

Six-Axis Robot Configuration Singularities

Use of the V+ MV.SL_MOVE Routine and the SPEED.LIMIT Parameter

March 30, 2007



Adept Technology, Inc., Copyright Notice

The information contained herein is the property of Adept Technology, Inc., and shall not be reproduced in whole or in part without prior written approval of Adept Technology, Inc. The information herein is subject to change without notice and should not be construed as a commitment by Adept Technology, Inc. The documentation is periodically reviewed and revised.

Adept Technology, Inc., assumes no responsibility for any errors or omissions in the documentation. Critical evaluation of the documentation by the user is welcomed. Your comments assist us in preparation of future documentation. Please submit your comments to: techpubs@adept.com.

Copyright © 2007 by Adept Technology, Inc. All rights reserved.

Adept, the Adept logo, the Adept Technology logo, AdeptVision, AIM, Blox, Bloxview, FireBlox, Fireview, HexSight, Meta Controls, MetaControls, Metawire, Soft Machines, and Visual Machines are registered trademarks of Adept Technology, Inc. Brain on Board is a registered trademark of Adept Technology, Inc. in Germany.

ACE, ActiveV, Adept 1060 / 1060+, Adept 1850 / 1850 XP, Adept 540 Adept 560, Adept AnyFeeder, Adept Award, Adept C40, Adept C60, Adept CC, Adept Cobra 350, Adept Cobra 350 CR/ESD, Adept Cobra 550, Adept 550 CleanRoom, Adept Cobra 600, Adept Cobra 800, Adept Cobra i600, Adept Cobra i800, Adept Cobra PLC server, Adept Cobra PLC800, Adept Cobra s600, Adept Cobra s800, Adept Cobra s800 Inverted, Adept Cobra Smart600, Adept Cobra Smart800, Adept DeskTop, Adept FFE, Adept FlexFeeder 250, Adept IC, Adept iSight, Adept Impulse Feeder, Adept LineVision, Adept MB-10 ServoKit, Adept MC, Adept MotionBlox-10, Adept MotionBlox-40L, Adept MotionBlox-40R, Adept MV Adept MV-10, Adept MV-19, Adept MV4, Adept MV-5, Adept MV-8, Adept OC, Adept Python, Adept sDIO, Adept SmartAmp, Adept SmartAxis, Adept SmartController CS, Adept SmartController CX, Adept SmartModule, Adept SmartMotion, Adept SmartServo, Adept sMI6, Adept sSight, Adept Viper s650, Adept Viper s850, Adept Viper s1300, Adept Viper s1700, AdeptCartesian, AdeptCast, AdeptForce, AdeptFTP, AdeptGEM, AdeptModules, AdeptMotion, AdeptMotion Servo, AdeptMotion VME, AdeptNet, AdeptNFS, AdeptOne, AdeptOne-MV, AdeptOne-XL, AdeptRAPID, AdeptSight, AdeptSix, AdeptSix 300, AdeptSix 300 CL, AdeptSix 300 CR, AdeptSix 600, AdeptTCP/IP, AdeptThree, AdeptThree-MV, AdeptThree-XL, AdeptTwo, AdeptVision, AVI AdeptVision, AGS AdeptVision GV, AdeptVision I, AdeptVision II, AdeptVision VME, AdeptVision VXL, AdeptVision XGS, AdeptVision XGS II, AdeptWindows, AdeptWindows Controller, AdeptWindows DDE, AdeptWindows Offline Editor, AdeptWindows PC, AIM Command Server, AIM Dispense, AIM PCB, AIM VisionWare, A-Series, FlexFeedWare, HyperDrive, IO Blox, IO Blox, 88, MicroV+, MotionBlox, MotionWare, ObjectFinder, ObjectFinder 2000, PackOne, PalletWare, sAVI, S-Series, UltraOne, V, V+ and VisionTeach are trademarks of Adept Technology, Inc.

Any trademarks from other companies used in this publication are the property of those respective companies.

Created in the United States of America

Table of Contents

1	Introduction	4
2	Understanding Robot Configuration Singularities	4
	Types of Singularities	4
	Wrist Singularity	4
	Alignment Singularity	5
	Elbow Singularity	5
3	Six-Axis Robot Configurations	5
	RIGHTY versus LEFTY	6
	ABOVE versus BELOW	7
	FLIP versus NOFLIP	8
	CONFIG Function	8
4	Dealing with Robot Configuration Singularities	9
	Define the areas where the robot is moving close to singularities	9
	Change the motion from straight-line to joint-interpolated	9
	Use the V+ Routine MV.SL_MOVE	9
	Use the SPEED.LIMIT V+ Parameter	11

1 Introduction

This document describes configuration singularities experienced by six-axis robots during straight-line motion.

The V+ MV.SL_MOVE routine and the V+ SPEED.LIMIT parameter are discussed, as well as other measures for avoiding configuration singularities.

2 Understanding Robot Configuration Singularities

A configuration singularity can be defined as a location in the robot workspace where two or more joints no longer independently control the position and orientation of the tool. As a robot executes a straight-line motion that moves close to a configuration singularity, the robot joint speeds necessary to achieve that motion become excessive.

Types of Configuration Singularities

The types of configuration singularities that can be experienced by a robot depend on the physical relationships between the robot joints. The following sections describe the most common types of configuration singularities that can occur.

Wrist Singularity

Wrist Singularity occurs when the axes of Joints 4 and 6 are aligned.

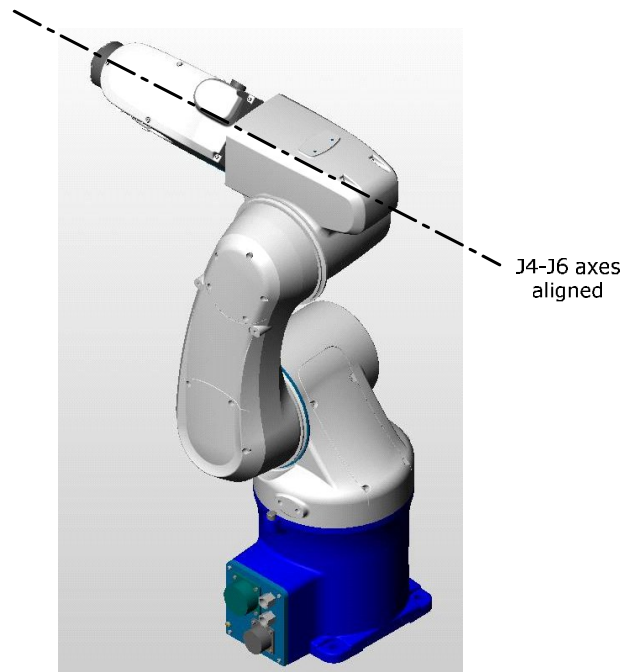


Figure 1. Wrist Singularity

Alignment Singularity

Alignment Singularity occurs when Joint 6 (wrist) and Joint 1 axes are aligned. (This has not yet occurred in the following figure, but is about to.)

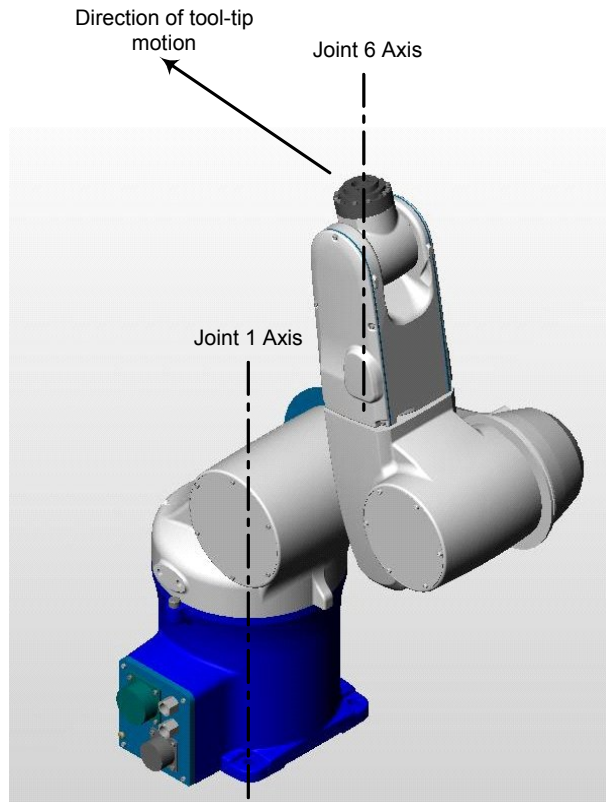


Figure 2. Alignment Singularity

Elbow Singularity

Elbow Singularity (not illustrated here) occurs when the arm is fully extended. In this case, as the elbow joint becomes further extended, higher joint speeds are required to maintain constant Cartesian speed. The robot cannot extend beyond its reach.

This document does not deal with this type of singularity, because it is not considered internal to the robot workspace.

3 Six-Axis Robot Configurations

A six-axis robot can move to a location defined by a Cartesian (World) transformation with different joint positions and, consequently, different robot configurations. If you want the robot to move to a location in space with a specific configuration you must use a precision point to define that point and orientation.

- Precision points specify the joint values for all the joints of the robot, so the configuration of the robot is specified explicitly. (Note the exception that follows.)
- Cartesian coordinates and orientation angles specify the tool position and orientation, but the configuration is not specified.

A robot tool-tip can move in a straight line to a location defined by a Cartesian transformation or precision point. This is achieved, in V+, by using **APPROS**, **DEPARTS**, and **MOVES** instructions. It is important to note that the robot configuration can **not** change during straight-line motions.

NOTE: These three commands calculate the position specified by a precision point, and convert that into a transformation. The configuration will remain as it was before the command, even if the joint positions specify otherwise.

Below are the different robot configurations for the Adept Viper s650 robot. Note that, for all the of configurations shown, the position and orientation of the robot tool-tip is represented by the same Cartesian transformation. That is, the robot tool can be moved to this location and orientation with all the different configurations shown.

RIGHTY versus LEFTY

The following figures illustrate RIGHTY versus LEFTY configurations of a Viper s650.

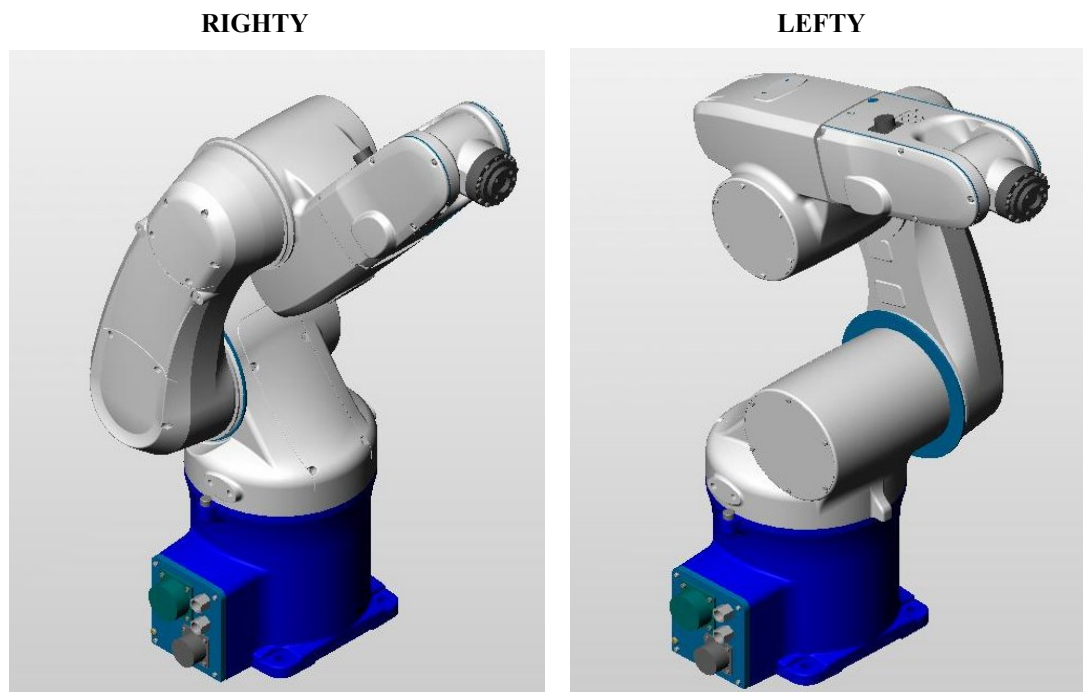


Figure 3. RIGHTY/LEFTY Configurations

The following V+ code snippet demonstrates the use of the RIGHTY and LEFTY program instructions:

```
RIGHTY      ; Request change in robot configuration during next motion
MOVE loc_a  ; Move to loc_a transformation in RIGHTY configuration

LEFTY       ; Request change in robot configuration during next motion
MOVE loc_a  ; Move to loc_a transformation in LEFTY configuration
```

NOTE: The RIGHTY and LEFTY program instructions can include the keyword ALWAYS, which causes the instruction to continue until it is explicitly disabled. For example:

```
RIGHTY ALWAYS ; Request change in robot config. during subsequent motions
MOVE loc_a    ; Move to loc_a transformation in RIGHTY configuration
```

ABOVE versus BELOW

The following figures illustrate ABOVE versus BELOW configurations of a Viper s650.

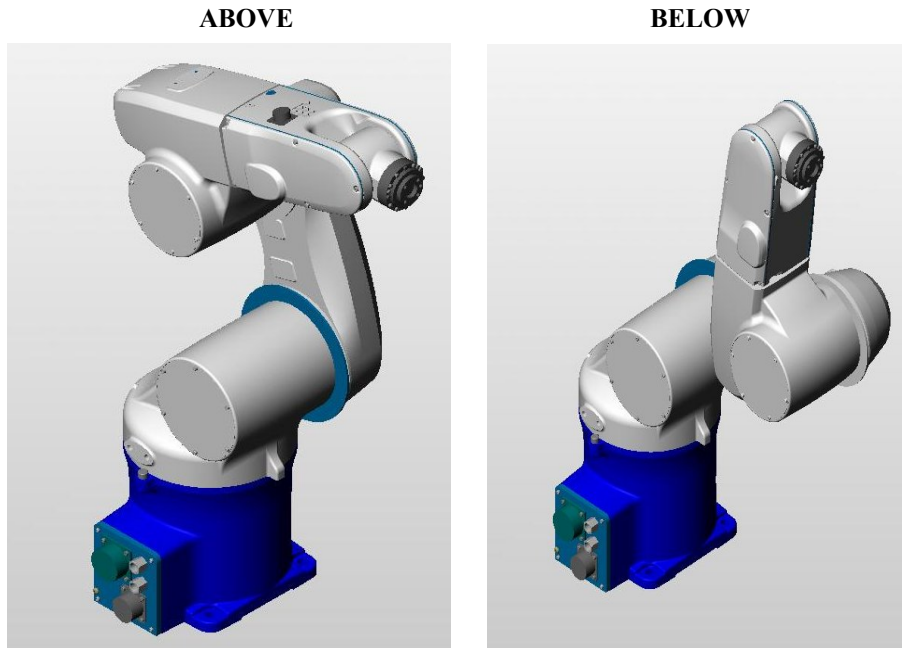


Figure 4. ABOVE/BELOW Configurations

The following V+ code snippet demonstrates the use of the ABOVE and BELOW program instructions:

```
ABOVE          ;Request change in robot configuration during next motion
MOVE loc_a    ;Move to loc_a transformation with ABOVE configuration

BELOW         ;Request change in robot configuration during next motion
MOVE loc_a    ;Move to loc_a transformation with BELOW configuration
```

As with the LEFTY and RIGHTY program instructions, the ALWAYS keyword can be used with ABOVE and BELOW.

FLIP versus NOFLIP

The following figures illustrate FLIP versus NOFLIP configurations of a Viper s650.

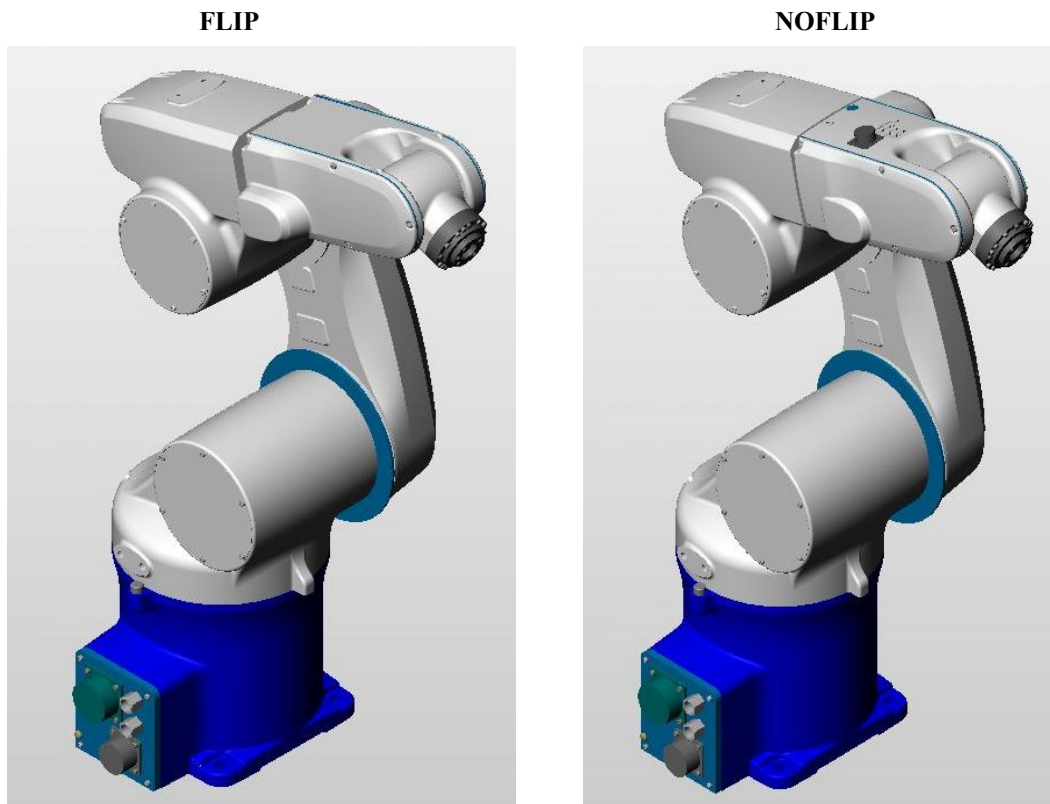


Figure 5. FLIP/NOFLIP Configurations

The following V+ code snippet demonstrates the use of the FLIP and NOFLIP program instructions:

```
FLIP           ;Request change in robot configuration during next motion
MOVE loc_a    ;Move to loc_a transformation with FLIP configuration

NOFLIP        ;Request change in robot configuration during next motion
MOVE loc_a    ;Move to loc_a transformation with NOFLIP configuration
```

The ALWAYS keyword can be used with FLIP and NOFLIP.

CONFIG Function

To learn the robot's configuration, you can use the V+ CONFIG real-value function:

- CONFIG(0) – The returned value represents the current (instantaneous) configuration of the robot.
- CONFIG(1) – The returned value represents the configuration the robot will achieve at the end of the current motion.
- CONFIG(2) – The returned value represents configuration the robot will achieve at the end of the next motion.

Examples:

CONFIG(0) = 0 indicates that the robot has the following configuration:

LEFTY, ABOVE, NOFLIP

CONFIG(0) = 4 (only bit #3 set) indicates that the robot has the configuration:

LEFTY, ABOVE, FLIP

Bit #	Bit Mask	Indication if bit SET
1	1	Robot has RIGHTY configuration
2	2	Robot has BELOW configuration
3	4	Robot has FLIP configuration

4 Dealing with Robot Configuration Singularities

This section describes several methods for avoiding robot configuration singularities.

Define the areas where the robot is moving close to configuration singularities

The first step in dealing with configuration singularities is to characterize where in the robot work-envelope the singularities exist. These configuration singularities are described in the section “Understanding Robot Configuration Singularities”. Try to design your application to avoid these areas of singularity.

Change the motion from straight-line to joint-interpolated

Straight-line motion is achieved with the V+ language through the APPROX, DEPARTS, and MOVES instructions. During motion driven by these instructions no changes in configuration are allowed, and the robot joints may require excessive speeds to achieve the requested Cartesian motion.

To avoid this situation:

- Use the MOVE instruction instead of the MOVES instruction, (since the robot can change configuration during a MOVE instruction to avoid a singularity). See the section “Use the V+ Routine MV.SL_MOVE” for information on using this routine.
- If you must use straight-line motion near robot configuration singularities, set the SPEED.LIMIT parameter so that the robot will stop when moving close to a configuration singularity, instead of attempting to move certain joints at excessive speeds in order to maintain the straight-line motion. See the section “Use the SPEED.LIMIT V+ Parameter” for information on setting this parameter.

Use the V+ Routine MV.SL_MOVE

The V+ routine MV.SL_MOVE, which is available in the SLMOTION utility, is provided as an alternative to the MOVES instruction, for use when the robot might move close to a configuration singularity.

The MV.SL_MOVE routine divides a transformation vector into smaller vectors. It then takes each small vector (transformation) and, using the V+ program instruction SOLVE.ANGLES, computes the robot joint positions for this transformation. After performing the relevant checks, the robot is moved, in joint-interpolated mode, as defined by the calculated joint positions. In other words, the program uses many MOVE instructions to move the robot by small steps along the desired straight-line path.

It is recommended that the MV.SL_MOVE routine be used when the robot is moving close to a Wrist Singularity (see Wrist Singularity), because during this motion, Joint 4 speed is controlled and there is little risk of other robot joints moving unexpectedly to maintain straight-line tool-tip motion.

If the MV.SL_MOVE routine is used when the robot is moving close to an Alignment Singularity (see the section “Alignment Singularity”), Joint 1 could rotate by 180° in order to maintain straight-line motion. This could damage cell tooling or peripheral equipment in the cell.

For more details, see the description of the [SLMOTION](#) utility in the [Instructions for Adept Utility Programs](#).

A Note on Conveyor Tracking

Conveyor tracking requires straight-line motions, but the MV.SL_MOVE routine is generally too slow to keep up with conveyor parts. Therefore, the program does not support use with conveyor tracking. The cell-designer must be aware of the possibility of configuration singularities as the robot tracks a moving belt, because the robot motion is effectively governed by the belt motion.

The following cell-design guidelines can prevent the robot from moving close to configuration singularities while conveyor tracking:

- Design the end-effector so that, while tracking parts on the conveyor, Joint 5 stays well away from zero, for example between 45° and 90°. This will prevent Joint 4 – 6 wrist singularities. You can use the V+ ALIGN program instruction to align Joint 6 with the belt surface.
- Design the cell so that the Joint 1 and Joint 6 axes will never align, thus preventing alignment singularities.
- Define wait locations with precision-points, and use the V+ MOVE instruction to go to the wait locations. This will ensure consistent robot configurations.
- Choose the conveyor belt window so that the robot cannot fully extend itself.

Figure 6 shows a wrist singularity (described in section 2), which occurs as the robot tracks a part on the conveyor belt. The robot has an end-effector that is offset from the robot flange by a distance of T_z in the tool Z direction, and T_x in the tool X direction.

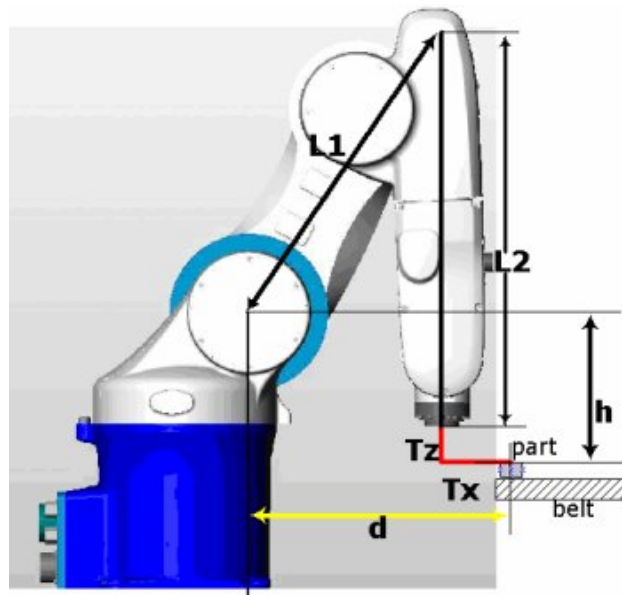


Figure 6. Wrist Singularity During Conveyor Tracking

This type of singularity can be avoided by adjusting the belt height (h) . To find the optimum belt height:

1. Move the arm so that Joints 4 and 6 are aligned (as shown in Figure 6).
2. Note the distance (d) where the robot will access the part on the near edge of the conveyor.
3. Calculate the minimum belt height (h) using the following equations:

$$L1^2 = (d - Tx)^2 + (L2 + Tz - h)^2$$
$$\therefore h = (L2 + Tz) - \sqrt{L1^2 - (d - Tx)^2}$$

Use the SPEED.LIMIT V+ Parameter

A system parameter is provided in the V+ system (version 16.3D3 and later) to offer more functionality in controlling robot behavior as the robot moves close to a configuration singularity.

For a one-robot system, the parameter can be set with an instruction or Monitor command in this form:

```
PARAMETER SPEED.LIMIT = value
```

For a system with multiple robots, the desired robot must be specified as follows:

```
PARAMETER SPEED.LIMIT[robot number] = value
```

NOTE: If the robot number is omitted or zero in a PARAMETER command or instruction, the settings for all robots will be altered.

In either case, the value specifies the maximum allowable joint speed as a percentage of the maximum joint speed of the robot.

Use of this parameter requires care, as it may introduce undesired limits on other existing robot motions. When the SPEED.LIMIT parameter has not been set, a constant speed is enforced on the tool-tip motion for every trajectory setpoint along straight-line motions, and additionally a speed limit is enforced on the joint motions on an **average** basis for the straight-line motion. This permits joints to experience transitory periods of exceeding the joint speed limits.

However, after the SPEED.LIMIT parameter is set, the joint speeds are limited to the specified percentage of their maximum values on every setpoint of every subsequent straight-line trajectory, instead of on an average over the straight-line motion. As a result, this might restrict some straight-line motions that were previously achievable.

This functionality is a secondary check on each setpoint of the V+ trajectory before it is sent to the servos. A trajectory will be started and tracked until the SPEED.LIMIT value is exceeded – when the setpoint is frozen (causing an abrupt stop in motion) and a V+ error is displayed as: “*Maximum setpoint speed or acceleration exceeded*”. Note that there is no way to “pre-trap” this error, since it is declared as the setpoints are sent to the servos. It is possible, however, to use an error reaction routine to catch the error and determine how the system responds to the error.

The default boot-up value of this parameter is zero (checking disabled) to avoid limiting robot motions unnecessarily. Therefore the SPEED.LIMIT parameter must be explicitly set in user programs to be active.

The current setting of the SPEED.LIMIT parameter can be determined with these Monitor commands:

```
PARAMETER SPEED.LIMIT
```

```
PARAMETER SPEED.LIMIT[robot_number]
```

(When the PARAMETER command is entered with no arguments, the SPEED.LIMIT parameter is not listed in the output. This particular parameter must be explicitly specified in order to see its setting.)

The current setting can be determined within a program with an instruction in one of these forms:

```
variable = PARAMETER(SPEED.LIMIT)
```

```
variable = PARAMETER(SPEED.LIMIT[robot_number])
```