Proceeding of the 2nd
RSI/ISM International Conference on Robotics and Mechatronics
October 15-17, 2014, Tehran, Iran

Implementation of Analytic Iterative Redundancy Resolution Technique on KNTU Cable Robot

Fereshteh Sabahi, Mohammad A. Khosravi and Hamid D. Taghirad *Senior Member, IEEE*
Advanced Robotics and Automated Systems (ARAS), K.N. Toosi University of Technology, Tehran, Iran.
Email: {Fereshte_sabahi@ee, makh@ee, taghirad@}kntu.ac.ir

***Abstract*—Analytic Iterative Redundancy Resolution (*AIRR*) is a semi-analytic method for redundancy resolution in cable-driven manipulators. As all previous redundancy resolution methods were based on numerical algorithms, they impose an uncertainty to execution time which is barely acceptable in real-time implementation. In this paper, *AIRR* is implemented as a fast solution to redundancy resolution problem by checking a set of analytic solutions instead of using numerical algorithms. Furthermore, the performance of this method is compared to previous numerical method implemented on KNTU robot with respect to execution time and accuracy. It is shown that the real-time performance of this implementation in closed-loop control structure is at least fifteen times faster than that of previously implemented methods. Such decrease in execution time in real-time implementation is very promising for future applications.**

I. Introduction

PARALLEL cable robots are a class of parallel manipulators whose moving platform is driven by cables. As the cables can be ultimately retracted, cable-driven robots may be applied in very large workspace [1], [2]. Large workspace and other important potential features of this kind of robot such as high speed and acceleration and high load to weight ratio make them a popular alternative in many applications [3], [4]. Since, cables can apply only tensile forces, actuator redundancy becomes a necessity in fully-constrained cable-driven robots [5], [6]. Although redundancy is mostly a desirable feature in robot manipulators, it leads to complexity in the forward kinematics and the robot control scheme [7].

Redundancy resolution techniques have been extensively studied during past four decades. Most of these approaches use pseudo-inverse of the Jacobian matrix of the manipulator and Jacobian null space concept. In some recent works, the redundancy resolution of a planar cable robot has been studied at kinematics and dynamics levels in order to minimize the norm of actuator forces, and the norm of mobile platform velocity, while considering positive tension in all the cables [8].

It is important to note that the numerical methods are the only way to find the optimal solution in implementation of all previous redundancy resolution techniques [9]. Due to the stop condition in numerical algorithms which depends to final solution precision, these methods are usually computationally expensive, and in contrast to the analytic methods, the solution time is variable and is not bounded to a certain limit. This can be an important drawback in real-time implementation of such methods in closed-loop control applications. These facts can clarify the importance of analytic methods versus numerical ones in redundancy resolution schemes. Analytic Iterative Redundancy resolution (*AIRR*) is an analytic and iterative method proposed in 2011 to solve redundancy resolution problem in cable-driven robots [10]. This technique can efficiently limit the required amount of time to find optimal solution to redundancy resolution problem using a semi-analytic approach based on *Karush-Kuhn-Tucker* (KKT) [11] method of optimization. In [10], this method is applied only in simulations and no real-time implementation of *AIRR* has been reported in the literature.

In this paper, first a brief review on *AIRR* technique is given and then this technique is evaluated by two steps. In the first step, it is compared to the fastest numerical optimization algorithm available in MATLAB environment (interior-point method [12]) and the previously implemented method (*CFSQP* [13], [14], [8]) through simulation. It is shown that the proposed method works as accurate as numerical algorithms. Then, *AIRR* and *CFSQP* are simultaneously implemented on KNTU cable-driven robot using RT-LAB as the real-time implementation software. It is shown that, the *AIRR* is considerably faster than the previously implemented method.

II. A review on analytic iterative redundancy resolution technique

A. Background Theory

Jacobian analysis is an important part in redundancy resolution study. Actually Jacobian is the matrix of all first-order partial derivatives of a vector valued function. This matrix describes the relation between length variable velocities ($\dot{\boldsymbol{L}}$) and moving platform velocities ($\dot{\boldsymbol{X}}$), as well as the relation between actuator forces ($\boldsymbol{\tau}$) and forces acting on the moving platform ($\boldsymbol{F}$) of a parallel robot [15]:

$$\dot{\boldsymbol{L}}_{n\times 1} = \boldsymbol{J}_{n\times m}\dot{\boldsymbol{X}}_{m\times 1} \tag{1}$$

$$\boldsymbol{F}_{m\times 1} = \boldsymbol{J}^T\boldsymbol{\tau}_{n\times 1} \tag{2}$$

Where n is the number of actuators and m is robot degrees of freedom (DOF).
Regarding the fact that all the cables must always remain under tension; $\boldsymbol{\tau}$ in Eq. (2) shall be larger than a positive constant $\boldsymbol{\tau}_{min}$.

$$\begin{aligned} \boldsymbol{F}_{m\times 1} &= \boldsymbol{J}^T\boldsymbol{\tau}_{n\times 1} \\ \boldsymbol{\tau} &\geq \boldsymbol{\tau}_{min} \end{aligned} \tag{3}$$

As it mentioned earlier, to satisfy the tensionability condition in a fully-constrained cable robot the number of active cables must be more than robot DOFs. Denoting actuator redundancy, Jacobian matrix is non-square and therefore, the problem stated in (3) leads to many solutions. The minimum norm solution of this problem is driven through pseudo-inverse:

$$\boldsymbol{\tau}_0 = \boldsymbol{J}^{T^\dagger} \boldsymbol{F} \tag{4}$$

However, it is notable that this solution does not guarantee positive tension in the cables. To generally guarantee the minimum threshold considered for $\boldsymbol{\tau}$; the problem may be redefined as a constrained optimization problem under the equality constraints of $\boldsymbol{F} = \boldsymbol{J}^T\boldsymbol{\tau}$ and the inequality constraints denoted by $\boldsymbol{\tau} \geq \boldsymbol{\tau}_{min}$. It is shown in [10], that the general solution to such constrained optimization problem may be obtained by $\boldsymbol{\tau} = \boldsymbol{\tau}_0 + \boldsymbol{A}\boldsymbol{y}$, in which $\boldsymbol{A}$ denotes the null-space of $\boldsymbol{J}^T$. As further detailed in [10], the function $\varepsilon(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is defined as (5) in order to drive optimum solution of redundancy resolution using KKT theorem [11] with $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_m]^T$ and $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_n]^T$ as Lagrangeian and KKT multipliers, respectively, for equality and inequality constraints defined as $g(\boldsymbol{y}) = 0$ and $r(\boldsymbol{y}) \geq 0$.

$$\begin{gathered}
\varepsilon(\boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{y}) + \boldsymbol{\lambda}^T g(\boldsymbol{y}) + \boldsymbol{\mu}^T r(\boldsymbol{y}) \\
f(\boldsymbol{y}) = \| \boldsymbol{\tau}_{n\times 1} \|_2 = \boldsymbol{\tau}^T\boldsymbol{\tau} = (\boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y})^T(\boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y}) \\
r(\boldsymbol{y}) = \boldsymbol{\tau}_{min} - \boldsymbol{\tau} = \boldsymbol{\tau}_{min} - (\boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y}) \leq 0 \\
g(\boldsymbol{y}) = \boldsymbol{F} - \boldsymbol{J}^T(\boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y})
\end{gathered} \tag{5}$$

Considering KKT theorem conditions, the optimal solution is within the following three cases:

- Case 1: All the forces lie on the boundary of inequality constraints.
- Case 2: All the forces are inside the boundary of inequality constraints.
- Case 3: Some of the forces are on the boundary of inequality constraints and the others are inside them.

In case 1, $\forall i \in [1, n]\ r_i(\boldsymbol{y}_0) = 0$ and the solution is directly driven through $\boldsymbol{\tau}_{n\times 1} = [\tau_{min_1}, \tau_{min_2}, \cdots, \tau_{min_n}]^T$.

In case 2, considering $\boldsymbol{\mu} = 0$, the optimization problem is simplified to :

$$\begin{gathered}
r = \text{nullity of } (\boldsymbol{J}^T) \\
2\boldsymbol{A}^T\boldsymbol{\tau}_0 + 2\boldsymbol{y}_0 - \boldsymbol{A}^T\boldsymbol{J}\boldsymbol{\lambda}_0 = 0 \\
g(\boldsymbol{y}) = \boldsymbol{F} - \boldsymbol{J}^T(\boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y})
\end{gathered} \tag{6}$$

Eq. (6) is rewritten in a matrix form:

$$\boldsymbol{B}_0 \cdot \boldsymbol{X} = \boldsymbol{C}_0 \qquad \boldsymbol{X} = \boldsymbol{B}_0{}^{-1} \cdot \boldsymbol{C}_0$$

$$\begin{gathered}
\boldsymbol{B}_{0_{(m+r)\times(m+r)}} = \begin{bmatrix} 2\boldsymbol{I}_{r\times r} & -(\boldsymbol{A}^T\boldsymbol{J})_{r\times m} \\ (\boldsymbol{J}^T\boldsymbol{A})_{m\times r} & (\underline{0})_{m\times m} \end{bmatrix} \\
\boldsymbol{X}_{(m+r)\times 1} = \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{\lambda}_0 \end{bmatrix} \quad \boldsymbol{C}_0 = \begin{bmatrix} -2\boldsymbol{A}^T\boldsymbol{\tau}_0 \\ \boldsymbol{F} - \boldsymbol{J}^T\boldsymbol{\tau}_0 \end{bmatrix}
\end{gathered} \tag{7}$$

$\boldsymbol{y}_0$ is obtained from (7) and subsequently the optimum $\boldsymbol{\tau}$ is driven by subscribing $\boldsymbol{y}_0$ in $\boldsymbol{\tau} = \boldsymbol{\tau}_o + \boldsymbol{A}\boldsymbol{y}$. Considering $\boldsymbol{B}_0$ as an invertible matrix, this problem will lead to a unique solution. As proved in [10], $\boldsymbol{B}_0$ is always invertible except at singular configurations of the manipulator.

Case 3 is the more often case to happen, which is the composition of two later cases. In this case, some of the forces perch on the boundaries (for some $i \in [1, n], r_i(\boldsymbol{y}_0) = 0\ \&\ \mu_i \geq 0$ therefore $\tau_i = \tau_{min_i}$) and the other ones are inside the boundaries of inequality constraints (for some $j \in [1, n],\ j \neq i, r_j(\boldsymbol{y}_0) < 0\ and\ \mu_j = 0$). In this case $\boldsymbol{y}_0$ is driven through solving the following matrix equation [10]:

$$\boldsymbol{B} \cdot \boldsymbol{X} = \boldsymbol{C} \qquad \boldsymbol{X} = \boldsymbol{B}^{-1} \cdot \boldsymbol{C}$$

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_0 & \begin{bmatrix} \left[\vec{\boldsymbol{a}_1}^T, ..., \vec{\boldsymbol{a}_{n_i}}^T\right] \\ \underline{0} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} \vec{\boldsymbol{a}_1} \\ \vdots \\ \vec{\boldsymbol{a}_{n_i}} \end{bmatrix} & \underline{0} \end{bmatrix} & [\underline{0}]_{m\times m} \end{bmatrix}$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{y}_0 \\ \boldsymbol{\lambda} \\ \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n_i} \end{bmatrix} \end{bmatrix} \quad \boldsymbol{C} = \begin{bmatrix} 2\boldsymbol{A}^T\boldsymbol{\tau}_0 \\ \begin{bmatrix} \tau_{min_1} - \tau_{0_1} \\ \vdots \\ \tau_{min_{n_i}} - \tau_{0_{n_i}} \end{bmatrix} \end{bmatrix} \tag{8}$$

Where, n_i is the number of $\mu_i > 0$ and for each $\mu_i > 0$, $\vec{\boldsymbol{a}_i}$ is the corresponding row vector of matrix $\boldsymbol{A}$. This matrix equation has a solution if matrix $\boldsymbol{B}$ is invertible. In [10] it is proved that $\boldsymbol{B}$ is invertible in the whole wrench feasible workspace of the manipulator.

III. Search Algorithm

In order to implement the prescribed method, this section deals with an overview on the method that is deliberately studied in [10].

First, it is assumed that all forces lie inside the boundaries of solution set defined by inequality constraints, i.e. $\forall i \in [1, n],\ r_i(\boldsymbol{y}) < 0$. With this assumption, the first loop is the same as case 2 and $\boldsymbol{\tau}$ is obtained as detailed. If this solution satisfies all the optimization constraints, i.e. $g(\boldsymbol{y}) = 0$ and $r(\boldsymbol{y}) \leq 0$; the algorithm is terminated and is valid and optimized. Otherwise, the combination of forces which may lie on the boundaries of inequality constraints must be found. There exist $C(m, S)$ different combinations for this search, where $\boldsymbol{m} = [1, 2, \cdots, n]^T$ and S represents the step which is between 0 and n ($s = 0, 1, \cdots, n$). The solution is checked by sweeping all possible combinations. Clearly, the last implementation loop is $S = n$ in which all the forces lie on the boundaries of constraints ($\forall j \in [1, n],\ r_j(\boldsymbol{y}) = 0$) and is the same as the case 1. Thus, $\boldsymbol{\tau}$ is simply found from $\boldsymbol{\tau}_{n\times 1} = [\tau_{min_1}, , \tau_{min_n}]^T$ [10].

It is shown that if there exists a solution, the number of iterations to perform redundancy resolution through this method lies within a certain range which is:

$$1 \leq \text{number of iterations} \leq \sum_{S=0}^{n} C(n,S) \qquad (9)$$

The flowchart of the search routine is given in Fig 1.

IV. IMPLEMENTATION RESULTS

A. *KNTU Cable-Driven Robot*

KNTU cable-driven robot is a planar cable-driven robot built in K. N. Toosi University of Technology for possible high speed maneuvers, and is used in this research to implement the analytic iterative redundancy resolution scheme. Like most other cable robots, this robot is composed of a fixed frame and a moving platform. The moving platform is supported by four cables which connect it to the motors placed on the corner of the fixed frame. Fig 2 illustrates the schematics of this robot.

B. *Simulation Results*

In this section both *AIRR* and *CFSQP* schemes are implemented on *KNTU cable-driven robot* through simulation. Furthermore, constraint optimization command in MATLAB, namely `fmincon`, is used to verify optimal solution obtained. Although this command constructs a reliable numerical optimization method, it only supports Mfile codes and is not executable in Simulink and real-time workshop environment. Therefore, it cannot be used in real-time implementation. This numerical optimization technique includes three different optimization methods, namely, interior-point [12], trust-region-reflective [16] and active-set optimization [17]. Since interior-point is the fastest one among these three methods, it is chosen as the reference for comparison. Fig 3 represents the results of these three methods in a typical simulation of the robot. As it is seen in this figure, both *CFSQP* and *AIRR* methods guarantee minimum force threshold which means the cables are always in tension and furthermore, since the results are identical as the ones obtained from `fmincon`, both methods lead to the right optimal solution.

Since *CFSQP* is C-based block and *AIRR* is M-based code in simulation environment, it is pointless to compare them in term of execution time. Hence, Table I compares `fmincon` as a numerical optimization method with *AIRR* as the analytic one for this trajectory. As it illustrates, *AIRR* executes about 23 times faster than that of numerical optimization methods.

V. REAL-TIME IMPLEMENTATION

1) Preparing the model: For real-time implementation, it is required to embed object code into a Simulink model. *User defined toolbox* in MATLAB is developed for this purpose.

TABLE I
AVERAGE ELAPSED TIME TO SIMULATE REDUNDANCY RESOLUTION

method	Average elapsed time (ms)	Speed (step/s)
AIRR	1.8	574.75
fmincon	43.5	23

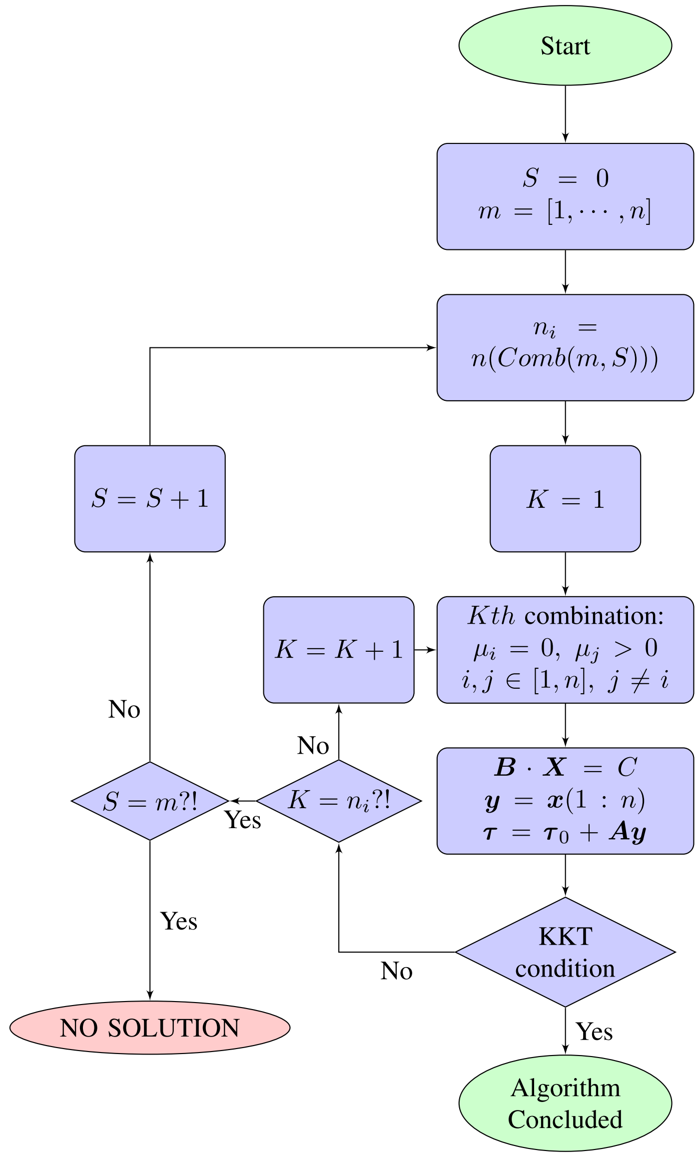

Fig. 1. Search algorithm flowchart

Among its blocks, *Embedded MATLAB Function* is the best for *M-based* codes as it is more convenient to use and a bit faster than *S-function*; however, if the code needs to be segmented into individual parts such as initialization, simulation or termination, separately or if the code is *C-based*, *S-function* block is the best alternative [18].
Both these blocks are used in this research.

2) Connecting to robot: To connect an offline dynamic software to a real-time robot, a real-time software is needed. RT-LAB is one such software that links the Simulink using interface cards and computer networks and enables Simulink models to interact with the real world in real-time. An RT-LAB model can be divided into multiple subsystems and execute complex codes on a network of destination computers, giving each subsystem to one of them simultaneously [19]. Fig 4 illustrates required steps to execute a model in RT-Lab

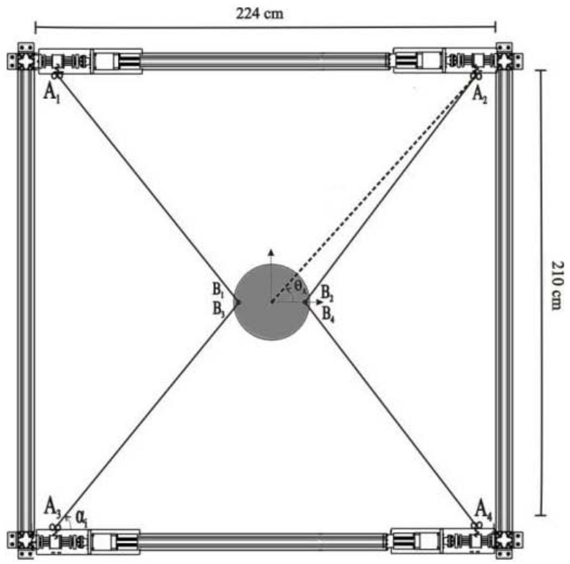

Fig. 2. KNTU Cable-Driven Robot

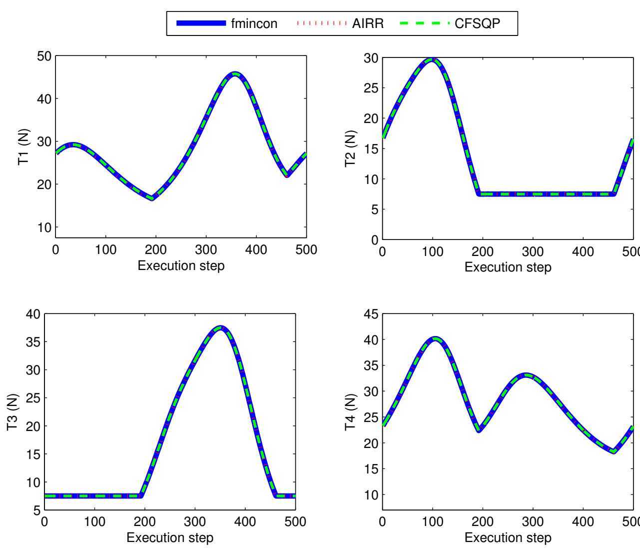

Fig. 3. Cable forces through AIRR, CFSQP and fmincon

environment.

3) Implementation results: In this section, in parallel to the previous numerical method used for redundancy resolution named *CFSQP* [13], the *AIRR* technique is implemented on *KNTU cable robot*. The performance of both methods are then evaluated in term of their execution speed.

For sake of comparison, both methods are implemented through *RT-LAB 8.1.3* and *MATLAB 7.3* as the medium software using an intel dual core CPU with $3GHz$ processor speed and $1.00GB$ ram. Step time for this system is set to $1ms$ and *MATLAB function block* is used to implement *AIRR* codes. As this block does not support some of the commands used in the proposed algorithm, we were forced to write suitable functions to correctly replace that commands.

As a result:

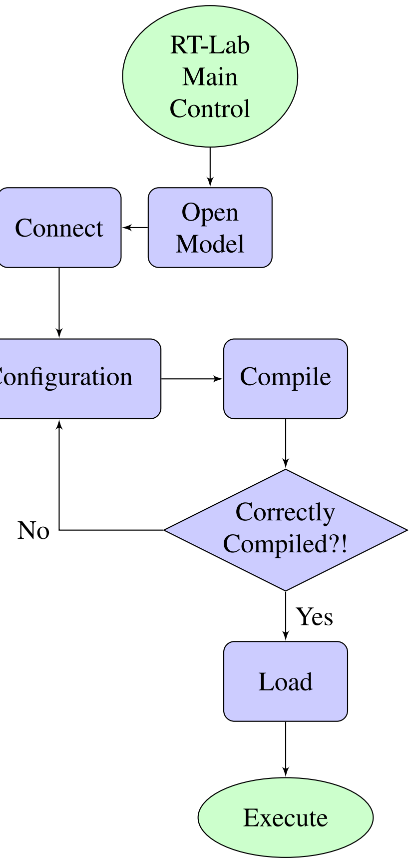

Fig. 4. Connecting to RT-Lab flowchart

1) Both *CFSQP* and *AIRR* lead to the same solution.
2) All the cables remain in tension and the forces are always higher than the minimum tension forces threshold which is set to $\boldsymbol{\tau}_{min} = 7.5N$.
3) When *CFSQP* is in close loop structure, the elapsed time for most steps exceeds the step time of $1\ ms$ and it can cause time over run.
4) The standard deviation of elapsed time vector for *CFSQP* is much higher than that of *AIRR*. This is due to the numerical nature of *CFSQP* which does not ensure a certain execution time period

Fig 5 which is magnified in Fig 6 shows that the elapsed time required to calculate the redundancy resolution scheme through *AIRR* is much less than that of *CFSQP*. The worst elapsed time to perform *AIRR* is $69.97\ \mu s$ which is 4.16 times better than the best case of *CFSQP* $(291.18\ \mu s)$. Table II represents a quantitative comparison of different aspects of these implementation methods. As it is seen in this table the average elapsed time for *AIRR* method is limited to about $54\ \mu s$, which is 15 times faster than that of *CFSQP*. This is emphasized in the fifteen times faster overall execution speed of this method as given in the table. Furthermore, the standard deviation of the execution time in *CFSQP* is very high compared to that of *AIRR*. This reveals the fact that due to numerical nature of *CFSQP* in some occasions the search

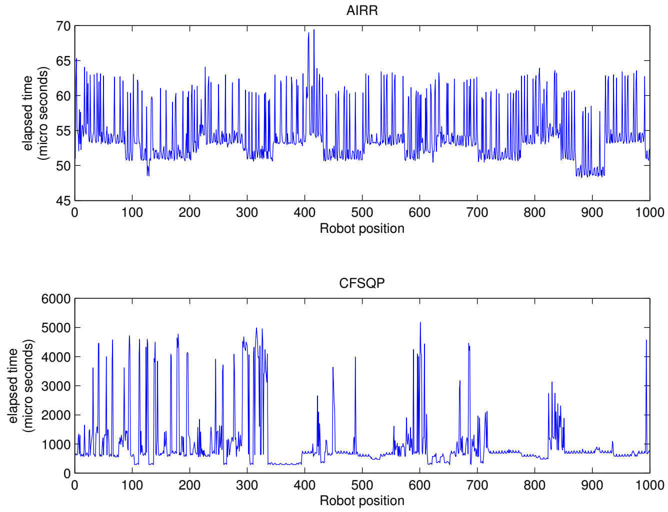

Fig. 5. Required elapsed time to calculate the redundancy resolution scheme

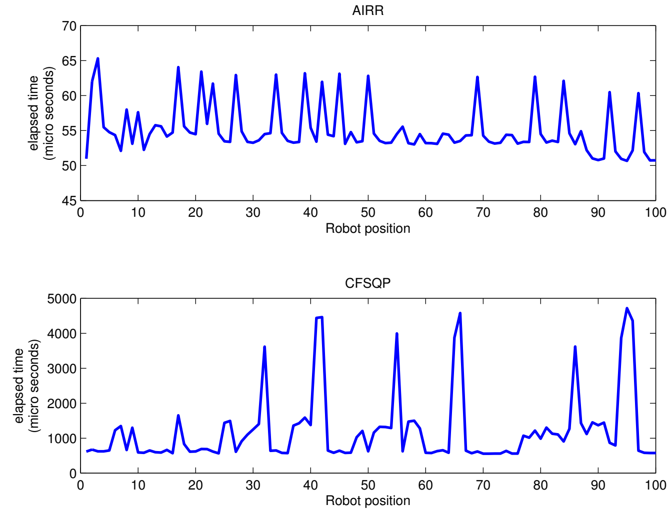

Fig. 6. Magnified Fig 5 over the first 100 seconds

for the optimal solution last much longer than other steps. However, in *AIRR* method the maximum of the iteration time is limited, and in worst case scenario the number of iterations is limited and is quite acceptable.

TABLE II
AVERAGE ELAPSED TIME TO EXECUTE REDUNDANCY RESOLUTION

method method	Average elapsed time (μs)	Speed (step/ms)	standard deviation
AIRR	53.97	18.53	3.89
CFSQP	814.71	1.23	706.30

VI. Conclusions

In this paper an analytic iterative solution to redundancy resolution problem in cable-driven robots is implemented on *KNTU* cable robot. As redundancy resolution for such robots involves nonlinear optimization problems with equality and inequality constraints, *Karush-Kuhn-Tucker* theorem is used to formulate the optimization problem, and an iterative method is presented to find the solution. The performance of the proposed method is verified in simulation and real-time implementation, and it is shown that due to the structure used in analytic iterative solution the execution time and its standard deviation in real-time implementation is significantly reduced compared to that of the numerical one. It is shown that the worst elapsed time to perform *AIRR* is 4.16 times better than that of the best execution time in *CFSQP*. Furthermore the average elapsed time of *AIRR* method is 15 times faster than that of *CFSQP*. Such increase in execution time in real-time implementation is very promising for future applications.

References

[1] Rendong Nan and Bo Peng. A chinese concept for the 1 km2 radio telescope. *Acta Astronautica*, 46(1012):667 – 675, 2000.
[2] M.M. Aref and H.D. Taghirad. Geometrical workspace analysis of a cable-driven redundant parallel manipulator: Kntu cdrpm. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*,
[3] J-P Merlet. *Parallel robots*, volume 128. Springer, 2006.
[4] S. Kawamura, W. Choe, S. Tanaka, and S.R. Pandian. Development of an ultrahigh speed robot falcon using wire drive system. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 215–220 vol.1, May 1995.
[5] Carlo Ferraresi, Marco Paoloni, and Francesco Pescarmona. A new methodology for the determination of the workspace of six-dof redundant parallel structures actuated by nine wires. *Robotica*, 25:113–120, 1 2007.
[6] K. Maeda, S. Tadokoro, T. Takamori, M. Hiller, and R. Verhoeven. On design of a redundant wire-driven parallel robot warp manipulator. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 895–900 vol.2, 1999.
[7] E. Sahin Conkur and Rob Buckingham. Clarifying the definition of redundancy as used in robotics. *Robotica*, 15:583–586, 9 1997.
[8] Maryam Agahi and Leila Notash. Redundancy resolution of wire-actuated parallel manipulators. *Transactions of the Canadian Society for Mechanical Engineering*, 33(4):561, 2009.
[9] B. Zi, B.Y. Duan, J.L. Du, and H. Bao. Dynamic modeling and active control of a cable-suspended parallel robot. *Mechatronics*, 18(1):1 – 12, 2008.
[10] H.D. Taghirad and Y.B. Bedoustani. An analytic-iterative redundancy resolution scheme for cable-driven redundant parallel manipulators. *Robotics, IEEE Transactions on*, 27(6):1137–1143, Dec 2011.
[11] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.
[12] Sanjay Mehrotra and M Gökhan Özevin. On the implementation of interior point decomposition algorithms for two-stage stochastic conic programs. *SIAM Journal on Optimization*, 19(4):1846–1880, 2009.
[13] Craig T Lawrence, Jian L Zhou, and Andre L Tits. User's guide for cfsqp version 2.0: Ac code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. 1994.
[14] Mohammad A Khosravi and Hamid D Taghirad. Robust pid control of fully-constrained cable driven parallel robots. *Mechatronics*, 24(2):87–97, 2014.
[15] M.A Khosravi and H.D. Taghirad. Dynamic modeling and control of parallel robots with elastic cables: Singular perturbation approach. *Robotics, IEEE Transactions on*, 30(3):694–704, June 2014.
[16] Hamid D Taghirad. *Parallel Robots: Mechanics and Control*. CRC Press, 2013.
[17] William W Hager and Hongchao Zhang. A new active set algorithm for box constrained optimization. *SIAM Journal on Optimization*, 17(2):526–557, 2006.
[18] Simulink dynamic system simulation for matlab. 1998.
[19] Rt-lab solo users manual. 2004.