Feasible Workspace for Robotic Fiber Placement

Serge R. Moutran

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

May 10, 2002
Blacksburg, Virginia

Keywords: Feasible Workspace, Path Verification, Fiber Placement, Trajectory Simulation

Feasible Workspace for Fiber Placement

by
Serge R. Moutran

(ABSTRACT)

Online consolidation fiber placement is emerging as an automated manufacturing process for the fabrication of large composite material complex structures. While traditional composite manufacturing techniques limited the products' size, geometrical shapes and laminate patterns, robotic automation of the fiber placement process allows the manufacture of complex bodies with any desired surface pattern or towpreg's direction. Therefore, a complete understanding of the robot kinematic capabilities should be made to accurately position the structure's substrate in the workcell and to compute the feasible product dimensions and sizes.

A Matlab algorithm is developed to verify the feasibility of straight-line trajectory paths and to locate all valid towpreg segments in the workspace, with no focus on optimization. The algorithm is applied preliminary to a three-link planar arm; and a 6-dof Merlin robot is subsequently considered to verify the towpreg layouts in the three-dimensional space. The workspace is represented by the longest feasible segments and plotted on parallel two-dimensional planes. The analysis is extended to locate valid square areas with predetermined dimensions. The fabrication of isotropic circular coupons is then tested with two different compaction heads. The results allow the formulation of a geometric correlation between the end-effector dimensional measures and the orientation of the end-effector with respect to the towpreg segments.

AUTHOR'S ACKNOWLEDGMENTS

This work was funded by the National Center for Advanced Manufacturing-NASA. Their support is gratefully acknowledged.

I would like to thank my advisor, Dr. Robert Sturges for his constant monitoring and assistance throughout my research. His great experience and his constructive comments guided me to complete this work and accomplish my research goals.

I would like to thank my NCAM teammates, Mark Abbott and Munki Lee, and my colleagues, Amnart Kanarat and Roger Anderson for their permanent support. Their help and friendship are gratefully acknowledged.

I would like to thank my beloved family and my precious friends, Linda Gallegos, Barbar Akle, Walid El-Aouar and Samer Katicha for your cherished love and constant support.

TABLE OF CONTENTS

AUTHOR'S ACKNOWLEDGMENTS………………………………………………………. iii
TABLE OF CONTENTS……………………………………………………………… iv
LIST OF FIGURES……………………………………………………………………….. vi
LIST OF TABLES…………………………………………………………………………ix
CHAPTER 1: INTRODUCTION ON THE MANUFACTURING OF COMPOSITE MATERIALS……………………………………………………………………………1
1.1 Composite Materials…………………………………………………………….. 1
1.2 The Manufacturing Techniques for Composite Material Structures……….………. 2
1.2.1 Molding…………………………………………………………………………. 2
1.2.2 Online Consolidation……………………………………………………………2
CHAPTER 2: THE KINEMATIC REQUIREMENTS OF THE ROBOTIZED FIBER PLACEMENT TASK……………………………………………………………….. 10
2.1 Constant Placement Velocity…………………………………………………….. 10
2.2 Continuous Fiber Placement……………………………………………………… 10
2.3 Freedom to Orient the End-Effector about All Axes……………………………... 11
2.4 Compression Force on the End-Effector…………………………………..………13
CHAPTER 3: ROBOTIC DEXTERITY AND TRAJECTORY VERIFICATION REVIEW…………………………………………………………………………….. 14
CHAPTER 4: ROBOT RESTRICTIONS IN FIBER PLACEMENT………………………20
4.1 The Reachable Workspace of the Robot……………………………..…………20
4.2 High Degree of Freedom Robots………………………………….……………… 20
4.3 The Dynamic Limits on the Robot Joint…………………………...……………21
4.4 Accuracy and Repeatability of the Robot……………………………………...…. 22
CHAPTER 5: THE EFFECTIVE WORKSPACE FOR FIBER PLACEMENT…………….23
CHAPTER 6: THE MERLIN ROBOT……………………………………………………27
6.1 The Merlin Robot Representation………………………………………………29
CHAPTER 7: THE THREE-LINK PLANAR ARM……………………………………….. 32
7.1 Techniques to Map the Feasible Workspace of the Three-Link Planar Arm………...33
7.1.1 Lines Parameters in the Vertical Plane…………………………………………... 33
7.1.2 Forward Kinematics for the Three-Link Planar Manipulator ………………………..37
7.1.3 The Compaction Head Velocity and Exterior Forces………………………………..38
7.1.4 Collision Detection for the Three-Link Planar Manipulator…………………………39
CHAPTER 8: THE DISTINCT TECHNIQUES FOR THE MERLIN ROBOT…………….42
8.1 Lines Parameters in the Three-Dimensional Space………………………….…… 42
8.1.1 The Towpreg Yaw, Pitch and Roll…………………………………………………..42
8.1.2 The Towpreg Offsets or Locations Parameters……………………………………43
8.1.3 The Orientation of the End-Effector with respect to the Line………………………43
8.1.4 The Different Robot Configurations…………………………………………………45
8.2 Inverse Kinematics……………………………………………………………….. 45
8.2.1 A Brief Comparison of the Forward Inverse Kinematics Techniques…………………48
8.3 The Head Velocity and Compaction Force…………………….……………………49

8.4 Collision Detection……………………………………………………………….. 50
CHAPTER 9: THE ALGORITHM TO DETERMINE ALL FEASIBLE TOWPREG PATHS……………………………………………………………………………54
9.1 The Input Parameters and Specifications……………………………………………54
9.2 Computing the Constant End-Effector Orientation, Velocity and Compaction Force Vectors…………………...……………………………………………………57
9.3 Collecting All Data along the Line…………………………………………………..58
9.3.1 Inverse Kinematics………………………………………………………………..59
9.3.2 Computing the Joints Velocities, Torques and the Jacobian Determinant……………..60
9.4 Determining All Feasible Segments on the Line………………………………….… 61
9.4.1 Checking the Kinematic Limits on Every Discrete Point………………………………61
9.4.2 Determining All Feasible Segments on the Line………………………………………62
9.4.3 Collision Detection and Determining Feasible Sub-Segments…………………………68
CHAPTER 10: RESULTS: THE MAPPED FEASIBLE WORKSPACE………………….. 73
10.1 Plotting All Valid Towpreg Segments in the Two-Dimensional Plane…………….74
10.2 Plotting the Longest Valid Towpreg Segments in the Two-Dimensional Plane… 75
10.3 The Feasible Workspace Variations with respect to the Line Parameters……….…79
10.3.1 The Orientation of the End-Effector with respect to the Tow Segment………………79
10.3.2 The Offsets Parameters………………………………………………………………. 81
10.3.3 The Line Yaw, Pitch and Roll Orientation Parameters………………………………83
10.4 The Feasible Layout on Squared Surfaces…………………………………………84
CHAPTER 11: FIBER PLACEMENT FOR THE FABRICATION OF ISOTROPIC FLAT COUPONS………………………………………………………………………90
11.1 Isotropic Composite Structures……………………………………………………..90
11.2 The Manufacture of Isotropic Flat Coupons………………………………………..93
11.2.1 The End-Effector with the Compaction Roller……………………………………… 95
11.2.2 The End-Effector with the Compaction Ball…………………………………….....100
CHAPTER 12: CONCLUDING REMARKS………………………………………………103
12.1 Summary and Discussion…………………………………………………………103
12.2 Future Work……………………………………………………………………… 105
REFERENCES……………………………………………………………………………...107
APPENDIX A: VERTICAL AND HORIZONTAL VIEWS OF THE MERLIN ROBOT REACHABLE WORKSPACE……………………………………………………………..110
APPENDIX B: THE ROTATIONAL SPAN OF SIX THE MERLIN ROBOT JOINTS….112
APPENDIX C: THE DENAVIT-HARTENBERG PARAMETERS………………………113
APPENDIX D: THE INVERSE KINEMATICS EQUATIONS FOR THE MERLIN ROBOT……………………………………………………………… 114
APPENDIX E: PICTURES OF THE ROLLER AND BALL COMPACTION HEADS….116
APPENDIX F: THE MATLAB CODE………………………………………………….117
VITA……………………………………………………………………………………….. 174

LIST OF FIGURES

Figure 1.1: Schematic of the online consolidation process………………………………….. 3
Figure 1.2: Filament winding machine…………………………………………………… 5
Figure 1.3: The unachievable change in the roller orientation……………………………..5
Figure 1.4: Filament winding apparatus: compaction pressure provided
by the fiber tension…………………………………………………………………..5
Figure 1.5: Possible configuration of the compaction head………………………………..6
Figure 1.6: Steered towpreg paths with different radii of curvature…………………………7
Figure 1.7: The relative manufacturing costs versus the layup rate…………………………..9
Figure 2.1: Rotation of the end-effector about the Y axis………………………………….11
Figure 2.2: Rotation of the end-effector about the Z axis…………………………………..12
Figure 2.3: Rotation of the end-effector about the X axis………………………………….. 12
Figure 5.1: Breaking curved tow paths to a series of verifiable straight lines………………24
Figure 5.2: The approach to verify structure fabrication using the feasible workspace……. 25
Figure 6.1: The Merlin robot…………………………………………………………….. 27
Figure 6.2: The Merlin robot six joints……………………………………………………...27
Figure 6.3: The Denavit-Hartenberg frames on the Merlin robot………………………….. 30
Figure 7.1: Freezing joints 1, 4 and 6…………………………………………………… 32
Figure 7.2: The three-link planar arm assumption…………………………………………..33
Figure 7.3: Exactly the same product can be produced but in different locations or
orientations. Tow alignment is shown by parallel line segments………………. 34
Figure 7.4: Structures that require different inclination angles in the vertical plane………..35
Figure 7.5: The orientation of the end-effector with respect to the towpreg segment………36
Figure 7.6: Determining the relation between the three joint angles in the forward
Kinematic method………………………………………………………………. 37
Figure 7.7: The Velocity and Force Vectors…………………………………………….. 39
Figure 7.8: The hydraulic cylinder pressure providing the controlled compression force…. 39
Figure 7.9: Possible collisions of links 2 and 3 with the tow……………………………… 40
Figure 7.10: The collision check point for link2 and towpreg segment impact detection…..41
Figure 8.1: The yaw, pitch and roll angles of the towpreg tape…………………………… 43
Figure 8.2: The three offsets of the towpreg tape………………………………………….. 43
Figure 8.3: The orientation of the roller with respect to the towpreg segment in
the 3-D space……………………………………………………………………44
Figure 8.4: The eight different robot configurations……………………………………….. 47
Figure 8.5: The Elbow Up/Down Inverse Kinematics Solutions……………………………48
Figure 8.6: The randomly spaced points on the linear path…………………………………48
Figure 8.7: Surfaces enclosing the link……………………………………………………...50
Figure 8.8: Locating the points with respect to the surface S………………………………51
Figure 8.9: The robot waist…………………………………………………………………52
Figure 8.10: The counterweight, connector and arm……………………………………….. 53
Figure 8.11: The robot base and trunck…………………………………………………….. 53
Figure 9.1: The dimensions of the end-effector……………………………………………. 55

Figure 9.2: The algorithm flowchart………………………………………………….. 56
Figure 9.3: Collecting all needed data on the discrete points……………………………… 58
Figure 9.4: Categorizing the set of angles solutions according to the robot configuration… 59
Figure 9.5: All kinematic data categorized and stored…………………………………….. 61
Figure 9.6: The algorithm partial flowchart determining all feasible segments…………… 62
Figure 9.7: Singularity occurring between points with opposite Jacobian
determinant signs…………………………………………………………….. 63
Figure 9.8: Checking for singularities when the determinant curve peaks
towards the zero line………………………………………………………… 64
Figure 9.9: Computing and checking the Jacobian determinant for the fine
points created………………………………………………………………... 64
Figure 9.10: a) Locating the critical interval on the determinant curve, b) Fine
Discrete points all meeting the velocity restrictions, c) The angles
derivatives show the actual velocity curve exceeding the maximum limits….. 67
Figure 9.11: Flowchart to determine the all feasible segments…………………………... 68
Figure 9.12: The kinematic data along a single valid segment…………………………….. 69
Figure 9.13: The six joints angles along a single valid segment…………………………… 69
Figure 9.14: Two collision-free sub-segments valid within the original valid segment……. 71
Figure 9.15: The algorithm flowchart to determine the feasibility of segments
in detecting collision………………………………………………………….. 72
Figure 10.1: The feasible workspace presented by all the valid segments…………………. 74
Figure 10.2: Segments overlapping…………………………………………………………. 75
Figure 10.3: The workspace envelope of all longest segments in the plane………………... 76
Figure 10.4: Feasible linear paths outside the longest segments envelopes………………... 77
Figure 10.5: The longest segments envelope in a vertical plane parallel to the YZ axes…... 78
Figure 10.6: The longest segments envelope in a vertical plane parallel to the XZ axes…... 79
Figure 10.7: The workspace envelopes for 90˚ (solid), 115˚ (dotted) and
140˚ (dashed) end-effector orientation parameter…………………………….. 80
Figure 10.8: The location of the contact point with respect to the wrist…………………… 81
Figure 10.9: Parallel workspaces plotted on the same figure………………………………. 82
Figure 10.10:The parallel vertical envelopes for 35 in.(solid), 40 in. (dotted) and
25 in.(dashed) X-offsets……………………………………………………….. 83
Figure 10.11: The workspace envelopes for (a) 0˚, (b) 45˚, and (c) 90˚ line pitch angles…. 84
Figure 10.12: Inclined lines Offset……………………………………………………….. 84
Figure 10.13: Locating square candidates and determining feasibility…………………….. 85
Figure 10.14: The algorithm flowchart to determine the feasible squares in the
desired plane…………………………………………………………………. 86
Figure 10.15: The feasible squares in the vertical plane…………………………………… 87
Figure 10.16: The inclined feasible squares in the vertical plane parallel to the XZ axes…. 88
Figure 10.17: The feasible squares for different workspace elevations……………………. 89
Figure 10.18: The longest segments envelope and the feasible squares on the
same planar workspace……………………………………………………… 89
Figure 11.1: Longitudinal and perpendicular loads applied on the laminate ply…………… 90

Figure 11.2: Stacking layers with different orientations……………………………………91
Figure 11.3: Overlapping of squares with different inclinations on the same plane……….. 92
Figure 11.4: The tolerance window to check the complete overlapping of squares………...93
Figure 11.5: The end-effectors configurations with the measures: S_1 and S_2……………… 94
Figure 11.6: The circular area covering the four overlapping square layers……………….. 95
Figure 11.7: The largest feasible circular coupons for a 90 degrees end-effector orientation…………………………………………………………………….. 95
Figure 11.8: The locations of the 10 in. squares with the different inclination…………….. 96
Figure 11.9: The reachable workspace offset in the direction of the lines orientations……. 96
Figure 11.10: The offset distance between the roller contact point and the wrist center……97
Figure 11.11: The contact point and the wrist center aligned on the normal to the towpreg line…………………………………………………………………..98
Figure 11.12: The largest (16-inch) isotropic coupons……………………………………...98
Figure 11.13: The feasible 16-inch squares that would form the valid the coupons layers: (a) -45˚ inclined squares, (b) 0˚, (c) 45˚ and (d) 90˚……………………… 99
Figure 11.14: The numerous 15-inch isotropic coupons…………………………………… 99
Figure 11.15: Horizontal view of the ball compaction head oriented in various directions in the consolidation process……………………………………...101
Figure 11.16: Relatively large coupons for β equal to 90 degrees. The layers orientations are {-45, 0, 45, 90} degrees in (a) and {-67.5, -22.5, 22.5, 67.5} degrees in (b)…………………………………… 102
Figure 11.17: Feasible coupons when alternating β for the different layer orientations….. 102
Figure A.1: The vertical view of the Merlin robot reachable workspace…………………. 110
Figure A.2: The horizontal view of the Merlin robot reachable workspace………………. 111
Figure C.1: The Denavit-Hartenberg parameters shown on the Merlin robot……………..113
Figure E.1: The roller compaction head mounted on the wrist of the Merlin robot………116
Figure E.2: The ball compaction head mounted on the wrist of the Merlin robot…………116

LIST OF TABLES

Table 1.1: Fabrication capabilities comparison……………………………………………….8
Table 6.1: The Merlin robot positional and orientational joints……………………………..28
Table 6.2: The Merlin joints velocity limits…………………………………………………29
Table 6.3: The Merlin joints torque limits…………………………………………………...29
Table B.1: The rotational extent of the six Merlin robot joints……………………………112
Table C.1: The Denavit-Hartenberg parameters for the Merlin robot…………………….. 113

CHAPTER 1: INTRODUCTION ON THE MANUFACTURING OF COMPOSITE MATERIALS

1.1 Composite Materials

Composite materials provide unique properties and features that have become the central focus in high performance part manufacturing. Their potential benefits attracted researchers and their multiple advantages over other materials were proven in many manufacturing applications. Composite material properties match the demanding and critical specifications in the aerospace industry so that recent product designs are now being dominated by the increasing use of composite material structures.

A composite is a structured combination of fibers and a binding matrix that maximizes specific performance properties. None of the composite elements merges completely with the other. The matrix transfers loads between the fibers and protects them from aggressive environments. The properties of the final composite are superior and possibly unique in some respects, to those of the individual constituents. Varying the fiber's volume fraction across sections optimizes the composite material properties to meet specific properties required for the final product. Furthermore, a wide variety of matrix/fiber combinations are available to give the exact properties that all distinct applications demand.

Composites cost, reliable performance, appearance, and safety attracted industries in several manufacturing fields: Composites are being intensively used as structural components in the aircraft and aerospace industries mainly because of their excellent fatigue performance along with their high strength-to-weight and high stiffness-to-weight ratios. Furthermore, their thermal characteristics allowed the design of high precision instruments with negligible thermal expansion coefficients. High creep resistance and low-moisture absorption give excellent dimensional stability and extend the use of composites in a variety of consumer and industrial products. Composites applications include circuit boards, thermal and electrical insulators.

Most composite materials are available in ready-to-manufacture forms wherein the resin and the fibers are premixed and arranged according to the future fabrication process and the application of the final product. Bulk Molding Compound (BMC) is a premixed blend containing resin, fibers, fillers and various additives. Composites are available in more structured arrangements, *e.g.* Sheet Molding Compounds (SMC), in which they are processed in a continuous sheet form. Unidirectional properties are achieved with SMC-Directional or SMC-Continuous that contains long continuous fibers oriented in one direction.

Composites are also stored as thin towpreg tapes, or unidirectional continuous fibers pre-impregnated with resin in a flat form. Towpreg tapes are used in almost all composite-manufacturing processes, discussed in the next section.

1.2 The Manufacturing Techniques for Composite Material Structures

Several different manufacturing processes were developed for the fabrication of composite material structures. Starting with the ready-to-manufacture forms, many fabrication techniques allowed the consolidation and curing of the individual sheets or tapes to form the final composite product.

1.2.1 Molding

Many molding procedures were developed for the manufacturing of composite structures:

Autoclave molding consists of placing the entire premixed assembly into an autoclave (or closed vessel with pressure and heat capabilities) to consolidate the structure layers. *Bag molding* and *vacuum bag molding* require the use of a flexible bag to cover the composite while pressure is applied by autoclave, vacuum, or by inflating the bag. In *compression molding*, the material is shaped by heat and pressure until it attains the final form of the mold.

1.2.2 Online Consolidation

On-line consolidation is an alternative technique for the manufacture of composite material. The focus of many current studies is being centered on this area where remarkable advances are being developed to improve this manufacturing process.

On-line consolidation can be described as a composite manufacturing technique where resin impregnated fiber tows (towpreg tapes) are simultaneously laid, heated, consolidated and cured in a single step [1]. Since heat and pressure are applied continuously during tow placement, the need for an autoclave or hot-press consolidation for the part is eliminated.

Multiple repetitions of the above procedure would achieve the desired thickness of a structure. Figure 1.1 shows a detailed schematic of the on-line consolidation technique. While the incoming towpreg tape is being fed in the desired orientation, a heat source is focused on the interface between the substrate and the incoming tape to ensure melting on both mating surfaces. Pressure is then applied to ensure the proper consolidation by squeezing the resin into the composite gaps [1].

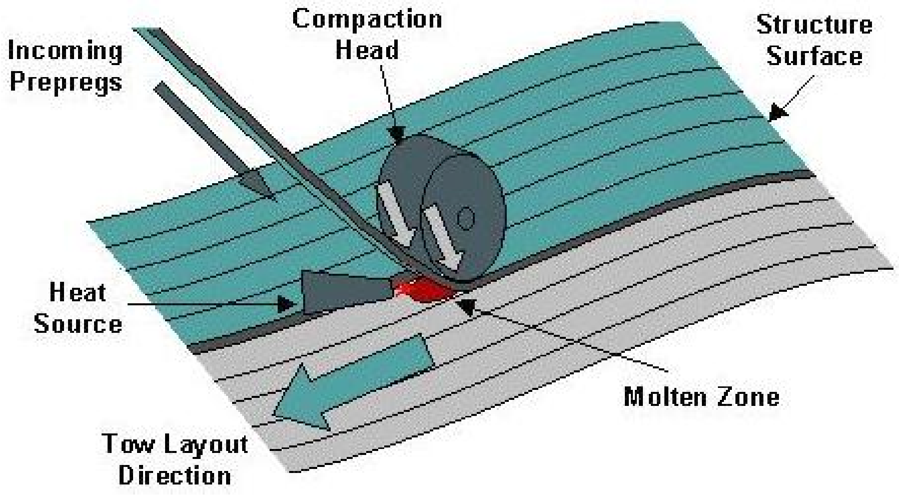


Figure 1.1 Schematic of the on-line consolidation process

A Manual Laying

Manual on-line consolidation is performed by manually placing and compressing the towpreg tapes on the substrate to ensure the curing of the part. This demanding procedure requires high manipulability to accurately place tapes to within 1 mm or less tolerances. Accurate manual pressure is also needed to compact the towpreg tape on the substrate. The difficulty increases with large part sizes and dimensions [2].

Lack of accuracy and precision of manual placement encouraged the automation of this process and the development of several techniques that provide more control and consistency while reducing direct labor costs.

B Automated Online Consolidation Processes

1) Tape Laying:

Automated tape laying provides the required repeatability and accuracy for composite manufacture. The fabrication technique resembles to a great extent manual laying but the automation of the mechanism is achieved by a well-controlled consolidation head, moved and oriented by a multi axis machine. Although direct labor costs are greatly reduced, the mechanism lacks the ability to manufacture a wide range of three-dimensional structures, and to generate several desired surface patterns.

2) Filament Winding:

Filament winding is another automated process where the fabrication of the composite material can be achieved with the on-line consolidation technique.

The continuous towpreg tape is unwound from the spool mounted on a tensioner and then wound again around a rotating mandrel [1]. Figure 1.2 shows the basic apparatus of the system. The consolidation head assembly and the delivery system are stationary whereas the mandrel rotates for the winding of the towpreg. A linear carriage motion moves the mandrel along its axis of rotation to enable the fabrication of parts along the whole length of the mandrel. The apparatus in Fig. 1.2 includes a pressure roller for the compaction of the towpreg, and air-cylinder to control the compression forces.

Filament winding suffers from the limitations on the feasibility of certain geometrical structures and pattern designs. Structures are generated by winding the towpreg around the mandrel and consequently flat surfaces, multiple axis bodies or complex three-dimensional products are not possible.

Since the roller should always be normal to the incoming towpreg tape, the fixed mechanism of the compaction head implies that towpregs are accordingly laid in only one fixed orientation and consequently many other important composite geometrical patterns are not feasible. Figure 1.3 explains the required change in the roller orientation to allow the generation of simple inclined surfaces (i.e. cones).

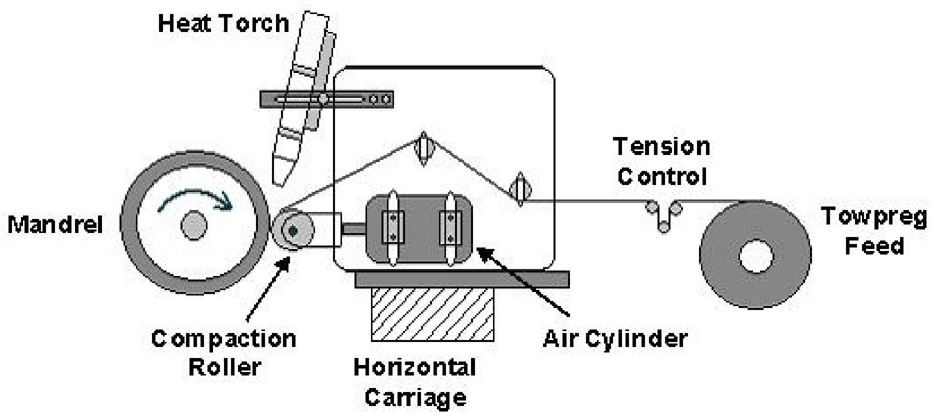


Figure 1.2 Filament Winding Machine [1]

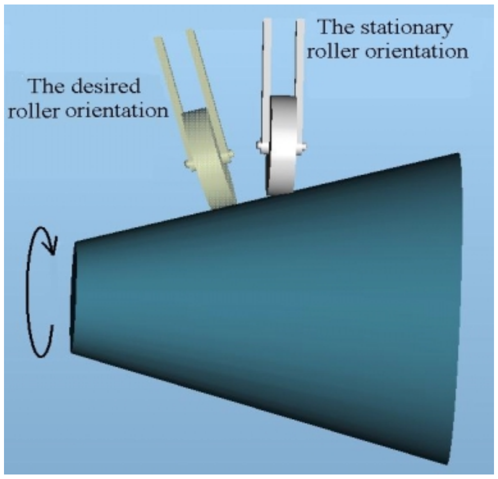

The stationary roller orientation
The desired roller orientation

Figure 1.3 The unachievable change in the roller orientation

Other filament winding mechanisms rely only on towpreg tension to provide the necessary pressure for consolidation (Fig. 1.4). Here again, fabricated structures are also limited to a narrow range of geometrical shapes. Winding tension requires the towpreg to follow the geodesic path - the shortest path between two points - and consequently, convex curvature fabrication is not possible [3].

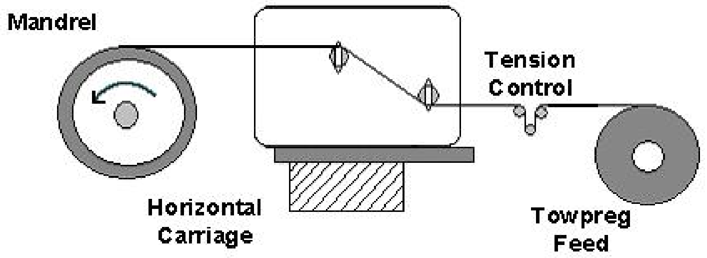


Figure 1.4 Filament winding apparatus: compaction pressure provided by the fiber tension

3) Robotic Fiber Placement:

Robotic fiber placement is the only feasible automated on-line consolidation process to fabricate complex-shaped three-dimensional composite structures to date. This technique is capable of achieving all desired towpreg tape orientations and patterns in the different layers of the generated product.

As in filament winding, robotic fiber placement involves continuous consolidation of a towpreg on an already cured material surface. Simultaneous heat and pressure applications ensure the curing and the bonding of the structure interface [4]. The fiber placement consolidation head is manipulated by a six-dof (or redundant) robot able to provide all desired three-dimensional locations for the towpreg layout and most importantly to provide all the desired towpreg orientations. The fiber placement process involves the 'tow cut and start' operation in which the layout stops at a chosen location, the towpreg is cut and the layout restarts at a different desired position. Accordingly, the consolidation head should include the cut, start and feed functions for separate tows. Figure 1.5 shows a possible configuration of the consolidation head that combines the compaction roller and the localized heat source needed for the proper curing of the material.

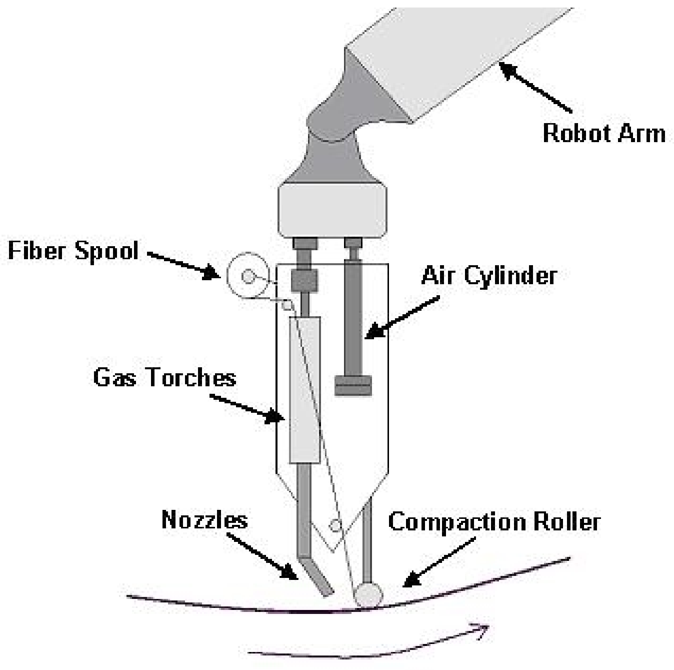


Figure 1.5 Possible configuration of the compaction head [3]

Towpregs are supplied by a creel that applies a slight tension for the control of the tapes in the delivery and transport systems. Tension is kept low in order to ensure the feasibility of fabricating concave shapes. Since the end-effector handles the ‘cut and start’ functions, the movement of the fabrication process is continuous with no halting or labor intervention while material waste during the manufacturing process is minimal [5].

The ability to freely manipulate the fiber placement consolidation head allows the generation of complex patterns and three-dimensional paths: by choosing the start and the end of every separate towpreg, the structure’s width can be varied in adding or dropping tows from specific bands [2]. The ability of the end-effector to bend towpregs enables the fabrication of curved and concave surfaces. In addition, the consolidation head is capable of steering the tow bands on the substrate. Steered paths reduce the generation of discontinuous paths by laying long continuous towpregs around holes and ports [6]. Figure 1.6 shows several towpreg tapes laid with different radii of curvature.

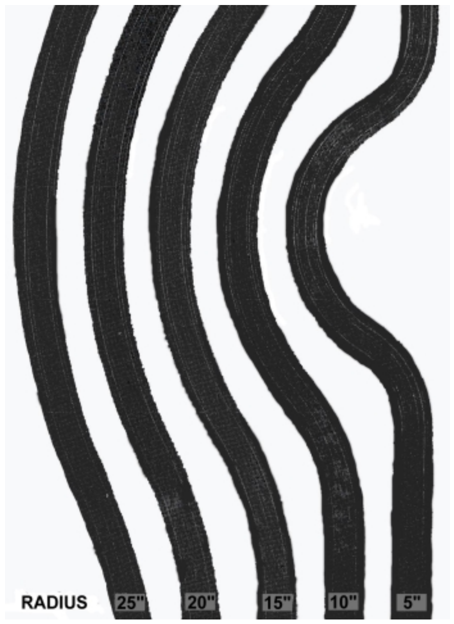


Fig. 1.6 Steered towpreg paths with different radii of curvature [6]

The fabrication capabilities of fiber placement are compared to other automated manufacturing techniques shown in Table 1.1: Fiber placement solves all limitation problems on the feasible geometrical shapes and paths.

Robotic computer simulation allows manufacture testing and predicts good approximations for parts cycle time with no investment costs in tooling or material. Figure 1.7 graphs the relative manufacturing cost versus the layup rate (in Kg/hr) and compares the performance of the fiber placement technique to the hand layup manufacturing process. Fiber placement proves to be cost effective allowing the fabrication of composite material at relatively high speeds.

Fabrication Process	Non-Geodesic Paths	Concave Surfaces	3-D Surfaces	Ply Drops	Compaction
Fiber Placement	√	√	√	√	√
Tape Laying	–	√	–	√	√
Filament Winding	Limited by Slip	–	√	–	√ or Applied by Tension

Table 1.1 Fabrication capabilities comparison [2]

Interface of a Six-dof Manipulator with a Rotating Mandrel:

As discussed earlier, *filament winding* relies on the rotation of the mandrel to wind the towpreg and manufacture the desired part. The consolidation head is stationary and a linear carriage moves the substrate only along its axis of rotation. To overcome the mechanism limitations to locate and orient the towpreg layouts, the system performance is greatly improved by including a six-dof robot in the manufacturing workcell. While a servo motor controls the rotation of the mandrel, the robot manipulates the consolidation head and provides all desired orientations for the towpreg tapes layout. The interface of the robot with the servo motor enables non-geodesic winding and allows the fabrication of any structure of one-axis of rotation. Cone sections are perfectly generated and precise coordination between the robot and the motor motions permits the fabrication of bodies with convex or concave surfaces [3].

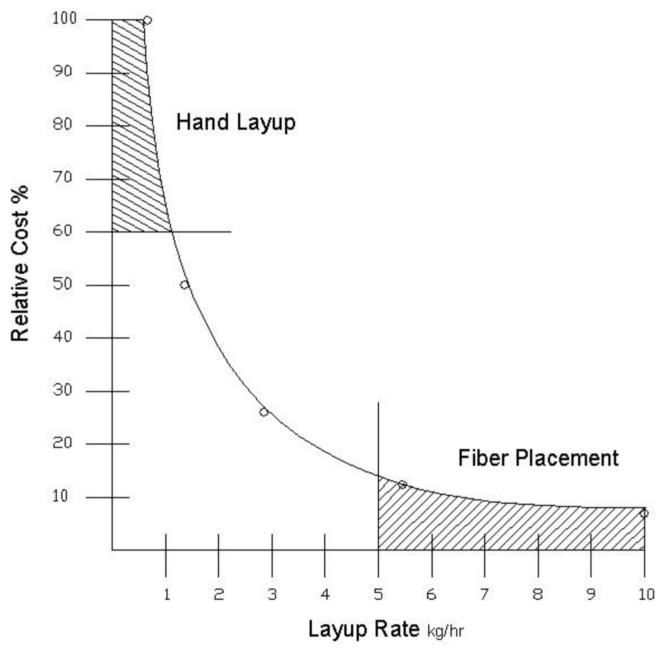


Figure 1.7 The relative manufacturing costs versus the layup rate [2]

CHAPTER 2: THE KINEMATIC REQUIREMENTS OF THE ROBOTIZED FIBER PLACEMENT TASK

To fabricate the highest quality products, the motion trajectory of the consolidation head should meet four kinematic requirements that greatly influence the proper consolidation and curing of the manufactured structure.

2.1 Constant Placement Velocity

The online consolidation task involves heating the tow while it is pressed into place. Therefore, a nearly constant velocity of the heat source is necessary to avoid overheating in some spots, which could lower the quality of the composite structure. Deviations from the desired layout velocity definitely vary the heat absorption along the length of the towpreg and consequently, might cause deterioration of the fibers or incomplete heat curing. In addition, since compression is needed for good consolidation, a constant velocity allows a more uniform final product as the compression period is constant and uniform along the prepreg path. The constant compaction time should be adequately predefined to fill the voids formed by entrapped air, uniformly over the composite structure to fabricate.

The constant velocity requirement makes the online consolidation task similar to the robotic welding task. As in the online consolidation task, we do not want to overheat some spots along the welding line trajectory.

2.2 Continuous Fiber Placement

Unlike welding or even wall spraying, the ‘cut and start’ process in fiber placement requires the continuous layout of the towpreg along the designed path. Once the prepreg tape is placed on the starting point on the structure surface, the motion of the placement head should not be interrupted until the designed path end point is reached, the towpreg is laid on the desired trajectory and cut to restart a new trajectory with a separate towpreg.

Many robotic applications can be perfectly ‘intervallic’ without affecting the quality of the task. They are flexible in halting the trajectory of the end-effector and rescheduling its continuation to a different time. This important feature controls the application process in case of unexpected events (*i.e.* power shut-down) or sudden malfunctions in the work-cell.

Robotized fiber placement *cannot* handle interruption of the towpreg layout. Once the tape is positioned on its starting point, the towpreg is consolidated gradually along its length on the structure surface. The fiber spool is meanwhile ‘attached’ to the structure till the designed trajectory is achieved and the towpreg is cut. The basic problem of interrupting the towpreg flow can be partially solved by cutting the tape at the stoppage point and laying a separate tape on the rest on the designed path. But still, the layout of multiple separate towpreg on the continuous designed trajectory causes losses in the mechanical strength or esthetic properties of the final structure and should definitely be avoided in all means.

Briefly, fiber placement cannot handle interruption in the towpreg flow and the continuous layout is required for the best quality products.

2.3 Freedom to Orient the End-Effector about all Axes

As stated earlier, the end-effector includes a compaction head, traditionally a roller, which guides and compresses the tow on the substrate and subsequent composite layers. The geometry of the roller adds new restriction on the path of the end-effector: to ensure uniform compression pressure, the roller should be normal to the tow path (Fig. 2.1). The roller should not rotate about the axis of the tow and should have a fixed orientation about the normal relative to the tow and its substrate, the Y-axis shown in Fig. 2.1.

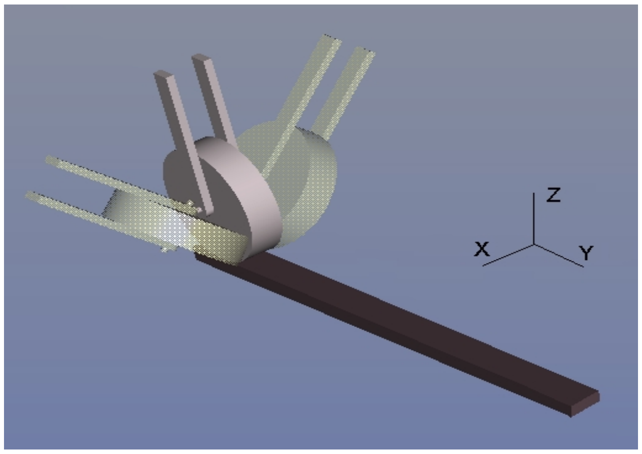


Figure 2.1 Rotation of the end-effector about the Y axis

For proper tow guidance on the predefined path, the roller also cannot divert from the desired trajectory (Fig. 2.2). The orientation of the roller about the Z-axis should follow the designed path to place accurately the prepreg in its predefined trajectory.

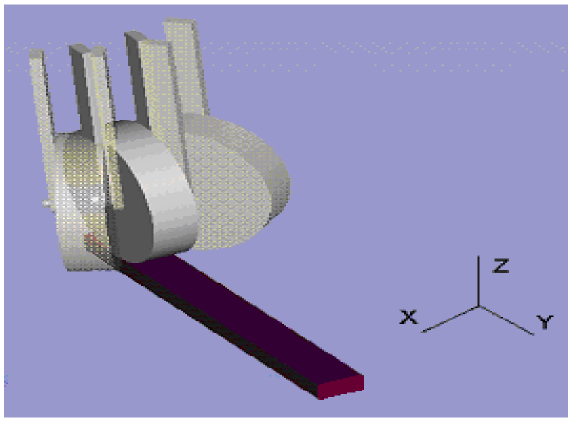


Figure 2.2 Rotation of the end-effector about the Z axis

The rotation of the end-effector about the third axis (the X-axis) is a design option (Fig. 2.3). According to the geometry of the roller, rotation of the end-effector about the X-axis is possible and very beneficial in some cases, but the problem will go back to the end-effector designer who will need to consider all the other components of the end-effector (*i.e.* the heat torch, the bearings, the air cylinder,…) which must accommodate a rotation about the X-axis.

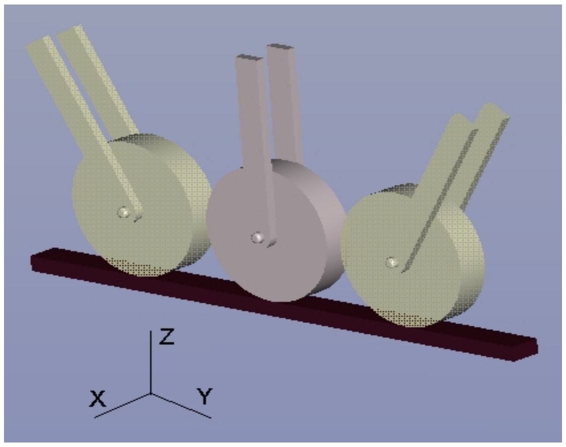


Figure 2.3 Rotation of the end-effector about the X axis

Briefly, the orientation of the compaction head should be relatively fixed in two axes and predefined according to the designed path of the towpreg. Rotational freedom to orient the roller in the third axis is also required to give wider possibilities for the path design.

2.4 Compressing Force on the End-Effector

As stated earlier, the end-effector head functions to compress the tow while placing it. A well-controlled force is needed to produce a uniform finished material structure. To prevent deterioration of the fibers, the compression should not exceed a predefined maximum threshold value. On the other hand, the material cannot be consolidated with low compaction pressure and, consequently, a force control mechanism should be included to supply the adequate compaction pressure.

CHAPTER 3: ROBOTIC DEXTERITY AND TRAJECTORY VERIFICATION REVIEW

To meet the kinematic requirements of the online consolidation fiber placement task, a full understanding and analysis of the manipulator workspace is necessary before planning the end-effector paths. The many kinematic restrictions of the tow layout task require a relatively high dexterous manipulator and accordingly, mapping the dexterous workspace of the robot is inevitable to ensure the feasibility of the task.

Placing the substrate in the appropriate location in the workcell has a strong influence on the dimensional and quality characteristic of the final product where tows are laid with no interruption on the entire pre-defined length of the substrate. Furthermore, the dexterous workspace analysis can define the exact limits on the dimensional characteristics of product geometries, since all tow trajectories are verified and checked to avoid all non–crossable virtual barriers within the reachable workspace of the manipulator. Geometrical design for the appropriate end-effector can be based on a study of the workspace and on the ability of the manipulator to achieve the fiber placement task. Feasibility analysis of predetermined products can specify and set design rules for the geometrical dimensions of the robot links and end-effector.

The following presents previous studies that focused on the analysis of the workspace. Methods were developed to determine boundaries for the reachable and dexterous regions within the robot workcell. Many formulations were developed to provide dexterity measures. Approaches were introduced to map the singularity surfaces that should be avoided in path planning. Other work focused on placing the end-effector path in the workspace and developing methods to verify the validity of predefined trajectories.

Sturges and Sainani reviewed several formulations that quantified robotic assembly capabilities and developed methods to measure manipulator dexterity [7]. The review covered relatively recent literature that studied the ability of a manipulator to control the end-effector. It is argued in [7] that the accuracy and repeatability of the robot definitely constitute significant criteria in assembly tasks, but it is stressed that the flexibility of the robot in positioning and orientating the end-effector is as crucial and necessary to achieve specific tasks. Many works introduced dexterity measures as a quantitative evaluation for the robot flexibility that can determine the feasibility of specific tasks. The kinematic dexterity measures were divided to (i) workspace based performance measures, (ii) Jacobian Matrix based measures, and (iii) other measures [7].

Workspace based performance measures (i) evaluate the kinematic extent over which the end-effector responds to all types of motion and reach all orientations. Sturges and Sainani mentioned the approaches that were studied to access robot motion capabilities by its workspace. The extreme reach of a hand was analyzed by many researchers [8-10] and many works studied the relations between the kinematic parameter with the dexterous space [11-13].

Jacobian Matrix measures (ii) are based on the evaluation of the determinant of the robot Jacobian. Yoshikawa [14] studied the determinant of non-square matrices for redundant manipulators. The square root of the determinant of the JJ^T (J being the Jacobian matrix) determined the manipulability measure. A geometric visualization of the manipulability measure was developed in the form of a manipulability ellipsoid. Larger ellipsoid volumes indicate higher manipulability. Singularity configurations reduce the ellipsoid volume to zero. Using the manipulability measure, the best postures for different types of robots are shown and discussed [15]. Sturges and Sainani [7] reviewed many extensions to Yoshikawa manipulability measure and listed many areas where the manipulability measure was applied [16-20]. Several researchers [21,22] studied the condition number of the Jacobian matrix as a measure of the proximity of the robot configuration to a singularity. Again, studies applied the condition number for many different purposes. The Global conditioning index used the condition number of the Jacobian over the entire workspace for optimization [23].

Sturges and Sainani reviewed the work done to relate the task physics to the Jacobian based measures and listed many studies that focus on specific tasks conditions [24,25]. The vortex theory (iii) was used to define manipulability measure applied primary for the optimal pre-shape of robot hand [26]. Vortices are created by the hand curling fingers before contact.

The recent work of Snyman *et al.* [27] focused on determining the boundaries of manipulator workspaces; an optimizing algorithm was developed for planar manipulators that can be easily extended to any general spatial manipulator: an automated numerical method was introduced [27] to draw the boundary of the workspace, based on an optimizing approach for mapping any manipulator workspace; a ‘radiating’ point is first selected in the output space by using mean values for the joint values. The workspace boundary ∂A is then numerically mapped by solving optimizing equations, for successive rays emanating from the radiating point. Each ray should have one boundary point that satisfies the problem kinematic constraints. The point is selected so that the distance from the radiating point to the candidate boundary points is maximized. This approach was applied to a planar serial manipulator and the results matched the workspace boundary obtained by simple geometrical construction [27]. For more general manipulator geometries, the problem of handling holes in the workspace and the non-convexity of the boundary can be solved by a judicious positioning of the radiating point, but still, this requires a good and delicate knowledge of the manipulator geometry.

Carretero *et al*. [28] define the dexterous workspace of a robot as a subset of the reachable workspace where a specific dexterity measure is met, *i.e.* predetermined limits on the Jacobian can keep the robot away from singularities and thus define the dexterous workspace of the robot. To map the dexterous workspace on a three-dimensional graph, a Matlab algorithm evaluates the dexterity of the manipulator at discrete points. The workspace is sliced to horizontal planes distanced by a chosen value. On each plane, the search of the boundary is performed radially. The boundary is set when distant points fail to satisfy the prescribed dexterity values [28]. Smaller resolutions are used on pre-specified planes where singularity configurations are likely to occur. Carretero *et al*. [28] also studied the variations in the dexterous workspace boundaries when the manipulator geometrical dimensions are varied. Voids in the workspace are not handled by the algorithm logic.

In a recent work, Abdel-Malek and Yeh [29] identified singular surfaces and curves in robots workspace envelope. They determined the crossability of a singular surface based on the rank deficiency of the Jacobian. Acceleration analysis allowed the determination of the direction of admissible movements of the end-effector on singular surfaces or curves.

The problem of avoiding the interruption of a planed path is discussed in [30]. A mathematical formulation verifies if a singular robot configuration occurs along the path and accordingly, allows the selection of another initial configuration that guaranties the completion of the path with no interruption. Different robot configurations correspond to the multiple inverse kinematics solutions. For every configuration, the correspondent inverse kinematics solution is checked for intersection with singular surfaces along the end-effector path. The path validation logic checks the joint angles against their mechanical limits and verifies the intersection of the path with a singular surface. Many other robot restrictions are not considered in the analysis imposing the assumptions of no physical obstacles in the workcell [30].

Chaney and Davidson [31] introduced a geometrical method to place the workpiece in the workspace of RPR planar robots. The approach does not treat the obstacle avoidance but it traces the position of the workpiece in the workcell and determines its orientation to ensure the feasibility of the task with pre-defined geometrical parameters.

Soman and Davidson [32] explained the differences between path planning (or finding) versus path placing. They discussed that most of the studies focused on path planning: the problem of finding an acceptable path that joins the initial state to the final state. Little literature considered the problem of path placing [32]. Instead of choosing one acceptable path out of many possibilities, the problem of path placing is to locate the path in the workcell. The form of the path is fixed and there is only one possibility to choose the path. The problem is reduced to just locating or shifting the path in the workcell. The fiber placement problem similarly involves tracing paths, but requires large numbers of these placements to be feasible simultaneously for a fixed position workpiece and a single end-effector configuration.

Nelson *et al*. located the workpiece to maximize the manipulability of a six-jointed robot with a computational optimization routine. Soman and Davidson [32] developed a formulation to place the workpiece in the workspace of planar 3-R robots: the pose of any point in the workspace of the planar 3-R is characterized by two Cartesian coordinates and the angle of orientation. Accordingly, the feasible paths or locations of the workpiece are shown with a three-dimensional configuration space. Soman and Davidson decomposed the three-dimensional abstract workspace to two different spaces. One is two-dimensional holding the two Cartesian

coordinates and the second is one-dimensional showing the orientation angle of the workpiece [32]. Kinematic inversion is applied by fixing the workpiece (or the path) and thus, the acceptable design regions are characterized by the position and orientation of the base of the robot with respect to the workpiece.
- The (geometrically constructed) two-dimensional space contains all the variables needed to decide on the feasibility of the task and to show the possible or acceptable regions where the workpiece can be placed. In the case where several different trajectories are part of one complete task, valid areas for each path are drawn, thus, the global task feasibility and path location are based on the intersection of all areas. When the intersection is null, the total desired tool path cannot be traced [32].
- The one-dimensional space shows all available orientation angles of the workpiece only if the two-dimensional space proved the feasibility of the task.

As mentioned earlier, the above approach requires geometrical construction for the two spaces, and consequently, expanding this method to involve a manipulator with more degrees of freedom is unfeasible because of the induced complexities in the geometrical construction.

Merlet [33] developed an algorithm to verify if specific straight-line paths lie fully inside the workspace of a 6-dof parallel manipulator. The line is discretized to points distanced by a chosen value. The end-effector can have fixed orientation angles over the entire trajectory, or they may vary from their initial state to the goal (final) state. In the latter case, the orientation of the end-effector is changed gradually on the elementary points [33].

Recent studies of Merlet [34] extended his previous work to focus on determining the validity (or feasibility) of any path of the end-effector. The method was applied to parallel manipulators only. According to [34], the path validity requirements are met when:
- The path lies in the reachable space.
- The robot is dexterous over the entire trajectory.

To avoid crossing singularities, the manipulability index value is computed at desired points and then checked with a chosen minimal threshold value. The following describes the algorithm logic for the constraints verification:

The upper and lower bounds of the constraints quantities - the joints and the manipulability index - on a chosen interval (on the planned path) are stored and compared to the pre-defined limits (or threshold). If one of the bounds exceeds the constraints limits, the path is considered unfeasible. On the other hand, if the upper and lower bounds are within the acceptable range, the bisection method subdivides the path to two smaller intervals that will be processed by repeating the above logic. Consequently, the bisection method imposes a finer resolution every time the initial interval is considered valid [34]. The same analysis was extended from one-dimensional path verification to the feasibility of surfaces (two-dimensional). Merlet [34] considered also uncertainties in the trajectories. He argued that model errors and practice control might slightly shift (or modify) the real trajectory from the specified one. He consequently, introduced an error range for the coordinates of the end-effector. Again, he used the same algorithm to check the validity of all paths considering all possible errors occurring in the real trajectory.

Many studies have analyzed the workspace of manipulators and developed methods and formulations to evaluate robot dexterity and kinematic capabilities. However, none of these previous works is suitable for the fiber placement task where multiple towpreg layouts should be verified in a high degree of freedom robot workcell. The following chapters present the approach developed within this work to address the same problem of evaluating robot kinematic capabilities. Some of the concepts resemble to some extent ideas reviewed in the above literature, and on the other hand, completely new methods are introduced to improve and solve new problems. The method is applied to the robotized online consolidation fiber placement task.

CHAPTER 4: ROBOT RESTRICTIONS IN FIBER PLACEMENT

The kinematic requirements of the fiber placement task set several restrictions on the capabilities of the robot and on the dynamic limits of its joints.

4.1 The Reachable Workspace of the Robot

The reachable workspace of the robot has a strong influence on the feasibility of the fiber placement task. As the compaction head should be in contact with the structure surface, the whole volume of the product should lie within the reachable workspace of the robot. The robot workspace depends mainly on two factors:

- The dimensions of the robot links and end-effector: The reachable workspace of the contact point is greatly increased when large robots are used or when large end-effectors are mounted on the hand of the robot.
- The joint hardware limits on the joints, on the other hand, can dramatically reduce the reachable workspace of the compaction head contact point. By limiting the motion extents of every joint, the workspace is decreased and non-crossable surfaces are generated within the workspace to even reduce the motion of the end-effector with specific joint angle configurations.

Briefly, the fabrication of large products depends on the size of the reachable workspace. and the workpiece location in the workcell should be checked with the hardware limits of the robot joints.

4.2 High Degree of Freedom Robots

As discussed earlier, the robotized compaction head should have the complete freedom to reach all points on the structure surface, with orientation angles specified and predefined by the surface patterns. Accordingly, the feasibility of fabricating complex shaped products depends mainly on the ability of the robot to locate its end-effector contact roller on any point in the three-dimensional volume enclosed by the structure. Thus, the robot should have at least three degrees of freedom moving the compaction head to all required contact locations.

Furthermore, since the surface patterns and orientations predefine specific orientation angles for the contact roller, the robot should also have the freedom to rotate or orient its end-effector around all three-dimensional axes to achieve the desired tow path angles. Consequently, three degrees of freedom should be added to meet the new requirements on the contact point motions and to provide its rotation around the three orientational axes.

By combining all the constraints on the pose of the end-effector, the robot should have at least six degrees of freedom necessary for the fabrication of any complex product with any desired patterns. Redundant robots (more than 6-dof) provide a wider range of possible robot configurations to perform the fiber placement task and that can definitely allows more optimizations on the feasible solutions to fabricate the product.

4.3 The Dynamic Limits on the Robot Joints

By specifying the layout path for every prepreg tape, the end-effector contact points are predefined and its compaction head trajectory is set. These paths might intersect with surfaces within the workspace where the robot can loose a degree of freedom, the maximum joint capabilities are exceeded and the motion of the robot is stopped. Furthermore, when the end-effector trajectory just passes close to those singularity surfaces without even intersecting them, the joint's kinematic performances increase dramatically and might again exceed their limits and interrupt the end-effector path.

- As discussed earlier, the adequate constant velocity of the end-effector allows the uniform heating and the appropriate consolidation of the towpreg during its layout. The joints actuators velocities are directly proportional to the velocity of the end-effector, and accordingly, for a specific chosen end-effector velocity, the actuators velocities should be checked with their maximum limits to ensure the feasibility and the continuity of every towpath.
- Similarly, the joint's torques should provide the desired end-effector compression forces necessary for consolidation. In addition, the joint's torques should be able to support the weight of the compaction head attached to the robot hand. Again, since the joint torques are proportional to the total forces on the end-effector, the joint torques should be checked with their maximum limits to ensure the uninterrupted towpreg layouts.

4.4 Accuracy and Repeatability of the Robot

The automation of fiber placement definitely requires high accuracy and repeatability to ensure good quality and consistency for the final products; placing and orienting the towpreg accurately on its designed path prevent the formation of voids in the resin and consequently, provide better consolidation of the final product.

For the fabrication of composite structures, all the above robotic constraints should be met on every towpath on the surface of the structure. A product is completely feasible if the appropriate robot manipulates the consolidation head without exceeding all kinematic restrictions.

CHAPTER 5: THE EFFECTIVE WORKSPACE FOR FIBER PLACEMENT

The method introduced here to map the workspace of the robot for fiber placement is totally based on the *feasibility verification* of every towpreg layout and consolidation. As previously stated, the fabrication of composite material in the fiber placement process involves laying every single towpreg on its predefined pattern and thus, stacking the prepreg tapes to achieve the designed volume of the structure. Consequently, the manufacture of structures with specific surfaces, patterns and dimensions is completely valid only if the feasibility of laying all its constituents towpregs is checked.

As discussed earlier, online consolidation fiber placement imposes many kinematic requirements and restrictions on the compaction head trajectory. In addition, this robotized fabrication process is also subject to several constraints that can limit the freedom of the manipulator to successfully place all towpregs on their predefined locations and orientations. Consequently, to verify the feasibility of placing *one* towpreg on its predefined (and required) trajectory, all the *several robot constraints should be satisfied* along the whole path. By repeating this verification for all towpreg paths on the structure surface, the fabrication of the final product could also be considered valid.

Since the fabrication verification of a structure requires the validation of its entire towpregs layout, the workspace of the online consolidation fiber placement can be formed and mapped by drawing all feasible towpreg paths. This workspace determines accurately the feasibility of a structure by checking the layout of all its towpregs. Problems arise here since the towpreg layout paths can have an unlimited number of different shapes, lengths, curvatures and orientations. The ability of a robot to manufacture many combinations of complex shaped structures definitely implies a feasibility check for an infinite set of towpreg curves. Therefore, it is practically impossible to map a workspace that includes all feasible towpreg paths. One possible solution for this problem is to only consider straight-line towpreg paths. Mapping only feasible straight lines has many advantages and benefits to represent the workspace of the robot for fiber placement:

- One of the important features of only considering straight lines is the simplicity of drawing and analyzing the feasible workspace envelope: by choosing the desired two-dimensional plane, the boundaries of the workspace envelope are determined by locating the end points of all feasible towpreg lines within the envelope. This workspace is easily analyzed, and the feasible towpreg path segments in the envelope are visually clear and easily used to verify the fabrication feasibility of specific structures.

- The feasible workspace method can include towpreg lines with many different orientations or inclinations, *i.e.* the layout of horizontal, vertical or inclined towpreg lines can be verified, and the envelope workspace for each of these parameters can be mapped. The next chapters discuss this topic in more detail.

- The ability to draw the feasible workspace of straight lines of all inclinations provides insight on the feasibility of any complex three-dimensional towpreg path. Many complicated curves are or can be simplified to a series of straight lines with different orientations (Fig.5.1-a and b). In the two cases, the feasibility of the whole curved path requires the validation of each line segment forming the trajectory. Consequently, the feasible workspace built with straight lines can verify the potential of fabricating many complex shaped structures with three-dimensional curved tow path patterns.

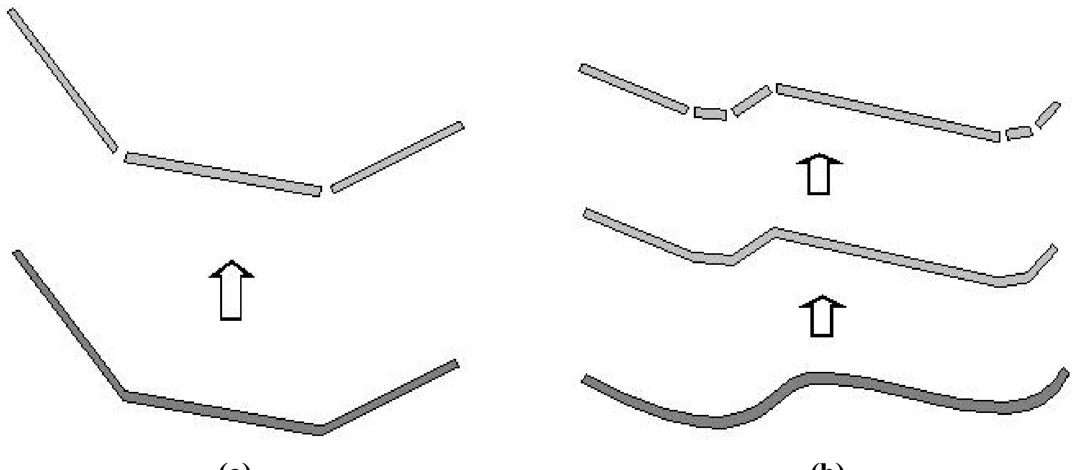


Figure 5.1 Breaking curved tow paths to a series of verifiable straight lines

- The straight lines workspace has a more direct application on deciding on the feasibility of structures; the fabrication of bodies formed by straight segment tows is easily verified by just checking the structure constituents' tow segments with the robot feasible workspace. If all segments lie within the envelope, the manufacture of the complete structure is determined to be valid. Flat coupons, boxes, cubes can be included in this structure category.

Figure 5.2 shows a flowchart that summarizes the procedure to verify the feasibility of fabricating composite structures. Two-dimensional effective workspaces check the validity of each towpreg on the structure surface, and accordingly decide on the feasibility to manufacturing the whole final product.

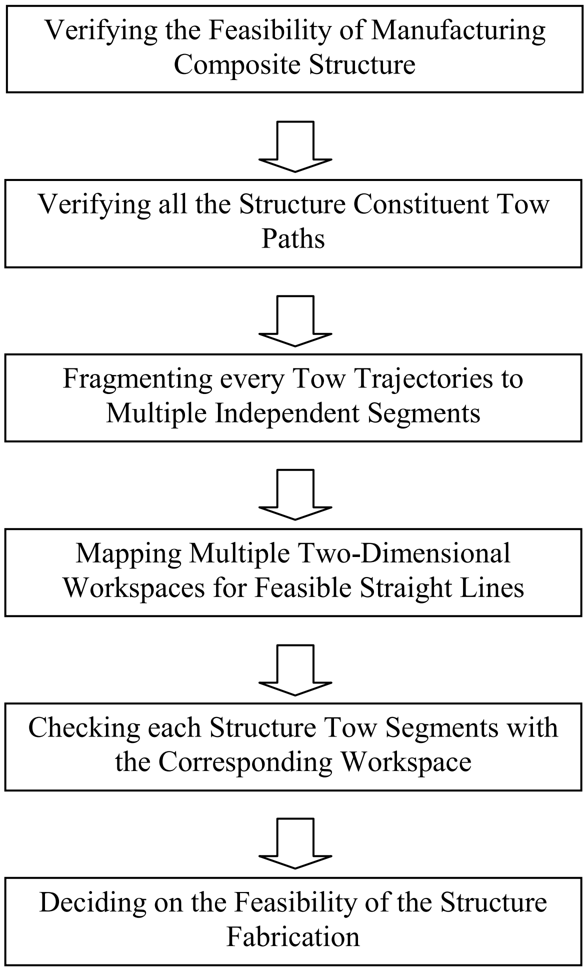

Verifying the Feasibility of Manufacturing Composite Structure
Verifying all the Structure Constituent Tow Paths
Fragmenting every Tow Trajectories to Multiple Independent Segments
Mapping Multiple Two-Dimensional Workspaces for Feasible Straight Lines
Checking each Structure Tow Segments with the Corresponding Workspace
Deciding on the Feasibility of the Structure Fabrication

Figure 5.2 The approach to verify structure fabrication using the feasible workspace

In the case where the feasibility of the desired composite structure is verified, the above method determines the specific locations and orientations of the workpiece in the workcell, which are definitely necessary for the fabrication of the structure. A detailed explanation of this concept is presented in the following chapters. Since many locations and orientations would often allow the manufacture of the composite body, the effective workspace method gives the process and product designer the widest set of practical choices with no focus on optimization.

The effective workspace approach can be easily combined with an optimizing technique, easily implemented in the feasible workspace logic; but optimization is neglected here to stress on all the feasible locations for the final structure in the robot workcell.

CHAPTER 6: THE MERLIN ROBOT

The Merlin robot (Fig. 6.1) is a reliable six-dof manipulator capable of performing a wide variety of tasks. Its unique mechanical design and work envelope allowed its presence in many manufacturing scenes ranging from very accurate and precise applications to three-dimensional and rugged tasks.

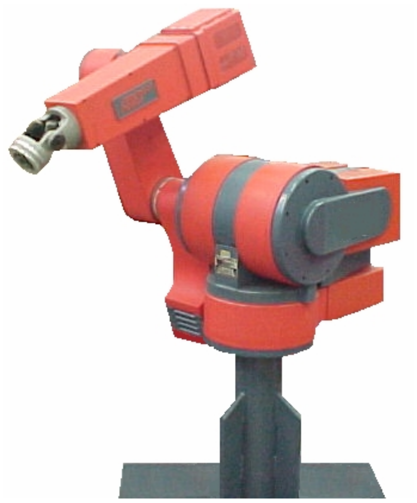
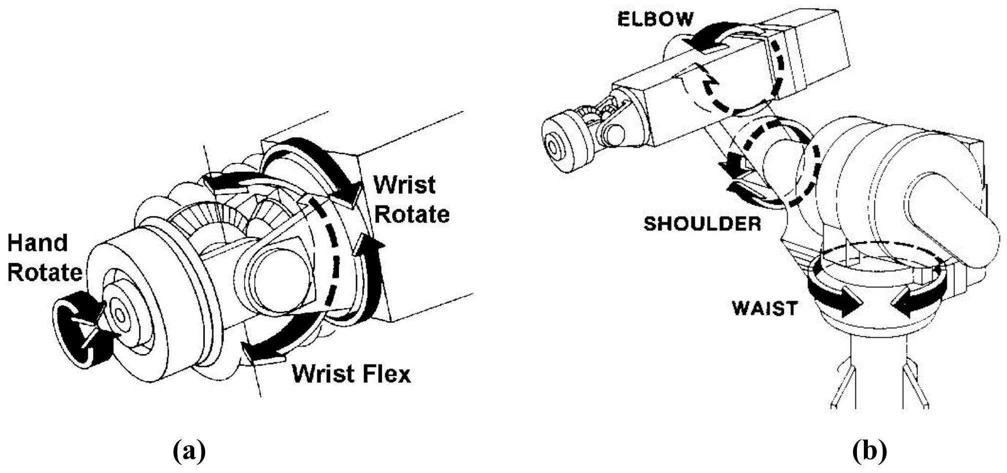


Figure 6.1 The Merlin robot

As required for the fiber placement task, the Merlin robot is capable of moving and rotating the compaction head in all three-dimensional positions and orientations. The robot has six joints providing the six-dof needed for the application. The six joints are categorized to two groups, positional and orientational, shown in Fig. 6.2 and listed in Table 6.1 [35]:



(a) **(b)**
Figure 6.2 The Merlin robot six joints

Positional Joints		Orientational Joints	
Joint Number	Joint Name	Joint Number	Joint Name
1	*Waist*	4	*Wrist Flex*
2	*Shoulder*	5	*Wrist Flex*
3	*Elbow*	6	*Hand Rotate*

Table 6.1 The Merlin robot positional and orientational joints [35]

The reachable workspace of the Merlin robot is a spherical volume centered on the shoulder joint axis on the robot. Vertical and horizontal work envelope views are shown in Appendix A. In the fiber placement task, the compaction head attached to the faceplate of the robot considerably increases the reachable workspace of the contact point, depending mainly on the size and orientation of the end-effector. As discussed in the robot restrictions section, the workspace is reduced by the axes' mechanical stops that limit the angular spans of most of the Merlin robot joints. The rotational extent for every joint is listed in Appendix B.

Six stepper motors drive the Merlin robot, and six encoders mounted on the back of each of the motors provide positional feedback with 1/2000-revolution steps. High gearing transmits the rotational power form the motors to the joint axes and greatly magnifies the resolution of the joints. Since the feedback control system stops each motor in 2000 discrete positions, the gear ratio of every joint translates and multiplies this resolution by the corresponding gear ratio. As a result, the Merlin robot is able to position its compaction head contact point to within ±0.001 inch of a previously defined point, and offers a high resolution critically required for fiber placement applications [35].

However, many kinematic and operating conditions must be met to achieve this high repeatability. The load on the end-effector should not be changed, and the exact same path should be followed to reach the target point. In addition, the ambient temperature cannot vary significantly, and the system should be warmed up to allow a stable operating temperature [35]. These operating conditions are easily met in the fiber placement task to offer the highest repeatability for the towpreg layout.

As discussed previously, one of the robots' constraints involve the dynamic limits on the joints. For a standard 20-pounds load, the Merlin's axes are limited by the motors speed-torque specifications. The joints gear ratio magnifies the axes torque limit at the expense of dramatically reducing the joints angular velocity limit. Table 6.2 and Table 6.3 list the motors specifications and compute the maximum limits on the joints.

Joint	**Gear Ratio**	**Motor Maximum Velocity (rev/sec)**	**Joint Maximum Velocity (rad/sec)**
1 Waist	**48:1**	16	2.π.16/**48**
2 Shoulder	**48:1**	16	2.π.16/**48**
3 Elbow	**48:1**	16	2.π.16/**48**
4 Wrist Roll	**24:1**	16	2.π.16/**24**
5 Wrist Flex	**20:1**	16	2.π.16/**20**
6 Hand Rotate	**24:1**	16	2.π.16/**24**

Table 6.2 The Merlin joints velocity limits

Joint	**Gear Ratio**	**Motor Maximum Torque (oz.in)**	**Joint Maximum Torque (lb.in)**
1 Waist	**48:1**	1125	1125.**48**/16
2 Shoulder	**48:1**	1125	1125.**48**/16
3 Elbow	**48:1**	1125	1125.**48**/16
4 Wrist Roll	**24:1**	400	400.**24**/16
5 Wrist Flex	**20:1**	400	400.**20**/16
6 Hand Rotate	**24:1**	400	400.**24**/16

Table 6.3 The Merlin joints torque limits

In laying every prepreg tape, the joints' velocity and torque values cannot exceed their corresponding maximum limits. Therefore the feasibility of the towpath should be checked with the Merlin joints restriction limits.

6.1 The Merlin Robot Representation

Robots can be represented schematically as a chain of rigid bodies or links connected by joints that provide the mobility of the manipulator [36]. A stationary base constitutes one end of the chain whereas end-effectors are mounted on the other end where the targeted motion is achieved.

Denavit-Hartenberg convention is a standard approach to describe the forward kinematics of the end-effector with respect to the base. It allows a kinematic relation between two consecutive joints and recursively provides the overall kinematic description of the end-effector motion with respect to the base [36]. This systematic approach sets rules for defining and locating frames on each link on the manipulator in order to develop methodical transformations between two consecutive frames. Each transformation describes the location and orientation of one frame with respect to the other, and recursively, Denavit-Hartenberg convention allows the construction of the overall direct transformation matrix composed by all individual coordinates' transformations. The final forward matrix expresses the position and orientation of the end-effector with respect to the base frame.

The Denavit-Hartenberg convention is applied to the Merlin robot and the frames are assigned on the links. Figure 6.3 shows the position and orientation of all the assigned frames on the Merlin robot [37].

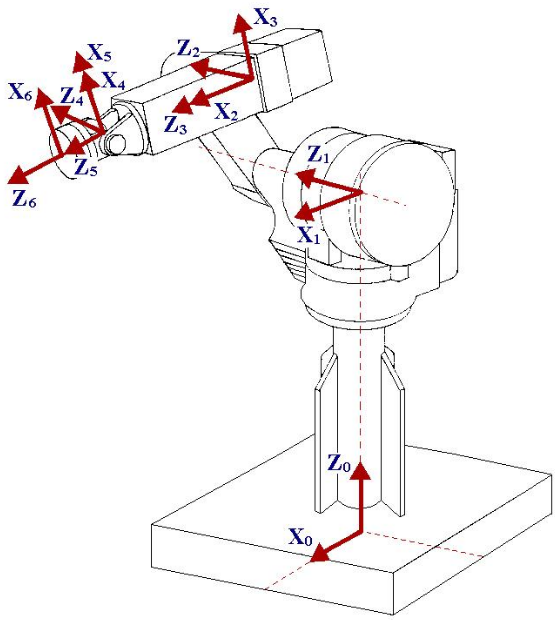


Figure 6.3 The Denavit-Hartenberg frames on the Merlin robot

Once the link frames are established on the links, four parameters specify the transformation matrix between two frames. Appendix C presents the Denavit-Hartenberg frames and corresponding parameters. The resulting basic transformation is a function of the four parameters and has the following form [36]:

$$A_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

The basic transformation matrices that relate each consecutive frames on the Merlin robot are given by:

$$A_1^0 = \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2\sin(\theta_2) \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^2 = \begin{bmatrix} \cos(\theta_3+\frac{\pi}{2}) & 0 & \sin(\theta_3+\frac{\pi}{2}) & 0 \\ \sin(\theta_3+\frac{\pi}{2}) & 0 & -\cos(\theta_3+\frac{\pi}{2}) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4^3 = \begin{bmatrix} \cos(\theta_4) & 0 & -\sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

$$A_5^4 = \begin{bmatrix} \cos(\theta_5) & 0 & \sin(\theta_5) & 0 \\ \sin(\theta_5) & 0 & -\cos(\theta_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_6^5 = \begin{bmatrix} \cos(\theta_6) & -\sin(\theta_6) & 0 & 0 \\ \sin(\theta_6) & \cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Products of many consecutive combinations of these (4 x 4) matrices can give the location and orientation of any chosen frame with respect to any desired frame on the Merlin robot.

$$T_j^i = A_{i+1}^i A_{i+2}^{i+1} \ldots A_{j-1}^{j-2} A_j^{j-1} \tag{6.3}$$

In particular, the direct transformation that gives the pose of any frame *n* with respect to the base frame can be expressed as [36]:

$$T_n^0 = A_1^0 A_2^1 \ldots A_{n-1}^{n-2} A_n^{n-1} \tag{6.4}$$

Additionally, the Jacobian matrix of the Merlin robot configuration is easily computed form the direct transformation matrices.

CHAPTER 7: THE THREE-LINK PLANAR ARM

The preliminary analysis for the feasible workspace problem involves reducing the global three-dimensional space to just considering the feasible towpath in the *vertical plane*. The analysis is further simplified by restricting the complicated Merlin robot configuration to a three-link planar arm. These preliminary assumptions facilitate the complexity of three-dimensional kinematics and provide better visual understanding and verification of the analysis. These steps are considered necessary before handling the total three-dimensional application.

Many previous studies decomposed the manipulator workspace to a combination of two-dimensional planes able to provide and define solution for the global problem [32]. In the fiber placement application, the vertical plane 'cuts' the fabricated product and covers the cross-sectional area of the overall structure volume to built and traversed by the compaction head.

By freezing joints 1, 4 and 6 (Fig. 7.1), the Merlin robot is reduced to a simple three-link planar arm:

- The angular velocity of joints 1, 4 and 6 are set to null.
- Joint 4 angle should be set to zero to restrict the motion in just one vertical plane; *i.e.*, joints 2, 3 and 5 axes are parallel only for a null value of joint 4 angle.
- Joints 1 and 6 angles should be given any *fixed* value and may be set to zero for simplicity.

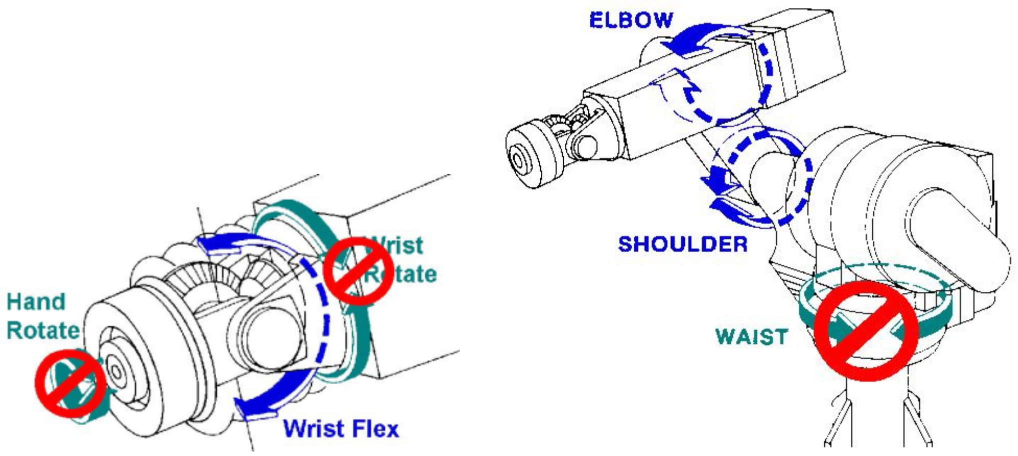


Figure 7.1 Freezing joints 1, 4 and 6

By meeting those three conditions, the trajectories of the assumed three-link planar arm are limited in the vertical plane passing through the joints and the links. Figure 7.2 illustrates a schematic view of the three-link planar arm and the XZ vertical plane.

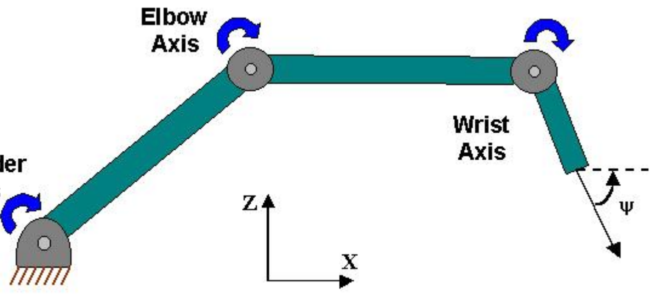


Figure 7.2 The three-link planar arm assumption

In reducing the Merlin robot to the three-link arm, the compaction head trajectories are restricted to just one plane supplying two degrees of freedom. Furthermore, the manipulator is capable of orienting the end-effector about one axis, and consequently, the three-link arm is then able to move and orient the tool-tip in the XZ plane, providing a total of three degrees of freedom. This is a natural result of fixing three joints on the six-degree of freedom Merlin robot and freeing the three remaining for the link planar analysis.

By reducing the workspace to the vertical plane, the compaction head is mounted on the last link such as its roller contact point also lies in the vertical plane. The end-effector considered in this two-dimensional analysis example is 10-inches long along the longitudinal axis of the last link.

7.1 Techniques to Map the Feasible Workspace of the Three-Link Planar Arm

Many distinct methodologies are developed to solve the feasible workspace problem with the three-link planar manipulator. These techniques match the simplification introduced by considering the three-link planar arm and are necessary to fit the established two-dimensional assumptions.

7.1.1 Line Parameters in the Vertical Plane

Since towpaths are fragmented into sets of straight lines, segments of many different parameters would be required to be laid on the body in order to achieve the final desired structure. Accordingly, the introduced analysis should be capable of handling all combinations of lines location and orientation parameters to ensure the feasibility study of all these required segments.

A The Inclination of the Line

The vertical workspace includes lines with different orientations often necessary for the fabrication of structures that require specific inclinations for the towpreg segments. The analysis of the *inclination* of the generated segments can be used to optimize the two-dimensional workspace problem in two separate ways:

Inclining the whole product during the stacking process might improve the dimensional characteristics of some products. Larger structures can be fabricated when the whole body is manufactured with a certain inclination. This procedure will not change the pattern on the product, since the whole structure is inclined (whereas the robot base is stationary in Fig. 7.3), but it might allow smother towpreg layouts or even larger structures. Thus, the analysis should focus on varying the inclination parameter of the segments to cover all manufacturing possibilities.

In addition, most product structures are fabricated using tows with different inclinations. Stacking layers with different orientation patterns is required for the manufacture of some complex shaped bodies (Fig. 7.4). Thus the need to generate lines with different inclinations is inevitable and this orientation variation in the analysis can be controlled by the *inclination* parameter variable.

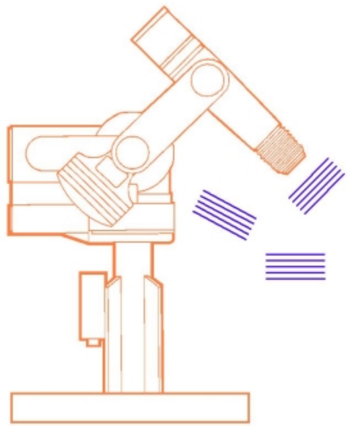


Figure 7.3 Exactly the same product can be produced but in different locations or orientations. Tow alignment is shown by parallel line segments

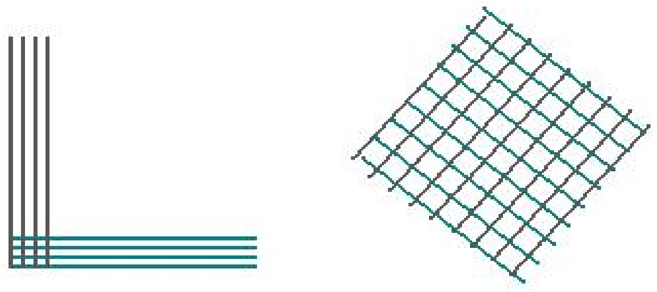


Figure 7.4 Structures that require different inclination angles in the vertical plane

B The elevation of the line

The *elevation* of the towpreg segments is a variable parameter to be considered during the lay up process. Similarly to the inclination parameter, different elevations of the product in the vertical plane affects the maximum dimension of the final structure and varies the ease of fabricating exactly the same product. Thus, the elevation parameter controls the tow layouts at different elevations for varied locations of the workpiece in the vertical plane.

In addition, every single tow (or segment) is stacked at close but different elevation to *build* the cross-sectional area of the product; consequently analyzing lines with different elevations is needed to determine the feasibility of building the desired shapes.

In the case where the inclination of the towpreg segments is ± 90 degrees, the lines are vertical, and the elevation parameter is substituted by the *X-offset* parameter, the distance of the line from the stationary joint 2 along the X-axis.

C The orientation of the end-effector with respect to the line

As discussed earlier, in the two-dimensional analysis, the *orientation of the end-effector* is only controllable around one axis. The two other axes cannot physically vary. Consequently, the orientation would be another parameter that can be specified and set to a fixed value for every line.

The workspace analysis involving the orientation θ of the end-effector with respect to the line is important since it may be the primary factor that would impact the design of the end-effector.

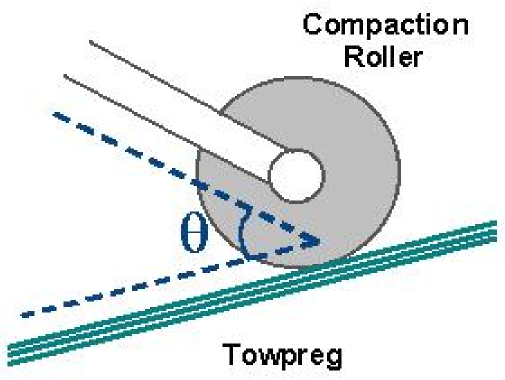


Figure 7.5 The orientation of the end-effector with respect to the towpreg segment

Additionally, the orientation of the end-effector (θ in Fig. 7.5) influences the feasible workspace by affecting, primarily, the feasibility of the individual towpreg trajectories. By maintaining fixed specific values for the orientation of the end-effector along the towpreg segment, the hardware joint limits of the manipulator can dramatically limit the feasible length of the tow segment, and interrupt the continuous layout of the towpreg. Consequently, setting this orientation as one of the line parameters enables a complete study that analyses the orientation of the end-effector with respect to the line.

Not only does the orientation of the end-effector play a role in defining the limits of the workspace (similarly to the two above parameters), the orientation has a direct effect on the compression pressure values during tow placement. This observation will be discussed in detail in the following sections.

D The elbow-up/elbow-down modes

The elbow mode is the other parameter to be controlled before generating a specific line. Even if we set the same desired inclination, elevation and orientation parameters for any line in the vertical plane, significant differences in workspace boundaries occur when the elbow up/down modes are alternated. The elbow mode greatly affects link collisions with the generated line and thus the ‘true’ boundaries of feasible towpreg segments.

Briefly, every line in the vertical plane of the three-link manipulator should be predefined with the above four parameters. This method allows us to first define and characterize every line and secondly, to study the effects of each of those variables on the feasible workspace boundaries.

7.1.2 Forward Kinematics for the Three-Link Planar Manipulator

To determine the robot configurations and capabilities in moving the compaction head on the previously discussed lines, a kinematical relation should be developed to give the angles of the manipulator for the end-effector line paths. However, the linear segments should be discretized to a finite number of points that accordingly, can provide a series of discretized set of joint angles required to move the compaction head on the desired linear path.

Two different methodologies can yield the relation of the manipulator joint angle with the points on the linear paths. Knowing that the *inverse kinematics* technique is a very robust approach to solve this problem, *forward kinematics* is used in the analysis of the simplified three-link manipulator whereas inverse kinematics is applied in the three-dimensional Merlin robot study. Forward kinematics generates the position and orientation of the end-effector for a given set of joints angle. The simple geometrical configuration of the three-link manipulator allows the use of this technique to determine points on the linear paths in the vertical plane.

As already stated, every line in the vertical plane has four parameters set to the desired values. By geometrically enclosing the polygon formed by the three links and the towpreg line (the red line in Fig. 7.6), a relation between the three joint angles can be determined to insure the motion of the end-effector tool-tip on the linear path and to fix the orientation angle of the end-effector to its desired value.

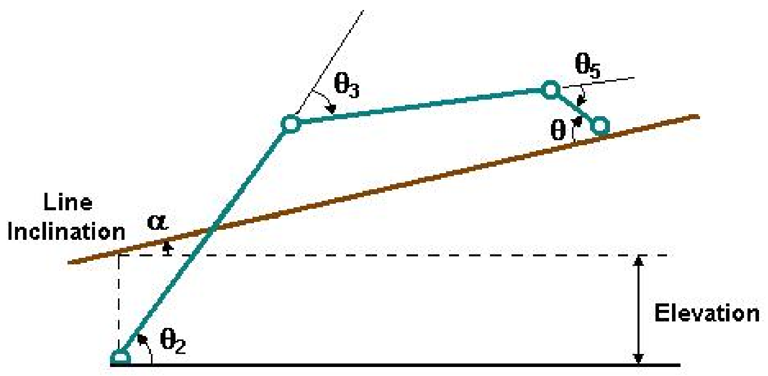


Figure 7.6 Determining the relation between the three joint angles in the forward kinematic method

The equations resulting from the geometrical analysis provide sets of the three joint angles:

$$\theta_3 = \alpha - \theta_2 - \sin^{-1}\left[\frac{L_1.\sin(\theta_2 - \alpha) - elevation.\cos(\alpha) - L_3.\sin(\theta)}{L_2}\right] \tag{7.1}$$

$$\theta_5 = \alpha - \theta - \theta_2 - \theta_3 \tag{7.2}$$

where α is the inclination of the line

θ is the orientation of the end-effector with respect to the towpreg line

L_1, L_2 are respectively the lengths of the links 1 and 2

L_3 is the total length of link 3 and the mounted compaction head

The above equations provide a set of three angles $\{\theta_2, \theta_3, \theta_5\}$that locates the compaction head on the linear towpreg line satisfying all the kinematic restrictions. In analyzing linear paths with the forward kinematics technique, the angle θ_2 is given a series of gradually incremented (or decremented) values within its feasible span limits, and accordingly, the two joints angles θ_3 and θ_5 are computed using equations (7.1) and (7.2) for the tow layout on the desired linear path. To control the elbow mode parameter, the farthest reachable point on the line is first located and the corresponding joint angle θ_2 is determined. Then, the choice of incrementing or decrementing angle θ_2 sets respectively the Elbow mode parameter to *elbow up* or *down*.

7.1.3 The Compaction Head Velocity and Exterior Forces

The adequate velocity *V* and compression force F_c on the contact point are chosen by the manufacture designer to allow the proper consolidation of the structure towpregs. However, for different line parameters, the velocity and the compaction force have different components with respect to the base XZ frame in the vertical plane. Consequently, for a complete kinematic representation, the velocity and force components are expressed in terms of the line inclination, being the only parameter affecting their orientation with respect to the base frame.

As shown in equations 7.3 and 7.4 and in Fig. 7.7, the velocity vector is parallel to the inclined line whereas the compaction force vector should always be normal to the towpreg line for proper compression. The three other line parameters have no influence on the velocity or compaction force orientation.

$$V_X = V\cos(\alpha) \qquad V_Z = V\sin(\alpha) \qquad (7.3)$$

$$F_X = F_c\sin(\alpha) \qquad F_Z = F_c\cos(\alpha) \qquad (7.4)$$

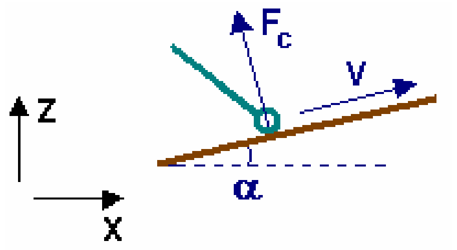


Figure 7.7 The Velocity and Force Vectors

Furthermore, a correlation can be established between the *orientation of the end-effector* θ and the *compaction pressure* on the hydraulic cylinder mounted on the end-effector; to ensure the proper control of the compressing force, the air cylinder is subject to a specific pressure that provides the required force for adequate compaction.

$$P = \frac{F_p}{A} = \frac{F_c}{A.\sin(\theta)} \qquad (7.5)$$

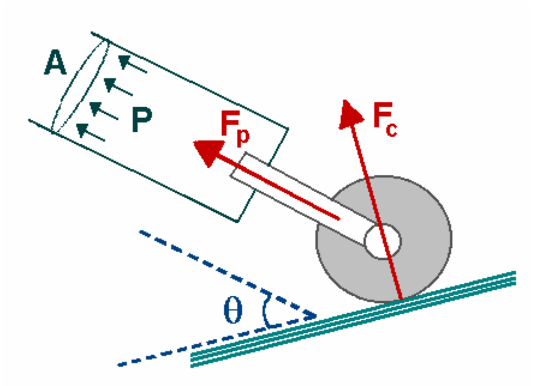


Figure 7.8 The hydraulic cylinder pressure providing the controlled compression force

According to equation 7.5, since the pressure subjected on the hydraulic cylinder is inversely proportional to the cosine of θ, the value of the pressure can dramatically increase and exceed its limit when the orientation angle θ is reduced to small values. Consequently, choosing values for the angle θ should be carefully considered as one of the design rules for the fiber placement task.

7.1.4 Collision Detection for the Three-Link Planar Manipulator

Throughout the towpreg layout on the composite structure surfaces, the links of the manipulator risk collision with the substrate and thus, interrupting the fiber placement and probably damaging the product or the robot links. Consequently, the feasibility analysis for the manufacture of composite bodies is not complete without a collision check between the composite structure (or its constituent towpregs) and the links of the robot, over the whole layout process.

In particular, collision is probable when considering the three-link planar arm in the vertical plane analysis; by reducing the whole three-dimensional workspace to just the vertical plane, the towpaths are more restricted in a narrower region where the likelihood of colliding with the three links is considerably high. Consequently, collision detection constitutes an important factor towards the complete verification of towpreg path feasibility.

Figure 7.9 shows all possible collision cases for the three-link manipulator; for all orientation angles of the end-effector with respect to the path, link 3 cannot collide with the towlines, whereas impact detection should be performed for links 1 and 2.

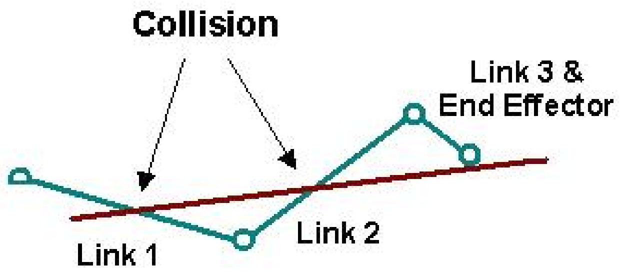


Figure 7.9 Possible collisions of links 2 and 3 with the tow

To detect collision of the towpreg path or the substrate with links 1 and 2, the following geometrical analysis is performed to verify the collision–free towpreg segment layout.

A Collision with link 1

Since joint 1 is stationary, one of the ends of link1 is fixed whereas the other end travels on a circular path. The circular arc has a fixed center located at joint 1 and a radius equal to the length of link 1. Consequently, the link covers a circular surface bounded by the initial and final positions of link 1 and by the arc traced by joint 2. As a result, collision is detected if only one point on the linear segment is located in the above circular area covered by link 1. Mathematically, the XZ coordinates of every point on the towpreg segment can be situated with respect to a collision region by determining the equation of the already mentioned circle and the equations of the two lines carrying the initial and final positions of the link 1.

B Collision with link 2

Since joint 3 is always above the linear path, link 2 risks collision with the towpreg segment only when joint 2 is below the linear path. The mathematical formulation checks the location of joint 2 with respect to the towpreg by determining the equation of the line carrying the towpreg segment. If the joint 2 is located below the line, and since joint 3 is fixed above the line, an additional detection should be performed to check for collision of link 2 with the towpreg segment.

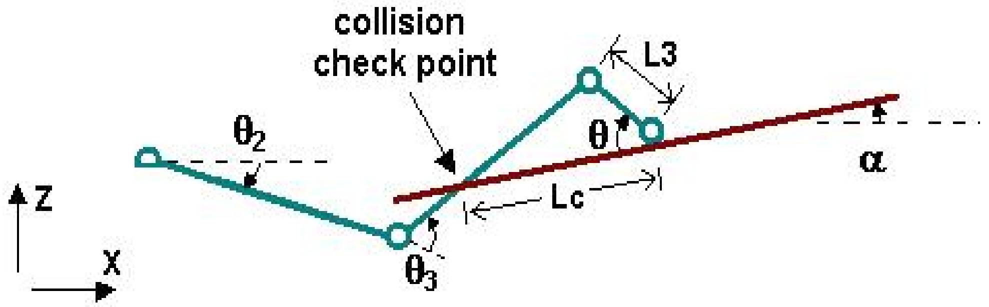


Figure 7.10 The collision check point for link2 and towpreg segment impact detection

As illustrated in Fig. 7.10, all points on the towpreg segment should not lie beyond the *collision detection point* shown in the figure. To mathematically control this condition, the length L_c is defined to be the distance between the contact point and the collision check point (eq. 7.6).

$$L_c = L_3 \cos(\theta) + \frac{L_3 \sin(\theta)}{\tan(\theta_2 + \theta_3 - \alpha)} \qquad (7.6)$$

If the distance between a point on the towpreg and the contact point is larger than Lc, this specific point is colliding with the link 2 and thus, impact is detected.

Briefly, to verify the feasibility of laying a towpreg segment, *all* points on the segment should meet the impact detection conditions stated above for *all* the robot configuration angles during towpreg placing.

CHAPTER 8: THE DISTINCT TECHNIQUES FOR THE MERLIN ROBOT

Different methodologies are used to approach the *global* feasibility problem in three-dimensional space. The Merlin robot is considered with its six joints, its physical links and its three-dimensional workspace. All six joints are free to rotate providing all desired positions and orientations required for the six-degree of freedom fiber placement task.

8.1 Lines Parameters in the Three-Dimensional Space

As already discussed in the vertical plane assumption, complex three-dimensional structures can be fragmented to many towpreg segments possibly with different orientations and positions in the workcell. Thus, three-dimensional path parameters should be established to control, study and verify the feasibility of all segments tapes constituting the desired complex structure.

8.1.1 The Towpreg Yaw, Pitch and Roll

The importance of handling line orientations is discussed in detail in the previous chapter that includes the different aspects in which the line orientations helps in describing all lines. In the three-dimensional space, the towpreg orientation can be specified by exactly three variables. The line *yaw*, *pitch* and *roll* are considered in this analysis.

- The line *yaw* is defined as the angle that the projection of the line on the XY plane makes with the X-axis (Fig. 8.1).
- The line *pitch* is described as the angle that the projection of the line on the XY plane makes with the Y-axis. It is the same angle considered in the vertical plane and called the *inclination* parameter (Fig. 8.1).
- The line *roll* is the angle that the surface of the towpreg tape makes with the XY plane (horizontal plane). In the vertical plane discussion, the towpreg segments are reduced to line segments, but the three-dimensional space requires considering the orientation of the *surface* of the prepreg tape. Since the fiber tows are laid on the structure surface *normally* to the tape

surfaces, the roll parameter allows controlling the orientation of the towpreg surface and providing the exact layout angles.

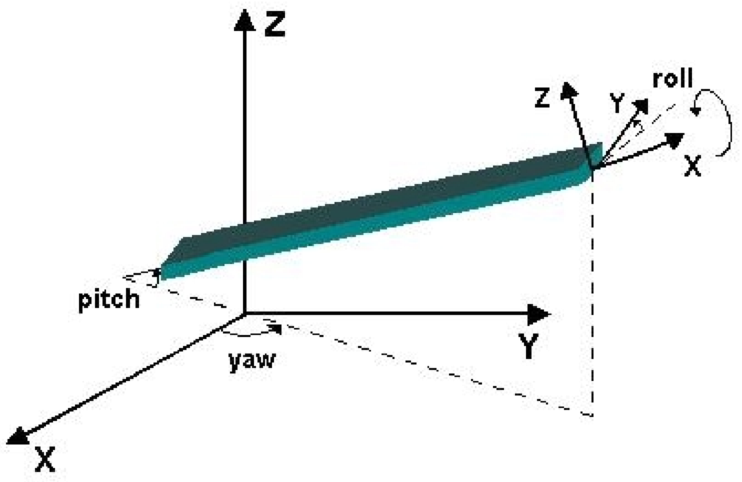


Figure 8.1 The yaw, pitch and roll angles of the towpreg tape

8.1.2 The Towpreg Offsets or Locations Parameters

As previously stated in the vertical plane discussion, defining locations for every fiber tape in the space helps in controlling the towpreg positions during the stacking process.
Three offset parameters are defined in order to locate all towpreg segments in the workcell. These offset parameters are the distances of the base frame origin to the towpreg segments, along the corresponding axis (Fig. 8.2).

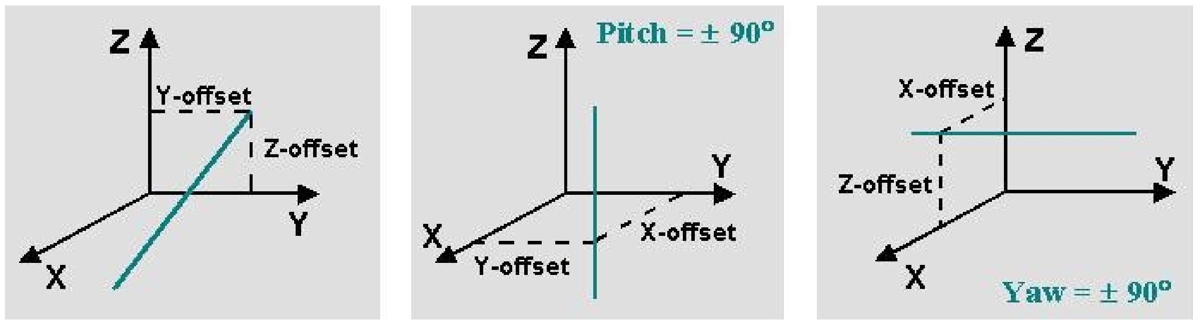


Figure 8.2 The three offsets of the towpreg tape

For every line in the space, only **two** offsets can sufficiently position the towpreg segment in the three-dimensional workspace of the robot.

- When the line pitch is equal to ± 90 degrees, the line is guaranteed to intersect with XY plane and accordingly, the *X-offset* and the *Y-offset* are the parameters that locate the towpreg segment in the workcell.
- If the yaw is ± 90 degrees, the line intersects with XZ plane, and thus, the *X-offset* and *Z-offset* should position the segment in the three-dimensional space.
- In all other orientational cases, the *Y-offset* and the *Z-offset* are used as the positioning parameters for the towpreg segment.

The elevation parameter used in the vertical plane corresponds to the Z-offset defined in the three-dimensional space.

8.1.3 The Orientation of the End-Effector with respect to the Line

The only controllable orientation angle of the compaction head can also be considered a parameter for the towpreg layout in three-dimensional space. As previously discussed in the vertical plane assumption, varying this orientation offers wider path possibilities by changing the boundaries of the feasible workspace. In addition, the end-effector orientation greatly influences the geometrical design of the compaction head and affects the performance of the hydraulic cylinder mounted on the end-effector. All these aspects are already explained in the two-dimensional assumption to stress on the importance of controlling this parameter for many different design considerations. Figure 8.3 shows the controllable orientation of the end-effector θ with respect to the towpreg segment in the three-dimensional space. All other orientation angles are predefined and fixed with respect to the fiber tape.

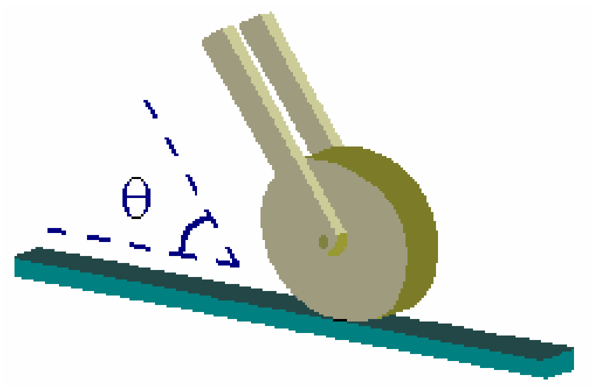


Figure 8.3 The orientation of the roller with respect to the towpreg segment in the 3-D space

8.1.4 The Different Robot Configurations

In laying the towpreg on its predefined segment, the robot can generally have eight different configuration angles. The elbow up/down modes are considered in the vertical plane workspace assumption. Similarly, more distinct configurations for the Merlin robot are able to trace the towpreg segments, all meeting the requirements of the fiber placement task. However, each configuration can generate a different segment length, often with very different start and end points. Consequently, choosing one of the eight configurations can be controlled by setting these configurations as three-dimensional towpreg path parameters. A detailed discussion in the following section focuses on describing the different configurations of the Merlin robot and the method to derive them.

8.2 Inverse Kinematics

Forward kinematics is used in the vertical plane analysis to provide the necessary relation between the towpreg paths and the angles of the manipulator. The complexity of the Merlin robot configuration prevents the application of the forward kinematics technique in the three-dimensional workspace; the required geometrical equations cannot effectively be developed to give the required relation between the towpreg paths and the angles of the robot.

As result, the *inverse kinematics* method is instead utilized in the complex three-dimensional study to provide the needed relations. It is a robust method capable of handling any pose in the workcell. This technique is based on finding the manipulator‘s set of angles when the position and orientation of a point in space is given. The configuration angles position the robot joints and locate the end-effector on the desired point with the given orientation. Since forward kinematics yields the position and orientation for given joints angles, the two methods are functionally opposite and provide the same needed kinematic relations in totally reversed approaches.

Here again, the towpreg paths are substituted by a series of points spaced by a chosen resolution, each point defining one pose for the compaction head. Since the position and orientation are available, the inverse kinematics equations compute the Merlin robot six joint angles.

By decoupling the Merlin Robot links to positional (1, 2, 3) and orientational (4, 5, 6), the Inverse kinematics formulations are based on the easy derivation of the wrist center position directly from the pose of the compaction head [37]. This property enables the development of the necessary equations that give the robot joint angles.

Appendix D presents the equations needed to find the set of six angles for a given position and orientation of the end-effector in the three-dimensional space. The following describes the equation's logic:

- Since the position and orientation of the end-effector are given in the inverse kinematics technique, the XYZ coordinates of the wrist center are easily computed and the point is located in the workspace.

  At this point, the following computations should be neglected if the wrist is located outside the reachable envelope or positioned too close to the robot center. In this case, the inverse kinematics solution does not exist, *i.e.*, there is no set of joint angles that can position the end-effector in the desired location.
- Otherwise, for the specific position of the wrist center, the waist joint has only two angles, θ_{1a} and θ_{1b} that provide the adequate arm configurations to reach the computed wrist center. The two angles correspond to the *Shoulder Front/Back* modes.
- The shoulder and elbow joints have two sets of solutions for **each** of the two waist angles: $\{\theta_{2a} \quad \theta_{3a}\}$ and $\{\theta_{2b} \quad \theta_{3b}\}$. Here again, the two sets of solution can be described as the *Elbow Up/Down* cases.
- For the four sets of solutions, the transformation matrices from the third to the final frame are computed. Comparing these matrices numerical values with their analytical expressions, two solutions for the orientational joint angles can be derived for **each** of the four sets of solutions. These two solutions are defined as the *Wrist Up/Down* configurations.

In combining all six different angle solutions, a point with a specific pose in three-dimensional space can be reached by the Merlin robot with eight different configurations. Figure 8.4 shows the eight different possible robot configurations.

By discretizing the towpreg segments to a series of points distanced by a chosen resolution, the inverse kinematic technique handles each point at a time and provides the set of robot joints angles to reach the considered point with the required orientation. In repeating this technique to target separately all the points on the towpreg segment, the global robot angles are computed to trace the desired path.

One of the crucial restrictions in placing towpregs on the predefined paths is the continuous and uniform layout of the tape with no flow interruption. To meet this condition, the same inverse kinematics solution should have the same robot configuration for all the discretized points on the towpreg segment. In other words, in tracing the desired lines, the robot angles cannot alternate between the different solutions, and the robot configuration should remain the same to ensure the smooth robot joint rotations. The attempt to change the inverse kinematics solution angles on the same towpreg path definitely causes sudden variations in the robot joints angles (*i.e.* shoulder front then back, or elbow up then down,….) and consequently, induces unallowable stoppage of the fiber placement.

The Merlin Robot Configurations	Shoulder ***Front***	Elbow ***Up***	Wrist ***Up***
			Wrist ***Down***
		Elbow ***Down***	Wrist ***Up***
			Wrist ***Down***
	Shoulder ***Back***	Elbow ***Up***	Wrist ***Up***
			Wrist ***Down***
		Elbow ***Down***	Wrist ***Up***
			Wrist ***Down***

Figure 8.4 The eight different robot configurations

The fixed inverse kinematics solution is chosen and controlled by the *lines configuration* parameter explained earlier. Every towpreg segment can be traced with generally eight different configurations unchanged along the whole layout path. However, each configuration generates very different feasible trajectories, with considerable changes in the towpreg segment lengths and dissimilar starting and end points. Therefore, considering all eight-configuration parameters includes ALL the different feasible trajectories in the effective workspace analysis.

Figure 8.5 shows the importance of considering all configuration inverse kinematics solutions, illustrated on the simplified three-arm manipulator for clarity. The Elbow Down solution prevents the feasibility of the whole length of the presented towpreg, whereas the Elbow Up solution provides another set of joint angles enabling the collision-free fiber layout on the whole length of the shown segment.

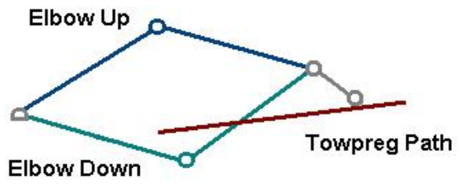


Figure 8.5 The Elbow Up/Down Inverse Kinematics Solutions

8.2.1 A Brief Comparison of the Forward and Inverse Kinematics Techniques

Even though the Forward Kinematics and Inverse Kinematics techniques provide the needed relations between the points on the linear paths and the robot configuration angles, they exhibit some differences that would affect the analysis of the effective workspace problem.

- In applying the inverse kinematics methodology, the linear paths are first discretized to a set of evenly spaced points, chosen with a fixed resolution. The linear path is thus, adequately represented by a uniform point set. The points are used to compute the correspondent robot joints angles. On the other hand, forward kinematics cannot achieve this property; since the joint angles are first are used to determine the discrete points on the linear path, the robot angles are evenly incremented yielding most probably a varying set of points on the tow path and depending on the *changing* Jacobian matrix to locate the points. Figure 8.6 illustrates the unevenly discrete points on the linear path; for fixed increments of θ_2, the distances between the successive end-effector positions vary considerably.

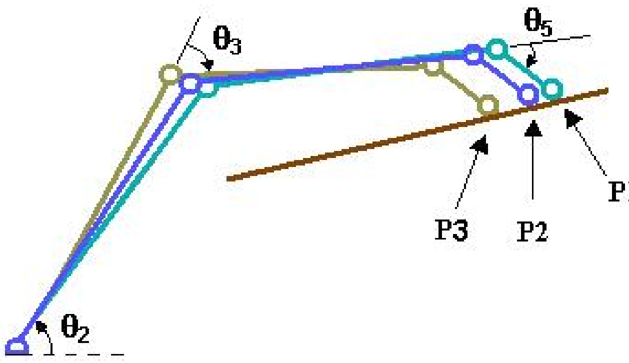


Figure 8.6 The unevenly spaced points on the linear path

- On the other hand, the only drawback of inverse kinematics is the long computation time to solve the many equations and to yield the solutions. Forward kinematics is found to be much faster with its few and simple formulations.

8.3 The Head Velocity and Compaction Force

By adding new parameters for three-dimensional linear segments, the expression of the velocity and the compression force on the compaction head becomes more complicated in order to include all the variables that influence these two dynamic properties.

As shown in Fig. 8.1, a frame is positioned on the surface of the towpreg tape where the X-axis is parallel to the towpath and Z-axis is normal to the prepreg tape. As already detailed in the vertical plane assumption, the velocity vector is directed along the path direction (parallel to the X-axis shown in Fig. 8.1), whereas the compression force is always normal to tape surface (parallel to Z-axis-for the proper compaction direction).

However, the frame attached on the towpreg is oriented as a function of the three angle parameters, the line yaw, pitch and roll. Accordingly, the orientation of the towpreg frame with respect to the global frame is expressed as:

$$T = \left[\begin{array}{c}\textit{Yaw Rotation}\\ \textit{about the Z-axis}\end{array}\right]\left[\begin{array}{c}\textit{Pitch Rotation}\\ \textit{about the new Y-axis}\end{array}\right]\left[\begin{array}{c}\textit{Roll Rotation}\\ \textit{about the new X-axis}\end{array}\right] \quad (8.1)$$

$$T = \begin{bmatrix} \cos(yaw) & -\sin(yaw) & 0 \\ \sin(yaw) & \cos(yaw) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos(pitch) & 0 & -\sin(pitch) \\ 0 & 1 & 0 \\ \sin(pitch) & 0 & \cos(pitch) \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(roll) & -\sin(roll) \\ 0 & \sin(roll) & \cos(roll) \end{bmatrix} \quad (8.2)$$

Now that the orientation of the frames is determined, the velocity vector V has the same direction as the X-axis and can be expressed by its magnitude $|V|$ multiplied by its direction:

$$\begin{bmatrix} V_x & V_y & V_z \end{bmatrix} = |V| . \begin{bmatrix} T_{11} & T_{21} & T_{31} \end{bmatrix} \quad (8.3)$$

Even though the rotation matrix T contains the roll angle, the velocity vector is only dependent on the yaw and pitch angles, sufficiently providing the towpreg path direction.

The compression force F_c can be expressed with its magnitude $|F_c|$ and the direction of the Z-axis on the towpreg frame:

$$\begin{bmatrix} F_x & F_y & F_z \end{bmatrix} = |F_c| . \begin{bmatrix} T_{13} & T_{23} & T_{33} \end{bmatrix} \tag{8.4}$$

8.4 Collision Detection

As discussed previously, the robot links risk collision with the structure substrate during fiber layout causing the interruption of the fabrication process and probably damage of the manufacturing tools. Consequently, collision detection is definitely required to check the complete feasibility of impact-free towpreg paths. In the vertical plane analysis, the manipulator's arms are represented by a series of straight lines and collision detection is based on the equations of theses lines and checks for line *intersection* with the towpreg segments. Limiting the manipulator's links and the workspace to just the vertical plane allows the use this simplified technique, as detailed in the previous chapter.

By extending the problem to the global workspace, the robot physical links should be represented as volumetric three-dimensional bodies. The towpreg segments can still be represented with just a line but intersection or collision should be performed to account for the volume enclosed by the several links and other physical obstacle. Any three-dimensional body can be totally enclosed by planes crossing the object surfaces. For the collision detection logic, this property allows an efficient representation of all three-dimensional bodies in the workcell of the robot; the links, and counterweights in the workcell of the robot are considered volumetric objects enclosed and limited by many planes crossing their surfaces. Figure 8.7 shows only three of the six surfaces covering the physical robot link.

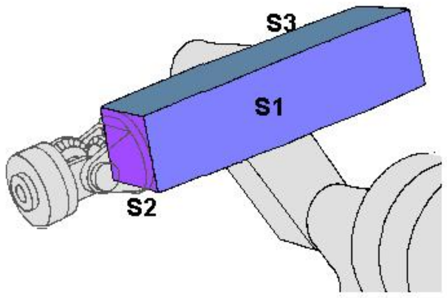


Figure 8.7 Surfaces enclosing the link

As a result, collision is detected if the towpreg segments cross one of the surfaces enclosing the link body. But since the fiber towlines are discretized to points along the length of the segment, impact occurs when towpreg points are located *inside the spatial volume enveloped by the surfaces*.

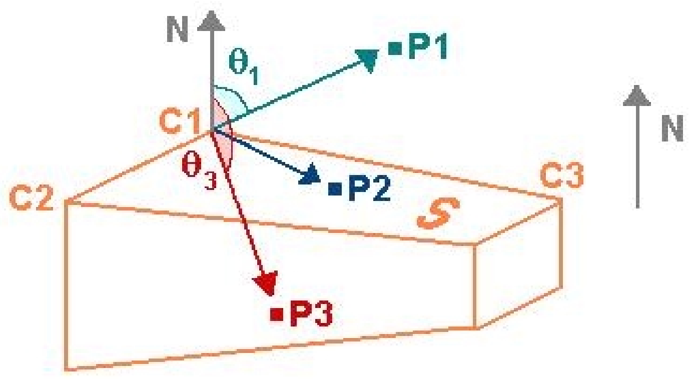


Figure 8.8 Locating the points with respect to the surface S

The approach introduced here determines the location of the towpreg discrete points with respect to each surface plane, and by repeating this detection for all the surfaces enclosing the body volume, the point can be positioned with respect to the link surfaces or the link volume, and collision can accordingly be checked:

- Simple transformation matrices locate the positions of the corners of the considered link with respect to the frames attached on the link.
- Once the XYZ coordinates of the body corners are computed, one of the body surfaces ***S*** is selected and represented by its normal vector ***N***. The normal vector is determined by calculating the cross product of two vectors joining the surface corners. $\vec{N} = \overrightarrow{C_1C_2} \times \overrightarrow{C_1C_3}$
- The normal vector is moved and positioned on one of the surface corners (e.g. point C_1 in Fig. 8.8) and then, the dot product of the normal vector ***N*** with the vectors relating the corner point to the towpreg point is computed, *i.e.* $\vec{N} \cdot \overrightarrow{C_1P_1}$
- Since $\cos(\theta_1) = \dfrac{\vec{N} \cdot \overrightarrow{C_1P_1}}{|\vec{N}| \cdot |\overrightarrow{C_1P_1}|}$, the cosine of the dot product and the angle formed by the two vectors is the same. Accordingly, the location of the point P_1 with respect to the surface ***S*** is determined:

1) A positive value of the dot product infers a positive value for the angle cosine. θ_1 is then smaller then 90 degrees and consequently, P_1 is on the top side of the surface ***S***.
2) A null value of the dot product indicates a zero value for the angle cosine. The angle formed is exactly equal to 90 degrees, locating the towpreg point P_2 on the surface ***S*** of the link body.
3) A negative value of the dot product gives a negative value for the angle cosine. θ_3 is then larger than 90 degrees and the towpreg point P_3 is on the bottom side of the surface ***S***.

By repeating these above steps for all surfaces enclosing the three-dimensional body, every point is located with respect to all the link surfaces separately, checking collision of the towpreg point with the link physical structure.
- If the towpreg point is inside the enclosed object, collision is detected with the specific link.
- Otherwise, the fiber layout on this point is valid.

The workspace of the Merlin robot includes mobile links and stationary obstacles that risk interfering in the fiber placement process and colliding with the towpreg substrate. Consequently, collision detection for ALL links and obstacles should be performed for the proper placement of the workpiece or substrate in the workspace of the robot.

The Waist Body

The Merlin robot link configuration and the predetermined six joints hardware limits defines the reachable workspace of the robot and accordingly prevents the robot faceplate from critically approaching the waist link body. On the other hand, the fiber placement task involves a compaction head with considerable dimensions capable of handling its multiple functions. Consequently, the end-effector mounted on the robot faceplate risks collision with the waist link for joint angles within their feasible span. As a result, collision detection should be performed between the end-effector positions (*i.e.* the substrate or towpreg locations) and the waist link. For geometrical simplicity, the robot waist shown in Fig. 8.9 is considered a rectangular prism enclosed by four vertical and two horizontal planes.

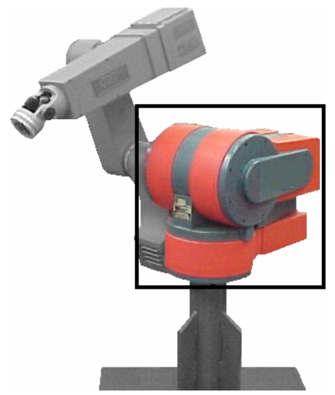


Figure 8.9 The robot waist

The Counterweight, Connector and Arm

During the tow layout, the compaction head travels along previously cured tows on the substrate located in the reachable workspace of the robot. On the other hand, the position of the substrate in the workspace might possibly intersect with the rotational space for the rotating link bodies. Accordingly, the position of the counterweight, connector and arm links (shown in Fig. 8.10) should be checked to avoid any possible collision with the substrate. The connector and the counterweight are geometrically represented as rectangular prisms enclosed each by six rotating planes, whereas the counterweight is enveloped by eight surface planes.

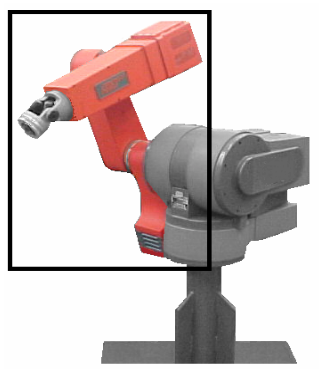


Figure 8.10 The counterweight, connector and arm

The Stationary Obstacles

Because of the considerable dimensional extensions induced by mounting the end-effector on the robot faceplate, the compaction head roller might collide with stationary obstacles, like the robot base, cylindrical trunk or even the floor. Geometrically, impact detection is performed by checking the Z coordinates of the roller contact point and comparing it to the robot base elevation and null elevation.

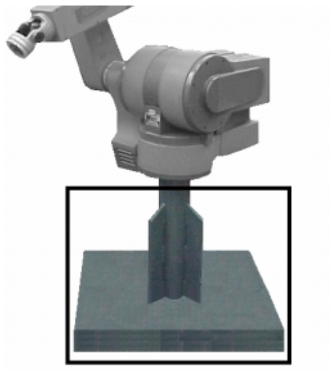


Figure 8.11 The robot base and trunk

CHAPTER 9: THE ALGORITHM TO DETERMINE ALL FEASIBLE TOWPREG PATHS

Since the effective workspace analysis is built on determining all feasible linear towpreg paths, a routine is developed to locate the boundaries of these linear paths, when the trajectory parameters are specified. This general method can be applied in all workspace analysis where straight-line paths are considered for feasibility. In particular, the same logic is followed to solve the trajectory validity problem for both the vertical plane assumption and in the global three-dimensional workspace.

A Matlab algorithm (included in Appendix F) is developed to check the feasibility of towpreg segments paths; the starting and end points of the end-effector trajectories are located while all kinematic restrictions of the robot are met. Prior to discussing and explaining the concepts followed in the algorithm, the flowchart shown on the next page summarizes the code logic: for a chosen unbounded line in space, the algorithm determines and outputs all feasible paths laying on the line.

9.1 The Input Parameters and Specifications

All the process parameters and robot specifications should be known to accurately represent the Merlin robot and the manufacturing variables. As previously discussed, the robot is represented by the five ***Denavit-Hartenberg parameters*** (in Appendix C) that set the required frames on the robot joints. Thus, the robot physical links sizes and the joints axis locations and orientations are given for the analysis in the Matlab algorithm.

On the other hand, the ***end-effector geometrical dimensions*** should also be entered to allow the correct positioning of the compaction head contact point with respect to the robot frames. Even though the kinematic relation can be developed for all end-effector configurations, the current analysis focuses on only two geometrical measures that locate the position of the roller contact with respect to the faceplate: S_1 and S_2 are the distances of the roller contact to the center of the faceplate along the Z_6 and X_6 respectively, as shown in Fig. 9.1.

To access the dynamic performance of the robot, the joints specifications should also be included in the algorithm in order to provide the required threshold limits. By entering the ***maximum motors torques*** and ***angular velocities*** along with the ***joints gear ratios***, the robot dynamic restrictions are set and ready to test the validity of towpreg trajectories. For the complete feasibility analysis, the ***hardware joint limits*** should be entered as well to verify the angle span of every joint.

On the other hand, the ***desired compression force*** and ***velocity values*** should be selected to give the necessary dynamic parameters along the compaction head paths. As previously discussed, the manufacture designer chooses the adequate compaction velocity and forces that insure the proper material consolidation.

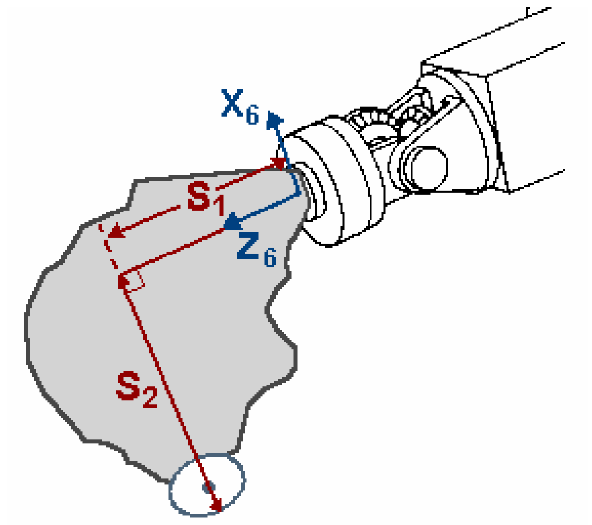
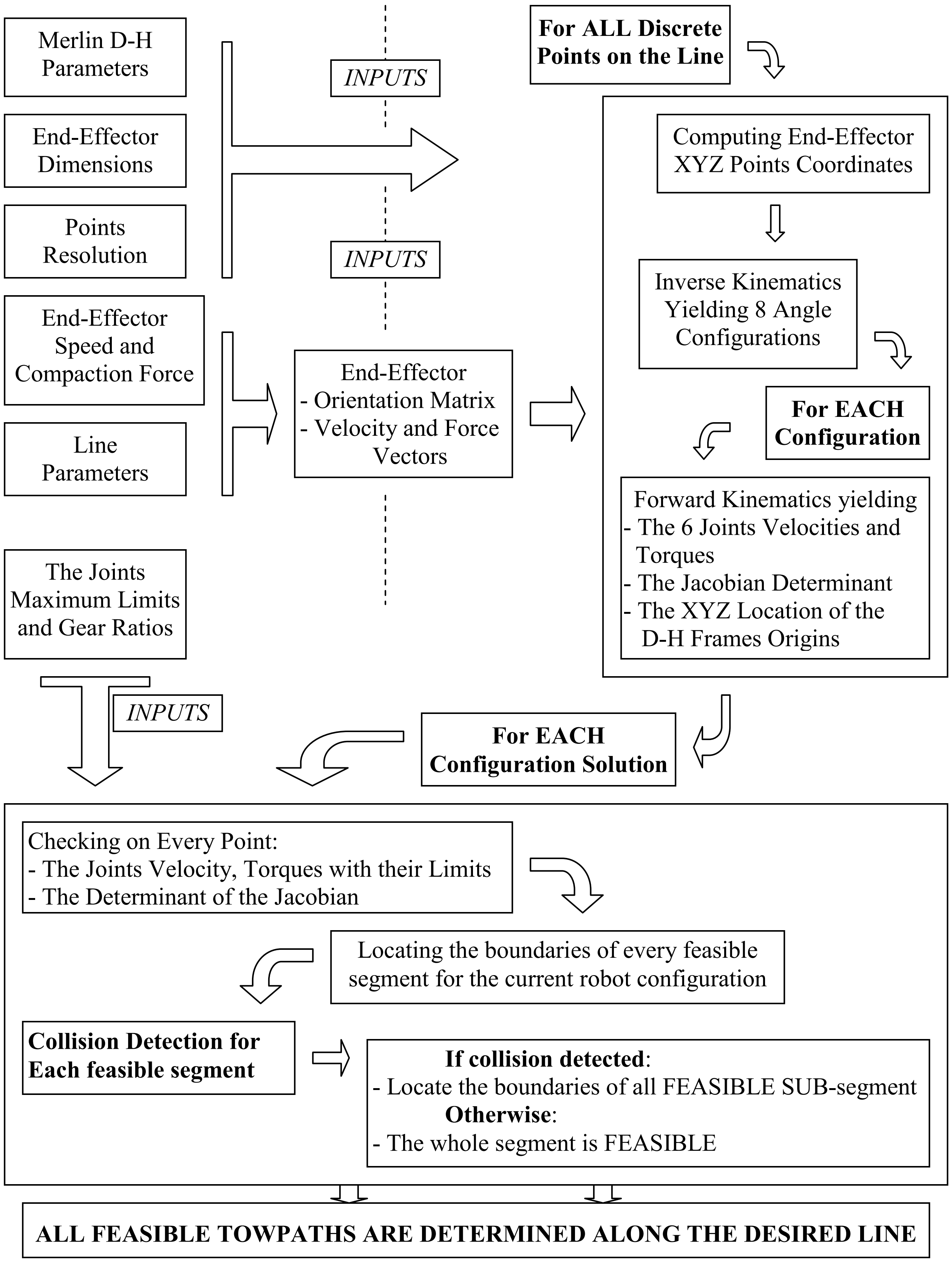


Figure 9.1 The dimensions of the end-effector

One unbounded straight line should be chosen to handle the path verification analysis. Possible segments are traced on the line and checked for feasibility. The line is selected by determining its multiple parameters. The ***yaw, pitch*** and ***roll*** set the desired orientation of the line whereas the corresponding ***two offsets*** provide it's positioning in the robot workcell (discussed previously). Finally, the ***end-effector orientation with respect to the line*** should also be chosen and included as one of the inputs to the algorithm.

Finally, the ***resolution*** should be carefully selected to define the spacing between the discrete points: a fine resolution increases the computation accuracy at the expense of the computation time and consequently, an adequate compromise should be achieved.



Merlin D-H Parameters
End-Effector Dimensions
Points Resolution
End-Effector Speed and Compaction Force
Line Parameters
The Joints Maximum Limits and Gear Ratios
INPUTS
INPUTS
INPUTS
End-Effector
- Orientation Matrix
- Velocity and Force Vectors
For ALL Discrete Points on the Line
Computing End-Effector XYZ Points Coordinates
Inverse Kinematics Yielding 8 Angle Configurations
For EACH Configuration
Forward Kinematics yielding
- The 6 Joints Velocities and Torques
- The Jacobian Determinant
- The XYZ Location of the D-H Frames Origins
For EACH Configuration Solution
Checking on Every Point:
- The Joints Velocity, Torques with their Limits
- The Determinant of the Jacobian
Locating the boundaries of every feasible segment for the current robot configuration
Collision Detection for Each feasible segment
If collision detected:
- Locate the boundaries of all FEASIBLE SUB-segment
Otherwise:
- The whole segment is FEASIBLE
ALL FEASIBLE TOWPATHS ARE DETERMINED ALONG THE DESIRED LINE

Figure 9.2 The algorithm flowchart

9.2 Computing the Constant End-Effector Orientation, Velocity and Compaction Force Vectors

For a specific line in space, the approaches to determine the velocity and compaction force vectors are previously derived in the vertical plane assumption and in the three-dimensional workspace methodologies. However, for fixed line parameters, these two vectors are constant in value and direction and do not change along the line since they only depend on the unvarying parameters of the line. In addition, the orientation of the compaction head is also predetermined and fixed along the line. As previously discussed, the orientation is fixed in two axes to allow adequate compaction layout, where as the angle about the third axis is chosen and set as one of the line parameters.

In placing the end-effector on the towpreg line and orienting it with the angle θ (the orientation of the end-effector with respect to the line), the line angle parameters are used to compute the orientation of the compaction head with respect to the base frame. Equation 8.2 gives the orientation T of the frame located on the towpreg surface (shown in Fig. 8.1). By simply adding the rotation matrix for the angle θ, the orientation of the end-effector can be computed:

$$R = T \cdot \left[\begin{array}{c} \theta \textit{ Rotation about} \\ \textit{the Y-axis on the} \\ \textit{towpreg frame} \end{array} \right] \tag{9.1}$$

$$R = T \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{9.2}$$

The orientation of the end-effector, the velocity and forces vectors on the end-effector can be considered as the indirect input constants of the algorithm, being fixed throughout the whole code.

9.3 Collecting All Data along the Line

To gather and analyze geometrical and dynamic data along the chosen line, discrete points spaced by the selected resolution substitute for the continuous line and allow the feasible computation of discretized data, utilized to determine the boundaries of the paths. Figure 9.3 illustrates the procedures followed to collect the needed data, constituting a part of the whole algorithm flowchart.

A loop is generated to retrieve the required data from possibly every point on the chosen line. The loop starts (and ends) at largely distant points from the robot center or the base frame. The XYZ position of all considered discrete points is computed: as previously defined, two line offsets determine the position of the intersection point of the line with generally the YZ plane (see the towpreg offsets section for details). Repeatedly, an incremented positional variable *p* locates new discrete points distanced initially from the intersection point, by the selected resolution value.

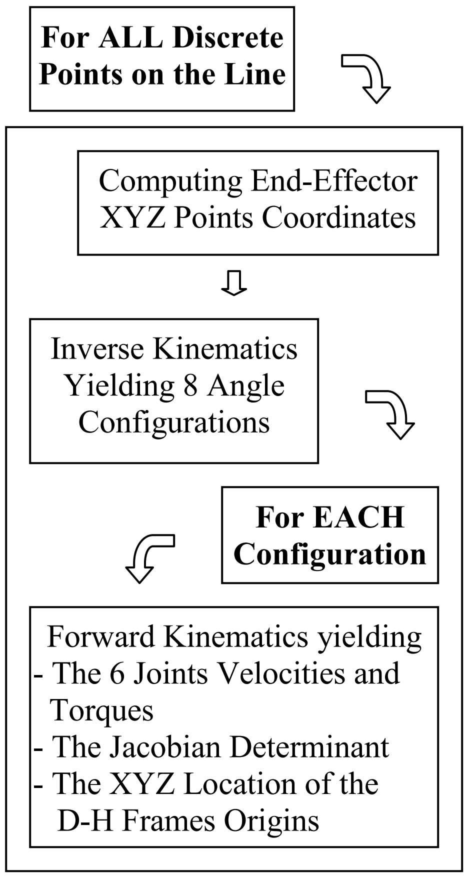

For ALL Discrete Points on the Line
Computing End-Effector XYZ Points Coordinates
Inverse Kinematics Yielding 8 Angle Configurations
For EACH Configuration
Forward Kinematics yielding
- The 6 Joints Velocities and Torques
- The Jacobian Determinant
- The XYZ Location of the D-H Frames Origins

Figure 9.3 Collecting all needed data on the discrete points

Equation 16 shows the product of four matrices needed to compute the end-effector XYZ position on the line.

$$D=\begin{bmatrix}1&0&0&0\\0&1&0&Y-offset\\0&0&1&Z-offset\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}\cos(yaw)&-\sin(yaw)&0&0\\\sin(yaw)&\cos(yaw)&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}\cos(pitch)&0&-\sin(pitch)&0\\0&1&0&0\\\sin(pitch)&0&\cos(pitch)&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}1&0&0&p\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} \quad (9.3)$$

The first matrix locates the intersection of the line with the YZ plane. A frame parallel to the base frame is positioned at this point. The following two matrices orient the new frame along the line direction or the end-effector velocity direction (already derived), and then the last matrix moves the oriented frame (a distance *p*) onto the discrete points on the line along its x-axis. The XYZ coordinates of the end-effector position fill column four in the ***D*** matrix.

9.3.1 Inverse Kinematics

By specifying the position and orientation of each discrete point on the line, the inverse kinematics technique computes the set of robot joints angles that locate and orient the roller contact point onto its predetermined pose. In addition, the inverse kinematics equations are capable of specifying the reachable discrete points on the infinite line and cut out all points that the compaction head contact point cannot attain.

All formulas and equations were reviewed in the previous chapter. However, a minor modification should be performed to account for the end-effector dimensions. Equation 9.4 accurately positions the wrist center when the pose of the roller contact point is given. All variables in equation 9.4 are already defined.

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_x - (d_6 + S_1).r_{13} + S_2.r_{11} \\ d_y - (d_6 + S_1)r_{23} + S_2.r_{21} \\ d_z - (d_6 + S_1)r_{33} + S_2.r_{31} \end{bmatrix} \tag{9.4}$$

As previously discussed in detail, eight different configurations or solutions are generally available. The *n* sets of angles are stored and categorized according to the geometric robot configurations in Fig. 9.4.

Shoulder ***Front***	Elbow ***Up***	Wrist ***Up***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.1}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.1}$
		Wrist ***Down***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.2}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.2}$
	Elbow ***Down***	Wrist ***Up***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.3}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.3}$
		Wrist ***Down***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.4}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.4}$
Shoulder ***Back***	Elbow ***Up***	Wrist ***Up***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.5}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.5}$
		Wrist ***Down***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.6}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.6}$
	Elbow ***Down***	Wrist ***Up***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.7}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.7}$
		Wrist ***Down***	P_1:$\{\theta_1 \ldots \theta_6\}_{conf.8}, \ldots, P_n$:$\{\theta_1 \ldots \theta_6\}_{conf.8}$

Figure 9.4 Categorizing the set of angle solutions according to the robot configuration

Alternatively, in the vertical plane analysis, the forward kinematics technique varies the joints set of angles and then computes the pose for each discrete point on the line.

9.3.2 Computing the Joint Velocities, Torques and the Jacobian Determinant

To gather all the required kinematic and dynamic data along the towpreg paths, the computation of the Jacobian matrix is necessary to relate the end-effector kinematics to the base frame. Therefore, a new transformation matrix is developed that would locate the compaction head contact point with respect to the frame attached to the robot faceplate. Using the measures S_1 and S_2, the transformation A_7^6 can be computed:

$$A_7^6 = \begin{bmatrix} \cos(\pi) & -\sin(\pi) & 0 & -S_2 \\ \sin(\pi) & \cos(\pi) & 0 & 0 \\ 0 & 0 & 1 & S_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9.5}$$

As a result, the Jacobian matrix J is developed to relate the end-effector kinematics to the base frame on the robot.

$$J = \begin{bmatrix} Z_0 \times (p_7 - p_0) & Z_1 \times (p_7 - p_1) & Z_2 \times (p_7 - p_2) & Z_3 \times (p_7 - p_3) & Z_4 \times (p_7 - p_4) & Z_5 \times (p_7 - p_5) \\ Z_0 & Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{bmatrix} \tag{9.6}$$

where Z_i is the Z-axis orientation of the ith frame with respect to the base frame
p_i is the XYZ position of the ith frame origin with respect to the base frame
Z_i and p_i are respectively the third and fourth columns in T_i^0
T_i^0 is defined in equation 6.3.

Consequently, for each of the eight robot configurations, and at every reachable discrete point on the line, the determinant of the Jacobian is computed (used later for singularity avoidance) as well as the joint angular velocities and torques. The end-effector velocity vector ***V*** and the exterior forces ***F*** on the end-effector are used in equation 9.7 and 9.8 for the calculation of the six joints speeds ***Q*** and torques **Γ** arrays.

$$Q = J^{-1}V \tag{9.7}$$

$$\Gamma = J^{-1}F \tag{9.8}$$

Figure 9.5 illustrates the kinematic data collected in the above loop.

$P_1:\{\theta_1\ldots\theta_6\}_{cong.1},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.1}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.1},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.1}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.2},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.2}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.2},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.2}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.3},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.3}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.3},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.3}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.4},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.4}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.4},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.4}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.5},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.5}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.5},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.5}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.6},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.6}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.6},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.6}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.7},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.7}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.7},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.7}$
$P_1:\{\theta_1\ldots\theta_6\}_{cong.8},\ldots, P_n:\{\theta_1\ldots\theta_6\}_{cong.8}$	$\Rightarrow$	$P_1:\{ det(J) ,Q ,\Gamma\}_{conf.8},\ldots, P_n:\{ det(J) ,Q ,\Gamma \}_{conf.8}$

Figure 9.5 All kinematic data categorized and stored

9.4 Determining All Feasible Segments on the Line

The second and final part of the algorithm processes the data collected and determines the boundaries of continuous feasible segments on the entered infinite line. Figure 9.6 summarizes and illustrates the logic followed. It is crucially important to consider the eight different robot configurations *separately*. As explained in the previous chapter, a single feasible segment requires the continuous fiber layout with only one unchanged robot configuration. Otherwise, the placement path is interrupted or paused to give enough time for changing (drastically) the robot angles. Therefore, Fig. 41 shows the feasibility analysis applied to each robot configuration data at a time.

9.4.1 Checking the Kinematic Limits on Every Discrete Point

Four robot restrictions are checked to decide on the fiber layout feasibility. First, the joints angles should be within the span extent limited by the hardware limits. The Jacobian determinant is then checked and compared to a threshold value, and finally, the six joints velocities and torques are checked with their maximum specifications limits.

A Checking the joints angles with the hardware limits

The stored joints angles are compared to the upper and lower hardware limits listed in Appendix B. Since joints 4 and 6 are free to rotate with no stops, only joints 1, 2, 3 and 5 angles are checked for EACH discrete point; the *hardware limits detection* is passed on a point if the corresponding angles lie within the feasible joint span.

Lower limit of joint i < θ_i < Upper limit of joint i

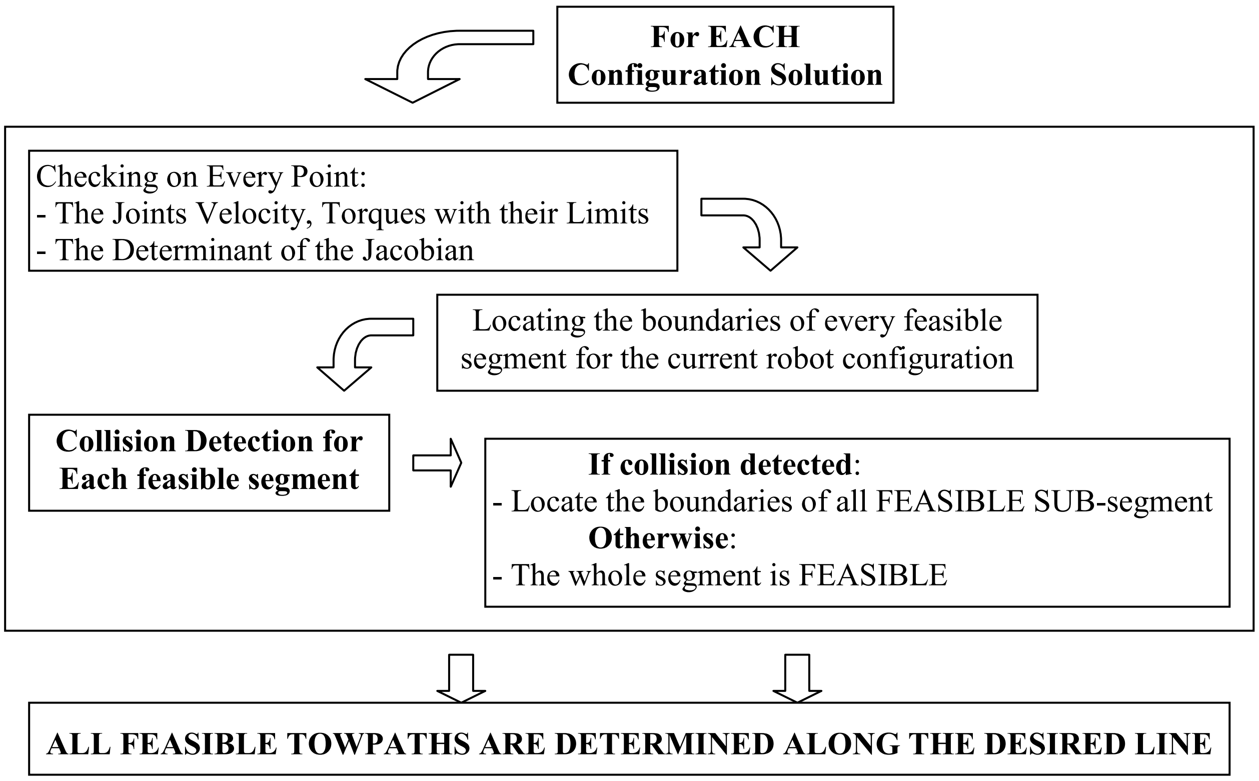

For EACH Configuration Solution
Checking on Every Point:
- The Joints Velocity, Torques with their Limits
- The Determinant of the Jacobian
Locating the boundaries of every feasible segment for the current robot configuration
Collision Detection for Each feasible segment
If collision detected:
- Locate the boundaries of all FEASIBLE SUB-segment
Otherwise:
- The whole segment is FEASIBLE
ALL FEASIBLE TOWPATHS ARE DETERMINED ALONG THE DESIRED LINE

Figure 9.6 The algorithm partial flowchart determining all feasible segments

B Checking the Jacobian determinant with a chosen threshold

Singularity avoidance is performed in checking the Jacobian determinant value. As previously reviewed, the Jacobian determinant infers the dexterity of the Merlin robot in laying the towpreg. High values indicate smoother fiber placement. On the other hand, values close to zero are caused by robot singularities inducing extremely high joint torques or velocities and consequently interrupting the layout process. Therefore, a minimum threshold dexterity value should be chosen by the manufacture designer to insure smooth fiber placement and accordingly, to spot all robot configurations close to singularities.

For each discrete point, the absolute value of the determinant is compared with the selected threshold D_{thr} and *Jacobian check* fails to pass for values lees than the threshold.

Absolute value of the Jacobian determinant > D_{thr}

If the Jacobian determinant is found to be higher than the threshold limit, additional verifications should be performed for the complete detection analysis: during the fiber layout, the occurrence of singularity configurations lowers the Jacobian determinant so rapidly that its critical ‘close to zero’ values cannot often be ‘seen’ or recognized by the discrete point spaced with the chosen resolution.

(i) One approach to solve the problem is to consider two consecutive discrete points. As seen in Fig. 9.7, when the Jacobian determinant has two consecutive values with different signs, the determinant curve necessarily crossed the zero line and consequently, the *Jacobian determinant check* fails to pass. Points 4-5 and 8-9 in Fig. 9.7 are on the opposite sides of the zero line. Even though the determinant at those points is larger than the minimal threshold, robot configuration singularities occur between the above discrete points couple.

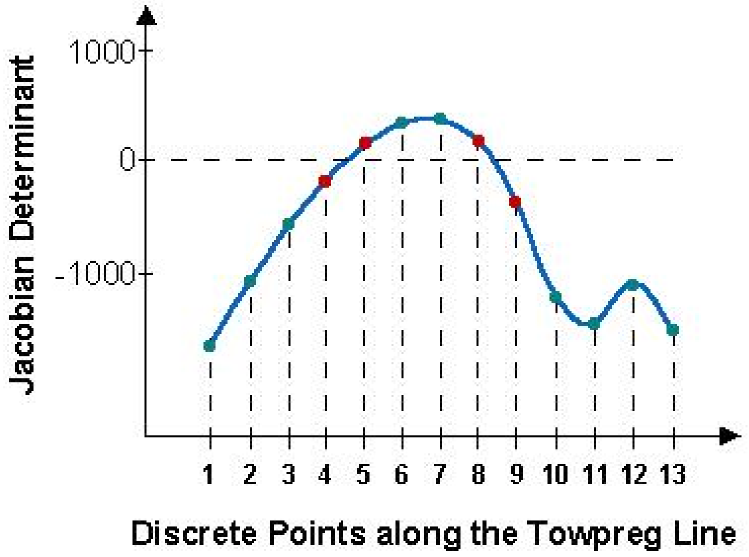


Figure 9.7 Singularity occurring between points with opposite Jacobian determinant signs

(ii) A second check should be performed when the Jacobian determinant curve peaks toward the zero line. The opposite slopes of two consecutive pairs of discrete points (3-4 and 5-6 in Fig. 9.8) can determine and locate a curve peak on the critical digital interval. Even though the determinant values stored on the discrete points are relatively large and far from the threshold limit, the real ‘hidden’ peak can be superimposed on the digital curve and carries points with Jacobian determinants smaller than the threshold limit (Fig 9.8).

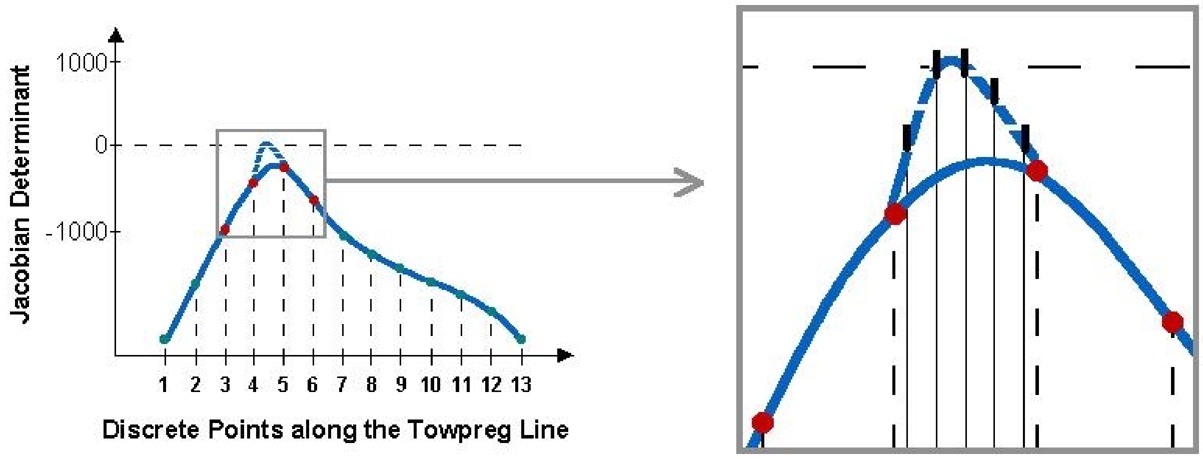


Figure 9.8 Checking for singularities when the determinant curve peaks towards the zero line

When a peak is determined, the towpath length interval separating the two discrete points on the determinant curve peak is digitized with a much finer resolution. The inverse kinematics method translates the new points to a set of joint angles that accordingly, allows the computation of the Jacobian determinant values for each fine point created (Fig. 9.9). Again, the selected inverse kinematics solution (or configuration) should fit the original robot configuration used to trace the original coarse segment.

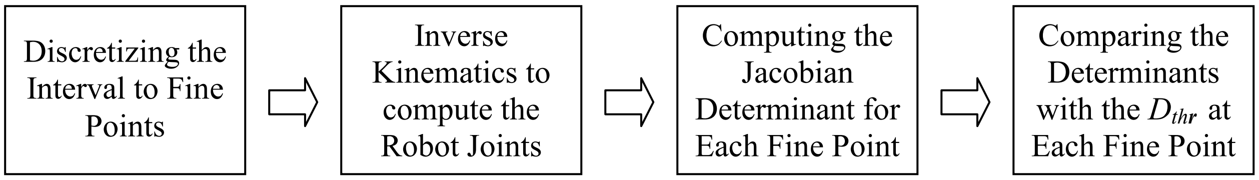

Discretizing the Interval to Fine Points
Inverse Kinematics to compute the Robot Joints
Computing the Jacobian Determinant for Each Fine Point
Comparing the Determinants with the D_{thr} at Each Fine Point

Figure 9.9 Computing and checking the Jacobian determinant for the fine points created

Once the Jacobian determinant is available for the new fine points, they are compared to the threshold limit and accordingly, the *Jacobian determinant detection* is completed.

C Checking the Joints Velocities

As already discussed, the joint angular velocities constitute one of the robot restrictions in verifying the feasibility of trajectories. The motor velocity specifications are listed in table 6.2. The maximum joint angular velocities are calculated and compared to the data stored on each discrete point on the line. However, the maximum velocities are divided by a factor of safety that would protect the joint motors from attaining their maximum dynamic limits. Here again, the factor of safety value is selected by the process designer. For each point on the line, and for every joint axis, the following detection should be performed:

Absolute value of the angular velocity of joint i < joint i maximum velocity / Safety factor

The *joint angular velocity* check fails to pass at a specific point if at least, one of the joint actual velocities exceeds its predefined limits.

(i) As previously explained in the Jacobian determinant check, discretizing the towpreg paths to considerably spaced points might 'hide' the real values or behavior of the joints angular speeds. Consequently, digitizing the critical intervals (on the curves peaks) would solve the problem by creating finer points on the towpreg linear path. However, all these extremely high joint velocities occur in (or close to) *singular robot configurations* that can increase drastically and very rapidly the eight joints velocity values simultaneously. Accordingly, the Jacobian determinant peaks are again considered as an indication to robot singularity occurrences and thus, determining the coarse intervals where finer digitizing is required to check the angular velocity limits.

Once the critical path regions are located, the digital interval is divided to finer points. Again, the current inverse kinematics solution provides the robot angles, used to compute all fine angular velocities for the six joints. The velocity restriction check fails to pass if at least one of the joint velocities on one of the fine points exceeds its maximum limit.

(ii) However, certain towpreg paths might perpendicularly intersect with singularity surfaces in the robot workcell. In this case, while tracing the towpreg line, the angular velocities keep considerably small values till the intersection point is reached. At this specific point, the angular velocities are suddenly set to an infinite value increased with an infinite slope. This high or infinite velocity value only occurs at the specific intersection point and cannot be 'seen' even when the critical digital interval is divided to very fine points.

To solve this problem, the stored joints angles are observed and analyzed to compute the high angular joint velocities. When the coarse critical interval is located on the towpreg path and divided to finer discrete points, the inverse kinematics technique computes the joint angles, whereas the calculation of joints velocities with the forward kinematics method is neglected. Instead, the angle slopes are determined by finding the difference between consecutive joints angles (Fig. 9.10) and accordingly, the *joint velocities* can be computed by using the angle slopes and including a time interval; the time to trace one coarse digital interval can be calculated by dividing the discrete spacing length *p*, by the layout or end-effector speed *V*.

D Checking the Joint Torques

The joints angular torque restrictions are also checked for trajectory feasibility. For each point on the line and again for each of the eight joints, the torques are compared to the motors torques specifications listed to table 6.3. The same factor of safety is added for the joint motors' protection:

Absolute value of the angular torque of joint i < joint i maximum torque / Safety factor

The *joint angular torque* check fails to pass if at least, one of the joints actual torques exceeds its predefined limits.

Since the joints torques and velocities are both subject to the same dramatic increase in values when the robot is in (or close to) a singular configuration, the previous velocity digitization detection would involve both joints velocity and torques restrictions and thus separate analysis for the joint torque values is not necessary.

In repeating the above restrictions detections for the eight robot configurations, the Pass/Fail detection results are stored on each discrete point for the four kinematic feasibility criteria: the hardware limit, the Jacobian determinant and the joint velocities and torques.

9.4.2 Determining All Feasible Segments on the Line

The previous kinematic limits data are then processed to locate the boundaries on all feasible segments on the line. The towpreg layout is verified on a discrete point only if the above four restrictions checks are all passed; in this case, the towpath point is defined as a *valid point*. In repeatedly joining consecutive valid points, feasible segments are determined and located on the line.

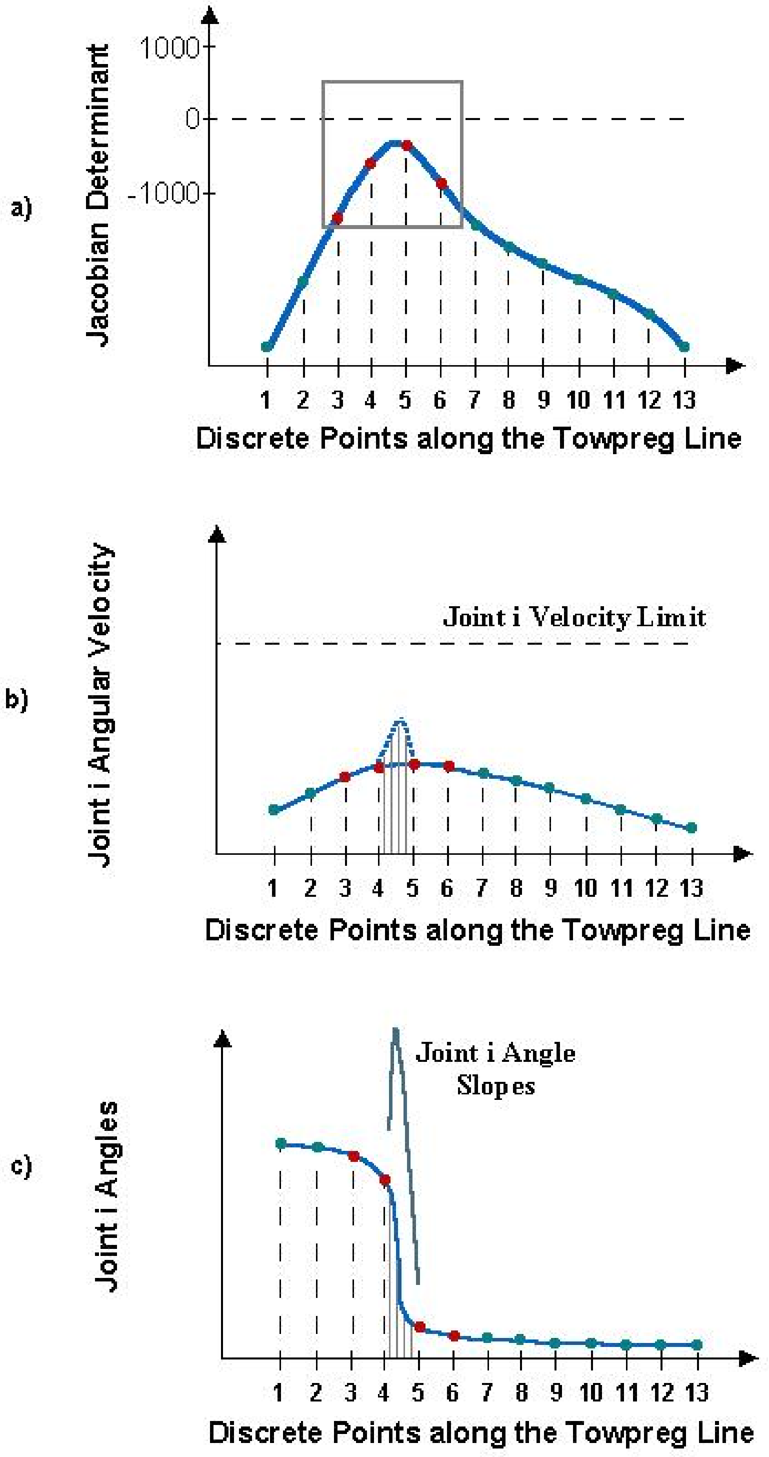


Figure 9.10 a) Locating the critical interval on the determinant curve b) Fine Discrete points all meeting the velocity restrictions c) The angles derivatives show the actual velocity curve exceeding the maximum limits

The algorithm starts on the farthest reachable discrete point on the line and the steps shown in Fig. 9.11 are repeated to find all feasible segments on the line.

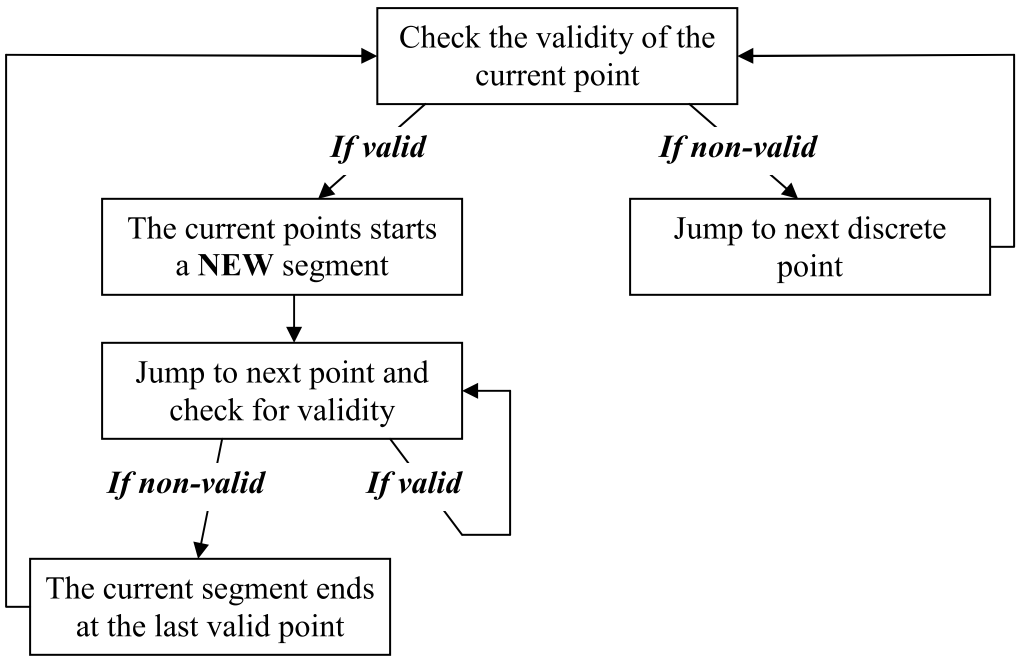

Check the validity of the current point
If valid
If non-valid
The current points starts a NEW segment
Jump to next discrete point
Jump to next point and check for validity
If non-valid
If valid
The current segment ends at the last valid point

Figure 9.11 Flowchart to determine the all feasible segments

The above code is separately repeated for the eight different configuration data sets, and accordingly, the boundaries of all feasible segments in each configuration are stored for later processing. A sample of the stored kinematic data is plotted in Fig. 9.12 and 9.13. This graphical visualization provides a concrete confirmation of the segment feasibility and a tangible verification on the validity of the discrete points where all restrictions limits and thresholds are met.

9.4.3 Collision Detection and Determining Feasible Sub-Segments

To complete the total and absolute path verification, collision detection should be performed on the previously determined valid segments. As previously detailed, the towpreg path trajectories risk collision with one of the robot links or any physical obstacle in the workcell. This occurs when the substrate fixed position intersects with the rotational space of any of the robots links during the fiber layout.

Since the collision detection algorithm uses and processes the results yielded by the previous kinematic checks, the collision logic should be performed separately after the complete execution of the kinematic checks. Once the locations and boundaries of all feasible segments are computed, the collision logic is applied on the yielded valid towpreg segments to finalize and confirm the paths feasibility.

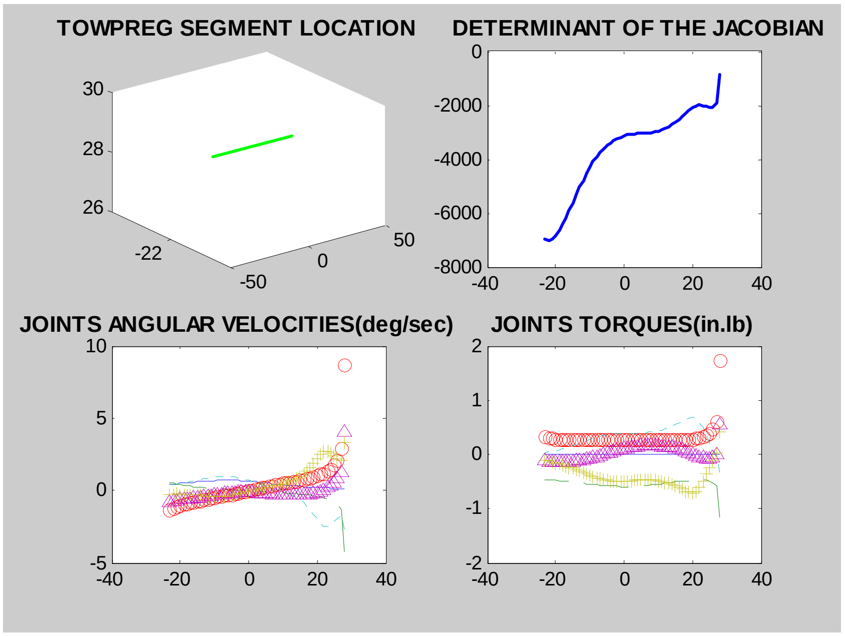


Figure 9.12 The kinematic data along a single valid segment

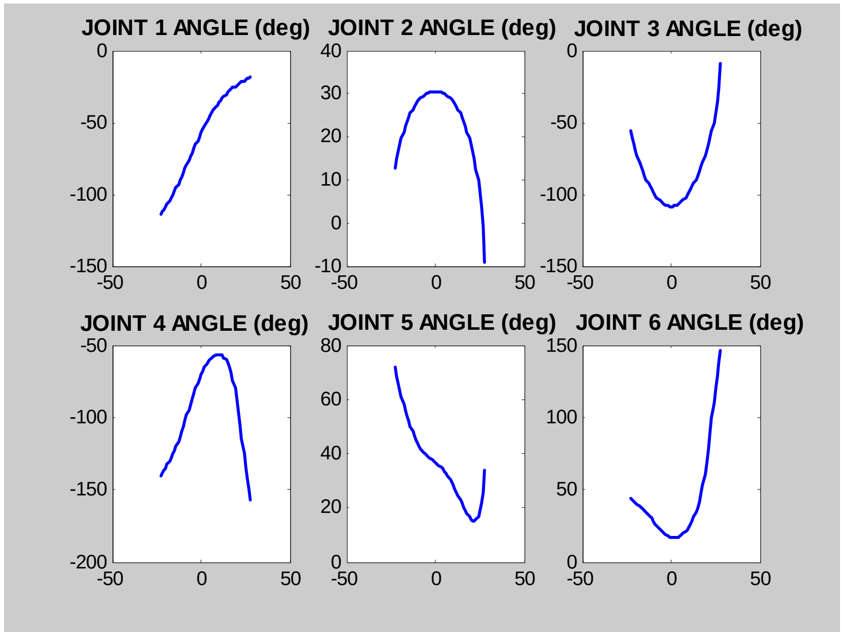


Figure 9.13 The six joints angles along a single valid segment

The algorithm for checking the intersection of single points on the towpreg segments with the robot links is detailed earlier for both the vertical plane assumption and the three-dimensional global workspace problem. The code logic introduced here decides on the possibility of laying the towpreg on the whole length of the segment, and if collision is detected, collision-free sub-segments are determined and located within the original path.

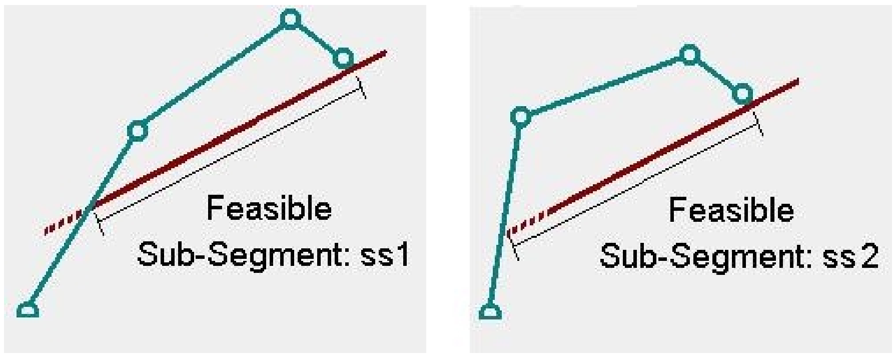
When the algorithm detects collision on a portion of the considered segment, the fiber layout definitely cannot be achieved on the entire segment. The robot links would impact the composite substrate causing the interruption of the towpreg placement. On the other hand, the towpreg layout on shorter sub-segments within the original path definitely meets all kinematic requirements and can possibly be achieved without any risk of collision. Since the purpose of this analysis is to find all feasible towpreg paths, the logic introduced here determines the boundaries of all those completely valid sub-segments.

<u>The collision algorithm</u>:

For all eight robot configurations, every feasible path segments is considered separately. A loop is developed to check first, if collision occurs in laying the towpreg on the whole length of the original feasible segment: EVERY discrete point on the segment is detected for collision for EVERY discrete set of joints angles required for the segment towpreg layout. As just mentioned, the method to detect collision of one single point with the robot links is introduced in the previous chapters. By repeating this elementary detection function independently for all the links discrete locations on the individual discrete points on the towpreg path, the feasibility of the segment is finalized.

If the initial segment is formed by p number of discrete points, then there are p discrete sets of joints angles to reach those points, and accordingly, p^2 elementary detections are required. If none of the p^2 tests detects collision, the segment is considered a completely valid towpreg linear path with no risk of any collision problems. On the other hand, if at least one of the p^2 collision tests detects collision, the segment is still processed to determine feasible sub-segments. The search for feasible sub-segments should be performed point by point on the original segment to locate all different possible paths; the dashed path portion is considered unreachable in Fig.

9.14-a whereas it is a part of a completely feasible sub-segment shown in Fig. 9.14-b. The three-link manipulator is shown here for a better two-dimensional visualization of the process.



(a) (b)

Figure 9.14 Two collision-free sub-segments valid within the original valid segment

The algorithm logic considers every discrete point on the original segment as a starting boundary for a collision-free sub-segment, and accordingly for each starting point, a loop travels on the consecutive discrete points to locate the sub-segment end boundary. The sub-segment with *q* digital points is considered valid and feasible if the q^2 collision checks detect null impact intersections. Figure 9.15 summarizes the algorithm logic to locate the feasible sub-segments: the considered original segment is formed by *p* discrete points labeled from *1* to *p*.

The collision detection algorithm was observed to dramatically affect the code computational time. As already mentioned, the elementary collision check should be performed p^2 times to verify the feasibility of one single towpreg segment. Since each elementary detection involves every surface on every obstacle in the workcell, fine discrete point resolutions can drastically increase the code computational time. As an example, by reducing the digital point spacing from *1* to *¼ inches* on a 50 inches segment, the computational time increases from *41 seconds* to *10 minutes and 38 seconds*. Data were taken on an 866 MHz processor station.

As a final conclusion, for the given desired line parameters, feasible segments are determined by satisfying the robot kinematic restrictions and then collision detection is performed to confirm the validity of the towpath segments or define new collision-free feasible sub-segments.

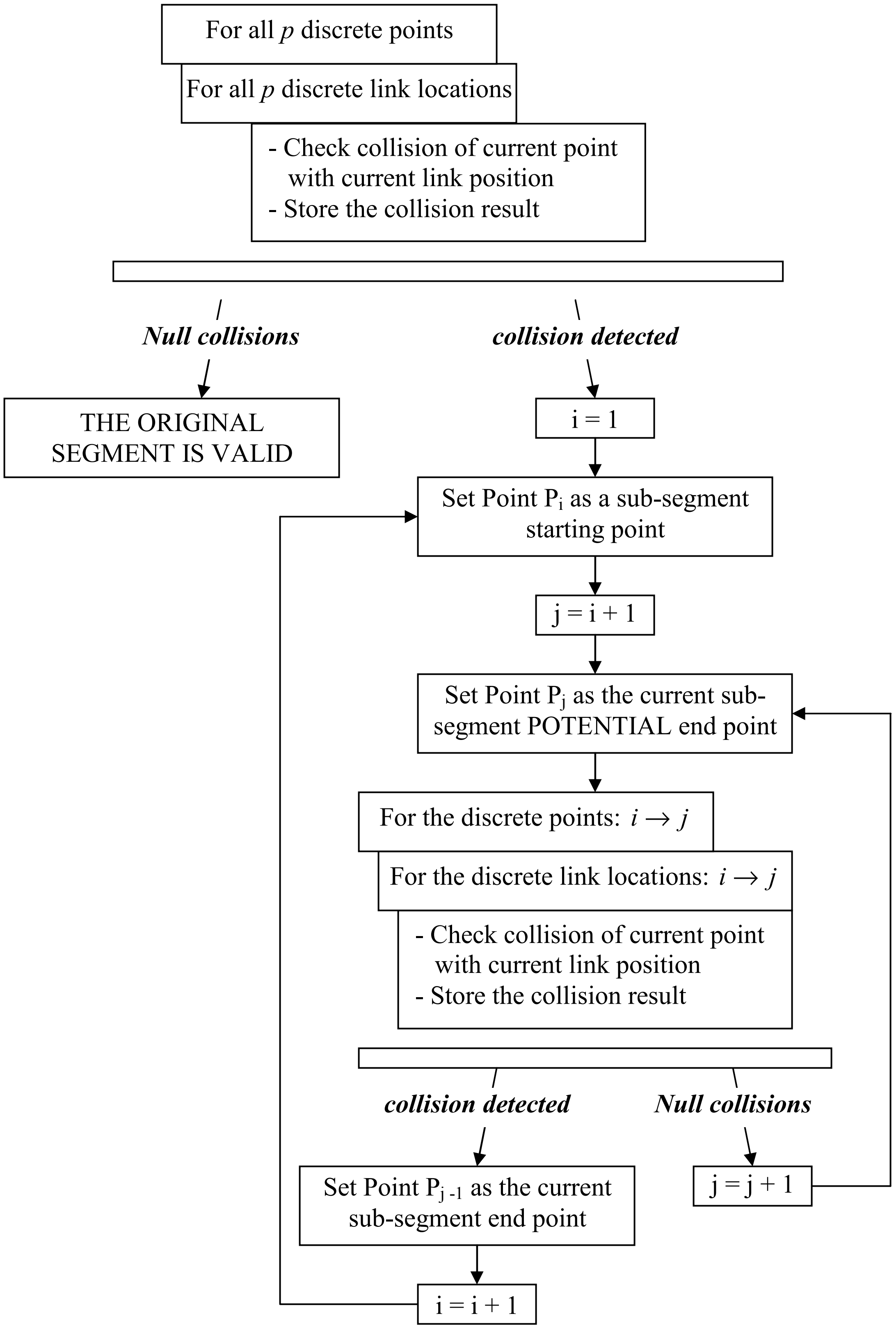

For all p discrete points
For all p discrete link locations
- Check collision of current point with current link position
- Store the collision result
Null collisions
collision detected
THE ORIGINAL SEGMENT IS VALID
i = 1
Set Point P_i as a sub-segment starting point
j = i + 1
Set Point P_j as the current sub-segment POTENTIAL end point
For the discrete points: $i \to j$
For the discrete link locations: $i \to j$
- Check collision of current point with current link position
- Store the collision result
collision detected
Null collisions
Set Point P_{j-1} as the current sub-segment end point
j = j + 1
i = i + 1

Figure 9.15 The algorithm flowchart to determine the feasibility of segments in detecting collision

CHAPTER 10: RESULTS: THE MAPPED FEASIBLE WORKSPACE

The previous chapter discussed the approach to determine all feasible towpreg segments and to locate their boundary points in the robot three-dimensional space. Accordingly, the methods introduced here to map the robot *feasible workspace* for fiber placement involve finding and graphing all these valid segments lying on multiple three-dimensional lines. Therefore, the lines should be arranged in an array or a matrix that positions the valid towpreg paths in specific (and desired) regions in the robot workcell. As a result, the mapped feasible workspace includes all valid towpreg segments located in the concerned space regions and thus can be used later on in the manufacturing verification of composite products.

The analysis introduced here focuses on mapping *two-dimensional* workspaces on chosen planes in the global workspace. Not only do two-dimensional plots provide a clearer visualization of the results, they offer the possibility of analyzing the results and allow graphical comparisons of different plotted data. In addition, decomposing the three-dimensional workspace to multiple two-dimensional planes proved to give insights on the global three-dimensional solutions without having to study and handle complex and often incomprehensible graphs.

Furthermore, the towpreg lines included in the feasible *planar* workspace are chosen to be parallel for visual clarity and clearer analysis. The lines are arranged in a one-dimensional array that positions all lines successively on the chosen plane. The lines are spaced by a chosen resolution decided by the manufacture designer. Since the prepreg tapes are ¼ inch wide, it is practically useless to select a resolution finer than ¼ inch. On the other hand, coarser and larger spacing values reduce the computation time without causing considerable variations in the yielded results. More specifically, choosing the same resolution used to discretize the lines has many advantages in uniformly digitizing the whole two-dimensional plane.

In mapping the feasible planar workspace, the lines carrying the valid towpreg paths should be selected to have the same *yaw* and *pitch* parameters in order to force the generation of parallel lines. In addition, the line's *roll* parameter is restricted to one predefined value: since the towpregs are laid on consecutive parallel paths to cover the planar layer, the surface of the tapes must be parallel to the plane to allow the proper layout angle. Therefore, the line roll parameter should be carefully selected and fixed for all lines on the plane. In addition, the *orientation parameter of the end-effector with respect to the lines* can have any designed value but it should be the same for all lines on the plane. On the other hand, the *line offsets* parameters should be carefully selected and varied to position the lines on the plane and to accurately provide *all* different possible line locations spaced by the chosen resolution.

10.1 Plotting all Valid Towpreg Segments in the Two-Dimensional Plane

As discussed in the previous chapter, all valid towpath segments can be determined and located for the given single line parameters. By varying the line's offsets to position the multiple lines on the desired workspace plane, the algorithm explained in the previous chapter is repeated for every line to determine *all feasible segments in the plane.* The XYZ coordinates of all valid towpreg segment boundaries are yielded and thus, the linear segments are easily plotted to cover the feasible workspace for fiber placement. Figure 10.1 shows all valid segments as a method to graphically present the Merlin robot feasible workspace. The plane shown is horizontal intersecting with the shoulder axis. The lines have zero yaw, pitch and roll.

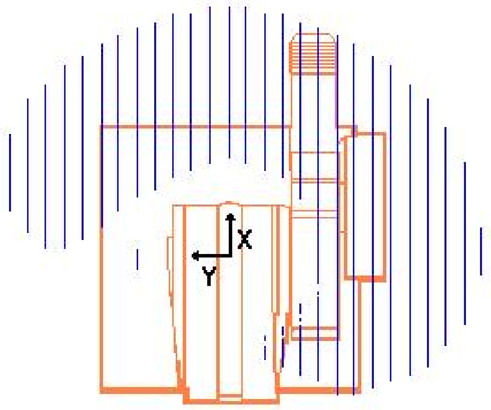


Figure 10.1 The feasible workspace presented by all the valid segments

Presenting the robot feasible workspace with all valid towpreg segments has a strong drawback in visually analyzing the workspace graph: since every line carries many different valid towpreg paths, valid segment portions might overlap and create a longer 'false feasible segment' formed by the multiple separate shorter segments.

If two or more valid towpreg segments (on the same line) are joined or partially overlapped, the newly created segment line *cannot* be considered a valid towpreg path. Even if the original segments are completely valid, the fiber cannot be laid on the whole length of the new long segment. Figure 10.2 illustrates the overlapping problem: towpregs can be perfectly laid on the segment AB, and/or on the other segment CD. Since the two segments are overlapping, *the segment AC is the path that would be (falsely) shown in the feasible workspace graph*. Towpregs cannot be laid on the whole length of the presented segment AC; the compaction head cannot cross point D or point B in the shown directions to cover the whole segment. By crossing those points, the process might be interrupted to allow modifications in the robot configurations.

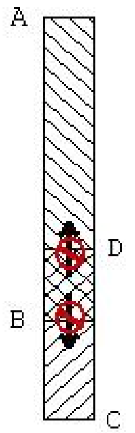


Figure 10.2 Segments overlapping

Presenting the feasible workspace with all valid segments should then be avoided due to the segments overlapping confusions and the *false feasibility determination* of the presented segments.

10.2 Plotting the Longest Valid Towpreg Segments in the Two-Dimensional Plane

The overlapping problem is solved by considering and plotting only one valid segment on each line. The presented feasible segments would preserve their real lengths since overlapping caused by the interference of many segments is impossible. Many advantages favor choosing and plotting the *longest valid segment* on every line to represent the feasible workspace for fiber placement: 1 - The longest valid segment is the most valuable feasible path on the line where the longest towpreg tapes are laid to build the desired structure. 2 - Many shorter valid segments are often located within the longest segment and therefore, the shorter paths boundaries can be ignored with minimal loss of information. 3 - The longest lines plots can be used as an assessment of the robot kinematic capabilities: comparing the longest lines for different planar workspaces allows a better evaluation of the robot performances in placing the towpregs on the considered planar workspaces.

To find the boundaries of the longest feasible segment on each line, a simple algorithm searches for the larger path length on all the stored valid segments. The path boundaries are then used to plot of the feasible workspace.

Figure 10.3 shows the feasible workspace bounded by the longest valid segments on every line. The same valid segments are considered in Fig. 10.1 and Fig. 10.3 but the methods to represent the feasible workspace differ. The shown plane is horizontal intersecting with the shoulder axis. The line's yaw, pitch and roll are all zero (similar to the parameters used in plot the workspace in Fig. 10.1).

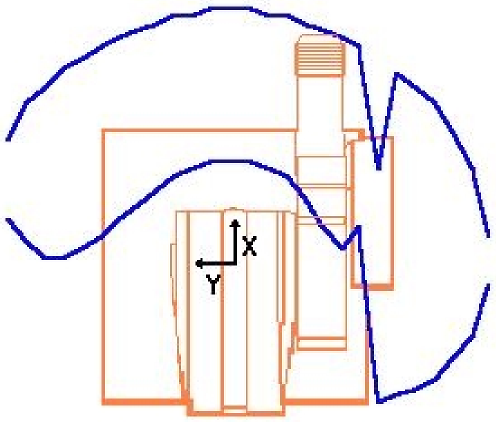


Figure 10.3 The workspace envelope of all longest segments in the plane

The envelope boundaries connect, respectively, the starting and ending points of all the longest valid towpreg paths in the shown horizontal plane. A careful comparison of the two different workspace representations shows that the graph in Fig. 10.3 does not include the many short valid segments plotted in the workspace of Fig. 10.1. On the other hand, the envelope of the longest paths (in Fig. 10.3) sets the 'true' boundaries of the overlapped segments shown with their falsely longer lengths and shifted boundaries in Fig. 10.1.

Since the longest segments envelope only considers the longest paths on each single line in the plane, many shorter and valid segments are located outside or partially outside the envelope and still meet all the feasibility restrictions (Fig. 10.4). The only reason why they are not completely included in the workspace envelope is simply because they are not the longest valid segments on their specific lines.

Accordingly, in connecting the starting or end points of all longest valid segments, the envelope boundary sometimes is generated and restricted to a strange intrusion shape. Even if the V-notch area (in Fig. 10.4) is kept outside the envelope, this excluded area involves many completely valid towpreg paths, not long enough to be fully integrated in the envelope.

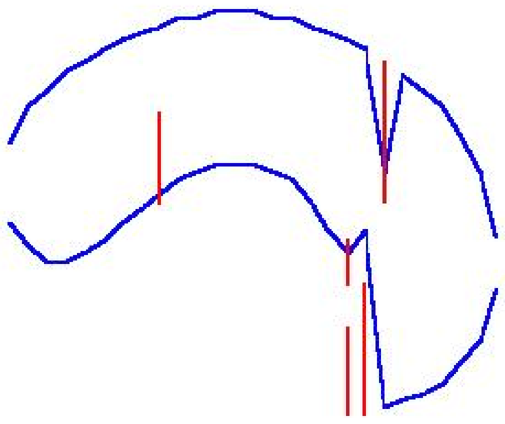


Figure 10.4 Feasible linear paths outside the longest segments envelopes

Although a considerable number of valid segments are not included in the envelope, mapping the longest valid segments is the preferred method considered in the following analysis to represent the robot workspace for fiber placement. As already discussed, this method offers many advantages to assess and study the graphed results.

In introducing the concepts to represent the workspace of the robot, the above discussion uses the XY horizontal plane to show the results and explain the differences between the different approaches. The same techniques can be applied to any plane in the three-dimensional space of the robot. In particular, the planar workspaces are mapped parallel to the base frame YZ and XZ vertical planes.

To find the feasible towpreg paths in the YZ plane, a line location array is defined to position the lines in the desired plane. The lines orientational parameters should be carefully chosen so that the towpreg tape surfaces should lie in the vertical plane. As an example, for a zero yaw angle, the pitch value should be set to 90 degrees and the roll to null. The generated lines are *vertical* and spaced by varying the Y-offset parameters.

On the other hand, to consider all *horizontal* lines on the same YZ plane, the parameters should be modified: the yaw and roll angles are respectively set to 90 and -90 degrees whereas the pitch value should be changed to null. The generated horizontal lines are spaced by the Z-offset instead of the Y-offset used earlier. Figure 10.5 shows the feasible workspace envelope in the YZ (vertical) plane distanced 25 inches from the robot base. The lines considered in the presented workspace are all horizontal.

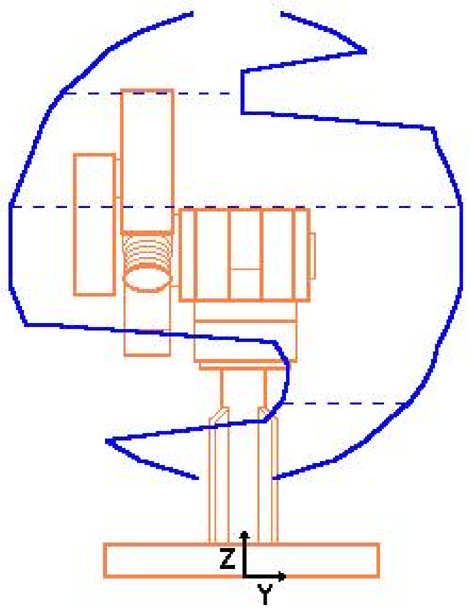


Figure 10.5 The longest segments envelope in a vertical plane parallel to the YZ axes

As previously discussed, the algorithm should be able to determine feasible segments on all lines with different orientations and locations. Lines lying in vertical planes parallel to the base frame XZ axes are now considered. Here again, the line parameters should be carefully selected to insure that the prepreg tape surfaces lie in the desired plane. *Horizontal* lines are spaced by incrementing the Z-offset parameters; the pitch and yaw angles should be null and the roll angles should be set to 90 degrees.

On the other hand, *vertical* lines within the same XZ vertical plane are located with the X-offset parameter. The lines have a pitch equal to 90 degrees and a yaw set to null. To place the prepreg tape surface normally to the plane, the roll angle should be set to 90 degrees. Figure 10.6 presents the effective workspace of the robot on a plane parallel to the base frame XZ axes. This vertical plane is distanced 20 inches from the base of the robot, and the lines generated are chosen to be vertical.

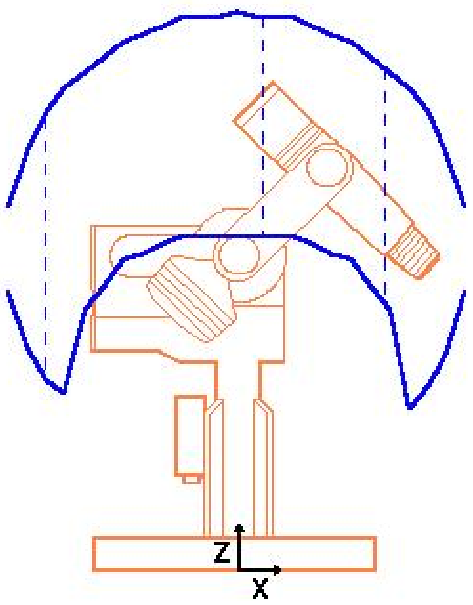


Figure 10.6 The longest segments envelope in a vertical plane parallel to the XZ axes

10.3 The feasible workspace variations with respect to the line parameters

Since the feasible workspace is presented in two-dimensional plots, the variation tendencies of the envelopes boundaries are easily studied and visually analyzed versus the different variables or parameters in the process. Multiple planar workspaces can be clearly plotted and compared on the *same* figures and accordingly, helpful data can be retrieved from the graphs to facilitate the process design analysis. Although the next sections present a detailed direct application on the workspace variations analysis, the following lists all kinematic variables that affect the envelope boundaries.

The trajectory lines parameters (discussed in chapter 8) involve all process variables that alter and influence the feasible workspace boundaries; the effective envelope varies for different parameters to provide and set kinematic rules on the manufacture design:

10.3.1 The Orientation of the End-Effector with respect to the Tow Segment

As stated in the previous chapter, the orientation of the compaction head with respect to the towpreg path greatly influences the geometric configuration of the end-effector. As an example, small orientations of the end-effector might cause an unexpected impact of the compaction head functional components with the substrate; in this case, the end-effector would be positioned or oriented very closely to the towpreg path, initiating many other contact points with the substrate. Consequently, the end-effector geometrical configuration should be cautiously designed in accordance with the desired end-effector orientation with respect to the towpath. In addition, as explained in chapter 7, this orientation parameter also affects the capability of the air cylinder to support the varying pressures.

To analyze the influence of the end-effector orientation on the feasible workspace, many valid segments envelopes are mapped on the same plane, all with similar lines parameters but with different compaction head orientation parameters. Figure 10.7 shows three different workspace envelopes drawn on the same horizontal plane, 10 inches below the shoulder axis. The segments are developed with similar line parameters: the yaw, pitch and roll angles are all set to zero. By fixing the elevation of the plane, the derived offsets locate the towpaths on identical positions. On the other hand, the orientation of the end-effector is the only varying parameter causing the shown envelope differences in the figure.

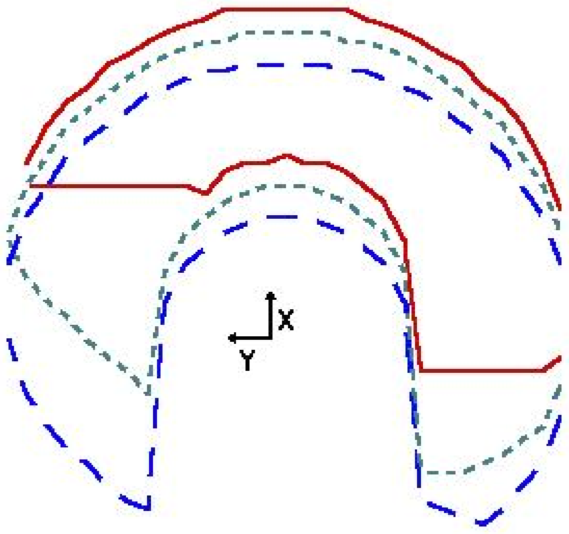


Figure 10.7 The workspace envelopes for 90° (solid), 115° (dotted) and 140° (dashed) end-effector orientation parameter

Not only does the end-effector orientation parameter change the size of the workspace envelopes (as shown in Fig. 10.7), different end-effector orientation parameters can also shift or move the whole envelope in the direction of the towpath lines. Figure 10.7 illustrates a gradual reallocation of the envelopes line boundaries for the three different end-effector orientations. For 90 degrees orientation, the solid line boundary is the farthest from the base of the robot; by increasing the angle to 115 and 140 degrees, the envelope is shifted towards the robot along the generated lines (parallel to the X-axis).

As a result to these observations, a geometrical rule is defined to control and shift desirably the feasible workspace envelope. For any orientation angle of the end-effector with respect to the towpath, the wrist center workspace has its own *fixed reachable* spatial envelope. However, by setting different orientation angles for the compaction head, the position of the roller-towpreg contact point with respect to the wrist center is changed, causing the appropriate shift in the contact point reachable workspace and consequently in the observed feasible workspace.

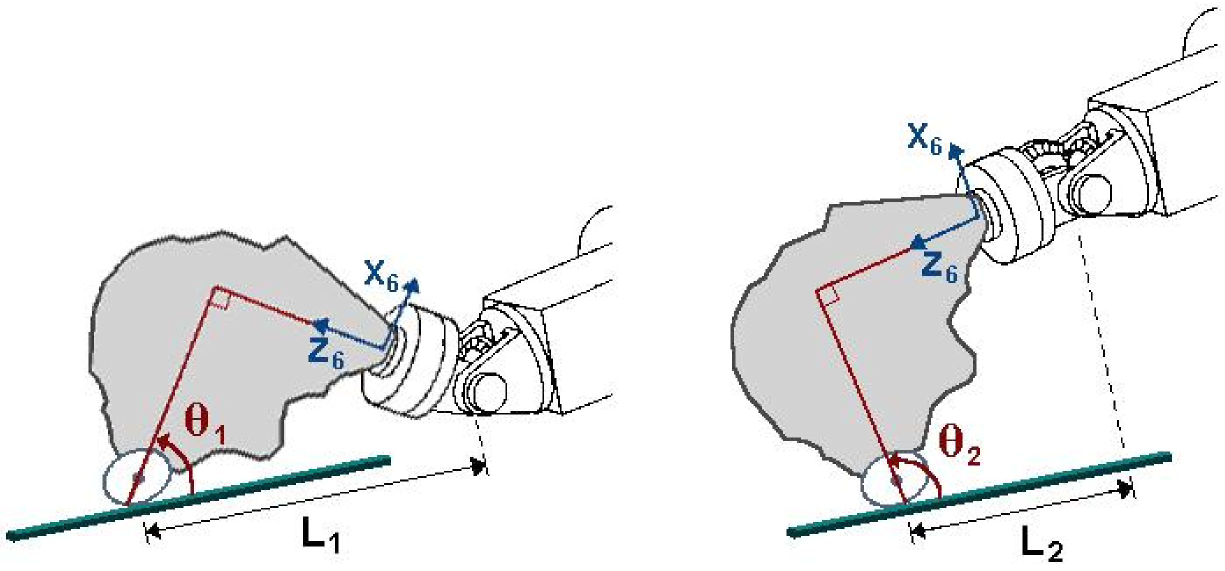


Figure 10.8 The location of the contact point with respect to the wrist

Figure 10.8 illustrates the position of the contact point with respect to the wrist. For the two different orientation angles, θ_1 and θ_2, the distances between the contact point and the respective projection of the wrist center, L_1 and L_2, differ considerably. Since the reachable envelope of the wrist center is fixed for any end-effector orientation, roller_1 workspace is shifted a distance L_1 along the direction of the towpreg line, whereas roller_2 envelope is moved considerably less (a distance L_2).

10.3.2 The Offsets Parameters

The offsets parameters obviously affect the feasible workspace envelope since their main function is to position the lines in the three-dimensional plane of the robot; the valid segments on different line locations are dissimilar in length and evidently in position.

As previously mentioned, every line is space has two offset parameters. Since one offset is used as a variable to create the multiple lines on a plane, the other is fixed to specify the location of the plane. Accordingly, changing this latter offset would generate many parallel planes and would allow a comparison study of all the parallel workspaces plotted on one figure.

Analyzing the segments envelope on many successive and parallel planes offers an insight on the variation of the feasible workspace of the robot in the direction normal to plane (Fig.10.9). This analysis would assist in positioning the substrate in the three-dimensional space of the robot. Accordingly, the manufacture feasibility of a specific structure is verified only when the effective workspace envelope of the chosen plane (*i.e.* chosen offset) should be large enough to include the whole surface of the desired product structure.

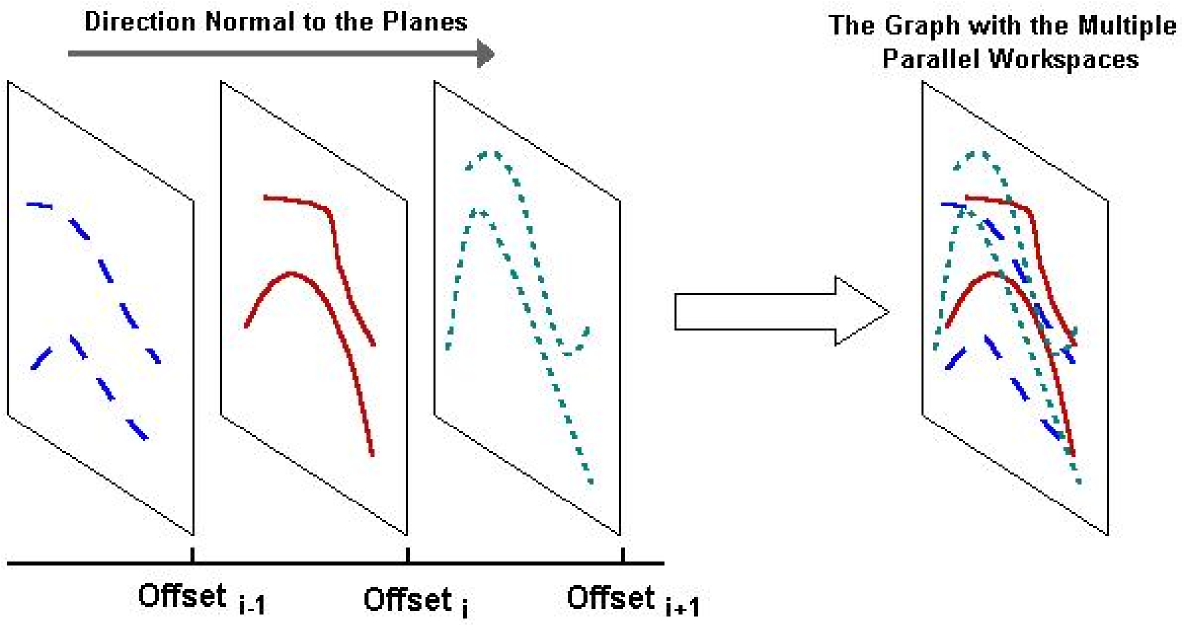

Direction Normal to the Planes
The Graph with the Multiple Parallel Workspaces
Offset $_{i-1}$
Offset $_i$
Offset $_{i+1}$

Figure 10.9 Parallel workspaces plotted on the same figure

To present and graph the envelopes variations induced only by the plane offset parameter, the towpaths orientation angles should be set to fixed values to generate parallel lines and accordingly parallel planes. The end-effector orientation should be unchanged (in order to prevent variations induced by this parameter) whereas the plane offset is varied to position the parallel planes in the desired locations. Figure 10.10 shows three workspace envelopes on vertical planes parallel to the base frame YZ axes. The X-offsets for three planes are 25, 35 and 40 inches. The lines pitch is set to null while the yaw and roll angles equal respectively 90 and

-90 degrees. The end-effector orientation parameter is fixed at 140 degrees for all the generated lines.

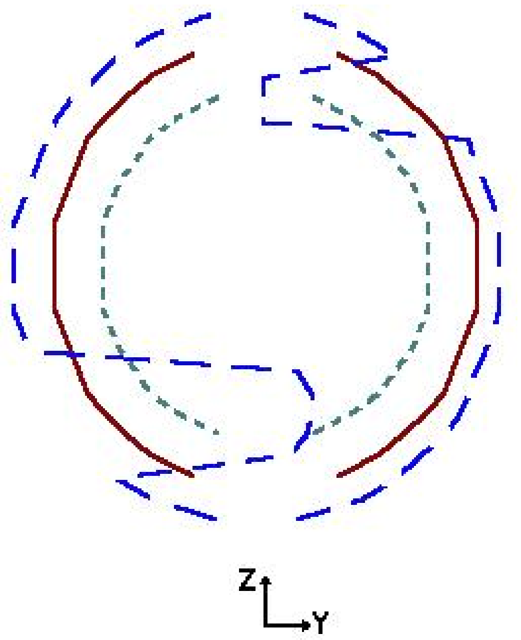
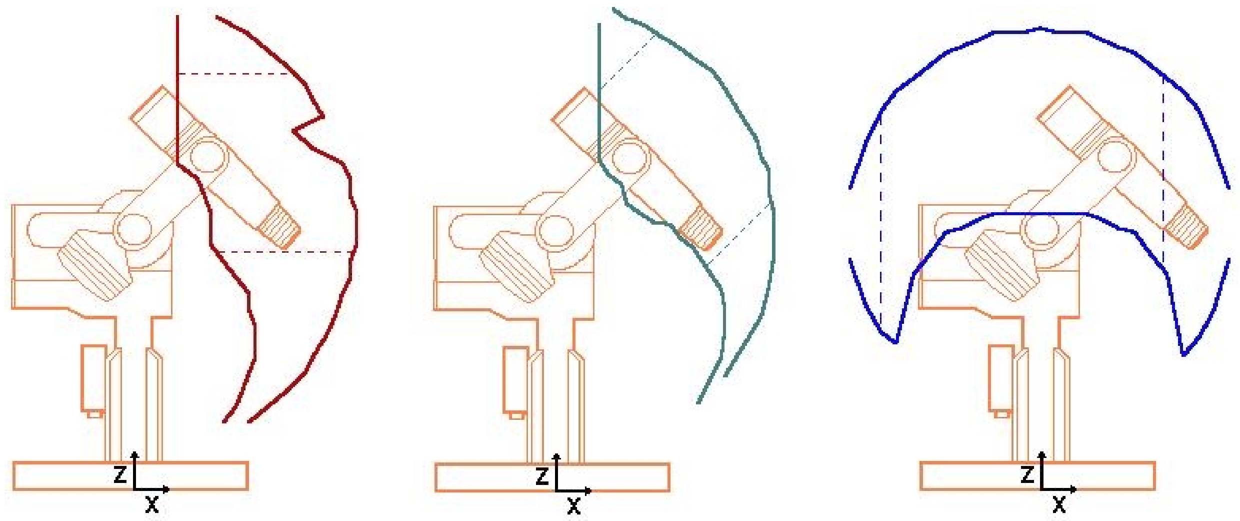


Figure 10.10 The parallel vertical envelopes for 35 in.(solid), 40 in. (dotted) and 25 in.(dashed) X-offsets

10.3.3 The Line Yaw, Pitch and Roll Orientation Parameters

Even though different line *roll* angles can greatly influence the size or location of the plotted feasible workspace, this variation cannot be plotted nor visually analyzed: since the surface of the towpreg tape is required to be parallel to the corresponding layout plane, tapes with different roll angles are placed on different intersecting planes. Accordingly, the multiple workspaces generated on each of the intersecting planes cannot be plotted on the same graph and thus, the variation analysis is visually impossible.

On the other hand, the workspace variations induced by the yaw and pitch parameters can definitely be plotted and analyzed on the same planar graph. The yaw or pitch angles are considered as the *inclination* of the lines: by fixing one of the two angles, the line inclination is controlled by the other orientation parameter. Figure 10.11 illustrates the variation of the workspace envelope when the lines inclinations are changed. In the shown vertical plane (Fig. 10.11), the pitch angle is varied to provide the different workspaces whereas the yaw angle is fixed and set to null. The vertical plane is distanced 20 inches form the base of the robot. The pitch angles used are respectively 0, 45 and 90 degrees in Fig. 10.11-a, b and c.



(a) (b) (c)

Figure 10.11 The workspace envelopes for (a) 0˚, (b) 45˚, and (c) 90˚ line pitch angles

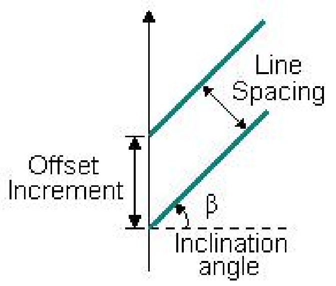


Figure 10.12 Inclined lines Offset

To insure the correct and desired spacing between the inclined lines, equation 10.1 computes the parameter offset differences to accurately locate the lines on the plane. The inclination angle *β* is either the yaw or the pitch parameter, depending on the direction of the considered plane.

$$Offset\_Increment = Line\_Spacing / \cos(\beta) \qquad (10.1)$$

10.4 The Feasible Layout on Squared Surfaces

The feasible one–dimensional analysis is extended to involve two-dimensional feasible surfaces. Since all valid towpreg segments are already determined, regions can be located in the planar workspaces where the fiber layout is completely valid on the entire specific surface. These feasible areas can be defined as two-dimensional workspace sections where all its constituent towpreg paths are verified and valid.

The two-dimensional feasibility analysis allows a faster and advanced approach to verify the manufacturing validity of many product bodies. All cross-sections areas of the candidate structure are checked with the determined feasible surfaces, and accordingly, fabricating the whole product would eventually depend on the size and location of the correspondent valid regions in the workspace.

Specifically, squared feasible surfaces are considered as the obvious two-dimensional extension to the valid towpreg trajectory analysis. As already mentioned, these square areas would be built by closely arranging multiple *valid* towpreg trajectories. The square side length is accordingly determined by counting the number of adjacent feasible towpaths, all required to be long enough to cover the whole squared area: the side length a, is computed analytically in multiplying the adjacent towpath number by the towpreg width. To satisfy the two-dimensional surface requirements, each considered feasible segment should be longer than the computed square side length a.

<u>The algorithm to locate the squares</u>

Since all feasible towpreg segments are previously determined and located on the chosen plane, the approach to find the valid square areas is based on the computed boundaries of every valid segment. The search for valid square surfaces is accordingly limited to the original chosen plane where all needed data is already yielded and available to be processed.

A Matlab algorithm (Fig. 10.14) is developed to search for feasible square areas in the planar workspace. The square size, along with the spacing resolution should be entered and used to find the valid squares. The line's inclination on the other hand, would define the inclination of the square surfaces. The code logic considers each feasible segment on the plane, as a potential towpreg candidate to form the entire square surfaces. Segments shorter than the square side are filtered out to save computational time. As shown in Fig. 10.13, the feasible segment bounded by the two points P_7 and P_8 is not long enough to cross the whole length of square S_3 and consequently, should be ignored.

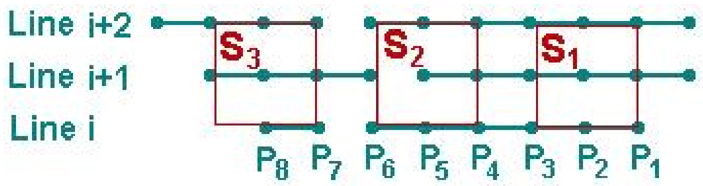


Figure 10.13 Locating square candidates and determining feasibility

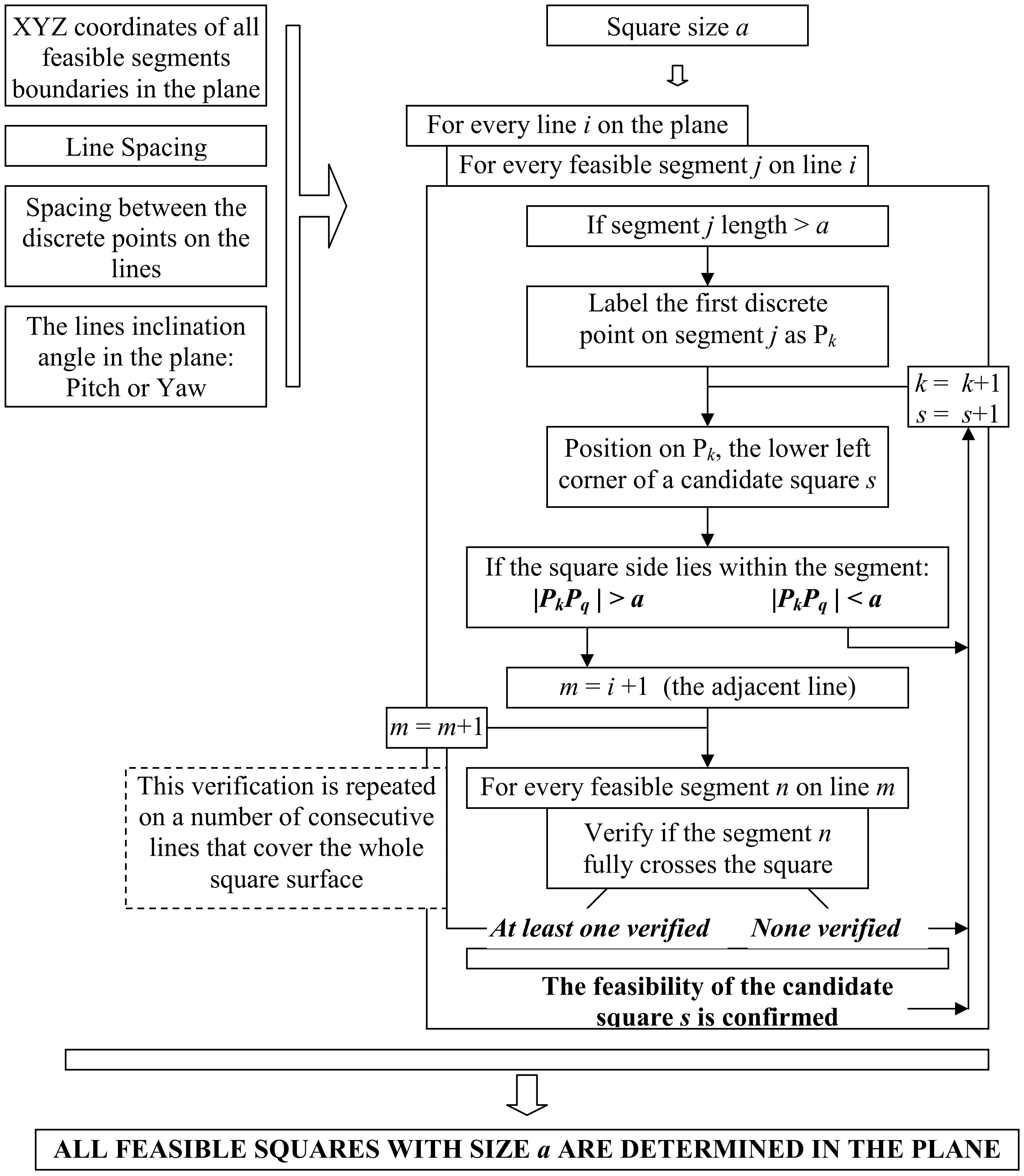

XYZ coordinates of all feasible segments boundaries in the plane
Line Spacing
Spacing between the discrete points on the lines
The lines inclination angle in the plane: Pitch or Yaw
Square size a
For every line i on the plane
For every feasible segment j on line i
If segment j length > a
Label the first discrete point on segment j as P_k
$k = k+1$
$s = s+1$
Position on P_k, the lower left corner of a candidate square s
If the square side lies within the segment:
$|P_kP_q| > a$
$|P_kP_q| < a$
$m = i+1$ (the adjacent line)
$m = m+1$
This verification is repeated on a number of consecutive lines that cover the whole square surface
For every feasible segment n on line m
Verify if the segment n fully crosses the square
At least one verified
None verified
The feasibility of the candidate square s is confirmed
ALL FEASIBLE SQUARES WITH SIZE a ARE DETERMINED IN THE PLANE

Figure 10.14 The algorithm flowchart to determine the feasible squares in the desired plane

The algorithm positions feasible square candidates on EVERY discrete point of the feasible segments and checks the validity of each considered square. The squares S_1 and S_2 (in Fig. 10.13) are located on the same valid segment [P_1 P_6], but on different discrete points. Square

S_1 is completely valid, since the adjacent lines (line $_{i+1}$ and line $_{i+2}$) hold segments that fully cross the square. On the other hand, square S_2 is unfeasible, as line $_{i+1}$ does not carry feasible segments at the needed locations.

Since the locations of all feasible square areas are determined and computed within the workspace, the four sides enclosing every square are easily plotted to offer a visual representation of the results. Figure 10.15 shows the generated feasible squares on a vertical plane parallel to the base frame YZ axes. The plane is 25 inches distanced from the center of the robot. The line inclination (or pitch) angle is set to null while the yaw and roll angles equal respectively 90 and - 90 degrees.

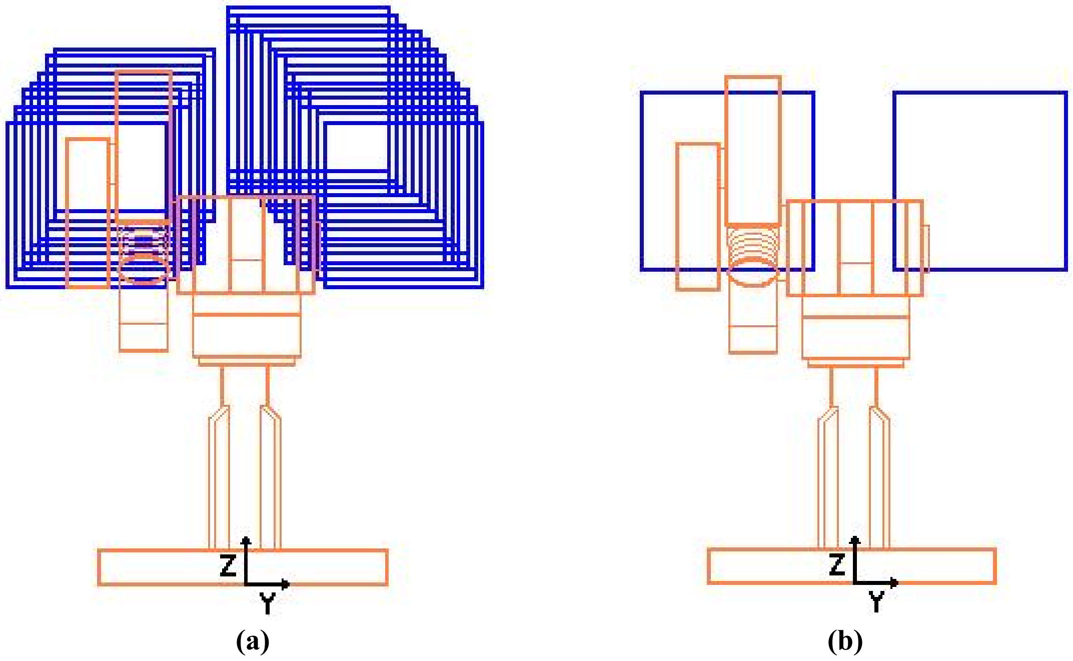


Figure 10.15 The feasible squares in the vertical plane

All 20-inch feasible squares are plotted in Fig. 10.15-a, while the *biggest* valid squares for the above line parameters are 22 inches, located in the vertical workspace in Fig. 10.15-b. As noticeable from the graphs, the location of smaller squares is much more diverse with a much larger frequency.

Inclined feasible squares can also be plotted in the two-dimensional workspace when inclined towpreg lines are considered in the planar analysis. The same algorithm is used to search for the valid square areas and to locate them in the effective workspace. Figure 10.16 shows 45 degrees inclined feasible squares. The planar workspace is 20 inches distanced from the robot base and the considered line's pitch is set to 45 degrees. Here again, the 17-inch squares shown in Fig. 10.16-b are more frequent and diverse than the larger 18-inch squares in Fig. 10.16-a.

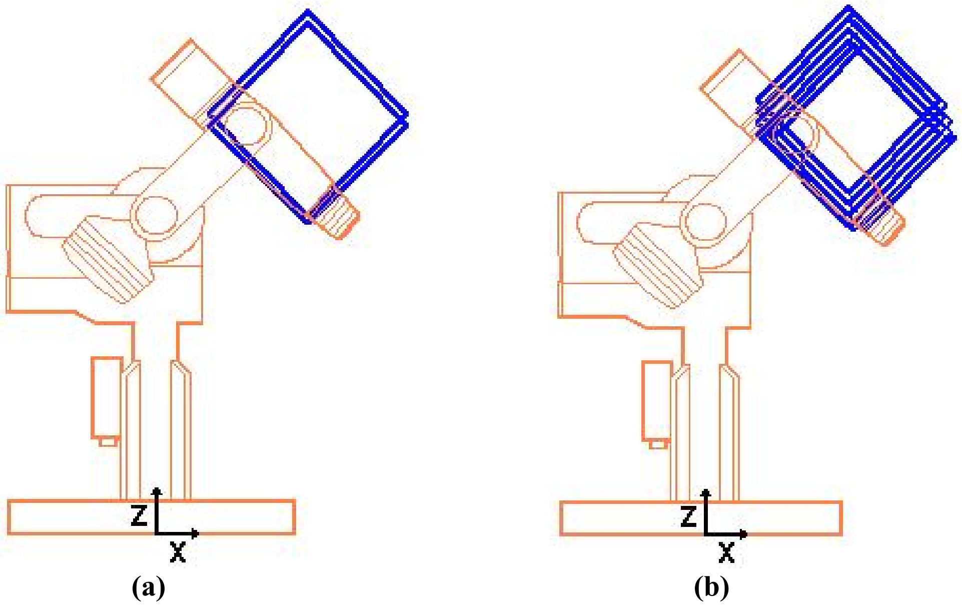
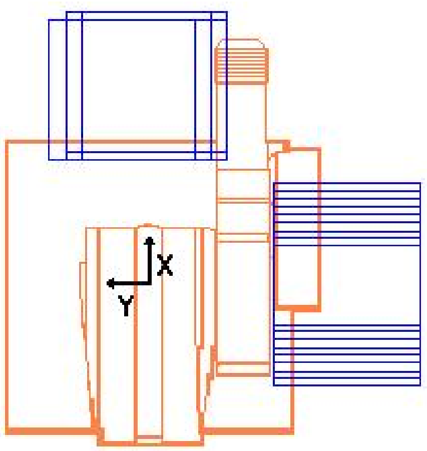
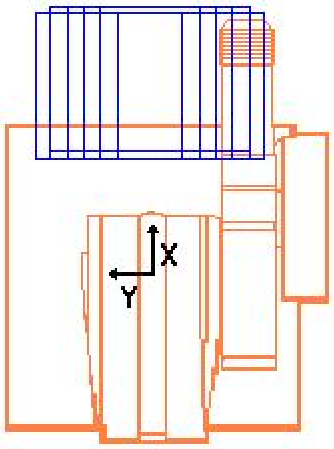


Figure 10.16 The inclined feasible squares in the vertical plane parallel to the XZ axes

The previous section studied the changes in the feasible workspace when the different process parameters are varied. Similarly, these variable parameters have the same influence on the size and location of the valid feasible squares areas, and accordingly, analyzing these variations allows a better process and fabrication design. Figure 10.17 shows dramatic differences in the square locations when the elevation of the considered horizontal planes is changed. The squares have the same size (18 in.) in the two shown workspaces but their positions with respect to the robot differ greatly. The planar workspace in Fig. 10.17-a is 10 inches lower than the plane shown in Fig. 10.17-b. These variations in the feasible regions locations affect directly the position of the substrate in the robot workcell.

By plotting the feasible squares as well as the longest segments envelope (discussed earlier) on the same planar workspace, it is observable that some squares are not completely positioned within the envelope boundaries (Fig. 10.18). The envelope generated only covers the longest feasible segments and there are definitely shorter valid towpaths *outside* the envelope that could take part in building the feasible squares. Consequently, since all verified segments are considered in determining the feasible square areas, the envelope may eventually not cover all valid squares in the planar workspace.





(a) (b)

Figure 10.17 The feasible squares for different workspace elevations

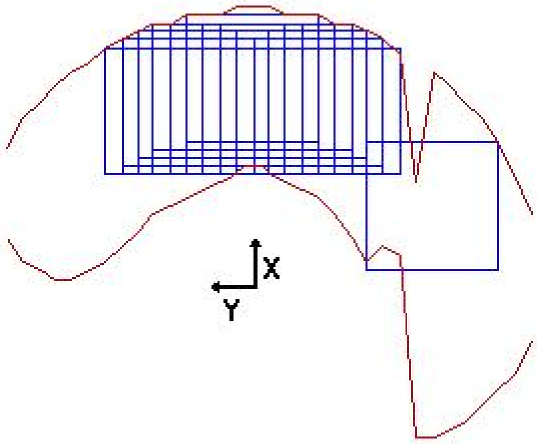


Figure 10.18 The longest segments envelope and the feasible squares on the same planar workspace

CHAPTER 11: FIBER PLACEMENT FOR THE FABRICATION OF ISOTROPIC FLAT COUPONS

As the effective workspace analysis determines and locates all possible and valid towpreg paths in the robot workspace, the fabrication of isotropic flat coupons is considered as a direct application of the feasible trajectory method. The layout of all towpregs forming the coupons structure is verified on their three-dimensional trajectories and accordingly, the size and position of the composite body are determined in the robot workcell.

11.1 Isotropic composite structures

One of the most attractive properties of composite products is characterized by their high strength-to-weight and high stiffness-to-weight ratios. These distinctive features dominated a wide range of manufacturing fields and allowed the use of composite material parts in many critical industrial areas. However, some important considerations should be met throughout the manufacturing process to achieve these crucial properties.

In the fiber placement process, towpreg tapes are placed adjacently to minimize the voids on the formed surface. Consequently, the produced laminate ply (or surface layer) is highly unidirectional since all its constituent fibers are parallel, creating high strength properties along the towpreg directions. On the other hand, the same layer withstands minimal loads when the forces are applied perpendicularly to the towpreg orientations. Figure 11.1 illustrates the loads applied on the laminate ply. The shown layer can resist high longitudinal forces but fails when large perpendicular loads dislocate the constituent towpregs.

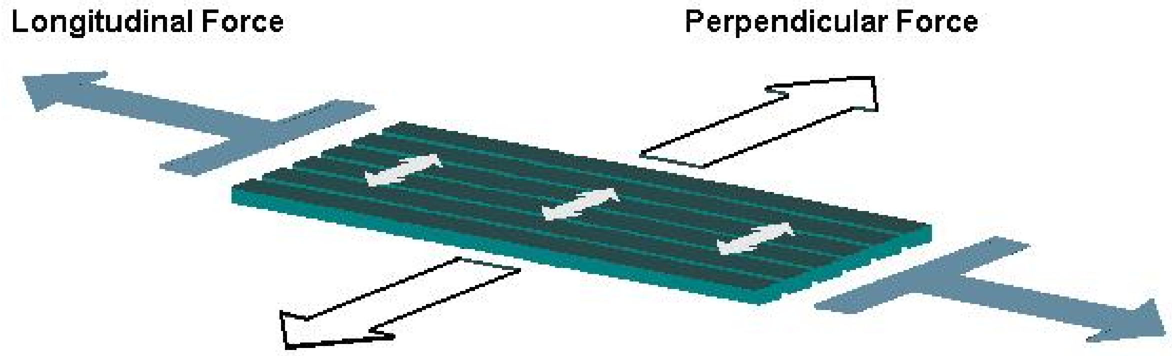


Figure 11.1 Longitudinal and perpendicular loads applied on the laminate ply

As a result, single layer resistance strength is directionally limited to withstand loads only along its constituent towpreg orientation and accordingly, mechanical failure is likely to occur when considerable large forces are applied perpendicularly to the fibers lines. Alternatively, isotropic properties are possible when several layers with different orientations are stacked to form the final structure. Every single unidirectional layer provides the needed strength along its towpreg directions, and by piling or assembling many layers with many different orientations, the final structure would uniformly resist loads in all directions.

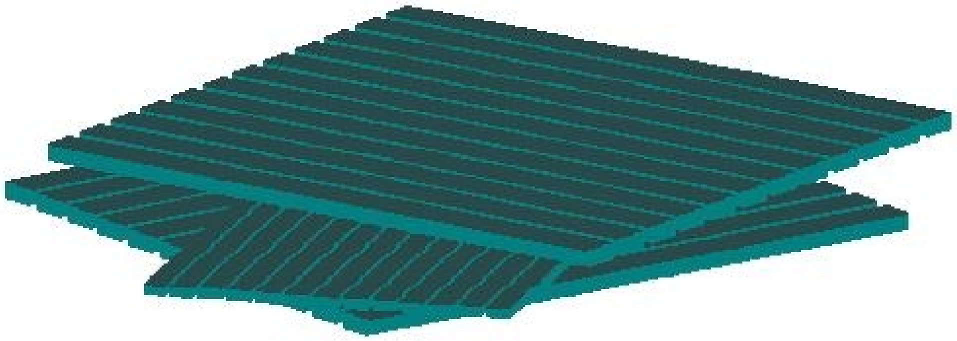


Figure 11.2 Stacking layers with different orientations

The feasible square areas in the robot workspace were discussed in the previous chapter; as already mentioned, each square is built by adjacent *parallel* valid towpaths. Since feasible squares with *different inclinations* can be located in the desired planar workspace, the manufacture of isotropic structures can be verified by considering every feasible square as a layer forming the total body. Accordingly, the squares represent the cross-sectional surfaces of the solid structures, and thus, all squares with different inclinations correspond to the final structure layers, each having its specific orientation with respect to the product to build.

Figure 10.16 in the previous chapter locates inclined feasible squares in the desired planar workspace. By simulating lines with different inclinations, more valid squares with many various orientations can be also plotted (and located) on the *same* plane. To create a three-dimensional isotropic structure, squares positioned exactly in the same regions of the plane should form the different layer of the composite body. Fig.11.3 shows a sample of feasible squares with zero and 45 degrees inclination all located on the same plane. Squares with different inclinations can either be completely overlapped, partially overlapped or not intersecting.

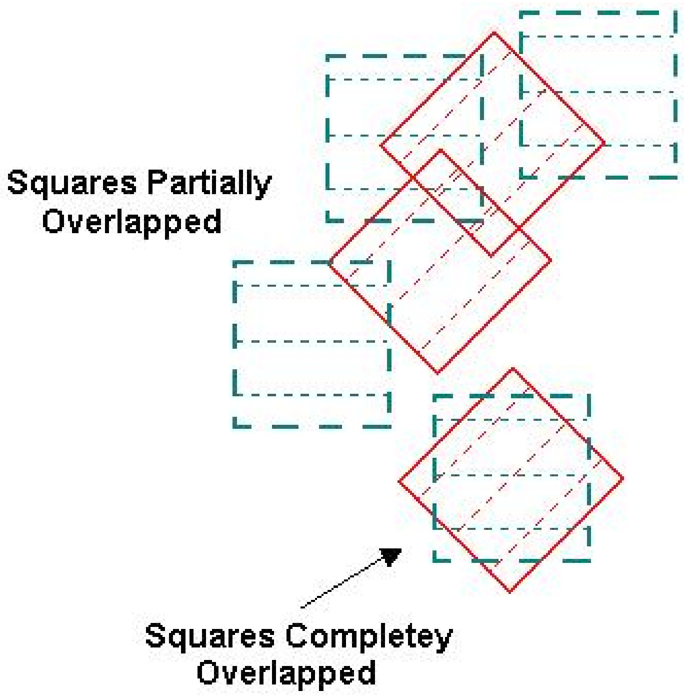


Figure 11.3 Overlapping of squares with different inclinations on the same plane

To determine the complete overlapping of two squares with different orientations, the XYZ coordinates of the square centers should theoretically be exactly equal. However, as previously discussed, the workspace is discretized to points spaced by a chosen resolution, and accordingly, the square centers are positioned on the grid-points in the workspace. Therefore, the coordinates of the square centers are rarely equal even when the considered squares are completely overlapping. To solve this discretization problem, a tolerance number should be chosen and used to practically decide on the overlapping of squares with different orientations. Even though the tolerance number greatly depends on the dimensional accuracy of the final product, the tolerance used in the following analysis is set equal to the digital spacing or resolution.

The feasible square in Fig. 11.4 is centered at point P_i while all discrete points P on the path trajectory are spaced by the selected resolution. To check squares overlapping, a *tolerance window* centered at point P_i is generated with sides equal to the resolution or the tolerance value. If the center c_1 of a square with different inclination lie within the window, the two squares are considered completely overlapping. On the other hand, if the square center c_2 is located outside the window, the squares positional offset exceeds the tolerance values and consequently, complete overlapping does not occur.

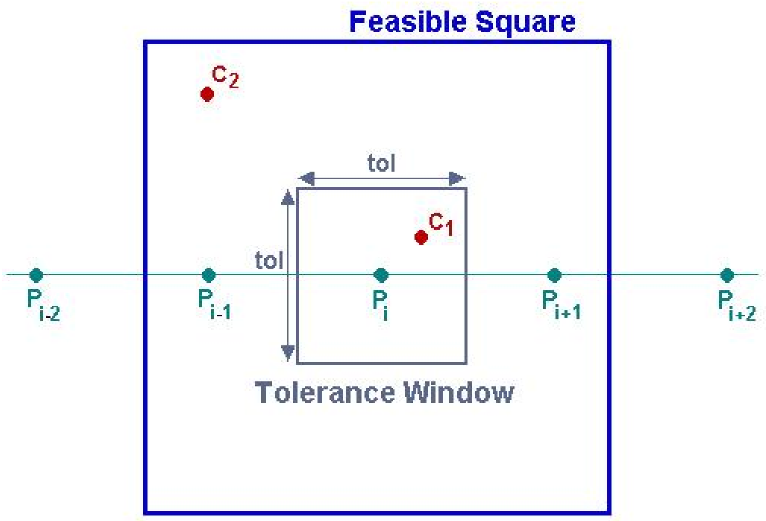


Figure 11.4 The tolerance window to check the complete overlapping of squares

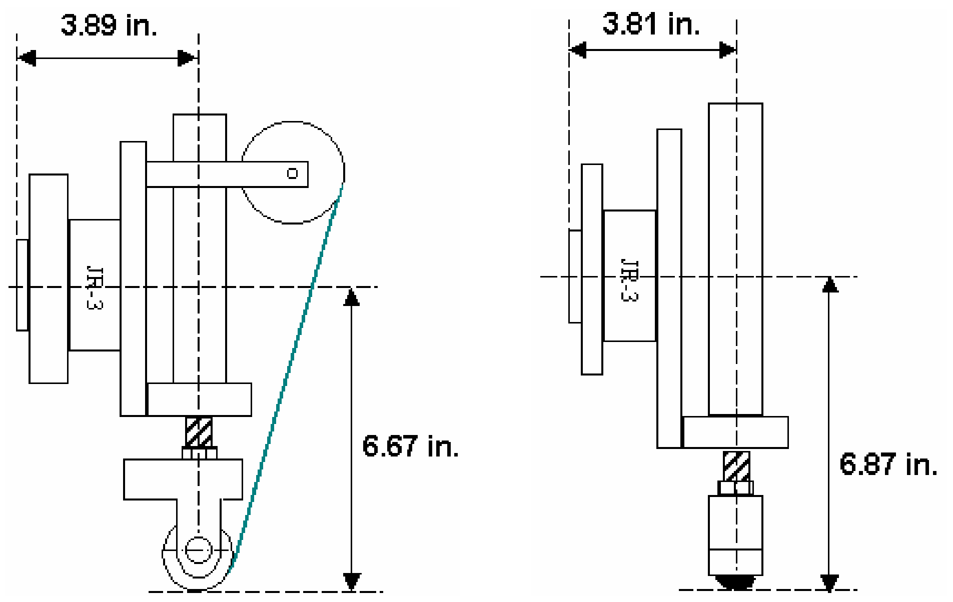
11.2 The Manufacture of Isotropic Flat Coupons

The feasible workspace method is applied and tested for the manufacturing of isotropic flat coupons. The task involves locating the layout process in the robot workspace and determining the biggest valid coupons.

For that purpose two different end-effectors were built for the fabrication testing process. A compaction roller provides the fiber layout contact point on one end-effector, while a compaction ball is mounted on the other. The heating and cutting functional components were not included in the compaction head configuration since they cannot affect the kinematic or dynamic testing problem; on the other hand, an air cylinder and a towpreg feed roller were assembled to provide respectively the required compression force control and the fiber feeding system. Figure 11.5-a and b respectively illustrates the end-effectors configurations and presents the two measures S_1 and S_2 (already mentioned) that locate the contact point with respect to the faceplate center. Appendix E shows pictures of the end-effectors in the robot workcell.

A horizontal table fixed in front of the Merlin robot represents the fiber placement substrate. The table horizontal surface is limited in a rectangular area (36.5 x 30.5 in^2) and is located 18 inches lower than the shoulder axis. Consequently, since the three-dimensional workspace is reduced to the plane of the table horizontal surface, some of several process parameters can be fixed to insure the proper towpreg layout:

- The *Z-offset* parameter for all towpreg line paths should be fixed to –18 inches. The generated fiber trajectories are thus positioned within the considered table surface plane.
- The lines *pitch* and *roll* parameters should be set to zero to allow the simulation of horizontal towpreg tapes normal to the table surface.
- On the other hand, proper variations of the lines *yaw* would provide the desired inclinations of the fibers direction.



(a) **(b)**

Figure 11.5 The end-effectors configurations with the measures: S_1 and S_2

Four layers with different orientations are stacked to form the flat isotropic coupons to fabricate. Each laminate ply should specifically be 45 degrees oriented with respect to the adjacent layer direction. In setting the first layer orientation to any random angle γ_1, the consecutive three laminate ply should respectively have $\gamma_2 = \gamma_1 + 45°$, $\gamma_3 = \gamma_2 + 45°$, $\gamma_4 = \gamma_3 + 45°$ orientation angles. Accordingly, to provide the *correct inclinations* for the parallel towpaths forming the layer, the lines *yaw* angles should be set equal to the corresponding laminate ply orientation γ_i. Squares for each γ_i orientation can then be determined to enclose the valid inclined areas. Consequently, by finding and stacking four *completely overlapped* squares having the four different orientations $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, an isotropic solid structure is built.

However, by stacking the four squares, the common region covered by the layers is bounded by a circle that would set the limits for the cross sectional area of the final isotropic structure (Fig. 11.6). Accordingly, the final composite product can be described as a flat isotropic circular coupon with four different layer orientations.

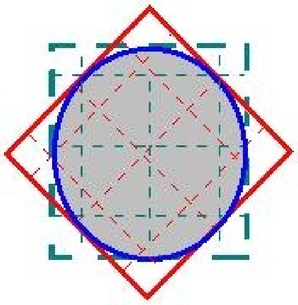


Figure 11.6 The circular area covering the four overlapping square layers

11.2.1 The End-Effector with the Compaction Roller

The end-effector orientation with respect to the towpreg lines is first set to 90 degrees and the four yaw angles (-45˚, 0˚, 45˚, 90˚} are chosen to provide the different layer orientations and to locate (if any) the valid circular coupons. After stacking the valid squares with the different inclinations, the yielded feasible circular coupons are considerably small and located in restricted regions. Figure 11.7 locates the table 'substrate' in workcell and presents the only two valid circular coupons with 6-inches radius; larger coupons are not feasible. Similar unsatisfying results are obtained when the set of yaw angles is varied (e.g., {67.5˚, 22.5˚, -22.5˚, -67.5˚}). After multiple trials, significant improvements are achieved only by changing the end-effector orientation angles whereas the same mediocre results are yielded for the 90 degrees end-effector orientation.

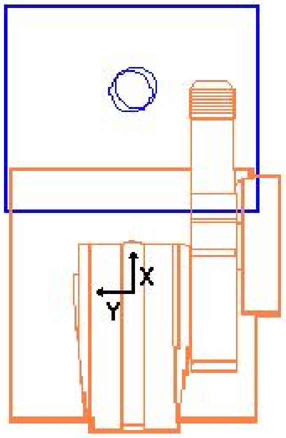


Figure 11.7 The largest feasible circular coupons for a 90 degrees end-effector orientation

Figure 11.8 explains the major limitations in the size and in the number of circular coupons yielded when the end-effector orientation is set to 90 degrees. Even though the shown 10-inch squares are numerous for each inclination angle {-45˚, 0˚, 45˚, 90˚}, there are no four squares from each orientation that can completely overlap to form the desired 10-inch isotropic coupon.

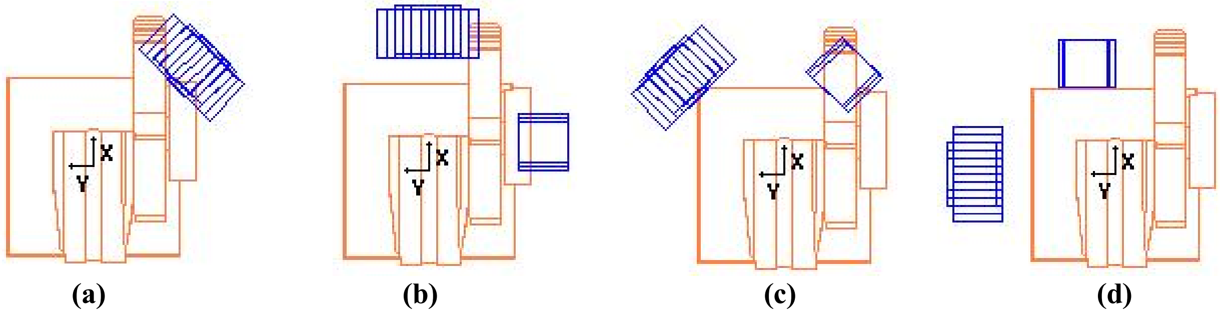
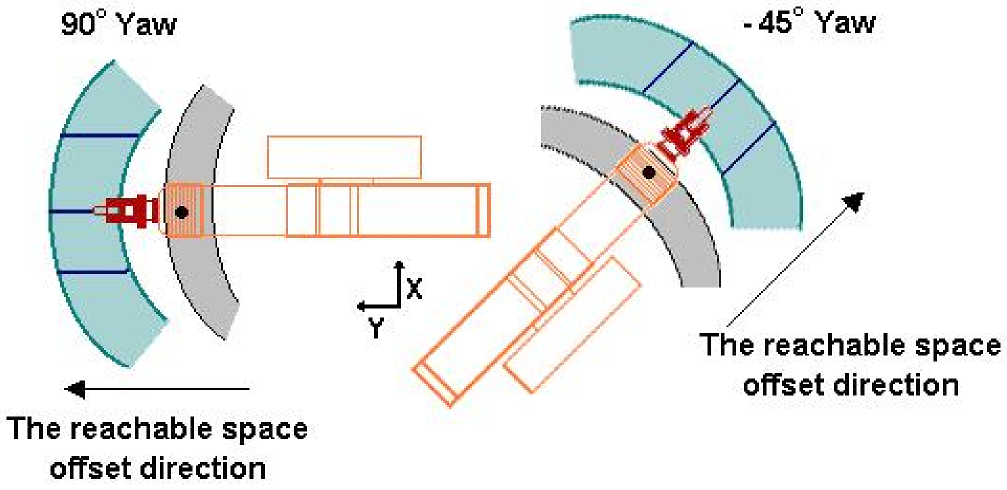


Figure 11.8 The locations of the 10 in. squares with the different inclination

The significant difference between the squares locations is caused by the position of the roller contact point with respect to the wrist center. As shown in Fig.11.9, the wrist location with respect to the wrist induces an offset in the reachable workspace of the contact in the direction of the *towpreg orientation*. As a result, the locations of large squares with significantly different orientations would largely differ as they are shifted toward their orientation direction.



90° Yaw
- 45° Yaw
X
Y
The reachable space offset direction
The reachable space offset direction

Figure 11.9 The reachable workspace offset in the direction of the lines orientations

To solve the positional offset in the roller reachable workspace, the end-effector orientation angle is manipulated to minimize or even eliminate the positional offset of the roller contact point with respect to the wrist center. The latter has a fixed and invariable reachable workspace for any line inclination, and accordingly, in reducing the distance between the roller and the wrist center, the end-effector contact point would have its fixed and unchanged workspace for all line orientations.

According to Fig. 11.10, the offset distance between the roller contact point and the wrist center is given by:

$$offset = (S_1 + d_6)\sin(\theta) + (S_2)\cos(\theta) \tag{11.1}$$

S_1 and S_2 are already defined as the measures that position the contact point with respect to the faceplate center. d_6 is the Denavit-Hartenberg parameter representing the wrist length, and θ is as previously mentioned, the orientation of the end-effector with respect to the towpreg path.

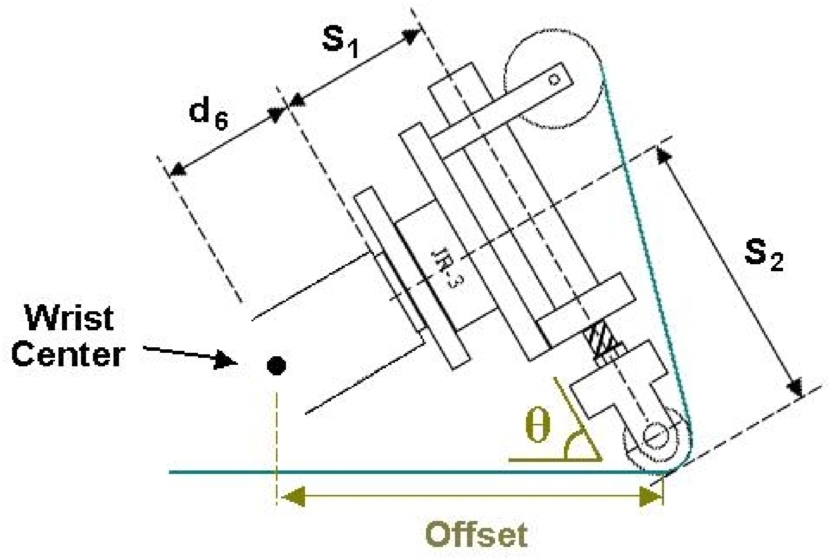


Figure 11.10 The offset distance between the roller contact point and the wrist center

In setting the offset value in equation 11.1 to null, the following expression for the end-effector orientation is derived:

$$\tan(\theta_c) = \frac{\sin(\theta_c)}{\cos(\theta_c)} = \frac{-S_2}{S_1 + d_6} \tag{11.2}$$

According to equation 11.2, the offset value can be zeroed, when the end-effector orientation angle satisfies the above equation. In plugging the numerical values in the equation 11.2, the value of θ_c is found by choosing the solution angle less than 180 degrees. For the known dimensions of the end-effector and the Merlin robot wrist length, the computed orientation angle θ_c is equal to 138 degrees. As shown on Fig. 11.11, the contact and the wrist center are aligned on the normal to the towpreg path, thus eliminating the offset distance along the towpreg line.

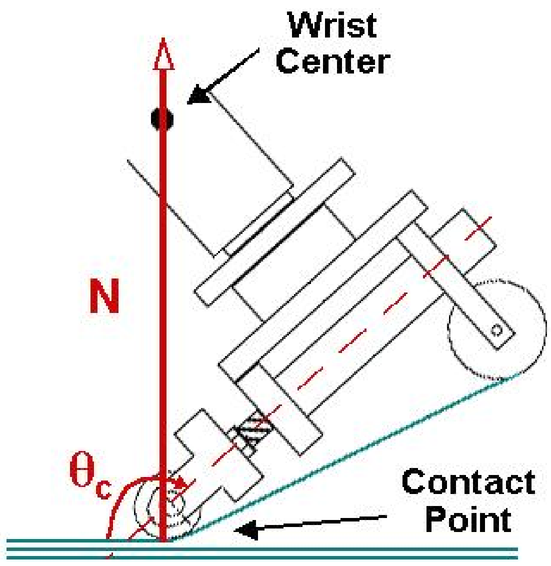


Figure 11.11 The contact point and the wrist center aligned on the normal to the towpreg line

In setting the end-effector orientation angle to θ_c, considerably large circular coupons are feasible in various regions in the plane. As shown in Fig. 11.13, numerous feasible 16-inch squares are checked for the four yaw parameters {-45˚, 0˚, 45˚, 90˚}. The formed 16-inch coupons are located in the horizontal plane in Fig. 11.12. Smaller feasible circular coupons are more frequent and more variously located on the fixed table; Fig. 11.14 positions all the feasible 15-inch valid circular coupons.

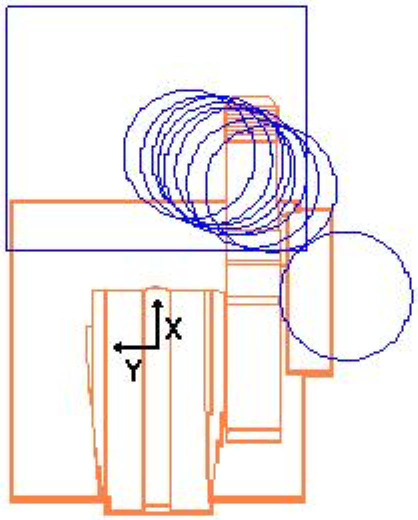


Figure 11.12 The largest (16-inch) isotropic coupons

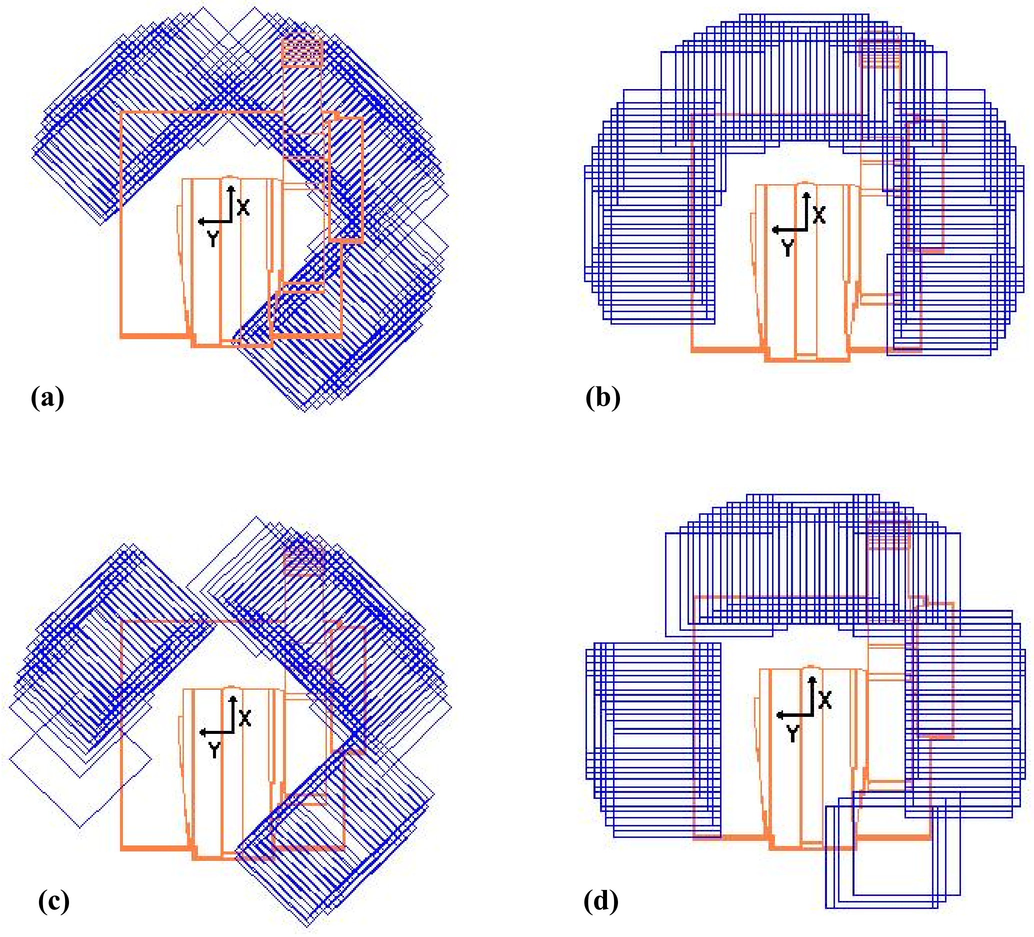


Figure 11.13 The feasible 16-inch squares that would form the valid the coupons layers: (a) –45˚ inclined squares, (b) 0˚, (c) 45˚ and (d) 90˚

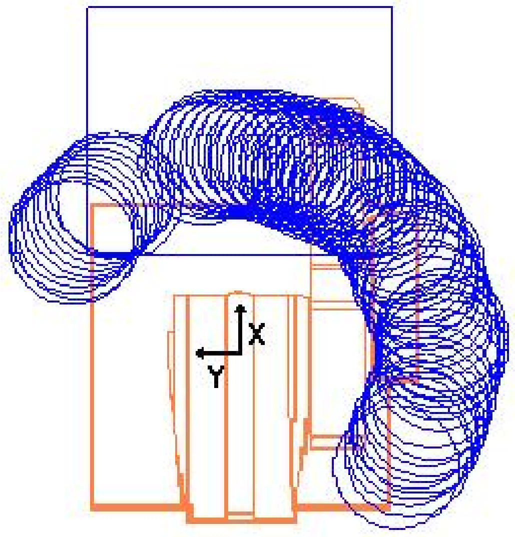


Figure 11.14 The numerous 15-inch isotropic coupons

11.2.2 The End-Effector with the Compaction ball

As shown in Fig. 11.5, the roller and ball compaction heads are built with almost the same dimensional configurations. However, the main kinematic difference between the two end-effectors involves the orientational behavior of the two distinct compression heads with respect to the towpreg paths. More specifically, the compaction *ball* configuration does not allow any variations in the end-effector orientation parameter θ which should be set to 90 degrees and unchanged at all times.

As a result, mediocre results are obtained when the ball compaction head is used to trace and locate the feasible isotropic coupons. Since the two end-effectors have almost the same dimensional measures and the same preliminary orientation angle (90 degrees) with respect to the towpregs, the size and frequency of the generated valid coupons are dramatically reduced (same as in Fig. 11.7). As already discussed, the results are greatly improved when the roller compaction head orientation is set to θ_c. As this orientational adjustment cannot be performed on the ball end-effector requiring a fixed θ equal to 90 degrees, *orientational variations about another axis* would allow the fabrication of coupons with considerable sizes and dimensions.

In setting θ to 90 degrees, the ball compaction head can rotate about the X_6 axis of the Denavit-Hartenberg frame attached on the robot faceplate (shown in Fig. 9.1). The ball contact point still follows the pre-designed towpreg path and provides the proper normal compression force for adequate consolidation. Figure 11.15 illustrates the rotational freedom of the end-effector to be oriented in various directions during the consolidation on a single towpreg trajectory.

This additional rotation β is added to equation 9.2 to compute the orientation of the ball compaction head with respect to base frame. As shown in equation 11.3, *θ is set to 90 degrees* while the ball orientational matrix is completed by including the rotation β.

$$R = T \cdot \begin{bmatrix} \cos(\pi/2) & 0 & \sin(\pi/2) \\ 0 & 1 & 0 \\ -\sin(\pi/2) & 0 & \cos(\pi/2) \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \tag{11.3}$$

In order to trace all four oriented layers on relatively large coupons, several simulation trials are performed to study the influence the angle β on the sizes and locations of the flat circular coupons. The best results are yielded when β is set to 90 degrees. For the two sets of yaw angles {-45˚, 0˚, 45˚, 90˚} and {-67.5˚, -22.5˚, 22.5˚, 67.5˚} degrees, Fig. 11.16 locates all possible traced circular coupons on the horizontal plane. The shown coupons represent the largest feasible isotropic coupons with a 10-inch diameter.

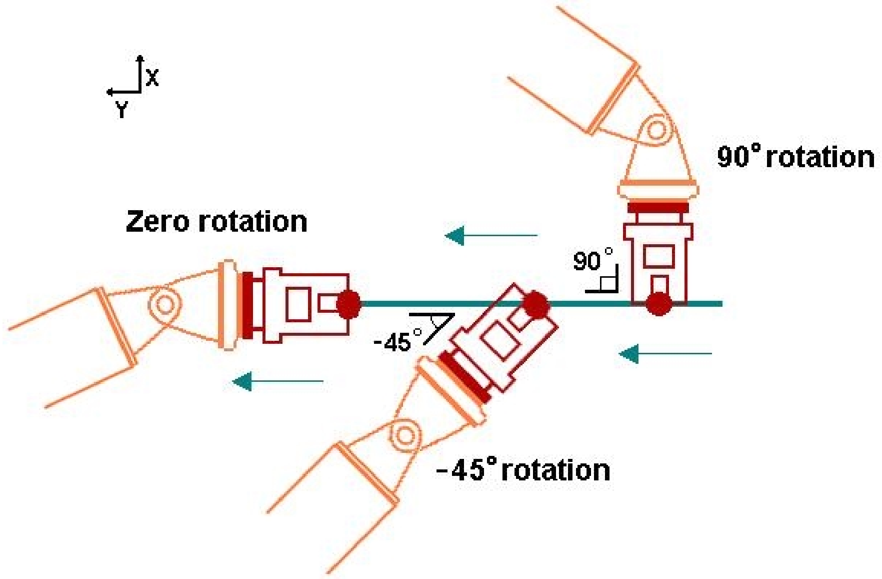
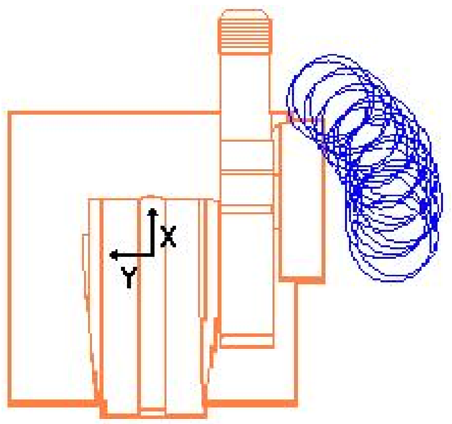
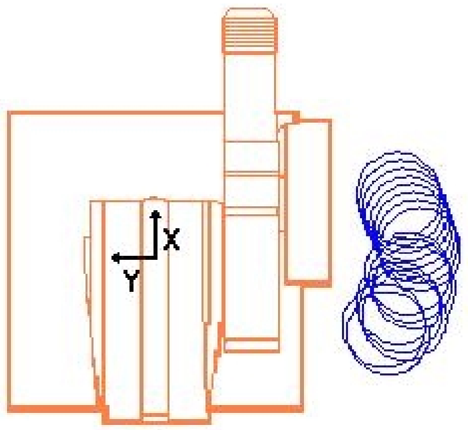


Figure 11.15 Horizontal view of the ball compaction head oriented in various directions in the consolidation process

For all yaw angles sets, mediocre results are yielded when the rotation angle β is set to 0, 180 or –90 degrees. On the other hand, alternating β between 90 and –90 degrees in accordance with the layers orientation would generate more circular coupons in diverse locations. Figure 11.17 positions all feasible coupons if β is set to 90 degrees when simulating the three layers {-22.5˚, 22.5˚, 67.5˚}, and then changed to -90 degrees while tracing the fourth layer {-67.5˚}.





(a) (b)

Figure 11.16 Relatively large coupons for *β* equal to 90 degrees.
The layers orientations are {-45, 0, 45, 90} degrees in (a) and {-67.5, -22.5, 22.5, 67.5} degrees in (b)

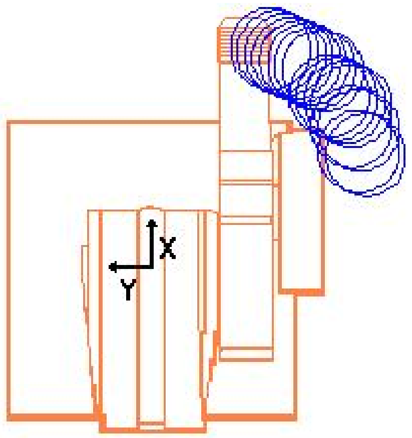


Figure 11.17 Feasible coupons when alternating *β* for the different layer orientations

CHAPTER 12: CONCLUDING REMARKS

12.1 Summary and Discussion

The presented method to map the feasible workspace of a robot is an effective approach that considers the kinematic capabilities of the manipulator along with the requirements of the robotic task. By simulating all required end-effector trajectories, the feasibility of specific tasks is verified prior to its physical execution, thus saving trial time and tool waste. All kinematic restrictions in the robotic workcell are checked for all end-effector paths required to accomplish the task and to meet its multiple requirements. Furthermore, one of the most valuable outcomes of the feasible workspace method is its ability to locate the workpiece in the robot workspace and, additionally, to determine the product (or task) limitations in size and complexity.

Acknowledging that the feasible workspace approach can be applied on any *non-redundant manipulator* performing *any desired task*, the kinematic capabilities of the *6-dof Merlin robot* are considered to manufacture composite bodies with the *6-dof online consolidation fiber placement* technique. After subdividing the robot three-dimensional workspace to multiple planes, all valid trajectories with pre-determined parameters are determined along with all their correspondent feasible squared areas. Consequently, the feasible workspace method has the advantage of finding alternative valid locations for a workpiece when the product manufacture fails in its original position in the workspace. Furthermore, the effective workspace approach would accurately determine the appropriate dimensions of product structures to insure the complete manufacture feasibility.

Not only does the feasible trajectory technique determine the valid locations and dimensions of the structure to fabricate, this workspace method interferes directly in the design of the end-effector configuration. Knowing that the end-effector size greatly affects the reachable workspace, it is shown throughout this work that specific dimensional proportions for the end-effector configuration should be met to allow the manufacturing of products structures with pre-determined pattern layers.

The kinematic fiber placement requirements are defined and related to the towpreg trajectory. The end-effector XYZ position and three-dimensional orientation are restricted to the location and direction of the towpreg path in the space. The multiple task requirements are also discussed to be included in the trajectory verifications. However, the feasible workspace method can easily be generalized by considering other robotic tasks and satisfying their specific requirements.

The Merlin robot restrictions involve the actuators dynamic specification limits: For the given end-effector velocity and compression forces, the joint's angular velocities and torques should be kept under the desired threshold limits. In addition, the manipulability of the robot is considered to avoid singularities. Finally, collision between the substrate and the robot rotating links (or the fixed obstacles in the workspace) is checked to complete the path verification. If impact is detected in the simulation, shorter feasible segments within the original trajectory are determined. Here again, the algorithm to check the robot restrictions can be extended to involve any non-redundant manipulator. Hydraulic or pneumatic actuators would be checked for their particular specifications limits (*e.g.* pressure). Singularity avoidance is verified by computing the Jacobian determinant for the manipulator kinematic configuration, and collision detection is definitely possible by considering the links dimensions or including any other obstacle in the workspace.

Accordingly, the feasible workspace technique can easily be applied to any non-redundant robot for any desired 6-dof task. By completing the above modifications, the same algorithm determines all feasible trajectories and areas in the manipulator workspace.

The feasible boundaries are plotted on two-dimensional planes for simpler visual analysis. For specific set of line parameters, different methods are introduced to present the valid workspace. All feasible segments, longest segments and feasible squared areas can be plotted to illustrate the planar workspace of the robot and to offer different interpretation of the results.

Two end-effectors were built to test the manufacture of isotropic coupons on a horizontal table surface facing the robot. A correlation between the end-effector configuration dimensions and the roller orientation is determined to locate satisfactory *large* isotropic circular coupons. These results necessarily interfere in the design of future end-effectors to allow the fabrication of isotropic products with the desired roller orientations. On the other hand, when a compaction ball is mounted to provide the needed layout compression forces, adequate changes in the introduced additional rotational freedom permit the tracing of relatively large isotropic coupons in diverse locations in the horizontal plane.

In verifying the fabrication of pre-defined product structures, the feasible workspace method yields all possible locations of the substrate where all the robot restrictions are met. Optimizing the workpiece location in the workspace is not considered in the analysis, as the objective of the feasible workspace method is to determine the manufacturing feasibility of structures in all *possible* positions in the robot workspace. However, optimizing the workpiece location can be very easily implemented in this technique, as the robot dexterity (or manipulability) would be considered the primary criterion for the optimization process. By recording the robot manipulability (on every discrete point) for each possible structure position in the space, the optimization process would select the 'optimized' structure location having the largest manipulability values. But again, implementing this simple optimization logic is left for future work since one of the feasible workspace method purposes involves mapping all feasible positions to provide the widest sets of practical choices.

In verifying the feasibility of end-effector paths, researchers set threshold limits for the manipulability values in order to avoid singular configurations and to insure smooth trajectories. However, the manipulability index is an abstract measure that cannot be physically controlled or verified, and accordingly, the chosen manipulability threshold might be too strict or possibly too lenient, causing serious uncertainties in trajectory verification. Alternatively, since the joint velocities dramatically increase in kinematic singularity configurations, this work uses the manipulability measure (or the Jabobian determinant) only to detect critical paths whereas the threshold limits are set on the joints velocity, a well controllable and concrete physical measure.

12.2 Future Work

As any curved surface can be fragmented to a series of straight segments, arcs with small radii of curvature are however, represented by large numbers of short segments, definitely causing very large computational time and effort to verify the feasibility of each segment. Consequently, it is recommended to expand the feasible workspace algorithm to verify the feasibility of circular paths. More accurate and faster path verification would be available to verify pre-defined cylindrical components and possibly to set new rules for the compaction head configuration design.

REFERENCES

[1] Shih, P.J., and Loos, A.C., 1997, “On-line Consolidation of Thermoplastic Composites,” Center for Composite Materials and Structures, Blacksburg.

[2] Bullock, F., Kowalski, S., and Young, R., 1990, “Automated Prepreg Tow Placement for Composite Structures,” 35th International SAMPE Symposium, pp. 734-745.

[3] Hummler, J., Lee, S.K., and Steiner, K.V., 1991, “Recent Advances in Thermoplastic Robotic Filament Winding,” 36th International SAMPE Symposium, pp. 2142-2156.

[4] Tierney, J., Eduljee, R.F., and Gillespie, J. W. Jr., 1998, “ Control of Warpage and Residual Stresses during the Automated Tow Placement Process,” 43rd International SAMPE Symposium, pp. 652-664.

[5] Hauber, D.E., Hardtmann D.J., and Budeck K.B., 1990, “ Recent Advances in Thermoplastic Composite Fabrication using ROWS,” 35th International SAMPE Symposium, pp. 767-772.

[6] Enders, M.L., and Hopkins, P.C., 1991, “Developments in the Fiber Placement Process,” 36th International SAMPE Symposium, pp. 778-790.

[7] Sainani, M.R., and Sturges, R.H., 1999, “Quantifying Robotic Assembly Capability: A Review and a Prospectus,” Proc. of FAIM, Tilburg, Netherlands.

[8] Shimano, B.E., and Roth, B., 1977, “Dimensional Synthesis of Manipulators,” Elsevier, pp. 18-27.

[9] Kumar, A., and Waldron, K.J., 1980, “The Dexterous Workspace,” Design Engineering Technical Conference, ASME paper no.80-DET-108.

[10] Kumar, A., and Waldron, K.J., 1981, “The Workspace of a Mechanical Manipulator,” ASME Journal of Mechanical Design, 103, 3, pp. 615-627.

[11] Roth, B., 1975, “Performance Evaluation of Manipulators from a Kinematic Viewpoint,” NBS Special Publication, NBS, pp. 39-61.

[12] Tsai, Y.C., and Soni, A.H., 1984, “The Effect of Link Parameters on the Working Space of General 3R Robot Arms,” Mechanism and Machine Theory, 19, pp. 9-16.

[13] Gupta, K.C., and Roth, B., 1982, “Design Considerations for Manipulator Workspaces,” ASME Journal of Mechanical Design, 104, 4, pp. 704-712.

[14] Yoshikawa, T., 1984, “Analysis and Control of Robot Manipulators with Redundancy,” Robotics Research: The First International Symposium, M. Brady and R. Paul, Eds. Cambridge, MA:MIT Press, pp. 735-747.

[15] Yoshikawa, T., 1985, "Manipulability of Robotic Mechanisms," The International Journal of Robotics Research, 4, 2.

[16] Chiu, S. L., 1988, "Task Compatibility of Manipulator Postures", The international Journal of Robotics Research, 7, 5, pp. 13-21.

[17] Tsai, Y.C., and Chiou, Y. H., 1990, "Manipulability of Manipulators," Mechanism and Machine Theory, 25, 5, pp. 575-585.

[18] Kim, J., and Khosla, P. K., 1991, "Dexterity Measures for Design and Control of Manipulators," IEEE/RSJ Intl. Workshop on Intelligent Robots and Systems IROS'91, Osaka, Japan, pp. 758-763.

[19] Roberts, R. G., 1995, "Quantifying the Local Fault Tolerance of a Kinematically Redundant Manipulator," Proc. American Control Conference, Seattle, WA, pp. 1889-1893.

[20] Doty, K. L., and Schwartz, E.M., 1995, "Robot Manipulability," IEEE Trans. Robotics and Automation, 11, 3, pp. 462-468.

[21] Klein, C.A., and Blaho, B.E., 1987, "Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators," the International Journal of Robotics Research, 6, 2, pp. 72-82.

[22] Angeles, J. and Lopez-Cajun, C., 1988, "The Dexterity Index for Serial-Type Robotic Manipulators," Proc. of the ASME Mechanisms Conference, Kissimmer, Florida, pp. 79-84.

[23] Gosselin, C., 1992, "The Optimum Design of Robotic Manipulators using Dexterity Indices," Journal of Robotics and Autonomous Systems, 9, pp. 213-226.

[24] Whitney, D. E., 1982, "Quasi-Static Assembly of Compliantly Supported Rigid Parts," Journal of Dynamic Systems, Measurement and Control, 104, pp. 65-77.

[25] Sturges, R. H., 1990, "A Quantification of Machine Dexterity applied to an Assembly Task", The International Journal of Robotics Research, 9, 3, pp. 49-62.

[26] Canbolat, H. and Erkmen, A. M., 1994, "Optimal Preshaping using Vorticity Based Manipulability and Satbility Criteria," Proc. of the 1994 IEEE conf on Robotics and Automation, pp. 1943-1949

[27] Snyman, Plessis, Duffy, 2000, "An Optimization Approach to the Determination of the Boundaries of Manipulator Workspaces," Journal of Mechanical Design, 122, 4, pp. 447-456.

[28] Carretero, J.A., Nahon, M., and Podhorodeski, R.P., 1998, "Workspace Analysis of a 3-dof Parallel Mechanism," Proc.of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria, Canada.

[29] Abdel-Malek, K., and Yeh, H.J., 2000, “Crossable Surfaces of Robotic Manipulators with Joints Limits,” Journal of Mechanical Design, 122, 1, pp. 52-60.

[30] Abdel-Malek, K., and Yeh, H.J., 1997, “Path Trajectory Verification for Robot Manipulators in a Manufacturing Environment,” ImechE Journal of Engineering Manufacture, 211 B, pp. 547-556.

[31] Chaney, K.D., and Davidson, J.K., 1998, “A Synthesis Method for Placing Workpieces in RPR Planar Robotic Workcells,” Journal of Mechanical Design, 120, 2, pp. 262-268.

[32] Soman, N.A., and Davidson, J.K., 1995, “A Two-Dimensional Formulation for Path Placement in the Workcells of Planar 3-R Robots,” Journal of Mechanical Design, 117, 3, pp. 479-484.

[33] Merlet, J.P., 1994, “Trajectory Verification in the Workspace for Parallel Manipulators,” Int. J. Robot. Res., 13, 4, pp. 326-333.

[34] Merlet, J.P., 2001, “A Generic Trajectory Verifier for the Motion Planning of Parallel Robots,” Journal of Mechanical Design, 123, 4, pp. 510-515.

[35] Merlin system Operator’s Guide, Version 3.0 / June 1985, American Robot Corporation.

[36] Sciavicco, L., and Siciliano, B., 1996, Modeling and Control of Robot Manipulators. McGraw-Hill.

[37] Spong, M.W., and Vidyasagar, 1989, Robot Dynamics and control. John Wiley & Sons, New York.

APPENDIX A: VERTICAL AND HORIZONTAL VIEWS OF THE MERLIN ROBOT REACHABLE WORKSPACE

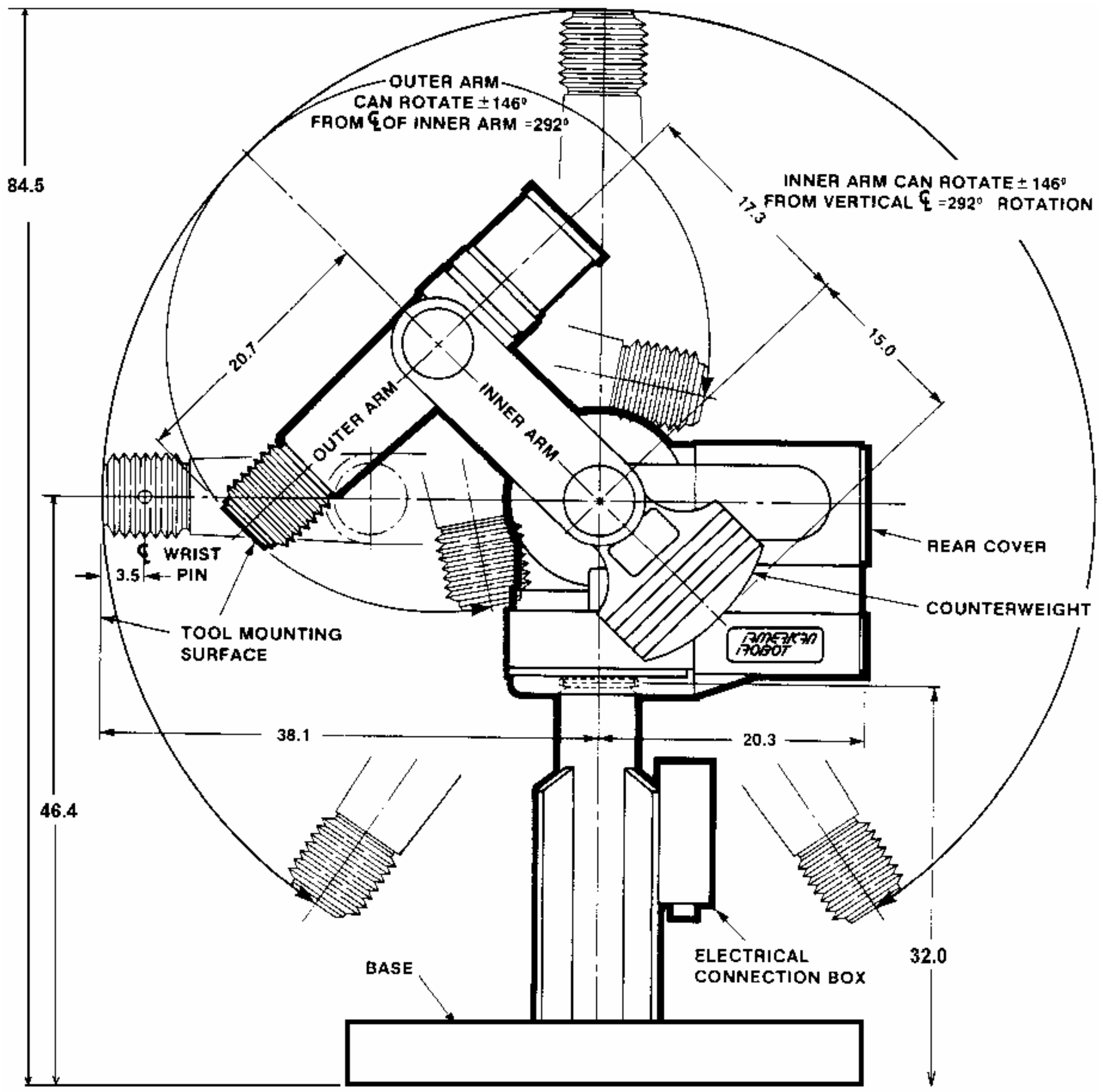

OUTER ARM CAN ROTATE ±146° FROM ℄ OF INNER ARM =292°
INNER ARM CAN ROTATE ± 146° FROM VERTICAL ℄ =292° ROTATION
84.5
17.3
15.0
20.7
OUTER ARM
INNER ARM
REAR COVER
COUNTERWEIGHT
℄ WRIST PIN
3.5
TOOL MOUNTING SURFACE
AMERICAN ROBOT
38.1
20.3
46.4
ELECTRICAL CONNECTION BOX
BASE
32.0

Figure A.1 The vertical view of the Merlin robot reachable workspace [35]

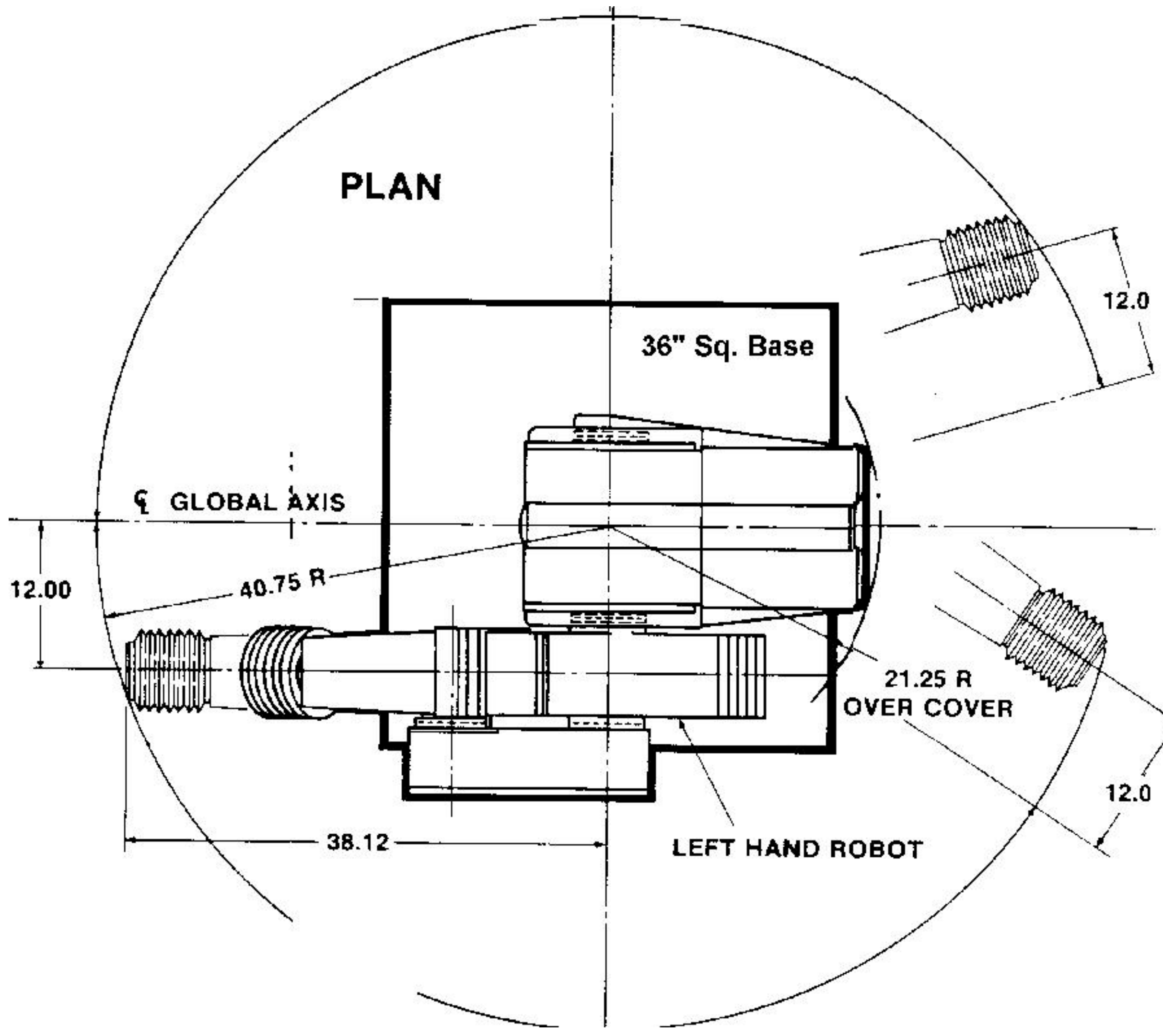


Figure A.2 The horizontal view of the Merlin robot reachable workspace [35]

APPENDIX B: THE ROTATIONAL SPAN OF SIX THE MERLIN ROBOT JOINTS

Joint	Rotation limited by the hardware stops	Upper hardware limit	Lower hardware limit
1 *Waist*	290 degrees	175 degrees	-115 degrees
2 *Shoulder*	292 degrees	236 degrees	-56 degrees
3 *Elbow*	292 degrees	146 degrees	-146 degrees
4 *Wrist Rotate*	Continuous	-	-
5 *Wrist Flex*	+/- 90 degrees	90 degrees	-90 degrees
6 *Hand Rotate*	Continuous	-	-

Table B.1 The rotational extent of the six Merlin robot joints

APPENDIX C: THE DENAVIT-HARTENBERG PARAMETERS

Link i	a_i	α_i	d_i	θ_i
1	0	+90	d_1	θ_1
2	a_2	0	d_2	θ_2
3	0	+90	0	θ_3+90
4	0	-90	d_4	θ_4
5	0	+90	0	θ_5
6	0	0	d_6	θ_6

Table C.1 The Denavit-Hartenberg parameters for the Merlin robot

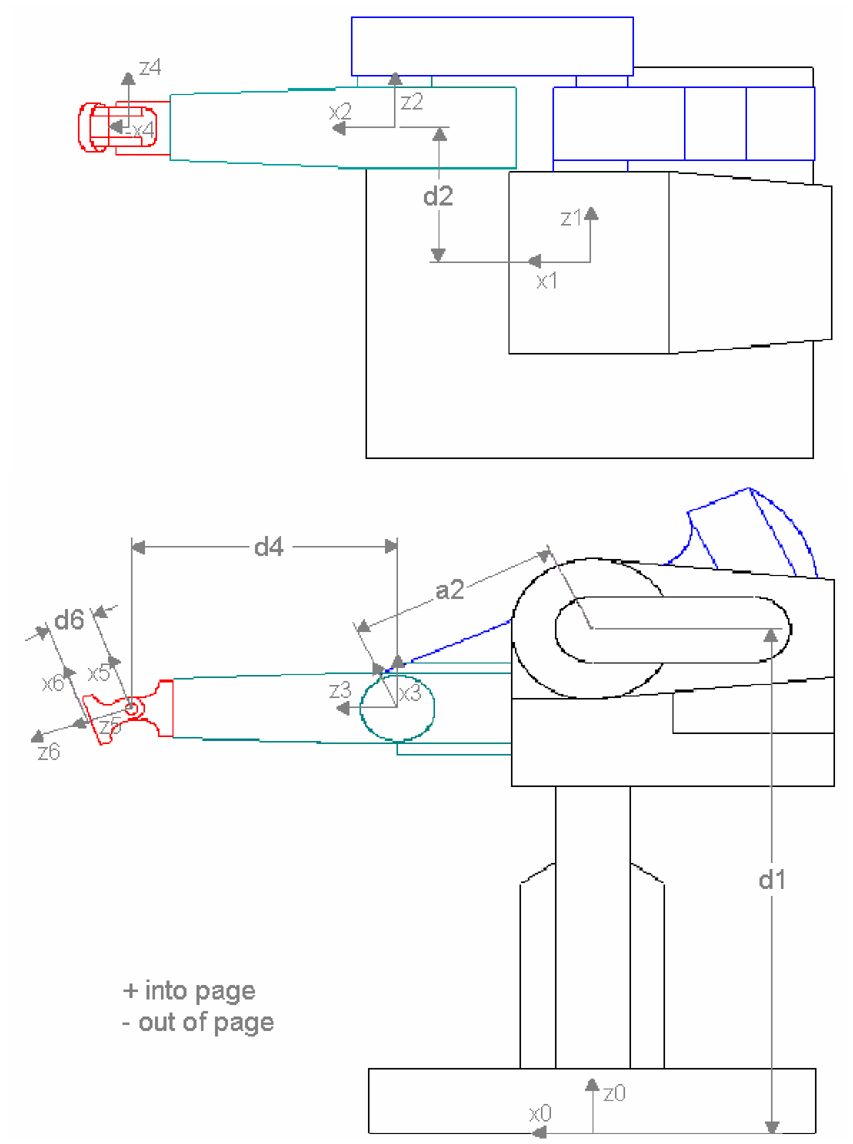


Figure C.1 The Denavit-Hartenberg parameters shown on the Merlin robot

APPENDIX D: THE INVERSE KINEMATICS EQUATIONS FOR THE MERLIN ROBOT

d_1, d_2, a_2, d_4 and d_6 are the Merlin Robot Denavit-Hartenberg parameters defined in Table C.1.

For the given position $\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$ and orientation $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$:

- The XYZ position of the wrist center is: $\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} d_x - d_6 r_{13} \\ d_y - d_6 r_{23} \\ d_z - d_6 r_{33} \end{bmatrix}$ (D.1)

Solutions exist if $\left(p_x^2 + p_y^2 - d_2^2\right) > 0$ and $\left|\dfrac{s^2 + (p_z - d_1)^2 - a_2^2 - d_4^2}{2a_2 d_4}\right| < 1$ (D.2)

Where $s = \sqrt{p_x^2 + p_y^2 - d_1^2}$ (D.3)

- The two solution angles for the waist joint are:

$$\theta_{1a} = \tan^{-1}\left(p_y, p_x\right) + \tan^{-1}\left(d_2, s\right) \quad \text{(D.4)}$$

$$\theta_{1b} = \tan^{-1}\left(p_y, p_x\right) - \tan^{-1}\left(d_2, s\right) + \pi \quad \text{(D.5)}$$

- Two solution sets for the shoulder-Elbow angles exist for each waist solution:

$$\theta_3 = \tan^{-1}\left(\pm\sqrt{1 - D_3^2}, D_3\right) \quad \text{(D.6)}$$

$$\theta_2 = \tan^{-1}\left(\frac{(a_2 + d_4 D_3)(p_z - d_1) \pm d_4 s\sqrt{1 - D_3^2}}{s^2 + (p_z - d_1)^2}, \frac{(a_2 + d_4 D_3)s \pm d_4 (p_z - d_1)\sqrt{1 - D_3^2}}{s^2 + (p_z - d_1)^2}\right) \quad \text{(D.7)}$$

where $D_3 = \dfrac{s^2 + (p_z - d_1)^2 - a_2^2 - d_4^2}{2a_2 d_4}$ (D.8)

- Two solutions for the orientational joint angles exist for each of the four positional set of angles:

$$\theta_{5a} = \tan^{-1}\left(\sqrt{1-b_{33}^2}, b_{33}\right) \text{ and } \theta_{5a} = -\tan^{-1}\left(\sqrt{1-b_{33}^2}, b_{33}\right) \quad \text{(D.9)}$$

$$\theta_{4a} = \tan^{-1}\left(b_{23}, b_{13}\right) \text{ and } \theta_{4a} = \tan^{-1}\left(b_{23}, b_{13}\right)+\pi \quad \text{(D.10)}$$

$$\theta_{6a} = \tan^{-1}\left(b_{32}, -b_{31}\right) \text{ and } \theta_{6a} = \tan^{-1}\left(b_{32}, -b_{31}\right)+\pi \quad \text{(D.11)}$$

$$\text{where } R_3^6 = \left(R_0^3\right)^T R_0^6 = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad \text{(D.12)}$$

APPENDIX E: PICTURES OF THE ROLLER AND THE BALL COMPACTION HEAD

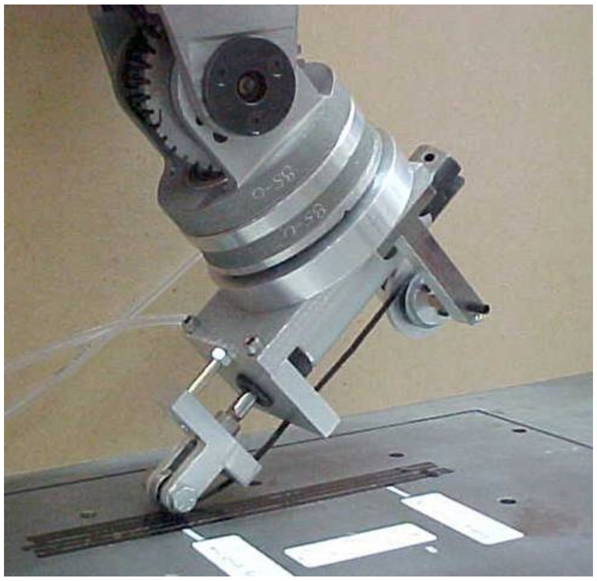


Figure E.1 The roller compaction head mounted on the wrist of the Merlin robot

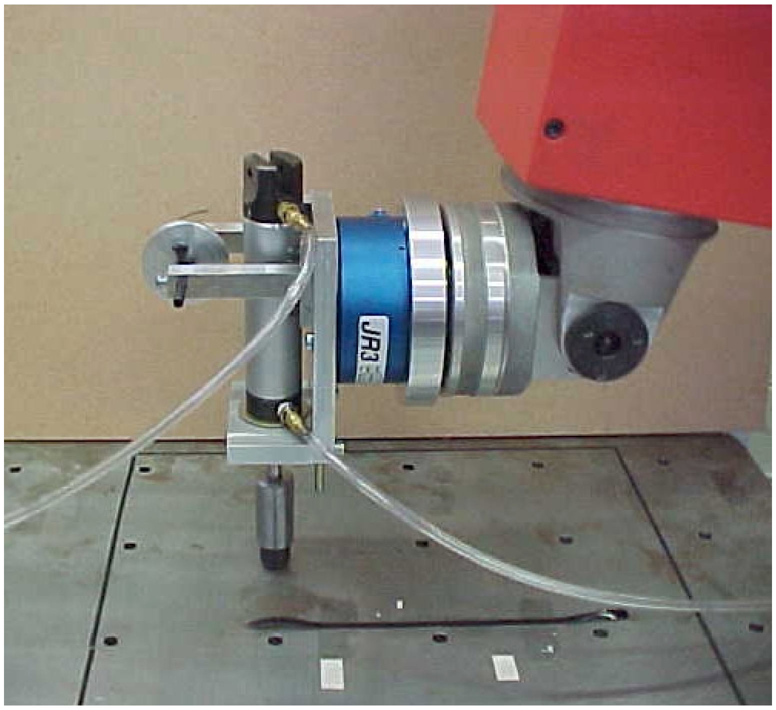


Figure E.2 The ball compaction head mounted on the wrist of the Merlin robot

APPENDIX F: THE MATLAB CODE

F.1 THE CODE TO SIMULATE THE LAYOUT ON A FEASIBLE SEGMENT

```
% This file simulates the layout of a line in a 3-D figure and in XY, XZ, YZ plane figures
% Graphs for the joint velocities, torques, determinant of the Jacobian and the angles are drawn
% The user chooses the solution number from the 8 possible solutions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To clear all stored data
clear all
% To close all Matlab figures
close all

% To control the speed of the simulation
p = 0.1;

% The dimensions of the Merlin
L1=46.4;          % length of link 1
D1=11.9;           % offset of joint 2 wrt the axis of joint 1
L2=17.375;        % length of link 2
L3=17.25;         % length of link 3
L4=3.5;           % length of link 5

% The dimensions of the end-effector
spec1 = 3.89;     % the distance from the center of the face plate to contact point on the roller along the Z axis
spec2 = 6.67;     % the distance from the center of the face plate to contact point on the roller along the X axis

% Dynamic properties of the Merlin and the end-effector
Sp = 0.25;        % in/sec
Fc = 5;           % the compression force in lbf
Wee = 15;         % the estimated weight of the e.e. in lbf

% The parameters of the line to generate
pitch = 0 *pi/180;          % the line pitch
yaw  =  0 *pi/180;          % the line yaw
roll  =   0 *pi/180;        % the line roll
orient = 138 *pi/180;       % the orientation of the end-effector wrt the line
rot    =  0 *pi/180;        % the rotation of the e.e. ball around the normal of the tow path
if spec2 == 0
   orient = orient + 90*pi/180;
end

Xoffset = 0;
Yoffset = 0;
Zoffset = 0;

% The line offsets from the origin of the base frame
% The Zoffset is compared to the shoulder joint center
if pitch == pi/2  |  pitch == -pi/2
   Xoffset = 25;
   Yoffset = 20;
else
   if  yaw == pi/2  | yaw == -pi/2
```

```
        Xoffset =  25;
        Zoffset = -18;
    else
        Yoffset =  -22 ;
        Zoffset = -18;
    end
end

% Calling the function ee_dynamics
[Mp,S,V,F] = ee_dynamics(yaw,pitch,roll,orient,rot,Sp,Fc,Wee,Xoffset,Yoffset,Zoffset);

% The loop that moves the points along the line
inc = 1;              % distance between the points on the line (in inches)
d=0;                  % counter set to zero

for distance = 100:-inc:-100
    d = d+1;          % incrementing the counter
    dist(d) = distance;

    % M4 is changing with the distance
    M4 = [ 1  0  0  distance; 0 1 0 0; 0 0 1 0;   0 0 0 1;];
    % Computing the matrix to give the position of the point
    M = Mp *M4;

    % The position of the point: the e.e. position
    X(d) = M(1,4);
    Y(d) = M(2,4);
    Z(d) = M(3,4);

    % Calling the function inverse
    [solution,out(d)] = inverse(X(d),Y(d),Z(d),S,spec1,spec2);

    if out(d) == 0
        [det_Jac, Vel, Tor, p0, p1, p2, p3, p4, p5, p6, pinter, p7] = forward(solution, V, F, spec1, spec2);

        % Storing the angles
        for i = 1:1:8
            for an = 1:1:6
                % Storing the angles
                t(d,i,an) = solution(i,an);
                % Storing the Joint Velocities
                q(d,i,an) = Vel(an,i);
                % Storing the Joint Torques
                to(d,i,an) = Tor(an,i);
            end

            % The determinant of the Jacobian
            deter(d,i) = det_Jac(i);
            deter(d+1,i) = det_Jac(i);

            % The positions of the origins of the D-H frames
            X1(d,i)= p1(1,i);
            Y1(d,i)= p1(2,i);
            Z1(d,i)= p1(3,i);

            X2(d,i)= p2(1,i);
            Y2(d,i)= p2(2,i);
            Z2(d,i)= p2(3,i);

            X4(d,i)= p4(1,i);
            Y4(d,i)= p4(2,i);
            Z4(d,i)= p4(3,i);

            X6(d,i)= p6(1,i);
            Y6(d,i)= p6(2,i);
            Z6(d,i)= p6(3,i);

            Xinter(d,i)= pinter(1,i);
            Yinter(d,i)= pinter(2,i);
            Zinter(d,i)= pinter(3,i);

            X7(d,i)= p7(1,i);
            Y7(d,i)= p7(2,i);
            Z7(d,i)= p7(3,i);
        end
    end
end

% Entering the solution number k
k = input('Enter the solution number (1-8): ');
nb = 0;
% Checking limit switches, joint velocity and joint torques
for j = 1:1:d
    if out(j) == 0
        [outlimit(j,k),detcheck(j,k),velcheck(j,k),torcheck(j,k)] = checklimits(t(j,k,:),t(j-
1,k,:),q(j,k,:),to(j,k,:),deter(j,k),out(j-1),out(j-2),out(j+1),deter(j-1,k),deter(j-
2,k),deter(j+1,k),X7(j,k),Y7(j,k),Z7(j,k),X7(j-1,k),Y7(j-1,k),Z7(j-1,k),S,spec1,spec2,V,F,k,inc);
        nb = 1;
    end
end

% Find the boundary points on the POSSIBLE lines using the above checks
% Finding the boundaries of all possible segments along the line
if nb ~=0
    [b1,b2,nb] = linebounds(out,outlimit,detcheck,torcheck,velcheck,d,k);
end
nblines = nb; % number of lines

posiseg = 0; % number of possible sub-segments
if nb ~= 0
    % Storing the positions of corners of every link for collision detection
    for i = 1:1:nblines  % for all possible lines possible from the eight solutions
        for u = b1(i):1:b2(i)   % for all possible segments
            An = [t(u,k,1);t(u,k,2);t(u,k,3);t(u,k,4);t(u,k,5);t(u,k,6);]; % storing the set of angles
            Point  = [ X7(50); Y7(50); Z7(50);]; % any point works here.
            % collision is not detected
            % the links corners are stored
            [Det,C1(u,:,:)] =  intersect(Point, An, spec1,spec2); % calling the function 'intersect'
        end
    end
    [posiseg, posiseginitial,seg1,seg2] = collision(b1,b2,nb,t,k,X7,Y7,Z7,spec1,spec2,posiseg);

    % Finding the longest line
    Lopt = 0;
    for y= 1:1:posiseg      % for all the sub-segments
       if ((seg2(y)-seg1(y))*inc) > Lopt
          Lopt = (seg2(y)-seg1(y))*inc;
          yopt = y;
       end
    end
    Lopt
    %stropt = [X7(seg1(yopt),k)  Y7(seg1(yopt),k)   Z7(seg1(yopt),k)]
    %endopt = [X7(seg2(yopt),k)  Y7(seg2(yopt),k)   Z7(seg2(yopt),k)]

    figure(1)
    % 3 D view of the Robot
    title('3D View of Manipulator');
    for i = 1:1:nblines
       for j = b1(i):1:b2(i)
          clf;
          % the Robot as a skeleton
          Xm = [0;   X1(j,k);    X2(j,k);    X4(j,k);    X6(j,k);   Xinter(j,k);  X7(j,k); ];
          Ym = [0;   Y1(j,k);    Y2(j,k);    Y4(j,k);    Y6(j,k);   Yinter(j,k);  Y7(j,k); ];
          Zm = [0;   Z1(j,k);    Z2(j,k);    Z4(j,k);    Z6(j,k);   Zinter(j,k);  Z7(j,k); ];

          % Link 1
          Xc1 = [C1(j,1,1);   C1(j,1,3);    C1(j,1,4);    C1(j,1,2);    C1(j,1,1); C1(j,1,5);   C1(j,1,6);    C1(j,1,2);
C1(j,1,6);   C1(j,1,10);   C1(j,1,9);    C1(j,1,5);    C1(j,1,9);   C1(j,1,11);    C1(j,1,12);    C1(j,1,10); C1(j,1,12);
C1(j,1,8);   C1(j,1,7);    C1(j,1,11);    C1(j,1,7); C1(j,1,3);   C1(j,1,4);    C1(j,1,8); ];
          Yc1 = [C1(j,2,1);   C1(j,2,3);    C1(j,2,4);    C1(j,2,2);    C1(j,2,1); C1(j,2,5);   C1(j,2,6);    C1(j,2,2);
C1(j,2,6);   C1(j,2,10);   C1(j,2,9);    C1(j,2,5);    C1(j,2,9);   C1(j,2,11);    C1(j,2,12);    C1(j,2,10); C1(j,2,12);
C1(j,2,8);   C1(j,2,7);    C1(j,2,11);    C1(j,2,7); C1(j,2,3);   C1(j,2,4);    C1(j,2,8); ];
          Zc1 = [C1(j,3,1);   C1(j,3,3);    C1(j,3,4);    C1(j,3,2);    C1(j,3,1); C1(j,3,5);   C1(j,3,6);    C1(j,3,2);
C1(j,3,6);   C1(j,3,10);   C1(j,3,9);    C1(j,3,5);    C1(j,3,9);   C1(j,3,11);    C1(j,3,12);    C1(j,3,10); C1(j,3,12);
C1(j,3,8);   C1(j,3,7);    C1(j,3,11);    C1(j,3,7); C1(j,3,3);   C1(j,3,4);    C1(j,3,8); ];

          % Link 2
          Xc2 = [C1(j,1,16);   C1(j,1,15);    C1(j,1,13);    C1(j,1,14);    C1(j,1,16); C1(j,1,20);   C1(j,1,18);
C1(j,1,14);    C1(j,1,13);    C1(j,1,17);   C1(j,1,19);    C1(j,1,15);    C1(j,1,19);   C1(j,1,20);    C1(j,1,18);
C1(j,1,17);];
          Yc2 = [C1(j,2,16);   C1(j,2,15);    C1(j,2,13);    C1(j,2,14);    C1(j,2,16); C1(j,2,20);   C1(j,2,18);
C1(j,2,14);    C1(j,2,13);    C1(j,2,17);   C1(j,2,19);    C1(j,2,15);    C1(j,2,19);   C1(j,2,20);    C1(j,2,18);
C1(j,2,17);];
          Zc2 = [C1(j,3,16);   C1(j,3,15);    C1(j,3,13);    C1(j,3,14);    C1(j,3,16); C1(j,3,20);   C1(j,3,18);
C1(j,3,14);    C1(j,3,13);    C1(j,3,17);   C1(j,3,19);    C1(j,3,15);    C1(j,3,19);   C1(j,3,20);    C1(j,3,18);
C1(j,3,17);];

          % Link 3
          Xc3 = [C1(j,1,21);   C1(j,1,22);    C1(j,1,24);    C1(j,1,23);    C1(j,1,21); C1(j,1,25);   C1(j,1,27);
C1(j,1,23);    C1(j,1,24);    C1(j,1,28);   C1(j,1,26);    C1(j,1,22);    C1(j,1,26);   C1(j,1,25);    C1(j,1,27);
C1(j,1,28);];
          Yc3 = [C1(j,2,21);   C1(j,2,22);    C1(j,2,24);    C1(j,2,23);    C1(j,2,21); C1(j,2,25);   C1(j,2,27);
C1(j,2,23);    C1(j,2,24);    C1(j,2,28);   C1(j,2,26);    C1(j,2,22);    C1(j,2,26);   C1(j,2,25);    C1(j,2,27);
C1(j,2,28);];
```

```
      Zc3 = [C1(j,3,21);   C1(j,3,22);   C1(j,3,24);   C1(j,3,23);   C1(j,3,21); C1(j,3,25);   C1(j,3,27);
C1(j,3,23);   C1(j,3,24);   C1(j,3,28);   C1(j,3,26);   C1(j,3,22);   C1(j,3,26);   C1(j,3,25);   C1(j,3,27);
C1(j,3,28);];

      % The body
      Xc4 = [C1(j,1,29);   C1(j,1,30);   C1(j,1,31);   C1(j,1,32);   C1(j,1,29);  C1(j,1,33);   C1(j,1,36);
C1(j,1,32);   C1(j,1,31);   C1(j,1,35);   C1(j,1,36);   C1(j,1,33);   C1(j,1,34);   C1(j,1,35);   C1(j,1,31);
C1(j,1,30);   C1(j,1,34);];
      Yc4 = [C1(j,2,29);   C1(j,2,30);   C1(j,2,31);   C1(j,2,32);   C1(j,2,29);  C1(j,2,33);   C1(j,2,36);
C1(j,2,32);   C1(j,2,31);   C1(j,2,35);   C1(j,2,36);   C1(j,2,33);   C1(j,2,34);   C1(j,2,35);   C1(j,2,31);
C1(j,2,30);   C1(j,2,34);];
      Zc4 = [C1(j,3,29);   C1(j,3,30);   C1(j,3,31);   C1(j,3,32);   C1(j,3,29);  C1(j,3,33);   C1(j,3,36);
C1(j,3,32);   C1(j,3,31);   C1(j,3,35);   C1(j,3,36);   C1(j,3,33);   C1(j,3,34);   C1(j,3,35);   C1(j,3,31);
C1(j,3,30);   C1(j,3,34);];

      plot3(Xc1,Yc1,Zc1, Xc2, Yc2, Zc2 , Xc3, Yc3, Zc3, Xc4, Yc4, Zc4, Xm , Ym, Zm);
      hold on

      % The Line Generated
      Xline = [X7(b1(i),k); X7(b2(i),k)];
      Yline = [Y7(b1(i),k); Y7(b2(i),k)];
      Zline = [Z7(b1(i),k); Z7(b2(i),k)];

      plot3(Xline,Yline,Zline,'r');

      axis('equal')
      axis([-50 50 -50 50  0 100]);
      view(-147,26)
      hold off
      pause(p)
    end
  end
  hold off

  figure(2)
  % Top view of the robot and the line
  for i = 1:1:nblines
    for j = b1(i):1:b2(i)
      clf;
      axis([-40 50 -40 50]);
      axis off
      hold on;
      % The robot as a skeleton
      plot([0,X1(j,k)],[0,Y1(j,k)]);
      plot([X1(j,k),X2(j,k)],[Y1(j,k),Y2(j,k)]);
      plot([X2(j,k),X4(j,k)],[Y2(j,k),Y4(j,k)]);
      plot([X4(j,k),X6(j,k)],[Y4(j,k),Y6(j,k)]);
      plot([X6(j,k),Xinter(j,k)],[Y6(j,k),Yinter(j,k)]);
      plot([Xinter(j,k),X7(j,k)],[Yinter(j,k),Y7(j,k)]);
      % The line generated
      plot([X7(b1(i),k), X7(b2(i),k)], [Y7(b1(i),k), Y7(b2(i),k)],'g');

      % Link 1
      plot([C1(j,1,1), C1(j,1,2)],[C1(j,2,1), C1(j,2,2)]);
      plot([C1(j,1,1), C1(j,1,5)],[C1(j,2,1), C1(j,2,5)]);
      plot([C1(j,1,2), C1(j,1,6)],[C1(j,2,2), C1(j,2,6)]);
```

```
        plot([C1(j,1,5), C1(j,1,6)],[C1(j,2,5), C1(j,2,6)]);
        plot([C1(j,1,5), C1(j,1,9)],[C1(j,2,5), C1(j,2,9)]);
        plot([C1(j,1,6), C1(j,1,10)],[C1(j,2,6), C1(j,2,10)]);
        plot([C1(j,1,9), C1(j,1,10)],[C1(j,2,9), C1(j,2,10)]);

        % Link 2
        plot([C1(j,1,18), C1(j,1,17)],[C1(j,2,18), C1(j,2,17)]);
        plot([C1(j,1,17), C1(j,1,13)],[C1(j,2,17), C1(j,2,13)]);
        plot([C1(j,1,13), C1(j,1,14)],[C1(j,2,13), C1(j,2,14)]);
        plot([C1(j,1,14), C1(j,1,18)],[C1(j,2,14), C1(j,2,18)]);

        % Link 3
        plot([C1(j,1,25), C1(j,1,26)],[C1(j,2,25), C1(j,2,26)]);
        plot([C1(j,1,26), C1(j,1,22)],[C1(j,2,26), C1(j,2,22)]);
        plot([C1(j,1,22), C1(j,1,21)],[C1(j,2,22), C1(j,2,21)]);
        plot([C1(j,1,21), C1(j,1,25)],[C1(j,2,21), C1(j,2,25)]);

        % The body
        plot([C1(j,1,32), C1(j,1,29)],[C1(j,2,32), C1(j,2,29)]);
        plot([C1(j,1,29), C1(j,1,33)],[C1(j,2,29), C1(j,2,33)]);
        plot([C1(j,1,33), C1(j,1,36)],[C1(j,2,33), C1(j,2,36)]);
        plot([C1(j,1,36), C1(j,1,32)],[C1(j,2,36), C1(j,2,32)]);

        pause(p);
    end
end
hold off


figure(3)
% Side view of the robot and the line
for i = 1:1:nblines
    for j = b1(i):1:b2(i)
        clf;
        axis([-40 60 0 100]);
        axis off
        hold on;
        plot([0,X1(j,k)],[0,Z1(j,k)]);
        plot([X1(j,k),X2(j,k)],[Z1(j,k),Z2(j,k)]);
        plot([X2(j,k),X4(j,k)],[Z2(j,k),Z4(j,k)]);
        plot([X4(j,k),X6(j,k)],[Z4(j,k),Z6(j,k)]);
        plot([X6(j,k),Xinter(j,k)],[Z6(j,k),Zinter(j,k)]);
        plot([Xinter(j,k),X7(j,k)],[Zinter(j,k),Z7(j,k)]);
        % The line generated
        plot([X7(b1(i),k), X7(b2(i),k)], [Z7(b1(i),k), Z7(b2(i),k)],'g');

        % Link 1
        plot([C1(j,1,2), C1(j,1,6)],[C1(j,3,2), C1(j,3,6)]);
        plot([C1(j,1,6), C1(j,1,10)],[C1(j,3,6), C1(j,3,10)]);
        plot([C1(j,1,10), C1(j,1,12)],[C1(j,3,10), C1(j,3,12)]);
        plot([C1(j,1,12), C1(j,1,8)],[C1(j,3,12), C1(j,3,8)]);
        plot([C1(j,1,8), C1(j,1,4)],[C1(j,3,8), C1(j,3,4)]);
        plot([C1(j,1,4), C1(j,1,2)],[C1(j,3,4), C1(j,3,2)]);

        % Link 2
        plot([C1(j,1,18), C1(j,1,14)],[C1(j,3,18), C1(j,3,14)]);
        plot([C1(j,1,14), C1(j,1,16)],[C1(j,3,14), C1(j,3,16)]);
        plot([C1(j,1,16), C1(j,1,20)],[C1(j,3,16), C1(j,3,20)]);
        plot([C1(j,1,20), C1(j,1,18)],[C1(j,3,20), C1(j,3,18)]);

        % Link 3
        plot([C1(j,1,22), C1(j,1,26)],[C1(j,3,22), C1(j,3,26)]);
        plot([C1(j,1,26), C1(j,1,28)],[C1(j,3,26), C1(j,3,28)]);
        plot([C1(j,1,28), C1(j,1,24)],[C1(j,3,28), C1(j,3,24)]);
        plot([C1(j,1,24), C1(j,1,22)],[C1(j,3,24), C1(j,3,22)]);

        % The body
        plot([C1(j,1,32), C1(j,1,36)],[C1(j,3,32), C1(j,3,36)]);
        plot([C1(j,1,36), C1(j,1,35)],[C1(j,3,36), C1(j,3,35)]);
        plot([C1(j,1,35), C1(j,1,31)],[C1(j,3,35), C1(j,3,31)]);
        plot([C1(j,1,31), C1(j,1,32)],[C1(j,3,31), C1(j,3,32)]);

        pause(p);
    end
end
hold off

figure(4)
% Front view of the robot and the line
for i = 1:1:nblines
    for j = b1(i):1:b2(i)
        clf;
        axis([-40 60 0 100]);
        axis off
        hold on;
        plot([0,Y1(j,k)],[0,Z1(j,k)]);
        plot([Y1(j,k),Y2(j,k)],[Z1(j,k),Z2(j,k)]);
        plot([Y2(j,k),Y4(j,k)],[Z2(j,k),Z4(j,k)]);
        plot([Y4(j,k),Y6(j,k)],[Z4(j,k),Z6(j,k)]);
        plot([Y6(j,k),Yinter(j,k)],[Z6(j,k),Zinter(j,k)]);
        plot([Yinter(j,k),Y7(j,k)],[Zinter(j,k),Z7(j,k)]);
        % The line generated
        plot([Y7(b1(i),k), Y7(b2(i),k)], [Z7(b1(i),k), Z7(b2(i),k)],'g');

        % Link 1
        plot([C1(j,2,6), C1(j,2,5)],[C1(j,3,6), C1(j,3,5)]);
        plot([C1(j,2,5), C1(j,2,1)],[C1(j,3,5), C1(j,3,1)]);
        plot([C1(j,2,1), C1(j,2,3)],[C1(j,3,1), C1(j,3,3)]);
        plot([C1(j,2,3), C1(j,2,7)],[C1(j,3,3), C1(j,3,7)]);
        plot([C1(j,2,7), C1(j,2,8)],[C1(j,3,7), C1(j,3,8)]);
        plot([C1(j,2,8), C1(j,2,4)],[C1(j,3,8), C1(j,3,4)]);
        plot([C1(j,2,4), C1(j,2,2)],[C1(j,3,4), C1(j,3,2)]);
        plot([C1(j,2,2), C1(j,2,6)],[C1(j,3,2), C1(j,3,6)]);
        plot([C1(j,2,2), C1(j,2,1)],[C1(j,3,2), C1(j,3,1)]);
        plot([C1(j,2,4), C1(j,2,3)],[C1(j,3,4), C1(j,3,3)]);

        % Link 2
        plot([C1(j,2,14), C1(j,2,13)],[C1(j,3,14), C1(j,3,13)]);
        plot([C1(j,2,13), C1(j,2,15)],[C1(j,3,13), C1(j,3,15)]);
        plot([C1(j,2,15), C1(j,2,16)],[C1(j,3,15), C1(j,3,16)]);
        plot([C1(j,2,16), C1(j,2,14)],[C1(j,3,16), C1(j,3,14)]);
```

```
            % Link 3
            plot([C1(j,2,22), C1(j,2,21)],[C1(j,3,22), C1(j,3,21)]);
            plot([C1(j,2,21), C1(j,2,23)],[C1(j,3,21), C1(j,3,23)]);
            plot([C1(j,2,23), C1(j,2,24)],[C1(j,3,23), C1(j,3,24)]);
            plot([C1(j,2,24), C1(j,2,22)],[C1(j,3,24), C1(j,3,22)]);

            % The body
            plot([C1(j,1,32), C1(j,1,31)],[C1(j,3,32), C1(j,3,31)]);
            plot([C1(j,1,31), C1(j,1,30)],[C1(j,3,31), C1(j,3,30)]);
            plot([C1(j,1,30), C1(j,1,29)],[C1(j,3,30), C1(j,3,29)]);
            plot([C1(j,1,29), C1(j,1,32)],[C1(j,3,29), C1(j,3,32)]);

            pause(p);
        end
    end
    hold off

    % Plots for the Lines, Determinant, the Joint Velocity and Torque
    for i = 1:1:nblines
        figure(5+2*(i-1))
        %size(['     XYZ E.E. POSITION ',' ',' ',' ',' ','          ' ])
        %size(['St pt: X = ', num2str( X7(b1(i),k)),'  Y = ', num2str(Y7(b1(i),k)), '  Z = ', num2str(Z7(b1(i),k))])
        %size([ 'Ed pt: X = ', num2str(X7(b2(i),k)),'  Y = ', num2str(Y7(b2(i),k)), '  Z = ', num2str(Z7(b2(i),k))])

        SUBPLOT(2,2,1), plot3( [X7(b1(i),k); X7(b2(i),k)],[Y7(b1(i),k); Y7(b2(i),k)],[Z7(b1(i),k); Z7(b2(i),k)],'g'),
title('XYZ E.E. POSITION') % ; 'St pt: X = ', num2str( X7(b1(i),k)),'  Y = ', num2str(Y7(b1(i),k)), '  Z = ',
num2str(Z7(b1(i),k)); 'Ed pt: X = ', num2str(X7(b2(i),k)),'  Y = ', num2str(Y7(b2(i),k)), '  Z = ',
num2str(Z7(b2(i),k));] )
        SUBPLOT(2,2,3), plot ( dist(b1(i):b2(i)), q(b1(i):b2(i),k,1)*180/pi,'-', dist(b1(i):b2(i)),
q(b1(i):b2(i),k,2)*180/pi,'--',dist(b1(i):b2(i)), q(b1(i):b2(i),k,3)*180/pi,'o', dist(b1(i):b2(i)),
q(b1(i):b2(i),k,4)*180/pi,':',dist(b1(i):b2(i)), q(b1(i):b2(i),k,5)*180/pi,'^',dist(b1(i):b2(i)),
q(b1(i):b2(i),k,6)*180/pi,'+'), title('JOINTS ANGULAR VELOCITIES(deg/sec) ')
        SUBPLOT(2,2,2), plot ( dist(b1(i):b2(i)), deter(b1(i):b2(i),k)), title('DETERMINANT OF THE JACOBIAN')
        SUBPLOT(2,2,4), plot ( dist(b1(i):b2(i)), to(b1(i):b2(i),k,1),'-', dist(b1(i):b2(i)), to(b1(i):b2(i),k,2),'--
',dist(b1(i):b2(i)), to(b1(i):b2(i),k,3),'o', dist(b1(i):b2(i)), to(b1(i):b2(i),k,4),':',dist(b1(i):b2(i)),
to(b1(i):b2(i),k,5),'^',dist(b1(i):b2(i)), to(b1(i):b2(i),k,6),'+'), title('JOINTS TORQUES(in.lb) ')

        figure(6+2*(i-1))
        SUBPLOT(2,3,1), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,1) *180/pi), title('Joint Angle 1')
        SUBPLOT(2,3,2), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,2) *180/pi), title('Joint Angle 2')
        SUBPLOT(2,3,3), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,3) *180/pi), title('Joint Angle 3')
        SUBPLOT(2,3,4), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,4) *180/pi), title('Joint Angle 4')
        SUBPLOT(2,3,5), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,5) *180/pi), title('Joint Angle 5')
        SUBPLOT(2,3,6), plot (dist(b1(i):b2(i)), t(b1(i):b2(i),k,6) *180/pi), title('Joint Angle 6')
    end
else
    display(' No Feasable Lines with the chosen solution number and lines parameters');
end
```

E.2 THE CODE TO PLOT THE WORKSPACE AND DRAW THE SQUARES

```
% This file plots the longest possible lines on a fixed plane
% It plots all feasible lines on the fixed plane (overlapping is possible and frequent)
% A histogram shows the distribution of the longest lines length
% Possible squares with a chosen dimension are fit and plotted in the plane
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To clear all stored data
clear all
% To close all Matlab figures
close all

% The dimensions of the Merlin
L1=46.4; % length of link 1
D1=11.9; % offset of joint 2 wrt the axis of joint 1
L2=17.375; % length of link 2
L3=17.25; % length of link 3
L4=3.5; % length of link 5

% The dimensions of the end-effector
spec1 = 3.89;  % the distance from the center of the face plate to contact point on the roller
% along the axis of the face plate
spec2 = 6.67;  % the distance from the center of the face plate to contact point on the roller
% along the perpendicular to the axis of the face plate

% Dynamic Properties of the Merlin and the e.e.
Sp = 0.25; % in/sec
Fc = 5;   % the compression force in lbf
Wee = 15;  % the estimated weight of the e.e. in lbf
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Entering the desired plane
plane = input('Enter the number representing the desired plane:  1 for XY,  2 for XZ,  3 for YZ: ');

% The parameters of the lines to generate
pitch = 0 *pi/180; % the lines pitch
yaw = 0 *pi/180; % the lines yaw
roll  = 0 *pi/180; % the lines roll
orient = 138 *pi/180; % The orientation of the face plate wrt the lines
rot = 0*pi/180; % the rotation of the e.e. ball around the normal of the tow path
if spec2 == 0
   orient = orient + 90*pi/180;
end

Xoffset = 0;
Yoffset = 0;
Zoffset = 0;

% The line offsets from the origin of the base frame
% the Zoffset is compared to the shoulder joint center
if pitch == pi/2  |  pitch == -pi/2
   Xoffset = 25;
   Yoffset = 30;
else
   if  yaw == pi/2  | yaw == -pi/2
```

```
        Xoffset = 25;
        Zoffset = -18;
    else
        Yoffset = 20;
        Zoffset = 0;
    end
end

% The loop starts here
% Index used in the loop
o =0;
% Parameter for the first line
off1 = -100;
% Parameter for the last line
off2 = 100;
% Parameter increment between the lines - spacing between the lines in inches -
offinc = 1;

% Varying the Lines
for off = off1:offinc:off2
    off
    % To choose the varying offset parameter
    % depending on the lines angle parameters
    switch plane
    case 1
        if yaw == pi/2 | yaw == -pi/2
            Xoffset = off;
        else
            Yoffset = off/cos(yaw);
        end
    case 2
        if pitch == pi/2 | pitch == -pi/2
            Xoffset = off;
        else
            Zoffset = off/cos(pitch);
        end
    case 3
        if pitch == pi/2 | pitch == -pi/2
            Yoffset = off;
        else
            Zoffset = off/cos(pitch);
        end
    end

    % Incrementing the index
    o = o+1;

    % Setting all the loop results to zero
    posiseg(o) = 0; % number of possible segment at each offset
    Lopt(o)=0;
    Xopt_st(o)= 0;
    Yopt_st(o)= 0;
    Zopt_st(o)= 0;
    Xopt_ed(o)= 0;
    Yopt_ed(o)= 0;
    Zopt_ed(o)= 0;
```

```
[Mp,S,V,F] = ee_dynamics(yaw,pitch,roll,orient,rot,Sp,Fc,Wee,Xoffset,Yoffset,Zoffset);

% Inverse Dynamics for the line
% The loop that moves the points along the line
inc = 1;  % distance between the points on the line (in inches)
d=0;      % counter set to zero

for distance = 100:-inc:-100
   d = d+1;             % incrementing the counter
   dist(d) = distance;

   % M4 is changing with the distance
   M4 = [ 1  0  0  distance; 0 1 0 0; 0 0 1 0;   0 0 0 1;];
   % Computing the matrix to give the position of the point
   M = Mp*M4;

   % The position of the point: the e.e. position
   X(d) = M(1,4);
   Y(d) = M(2,4);
   Z(d) = M(3,4);

   [solution,out(d)] = inverse(X(d),Y(d),Z(d),S, spec1, spec2);

   if out(d) == 0
      [det_Jac, Vel, Tor, p0, p1, p2, p3, p4, p5, p6, pinter, p7] = forward(solution, V, F, spec1, spec2);

      % Storing the angles
      for i = 1:1:8
         for an = 1:1:6
            % Storing the angles
            t(d,i,an) = solution(i,an);
            % Storing the Joint Velocities
            q(d,i,an) = Vel(an,i);
            % Storing the Joint Torques
            to(d,i,an) = Tor(an,i);
         end

         % The determinant of the Jacobian
         deter(d,i) = det_Jac(i);
         deter(d+1,i) = det_Jac(i);

         % The Positions of the origines of the D-H frames
         X1(d,i)= p1(1,i);
         Y1(d,i)= p1(2,i);
         Z1(d,i)= p1(3,i);

         X2(d,i)= p2(1,i);
         Y2(d,i)= p2(2,i);
         Z2(d,i)= p2(3,i);

         X4(d,i)= p4(1,i);
         Y4(d,i)= p4(2,i);
         Z4(d,i)= p4(3,i);

         X6(d,i)= p6(1,i);
```

```
            Y6(d,i)= p6(2,i);
            Z6(d,i)= p6(3,i);

            X7(d,i)= p7(1,i);
            Y7(d,i)= p7(2,i);
            Z7(d,i)= p7(3,i);
        end
    end
end

% Considering the eight solutions
for k = 1:1:8
    nb(o,k) = 0;
    % Checking limit switches, joint velocity and joint torques
    for j = 1:1:d
        if out(j) == 0
          [outlimit(j,k),detcheck(j,k),velcheck(j,k),torcheck(j,k)]=checklimits(t(j,k,:),t(j1,k,:),q(j,k,:),to(j,k,:),deter(j,k
          ),out(j-1),out(j-2),out(j+1),deter(j-1,k),deter(j-k),deter(j+1,k),X7(j,k),Y7(j,k),Z7(j,k),X7(j-1,k),Y7(j-
          1,k),Z7(j-1,k),S,spec1,spec2,V,F,k,inc);
           nb(o,k) = 1;
        end
    end

    % Find the boundary points on the POSSIBLE lines using the above checks
    % Finding the boundaries of all possible segments along the line
    if nb(o,k) ~= 0
        [b1,b2,nb(o,k)] = linebounds(out,outlimit,detcheck,torcheck,velcheck,d,k);
    end
    if nb(o,k) ~= 0
          [posiseg(o),posi,seg1((posi+1):posiseg(o)),seg2((posi+1):posiseg(o))]
          =collision(b1,b2,nb(o,k),t,k,X7,Y7,Z7,spec1,spec2,posiseg(o));
    end
end

% Finding the optimal line
Lopt(o) = 0;
for y= 1:1:posiseg(o)       % for all the sub-segments
    % Storing ALL the lines for fitting a square
    % Storing the boundaries of those lines
    S1(o,y,1) = X7(seg1(y));
    S1(o,y,2) = Y7(seg1(y));
    S1(o,y,3) = Z7(seg1(y));
    S2(o,y,1) = X7(seg2(y));
    S2(o,y,2) = Y7(seg2(y));
    S2(o,y,3) = Z7(seg2(y));

    % Comparing the lines length
    if ((seg2(y)-seg1(y))*inc) > Lopt(o)
        Lopt(o) = (seg2(y)-seg1(y))*inc; % The max length
        % Storing the boundaries of the longest lines at every offset
        Xopt_st(o)= X7(seg1(y));
        Yopt_st(o)= Y7(seg1(y));
        Zopt_st(o)= Z7(seg1(y));
        Xopt_ed(o)= X7(seg2(y));
        Yopt_ed(o)= Y7(seg2(y));
        Zopt_ed(o)= Z7(seg2(y));
        end
    end
end

% Ignoring the lines of zero length
% at the begining and at the end
for p =1:1:o
    if Lopt(p) ~= 0
        goodo1 = p;     % the index of the first line
        break
    end
end

for p = o:-1:1
    if Lopt(p) ~= 0
        goodo2 = p;     % the index of the last line
        break
    end
end


% The enveloppe of the longest lines
switch plane
case 1
    % The enveloppe of the longest lines
    figure(1)
    plot(Xopt_st(goodo1:goodo2),  Yopt_st(goodo1:goodo2),  Xopt_ed(goodo1:goodo2),  Yopt_ed(goodo1:goodo2)),
title(' The envelope of the LONGEST lines')
    axis([-40 60 -50 50])

    figure(2)
    % ALL possible lines (maybe overlapped)
    for o = goodo1:1:goodo2                    % for all offsets
        for y = 1:1:posiseg(o)                 % for all the sub-segments
            plot([S1(o,y,1), S2(o,y,1)],[S1(o,y,2), S2(o,y,2)]), title(' ALL FEASABLE lines')
        end
        hold on
    end

case 2
    % The enveloppe of the longest lines
    figure(1)
    plot(Xopt_st(goodo1:goodo2),          Zopt_st(goodo1:goodo2)          -          L1,          Xopt_ed(goodo1:goodo2),
Zopt_ed(goodo1:goodo2) -L1), title(' The envelope of the LONGEST lines')
    axis([-40 60 -50 50])

    figure(2)
    % ALL possible lines (maybe overlapped)
    for o = goodo1:1:goodo2                    % for all offsets
        for y = 1:1:posiseg(o)                 % for all the sub-segments
            plot([S1(o,y,1), S2(o,y,1)],[S1(o,y,3)- L1, S2(o,y,3)- L1]), title(' ALL FEASABLE lines')
        end
        hold on
    end

case 3
```

```
   % The envelope of the longest lines
   figure(1)
   plot(Yopt_st(goodo1:goodo2),          Zopt_st(goodo1:goodo2)          -          L1,          Yopt_ed(goodo1:goodo2),
Zopt_ed(goodo1:goodo2) -L1), title(' The envelope of the LONGEST lines')
   axis([-40 60 -50 50])

   figure(2)
   % ALL possible lines (maybe overlapped)
   for o = goodo1:1:goodo2                    % for all offsets
      for y = 1:1:posiseg(o)                  % for all the sub-segments
         plot([S1(o,y,2), S2(o,y,2)],[S1(o,y,3)- L1, S2(o,y,3)- L1]), title(' ALL FEASABLE lines')
      end
      hold on
   end
end
axis([-40 60 -50 50])


figure(3)
% The histogram of the line length
hist(Lopt(goodo1:goodo2), 30)

% fit a square
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The square side
a = 14;

figure(4)
hold on
% Calling the function square
switch plane
case 1
   squarecenterloc    =    squareXY(a,    yaw,    offinc,    inc,    S1(goodo1:goodo2,:,:),    S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
   % Plotting the possible squares in the longest lines enveloppe
   for i = 1:1:size(squarecenterloc,1)
      % Plotting each side at a time
      % Computing the 4 corners of every square
      plot([squarecenterloc(i,1)+(a/2)*sin(yaw)-(a/2)*cos(yaw),
squarecenterloc(i,1)+(a/2)*sin(yaw)+(a/2)*cos(yaw)],[(squarecenterloc(i,2)-(a/2)*cos(yaw)-(a/2)*sin(yaw)),
(squarecenterloc(i,2)-(a/2)*cos(yaw)+(a/2)*sin(yaw))]);
      plot([squarecenterloc(i,1)+(a/2)*sin(yaw)+(a/2)*cos(yaw),                    squarecenterloc(i,1)+(a/2)*cos(yaw)-
(a/2)*sin(yaw)],[(squarecenterloc(i,2)-(a/2)*cos(yaw)+(a/2)*sin(yaw)),
(squarecenterloc(i,2)+(a/2)*sin(yaw)+(a/2)*cos(yaw))]);
      plot([squarecenterloc(i,1)+(a/2)*cos(yaw)-(a/2)*sin(yaw),                    squarecenterloc(i,1)-(a/2)*sin(yaw)-
(a/2)*cos(yaw)],[(squarecenterloc(i,2)+(a/2)*sin(yaw)+(a/2)*cos(yaw)),        (squarecenterloc(i,2)+(a/2)*cos(yaw)-
(a/2)*sin(yaw))]);
      plot([squarecenterloc(i,1)-(a/2)*sin(yaw)-(a/2)*cos(yaw),                    squarecenterloc(i,1)+(a/2)*sin(yaw)-
(a/2)*cos(yaw)],[(squarecenterloc(i,2)+(a/2)*cos(yaw)-(a/2)*sin(yaw)),        (squarecenterloc(i,2)-(a/2)*cos(yaw)-
(a/2)*sin(yaw))]);
   end

case 2
   squarecenterloc    =    squareXZ(a,    pitch,    offinc,    inc,    S1(goodo1:goodo2,:,:),    S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
```

```
    % Plotting the possible squares in the longest lines envelope
    for i = 1:1:size(squarecenterloc,1)
        % Plotting each side at a time
        % Computing the 4 corners of every square
        plot([squarecenterloc(i,1)+(a/2)*sin(pitch)-(a/2)*cos(pitch),
squarecenterloc(i,1)+(a/2)*sin(pitch)+(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)-(a/2)*sin(pitch)),
(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch))]);
        plot([squarecenterloc(i,1)+(a/2)*sin(pitch)+(a/2)*cos(pitch), squarecenterloc(i,1)+(a/2)*cos(pitch)-
(a/2)*sin(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*sin(pitch)+(a/2)*cos(pitch))]);
        plot([squarecenterloc(i,1)+(a/2)*cos(pitch)-(a/2)*sin(pitch), squarecenterloc(i,1)-(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*sin(pitch)+(a/2)*cos(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
        plot([squarecenterloc(i,1)-(a/2)*sin(pitch)-(a/2)*cos(pitch), squarecenterloc(i,1)+(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*cos(pitch)-(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1 -
(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
    end

case 3
    squarecenterloc = squareYZ(a, pitch, yaw, offinc, inc, S1(goodo1:goodo2,:,:), S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
    % Plotting the possible squares in the longest lines envelope
    for i = 1:1:size(squarecenterloc,1)
        % Plotting each side at a time
        % Computing the 4 corners of every square
        plot([squarecenterloc(i,2)+(a/2)*sin(pitch)-(a/2)*cos(pitch),
squarecenterloc(i,2)+(a/2)*sin(pitch)+(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)-(a/2)*sin(pitch)),
(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch))]);
        plot([squarecenterloc(i,2)+(a/2)*sin(pitch)+(a/2)*cos(pitch), squarecenterloc(i,2)+(a/2)*cos(pitch)-
(a/2)*sin(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*sin(pitch)+(a/2)*cos(pitch))]);
        plot([squarecenterloc(i,2)+(a/2)*cos(pitch)-(a/2)*sin(pitch), squarecenterloc(i,2)-(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*sin(pitch)+(a/2)*cos(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
        plot([squarecenterloc(i,2)-(a/2)*sin(pitch)-(a/2)*cos(pitch), squarecenterloc(i,2)+(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*cos(pitch)-(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1 -
(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
    end
end
axis([-40 60 -50 50])
```

E.3 THE FUNCTION TO CALCULATE THE VELOCITY AND FORCE VECTORS AND THE END-EFFECTOR ORIENTATION

```
function [M,S,V,F] = ee_dynamics(yaw,pitch,roll,orient,rot,Sp,Fc,Wee,Xoffset,Yoffset,Zoffset)
% EE_DYNAMICS returns the velocity V and force F vectors of the e.e.
%               the orientation S of the e.e.
%               the matrix M that stores the position of the e.e.

L1=46.4; % length of link 1
% Matrices to give the orientation of the end-effector when the tow line parameters are given
S1 = [  cos(yaw)  -sin(yaw)       0;
        sin(yaw)   cos(yaw)       0;
        0          0              1;];

S2 = [  cos(pitch)   0     -sin(pitch);
          0          1      0;
        sin(pitch)   0      cos(pitch);];

S3 = [  1          0              0;
        0   cos(roll)     -sin(roll);
        0   sin(roll)      cos(roll);];

S4 = [  cos(orient)     0   sin(orient);
          0             1      0;
        -sin(orient)    0   cos(orient);];

S5 = [  1       0              0;
        0    cos(rot)    -sin(rot);
        0    sin(rot)     cos(rot);];

% The product yields the orientation of the end-effector wrt the base frame
S = S1*S2*S3*S4*S5;

% The velocity vector the end-effector only dependent on the line pitch and yaw
V = [Sp*cos(pitch)*cos(yaw); Sp*cos(pitch)*sin(yaw); Sp*sin(pitch);  0;  0;  0;];

% Matrice to find the orientation of the compression force
Sf = S1*S2*S3;
Fcx = Fc * Sf(1,3); % Component of the compression force on the X-axis
Fcy = Fc * Sf(2,3); % Component of the compression force on the Y-axis
Fcz = Fc * Sf(3,3); % Component of the compression force on the Z-axis

% The weight of the e.e.
Weex = 0;     % Component of the e.e. weight on the X-axis
Weey = 0;     % Component of the e.e. weight on the Y-axis
Weez = - Wee; % Component of the e.e. weight on the Z-axis

% The overall force F at the e.e.
% No torques are applied on the e.e. because the contact point is a roller
F = [ Fcx+ Weex;  Fcy+ Weey; Fcz + Weez; 0 ; 0; 0;];

% Matrices to give the position of the points on the LINES, they will be used later in the loop
if pitch == pi/2  |  pitch == -pi/2
   M1 = [ 1 0 0 Xoffset;  0 1 0 Yoffset; 0 0 1 0;   0 0 0 1];
   M2 = [ cos(yaw)  -sin(yaw)  0 0;  sin(yaw)  cos(yaw)  0 0;  0 0 1 0;  0 0 0 1];
   M3 = [ cos(pitch) 0  -sin(pitch) 0;  0  1  0  0; sin(pitch) 0 cos(pitch)  0;  0 0 0 1];
else
   if  yaw == pi/2  | yaw == -pi/2
      M1 = [1 0 0 Xoffset;  0 1 0 0; 0 0 1 L1+Zoffset;   0 0 0 1];
      M2 = [ cos(yaw)  -sin(yaw)  0 0;  sin(yaw)  cos(yaw)  0 0;  0 0 1 0;  0 0 0 1];
      M3 = [ cos(pitch) 0  -sin(pitch) 0;  0  1  0  0; sin(pitch) 0 cos(pitch)  0;  0 0 0 1];
   else
      M1 = [ 1 0 0 0;   0 1 0 Yoffset;   0 0 1 L1+Zoffset;    0 0 0 1];
      M2 = [ cos(yaw)  -sin(yaw)  0 0;  sin(yaw)  cos(yaw)  0 0;  0 0 1 0;  0 0 0 1];
      M3 = [ cos(pitch) 0  -sin(pitch) 0;  0  1  0  0; sin(pitch) 0 cos(pitch)  0;  0 0 0 1];
   end
end
M =M1*M2*M3;
```

E.4 THE FORWARD KINEMATICS FUNCTION

```
function [det_Jac, Vel, Tor, po0, po1, po2, po3, po4, po5, po6, pointer, po7] = forward(angles, eV, eF, sp1, sp2)
%FORWARD returns:   the determinant of the Jacobian
%                   the joint velocities
%                   the joint torques
%                   the position of the origins of the frames
%The inputs are : the 8 joint angles sets
%                   the velocity of the e.e.
%                   the forces on the e.e
%                   the dimensional specifications of the e.e.

% The dimensions of the Merlin
L1=46.4; % length of link 1
D1=11.9; % offset of joint 2 wrt the axis of joint 1
L2=17.375; % length of link 2
L3=17.25; % length of link 3
L4=3.5; % length of link 5

% i represents a solution,
for i = 1:1:8
   % Storing the angles solution set
          t1(i) = angles(i,1);
          t2(i) = angles(i,2);
          t3(i) = angles(i,3);
          t4(i) = angles(i,4);
          t5(i) = angles(i,5);
          t6(i) = angles(i,6);

   % The D-H Transformation Matrices
          A1=[cos(t1(i)) 0 sin(t1(i))  0;
             sin(t1(i)) 0 -cos(t1(i)) 0;
             0       1      0       L1;
             0       0      0       1;];

   A2=[ cos(t2(i)) -sin(t2(i)) 0   L2*cos(t2(i));
            sin(t2(i)) cos(t2(i))  0   L2*sin(t2(i));
                                  0      0       1      D1;
                                  0      0       0      1;];

          A3=[ cos(pi/2+t3(i)) 0 sin(pi/2+t3(i))  0;
              sin(pi/2+t3(i)) 0 -cos(pi/2+t3(i)) 0;
                                          0      1   0              0;
                                          0      0   0              1;];

          A4=[cos(t4(i)) 0 -sin(t4(i))  0;
             sin(t4(i)) 0 cos(t4(i))   0;
          0       -1    0         L3;
          0        0    0         1;];

          A5=[ cos(t5(i)) 0 sin(t5(i))   0;
           sin(t5(i)) 0 -cos(t5(i))  0;
                  0     1     0         0;
                  0     0     0         1;];

   A6=[ cos(t6(i))  -sin(t6(i))   0   0;
```

```
        sin(t6(i))  cos(t6(i))    0   0;
                                  0        0        1  L4;
            0         0        0   1;];

    A7=[cos(pi)  -sin(pi)       0    -sp2;
        sin(pi)   cos(pi)      0       0;
          0        0          1     sp1;
          0        0          0       1;];

            T1 =A1;
            T2 =T1*A2;
            T3 =T2*A3;
    T4 =T3*A4;
    T5 =T4*A5;
    T6 =T5*A6;
    T7 =T6*A7;

    % The Positions of the origins of the D-H frames
            po0(:,i) = [0; 0; 0];
            po1(:,i) = T1(1:3,4);
            po2(:,i) = T2(1:3,4);
    po3(:,i) = T3(1:3,4);
            po4(:,i) = T4(1:3,4);
    po5(:,i) = T5(1:3,4);
    po6(:,i) = T6(1:3,4);

    % Computing the location of a point on the end-effector to be plotted
    Tinter= T6*[ 1 0 0 0; 0 1 0 0; 0 0 1 sp1; 0 0 0 1;];
    pointer(:,i) = Tinter(1:3,4);
    po7(:,i) = T7(1:3,4);

    % The Z axis orientation of every frame to be used in the Jacobian Calculations
    z0 = [0; 0; 1];   z1 = T1(1:3,3);    z2 = T2(1:3,3);    z3 = T3(1:3,3);    z4 = T4(1:3,3);    z5 = T5(1:3,3);

    p0 = po0(:,i);     p1 = po1(:,i);      p2 = po2(:,i);      p3 = po3(:,i);     p4 = po4(:,i);     p5 = po5(:,i);
    p6 = po6(:,i);     p7 = po7(:,i);

    % The Jacobian
            Jac = [ cross(z0,(p7-p0))     cross(z1,(p7-p1))     cross(z2,(p7-p2))   cross(z3,(p7-p3))   cross(z4,(p7-p4))
cross(z5,(p7-p5));
    z0                                    z1            z2                            z3                  z4              z5];

    det_Jac(i) = det(Jac);

    % The Joint Velocities
    Vel(:,i) = inv(Jac) * eV;
    % The Joint Torques
    Tor(:,i) = inv(Jac) * eF;
end
```

E.5 THE INVERSE KINEMATICS FUNCTION

```
function [solution,out] = inverse(X,Y,Z,S,spec1,spec2)
% INVERSE returns the 8 set of solutions
% and returns 1 if the point is out of reach
% The inputs are X, Y, Z, the coordinates of a chosen point
% and S, the orientation of the end-effector wrt the base frame

out = 0;    % 0 unless set to 1 later in the function

% The dimensions of the Merlin
L1=46.4;  % length of link 1
D1=11.9;  % offset of joint 2 wrt the axis of joint 1
L2=17.375;% length of link 2
L3=17.25; % length of link 3
L4=3.5;   % length of link 5

% Computing the position of the wrist center (P4),
% using the position and orientation of the end-effector
X4 = X - ((L4+spec1)* S(1,3) + spec2 * S(1,1));
Y4 = Y - ((L4+spec1)* S(2,3) + spec2 * S(2,1));
Z4 = Z - ((L4+spec1)* S(3,3) + spec2 * S(3,1));

% s is the projection of link 2 and 3 on the XY plane
if (X4*X4+Y4*Y4-D1*D1) <0
   out = 1;    % set to 1 : no solution: the wrist is too close to center of the robot
   solution = zeros(8,6);
else
   s = sqrt(X4*X4+Y4*Y4-D1*D1);

   % The two possible angles for joint 1
   Angle1a = atan2(Y4,X4) + atan2(D1, s);
   Angle1b = atan2(Y4,X4) - atan2(D1, s) + pi;

   % Calculating the cosine of joint 3
   cos3 = (s*s + (Z4-L1)*(Z4-L1) - (L3*L3+L2*L2))/(2*L3*L2);

   % Checking if a solution exists
   if abs(cos3) > 1
      out = 1;                    % in this case, the chosen point is not in
                            % the reachable workspace of the robot
      solution = zeros(8,6);
   else
      sin3a = sqrt(1 - cos3*cos3);
      sin3b = - sqrt(1 - cos3*cos3);

      % Two possible angles for joint 3 when joint 1 angle is equal to Angle1a
      Angle3aa = atan2(sin3a, cos3);
      Angle3ab = - atan2(sin3a, cos3);

      % Two possible angles for joint 3 when joint 1 angle is equal to Angle1b
      Angle3ba = atan2(sin3b, cos3);
      Angle3bb = - atan2(sin3b, cos3);

      % Calculating cosines and sines of joint 2. Two possible angles
      sin2a = ((L2+L3*cos3)*(Z4-L1) - L3*s*sin3a)/ (s*s+(Z4-L1)*(Z4-L1));
```

```
        sin2b = ((L2+L3*cos3)*(Z4-L1) - L3*s*sin3b)/ (s*s+(Z4-L1)*(Z4-L1));
        cos2a = ((L2+L3*cos3)* s + L3*(Z4-L1)*sin3a)/(s*s+(Z4-L1)*(Z4-L1));
        cos2b = ((L2+L3*cos3)* s + L3*(Z4-L1)*sin3b)/(s*s+(Z4-L1)*(Z4-L1));

        % Two possible angles for joint 2 when joint 1 angle is equal to Angle1a
        Angle2aa = atan2(sin2a, cos2a);
        Angle2ab = pi - atan2(sin2a, cos2a);

        % Two possible angles for joint 2 when joint 1 angle is equal to Angle1b
        Angle2ba = atan2(sin2b, cos2b);
        Angle2bb = pi - atan2(sin2b, cos2b);

        % Four possible solutions with joints 1, 2 and 3
        sol1 = [Angle1a ; Angle2aa ; Angle3aa];
        sol2 = [Angle1a ; Angle2ba ; Angle3ba];
        sol3 = [Angle1b ; Angle2ab ; Angle3ab];
        sol4 = [Angle1b ; Angle2bb ; Angle3bb];

        % For each of the above 4 solutions, the rotation matrix
        % from the base frame to frame 3 is computed below

        % For solution 1
        R13sol1     =[-cos(sol1(1))*cos(sol1(2))*sin(sol1(3))-cos(sol1(1))*sin(sol1(2))*cos(sol1(3)),          sin(sol1(1)),
cos(sol1(1))*cos(sol1(2))*cos(sol1(3))-cos(sol1(1))*sin(sol1(2))*sin(sol1(3));
              -sin(sol1(1))*cos(sol1(2))*sin(sol1(3))-sin(sol1(1))*sin(sol1(2))*cos(sol1(3)),                 -cos(sol1(1)),
sin(sol1(1))*cos(sol1(2))*cos(sol1(3))-sin(sol1(1))*sin(sol1(2))*sin(sol1(3));
                              -sin(sol1(2))*sin(sol1(3))+cos(sol1(2))*cos(sol1(3)),                                        0,
sin(sol1(2))*cos(sol1(3))+cos(sol1(2))*sin(sol1(3))];

        % For solution 2
        R13sol2     =[-cos(sol2(1))*cos(sol2(2))*sin(sol2(3))-cos(sol2(1))*sin(sol2(2))*cos(sol2(3)),          sin(sol2(1)),
cos(sol2(1))*cos(sol2(2))*cos(sol2(3))-cos(sol2(1))*sin(sol2(2))*sin(sol2(3));
              -sin(sol2(1))*cos(sol2(2))*sin(sol2(3))-sin(sol2(1))*sin(sol2(2))*cos(sol2(3)),                 -cos(sol2(1)),
sin(sol2(1))*cos(sol2(2))*cos(sol2(3))-sin(sol2(1))*sin(sol2(2))*sin(sol2(3));
                              -sin(sol2(2))*sin(sol2(3))+cos(sol2(2))*cos(sol2(3)),                                        0,
sin(sol2(2))*cos(sol2(3))+cos(sol2(2))*sin(sol2(3))];

        % For solution 3
        R13sol3     =[-cos(sol3(1))*cos(sol3(2))*sin(sol3(3))-cos(sol3(1))*sin(sol3(2))*cos(sol3(3)),          sin(sol3(1)),
cos(sol3(1))*cos(sol3(2))*cos(sol3(3))-cos(sol3(1))*sin(sol3(2))*sin(sol3(3));
              -sin(sol3(1))*cos(sol3(2))*sin(sol3(3))-sin(sol3(1))*sin(sol3(2))*cos(sol3(3)),                 -cos(sol3(1)),
sin(sol3(1))*cos(sol3(2))*cos(sol3(3))-sin(sol3(1))*sin(sol3(2))*sin(sol3(3));
                              -sin(sol3(2))*sin(sol3(3))+cos(sol3(2))*cos(sol3(3)),                                        0,
sin(sol3(2))*cos(sol3(3))+cos(sol3(2))*sin(sol3(3))];

        % For solution 4
        R13sol4     =[-cos(sol4(1))*cos(sol4(2))*sin(sol4(3))-cos(sol4(1))*sin(sol4(2))*cos(sol4(3)),          sin(sol4(1)),
cos(sol4(1))*cos(sol4(2))*cos(sol4(3))-cos(sol4(1))*sin(sol4(2))*sin(sol4(3));
              -sin(sol4(1))*cos(sol4(2))*sin(sol4(3))-sin(sol4(1))*sin(sol4(2))*cos(sol4(3)),                 -cos(sol4(1)),
sin(sol4(1))*cos(sol4(2))*cos(sol4(3))-sin(sol4(1))*sin(sol4(2))*sin(sol4(3));
                              -sin(sol4(2))*sin(sol4(3))+cos(sol4(2))*cos(sol4(3)),                                        0,
sin(sol4(2))*cos(sol4(3))+cos(sol4(2))*sin(sol4(3))];

        % Using the above rotations and the orienation of the end-effector,
        % a rotation matrix from frame 4 to frame 7
        % is computed for each of the 4 solutions
```

```
R47sol1 = inv(R13sol1) * S;
R47sol2 = inv(R13sol2) * S;
R47sol3 = inv(R13sol3) * S;
R47sol4 = inv(R13sol4) * S;

% Two possible joint 5 angles for solution 1
Angle5a1 = atan2(sqrt(1-(R47sol1(3,3)*R47sol1(3,3))), R47sol1(3,3));
Angle5b1 = -Angle5a1;

% Two possible joint 4 angles for solution 1
Angle4a1 = atan2(R47sol1(2,3), R47sol1(1,3));
Angle4b1 = Angle4a1 +pi;

% Two possible joint 6 angles for solution 1
Angle6a1 = atan2(-R47sol1(3,2), R47sol1(3,1));
Angle6b1 = Angle6a1 +pi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two possible joint 5 angles for solution 2
Angle5a2 = atan2(sqrt(1-(R47sol2(3,3)*R47sol2(3,3))), R47sol2(3,3));
Angle5b2 = -Angle5a2;

% Two possible joint 4 angles for solution 2
Angle4a2 = atan2(R47sol2(2,3), R47sol2(1,3));
Angle4b2 = Angle4a2 +pi;

% Two possible joint 6 angles for solution 2
Angle6a2 = atan2(-R47sol2(3,2), R47sol2(3,1));
Angle6b2 = Angle6a2 +pi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two possible joint 5 angles for solution 3
Angle5a3 = atan2(sqrt(1-(R47sol3(3,3)*R47sol3(3,3))), R47sol3(3,3));
Angle5b3 = -Angle5a3;

% Two possible joint 4 angles for solution 3
Angle4a3 = atan2(R47sol3(2,3), R47sol3(1,3));
Angle4b3 = Angle4a3 +pi;

% Two possible joint 6 angles for solution 3
Angle6a3 = atan2(-R47sol3(3,2), R47sol3(3,1));
Angle6b3 = Angle6a3 +pi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Two possible joint 5 angles for solution 4
Angle5a4 = atan2(sqrt(1-(R47sol4(3,3)*R47sol4(3,3))), R47sol4(3,3));
Angle5b4 = -Angle5a4;

% Two possible joint 4 angles for solution 4
Angle4a4 = atan2(R47sol4(2,3), R47sol4(1,3));
Angle4b4 = Angle4a4 +pi;

% Two possible joint 6 angles for solution 4
Angle6a4 = atan2(-R47sol4(3,2), R47sol4(3,1));
Angle6b4 = Angle6a4 +pi;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The eight solutions
solution(1,:) = [ Angle1a  Angle2aa  Angle3aa  Angle4a1  Angle5a1  Angle6a1 ];
```

```
            solution(5,:) = [ Angle1a  Angle2aa  Angle3aa  Angle4b1  Angle5b1  Angle6b1 ];
            solution(2,:) = [ Angle1a  Angle2ba  Angle3ba  Angle4a2  Angle5a2  Angle6a2 ];
            solution(6,:) = [ Angle1a  Angle2ba  Angle3ba  Angle4b2  Angle5b2  Angle6b2 ];
            solution(3,:) = [ Angle1b  Angle2ab  Angle3ab  Angle4a3  Angle5a3  Angle6a3 ];
            solution(7,:) = [ Angle1b  Angle2ab  Angle3ab  Angle4b3  Angle5b3  Angle6b3 ];
            solution(4,:) = [ Angle1b  Angle2bb  Angle3bb  Angle4a4  Angle5a4  Angle6a4 ];
            solution(8,:) = [ Angle1b  Angle2bb  Angle3bb  Angle4b4  Angle5b4  Angle6b4 ];
   end
end
```

E.6 THE FUNCTION TO CHECK THE KINEMATIC LIMITS

```
function
[outlimit,detcheck,velcheck,torcheck]=checklimits(t,tm1,q,to,deter,outm1,outm2,outp1,determ1,determ2,deterp1,X7
,Y7,Z7,X7m1,Y7m1,Z7m1,S,spec1,spec2,V,F,k,inc)
% CHECKLIMITS returns
        % oulimit = 1 % if the hardware limits are exceeded
        % oulimit = 0 % if the hardware limits are not exceeded
        % detcheck = 1 % if the Jacobian determinant is close to zero
        % detcheck = 0 % if the Jacobian determinant is far from zero
        % velcheck = 1 % if the joints velociy limits are exceeded
        % velcheck = 0 % if the joints velocity limits are not exceeded
        % torcheck = 1 % if the joints torque limits are exceeded
        % torcheck = 0 % if the joints torque limits are not exceeded
% at a specific point on the path


% Limit angles
UL1 = 175*pi/180;
LL1 = -115*pi/180;
UL2 = 236*pi/180;
LL2 = -56*pi/180;
UL3 = 146*pi/180;
LL3 = -146*pi/180;
UL5 = 90*pi/180;
LL5 = -90*pi/180;

gearratio1 = 48; % the gear ratio of joint 1
gearratio2 = 48; % the gear ratio of joint 2
gearratio3 = 48; % the gear ratio of joint 3
gearratio4 = 24; % the gear ratio of joint 4
gearratio5 = 20; % the gear ratio of joint 5
gearratio6 = 24; % the gear ratio of joint 6

FS = 0.5;       % Factor of safety for torque and velocity checks

% Angular velocity of the 6 joints
vel1max = 16*2*pi/gearratio1;  % rad/sec
vel2max = 16*2*pi/gearratio2;  % rad/sec
vel3max = 16*2*pi/gearratio3;  % rad/sec
vel4max = 16*2*pi/gearratio4;  % rad/sec
vel5max = 16*2*pi/gearratio5;  % rad/sec
vel6max = 16*2*pi/gearratio6;  % rad/sec

% Torques limits of the 6 joints
torque1max = 1125 * gearratio1 / 16; % lb.in (divided by 16 to convert oz to lb)
torque2max = 1125 * gearratio2 / 16; % lb.in (divided by 16 to convert oz to lb)
torque3max = 1125 * gearratio3 / 16; % lb.in (divided by 16 to convert oz to lb)
torque4max = 400 * gearratio4 / 16;  % lb.in (divided by 16 to convert oz to lb)
torque5max = 400 * gearratio5 / 16;  % lb.in (divided by 16 to convert oz to lb)
torque6max = 400 * gearratio6 / 16;  % lb.in (divided by 16 to convert oz to lb)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Limit switch check
if t(1) < UL1 & t(1) > LL1 & t(2) < UL2 & t(2) > LL2 & t(3) < UL3 & t(3) > LL3 & t(5) < UL5 & t(5) > LL5
```

```
   outlimit = 0;
else
   outlimit = 1;
end

% Jacobian check
if outlimit == 0  % to save time

   if abs(deter) > 20
      detcheck = 0;

      if outm1 == 0  &  (deter * determ1 < -0.0000001)
         detcheck = 1;
      else
         if outm1 == 0  &  outm2 == 0  &  outp1 == 0  &  (((determ1 - determ2) * (deterp1 - deter)) < 0)  &  ((deter > 
0 &  (deterp1 - deter) > 0)  |  (deter < 0 &  (deterp1 - deter) < 0))
            finer = 10;
            for f = 1:1:(finer-1)
               x = X7m1 + (X7- X7m1)*f/finer;
               y = Y7m1 + (Y7- Y7m1)*f/finer;
               z = Z7m1 + (Z7- Z7m1)*f/finer;
               [solu,outc] = inverse(x,y,z,S,spec1,spec2);
               [det_Jac, Vel, Tor, p0, p1, p2, p3, p4, p5, p6, pinter, p7] = forward(solu, V, F, spec1, spec2);
               % abs(det_Jac(k))
               if abs(det_Jac(k)) < 20
                  detcheck = 1;
                  break
               end
            end
         end
      end

   else
      detcheck = 1;
   end

   % Joint velocity check
   if  (abs(q(1)) < (FS * vel1max)) & (abs(q(2)) < (FS * vel2max)) & (abs(q(3)) < (FS * vel3max)) & (abs(q(4)) < 
(FS * vel4max)) & (abs(q(5)) < (FS * vel5max)) & (abs(q(6)) < (FS * vel6max))
      velcheck = 0;

      if outm1 == 0  &  outm2 == 0  &  outp1 == 0  &  (((determ1 - determ2) * (deterp1 - deter)) < 0)  &  ((deter > 0 
&  (deterp1 - deter) > 0)  |  (deter < 0 &  (deterp1 - deter) < 0))
         finer = 25;
         for f = 1:1:(finer-1)
            x = X7m1 + (X7- X7m1)*f/finer;
            y = Y7m1 + (Y7- Y7m1)*f/finer;
            z = Z7m1 + (Z7- Z7m1)*f/finer;
            [solu,outc] = inverse(x,y,z,S,spec1,spec2);
            [det_Jac, Vel, Tor, p0, p1, p2, p3, p4, p5, p6, pinter, p7] = forward(solu, V, F, spec1, spec2);
            if (abs(Vel(1,k)) > (FS * vel1max)) | (abs(Vel(2,k)) > (FS * vel2max)) | (abs(Vel(3,k)) > (FS * vel3max)) | 
(abs(Vel(4,k)) > (FS * vel4max)) | (abs(Vel(5,k)) > (FS * vel5max)) | (abs(Vel(6,k)) > (FS * vel6max))
               velcheck = 1;
               break
            end
         end
      end
```

```
          if velcheck == 0
            ve=0.25;
            ang1 = tm1(1);
            ang2 = tm1(2);
            ang3 = tm1(3);
            ang4 = tm1(4);
            ang5 = tm1(5);
            ang6 = tm1(6);
            for f = 1:1:(inc/ve)
              x = X7m1 + (X7- X7m1)*f/(inc/ve);
              y = Y7m1 + (Y7- Y7m1)*f/(inc/ve);
              z = Z7m1 + (Z7- Z7m1)*f/(inc/ve);
              [solu,outc] = inverse(x,y,z,S,spec1,spec2);
              if (abs(solu(k,1)-ang1) > (FS * vel1max)) | (abs(solu(k,2)-ang2) > (FS * vel2max)) | (abs(solu(k,3)-
ang3) > (FS * vel3max)) | (abs(solu(k,4)-ang4) > (FS * vel4max)) | (abs(solu(k,5)-ang5) > (FS * vel5max)) |
(abs(solu(k,6)-ang6) > (FS * vel6max))
                velcheck = 1;
                break
              end
              ang1 = solu(k,1);
              ang2 = solu(k,2);
              ang3 = solu(k,3);
              ang4 = solu(k,4);
              ang5 = solu(k,5);
              ang6 = solu(k,6);
            end
          end
        end
    else
        velcheck = 1;
    end
else
    detcheck =0;
    velcheck =0;
end

% Joint torque check
if   (abs(to(1)) < FS * torque1max) & (abs(to(2)) < FS * torque2max) & (abs(to(3)) < FS * torque3max) &
(abs(to(4)) < FS * torque4max) & (abs(to(5)) < FS * torque5max) & (abs(to(6)) < FS * torque6max)
    torcheck = 0;
else
    torcheck = 1;
end
```

E.7 THE FUNCTION TO DETERMINE THE LINE BOUNDARIES

```
function [b1,b2,nb] = linebounds(out,outlimit,detcheck,torcheck,velcheck,d,k)
% LINEBOUNDS returns the number of lines and the boundary of the lines
% according to the limit checks values

% Find the boundary points on the POSSIBLE lines using the above checks
% Finding the boundaries of all possible segments along the line
nb=0;              % number of possible segments
b1=0;
b2=0;
newlineready = 1;   % number set to 1 to have new segments
for h = 1:1:d       % moving along the line
   if out(h)==0                     % checking if the point is reachable
      if outlimit(h,k)==0            % checking if the hardware limits are exceeded
         if detcheck(h,k)==0
            if torcheck(h,k) ==0         % checking if the torque limits are exceeded
               if velcheck(h,k) ==0    % checking if the velocity limits are exceeded
                  if newlineready == 1  % 1 if ready for a new segment
                     nb = nb + 1;        % incrementing the number of segments
                     b1(nb) = h;         % first boundary of the segment
                     newlineready = 0;   % 0 because the loops is waiting for
                                         % the second boundary of the actual segment
                  end
               else
                  if newlineready == 0
                     b2(nb) = h-1;        % second boundary of the segment
                  end
                  newlineready = 1;
               end
            else
               if newlineready == 0
                  b2(nb) = h-1;
               end
               newlineready = 1;
            end
         else
            if newlineready == 0
               b2(nb) = h-1;
            end
            newlineready = 1;
         end
      else
         if newlineready == 0
            b2(nb) = h-1;
         end
         newlineready = 1;
      end
   else
      if newlineready == 0
         b2(nb) = h-1;
      end
      newlineready = 1;
   end
end
```

E.8 THE FUNCTION TO DETECT COLLISION ON A FEASIBLE SEGMENT AND DETERMINE ALL FEASIBLE SUB-SEGMENTS

```
function [posisegf,posiseg, seg1,seg2]= collision(b1,b2,nb,t,k,X7,Y7,Z7,spec1,spec2,posiseg)
% COLLISION returns the new number of possible segment and their boundary
% based on collision detection

posisegf = posiseg;
% collision detection
for i = 1:1:nb  % for all possible lines possible from the eight solutions
   colproblem(i) =0;
   for h = b1(i):1:b2(i)  % for all possible segments
      An = [t(h,k,1);t(h,k,2);t(h,k,3);t(h,k,4);t(h,k,5);t(h,k,6);];
      for j = b1(i):1:b2(i)
         Point  = [ X7(j); Y7(j); Z7(j);];
         [colcheck(j,h),UU] = intersect(Point, An, spec1, spec2); % the link at h and the point at j
         if colcheck(j,h) ==1                              % if collision is detected
            colproblem(i) = colproblem(i) +1;
            touch(j,h)=1;
         else
            touch(j,h)=0;
         end
      end
   end
end
colproblem(i);      % to check if there is a collision at any line
                  % 0 : no collision
                  % > 0 : collision detected

% Storing all the possible sub-segments within every segments
jump = 0;
for i = 1:1:nb
   if colproblem(i)~=0
      for stpt = b1(i):1:b2(i)-1          % the starting point of the sub-segments along every segments
         posisegf = posisegf+1;      % incrementing the number of possible segments
         seg1(posisegf-posiseg) = stpt;
         for h = stpt:1:b2(i)
            Point  = [ X7(h); Y7(h); Z7(h);];   % the points along the tow line
            for j = stpt:1:h
               if colcheck(h,j)==1
                  jump = 1;
                  seg2(posisegf-posiseg) = h-1;
                  break
               else
                  jump=0;
               end
            end
            if jump ==1
               break
            else
               if h == b2(i)
                  seg2(posisegf-posiseg)= b2(i);
               end
            end
         end
```

```
        end
    else
        % No colllision at all
        posisegf = posisegf+1;
        seg1(posisegf-posiseg) = b1(i);
        seg2(posisegf-posiseg) = b2(i);
    end
end
```

E.9 THE FUNCTION FOR ELEMENTARY COLLISION INTERSECTION

```
function [detect, Corners] = intersect(P, sol, spec1, spec2)
%INTERSECT returns    1 if intersection of the INPUT point
%                     occurs with the robot links
%                   0 otherwise
%
%         returns    the position of the corners of the links
%
% The inputs are : the six angles of the Robot
%                (to calculate the positions of the Links)
%                and a Point in workspace

% The dimensions of the Merlin
L1=46.4; % length of link 1
D1=11.9; % offset of joint 2 wrt the axis of joint 1
L2=17.375; % length of link 2
L3=17.25; % length of link 3
L4=3.5; % length of link 5

% The Merlin six angles used to compute the location of the links
% The angles are inputs for the function
t1 = sol(1);
t2 = sol(2);
t3 = sol(3);
t4 = sol(4);
t5 = sol(5);
t6 = sol(6);

% The D-H Transformation Matrices
% used to get the positions of the D-H frames origines
A1=[    cos(t1) 0   sin(t1)     0;
             sin(t1) 0   -cos(t1)    0;
                  0       1   0           L1;
                  0       0   0           1;];

A2=[    cos(t2) -sin(t2)    0   L2*cos(t2);
             sin(t2) cos(t2)     0   L2*sin(t2);
             0       0           1      D1;
                  0       0           0      1;];

A3=[    cos(pi/2+t3)    0   sin(pi/2+t3)  0;
             sin(pi/2+t3)    0   -cos(pi/2+t3) 0;
                                   0       1       0         0;
                                   0       0       0         1;];

A4=[    cos(t4)   0  -sin(t4)   0;
             sin(t4)   0  cos(t4)    0;
                          0    -1    0       L3;
                          0     0    0       1;];

A5=[    cos(t5)     0   sin(t5)   0;
             sin(t5)     0   -cos(t5)  0;
                  0          1     0       0;
                  0          0     0       1;];
```

```
A6=[    cos(t6)  -sin(t6)  0   0;
        sin(t6)  cos(t6)   0   0;
                     0         0        1  L4;
                     0         0        0   1;];


A7=[    cos(pi)  -sin(pi)   0  -spec2;
        sin(pi)   cos(pi)   0       0;
        0         0         1   spec1;
        0         0         0       1;];

% The transformation matrices
T1 =A1;
T2 =T1*A2;
T3 =T2*A3;
T4 =T3*A4;
T5 =T4*A5;
T6 =T5*A6;
T7 =T6*A7;

% Link 1

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans1 = [  1 0 0  -15.5;
            0 1 0  3;
            0 0 1  -3.375;
            0 0 0  1;];
Trans1 = T2*Trans1;
Final1 = Trans1(1:3,4); % the point of link1 up front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans2 = [  1 0 0  -15.5;
            0 1 0  3;
            0 0 1  3.375;
            0 0 0  1;];
Trans2 = T2*Trans2;
Final2 = Trans2(1:3,4);% the point of link1 up front away to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans3 = [  1 0 0  -15.5;
            0 1 0  -3;
            0 0 1  -3.375;
            0 0 0  1;];
Trans3 = T2*Trans3;
Final3 = Trans3(1:3,4);% the point of link1 down front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans4 = [  1 0 0  -15.5;
            0 1 0  -3;
            0 0 1  3.375;
            0 0 0  1;];
Trans4 = T2*Trans4;
```

```
Final4 = Trans4(1:3,4);% the point of link1 down front away to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans5 = [  1 0 0  -26.5;
          0 1 0  6.5;
          0 0 1  -3.375;
          0 0 0  1;];
Trans5 = T2*Trans5;
Final5 = Trans5(1:3,4);% the point of link1 up middle close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans6 = [  1 0 0  -26.5;
          0 1 0  6.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans6 = T2*Trans6;
Final6 = Trans6(1:3,4);% the point of link1 up middle away to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans7 = [  1 0 0  -26.5;
          0 1 0  -6.5;
          0 0 1  -3.375;
          0 0 0  1;];
Trans7 = T2*Trans7;
Final7 = Trans7(1:3,4);% the point of link1 down middle close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans8 = [  1 0 0  -26.5;
          0 1 0  -6.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans8 = T2*Trans8;
Final8 = Trans8(1:3,4);% the point of link1 down middle away to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans9 = [  1 0 0  -33;
          0 1 0  6.5;
          0 0 1  -3.375;
          0 0 0  1;];
Trans9 = T2*Trans9;
Final9 = Trans9(1:3,4);% the point of link1 up back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans10 = [  1 0 0  -33;
          0 1 0  6.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans10 = T2*Trans10;
Final10 = Trans10(1:3,4);% the point of link1 up back away to center
```

```
% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans11 = [  1 0 0  -33;
          0 1 0  -6.5;
          0 0 1  -3.375;
          0 0 0  1;];
Trans11 = T2*Trans11;
Final11 = Trans11(1:3,4);% the point of link1 down back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 1
Trans12 = [  1 0 0  -33;
          0 1 0  -6.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans12 = T2*Trans12;
Final12 = Trans12(1:3,4);% the point of link1 down back away to center
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Link 2

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans13 = [ 1 0 0  3.5;
          0 1 0  3.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans13 = T2*Trans13;
Final13 = Trans13(1:3,4);% the point of link2 up front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans14 = [ 1 0 0  3.5;
          0 1 0  3.5;
          0 0 1  5.5+3.375;
          0 0 0  1;];
Trans14 = T2*Trans14;
Final14 = Trans14(1:3,4);% the point of link2 up front away to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans15 = [ 1 0 0  3.5;
          0 1 0  -3.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans15 = T2*Trans15;
Final15 = Trans15(1:3,4);% the point of link2 down front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans16 = [ 1 0 0  3.5;
          0 1 0  -3.5;
          0 0 1  5.5+3.375;
          0 0 0  1;];
Trans16 = T2*Trans16;
Final16 = Trans16(1:3,4);% the point of link2 down front away to center
```

```
% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans17 = [ 1 0 0  -20.87;
          0 1 0  3.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans17 = T2*Trans17;
Final17 = Trans17(1:3,4);% the point of link2 up  back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans18 = [ 1 0 0 -20.87;
          0 1 0  3.5;
          0 0 1  5.5+3.375;
          0 0 0  1;];
Trans18 = T2*Trans18;
Final18 = Trans18(1:3,4);% the point of link2 up back away to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans19 = [ 1 0 0 -20.87;
          0 1 0  -3.5;
          0 0 1  3.375;
          0 0 0  1;];
Trans19 = T2*Trans19;
Final19 = Trans19(1:3,4);% the point of link2 down back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans20 = [ 1 0 0 -20.87;
          0 1 0  -3.5;
          0 0 1  5.5+3.375;
          0 0 0  1;];
Trans20 = T2*Trans20;
Final20 = Trans20(1:3,4);% the point of link2 down back away to center
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Link 3

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans21 = [ 1 0 0  3;
          0 1 0  -3.375;
          0 0 1  17.5;
          0 0 0  1;];
Trans21 = T3*Trans21;
Final21 = Trans21(1:3,4);% the point of link3 up front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans22 = [ 1 0 0  3;
          0 1 0  3.375;
          0 0 1  17.5;
          0 0 0  1;];
Trans22 = T3*Trans22;
```

```
Final22 = Trans22(1:3,4);% the point of link3 up front away to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans23 = [ 1 0 0  -3;
          0 1 0  -3.375;
          0 0 1  17.5;
          0 0 0  1;];
Trans23 = T3*Trans23;
Final23 = Trans23(1:3,4);% the point of link3 down front close to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans24 = [ 1 0 0  -3;
          0 1 0  3.375;
          0 0 1  17.5;
          0 0 0  1;];
Trans24 = T3*Trans24;
Final24 = Trans24(1:3,4);% the point of link3 down front away to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans25 = [ 1 0 0  3.75;
          0 1 0  -3.375;
          0 0 1  -14;
          0 0 0  1;];
Trans25 = T3*Trans25;
Final25 = Trans25(1:3,4);% the point of link3 up  back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans26 = [ 1 0 0  3.75;
          0 1 0  3.375;
          0 0 1  -14;
          0 0 0  1;];
Trans26 = T3*Trans26;
Final26 = Trans26(1:3,4);% the point of link3 up back away to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans27 = [ 1 0 0  -3.75;
          0 1 0  -3.375;
          0 0 1  -14;
          0 0 0  1;];
Trans27 = T3*Trans27;
Final27 = Trans27(1:3,4);% the point of link3 down back close to center

% Transformation from the origin of the frame
% to the desired corner on the link 3
Trans28 = [ 1 0 0  -3.75;
          0 1 0  3.375;
          0 0 1  -14;
          0 0 0  1;];
Trans28 = T3*Trans28 ;
Final28 = Trans28(1:3,4);% the point of link3 down back away to center
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% The Body

% Transformation from the origin of the frame
% to the desired corner on the body
Trans29 = [ 1 0 0  6.7;
           0 1 0  10;
           0 0 1  8.5;
           0 0 0  1;];
Trans29 = T1*Trans29;
Final29 = Trans29(1:3,4);% the point on the body up front right

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans30 = [ 1 0 0  6.7;
           0 1 0  -14.4;
           0 0 1  8.5;
           0 0 0  1;];
Trans30 = T1*Trans30;
Final30 = Trans30(1:3,4);% the point on the body down front right

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans31 = [ 1 0 0  6.7;
           0 1 0  -14.4;
           0 0 1  -8.5;
           0 0 0  1;];
Trans31 = T1*Trans31;
Final31 = Trans31(1:3,4);% the point on the body down front left

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans32 = [ 1 0 0  6.7;
           0 1 0  10;
           0 0 1  -8.5;
           0 0 0  1;];
Trans32 = T1*Trans32;
Final32 = Trans32(1:3,4);% the point on the body up front left

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans33 = [ 1 0 0  -20.3;
           0 1 0  8.6;
           0 0 1  8.5;
           0 0 0  1;];
Trans33 = T1*Trans33;
Final33 = Trans33(1:3,4);% the point on the body up back right

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans34 = [ 1 0 0 -20.3;
           0 1 0 -14.4;
           0 0 1  8.5;
           0 0 0  1;];
Trans34 = T1*Trans34;
Final34 = Trans34(1:3,4);% the point on the body down back right
```

```
% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans35 = [ 1 0 0  -20.3;
          0 1 0  -14.4;
          0 0 1  -8.5;
          0 0 0  1;];
Trans35 = T1*Trans35;
Final35 = Trans35(1:3,4);% the point on the body down back left

% Transformation from the origin of the frame
% to the desired corner on the link 2
Trans36 = [ 1 0 0 -20.3;
          0 1 0  8.6;
          0 0 1  -8.5;
          0 0 0  1;];
Trans36 = T1*Trans36;
Final36 = Trans36(1:3,4);% the point on the body up back left
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with link1

% The dot product of the point vector with the normal vector of the surface
if  dot(P - Final9 , cross(Final10 - Final9 , Final11 - Final9)) >= 0
  col1 = 1;
  % point inside the arm on one side
else
  col1 = 0;
  % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot ( P - Final4 , cross(Final4 - Final3 , Final1 - Final3)) >= 0
  col2 = 1;
  % point inside the arm on one side
else
  col2 = 0;
  % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot (P- Final2, cross(Final4 - Final2 , Final6 - Final2)) >= 0
  col3 = 1;
  % point inside the arm on one side
else
  col3 = 0;
  % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot( P-Final3 , cross(Final1 - Final3 , Final7 - Final3)) >= 0
  col4 = 1;
  % point inside the arm on one side
else
  col4 = 0;
  % point outside the arm on one side
end
```

```
% The dot product of the point vector with the normal vector of the surface
if  dot( P- Final9, cross(Final5 - Final9 , Final10 - Final9)) >= 0
   col5 = 1;
   % point inside the arm on one side
else
   col5 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot( P - Final11, cross(Final12 - Final11 , Final7 - Final11)) >= 0
   col6 = 1;
   % point inside the arm on one side
else
   col6 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot( P - Final5, cross(Final1 - Final5 , Final6 - Final5)) >= 0
   col7 = 1;
   % point inside the arm on one side
else
   col7 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final7, cross(Final8 - Final7 , Final3 - Final7)) >= 0
   col8 = 1;
   % point inside the arm on one side
else
   col8 = 0;
   % point outside the arm on one side
end

% Check all above conditions to confirm collision with link 1
if col1 == 1 & col2==1 & col3== 1 & col4==1 & col5==1  &  col6==1 & col7==1 &  col8==1
   collision1 =1;
else
   collision1 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with link2

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final15 , cross(Final16 - Final15 , Final13 - Final15)) >= 0
   col15 = 1;
   % point inside the arm on one side
else
   col15 = 0;
   % point outside the arm on one side
end
```

```
% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final19 , cross(Final17 - Final19 , Final20 - Final19)) >= 0
   col16 = 1;
   % point inside the arm on one side
else
   col16 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final16 , cross(Final15 - Final16 , Final20 - Final16)) >= 0
   col17 = 1;
   % point inside the arm on one side
else
   col17 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final13, cross(Final14 - Final13 , Final17 - Final13)) >= 0
   col18 = 1;
   % point inside the arm on one side
else
   col18 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final16 , cross(Final20 - Final16 , Final14 - Final16)) >= 0
   col19 = 1;
   % point inside the arm on one side
else
   col19 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final15 , cross(Final13 - Final15 , Final19 - Final15)) >= 0
   col20 = 1;
   % point inside the arm on one side
else
   col20 = 0;
   % point outside the arm on one side
end


% Check all above conditions to confirm collision with link 2
if col15 == 1 & col16==1 & col17== 1 & col18==1 & col19==1  &  col20==1
   collision2 =1;
else
   collision2 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with link3
% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final24 , cross(Final22 - Final24 , Final23 - Final24)) >= 0
   col9 = 1;
   % point inside the arm on one side
else
   col9 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final27 , cross(Final25 - Final27 , Final28 - Final27)) >= 0
   col10 = 1;
   % point inside the arm on one side
else
   col10 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final23, cross(Final21 - Final23 , Final27 - Final23)) >= 0
   col11 = 1;
   % point inside the arm on one side
else
   col11 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final24 , cross(Final28 - Final24 , Final22 - Final24)) >= 0
   col12 = 1;
   % point inside the arm on one side
else
   col12 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final22 ,cross(Final26 - Final22 , Final21 - Final22)) >= 0
   col13 = 1;
   % point inside the arm on one side
else
   col13 = 0;
   % point outside the arm on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final24 , cross(Final23 - Final24 , Final28 - Final24)) >= 0
   col14 = 1;
   % point inside the arm on one side
else
   col14 = 0;
   % point outside the arm on one side
end

% Check all above conditions to confirm collision with link 3
if col9 == 1 & col10==1 & col11== 1 & col12==1 & col13==1  &  col14==1
```

```
  collision3 =1;
else
  collision3 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with the body

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final29 , cross(Final32 - Final29 , Final30 - Final29)) >= 0
  col21 = 1;
  % point inside the body on one side
else
  col21 = 0;
  % point outside the body on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final35 , cross(Final36 - Final35 , Final34 - Final35)) >= 0
  col22 = 1;
  % point inside the body on one side
else
  col22 = 0;
  % point outside the body on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final35 , cross(Final31 - Final35 , Final36 - Final35)) >= 0
  col23 = 1;
  % point inside the body on one side
else
  col23 = 0;
  % point outside the body on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final34, cross(Final33 - Final34 , Final30 - Final34)) >= 0
  col24 = 1;
  % point inside the body on one side
else
  col24 = 0;
  % point outside the body on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final36 , cross(Final32 - Final36 , Final33 - Final36)) >= 0
  col25 = 1;
  % point inside the body on one side
else
  col25 = 0;
  % point outside the body on one side
end

% The dot product of the point vector with the normal vector of the surface
if  dot(P-Final35 , cross(Final34 - Final35 , Final31 - Final35)) >= 0
  col26 = 1;
```

```
   % point inside the body on one side
else
   col26 = 0;
   % point outside the body on one side
end

% Check all above conditions to confirm collision with the body
if col21 == 1 & col22==1 & col23== 1 & col24==1 & col25==1  &  col26==1
   collision4 =1;
else
   collision4 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with the base

% Check if the point is within the base
if  P(3) <= 7  &  abs(P(1)) <= 18  &  abs(P(2)) <= 18
   collision5 =1;
else
   collision5 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with the cylinder base

% Check if the point is within the cylinder
if  sqrt( P(1) * P(1) +  P(2) *P(2) ) <= 3  &  P(3) <= 25  &  P(3) >= 7
   collision6 =1;
else
   collision6 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check collision with the floor

% Check if the point is within the cylinder
if  P(3) <= 0
   collision7 =1;
else
   collision7 =0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Overall Collision

 % Detecting collision with either one of the three links
 if collision1 ==1  | collision2 ==1  |  collision3 ==1  | collision4 ==1  | collision5 ==1  |  collision6 ==1  |  collision7
==1
    detect = 1;
    % Collision detected as the first output of the function
 else
    detect = 0;
    % No collision detected as the first ouput of the function
 end
```

```
% Store the positions of the corners for the second output
% Those outputs are used for plotting the actual links
Corners= [Final1, Final2, Final3, Final4, Final5, Final6, Final7, Final8, Final9, Final10, Final11, Final12, Final13,
Final14, Final15, Final16, Final17, Final18, Final19, Final20, Final21, Final22, Final23, Final24, Final25, Final26,
Final27, Final28, Final29, Final30, Final31, Final32, Final33, Final34, Final35, Final36];
```

E.10 THE FUNCTION TO DETERMINE THE FEASIBLE SQUARES

```
function [centerloc] = squareXY(side, yawval, line_inc, pos_inc, P1, P2, segment)
% returns the X, Y, Z positions for the centers of all
% the possible squares within the envelope in the XY plane
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% side is the side length of the square
% yawval is the yaw of the lines building the squares
% line_inc is the distance between the lines
% pos_inc is the distance between the points along the lines
% P1 is a 3 dimensional array storing the position of the
% first boundary of all the lines
% P2 is a 3 dimensional array storing the position of the
% second boundary of all the lines
% segment is 2 dimensional array that stores the nb of segments at every offset


% In the case where the lines simulated are not parallel to the Y axis
if yawval ~= pi/2  &  yawval ~= -pi/2

   % Distance between the points along the lines
         var_inc = pos_inc;
         % Setting the number of possible squares to zero
   posisquare =0;

   % For all offsets
         for o = 1:1:size(P1,1)
                 o
      % To cut out the lines where NO squares can be built next to them
            if (size(P1,1)-o)*line_inc >= side -0.000001   % the small number is included for the >= inequality

         % For all the segments at each offset
         for i = 1:1:segment(o)
             % To cut out the segments smaller than the side length of the square
           if   sqrt(   (P1(o,i,1)-P2(o,i,1))*(P1(o,i,1)-P2(o,i,1))      +      (P1(o,i,2)-P2(o,i,2))*   (P1(o,i,2)-P2(o,i,2))   +
(P1(o,i,3)- P2(o,i,3)) * (P1(o,i,3)- P2(o,i,3)) )  >= side-0.0000001
            v =  0;
              % To store the Y position of the center of the potential squares
            yvariable = P1(o,i,2)+ var_inc*sin(yawval);
                     % Moving along the segment when the segment is longer than the side length of the square
             for variable = P1(o,i,1): -var_inc*cos(yawval): P2(o,i,1)
                       % Computing the Y position of the centers
                yvariable =  yvariable - var_inc*sin(yawval);

                % To stop moving along the segment
                % when the other boundary of the segment is hit
                if variable >= (P2(o,i,1)+side*cos(yawval)) -0.0000001  % to enter the 'if' the numbers are equal
                        v = v+1;
                   % Variable to count the number of offsets that can fill the square
                        candidate(o,i,v) =0;
                   % Checking the offsets next to the original offset
                        for j = (o+1):1:(o+(side/line_inc))
                     % To stop the loop if one of the offsets cannot hold the square
                     if candidate(o,i,v) ~= j-o-1
                        break
                              else
```

```
                                       % To check all the segments in one offset
                              for k = 1:1:segment(j)
                                % To check if the next segment can hold the squares
                                if  (P1(j,k,1)+(j-o)*  line_inc  *  sin(yawval))  >=  variable-0.00001  &  (P2(j,k,1)+(j-
o)*line_inc*sin(yawval)) <= variable-side*cos(yawval)+0.00000001
                                         % Incrementing the variable that counts the nb of offsets
                                  candidate(o,i,v) = candidate(o,i,v) +1;
                                         break
                                       end
                              end
                                    end
                                    end
                         % If the number of offsets is large enough to fit the square
                                             if candidate(o,i,v) >= side/line_inc -0.0000001
                                       % Incrementing the number of possible squares
                           posisquare = posisquare+1;
                           % To store the X, Y, Z positions of the center of the square
                                    centerloc(posisquare,     1)     =     ((variable+variable-side*cos(yawval))/2)-
(side/2)*sin(yawval);
                                    centerloc(posisquare,    2)    =    ((yvariable+yvariable-side*sin(yawval))/2)    +
(side/2)*cos(yawval);
                                    centerloc(posisquare, 3) = P1(1,1,3);
                                    end
                                                        end
                                  end
                                  end
                  end
                end
                end
end


% In the case where the lines simulated are parallel to the Y axis
%         and the motion of the e.e. is in the negative direction
if yawval == pi/2

   % Distance between the points along the lines
           var_inc = pos_inc;
           % Setting the number of possible squares to zero
   posisquare =0;

   % For all offsets
           for o = 1:1:size(P1,1)
                  o
      % To cut out the lines where NO squares can be built next to them
             if (size(P1,1)-o)*line_inc >= side-0.00000001     % the small number is included for the >= inequality
          % For all the segments at each offset
          for i = 1:1:segment(o)
             % To cut out the segments smaller than the side length of the square
            if   sqrt(   (P1(o,i,1)-P2(o,i,1))*(P1(o,i,1)-P2(o,i,1))     +     (P1(o,i,2)-P2(o,i,2))*   (P1(o,i,2)-P2(o,i,2))   +
(P1(o,i,3)- P2(o,i,3)) * (P1(o,i,3)- P2(o,i,3)) )  >= side-0.0000001
              v =  0;
             % To store the X position of the center of the potential squares
              xvariable = P1(o,i,1);
                    % Moving along the segment when the segment is longer than the side length of the square
              for variable = P1(o,i,2): -var_inc: P2(o,i,2)
```

```
                    % To stop moving along the segment
                    % when the other boundary of the segment is hit
                    if variable >= (P2(o,i,2)+side) -0.0000001  % to enter the 'if' the numbers are equal
                        v = v+1;
                        % Variable to count the number of offsets that can fill the square
                            candidate(o,i,v) =0;
                        % Checking the offsets next to the original offset
                            for j = (o+1):1:(o+(side/line_inc))
                            % To stop the loop if one of the offsets cannot hold the square
                            if candidate(o,i,v) ~= j-o-1
                                break
                            else
                                % To check all the segments in one offset
                                for k = 1:1:segment(j)
                                    % To check if the next segment can hold the squares
                                        if P1(j,k,2) >= variable-0.000001  & P2(j,k,2) <= variable-side+0.0000001
                                            % Incrementing the variable that counts the nb of offsets
                                    candidate(o,i,v) = candidate(o,i,v) +1;
                                            break
                                        end
                                        end
                                    end
                                    end
                                            % If the number of offsets is large enough to fit the square
                                    if candidate(o,i,v) >= side/line_inc-0.00000001
                            % Incrementing the number of possible squares
                                        posisquare = posisquare+1;
                            % To store the X, Y, Z positions of the center of the square
                                    centerloc(posisquare, 2) = ((variable+variable-side)/2);
                                    centerloc(posisquare, 1) = (xvariable) + (side/2);
                                    centerloc(posisquare, 3) = P1(1,1,3);
                                    end
                                                    end
                                end
                                end
                end
            end
            end
end


% In the case where the lines simulated are parallel to the Y axis
%         and the motion of the e.e. is in the positive direction
if yawval == -pi/2

    % Distance between the points along the lines
            var_inc = pos_inc;
            % Setting the number of possible squares to zero
    posisquare =0;

    % For all offsets
            for o = 1:1:size(P1,1)
                    o
        % To cut out the lines where NO squares can be built next to them
            if (size(P1,1)-o)*line_inc >= side-0.00000001    % the small number is included for the >= inequality
```

```
          % For all the segments at each offset
          for i = 1:1:segment(o)
             % To cut out the segments smaller than the side length of the square
             if   sqrt(   (P1(o,i,1)-P2(o,i,1))*(P1(o,i,1)-P2(o,i,1))      +      (P1(o,i,2)-P2(o,i,2))*   (P1(o,i,2)-P2(o,i,2))   +
(P1(o,i,3)- P2(o,i,3)) * (P1(o,i,3)- P2(o,i,3)) )  >= side-0.0000001
               v =  0;
              % To store the X position of the center of the potential squares
               xvariable = P1(o,i,1);
                        % Moving along the segment when the segment is longer than the side length of the square
               for variable = P1(o,i,2): var_inc: P2(o,i,2)

                  % To stop moving along the segment
                  % when the other boundary of the segment is hit
                  if variable <= (P2(o,i,2)-side) +0.0000001  % to enter the 'if' the numbers are equal
                     v = v+1;
                     % Variable to count the number of offsets that can fill the square
                           candidate(o,i,v) =0;
                     % Checking the offsets next to the original offset
                           for j = (o+1):1:(o+(side/line_inc))
                        % To stop the loop if one of the offsets cannot hold the square
                        if candidate(o,i,v) ~= j-o-1
                           break
                                 else
                           % To check all the segments in one offset
                           for k = 1:1:segment(j)
                              % To check if the next segment can hold the squares
                                    if P1(j,k,2) <= variable+0.000001  &  P2(j,k,2) >= variable+side-0.0000001
                                       % Incrementing the variable that counts the nb of offsets
                                 candidate(o,i,v) = candidate(o,i,v) +1;
                                       break
                                    end
                                    end
                                 end
                                 end
                     % If the number of offsets is large enough to fit the square
                                             if candidate(o,i,v) >= side/line_inc-0.00000001
                        % Incrementing the number of possible squares
                                    posisquare = posisquare+1;
                        % To store the X, Y, Z positions of the center of the square
                                 centerloc(posisquare, 2) = ((variable+variable+side)/2);
                                 centerloc(posisquare, 1) = (xvariable) + (side/2);
                                 centerloc(posisquare, 3) = P1(1,1,3);
                                 end
                                                   end
                               end
                               end
              end
            end
            end
end
```

E.11 THE CODE TO DETERMINE THE FEASIBLE ISOTROPIC COUPONS

```
% This file locates and draws feasable circle coupons
% The coupons have layers with four different orientations
% Squares with a different orientation are plotted in the plane
% The squares build the circles coupons
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% To clear all stored data
clear all
% To close all Matlab figures
close all

% The dimensions of the Merlin
L1=46.4; % length of link 1
D1=11.9; % offset of joint 2 wrt the axis of joint 1
L2=17.375; % length of link 2
L3=17.25; % length of link 3
L4=3.5; % length of link 5

% The dimensions of the end-effector
spec1 = 3.81;  % the distance from the center of the face plate to contact point on the roller along the Z axis
spec2 = 6.87;  % the distance from the center of the face plate to contact point on the roller along the X axis

% Dynamic Properties of the Merlin and the e.e.
Sp = 0.25; % in/sec
Fc = 5;   % the compression force in lbf
Wee = 15;  % the estimated weight of the e.e. in lbf

Xoffset = 0;
Yoffset = 0;
Zoffset = 0;

% Entering the desired plane
plane = input('Enter the number representing the desired plane:  1 for XY,  2 for XZ,  3 for YZ: ');

% For every layer in the coupon
for layer = 1:1:4
   % Choosing the orientation of every layer
   switch layer
   case 1
      pitch = 0*pi/180;
      yaw = -22.5*pi/180;
      rot = 90*pi/180;
      roll = 0*pi/180;
   case 2
      pitch = 0*pi/180;
      yaw = 22.5*pi/180;
      rot = 90*pi/180;
      roll = 0*pi/180;
   case 3
      pitch = 0*pi/180;
      yaw = 67.5*pi/180;
      rot = 90*pi/180;
      roll = 0*pi/180;
   case 4
```

```
    pitch = 0*pi/180;
    yaw = -67.5*pi/180;
    rot = 90*pi/180;
    roll = 0*pi/180;
end

% In order to generate the desired lines, make sure you
% insert the right parameters: yaw, pitch, and roll make big differences

% The parameters of the lines to generate
% pitch = 0 *pi/180; % the lines pitch
% yaw = 0 *pi/180; % the lines yaw
% roll  = 0 *pi/180; % the lines roll
orient = 90*pi/180; % The orientation of the face plate wrt the lines
% rot    =  90 *pi/180; % the rotation of the e.e. ball around the normal of the tow path
if spec2 == 0
    orient = orient + 90*pi/180;
end


% The line offsets from the origin of the base frame
% but the Zoffset is compared to the shoulder joint center
if pitch == pi/2  |  pitch == -pi/2
    Xoffset = 20;
    Yoffset = 15;
else
    if  yaw == pi/2  | yaw == -pi/2
        Xoffset = 20;
        Zoffset = -18;
    else
        Yoffset = 0;
        Zoffset = -18;
    end
end

% The loop starts here
% Index used in the loop
o =0;
% Parameter for the first line
off1 = -100;
% Parameter for the last line
off2 = 100;
% Parameter increment between the lines - spacing between the lines in inches -
offinc = 1;

% Varying the Lines
for off = off1:offinc:off2
    off
    % To choose the varying offset parameter
    % depending on the lines angle parameters and the desired plane
    switch plane
    case 1
        if yaw == pi/2 | yaw == -pi/2
            Xoffset = off;
        else
            Yoffset = off/cos(yaw);
    end
case 2
    if pitch == pi/2 | pitch == -pi/2
        Xoffset = off;
    else
        Zoffset = off/cos(pitch);
    end
case 3
    if pitch == pi/2 | pitch == -pi/2
        Yoffset = off;
    else
        Zoffset = off/cos(pitch);
    end
end

% Incrementing the index
o = o+1;

% Setting all the loop results to zero
posiseg(o) = 0; % number of possible segment at each offset
Lopt(o)=0;
Xopt_st(o)= 0;
Yopt_st(o)= 0;
Zopt_st(o)= 0;
Xopt_ed(o)= 0;
Yopt_ed(o)= 0;
Zopt_ed(o)= 0;

[Mp,S,V,F] = ee_dynamics(yaw,pitch,roll,orient,rot,Sp,Fc,Wee,Xoffset,Yoffset,Zoffset);

% Inverse Dynamics for the line
% The loop that moves the points along the line
inc = 1;  % distance between the points on the line (in inches)
d=0;      % counter set to zero

for distance = 100:-inc:-100
    d = d+1;              % incrementing the counter
    dist(d) = distance;

    % M4 is changing with the distance
    M4 = [ 1  0  0  distance; 0 1 0 0; 0 0 1 0;   0 0 0 1;];
    % Computing the matrix to give the position of the point
    M = Mp*M4;

    % The position of the point: the e.e. position
    X(d) = M(1,4);
    Y(d) = M(2,4);
    Z(d) = M(3,4);

    [solution,out(d)] = inverse(X(d),Y(d),Z(d),S,spec1,spec2);

    if out(d) == 0
        [det_Jac, Vel, Tor, p0, p1, p2, p3, p4, p5, p6, pinter, p7] = forward(solution, V, F, spec1, spec2);

        % Storing the angles
        for i = 1:1:8
```

```
            for an = 1:1:6
                % Storing the angles
                t(d,i,an) = solution(i,an);
                % Storing the Joint Velocities
                q(d,i,an) = Vel(an,i);
                % Storing the Joint Torques
                to(d,i,an) = Tor(an,i);
            end

            % The determinant of the Jacobian
            deter(d,i) = det_Jac(i);
            deter(d+1,i) = det_Jac(i);

            % The Positions of the origines of the D-H frames
            X1(d,i)= p1(1,i);
            Y1(d,i)= p1(2,i);
            Z1(d,i)= p1(3,i);

            X2(d,i)= p2(1,i);
            Y2(d,i)= p2(2,i);
            Z2(d,i)= p2(3,i);

            X4(d,i)= p4(1,i);
            Y4(d,i)= p4(2,i);
            Z4(d,i)= p4(3,i);

            X6(d,i)= p6(1,i);
            Y6(d,i)= p6(2,i);
            Z6(d,i)= p6(3,i);

            X7(d,i)= p7(1,i);
            Y7(d,i)= p7(2,i);
            Z7(d,i)= p7(3,i);
        end
    end
end

% Considering the eight solutions
for k = 1:1:8
    nb(o,k) = 0;
    % Checking limit switches, joint velocity and joint torques
    for j = 1:1:d
        if out(j) == 0
            [outlimit(j,k),detcheck(j,k),velcheck(j,k),torcheck(j,k)] = checklimits(t(j,k,:),t(j-
1,k,:),q(j,k,:),to(j,k,:),deter(j,k),out(j-1),out(j-2),out(j+1),deter(j-1,k),deter(j-
2,k),deter(j+1,k),X7(j,k),Y7(j,k),Z7(j,k),X7(j-1,k),Y7(j-1,k),Z7(j-1,k),S,spec1,spec2,V,F,k,inc);
            nb(o,k) = 1;
        end
    end

    % Find the boundary points on the POSSIBLE lines using the above checks
    % Finding the boundaries of all possible segments along the line
    if nb(o,k) ~= 0
        [b1,b2,nb(o,k)] = linebounds(out,outlimit,detcheck,torcheck,velcheck,d,k);
    end
    if nb(o,k) ~= 0
```

```
            [posiseg(o),posi,seg1((posi+1):posiseg(o)),seg2((posi+1):posiseg(o))] =
collision(b1,b2,nb(o,k),t,k,X7,Y7,Z7,spec1,spec2,posiseg(o));
          end
        end
        posiseg(o)
        % Finding the optimal line
        Lopt(o) = 0;
        for y= 1:1:posiseg(o)      % for all the sub-segments
          % Storing ALL the lines for fitting a square
          % Storing the boundaries of those lines
          S1(o,y,1) = X7(seg1(y));
          S1(o,y,2) = Y7(seg1(y));
          S1(o,y,3) = Z7(seg1(y));
          S2(o,y,1) = X7(seg2(y));
          S2(o,y,2) = Y7(seg2(y));
          S2(o,y,3) = Z7(seg2(y));

          % Comparing the lines length
          if ((seg2(y)-seg1(y))*inc) > Lopt(o)
            Lopt(o) = (seg2(y)-seg1(y))*inc; % The max length
            % Storing the boundaries of the longest lines at every offset
            Xopt_st(o)= X7(seg1(y));
            Yopt_st(o)= Y7(seg1(y));
            Zopt_st(o)= Z7(seg1(y));
            Xopt_ed(o)= X7(seg2(y));
            Yopt_ed(o)= Y7(seg2(y));
            Zopt_ed(o)= Z7(seg2(y));
          end
        end
        Lopt(o)
      end

      % Ignoring the lines of zero length
      % at the begining and at the end
      for p =1:1:o
        if Lopt(p) ~= 0
          goodo1 = p;     % the index of the first line
          break
        end
      end

      for p = o:-1:1
        if Lopt(p) ~= 0
          goodo2 = p;     % the index of the last line
          break
        end
      end

      figure(layer)
      % The enveloppe of the longest lines
      switch plane
      case 1
        % The enveloppe of the longest lines
        plot(Xopt_st(goodo1:goodo2), Yopt_st(goodo1:goodo2), Xopt_ed(goodo1:goodo2), Yopt_ed(goodo1:goodo2))
      case 2
        % The enveloppe of the longest lines
        plot(Xopt_st(goode1:goode2), Zopt_st(goode1:goode2) - L1, Xopt_ed(goode1:goode2),
Zopt_ed(goode1:goode2) -L1)

    case 3
        % The enveloppe of the longest lines
        plot(Yopt_st(goode1:goode2), Zopt_st(goode1:goode2) - L1, Yopt_ed(goode1:goode2),
Zopt_ed(goode1:goode2) -L1)
    end
    axis([-40 60 -50 50])

    % fit a square
    switch layer
    case 1
        one1 =S1(goodo1:goodo2,:,:);
        one2 =S2(goodo1:goodo2,:,:);
        one3 =posiseg(goodo1:goodo2);
    case 2
        two1 =S1(goodo1:goodo2,:,:);
        two2 =S2(goodo1:goodo2,:,:);
        two3 =posiseg(goodo1:goodo2);
    case 3
        three1 =S1(goodo1:goodo2,:,:);
        three2 =S2(goodo1:goodo2,:,:);
        three3 =posiseg(goodo1:goodo2);
    case 4
        four1 =S1(goodo1:goodo2,:,:);
        four2 =S2(goodo1:goodo2,:,:);
        four3 =posiseg(goodo1:goodo2);
    end

    % The square side
    a = 10;
    figure(layer)
    hold on
    % Calling the function square
    switch plane
    case 1
        squarecenterloc = squareXY(a, yaw, offinc, inc, S1(goodo1:goodo2,:,:), S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
        % Plotting the possible squares in the longest lines enveloppe
        for i = 1:1:size(squarecenterloc,1)
            % Plotting each side at a time
            % Computing the 4 corners of every square
            plot([squarecenterloc(i,1)+(a/2)*sin(yaw)-(a/2)*cos(yaw),
squarecenterloc(i,1)+(a/2)*sin(yaw)+(a/2)*cos(yaw)],[(squarecenterloc(i,2)-(a/2)*cos(yaw)-(a/2)*sin(yaw)),
(squarecenterloc(i,2)-(a/2)*cos(yaw)+(a/2)*sin(yaw))]);
            plot([squarecenterloc(i,1)+(a/2)*sin(yaw)+(a/2)*cos(yaw), squarecenterloc(i,1)+(a/2)*cos(yaw)-
(a/2)*sin(yaw)],[(squarecenterloc(i,2)-(a/2)*cos(yaw)+(a/2)*sin(yaw)),
(squarecenterloc(i,2)+(a/2)*sin(yaw)+(a/2)*cos(yaw))]);
            plot([squarecenterloc(i,1)+(a/2)*cos(yaw)-(a/2)*sin(yaw), squarecenterloc(i,1)-(a/2)*sin(yaw)-
(a/2)*cos(yaw)],[(squarecenterloc(i,2)+(a/2)*sin(yaw)+(a/2)*cos(yaw)), (squarecenterloc(i,2)+(a/2)*cos(yaw)-
(a/2)*sin(yaw))]);
            plot([squarecenterloc(i,1)-(a/2)*sin(yaw)-(a/2)*cos(yaw), squarecenterloc(i,1)+(a/2)*sin(yaw)-
(a/2)*cos(yaw)],[(squarecenterloc(i,2)+(a/2)*cos(yaw)-(a/2)*sin(yaw)), (squarecenterloc(i,2)-(a/2)*cos(yaw)-
(a/2)*sin(yaw))]);
        end
```

```
   case 2
      squarecenterloc = squareXZ(a, pitch, offinc, inc, S1(goodo1:goodo2,:,:), S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
      % Plotting the possible squares in the longest lines enveloppe
      for i = 1:1:size(squarecenterloc,1)
         % Plotting each side at a time
         % Computing the 4 corners of every square
         plot([squarecenterloc(i,1)+(a/2)*sin(pitch)-(a/2)*cos(pitch),
squarecenterloc(i,1)+(a/2)*sin(pitch)+(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)-(a/2)*sin(pitch)),
(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch))]);
         plot([squarecenterloc(i,1)+(a/2)*sin(pitch)+(a/2)*cos(pitch), squarecenterloc(i,1)+(a/2)*cos(pitch)-
(a/2)*sin(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*sin(pitch)+(a/2)*cos(pitch))]);
         plot([squarecenterloc(i,1)+(a/2)*cos(pitch)-(a/2)*sin(pitch), squarecenterloc(i,1)-(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*sin(pitch)+(a/2)*cos(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
         plot([squarecenterloc(i,1)-(a/2)*sin(pitch)-(a/2)*cos(pitch), squarecenterloc(i,1)+(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*cos(pitch)-(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1 -
(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
      end

   case 3
      squarecenterloc = squareYZ(a, pitch, yaw, offinc, inc, S1(goodo1:goodo2,:,:), S2(goodo1:goodo2,:,:),
posiseg(goodo1:goodo2));
      % Plotting the possible squares in the longest lines enveloppe
      for i = 1:1:size(squarecenterloc,1)
         % Plotting each side at a time
         % Computing the 4 corners of every square
         plot([squarecenterloc(i,2)+(a/2)*sin(pitch)-(a/2)*cos(pitch),
squarecenterloc(i,2)+(a/2)*sin(pitch)+(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)-(a/2)*sin(pitch)),
(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch))]);
         plot([squarecenterloc(i,2)+(a/2)*sin(pitch)+(a/2)*cos(pitch), squarecenterloc(i,2)+(a/2)*cos(pitch)-
(a/2)*sin(pitch)],[(squarecenterloc(i,3)-L1 -(a/2)*cos(pitch)+(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*sin(pitch)+(a/2)*cos(pitch))]);
         plot([squarecenterloc(i,2)+(a/2)*cos(pitch)-(a/2)*sin(pitch), squarecenterloc(i,2)-(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*sin(pitch)+(a/2)*cos(pitch)), (squarecenterloc(i,3)-L1
+(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
         plot([squarecenterloc(i,2)-(a/2)*sin(pitch)-(a/2)*cos(pitch), squarecenterloc(i,2)+(a/2)*sin(pitch)-
(a/2)*cos(pitch)],[(squarecenterloc(i,3)-L1 +(a/2)*cos(pitch)-(a/2)*sin(pitch)), (squarecenterloc(i,3)-L1 -
(a/2)*cos(pitch)-(a/2)*sin(pitch))]);
      end
   end
   axis([-40 60 -50 50])

   % Storing the data returned from the squareXY function
   switch layer
   case 1
      circleone = squarecenterloc ;          % Storing the location of the square centers
   case 2
      circletwo = squarecenterloc ;          % Storing the location of the square centers
   case 3
      circlethree = squarecenterloc ;        % Storing the location of the square centers
   case 4
      circlefour = squarecenterloc ;         % Storing the location of the square centers
   end
```

```
end

% The tolerance in matching the squares
tol = 1;

switch plane
case 1
    coupon = match(circleone(:,1:2),circletwo(:,1:2),circlethree(:,1:2),circlefour(:,1:2),tol);
case 2
    coupon = match(circleone(:,[1,3]),circletwo(:,[1,3]),circlethree(:,[1,3]),circlefour(:,[1,3]),tol);
case 3
    coupon = match(circleone(:,2:3),circletwo(:,2:3),circlethree(:,2:3),circlefour(:,2:3),tol);
end

% If matching occured with the three layers, the circle is drawn
figure(layer+1)
for i = 1:1:size(circleone,1)
    if coupon(i) == 4
        switch plane
        case 1
            C1 = circleone(i,1)
            D1 = circleone(i,2)
        case 2
            C1 = circleone(i,1)
            D1 = circleone(i,3)
        case 3
            C1 = circleone(i,2)
            D1 = circleone(i,3)
        end

        % For all angles on the circle
        for teta = 1:1:360
            Ccircle(teta) = C1+(a/2)*cos(teta*pi/180);
            Dcircle(teta) = D1+(a/2)*sin(teta*pi/180);
        end
        % Plot the circle
        plot(Ccircle, Dcircle);
        hold on
    end
end
figure(layer+1)
hold on
plot([12,42.5],[-18.5,-18.5]);
plot([42.5,42.5],[-18.5,18]);
plot([12,42.5],[18,18]);
plot([12,12],[-18.5,18]);
% Specifying the axis
axis([-40 60 -50 50])
```

E.12 FUNCTION TO MATCH THE POSITION OF SQUARE LAYERS TO FORM THE COUPONS

```
function [coupon] = match(circleone,circletwo,circlethree,circlefour,tol)
% MATCH returns an array that has the matching number of the squares
% The matching number 4 indicates matching of the 4 layers at specific points

for i = 1:1:size(circleone,1)
   C1 = circleone(i,1);
   D1 = circleone(i,2);
   % Array to display the possible matches
   coupon(i) = 1;
   % Variable used to check any possible matches in layer 2
   candidate2 = 0;
   % For all squares on the second layer
   for j = 1:1:size(circletwo,1)
      C2 = circletwo(j,1);
      D2 = circletwo(j,2);
      % Comparing the locations of the square centers
      if abs(C2-C1) <= tol & abs(D2-D1)<=tol
         coupon(i) = 2;
         candidate2 = 1;
         break
      end
   end
   % If no matching occured, the rest of the loop is ignored
   if candidate2 == 0
      continue
   end

   % Variable used to check any possible matches in layer 3
   candidate3 = 0;
   % For all squares on the third layer
   for j = 1:1:size(circlethree,1)
      C3 = circlethree(j,1);
      D3 = circlethree(j,2);
      % Comparing the locations of the square centers
      if abs(C3-C1) <= tol & abs(D3-D1)<=tol
         candidate3 = 1;
         coupon(i) = 3;
         break
      end
   end
   % If no matching occured, the rest of the loop is ignored
   if candidate3 == 0
      continue
   end

   % Variable used to check any possible matches in layer 4
   candidate4 =0;
   % For all squares on the fourth layer
   for j = 1:1:size(circlefour,1)
      C4 = circlefour(j,1);
      D4 = circlefour(j,2);
      % Comparing the locations of the square centers
```

```
        if abs(C4-C1) <= tol & abs(D4-D1)<=tol
            candidate4 = 1;
            coupon(i) = 4;
            break
        end
    end
end
```

VITA

Serge Riad Moutran was born on May 28, 1978, in the beautiful city of Beirut, Lebanon. He attended Champville high school in Lebanon, where he received his French and Lebanese Baccalaureates, both with distinction in 1996. He enrolled in the American University of Beirut where he was awarded in 'recognition of outstanding academic achievement.' In the summer of 1999, he worked as an engineering co-op in Schlumberger, France. He received, in June 2000, a Bachelor degree in Mechanical Engineering with distinction. He subsequently enrolled in Virginia Polytechnic Institute and State University to pursue a Masters degree in Mechanical Engineering. His graduate coursework and research focused on mechatronics and robotics.