, delays cannot be evaluated. The data was aligned in correspondence of the time at which the robot started to move so that desired vs measured data comparison is still possible. Oscillations on the accelerometer and Gyroscope data can be mainly addressed to the saturation of joint accelerations, to measurement noise, and to a non perfectly stiff fixing of the accelerometer on the robot structure.

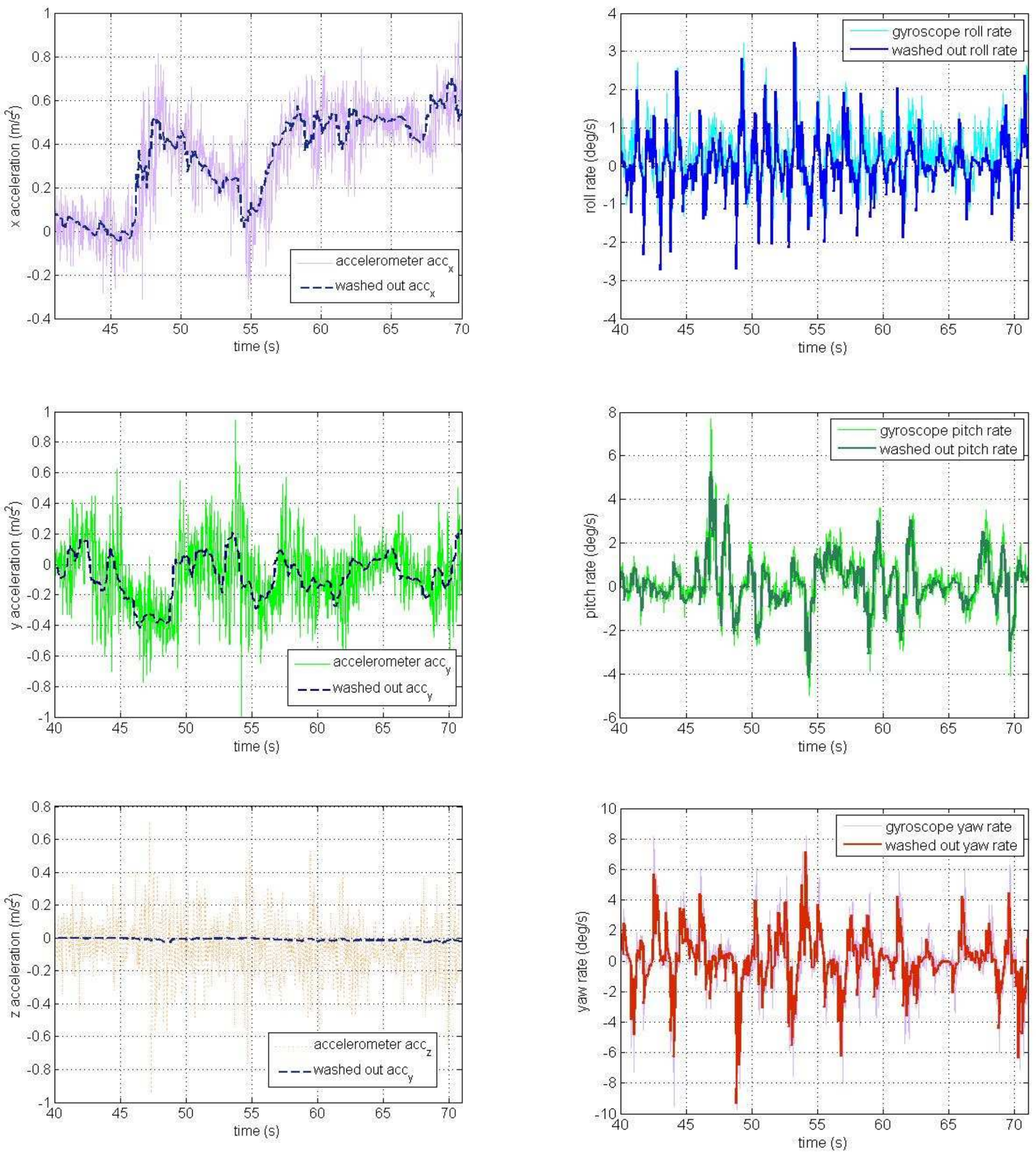

**Figure 10. Desired vs Measured cabin accelerations and angular rates.**

## III. Conclusions and Future Work

The paper presented the recent achievements obtained toward real-time flight simulation with the MPI Motion Simulator. A modified Damped Least Square Inverse Kinematics algorithm was developed in order to generate joint trajectories that always satisfy the robot joint velocity and acceleration limits. The complete system was tested using a cabin with a panoramic screen projector, a joystick and a tri-dimensional scenery generator. Experimental results have shown the feasibility of the approach but further testing and developments are needed: in particular the tests highlighted the presence of unwanted accelerations and angular velocities artifacts due to operation in the vicinity of robot singularities and consequent activation of the mDLS algorithm. In order to asses the amount of cueing distortion induced by such artifacts, it is worth using some kind of mathematical model of the pilot perception system, and to compare the desired specific forces and desired angular velocities, as outputted by the aircraft dynamic model, and the ones really produced by the motion platform. It is a critical point for the cueing algorithm to avoid any improper motion cue since it is commonly known that false cues have a negative effects on the simulation. Indeed those negative motion cues could break the illusions introduced by the visual system and so they should always be avoided whenever possible. Another issue that needs attention is the exploitation of the complete robot motion envelope. Enlargement the range of motion requires the development of a new class of washout filters; very likely, given the cylindrical symmetry of the robot workspace, a filter that works on cylindrical coordinates rather than Cartesian as the CWF does, might enlarge the span of the generated trajectories; nevertheless this approach could lead to unwanted cross-axis effects that need particular study and attention.

## IV. References

[1] L. Pollini, M. Innocenti,A. Petrone, “Novel Motion Platform for Flight Simulators Using an Anthropomorphic Robot,” Journal of Aerospace Computing, Information, and Communication, Vol. 5, July 2008.

[2] Teufel, H. J., H.-G. Nusseck, K. A. Beykirch, J. S. Butler, M. Kerger, and H. H. Bülthoff, “MPI Motion Simulator: Development and Analysis of a Novel Motion Simulator,” *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2007.

[3] C.W. Wampler, "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Method", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, No. 1, January/February 1986, p93-101.

[4] F.M. Cardullo, R.J. Telban, “Motion Cueing Algorithm Development: Human-Centered Linear and Nonlinear Approaches,” NASA Report, *State University of New York, Binghamton, New York.*

[5] L. Pollini, M. Innocenti, “A Synthetic Environment for Dynamic Systems Control and Distributed Simulation,” IEEE Control System Magazine, Vol. 20, No. 2, pp. 49-61, 2000.

[6] L. Siciliano, L. Sciavicco, L. Villani, G. Oriolo “Robotics,” Springer, 2008.

[7] D.E. Whitney, “Resolved motion rate control of manipulators and human prostheses,” IEE Transaction on Man-Machine Systems, vol. 10, pp.47-53, 1969.

[8] L. Sciavicco, B. Siciliano “Coordinate transformation: A solution algorithm for one class of robots,” IEEE Trans. on Syst., Man, Cyber., vol. 16, pp. 550-559, 1986.

[9] C.W. Wampler, L.J. Leifer, “Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators,” J. Dyn. Syst., Meas., and Control, vol. 110, pp. 31-38, 1988.

[10] C.A. Klein, C.H. Huang, “Review of pseudoinverse control for use with kinematically redundand manipulators,” IEEE Trans. on Syst., Man., Cyber., vol. 13, pp. 245-250, 1983.

[11] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation,” IEEE J. of Robot. Automat., vol. 4, pp. 43-53, 1988.

[12] Y. Nakamura, H. Hanafusa, “Inverse kinematic solutions with singularity robustness for robot manipulator control,” ASME Journal of Dynamic Systems, Measurement and Control, vol. 108, pp. 163-171, 1986.