Contents

[Table of Figures 2](#_Toc497384283)

[Blender Assembled Fetch 3](#_Toc497384284)

[DH Dimensions 3](#_Toc497384285)

[Dh Parameters 4](#_Toc497384286)

[Kinematic List 4](#_Toc497384287)

[Fetch Model Generation 6](#_Toc497384288)

[Task 1 – Visual Servoing 8](#_Toc497384289)

[Simulated Camera on Gripper 8](#_Toc497384290)

[Projection View (Initial View) 8](#_Toc497384291)

[Visual Servoing Results 9](#_Toc497384292)

[Task 2, Dynamic torque 10](#_Toc497384293)

[2kg Load 10](#_Toc497384294)

[Motion 1 10](#_Toc497384295)

[Motion 2 10](#_Toc497384296)

[Motion 3 11](#_Toc497384297)

[Motion 4 11](#_Toc497384298)

[5kg Load 12](#_Toc497384299)

[Motion 1 12](#_Toc497384300)

[Motion 2 (Joint Failure here) 12](#_Toc497384301)

[Process 13](#_Toc497384302)

[“Robots are increasingly impacting industry, our job and our daily lives in both positive and negative ways” 15](#_Toc497384303)

[“how my job may require me to create/integrate/install robots which may result in other people no longer being required to work due to no fault of their own” 15](#_Toc497384304)

[Bibliography 17](#_Toc497384305)

[Appendix 18](#_Toc497384306)

[TorqueFun Function 18](#_Toc497384307)

[Plot Torques Function 20](#_Toc497384308)

[Dynamic Torque Function 21](#_Toc497384309)

[Draw Fetch Function 24](#_Toc497384310)

[PerfromVS function 26](#_Toc497384311)

[Calculate ‘W’ Function 31](#_Toc497384312)

[RunMe Script 32](#_Toc497384313)

[GitHub Clone Link 32](#_Toc497384314)

# Table of Figures

Figure 1:Fetch in Initial Pose with Simulated Camera 8

Figure 2: Camera Projected View, only Centre 8

Figure 3: Camera Projected View with the extremes of the circle labelled 9

Figure 4: Visual Servoing, Just before Completion 9

Figure 5: 2kg Load Motion 1 10

Figure 6: 2kg Load Motion 2 10

Figure 7: 2kg Load Motion 3 11

Figure 8: 2kg Load Motion 4 11

Figure 9: 5kg Load Motion 1 12

Figure 10: 5kg Load Motion 2 12

# Blender Assembled Fetch

![A picture containing floor

Description generated with very high confidence](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RCaRXhpZgAATU0AKgAAAAgABAE7AAIAAAANAAAISodpAAQAAAABAAAIWJydAAEAAAAaAAAQeOocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAERlbmlzICBEcmFjYQAAAAHqHAAHAAAIDAAACGoAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEQAZQBuAGkAcwAgACAARAByAGEAYwBhAAAA/+ELH2h0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTctMTAtMjRUMTM6MTI6NTguOTk3PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkRlbmlzICBEcmFjYTwvcmRmOmxpPjwvcmRmOlNlcT4NCgkJCTwvZGM6Y3JlYXRvcj48L3JkZjpEZXNjcmlwdGlvbj48L3JkZjpSREY+PC94OnhtcG1ldGE+DQogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgIDw/eHBhY2tldCBlbmQ9J3cnPz7/2wBDAAcFBQYFBAcGBQYIBwcIChELCgkJChUPEAwRGBUaGRgVGBcbHichGx0lHRcYIi4iJSgpKywrGiAvMy8qMicqKyr/2wBDAQcICAoJChQLCxQqHBgcKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKir/wAARCAKaAmwDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwCpRTzF6GmlGHatDMSikzRmmMWikzQAT05oAWgAnpUix/3vyp4GOlIQxYv735U8ADpS0UAFFFFIQUUUUDCiiigAooooAKKKKACiiigQUUUUDGlFPamGH0P51LSUxkJjYe/0pvOcY5qVpQPu81GXJPJoAesZPXipAoXoKhEjDv8AnThN6j8qBEtFMEinvj604HPSgBaKKKQBRRRQAUUUUDCiiigAooooEFFFFMAooooGFFNZwv1qJnLewpgSNIB7mo2Yt1pKSgANJRRQAtKHYd6bRQBKJj3FOEqn2qCigCyCD0OaWqtODsOhoAsUVCJT3GaeJVPXigB9FICD0OaWkAUUUUAFFFFIAopCwXqaiaUn7vFMCRnC9evpUTSFvYU2koAXNLTaKYDqKSjNABRSUUAFFFFAFqiikLBepqRAQD1GaYyIOvFNaU/w8UwnPWmMVdufmzUysv8ACRVeigC1RVYMR0NPEpHXmgCaioxKp68U8MD0NAhaKKKQBRRRQMKKKKACiiigAooooAKKKKACkpjSAdOTUbMW60wJGlA6c1GzFutIAScAZNWI7C4k/g2j1bitIUp1HaCuY1a1Kir1JJepWorSTSsf66UD2FMkOk2pK3F1EGHUNKAfyrsWX17XlaPqzzpZxhb2heT8kUKKtjVNEddqyxkeynP8qkRtKnXCTKpPq2D+tH1K7tGpF/MP7T5VepRmvkUKXp0rRbS1YAwzZz0zzmq8mn3Ef8IYeqms6mCxENXH7tTalmeEq6KaT89PzIBIw75+tOE3qPyqNlKnDAg+hFJXJa256CaauicSKe+PrTuvSq1KDjpSGWaKgEjDvn604Teo/KgCWimCRT3/ADp3XpQAtFFFIAoprMF61G0pPTgUwJGcL1qJpCenAptJQAUUUUwFpKKKACiiigAooooAKKKKACiiigAooooAKcHYdDTaKAJBKe4zThKvfioaesZP3uBSAl3rjqKY0v8Ad/OnhQo4FIVU9RQBCTnrSVKYh2OKaY2HTmmAyilII6ikoAKKKKACiiigAooooAM0ZpKKAJWlJ+6MUzk9aWikAlFLRQA2ilNJQIKKKKACiiigBwdh0NOEp7io6KAJxKp9qcCD0OarUtAFmiq4dh3/ADp4m9R+VAEtFMEinvj604HPSgBaKbuycICx9BU8enzzcyERr6GtadCpVdoK5z18VRw6vVkkV2kA6c01VlmOEUt9BV9orGyGZm3t6HnP4VBNraoMW8QAHdq7PqlOl/Hml5LVnnrMK1f/AHWk2u70QselytzIwQfnUxtbK1XdcyA/7zYrFutYuZgQJCB/s8Cs9nZzlmJPvR9YwtL+HTv5v/IPqeOr/wAaryrtFfqdFJrtlbrttYjJ9BgVmXevajMCLaSO3HY7NzD8zj9KzqKznmGIkrJ2Xlob08owkHdx5n3ev/AM+8ttSvM+ffvMD/C7HH5dKzpNOuouqEj1Xmuhorhk3J3bPTjGMFaKsjlCJEPOQael5PEfldh+NdM8SSDDorfUVVl0q2k6KUP+yaRRStfEN3bkbXP4EjNbdp42mTAl+Yf7S5/lWJLojf8ALKQH6jFQQaVcyzFI0HynDPngfjW9GrWi7U2/kcuIoYacb1oq3md3b+KrO5UCeMHPoQf0NXkbTLxQY3CEjpnB/KuPtNIgtsNMfOkHYj5R+Her5Yn2Fe7RVaa/2hJ/LU+WxH1elL/ZHKPo9PuZ0MmlHGYZA3saoyRvE22RSD7iqEF3PbMDDIy47Z4/KtaDXkZQt3Fk9yo60qmAoVPgfK/vQ6Oa4qjpUXOvuZVorTjFhf8AywELJjOAMHH0rPkQxSsh6qcV5WJwk8PZvVPqj3sHmFLF3UU010YylBx0oAJ6VIsX978q4z0BFdz05qQ7ivGAaUAAcUtAyAo3cZptWaQgHqM0AVqKmManpxTTF6GgRHRTijDtSUAJRRRQAUUUUAFFFFACUUtFIAooopgFFFOVS3SgBtOVC3sKkVAPc06gBFQL06+tLRRQAUUUUAFFFFABSFVPUUtFADDEOxIppibtzUtFAEBVh1BpKsUFQeooArUVOYwenFNMR7c0ARUU4qR1FJQMXNGaSikAtLTaWgBaSjNFAhKKKKACiiigYUUUUAFFFFAgop6xlvYVKkYBGBk0wI0iZu1X4NOwu6Y7R3zTy0OnQiSbDSHoO9ZF1fzXTfO2F7KOlekqVHDK9fWX8v8AmeL7fEY2TjhfdgvtPr6I05b+0sgUtkDv7f41nXGp3E+cvsX0XiqbFgPlXJ+tV3WRvvfzrCrjatRcq91dlodeHyzD0XzyXNLu9WSPcDJx8x9TUDOzn5jQVI64pMVxHpiUUtJQAUUUUAFFFJuy2xAXfGdq9f8A61NJt2QpSUVdvQWmeYDJ5cYMj9wvb6ntUn2fcM3EnH/POI9fqev5Y/GpAdqBIkEaDoFGK9KjgJy1qaI8fEZrCGlJXf4DPIA/4+WDf9M0zj/E/wCeKlDfKFVQijgKB0ptFevTowpK0EeBWr1KzvN3Foox8uSQq+prOu9bggBW1HmyD+M/dH+NOU4w3M4U5Tfumi2ETfIQierHFZd3rscQK2a72/56N0H0FYl3qEk7F55C38hVAXKzf6t1cf7JzXHUxLekT0aWDS1lqdL4c1KZ/Fdo0sjSGR9hyexFdxfQ4vXPY84rzfwxn/hKLDcCD5w4Nem3/wDx9H6Csa3vYRt/zfob4dcuYJLrD9SsAAOBilpBS15B9AJRmiigAzRmkooAKKKKQgoIB6jNFFADTGp9qaYj2NSUUDISjDtTasUEA9RQBXoqYxqenFNMR7EGgZHRSlWHUUgyenNMApQCelPWP+9+VSAY6UhDFjA680+iigAooooAKKKKACiiigAooooAKKKKACiiigAooooGFFFFABQQD1A/KgsFHNRmbngUwI6KlMI7GmmNh2z9KQDKKUjHWkoAKKKKACiiigAooooAKKeqFvYVIsYX3NAEaxlvYVIqBfc1NHBJKfkXPvWpYaJLdSBYommf0UcCuujhKtZcyVl3ex5+JzChh3yt3l2WrMuOCSU/IufetbTtFluJ1WJGlfPYcLXX6b4NVAG1B/8AtnH0/P8AKumgtobWPy7eJY19FFdHtMLhvg9+XfocnssdjV+8/dwfT7T/AMj561XVfO1GTghIzsUfSqTago7moL3i8mBOfnbk9+aoS9686pOVSTnLdntUqUaNNU4bIvyaiOzGoftyn+I1mvTKg0NZbwHo361MtznqQaw8kdDT1lZehoA3ROp68U8Mp6GsRbth1qdLwHrQBq013Ea7iCfYDNU0uh2ap0uMnnmmBCb6Ld+/dlUHlIxyfq3+H51aj1G0MeyNljX+6RinNaQ3S/OoPuOorH1DTZLPMiZeL19K9bC16UfdirM8DHYbES96Urr8vkbqsGX5W3e+aWuWildOY3I+hq7Bqk8TAuRIo6hq9RVEzxXSaN0LnngAdSTgCqN3q9va5WL99KOP9kVi3urT3Jw77V7IvAFZN1eCCFnboOgHeuSpirLQ7aODu/e1NK81Se6P76QkdlHAFUJJuCScAda5u61O7mmCxkqD0VR1NdT9jit4yl9LufGPLiOT+J7VwRquq3Y9OVBUUrnNXM1zehlG4Ieij0rT0zT5dPtFlm+WSZtyg9Qo4BI7c5q5ALazUG2iLTY5klw2D/sjoP1pskrSOWdizHqSazhTcXzSeptUqKS5ILQ2PDJkufF1ix+ZzLuPboCT+gr0m9Ia6b2wK4r4eWEk2ryXzAiGFCgPPLH/AAHX6iuwnffO7DoTxXVXfLhEu7/JHDhUpY+TX2Y2+bYylpKK8k94WkzRSUALRSUUCFopKKAFopKKAFopKKACiiigYUU1pAvuaiZy3Xp6UwJGlA+7zUZdic5ptFADxKw6808SjuMVDRQBYDqehparUoYjoaALFFRCVh15pwlHcYoAfRSBlPQ0uaACijNFABRRRSAKKKKACiiigAooprSBfc0wHVG0v9386YzFutNoAUkk80lFFMC3SUtIakBDz1ppjU9sU6igCMxehppRh2qaigCvRUzso+9yaiDANnaMelMBVQt06etSrGF9zSCYHrxTgwPQ0AOrQ0/TJLqVVEbO7nCoB1qCxt/OmyRkKfzNeqeHtBGmWfmzIPtUg5z1Qeld9GEKNP29VX7Lv5+h5GJqVcRW+q0Ha3xS7eS82Zem+DVQBtQfP/TOPp+f5V0tvbQ2sYjt4ljX0UVKQR1FJmuWviqtd++9O3Q7sNgaGFX7uOvfr94tFJmjNcx2HzVr0TW/iDUInBBS5kHIx/EayJe9elfFTwu9nqJ1y1TNtdHE4Uf6uT1Ps38/rXmslMCs9MpztTM0wFopM0ZoAWiiigBQSOhp6TunQ8elR0UgNqwvcyZGM+h71rO0U0B3DKMMMDXIpI0bZU4rVhu22fMcgjB56GmBmXlv9lumVM+WeVz6VED8pqzfXPn4GBhT1qqOle3hqntIXe581jKKpVbLZmcfmfBOMnrUd7Y+axt/NjJVs7iflp7feNNJrmkk00zrg3Fpofb29rYOkkQE9whyshGFQ+oHcj1PtxTC2SSep70mc1u6R4Q1PViHEfkQ/wDPSUY9e31FEIOT5aaFVqRguerK3qYahnYKgLMTgAd66PRvBOoamUluB9ltzg7nHzEcdB9DXX6Z4e0nw98/Nzc/3m7de3Qf/WFW7jUZZ8qvyJ6Cuhwo0Nazu+y/VnJGpicU7YaNo/zP9ESRRWul2C2OnKFRRgkd/c+9QVX6dKUSMO+frXn4ivKvK70XRdj18JhYYWHLHVvVvq2T0VEJvUflThIp7/nXOdg+ik69KKAFopKKAFoozRmgAooopAFFBIHU4qJpSfu8UwJGcL1/KoWkLewplFABS5pKKYhc0UlFAC0UlLmkAUUZopgFFFFAwNAYjoaSigQ8SsOvNPEoPXioaKBlgMD0NOzVWlDEdDSAs5oqESnuM05ZAxxg5oAkpGYL1pjyMOgI96iznrQIe0hbpwKZRmigAooopjCiiigC1mkooqQCikZwvWomkJ6cCmBKzhev5VE0hPTimUUAFFFFMAp0cbSyKidWOKbWrotoZpt4UsSdqgDJ/wA9q3w9H21VQ/qxyYzELDUJVOvT16HXeD9IXzRcPny4MFc5G5+o/Lr+Vdk19zhs/UVQhiSysYraLHyr8xHc9zTCxzU4qt7apePwrRegYHDPD0bS1k9W/Nmql2D0cH61KJlPVfxFYwf1p6uV+6xH0Ncp3GvlD0JH1oYbQT6VmC6kUfMcj3FV7zUGW1cFc7hg554oAtaitnc28trfCN4plKSRPzkfSvBPGHhWTw/es0DNPZMf3cpHK+gP+Neqy3HmKdxYE9+9VbrZNbtExyMcZH5j6UAeEP1ptdtq/hC0m3S6bIbZxyYj8yn+orl5tE1GDP7jzQO8Zz+nX9KYFGikbKNtkVkb0YYNGc9KAFooooAM0uaSigBaYJQnfn6U6qE822TGaAHS3T+Zwxxu/OrgOFJPpWWoLyA/jW1a6feaiqxWULSO+ATjgdMkn8a9TAp8rZ4mZOPPFNmK5yxxWlo/h++1qYLbRER5G6Vgdo/H8D+VdppfgawsUWfWJPPkPJjPCj/HqK2ZNQEaeVZRrFGOmBiuiVGFL3sQ7eS3/wCAcccRUxD5MJG/957f8EzNM8H6XoyrNfsLqdcHDfdB46D6itGbUnZQluoiQDAA7CqjOzsWckk9zTa5qmNk1yUVyr8fvO2jlkFL2mIfPLz2XohSSTknJ96M0lFcB6wuaSiigYUUUUAKDjpThIw9/rTKKAJRMO4pwdT3qCigCzRVcEjocVKm8/e/UUAPJx1qNpf7v50MjHvmoyrDqKAAsT1pKKKQgooopjCiiigAooooAKKKKACiiigAooopCCiiimMKKKKACinKhbp09alVAvufWgCNYyevAqUKF6ClooAKQqp6ilooAYYh2OKYYmHTmpqKBFcgjqKM1YpCinqKQEGaSpTEOxxSeU3qKAJWYL1qNpSenFR5ozQAUUZozQAUlFFABS5pKKAFzXc+E7ILMryDiFM88/Mf8k/hXGWcfm3ka9s5P4V6LosQi0zcPvSNnPsP8mu+i/ZYadXq/dX6nk4j9/jaVHpH3n8tF+JqMxJJ9aQMag3kHg04S+o/KvMPaJwwp24VAHB704N70ASO2VwOc8HI7Vn6nPsjVRjcecVZchm4JyB0zWRqzfv1A9Mkev8AnFAFZpmPQLULTP3Ao83HBUfhTGlH92gDN1GMHMyAq38WPT1rC3kOV6gHiumlYH+HNc3coEvZAv3eoHp/nFAFTUtkkSiRFbHZlBrGl03TpfuHynx0Vun4Gti/B2Cs7djtQBlTaSyE+TcI4HZuDVN4Zoj88efdTn+Vb7OM0wt6UAc/uFLketbMkMUv+sRW+oqBtIglPyM8f0ORTAyJ51jTrzV/SPBeqaziebFtAT99xknp0H5/liuu8M6HZafYSXsq+fMGOGdRlQMcD8hWhcalNNkKdieg716UMPRpQVSu73V0l/meLUxeJr1JUcKkuV2cn+iM218CaTZESXc8ku3khmAU8+mK1zeW9nCINOhWNVGAQMDj+dZxOetFXLHuMeWjFRX4kRymM58+Jm5v7l9w55HlbdIxY+9Noorzm23dnsRioqyWgUUUUigooooAKKKKYBRRRQAUUU5VLdKAG05Yy3sKkWMDryafQA1UC9OvrTqKSgBaKSlzQAhUHqKaYgenFPooAhMTDpzTSpHUVYooArUVYKKeophiHY4oAiop5iYe9NII6jFACUUUUgCiiigAooooAKKKkWLPLce1MBgBJ4qVYgPvc08AAYFFABRRRQAUUUUAFFFFABRRRmgAoozRQAUUUZoAr0UpUjqKTNSAUUZozTAKKM0lAC0UlFAF7SlJvM9gpzXoEC+XaxKD0QdK4LR/9dJ/u12qsQowewrtraYSn5tnl0HzZhVb6RivzZaEjD3p4lHfiqglI6808SqevFeceuWgw7GnBx3qqGHY04SetAE3m8kA5HrWBqMrNfPg9OOtbHmKM5Ofwrlp5Q9xI2erGgCxvbHJpC5xziqvmClEg7GgCVyMZ6VgW7rPrckMnIeLA+uc/wBTWxLJiFz1wprnrGQDxGjMQNxK/pigBNWt3t2CvyOx9ayiBXXa1Gslttb6g+lcs8eGIPBFAFZl5pjLU7JzxUTjbkscUARYqWL71MAL8jhfX1qROG44FAHUaYM+HpB/tGqhRh2q3pX/ACL7/wC8air08X8FL/Cjxsv/AIlf/G/0K/TrRVggHqM00xKfavPPWIaKeYj2NNKMOopgNooopgFFFFABRRR16UALmlAJ6U5Yv735VKAAMCkAxYwPvc1JSUUgFzRmkooAKKKKACiiigAooooAXNGaSigBc0ZpKKAFzRSUUANKKe1NMPofzqSimBCY2HbP0ppGOtWKRiAPmoAr09Yy3sKQsN2Qop4m9R+VAD1UL0/OnZpgkU9/zp1AC5opKKAFopKKQC0UlFAC0UlFABRRRQAUUUUAFFFFABSFFPalzRQBGYfQ/nTTGw7Z+lTZopgVunWirJAPXmmmJT04oAgoqQxEdDmoyrDqKANDSGAuXXuVrsh9wfQVxOlf8fw/3TXag5jUjuorrqtvCw8m/wBDzaMUsfV84x/VChjS7xTKK4D1SQN6U4SEdeahpcmgBSxWInAHy9ulcu0vzE+9dDIx8t+exrlieaAJ/O9qaZj6VDuozQAtxL/o8n+6a5uWR47nzIzh1OQa35j+4f8A3TWBMP3hoA0bjULt7NZ5YT5Z6sOQPy6VmyXsUgyMlvQDNMpCaAGmSV1yihB6tTNg3bnO9vftTix/CmFqAHM1Ef3qjLU+I/NQB1Wlf8i+/wDvGoql0r/kX3/3jUVeni/gpf4UeNl/8Sv/AI3+gUUUleeesLmjNJSgFjgcmnuJu2rEIB6imGMduKvx6dK4y5C+3U1IdKbtJ/47/wDXrsjgcTJXUTzZZrgoy5XU/N/kjKMZHTmmkEdRV+azmhUtt3gf3aoM7E4PHtXPUpVKTtNWOyjXpV481OSaHLET14FSqoXpUAkYd6eJv7w/KszclopolU98fWnZz0oAKM0UlIBc0ZpKKAFzRmkooAXNFJRQAtFJRQAtFJRQAtGaSigAzS5pKKAFzSE460xpAOnJqJmLdaYEjS/3fzqLJPWiigAozRRTAXNKCR0OKbRQBIJWHvThKO4qGlzSAsB1PQ0tVs0oYjoaALFFQiZu/NOEoPXigCSjNNDA9DS0AFFFFAC5ozSUUALRmkooAKKKKACiiigAooopAFFFFMCzp4H2wHHODXWwf8eyf7tcnp//AB+D6Gusg/49Y/8Adrun/uUf8T/I8qn/AMjOX+BfmOooorzT2AooooAhk5Rvoa5VvvH611b/AHW+hrlG++31oASlpKKAGTf6l/8AdP8AKsOb75rdk/1T/wC6aw5f9ZQBDimVKRURNADG5yewNRnrT24Q/Uf1qI8mgAz6c1LCpLgYyTwAO9X9D0C+12+Ftp8W9sZd24SMerHtXrfh/wAPaR4TUPGBc32PmuJFyR/uj+EfrQBymi+HdWGjhbjTLpFc5AaI5IPt1q7/AMI5cRIDJZNGD08xcH9a7GXxEc9f/HsfypE8Qlm6n8GBr0oZhKMFHlTtpdo8aplMJ1JT9pJKTvZO2pxUuhsBlrb/AL5/+tVJ9IVWIO9T6GvUIoYbyLzZraNE5O/7lRvo1tIAbe54PPZgfpT+vQl/EpJ+mhP9l1Yfwa8l66/5Hl39lHdzIcf7tatjo43D5fLBBOT1IB5x+f6iuovND1Ca7ENg8SRqPnuZV5B9FHOT79O3Xotv4XmtGYxJ50j8vMz7mb8T29ugo+u06avRpqL77/cN5bVrO2JquUeyVr+tihDZQQrhYwx/vMM1KLWJj/qUP/ARWgdMuUODBJkeikj8xTApVijAqy9QRXDKtUk+Zydz04YajCPLGCt6FCbSYJlOF2P2K8Vyur6IYpGG0LJ1UgcMK7rIUZPFUb8R3UJjbAHUN/dNddDFv+HX96L+9eaODE5er+2wvuzXbZ+TPMmUoxVhgjqDSVq6ta4zMAAyna+P51lVniKLo1HHp09DfB4lYmiqi0ezXZhQDjpRRXOdg8SMPf604TDuKiooAsB1PQ0tVqUEjocUAWKKhErDrzTxKD14oAfRSBgehpaACiiigAooooAKKCQBzxUTSk/d4oAkZwvX8qhaQt7Cm0UAFFFFMApKWigBKUUYooAKKKKACiiigAooooAKKKM0AFKHYdDTaKQEglPcZpwlU9eKhopgWAwPQ5pc1XAJOBUyqQOWJoAVpAv19KiMrE9cU4xDsfzpvlN7UAT0UUUAFFFFABRRUbSgfd5pASE461G0v9386jLFuppKALulkm/GeflNdlB/x6x/7tcbpX/H8P8AdNdlB/x6x/7td1T/AHKP+J/keVT/AORnL/AvzHUUUV5p7AUUUUARPyGrlH++31rrG71ykn+sb6mgBtFFFADZP9U/+6aw5fv1uPzG30NYcv36AGHpUJOBU1VnPagCNj19c1paBod1r1+Le1GFHMkrfdjX1P8AhWWf4s9q774fEJo92VHLzgMfXC8fzNAHXWUNn4f0xbHTU2r1eUj5pW9T/nAqrNO0hO4nB7VAJy95JGxyVB/mKcaAG1la54ntvDtuCNsl44/dxkZC/wC0R/IVqPIsUbSSHaigsx9AK8X1XUZdV1Se7mOTIxIH90dh+VNAX9S8U6hqs5lvJpJjnjzHJA+g6D8KjtdYa1uVmhlkhlT7ssbFWGfQisgjHWimB7B4T+Jl5B5UOrt9vtPu+b0lQBQB/vdO/PJOT0r1ywvLfUbKO6spFlgkGVYV8lWl3JZzB4zx3X1FeoeBvGP9lXKbpSLG4OJQT/qz/e/Dv7fSlYD2xnCfWszUHSVMYDMpzyOBUNzqMaAhTvb2rEvdR2qfNfA/uL3pAW7zUNOgs1neBpVLhT5KlinB5IXkjIx0PUVPZ6joupQ5gW2lTO0lVBwfQ46GuGutWZ2ktkXYG+YHP+f8mjT7Rrq5jlRN85+4ycN+YoA6nVvCcN5++0kIMrh4s5DfmeK5e88G3MBPmWUi98x/Nj8sivR9KsZLW2Q3blpMcJnO3/E1fIBrvp46UYKE4qSXc8qrlkJVHVpTlBvez0+48Pm0N0JCOQQcYcVTlsLiLrGWHqvNe53Gm2t1/r7eOQ+pXn865TxVpui6HpbXs5khJO2ONDu8xiOnP0znNbRlg675eVxb7ao55xzHDRcueM4rvozy8gg4Iwfekq+NftZQBNbsAeowGH+fwqRf7Lux8jqjHtnafyreWVy/5dyT/A56eeQ/5e02vTUzKK0n0hv+WUob2IqpNaTwcyIceo5rhqYSvS1nE9SjmGFru1Oav935kFFFFcx3BTg7DoabRSAkEp7iniRT7fWoKcqFvYetAE+QRkGo2lA+7zTgiqOn50hRT2x9KAIiSTzzSVIYvQ00ow7UANooooAKKKKACiikoAM0uaSigBaKSigBaKTNLmgAooooAKQ0tIaACiinKhbp+dIBtPWIn73FSKgX3PrTqYCAADApaKKACiiigBaKKa0gX3NADqY0gHTk1GzluvT0ptACsxbrSUUUAFFFFAFnT3KX0eO5wa7eD/j1j/3a4ax/4/ov96u3tm3WqewxXW7vB+kv0PNVlmPm4fqPooorgPVCiiigCNuprlZRiZx/tGuqbqa5Wb/Xv/vGgBlFFFACP9xvpWFN9/8ACt1uUP0rCm+/+FADKqyHmrJ6VUkPNAERP3/p/Wu/8AEf2PcDv5/T/gIrz0tyfeu9+HrA6ddgHkSAkfhQBqLNt8Syxf3o8/y/wrRNYUz7fHUKD+ONv0XNbxFAGZ4gYr4d1Ar1+zuP/HTXjPSvcb21+12NxbE4E0TR59MgivEJY2jd4pAVdSVYEdCKaAQknGcZx2opQq7S2cnPI6HFNCMOoIGM5x+VMBau6beNbyFCflf9DVEdW5HHA96CccjtzQB7N4d8RyXnhuEfemiJhdyPTGPqcEc+tZ2ra55ZZfNyx6kHNcV4a1CUWt9EspVS0ZIH0bPP5VLNONx2/makC42os8pOSq9Tk9eRXsfhG0htdJgu8bpZowVOPuqeleGJkwyyEE52oOepJz/7LXtukXgfTICOB5a4Gc9qAOlNzlutPWc/WsVbn3qVLkdnxQBrm6RELyNsVRkk9AK8K8ZeKG8S6606M32SH5LdSMYXuT7k/wBK7r4h68+neFpIUZfMvD5I9QpHzH8uPxrxoSfUV7WW0VrVfoj53OK7bVGPqy6Hre8KeGLvxTqnkQlobWLDXFxtyFH90dix7enX2PLiQnhQXPZVHJ9hX0D4Q0iPw54cgstv79v3tww/ikPXuegAH0FdeOxPsYWjuzhy3CfWKl5/Ci7F4d0XTtNjtY7JFhQYDdW+pbqT75rndd8OxQRvLaKQo5MZ5GK7QSJKhRsEGoJrLfCY85HbPavEo4utSldS+XQ+ixGAw+Ijyyir91ujxHULUW1x8gwjcj29qqV1/iPRZ4ZpYWjKDdujYg7T7Z/MVzEun3MXWMsPVea6sVhpc3tKSvF6+hxYHGx5PY15WnHTXrbqV6VVLdKesJBw+c+lXrewkkAJGxfeuSnRqVZcsFdnoVsRSoR56krIppFz03Gr0WnzSfe+Qe/Wul0TwpNdYkC+XF3kc8n6D/PWuzs/D+n2iACBZWxy0o3Z/DpXW6WHw/8AGfNLsv1Z5yxGLxeuGjyx/ml19EeVPpUijIf/AL6GKrTWssP3149RXtH9kWMi4NlBg+kYH8qy9S8GWs8DNYfuZccISSre3P40c2Cqu1nD8UPlzKiuZtVF22fy6HktFaep6XJbTOPLKOhw6HqKzK5a9CdCXLL/AIc7sLiqeJhzw+a6p9mIQD1FNManpxT6KwOoiMR7c00qV6ip6KAK1FWCqnqKYYh2OKAIqKeYmHvTCCOoxQAUUUUAFFFFABRRRQAZpevSnLGW68CpVUL0oAYkXdvyqXpSUUALRRmjNAAaSiigAooooAid2PB4FMqzTTGp7Y+lMCCipDD6H86aUYdqAG0UUUAFFFFICex/4/ov96u1sv8Aj1X6muKsT/p0P+9Xa2X/AB6L9TXf/wAwL/xL8jyn/wAjOP8Agf5ktFFFeaewFFFFAEbfeNctOMXEg/2jXVN941ytx/x8yf7xoAjooooAQ/dP0rCm+9W6fun6VhT/AHhQBExqnIatMapyc8e9AEWM7var2laxdaNL9otGGc4ZG+649DUuk6De6tIRbxnZ0aQ8KOfX/CrfijQE0GG0RZDI8oYuSMDjHQV04VJ1UmceNbjQbXl+Zbj8T2F5rcGo3Mz2csQwUKblOVIPIrXm1q3mG+11xBnkKCn5c15jJ96qr9K9CeCpvVaHl08xqrSSTPT08VXFodt40Eq9iGwSPWuK8SS2dzrMl1YuCtx87pkHa/fp2PX8TXNyLkYNVwXhkBB+hrllhVHZnbDHOT1RrU4HauVYgnIIB7VBDOsy+jdxUtcbTTsz0YyUldB1I4yR0psh2RsTxgU6pY9L1DWpDFYW5cH7z9FH4/hTjGU3yxV2KdSFOPNN2Q7w7KyWd2zA/vHQKe3AbP8ANfzrXtbS51Cby7ZCx7tjgfjXQ6R4YsNEs4xdMJ5VJYk9C3XOPw/LitKTUFRNtsm0DjJFegsFCkr4idvJas8d5lVrvlwlO/8Aeei/4JUi8P20EUMdxP8AMh3cYG5vX3/+t9a6zS7nyLUW5Odv3SfT0rjpQ0rbndifetTSJSsJjJ5Q8fQ/5NaQ+r4hulCFnbR31v5mFX63g0sRUq8yuuZW0s+3odcl171Mt1/eH5VzUeoqTtJ+YHBGatR3w7Pj615DTTsz6FNNXRyvxNv/ADdRsrZT8scRfHuxx/7LXECT8K6H4gyl9ehYkEGAAEfU1ywkr6DCSSoxR8rj4uWIkzqvA9st94ysFk5jhYztjqNgyv8A49tr3NLk/wALhvrXjHwyONbu5gMlbfZnHqwP/stenrdD+IEe4rzMdNyrW7Hs5ZTUMPfuzdFyD94FfcVMlyf4HyPrWGl0f4XB9jTxc88gqfUVwnpGzN9nu123UMcv++vP51nT+FdOuz/o4kt39juH68/rVW91mDS9Plu7uQCGMZPqfb615Lr/AMSdR1OV44J5ILbPEULbRj3PU1vSxFal8EmjmrYTD1/4sE/67nqMvgeQTfJcwOoP3jkEfhg/zrW03w1Z2W15B58o53N0B9hXz/B4pvkk3C+vIiTklZm5+vNeg+F/iVewvHHqzC+tmJ3TDiReOOnBAOPfk9a3qY7EVI8rl92hy0sqwlKfOo3fnrb7z1lI+AFGAOAPSpljx15NRWF5bahZR3VnIJIZBlWFSyTLGPU+lcJ6Y9cE4zz6VCZWJO1l+lZ15qCwMJHbleijvXO3fiIwai/lt8jfMMGgC/4r0lru1+1RRfvYVJdhj5l7/lXmtxYSCVmiXKnnHpXplj4kExCt+OR/UVafQNM1MGUwyIT1dW6/nmvRo4ilKl7Kvey2a6HkYjCV4VniMLa73T2fn6nkDRSJ99GX6imV6fdeCVOTaXGPRZB/Uf4VgX3g68iJza+aMfeh5/Qc1f1ahU/hVV6PQz+vYmj/AB6Lt3jqcfRWpPozRsQNyMOoYVTksZ4/4dw/2eayqYKvT1cbry1N6OZ4Ss7Kdn2en5leiggg4IwaK5D0AooopDGmNT2x9KYYfQ/nUtFAEBjYds/Sm1ZqJ5B0Az7mgBiqWPFSrGF68moxIy/T6U8TDuCKAJKKQOp6GloAKKKKACiiigAooooAKKKKAFooopgFFFFACFQeoppiHY4p9FAEJiYe9MII6jFWaOvWgBlj/wAf0P8AvV21l/x6L9TXIWiKLyMgY+auwsv+PRfqa7l/uL/xL8jyn/yM4/4H+ZJRRRXmnsBRRRQAxvvGuWuuLuX/AHjXUt941y93/wAfcv8AvGgCGiiigArBn+8K3qwLj7w+lAEKxtPMsUeN7nAyQP1Nd54d+HgkRZ7/AGyk8jOQn+Lfyrg7bm9i/wB6vR9KvLi2tYfJmdPkHQ+1AHbWWgWdtGF++AMBQNoA+grkfin4dn1KzsZdNjBkh3gxDC7gdvT3rXttfuVx5yrJ79D+lWbnUrXUYVSYyQspypIyBV05unJSRnVpxqwcJdT50uY3hlKSoyMMZVhgjiqb17nrGm2F3JGl7bwXJZiFYqCQME8HqOlc5d+CdDZmMdtJEOOEmbj/AL6Jr1FjoNe8jxZZbUi/daZ5O9ROoIOa9KbwNo6MDvvG9jKv/wATULeF9Jt1O2080knDSytx+XH6VMsVTLhgqq3sec21vPcz+XbxtI/XCjoPU+g969S0bRrGz0GA3kUc7sv7xiA4yT/+r8hWVOvlAwxqsUYP+rjUKv5DjPvW5ZSeZ4cjz/C239arC1k5TlbaLauZ47DyUaceZ2cknbTuSC00VCGFpASORiED+lSPfhYxHap5agYHGMVSorOWYVmrRtH0RrDKcPGSlO8rd3cVmZ2yxJPvSUUVwNt6s9VJJWQU+GVoZA6dR29aZRTjJxaktxThGcXGSumaDLaX+C5Mcv8AeBwaabK8h/49rrcP7sgz+tUafHNJF9xyPau/61Rq/wAend91ozyfqOIof7rVsuz1Rj+LLK+uYI5zaEvDwxiGcqfb2rjvM2sVbKsDgg8Yr1FNSlXiQK4754olvbC4Xy7m3EinqGUFa6qTw1rQqW9UcddY3m5qlLm84v8ARnM+Ab1YNamjbrLCdvPUgg/yz+Vekx3w7MR7NXILoGizXSXNn/o8yHI8o7c/h09unetD7Jew/wCouN6gdH5rDEYKrOXtKdpLyZ04TMsPTj7KreD81Y6ZboHr+YNTLc4HDGuUF7dwf6+3bHcx8/pViHV4mOC+0+jcV5c6c4O0lY9unVp1FeEk/QwviXrTzXFtpkTfu4182THdjwB+Az+dcFW34vkMvie4cnIKpt+mwViqpZsDrSNBOlWLO8ls5t8fK/xL2IqBiSxJ696Nrbd207fXFAHq/gbxf/Z15HGZM2V0drA/wN2P58H8+3Po97qe1flP4182WN2YQ0Z+63PPrXpcHib7V4etZpWJkaPaxJyWZflJP1xn8aTA39T1kIxEfzyep6CsOG7N3cqjYc5O5lI/KuavtWlnJAbYnt1NaXggibxDBE6/usGRvcDt+dID03QNBXy1ubsYU8onTP19q6kSKAFUYA4AHastbvI3Z47Uq3GejUAaoYHvQQDVBbj+8M/Ss/xH4gj0Lw/c32751XbGvq56djVRi5SUVuyZzUIuUtkYXjzxnbaUW02xiiuL7HztIoZYM89D1Pt2rzaPxBeIxeZ1kHJO4AY/Ksme8lu7mS4uJDLNKxd3J5Ynqa7n4aeF01e9bVdSiR7K2bbFG/8AHKMHJHdQD+JI9DX0sVDA0b3/AOCz46fPmWI5WtPyRpaX4c1PWtLW8ewEKuMxxynDMPUA9B6Zx61Qv/D01rkSwSQH1YHFevXF4ltDgY3muZ1RTdRsZOQe1eZ/aHtH++gmvx+89f8Asn2Uf9mqSi/W6+48sliaF9rjBplbGq2+I3/vRnisYkKMk4rDE0VSn7rumrr0OrBYiWIptzVpRbTXmhaazhfc+lRtKT93gUwAscAZPtXMdorOW6/lSAZOBWpZaNLO6+YDk9EUcmuxsPBl00YLiO2X+6T81d8cG1HmrS5F57/ceVUzJOfJhoOo1vbZfM86xjrRXqM3giQ/6i5jf13qV/lmud1fwhc2qGSS2Kj/AJ6R8r684+hprCU5r9zUTfbYTzCrSf8AtFFxXdar8DkKUMR0NS3NrJaviQcdmHQ1DXDOEoS5ZKzPUp1IVYqcHdMeJT3GaeJVPXioaKgssAg9DmlqrThIw7/nQBYoqITf3h+VOEqnvj60CH0UdelFAxaKM0ZoGFFFFABRmkooAXNGaSigB8ZxKhHXcK7Ky/49F+prjI/9Yv1FdhYNmAjsGrtim8JPya/U82pJLH0vOMv0ZPRRRXnnqhRRRQAxvvVzF7/x+y/7xrp2+9XMX3F9L/vUAQUUUUAFYN19+t6sC7+/QBXtzi+i/wB4V6FY/wDHrF/uD+VeeW//AB/Q/wC+K9Dsf+PaL/cH8qANKPpT/pTI+lPoARvu1Sm6vV1vumqc4+9QBnzDkVRnXK/jV+XqKpzD5fxoA529UiZ/rWpp3Hh0f9dTWfe/65/rWhp//Ivf9tTXbhPt/wCFnm4/al/jj+o2ikzRmuU7xaKTNGaAFoozSFgo5NAC01pAvufSomlJ+7wKZQIczluvT0ptFFAgqWK6nh/1cjAenWoqKqMpQd4uxM6cKi5Zq68zQj1iUcSIrj8qnGoWdwMXEW0+pGayKK7Y4+ulZu689Tzp5ThJO8Vyvunb/gEXiXQobi0bUbBgTGuXXPUDr+XWuQgIEyluBXoOm3Qhl8qT/VyHv2Nc14j8PyabdNcWsZNo4yMfwHuP6/8A6qqtTjWpqvSVu6XTz9CcLXnh6zwteV76xb6rt6ow5xidqIpNhwT8p68ZpjHccmhcFhu4HevNPaG3BEe4r0Hoc1uaLcs3h0Fj/wAvLhfptT+ua528VnIt4VMkjHhUGSfyrutG8LzLpttHfEwJEmWXjJYnJ6fln2Brajh6leXLTVzlxGLo4aPNVlb8/uMoeZPIEiVncngAZrovCaPa64ksoMeB5WGGCc9f1q/E1hpyeXaRqzdCe5+p/CpY9RiZl3xlOeCO1dn1OgvdlVXN+H3nm/2jiZe/Cg3D1s/uO7S5xGOalW59a52C9YxgMeRVpLz3rgq0pUpuEt0erh68MRSVWGzN5Lk/wv8AhXnfxU1lpJ7LTAcBFM74PBJ+Vfyw3511q3IbrXkfjW8Nx4vvTuJVCqLnthR/XNdWAinWu+hx5pJrD8q6szkZnYJGAzscKM4yT0r6K0DTv7L0S0062VQtvEFYg8M3Vjn3JJ/GvBPBsAu/GGnKwyscvmkgZxsBYfqBX0LplzC9uYwwEgOSCevpW2ZVHKSgc+UUVGEqnfQlNiZJN004H+yozj8apXv2dVKRsX9WJ4p2rvcw4kRHkgA+YIM4Pv7V5/rni8DdHZEMf7/8I+nrXlHtk2soomm2nIKnNccWLHJOa0dIup75rszvuLDC/r/9ar9hoKyuFYCRu5Jwor2/q88Rh6TWiSd2/U+aWLp4TF14yu22rJehj21jNcn5RtX+8RXXaF4TmucNGvlp3lkB547fp+ddJpPhWCBFkuGimfsgGVU/gRnvW+C8K/NB8g7xc/8Ajv8AhmsXXo4bSh70v5n+iOhYXE43XEvlh/Kt36srabo9tpiYgXMh6yHqf8/41obG7U+ILIoZGDD/AGalcbVHFebOpKpLmm7s9mlShSgoU1ZIrhyp5TIp/mwum2ReD1DDINPwDTWjHcVBoef+LtAhtpPMtx/o038I/wCWbe3t6V5/PC8ErI45B6+te73mnxXlrJBKMo4wRXKX3giQgm3kjlXP3HXGB+terGpSxVNRrStNde68zwZUa+BrSnh4c1OWrS6PyPL6SuuvfCM8IJls5EHdk5A/LisOXRnAJhkDexGKmWAq25qdpLyZpHNsPflq3g/7ysZtFPkhkiOJEZfqKZXC04uzPUjKMleLugoooqRgCR04p3mP602igZaooooGFFFFABRRSUALRSUUAPj/ANYv1Fdfp/8AqW/3q4+M/vF+orsNP/1L/wC9XfT/ANzq+qPKr/8AIwo+kvyLNJS0V5p7AlFFFADG+9XM3/8Ax/S/71dM33q5rUOL+X/eoArUUUUAFYN2MSGt6sK75c/WgCrB/wAfkWf74/nXoNhzaxf7orzwNslVx1Vga7bS9UQ2kYcdBjIoA6CPpT6qwXcLj5XH51ZDA9DQAH7pqnP3q4elU5+9AFGWqcv3auS1Tl6GgDAvf9a1aFh/yL//AG1NZ97/AKw1rRQG10KKN+GZtxB9Tz/n6V3YOLtUl05WeXmEknRj1c1+BWooorkPRCjp1pjSBenJqJmLdaAJGl/u/nUROTzRRQAUUUUCCiiigAooooAKKKKBhV611ExqYrgeZEeCCM1RoralWnRlzQZz4jD0sRDkqK6JZfDGi35325aAnGVjbH6HP+RVI+BV83i/Oz08vn88/wBKsUV1vFUJazpK/k7HAsDioaU8Q7eaT/Et2ek6PoLGSJPOuf78nLD/AA6U25vprk/Mdq/3QarUVnWxk5x5ILlj2Rrh8vp0pe0qNzn3f6dgpQ7DoaSiuI9E37O6a4swycyKMEe9JHq6h9kvysP4WGDWLDPJbyb4mwf51orqVvOu27hH1xkV6cnRxaXPLlmlbyZ4sY4jASkqUOem3ey3X+aNaPUUP8ZX68ivM/E7f8VNe85y4bP1ANdwtvZynNrOY27AN/Q1g+IPCl7e3H2qzljkbaFZSNpPvWmHwdalLmVpLyZjisxw9eChK8ZdmrFf4fuF8UBiQCsDkHP0r1kSHaHU4OOorxPRDd6J4gtpb61lhTJRmkUheRjr06kV7BZXAmsIpFOVZeDXDjL+1uz1Mvt7Gy7mlDq95b/dlJHoaLmXS9V+XV9KgnJ6yBdr/wDfQwaobgaxfF2ptpfhu4libbLLiJD7t1/TNcZ6By+ueI9M0vU5ofDELrBnBaVy+4+2egrmZdYmnkLyjcx6nNZxOTzRWjqTcVBvRGapU1N1EtX1On0HxNdaXcbtOu5LRySSoPysSCOQeCeT1Fe0eEPHcOs4tdTWO3vC2EK/ckGO2eh68fTk84+cK3dE1d0kSGVyrKQY5A2CCORz2Poag0Ppi6U2TG4tQB3lQDhh6/Wl+2x3FqzRNu4DDFcl4d8V/wBsaJidi13D8kxwBn0bj159OQeKdYaksV5PAGbYynYB/n/OKQHTx3QPf86sLMD3rnYbvp8351eS4OORn6UAbAkzS7l78VmLcj1Irn/HHiQ6N4fcQsrzzjaqkE/L0J47cgfjVRi5yUVuyJzUIuUtkc34+8avdynTdNcfZCuXdTzJkd/buB9D6Y4eDUri15jmIUdQx4qhJcvNM8sp3O7FmI7k9arXV0EXHUDsw6n/AOt1/KvqaNGGGp6bnxWIrTxlX3tu3Y6AeNIUyLu3JHrHySPoen51oQvY63aNcaawDqfmXGCD7ivPVikuH2xqXY1q+GJ5tO8SxW5ZWWf93IqtnHoevXP6E1m6ntnyVVdMv2H1dOph5WktfJ/I3SCpIIwRwQe1FWtSjEd84HQ81VrwKsHTqOD6M+roVFWpRqLqkwooorI2LOaM0UUDFopKKYBRRSFgvWgBaazhevX0qNpSfu8UykBIshaVOw3Cu30//Uv/AL1cKh2upPY5rutNObdiO7D+Vd9N/wCyVV5xPKrp/wBoUX5S/Is0UUV5p7AlFLSUAMf71c3qP/IQl+tdI/3q5zUxjUJKAKlFFFABWHdDD4963Kxbv/Wt/vGgCk6g1Ml5PGQUlIIAXt0HTimEVGRQBqQazKp/eID7qcGtK28QIMfvSh9HFcyDtpu6gDvoNaLLk4ceqnNJNq8H8Rx9a4RJWjbKMyH1BxU/22Z+Gfd9RQB0s+tWq/x5+lUJtaV+IUbJ6ZFGiWcE6TT3CB/LP3SAR096v/2rbouIoD7AgAV6FPBwdNVKs+VP5nkVswqKrKlQpOTjvrZamdZ6ddXl0s10higU7m3jBbHbHpV2+n82XYv3E4Huagn1G4uflB2j0WowG28nn6U6tWlCn7Ghs92+oUMPXqVvrGJtdaJLZf8ABBmC9aiaQt7ClMTdjmmEEdRiuE9QKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAClVSx4p6xd2/KpAMdKAGrGF68mrEd1NEflc/QnNRUVcZyg7xdjOpThUXLNXXmXk1LP+uiB9xW1YXUYtwi8R4+X2/wA5rl6lhuZID8jceh6V3xxntI+zxOq79UeVPLvYy9rg/dl26M68MG5U5rkviKrHQbcjO0XAz/3y1W4tVwR5iEe6ml1BYNd02Sykk+9hhkcqR3FT9Sp1P4NRPyejK/tKrR/3mi0u61R5XRVi/spdPvZLacYZD1/vDsar158ouLs9z2IyjOKlF3TCgEggjgjpRRUlHW+E9dkt9YjQPtFzGY3JOADjIOPqMfjXU2OsPNrAVJBtGcY6mvLNNmI1+xRT/wAvMeee24Z/TNdjpNyINQa5Y4jh4J9c8CgD0q3u8qDuP41oxXPHXFcxBc/u1+Y9OvWr8V3xwQaQHQrct7NXl3xJ1X7RrKWoOBCBlSfbg/qfyruEul91rx7xNdfaPEl42c4kIB9a7sBFOtd9DzMzk1h7Lqyr5uBk9B6VnzSlpMHqOuPXvUkkm2Mk1ShffOoPc5NezWldqJ8/QhZORfurk2FiIoXxNNy5HVR/n+tReGSf+EnsP+uwqhdymW4Y9hwK2PBdr9q8UQHGRADKR9OB+pFc6bnXil3OqcVTw0nLez/I7fVv+Qg3+6KpVY1CTzL6Q9gcColjz14FeZipKVebXdntYGLhhacX2X5DQCxwKkEQxyefangBRgUZrnOwaJVPXinAg9Dmq2aAfSkBapCcDmmoH/iPHoaRo2PO7NMBGl/u/nUZOetKVYdRSUAFFFIaQBXc6T/x5D6D+VcNXbaQf3OP9ha7aMb4er8vzPMxMuXF0PPm/Iv0UUV556wUlLRQBG/Wuc1P/kISfhXRv1rndV/5CD/QUAU6KKKACsa8H75v941s1jXn+ub/AHjQBWIphFSUnG7mgCLbnpTStTAfNzxwf5UwqaAI8UKMGnFaAKAOj0D/AI8Lv/PaqqxE/e4Fafh6zkGkSXDDCSyFV9wBjP55/KqkqeVKyH+E4r06yvhaUl0uvxPFwr5cdXi93Z/KwwKFHApaKK4D1wopKKAEMantj6Uww/3T+dSZpc0AQFGHam1ZpCoPUUAV6KlMQPTimmJh05oAjozQQR1GKKQBmiiigBaKSjNAC0UmaWgAooooAKKACTwKlWMD73JpgMVC3sKlVQvT86WikAuaKSimAtFJRmgBaKKKYBTo3aKQOvUGm0U02ndEyipJxezF1zRotesxPbYW6jHGf4vY15/LE8ErRTIUdThlI5Br0OGZ4JNyHHr71LdW2mauv+nQqJMY39CPx/CvRkqeMXNdRn57P/gniw9tlr5VFzpdLbx/zR5tgbQc85PFNJCqSeAK7s+CNOYkpPPtJ4w4/wDiadF4O0i2fzLppJlH8EjDafyAqP7Mr9bJd7mn9t4XaKbfaxx/hnRr2/1NryOGRUQHYzLgEsCOp9Bzx32+tdz/AMI0n2WOL7QVdW3swHBP+FWHvxGnlWcaxxjpgY/QVTLszbixLeuaf+x0dLc7+5fIm2YYn3ub2S6Ld/Mu+Tf24G3bMB3B2n/CnLqjxHFwhU/7Yx+vSq0d7PH0csPRuasLqe4YmiBB6kUnTwdT4ZOD89UUqmZUfiiqi8tH/kXodVQ4+Yr9eR+deZ66duvXfvISPxr0Dbp85+XETE9vlrC13whJeyfarCfdJtAKMB83oc8Vvh8HUpycoNSXkzlxWY0qsFCrFwd+q/U4md/3LfT1qlbyYnUngVpXumahaIwu7OWMDOW25Ude44rGzzVVJNSuwoxi4tLUnY5avQPBWnHT9Hl1GdSJLgZQHjCjp275P4bTXLeGtAl1vUF3oyWqDdI+MA+wPuf5GvQr140CW1uqrHEMAKMAdgPwpwfsabrv0XqRVj9Zqxwsdt5ei6fMphfmLMdzE5JNOoorxT6RaaIKKKKBkCxlvYVMqBen506ikAUUUUAFNKKeop1FMCMxf3T+dRmNh2z9KsUlAitXbaR/q/8AgArkiAeozXW6T/qz/uCu7D/7vW9F+Z5WL/3zD+svyL9FFFeaeyFFFFADH61zurf8hBvoK6J+tc9q/wDx/H/dFAFGiiigArHveJW/3jWxWRff64/U0AVaG6/hRSUANl6qf9rtRS7Q3XtzSUAJU9lY3Go3S29nGZJGPpwvuT2Fbuh+ELnUgJrzdbQdQCPmcfTsK2bvX9L8OgWGiQRzXLNtwh+UMfVu5ppXdkJtJXZZu1j0+wgsk4WBPm49B/8ArNcvI/mSs543HNaGqXLsirI+935dvWsyvSxbVKMcOvs7+rPGy9OtKeMl9rReSX+YtJRRXAeuFFFFIAooooAKKKKYBmiiigAppjU9sfSnUUARGH0P50wow7VYooArUVYKg9RTDEOxxQBFRTjEw96QIxPTH1oASpFjJ+9wKeqBfc+tOoAAABwKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKYBT4ppIWzGxFMqJ5ey/nRGTi7xdiJRjNcsldGkNW2gefGrfSq7W+gTt5kljbhj13QAn+RqgTnrSV3xzGulaVn6o8qpk+Fk7wvH0ZpPqEUEPkadCsEf+woUdPSqPnN35+tR0VzVq9Su7zZ3YbCUcNHlpr1fVk4mB6ginhg3Q1VorA6i3RVYOy9DTxMe4FICeiiigAooooAKKKKACiiigBMV12k/wCrP+4K5Kup0aUGIZOCyDFd+H1oVYrsvzPKxmmKw8ntd/ijSooorzT2AooooAY/UVz2r/8AH8f90V0Mnauf1j/j9/4CKAKFFFFABWRfDEx+p/nWvWTqH+uoAqUlT2lnc384hs4Xlc/3R0+p7V2WmeA0tbYX2vyKsS/MVLbYx+J5b6D9aAOT0zSb3VZCtnCWUcNIeFX8f6Cutt9J0bwtAt5qconuMZTIyc/7K/1qrq/jWK1U2egxKqqMCbGAP91e1cZdXctzI011KzsTks7UAb2t+MLvUw0FsPs1t2VT8zD3NYmlzxWurxSXDjHPJ7cdfyzVTLODtyg/vEc/lVnSIkGrQGcjaGzljjJHTn26/hit8PdVY27o5sVZ4efNtZ7eh0+pxkThyc5HAqlirupBxOM/dAwtU66Mdb6xOy6nHll/qdO7voJijFLRXGeiJijFLRTGJijFLRSATFGKWimAmKMUtFACYoxS0UhCYoxS0UwExQcDrTXkC9OTUJcseaAJGk7L+dR9855pM0UDHh2HfP1pwlHcVFmigCwGU9DS1WpQzDoaALFFRCU9xThIp9vrSAfRSA56UtMAooooAMUYoooAKKKKACiiigAooooAKKKQnHWgBaazhevX0pjS9l/Oo6AFZy3Xp6U3FLRSATFGKWigBMUYpaKYCYoxS0UAJijFLRSAtUUlGaYC0UUUgCiiigAooooAKtWt61uNpG5PT0qpRW1KrOlLng7Mwr0KeIh7Oqro2rnxJHpWni9mSSWHeFKqfmGfrxVnTvFui6pgW99Gkhz+7l+RuPr1/CuX11PM8JtkZAmBPtXBtbD+A/ga9Svh411Gps2lseHhcVLDOdG91GTSv2PfaK8Q0/X9Z0cj7HeSrGCD5bHcnp0PtXU6b8UJFCpq1kHwADJbnBP/AAE+3uP8PNnhKkdtT16eOpT30PQ5O1YGsj/TB/u1JZeMdD1IIIr5InPHlz/IQfTng/gag1meHzkk81NjJkNuGD+Nc0oyi7NHbGcZK8XcpE4pYkknfZBG8rf3UUsfyFa3hjVfD8bSNqcdvdYPAYg44HQHg1sXfxW0fTlMOl2K/KMhSR/6CtSUZNp4R1y9I2WTRKf4pmCY/Dr+lasHwxjV/tXiHUEjhXqkR2g/Vj/hXPah8V9auty2iiBT0KKE/U5NcvqGt6nq0m/UbuSX0UuSB+dAHoepeMtA8N25sfDNlHNIv/LYj5AfUdz+lcDq/iDUdbuDLf3Lyeik8L9B2rJaQK2OWY87R1pNrNzIcf7Knj8+9MBS+7/VYY9M54FAT5txO5vU9vpQzqpC9Wxwo60bS3LnH+yD/OgAL5B8rDH1zwKm0q38/WIBMTgOCc9PYY+uBUJcKQOpxwoFSadHLd6nCgfZ83OOw78+vYe5rWj/ABY6X1Rhif4M9baPU6vUt3nqCPlAwD61SzV7U5CZVTHKjk+v+f61Rrqx1vrM7PqcOV3+p07q2n9feGaWkorjPSFopKM0ALRRmikAUhozRTAKKKKYBmjNFRtL2X86QEhYKOTULyFunAphOetFIAxRRmigQUUUUDCiiigAooooAKKTNJQA4HHSnCRh70yjNMCYSg9RinhgehqtS0AWaKrh2HQ08THuKAJaKYJFPfH1p2c9KBC0UlGaAFopCcDmomlJ4Xj3oAkZwvufSoWYt1puaM0DFooopAFFFFABSUtJQAZozSUUALS02loAWikzRQBaoooqgCiiikAZpaSigBaSiigAooopgTXyhvCl1uGcAn9RXn1ehXn/ACKt3/un+Yrz2vpI/wAGn6I+Mf8AvFb/ABMKa0SN1HPqKdRSLvYqXEBWFypyAKgsIGll3nIRevPU+lX5xm3k/wB01QsTM0+yFyqjlu4rz8WrLQ9XAu71Ojs4g0R3ZIz0JOPyq0qhRhQAPQCqlkHMZG8gBuw5NWfJQ/fG/wD3jn+deSe8Hmx84bcR1C84/Kj5n/2F9e9PwAvYAUzzN2PLG4f3s8f/AF6QB8sS8naPc9aQ73yB8i+vc/4Uqpjknc3qe1G/cf3Y3f7WeBQAfJEv90Z7nrQdzZH3V9e5oVMcklj6mk8wv/qcH/aPT/69AC/LEnJwP51JYPNLqUK23yFmxuxzj1xkdOtRqmDk5ZvU/wBPSnWc5TUIWtU3vvAHQBjnpz/n3rSlZVFfuY103SklvZ7nWatMoljUj5tuT7f55qgJFPfH1q7rOzdGR98jJ/z/AJ6Vl124+/1mVzzspt9Shb+tS1RVYMR0OKeJT35rhPUJqKYJVPXinAg9DmgBaKKKYBRRSFgoyTigBaazhfc+lRNKT93gUykA5nLdfypKTNGaQC0UlFAC0UlGaACjNFJQAuaWm0UALRSUUAFFFFABRRRQAUtJRQAuaM0lFABSgkdDikpQCTgc0AOErD3qQMSuQtIkYHLcmpKBFd9xPzU2rdNManqKBlaipjD6H86YY2HbP0oAZS0UlAC5ozSUUALSUUUAFFFFABRRRQAUUUUAW6KqhivQ4qQTEdRmmBNRTBKp9vrTs56UALRRRQAUUUUwCimNIF9zUTOW60gL12QfCl3j0P8AMV5/XfT/APIo3f0P8xXA19LH+DT9EfGP/eK3+JiUUUUixk//AB7yf7pqppt5Lby+XFGj7jySOfzq5LzC/wDumqml3UNrMfMiZ3Y4DLzj8K4MZstD08A7N62Ojtp/MiYLEoOeWLdP8afsc/ekP0UYH+NFtLbtE+1WL5zhRj/61SFjg7UUccFjn9Bj+deQfQEYhTIO3JByC3JH4mhpApxgs391aVlLAgscEdBxj8RzSYSNeAFHsOtIBNhf/W4I/ujp/wDXpS4X5V+ZgPuik+Z/VB+p/wAKX5Y15woz+ZoATYXOZTn/AGR0/wDr0rOAcD5m9B/nik+eTrmNf1P+FL8sa+gz+JP9aAE2b/8AW4b/AGe3/wBeprGdLXVIWUM7BgNq5J5+lQ4d+uUX0HU/4VJZFLXUIJJMbFcHHH58/nWlJ2qRfmY11zUpK19GdRrKKJUk5yw/z/n3rMrS1hMtFL0BXG30rMrsx6tiZaHnZS74KGt/+HCiiiuE9QWgHFJRQBIJGHfP1p4lHcVEqFun51MsYX3PrTAHcqOAT74qAksck5q1TSit1FAFaipjD/dP50wxsO2fpSAZRS0lABRRRmmAUUZozQAUUZozQAUUUUgCiiimAZpKKKAFzRmkooAWikpc0AFFKoLHgVMkYXk8mkAxYy3J4FSgBRxS0UAFFFFABRmiigAzS0lFAAQD1GaYYVPTin0ZoAhMLDpzTCCOoIq1RQBUoqwY1PbH0phh/un86AIqKcY2Ham0AFGaKSmAtFJRQAtFFFIApQSOhxSUUASCVh15p4lU9eKhAJ6c1KsP978qAHl1Azn8qiaUnpwKmwMYxxTTEp7Y+lMCDNGakMJHQ5qMqV6jFIC3K+7wrer3Uf4Vwh6mu5KlvDmohRk7M/lzXDHqa+jpNuhTb7fqfIVYqOJqpd/0QlFFFUIbL/qX/wB01U0s2qTs91JtYcKD0/Orkn+rb6VT0q1W4uiZGUKvRSeSfpXDjLW1PTy+/M7HTWqQCNisqqSeBuzmpNycYDt9Bj+f+FR2dowVyh3cjipDtBIZ1yOoB3H8hXjn0Am9hjCIvPOct/h/Km7Ru3HlvUin71HRGb6kKP60x1Eh+YDH90Zx+PrQA3eWyI/++iOP/r0oQD5m5bH3jQz4OACzeg7UbSWy5z/s9hSATeWH7rBH949P/r0oVVyxPOOWJoL4baAWb27UBCTmQ7jnIGOBQAm5n/1eAv8AeP8ASpLGFP7RiM43Luyc87vY8Hj+lML84Ubj+gpbSLz9QhjuPmRnGV6Aj0H16VpT+NW7mVa3spX7M6rWVbfGd2UC8Dj/AD6Vl1qayzh4158sDg88n/P9ay67cfb6zKx5uU3+pQv/AFqFFJUiRluTwK4T1BoBJwKlSLu35U9VCjilzSAXp0ooooAKKSigAozRRQAEA9RmmGJT04p9FAEDRMOnNMII6jFWqTr1pgVaKsGNT2x9KYYT2NAEVFOKMvUU2mAUUUUgFpKKKACiiigAoopyqWPFADakSInluBUiRhfc+tPoAQAAYHFLRRSAKKKKACiiigAooooAKKKKACiiigAooooAM0ZoooAWmsA3UZpaKAIzCp6cUwxMOnNT0UwKpBHUYpKtU0xqe1AEFFFOVC3Tp60ANp6xluvAqRUC+59afQAigL0FLRRSAWikozQAtFJmjNAF21Uf2bd8fwH+Rrzk/eP1r0e0/wCQbd/7h/ka84b7x+tfSUf92p/P8z4+t/vlb1X5CUUUUwEf7jfSs/T7N7y54HyIcsT/ACrRb7p+lZlksz3YS3d0OckqelcWL2R6OBtzO6udRa2zhX3jcnHGcj8qshDtzjC46ngVWtGuArK7uAMdhz+NTbAWBI3N2LHJ/M14x9EO3R933f7g3fr0/WmEFjxlV9O5/wAKkKbceYQmTgbiBn86iYk8REHnliDj/wCvQAfJEvZRnoB1pMM/3soPQHk0oRUyxPOOWbrik3F/9Xwv94jr9KQDuEXH5Ad6bhnHzZUf3QeT+NLtVMscD1Ymkyz/AHPlX+8RyfoKAFyqYUYHoopIFe5vI43by13jO3qo9fwpfljUk8epPeoxOwkV4xtKnKt3zVRaTTZMk3FpbnX6020QxBcBR1A4+lZiqWPFXINf0q4sYo9RlxN90jy2YD0+YD06mrJl0Tyx/p1tGD0zMB/M17OIoLE1HVpzWvnqfN4TFPBUlQrUpJq+qV0/Mz0jC9eTUlaa6bbypuim3A9CGBFN/shu0uf+A/8A165nl2JWyv8ANHZHOMG95W9UzOoq4dLnGcFD+JqM2FyD/q//AB4VjLCYiO8H9x0xzDCT2qL7yvRUhtpgceU//fJqMgg4IwawlCUfiVjqjUhP4WmFFFFQaBRSZozTAWikpaQBRRRQAUUUUAFNKK3UU6igCIw/3T+dMMbDtn6VYpKYFWirRAPUZphhU9OKAIKKeybTyRUqKo6HJ9aQDEizy35VKBgYFLRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFJRmmAtFJS0gCiiigAopKjeXsv50APZwnXr6VCZWJ4OKbnPWkoAnSLH3ufapKKKACiikpgLRSZooAWiiikAUUUUAXrT/kG3f+4f5GvOG+8frXpFsCul3RYYBQ4/KvOG+8a+kpaYan6P8AM+OrNPGVrd1+Q2iiimMDyDWbZXNxBcbLbGWPQjNaVZ1jc/ZLksIRKzYA55H0rixfwrQ9DAu0nrY6ezupDvXameMkgn9Kn3Of42Hsvy/yqGzuUcODCSxA7jA/Gpt7fwqi/wDjx/w/SvGPoxFiGfkXn2HWmsdrbcZb09Pr6U5gzDDMzD0J4/LpTSQi9PoAKAECEnMhDHqBjgUF/m2qCx7+g/GkIZwN2UH90Hn86UlUAUD6KKQAFPVjuP04FBcniMbj0JJ4FIVLjEnA/ug/zpSwTAAyccKopgAQbtx5b1Pb6VFI6scRjnu3b/69SbC3+sOf9kHj/wCvUNxJGjbVOZP7q9vf2oAhJG47RgDiigdKKQCMqsMMoI9xVhb27RAkd3cIo6BJWGPyNQUVSlKOzJlCMviVzQi13VIYwkd7Jgd2wx/Mg1Yh8U6pEhDyRzH+9JGP/ZcVkAFjgDJqZLUn75x7Ct44qvHab+85Z4HCz+KmvuRtxeMbkL+/tYpW7bCUH9a0rbxFaXa7b63aDjqRvX6cc/pXMpGqfdHPrT63jmGIWjd15o5Z5Pg5O8Y8r7ps7UaXBdQtNaMHjU4LwuHUH0yM889KpTaZPHnbhwPTg1zMUkkEolgkeKVQQskbFWXPBwRyK1I/EuppgSypcKMD97GM47/MMEn3JNX7fDVf4kOXzj/kZfVMdQ1o1eZdpf5k7KyNhgQfQ02tGw1Wz1VkguPLgnc4VZGwrH2Y8ewB5J4FSX+iS2sjKEZWXqjDmlLB8y5qEuZfj9xUMxcJKGKg4PvuvvMqloIKsQRgikrhPVTvqhc0UlFIY6im0UgHUUlFABRSUjOF+vpTAWo2l7L+dMZy3X8qbQAuc9aM46UlFIB4lYd8/WniYdxioaKALIdW6GnVUpwdl6GgCzRUIm/vD8qeJVPfH1oAfSUZpKYC0ZpKKAFopKKAFpabRQA6kpKKACiiimAUtJRQAuaRnCjmmPLjhefeoicnJpAOeQt7D0plFFIAooooAtUtFFACUYpaKYCYoxS0UAJRS0UAABJAHJPQCtmw0fzGBkwSOSScKo9Sf89Kg0izM0m/aSc4QYrJ8R6k8961pGWSKE4YAkbj6H26V6NOMMPSVaavJ7L9WeNWlVxld4elLljH4mt79l+pa8Q65ZDTp7HT907spV7gfKi9MhR1bPIzx2xnNcC33jWvN/qH/wB01jM3zGurD1p1ouU3rc48VhaWGkoUlZW/zCijNFdJyBWfZS28F2XuVY4+7joK0KzrJYGvibqQIq8gHua48X8KPQwN+d2OotHtmDNyCQMAKc/lUxYDO2MnB6sQoP5ZqK0hhw8iSqFwOS3H51KSgz824/7Az+p4rxT6MC787Sq88bV5/XNMYgEu557sxp5fGdsYHoWbP6D/ABpjDewZwDjkDHAoATczY2cL/eP9BQFWME9PUnqaN+SQnzHuewoVMHcx3N6nt9BSAT5pB8uUX1xyaGaOCPLnavv3/wAahlvACVgAdh1Y/dH+NV8F23uxd/U9voKAJJLmSXIjzGmOv8R/w/n9KjVQq4Axzn606jBzjHPpQAUVIsEjfw4+tSLa/wB5vwFAFepo7dm5b5R+tTpEidBz6mn0wGoioMKMU6iigAooooAM0ZoooAM1saf4kvrRI7aedp7NSB5UnzbABgbT1AHoOKx6KqMpQlzRdmRUpwqRcJq6Z3F/YCe3S4iwd67kcdHFYmCOta3gi7N/DNokrZbBltc4+93XJ7H+eKg1S28mbzFHDHB+tejWtiaXt4/Evi/zPFw7lgsR9Vk7wlrF/mv8ihijFGaWvOPaExRilpKQBSHgZJpryBfc+lQsxbrQA9pc8L+dR0lIWA6kCgB1FRmZR0yfpTftHov60ATUVB57egpROe4oAmoqMTKeuRTwwboc0ALRRRQAUUmaKAHAkdDiniUjrzUeaM0wJhKp68U8EHoQaq0uSOlAFrFGKriVx3z9aeJx3GKAJcUU0OG6GloAWikopALRSUx5QvA5NMB7EKMk1C0hb2FMLEnJOaM0AFFGaM0AFFGaKQBRRmkzQBapc0lFAC5ozSUUALmikooAdQql3Cr1Y4FJVrTU33q552gmtaNP2lSMO7OfEVfY0ZVOybO98GaaE3XW35YhtT3J6n8v/Qq8j1C6e91O5upF2PNM0hX+7kk4/DpXu2ggWelQxuBhxvJA5ye/5YrmfFPw3g1JnvtCdYLmRtzQscRv64wOD+n0rTGVVUru2y0XojLLqLpYaN93q/Vnkc3+of8A3TWK/wB810d9YXNjM9tfW8kEoHzJIpU49f8A69Ys1k4clDkduK2wtaEIuMmY4/D1KklKCvoVc0uaGRl+8p+opn0NenGSaumeNKEou0lYfuqhZ232u/KllVVOWz3q7kis6C1e6vjGg75J9K5cU/dOzBL33pc6qzsiJiyYOFwOMVZK7SA7KpPYnn8qp2dvLGxUFigXorGraRnGI0wPYV4x9GG5BjAdsnsMY/P/AApjDf1G1f7oOc/U0PNBHxJOmc4Kqdx/IZqtNdFuLfKr/fdefwH+P5UgJpZ44AN3J/hRep/Cqckkk4Ik4T+4p/maI4iWO0Ek8knkn61YS16bzj2FAFcLyAPoAKmS3ZvvfKKsqioPlGKdTAYkKJ0GT6mn0UUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAdH4B2HxvYLL0JfH1CEj9QK7HxjYCO+l2DidPMAGevf8AUZ/GuO8AIreNrBnbaqFj+O0gfqRXoHjOYG/tYe4jZj+JH+Fd+AlavyPaSseTm8G8N7SO8GpL5HnFFSTJ5c7r6MageQLwOTXHJOLaZ6MZKcVJdR5OBkmoXlJ4XgetQyTjPJyfQVC0zHpwPapKJ2cL941EZ/7o/OoqKAHGRj1NNoooAKKKKBhRRRQAUZoooAcJWHQ/nTxP/eH5VFRQImM47LSef/s/rUVFAEhnPYCjz29BUdFAEnnt6ClE57ioqTNAFgTjuCKcJFPRhVWimBcpwdl6GqQdl6E1Is5H3hmgC4Jv7w/KnrIrdP5VWjZX6naPerS4C/L0pARPITwOBUdWiAeozTDCp6cUwIKKkMTDpzTCCOoxQAlFFFMAooopAFFFFAEgmYdeaeJVPXioKKALQOenNFVgSOnFPErDrzQBNRTBKp68U8EHoc0AFaGkf66T/drPrR0j/XSf7tduX/7zA8vNv9xqen6o7DxZ4mm8NaHZXMECTSTSLFhzgAbST/LFWfC/jaw19QkDeReYy9q5ycexxhvw5rnfieufCenN/duVH/kNq8tVmRgyEqynIIOCDXDLdnqQ+FH0Xq+h6Z4ktTBfwAvjCSqAJI/91sfpXlHibwNqPh9mmRTd2WSRNGpJQf7Yxx9elX/C/wASpIitr4hZpFzhboDlR/tAdfr1r1G01C3vrVXV47iCVeHXDBgfX1qdiz52aJW6iq0thG/OMH2r2LxT8Nre9V7zw+EgnJybfIWJh/s8fKf0+leX3VpcWNy9veQvDMhwyOuCKuMnHVMmUIzVpK5hvYOv3GzUdpp5imdwXZyedpwBWzjNLWk605rlkzGnhqVOXNFFMTXMAIWQrkfe2gkVGyy3H+sMkoP98kr+XStCisToKaWj4A+VB6CpltkH3iWqaigBAABgDA9qWiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKK3/CvhW68SX4CqyWiHMsx6Aeg96AOs+Gvh2OOM63dkbhkRg9FGBzn8/px61Dq+oDUfEUky/cJKR/7o/zn8a6DxBeQadpI0fTcRxRoFcr/Co6j/GuJtpxc6u7r9xVJA/GunBJvEw9Tz8yaWDqX7MydTudt9Mi9mrPaRm6nj0qzfDzb6Z1PVzVQgr1FRWadSTXdm2GTjRgn2X5BRSZozWRuLRRRQAUUUUAFFFFABRSUUDFopM0ZoELRRRQAUUUUAFJmg0UAFFFFAwoopVQt7CgQgGTxUix4+9yaeFC9KWgYUAkdDj6UUUASLOw6/MKsI4dcrVOnxyGM8DOaBFujr1pkcwkOMEGn5piGmNT2x9KYYT2OakpaAKxVl6ikq1mmsqnqKQFeipTD6H86b5belADKKKKBhRRRQAUZx0oooAeJWHv9a09HnBumTGCy1k1f0f/AJCK/wC6a68E2sRC3c8/MoqWDqJ9mdX8R4vM8EQuF3eVNG2fTIK5/WvJK9h8aK158PZCgPyqkjY/2SCf5GvHq5ZpqTTO+lJSpxa7IK2vD/ifUPD9xutJN0Tfehckqf8AA+9YtFSaHu3hvxrY6zGBFJ5U38UEhGT7j1HvWrrOg6X4ntPLvYh5gGI5V4kT6HuPavnq3uJIJVeJ2jdTlWU4I/Gu/wDDfxClg2W+qksvAWZf/Zv8RSsBleJvBuoeG5WkkXz7IthLleh9iOoP6Vz1fQVjrFrqNuPnSWOQYyCGDD+tcb4n+GUUytd+G9sb4ybQt8rnPVWJ4+h4+lFwPL6KlubaazuZLe6jaKaNtrowwQaipgFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUoBZgFBJJwAO9b2ieC9Z1xkaC1aG3Y8zzDauPbufwr07SPDeh+FIVYILi9A+aVwGfPt2Ufr9aLgcH4Z+H19qzi41NWs7MHncMO/0HYe9d5cX1ro2nLp+jokUUa4Mi9B6/j71FrOukxnzn8uLtGvVv8AGuE1bVpLxMf6qFT90Hr9akBus6sJmZIW/dj7z/3qj0slNKubpfvnOPbA4/r+dYv7y+u47e3XLSOFQepNd0dIi0zSYIEGWxiRuzMRya9DLmvrCT3aaXrY8nOE/qjktk02u6ucPSU+VPLmdP7rEU2uN6OzPSi01dDCoPUU0xeh/OpaKQyuUYdqTNWaayg9RQBBmlp5iHY4ppRh2z9KAEpKPrRQAUUUUDCiiigQUUUUAGaKKKACiiigAo69KVVLfSpVUL0oAasfdvyqSiigAoopaAEoooAJOByaACnxxF+TwPWpI4McvyfSpqYCKoQYUUtFFAgooopAFFFFABRRRQBWooooGFFFFABRRRQAVf0f/kIr/umqSoW6fnWjpKBb4eu0811YP/eIeqODMf8AdKnoz0P7GNS8ITWZ/wCW9syfmCK8HByAa+gNEYHT41P92vFPEtj/AGb4m1C1AwqzMyY6bW+YfocVjW/jT9X+Z04X/d6fovyMuiiisjpCpI3x1qOpbe2nu5hDawyTyt0SNCzH8BQBt6N4gvNJkHkSFo88xseD/hXrvhnXLjUtNjukTMTEghm6Eda8/wBC+GmoXSrNrEwsozyIlw0h+vZf1+lekWFpa6Tp8VnZrtiiXA9T6kn1NJgP13QtJ8TW4ivo9s6qRFMMhoz9eh+hrz3U/hZrNr82nSw36dgD5bn8GOP1r0CS9QjaACfVTTor1k/1cpHsaQHiN9pd/pkmzULOa2bt5iEA/Q9+lVK99uZo7+1a3voFmjbr/npWBL4A8L3OSDdW7nn5ZAMfhgincDyGivRNV+FvloZdGv2uBniJ0BP/AH0D/Sual8D+I4mI/sqZ/dcH+tMDAoraufCGvWiK1xpkyhl3DBDcfgePpVa28P6vdvtttOuXPtGaAM6iugHgXxKRkaVL+LqP61Onw68TscHTlT3a4j/o1AHMUV2UPwu8QygbvssRPZ5Se3sDVmP4T6wyAy3drCe4Ytj8wKAOEor0u0+FdlE27VdZDDP3IAAfzOf5VuWvhDwjYKv+im6ded8hZyfwOF/Si4HjUcbzSLHEjO7HCqoySfpWxa+D/EF5t8nSbkBuhkTyx/49ivZYLixsk2afp8UA9FUIPyUe1QXWuSJnEqR+y8E/1pXA4jTvhReMqyazfRWid0j+ZvzOB/Oup07w14a0Eh7a0+2XA6STfMR9MjA/AVBPrKMxYb5Wz1Y1nXOqysCWdYUz1Bx+tIDo7zW3CnfKsCY4VOp/rXNXmtMx22y7f9tuv5Vh3WuQRswj3TP6jp+dYN5qk9wCZXCJ/dXgf/XoA0r/AFFVmcl/Nc9TnP61hyyy3dwsMStJI5wkajkmptO0y/1ubZYxYjH3pn4Rfx7n2Fd9ovh2z0VN0QMtwww07/ePsPQUAV/DnhxNIi8+4xJeuPmbsg/uj/GruryBY4o+7Pn8AP8A69aFczf3wudcVEOUjUqPc9zXVg/95h6o4Mx/3Or/AIWcxqP/ACEZ/wDfqtVnUf8AkIz/AO+arUq/8WXqy8L/ALvD0X5BRRRWJ0hRSZozQAUUUUAIQD1FIIA54IX6mnUUARvbunXH51GRjrVik69aAIKKlMan2ppiPbmgBlFBBHUUUAFFFKqlulACU9Y+7flT1QL9fWnUAJS0UUAFFFFABRRU0cHd/wAqAI0jaQ8cD1qyiBBx+dOAx0ooEFFFFAwooooAKKKKACiiigAooooArUmaKKAFzRSU9Iy3J4FACAZOBUixd2/KnqoUcUtAgq5pf/H8Poap1YsGK30WO5xXThXy14PzRx46LlhaiXZ/keiaPJthjHqtcN8SdEmS6TVYkZoseXKQM7eflJ+ucZ+nrXUaXepJCADh4nKOM9OBj9CDXQAW97DsuI0lQjDI6hlI9CD1FZV9K0/V/mb4T/d6fovyPnqrNlp15qMojsbaWds4+RcgfU9BXrEngHRjqDTR2ipGTkJvYr+Wf0roba1tNPtxDbRRxRr0CqFH5Vjc6TzrSfhlcS7ZNWuBGvUxQ8n6Fug/WvQ9L0rTtEtRDptrHDx8zKMs31bqaWS8UDCDP6CqUt2X6nPsOlIDQmvQOAdx9B0qhNdl+Cc+w6VVZ2bqaZ0oAl85w2QaeLr++v5VSnuordcyHFRRajbTHCyAH3oA2EuV/hfb7GpRcsBzhqywUYZBDfjSF2X7pI+lAGn9rXOSCPpT01Fg3F1Ig9ATWP8AaHAOSD9RURuiM5UfnQB0H9szKfluj+PNKNZm/wCfpR+ArmjdjPKfrTTeLn7p/OgDpjrMx/5ewB7Ypra1J3vH/AmuZ+2jsh/OmG+9I/1oA6R9Yz1uZW/E1A2px+jt9a583zdkH50xr2XttH0FAG+2qH+CL8SaifU7g/d2p9B/jWA11M3Vz+HFVZbyNc+bOPxagDdm1En/AFtz+G7+gqjJqUS/dDN+lYU+q26/dLP9Bj+dRRz3t/xp9nJKM43KpYD8elAGnPq0xB8sLH9OTWRdXe4lpZdx6ZJzitS38Jatec308dqv90He3X0HH61u2PhHSrNg7xNdSA5Dztux+HT9KAOJs7O+1Vium2rSr0MjfKg/E102meBoImWbVpjdSDnyk4jH9T+n0rqwAAABgDoKWgBqIkcYSNVRFGAqjAFLQTisvU9XW2QpCVMnQseiUAJrepizgMcTDzWHJ/uD1+tcfplx5+uAj7oVgKr6pqJuJHVGLbjlmPU1X0K4H/CRQwL1KOx9uK6sH/vEPVHDmP8AudX/AAssaif+JlP/AL5qrmrOo/8AISn/AN+q1Kv/ABZer/MrC/7vD0X5BRmilCk9OaxOkSigjHWigAooooAKKKKACiigAk4HNABRUq27H7xxSNAy5IwRQBHTSintilLBetRM5b2FABhQ3UkVIGXoDUNFAFiioA5XoaUS+o/KgCaimB1PenUALQMZ+Y4HtSZooAtR+WPuEZ9+tSVSpyyMvQ0AW6KgW4/vD8qkEyN0OPrQA+jNJRQAtFJRQAtFGaKACiiigAozQaSgCtSqpY8VIkXdvyqUDAwKAGJGF5PJp9FFABRRRQAVPZf8fsX+9UFT2X/H7F/vVtQ/ix9V+ZzYr/d5+j/IlbWf7I8VfvSfs0wQSf7P+1+Fd5HchUEkbhkIyGU9R615V4o/5DDf7i/ypuj+LLvSkNtMWuLUjAQnlPof6VWMX+0T9WTl3+50v8KPW21A7fvnH0FQtdGTpz7k1z9hr9hqVogtbhWlBAaNjtYcc8Hr26Vcs7jc5UnrXIdxfLFuppDRQaAEooooAyNRieWYheucA+lWV0q1EQQoTgY3Fjk+9XPJWR+etWWsnYBodp3dUzgj8+ooAw5dNMQ3W07qR0VjxWa+s3FrKyTn7o7jOa6K6VokxIjKcZwwxXK32J4mBGfQ+lAFhPEkRX515PYUHX7YnDBhXOfZLxpNkcDyd8opPFXIPDmp3GN1uyA93YL+nWgDTOv2e8gsRz6Gof8AhIbInhifwNTW/gp3/wCPmZV9kUt/PFa1t4P0yLBkSSX2ZgB+nP60AYDeIbYZCozemKSPWXnbbbWc0p9FTP8AKu0g0jTrYDybKEY7su4/mc1cA2qFXhR0A6CgDhAdamYiPSp1/wB9Cv8AOpF0jxFcHmJIAf70i/05ruKKAONXwfqU/wDx938Sj/ZDP/PFXYPBNkhBuLieb1AIUH+v610tFAGda6BpdntMNlGWU5DuN7A+uTmtAAAAAYA6AUtIPmYBeSTgAd6AFoqZbO6dCwgYKOpf5R+tDW8aZ825Un+7Epb9TgfzoAgJA61DPdRwLmRgvt3P4Ul5eWlujB2CD1L5b6elcjqOv24ZhZx7mP8AGxzQBpanrYEbKG8pfr8x/wAK5G91J7klEO1BVe4neQlnbNZt1qEVnBvkOWPRR1NAD7y7S0gLueew9TTPBkzT+Klkfq0b/wAq5q5vJbybfKeB0UdBXQ+Bf+Rkj/65v/KuvB/7xD1Rw5j/ALnV/wALOh1L/kJT/wC+aq5qzqX/ACEp/wDfqrU1/wCLL1f5l4X/AHeHovyFzRmkorE6BwcjjqPQ0uVPUFfpTKKAJfKJGUIYe1N2MOqn8qarFTlTg1ail3j0PpQBX2t6H8qBG56Kfxq3RQBAtv8A3z+AqZVC/dGKWkLBRk0ALUckgwQOaY8hbgcCmUAQPGwOfvCo6t0hUN1ANAFWipzCh9vxpPJX1NAEFKBnpU4hUepp4UL0GKAI0h7v+VLLkIMcYNSU113IRQBCJT3GacJFPt9ajKMvUU2gCxmlzVcEjpxThIR15oAmpKaJAevFOBB6c0AKGK9CR9KkWdh15qKigC0s6tweDUlUauqdyg+ooAWiijp1oAKKjadB3z9Kb9p9F/WgCaioRcDupH0pwnQ98fhQBLRRRQAUUUUAFFFFABUts+y6jY9mFRU6P/Wp/vCrptxmmjKrFSpyi+qZR8UgjVmyOsYxXOSferq/FygXcJxyYzk/jXKy9RXRjo8uJmvM5srlzYKm/L8tCEsyNuUkEHIIrX0/xZqNi/zP568cSdfzrGfrTDXGeien6Z4/0y72peB7SQkDLfMv5j/Cujtry2vYhJaTxzKRnKNmvDKlgup7WTfbTSROOjIxBpWA90pK8u0/x9qtphbkpdJn+MYP5iup0/x/pd3tW6D2rnA+YZXP1FFgOpBIPFWI7wqMOM1Qtry2vIw9rPHMuM5RgamNIDSS5jkXaSCD1VhkfkafHBZb932OAn1WNf8ACsmgTuhwGP40AdMk9oy7HClf7rrx/hTja6fNkrGgJ4zG3T8K5gX8gPzc00amMfMtAHTNpEBX5HlU+pAIqP8Ashf+fg/9+/8A69c+uqovIyp9iacdaG7iR/8Avo0Abp0gdrgf98f/AF6QaQ3eYfgprDOuekrj/gZpG18FcGWQ+oLGgDfGjZ/5bN+Ef/16a2khfvXAX/eXH9a55tcjxg7m+pP+FQ/24in5Ysn3oA6cWVkq/vLoZ7lXH8uabs02IfM0k2egAOf6Vyja7KTlUGfXFVpdau3Y4bGfT/61AHZtc2UKfurMNnvI3T88/wA6rza8sMWBJDEB/DGvH5muJmvLh2+aVvwqrKxLDcST3yaAOmvPFEO4nc0rDuTmsG78TXUwYR4ReaznOOKzr++t7KOQzSqrKcMm4bh+FAD5riafLTyM5HTJqnNNHDHulYKPesW78VIFKWsO7nIZj/SsC4vJ7tt00hb27UAbeo6+mNlp8x7k9B/jWDJK80heRsk1HS0wJErqPA3/ACMkf/XNv5Vyqn5hXVeBv+RkT/rm38q6sH/vEPVHDmP+51f8LN29fzb6Zx0LmoafN/rn/wB4/wA6ZWVRtzbZtSio04xXRISlpKKg1FoozRmgAoUlWBHWiloAtq25QRS1DA+AwJ460PKTwvAoAe8oXgcmoCxY5NFFABRRRQAUlLRQAlFLRQAlFLRQAlFLRQAU0xq3UflTqKAIjB6H8xUZicds/SrOaXNAFOkzjpVxgG6jNRmFT04oAhEjD3+tOEg7jFBhYe/0phBHXigCYEHoamScqoBXIFUqcHYd6ANATpjPP0xULyM/sPSoBL6j8qcGB6GgB1FJnA5qN5M8LwKAHM+OByaiJyeaSigDTE394flTwyt0NVqKALVFVxIy9/zp4nH8Q/KgCWimhg3Q5p2aACnRnEi59RTaKadncTV1YZ4u5uIP+uZ/nXKS9q6/xSgNraSH72CufXiuQl7V2Zhf6zJ97fkedlDX1KC7XX4srv1phqR+tRmuE9USiiigBDRSmkoAlgup7aQPbzPEw6FGxiuhsfHmr2mFnZLpOc+YvP5iuZopAem2HxB0y5ULdo9q/Gc/MufqK3oL61vG3WlxFMo4JjcHn8K8ToyR0JH0oA9wbvVdvu15Rb+J9Zs2LRahM2RjEp3gfg2avR+PNVSEJIsErD+MqQf0OP0osB6I3ao26muBHxA1Hjdb2549D/jTf+E+1A/egt8+yn/GiwHemmgZ5NcGfHt+R/x7QZ9ef8aiPjnUsfdj/KkB3zngmmI2c155N4w1WUECRVznooqq/iPVHZWF1IhXpsYimB6cTgEnsCaqtqNlFuMlzF8oJIDA4xXl0t/dStmSdyfXOKhZ2c5dix9Sc0Aeh3HinTICT53mHnhefT/H9DWLd+NdxP2WAj0LVyhppoA0LrXr+6zvmI9dv+fesx3Z2y7Fj0yTmlPSmGgBppRRjNLQAYooooAcn3xXZfD+ASazLKesUPA+p6/59a49I22iQjC5wCe9dt8O/wDkIXf/AFyH867cAk8VC/c8zNm1gajXb9S5N/rn/wB4/wA6jpsshFxJ3+Y/zpBID14rml8TO2HwofRQCD0oqSwooozjrSAKRpAvuaY0meF/Oo6AJomLSc+lTVDAOSfwqamAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUhAPUZpaKAK8oCthRio6svGH9j61C0TL2z9KAGU5VLdOlOWP+9x7VJ06UgE2Dbg800xehp9SJCzHkYHqaAK3lv2Un6U0jBweK1FQIMKKhmjLSZAzxTAbmikooAWkoooAKcJGHfP1ptFAEwmB6jFPDA9DmqpIUZY4FVpbrtFx71E6kYLU6cPhauIlaC+fQ2vEa+bo9tKvRWGfxFcbL0H1rsJD9r8G5z80Rycn0P8AhXITDj8a9DGvnlCovtRTPGy6m6KqYd/YnJfiVn7Uw0+TtUZrhPTEooooAKSiigAooooASkNLSGgCNqYae9NNADBSHrS0h60AJSGlpDSAKKKKAENFBooAKaadTTQA1qaac3SkxnrxQA0UUo6N9KsWdhNeviMYQdXPQUAV1UswVQST0ArUg0yO3hFzqR2qOkfc+1TCax0lMQ4uJ+5HIH41mXV1JdTb5n3EdAOgoAfc3LXM2SoRVGEQdFFdn8P4tkd/dHOFQKAPxJ/pXCr1r0DwrE9l4QvLo9ZyfL544G0frmu/L9K/P2Tf4HlZt72G9mvtNL72V2Yu5Y9Sc02qnnTxf6xMj1/+vUqXcbdcqfevPVWLep7ssHVirxV15ak9OEjD3pikN90g/SpFj7t+VWclmtBwfI6VGxYn5uKlooAgpQMnAqQxg9sfSpo7YoN3U+npQAiLtUCnUEFeoxRQAUUUUAFFFFABRRRQAUUmaM0ALRRRQAUUUUAFFFFABRSUUALRSUZoAWiinpGW9h60ANALHAqZIwvJ5NOVQowKWgAIB6jNN2J/dX8qdRQAgUDoAPoKWiigAooooAq0UmaWgAooprOqLljgUXtuOMXJ2W46opbhY+PvN6CoJbpm4T5R696r1yVMR0ge/hMob9+v93+Y+SVpDlj9B6U0c9OadHC8vQYHqauRQrF05PrWMKU6juz0cRjaGDjyQ3XRfqaugAyWNxZz8K4JAzzgjB/z71yd9A9tcvBKMMjEH3963Fvm04/aVBbZ1UHG72q3rOmprlgmo6dhpNvK/wB8en1HNfRQp+3wqhH4ofiv+AfnVfEPD5hKtU0hV69FL/gnFPUZqWQFWIYYI4IPaoq809cSiikoAKKM0UAFFFFIBKSlptMBr9aYelOftTaAI6DRQaAEpKdTaQBRS9qSgBDRQetFAC4pp68UtNPWgBrcYpp6U5qaTxQAK+3sG9jU0t9cyx7C5WPoEThaq0pOetADs8YoFJTkBZgqgkk4AA60AWrC0kvbyK2gGZJWCr149zjsOtek6qI7DSbXToDwqgHkZIHr9T+oNZ/h3RU8O6e+q6r8s7L8id0Hp9Tx/nNZ66hLf3c0k5yzHcB6D0/lXqOP1Wg1L45/gv8AgnhqX17FqUf4dPr3l/wCWmm2SX+AD3HFTLHnlvyqTp0rzGk9z24zlB3i7FM2LRndBIc+/FJ51zD/AKxdw9cf1q7RWfs0vhdjsWMlLSrFS/P7yul7G33soferCES/6shvpS/YEl5kQL9ODTDphj5t5CD78frRepHfUfLhanwtxfnqi5HCE5PJ/lUlZ/nXlv8A6xN6+uM/qKli1CKThgUP5imqseuhEsFVteHvLy1LdNMantj6UqurjKsCPY0tabnI007MiMJ/hP50wqV6jFWKKYitRU5jVu35Uww/3T+dAEdJTijDqKbSAKKKKACiiigAzS0lFAC0UlFABRRRQAUUqqWPFTpGE9z60AMSLu35VLRRQAuaKSimAtFJRmgBaKM0ZoADSUUUgKtFMeRYxlj+FVJbhpOB8q+nrWVSrGG+53YXA1cS7rRdyeW5CcJ8zfoKqNIztlzmm1LFbtJz0X1rilOdV2PpqWHw2BhzP73v/XkhgBY4AyfarMVr3l/Kpo4ljGFH1PrT66aeHS1keNi82nU92jou/X/gCAY6UtFFdJ4hR1U/6Ko77/6GoNJ1i40qbMRDRt99COvvUmrDMcf1NZdVCpKnLmg7MzqUoVoOFRXTOtmstL8TIZrSQW12eSpxycdx/np71z2oeHdR08sZITJGDw6c98dKqKzKwZGKkdCDityx8V3tthLnbcR9933sZ9a7vbYfEfxlyy7rb5o8v2GMwn+7y54/yvdej/zOYoruDdeH9cwtzELeZv4iNpyT6j3/AJ1Su/BD7TJpt0sq4yFfqfTke3tSlgZtc1Fqa8t/uNIZrST5cRF0357fecpRV280e/sCftNs6qD94DK/mKo5rhlGUHaSsepCpCpHmg7ryFopM0VJYU3vTqaetADZO1NpzU2kBFQaKDTASkpaSkAUUUlAAaKKQ0ALmmE80ppDQA1qaelKxwOaYWGKAEFLQivI4WNWdj0VRk11ej+Ab69xJqDfY4fQ8uevb8q2o0KlZ2pq5zYjFUcNHmqysc1bW095OsNtG0kjdFWvQNG8NWfhuIahrLq9yB8iDkKfb37f/rqcXuj+GIWt9Ji82f8AiY5Pfufy6e3Wucvb641Cfzbpyzdh2H0rvSo4PV+9P8F/meVJ4jMdEnCl/wCTS/yRPq+rzatdb3+WJfuR+n/16j01gt1g9WUgVUqxYnF5GeTzjj34rzZ1JVJOcnds9ilThSgoU1ZI2aUDPSpUtyfvnHsKmVFT7oxSLIUgJ+9wPSpljVPuj8adRTAKKKKQBUbwRPncgye461JRQ0noyoTlB3i7MqNZYOYnIPvTd91D94bh+dXaKy9il8Lsd8cwqNWrJTXmtfvKyXyH74K/rU6yK/3GB+lI8Mcn3kB96rvY94nIPvSvVj5jtgq2zcH96/zLdFU911D94bx+dOS+Q8OpU/nTVaOz0Ill9a3NTtJeTuWqaUVuooSRJPuMD9KdWt77HDKLi7SViIw/3T+dRlGXqKs0UCKtFWSqt1FRmH+6fzoAiopxjYdvyptABRRQBk4FABT0jLcngU9IsctyfSpKAEAAGBxS0UUAFFFFMAooooAKKKKACiiigAooooApsiv95Qage0U/cbH1qzRWcqcZbo6aOLr0f4crFEwyRNnaGx+NSpdr0dcfSrNMeNH+8oPvWSpSh8DO2WOp4iyxML+a/wAhVkV/usDTqrNaDrG2PrTd1xF94bh+dP2ko/GiPqdGr/AqL0ejLdFV0u1P3wVqzEBLypBHrWkakZbM5K2FrUf4kbFDUhuVB9f6VkkYPNbephQY0A5AJJ/z9KypEzyKpmBDRS4opAJirFtqF3ZHNtcPH7A8fl0qvS1UZSi7xdiZQjNcsldHRWvjG4TC3sCTD+8vBqyy+HNc4I+zTHuBsbp+R/8ArVymKTFd0cfVty1LSXmeXPKqN+ei3CXk/wBDdvPA9zHlrGdJl4+VuDWDc6VfWeftNrKgHVtuR+Y47VctNVvrEj7PcOqj+EnI/I1tW3jKUKBeWyy4GNynH6VX+xVe8H96JvmVDtUX3P8AyONpO9d2f+Ea1nh0W3lbAGfkP59D6f5FU7jwLuG7T71XBGVWQdvqPw7UngKjV6TUl5MqObUU+WunB+a/U45uaYa2Lnwxq9sCWtGcDvGQ2fw6/pWVPbz2zbbiGSJvSRSp/WuOdKpD44tHpU69Kr/Dkn6MhpDRTT1rI2FpKQmkpgONJ0oVGkcLGpZj0CjJNWxouqMwA066yTjmFh/SnGEpbIiVSEPidinkU3eK2ovB+tzYxZ7QwzlpF4+ozmtKD4eX7N/pV1BEvqgLH8jiuiODxE9oP8jjqZjhKfxVF99/yORL+gpjMT3r0OHwNo9iofU70yf7zhFPT8f171aW58MaOR9jto5JB0aNAzD/AIEfp69q6P7PlHWrNR+epyf2vCbth6cp/Ky+84Cw0HU9TZfsdnIyk/fI2r69TXT6d8OnAWXV7tY0ADNHHz9QSfar914xuHBWygSFeOW5NYt3qV5ff8fU7SD06D8h9Kd8FR7zf3IVsyxG9qa+9/5HRxXPh7w4pXToFlnXgsOTkcfeP1NY+o+Ir7UcqX8qP+5GcfrWVS1jVxtWpHkj7sey0N6GW0KUvaSvOXeWoUUuKK4j0gxVmxH+lw/74/nVcc1ash/pUX++P50wOjooopiCiiigAopKKAFopKM0ALRRRQAUUUUAFMeOOT76A+9OooaT3KjKUHeLsyq1kvWNip96bm6g/wBtfzq5RWLox3jod0cwqtctVKa81+pWS+XpIpX6VOkqSfcYGh40f76g1A9kp5jYqffmj97HzK/2Kt3g/vX+Zaoqli6h6fOPzpyXw6SIQfahVo7S0Jll9VrmpNTXk/0LdIVDdRTUmjk+44PtSPLjheT61smnscMoSg7SVmNdFXvj2p0bIBxwfeoSc9aKCS1RVUMV6GniY9xmgCeimCVT3x9aeDmgAooooAKKKKACiiigAopKKAFopKKAK1FJRmgBaQ0UUAFABJwKekZbk8CplUKOKAIfssbcyKCfao2siDmJ8Htn/GrdFZypwlujso4yvR0jLTtujFvTMkgEuT8vU1TLg9eK1dR/1y/7v9aznQHtUck4/C/vOj6xhq38anZ94/5EBUHpTCDUhTHQ005HWl7Rx+NWH9RhV1w9RPyejGUU7g0EelaRlGWzOKrQq0XapFoSjNJRVGItGKSigAxU8F7dWv8Ax73EkY9FY4/KoKKqMnF3TJlGMlaSujdg8XajFgSiOX1JGCfy4/SrsXjGFji4smUeqMCf6VytGK7IZhiYfav66nm1MpwdTXkt6aHVz+JdIuU23FjJKvo8at/Nqq/2p4d/6BK/+A0f+Nc7ijBqnmFaTu7fcRHKcPFWi2vmzpF1bw6rBhpKgg5H+jR8frVxfFmmqoVbacADAAUcf+PVx+DRg045jXjtb7hSyfDT+K7+bOtl8Y2iD/R7OR2/2yFH8zVWTxrLt/cWUaP2LPn9ABXOYppG7PqKUsxxL+1b5IccnwUd4X9WzafxjqciFV8qM/3lU5/Umqcms6jOdz3koP8AsNtH6YrPK9DTkHBrnnia0/ik/vOyngsNT+CC+4c7M7FnYsT1JOabT8UVznVYbzTgPWkpc0DF4pKKQn8aAFo60nWngUACirdmP9Ki/wB8fzquoq1aD/SYv94fzpiN6jNFFMQuaSiigAooooAKKKKBhRRRQAZozRRQIKKKKACiiigYUUUUAFNkVGH7wA/UUjyBfc+lQsxY5NDSe4KTi7xdiGW3R/8AVkr7dai23EX3TuH51aorJ0Y7rQ745hWS5alpLz1K63fOJFx9KmWVH+6wz6UMqt94A/WomtVP3SV/WlarHzK5sFW3Tg/vX+ZPRVXZPH907h+dKt1g4kTB9qPbJaSVhPL5yV6MlNeT1+4s0AkdDio1mR+jc+hqStU09jhnTnTdpqzJBMw6808Sqe+PrUFFMgtA56UZqqCR0OKeJmHXmgCeioxKp68U8HPTmgBaKKKACiiigCrRRRQABSxwBmp0iC8tyaWIfuxT6ACikooAWikFLQBnaj/rl/3f61QYVp6j0j/H+lZzUDISKaRUhpppARFRTSnpUtJWcqUJdDtpY6vSVlK67PVEXI60nBqWo261hOUqXW56WHpUMfe8OVrqv8hMUlKKdW1OfOrnm4zC/VqnJe4yilNJWhxhRRRQAUUUUAFFFFABRRRQAYFFFIaADNFJSHpQA4nHWkLelNoFAgyTThQKUUwFFSKKYKcOtAEqirNqP9Ij/wB4fzquKs2v/HxH/vD+dAG3RRRTELmikoFAC0UUUAFFFJQAUUUUAFLmkooAWikFLQAUUUhoAM1E8vZPzpZj8oqGgAooooAKKKKACiiigApGUMMMAfrS0UbjTad0QNao33crTPLni+4dw9KtUVk6MXqtDuhmFeK5Z+8uz1Ky3RXiRPyqVZo36N+BpzAMpyAfrWe33j9axnUnSdm7no0MJQx0HKMeR+TujSoqpbMfMxk49M1brphLmVzxsRR9hUcL3CgEjocUUVRgPErDrzTxMp68VDRQBZBB6HNLVXp0qwhygzQB/9k=)

# DH Dimensions

A close up of a device

Description generated with high confidence

A picture containing indoor

Description generated with high confidence

# Dh Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Θj | dj | aj | αj |
| Q1 | 0.06 | 0.1170 | π/2 |
| Q2 | 0 | 0 | -π/2 |
| Q3 | 0.352 | 0 | π/2 |
| Q4 | 0 | 0 | -π/2 |
| Q5 | 0.3215 | 0 | π/2 |
| Q6 | 0 | 0 | -π/2 |
| Q7 | 0.3049 | 0 | 0 |

End effector not pictured, all length dimension in meters, all angles in radians.

# Kinematic List

The transform of a joint, j, relative to the previous can be given by the following transformation matrix:



Aj follows the DH transformation rule:

Therefore, the transform of a joint, j, relative to the base can be given by the following transformation:

Also, it can be expressed more simply as:

In words, the transform of joint, j, can be expressed by the product of all the previous transformation matrixes.

For Fetch, which has a 7DOF arm, the transform of the end effector can be expressed simply with the following:

Also, given an initial an initial XYZ location and:

The base rotation can be expressed by:

The base position can then be expressed as:

Where:

Finally, we can say that the position of the end effector relative to the world frame can be given by:

# Fetch Model Generation

Joint limits defined based on limits provided on the fetch robotics website.

jointLim1 = deg2rad([-92 92]);

jointLim2 = deg2rad([-70 87]);

jointLim3 = [-2\*pi 2\*pi];

jointLim4 = deg2rad([-129 129]);

jointLim5 = [-2\*pi 2\*pi];

jointLim6 = deg2rad([-125 125]);

jointLim7 = [-2\*pi 2\*pi];

By importing the fetch urdf using MATLAB’s inbuilt capabilities and setting the fetch to be in its home position (arm outstretched). The transform of each joint in the world frame can be requested. Using this information, the transform of each joint relative to each other can easily be extracted. Since in the home position all the joints translate along only one axis in the world frame, the values can easily be converted to DH.

l1 = Link('d',0.06,'a',0.1170,'alpha',pi/2, ...

'offset', 0, 'qlim', jointLim1);

l1.I = body1.Inertia;

l1.r = body1.CenterOfMass;

l1.m = body1.Mass;

l2 = Link('d',0,'a',0,'alpha',-pi/2,...

'offset',-pi/2, 'qlim', jointLim2);

l2.I = body2.Inertia;

l2.r = body2.CenterOfMass + body1.CenterOfMass;

l2.m = body2.Mass;

l3 = Link('d',0.3520,'a',0,'alpha',pi/2,'offset',0, ...

'offset', 0, 'qlim', jointLim3);

l3.I = body3.Inertia;

l3.r = body3.CenterOfMass;

l3.m = body3.Mass;

l4 = Link('d',0,'a',0,'alpha',-pi/2,'offset',0, 'qlim', jointLim4);

l4.I = body4.Inertia;

l4.r = body3.CenterOfMass + body4.CenterOfMass;

l4.m = body4.Mass;

l5 = Link('d',0.3215,'a',0,'alpha',pi/2,'offset',0,...

'offset', 0, 'qlim', jointLim5);

l5.I = body5.Inertia;

l5.r = body5.CenterOfMass;

l5.m = body5.Mass;

l6 = Link('d',0,'a',0,'alpha',-pi/2,'offset',0, 'qlim', jointLim6);

l6.I = body6.Inertia;

l6.r = body6.CenterOfMass + body5.CenterOfMass;

l6.m = body6.Mass;

l7 = Link('d',0.3049,'a',0,'alpha',0,'offset',0, 'qlim', jointLim7);

l7.I = body7.Inertia;

l7.r = body7.CenterOfMass;

l7.m = body7.Mass;

links = [l1 l2 l3 l4 l5 l6 l7];

robot = SerialLink(links, 'name', 'test');

robot.base = robot.base \* transl(0,0,0.7260);

Where:

robot = importrobot('fetch.urdf');

body1 = getBody(robot,'shoulder\_pan\_link');

body2 = getBody(robot,'shoulder\_lift\_link');

body3 = getBody(robot,'upperarm\_roll\_link');

body4 = getBody(robot,'elbow\_flex\_link');

body5 = getBody(robot,'forearm\_roll\_link');

body6 = getBody(robot,'wrist\_flex\_link');

body7 = getBody(robot,'wrist\_roll\_link');

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Body1 | Body2 | Body3 | Body4 | Body5 | Body6 | Body7 \*10-3 |
| Intertia | 0.0125 0.0388 0.0308 0.0007 -0.0124 0.0012 | 0.0029 0.0657 0.0659 0.0000 0.0000 -0.0048 | 0.0019 0.0361 0.0363 0  0  -0.0005 | 0.0025 0.0430 0.0434 0  0  -0.0036 | 0.0028 0.0229 0.0246 0  0 0.0045 | 0.0018 0.0176 0.0176 0.0000 0.0000  -0.0002 | 0.1000 0.1122 0.1122 0.0000 0.0003  -0.0005 |
| CenterOfMass | 0.0927 -0.0056 0.0564 | 0.1432 0.0072 -0.0001 | 0.1165 0.0014 0 | 0.1279 0.0073 0 | 0.1097 -0.0266 0 | 0.0882 0.0009 -0.0001 | 9.5000 0.4000  -0.2000 |
| Mass | 2.5587 | 2.6615 | 2.3311 | 2.1299 | 1.6563 | 1.7250 | 135.4000 |

All Values are in SI units.

# Task 1 – Visual Servoing

## Simulated Camera on Gripper

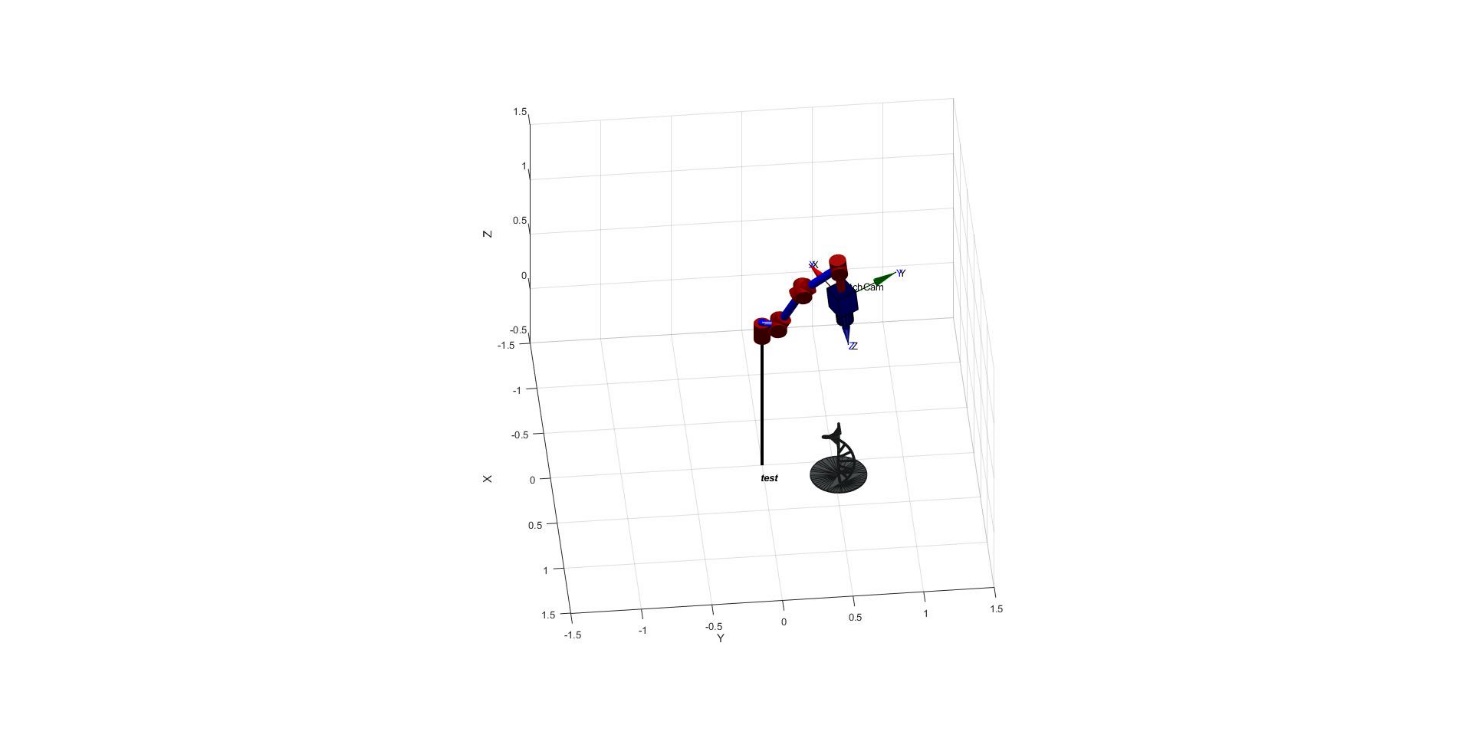
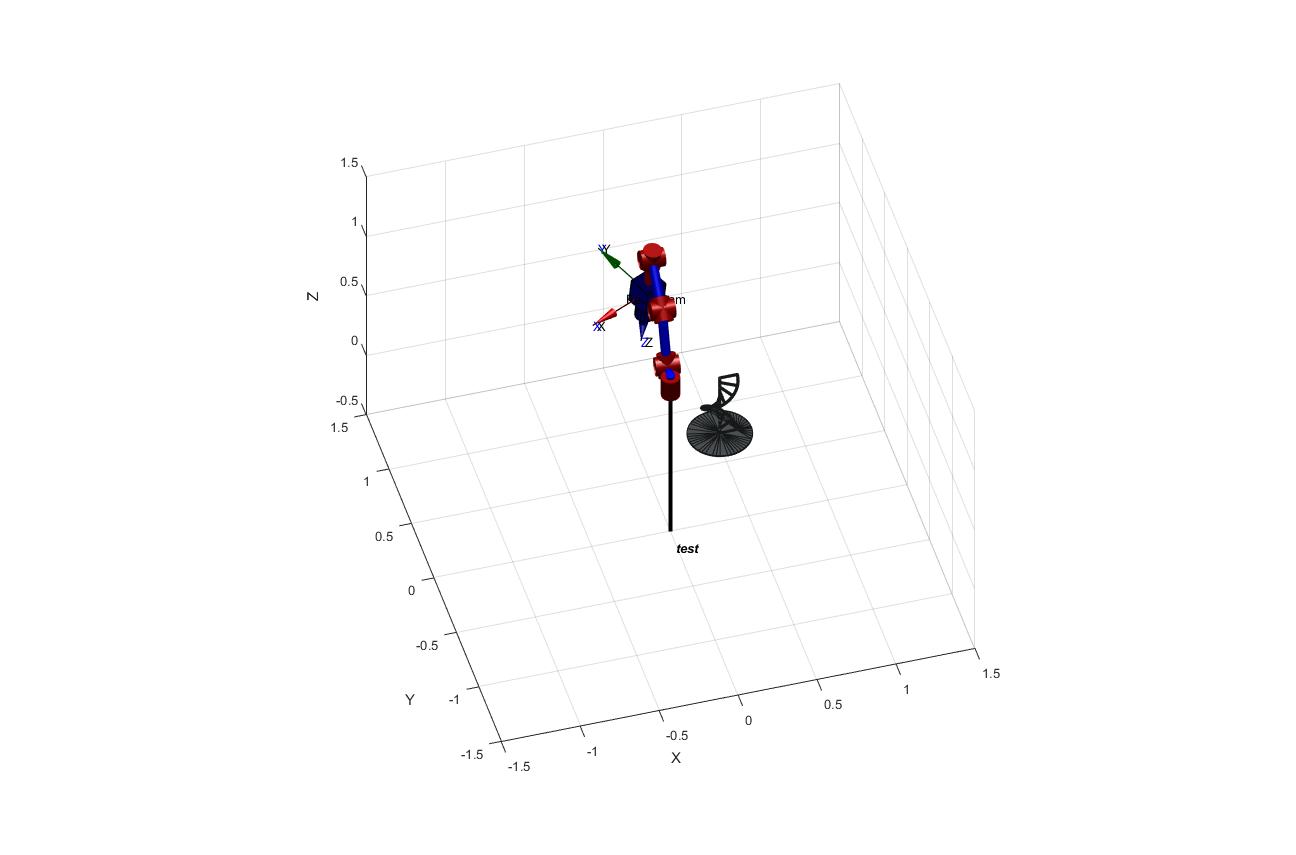


Figure 1:Fetch in Initial Pose with Simulated Camera

We can see above the fetch carrying a simulated camera at its endeffector.

## Projection View (Initial View)

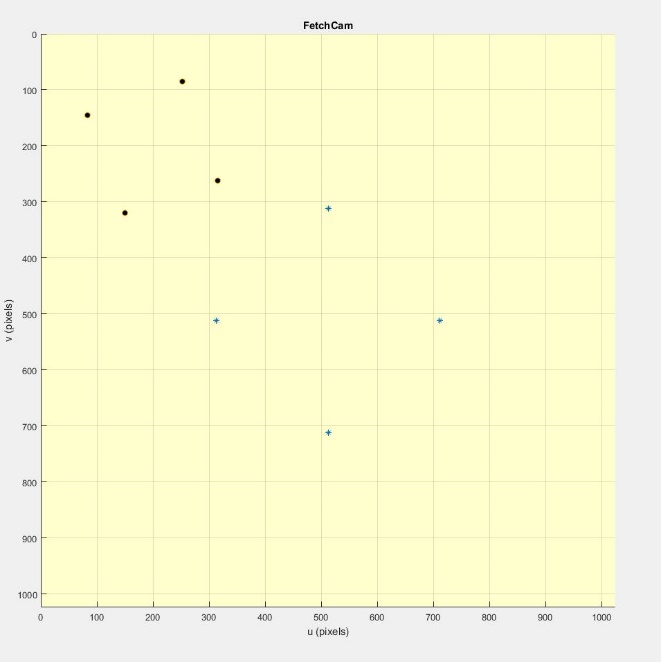


Figure 2: Fetch Initial View

## Visual Servoing Results

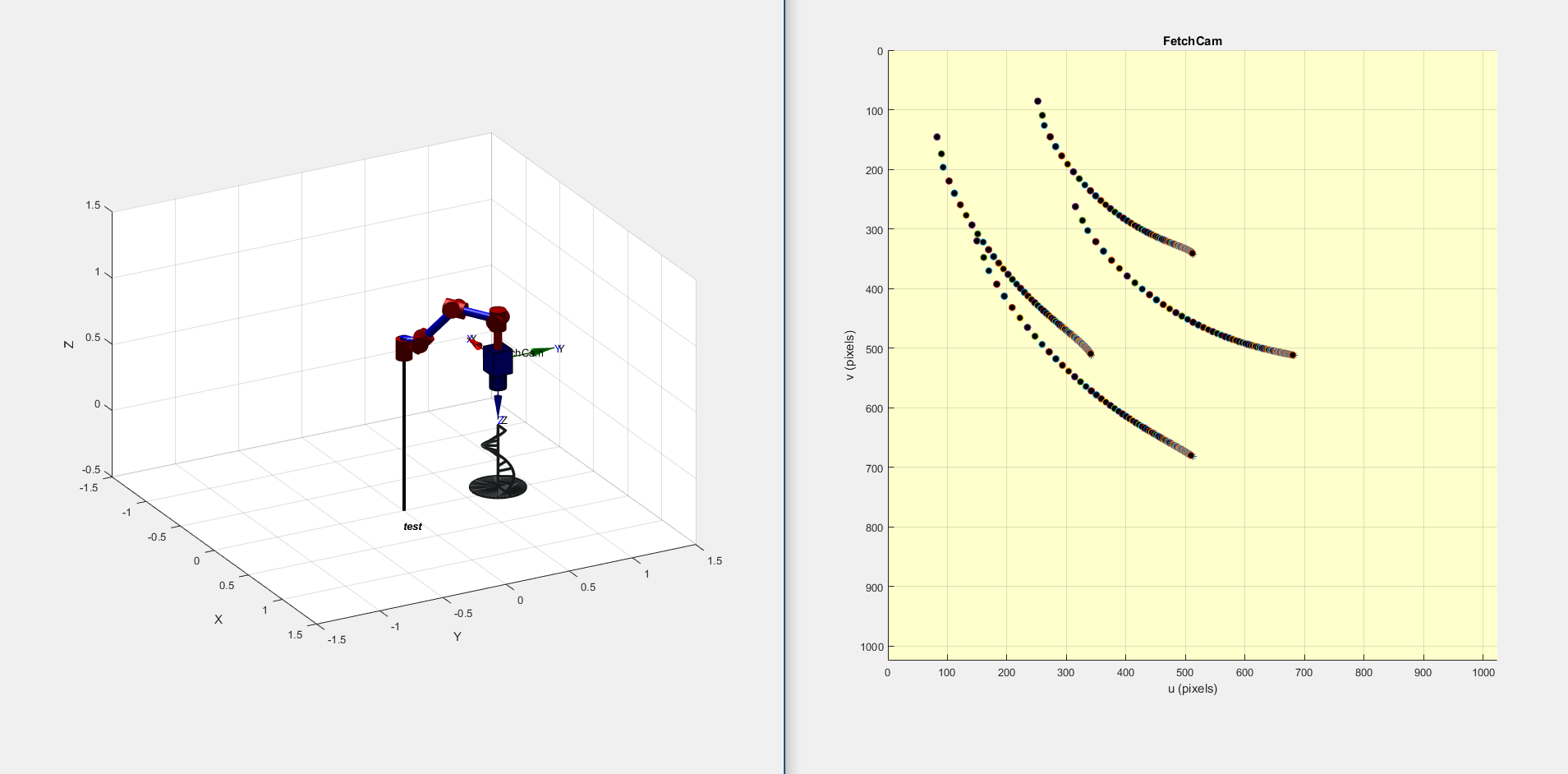


Figure 3: Visual Servoing, Just before Completion

Four points were selected that represent the outer edge of the circular base and these were used to perform the visual servoing task. Using one point would make the solution simpler to implement however it would have a final rotational error as well. As looking at a single point can be done from a variety of poses. However, looking at four can only be done from a single pose, which in this case is straight down.

## Process

**Refer to Appendix for full code solution. This is simplified inline code.**

Initialize Fetch and helix model:

bot = drawFetch(startPos);

modelLocation = transl(0.45, 0.5, 0);

hold on;

helixModel = PartLoader('helix3.ply', modelLocation);

W = eye(7);

c = [1 1 1 1 1 1 1];

Pick 4 points on the other extremes of the circle base:

mod1 = modelLocation \* transl(0.15,0,0);

mod2 = modelLocation \* transl(-0.15,0,0);

mod3 = modelLocation \* transl(0,0.15,0);

mod4 = modelLocation \* transl(0,-0.15,0);

Select 4 points in the centre of the image view that will represent the goal of the visual servoing task:

pS1 = [(512 + 170);512];

pS2 = [(512 - 170);512];

pS3 = [512;(512 - 170)];

pS4 = [512;(512 + 170)];

Setup camera

cam = CentralCamera('focal', 0.08, 'pixel', 10e-5, ...

'resolution', [1024 1024], 'centre', [512 512],'name', 'FetchCam');

fps = 25;

lambda = 1;

depth = mean (P(1,:));

Draw initial State:

Tc0= bot.fkine(bot.getpos);

cam.T = Tc0;

cam.plot\_camera(P, 'label','scale',0.15);

Project views

p = cam.plot(P, 'Tcam', Tc0);

%camera view and plotting

cam.clf()

cam.plot(pStar, '\*'); % create the camera view

cam.hold(true);

cam.plot(P, 'Tcam', Tc0, 'o'); % create the camera view

pause(2)

cam.hold(true);

cam.plot(P); % show initial view

Iterate until error is less than 3 pixels across all 4 points.

Update Camera View

uv = cam.plot(P);

% compute image plane error as a column

e = pStar-uv; % feature error

e = e(:);

Zest = [];

Calculate Image Jacobian

J = cam.visjac\_p(uv, depth );

Calculate Velocity in image frame

v = lambda \* pinv(J) \* e;

Due to initial camera starting view and orientation. Some of the velocities didn’t line up. This is fixed with the following velocity assignment.

v(1) = -v(1);

v(2) = v(2);

v(3) = -v(3);

v(4) = -v(4);

v(5) = v(5);

v(6) = -v(6);

Calculate Joint velocities for over actuated arm:

J2 = bot.jacob0(bot.getpos);

jV = (inv(W)\*J2')\*inv(J2\*inv(W)\*J2')\*v;

Limit velocities to maximums. To maintain correct trajectory, each joint will be reduced by the same ratio that the over speed joint was reduced by.

for z = 1:length(jV)

if(jV(z) > maxSpeed(z))

ratio = maxSpeed(z)/jV(z);

jV = jV\*ratio;

elseif (jV(z) < -maxSpeed(z))

ratio = -maxSpeed(z)/jV(z);

jV = jV\*ratio;

end

end

Update W (refer to calc ‘W’ function in appendix)

W = calcW(W, bot,bot.getpos,c);

Update Joints

q = q0' + (1/fps)\*qp;

bot.animate(q');

Update Camera Location

Tc = bot.fkine(q);

cam.T = Tc;

drawnow

Test Error against threshold:

test = (abs(e) < 3);

if(test)

e

test

break

end

# Task 2, Dynamic torque

Orange = Joint Limit

Blue = Joint Torque

## 2kg Load

### Motion 1

Move up along Z 1m and along X -0.4m from where the object was grasped.

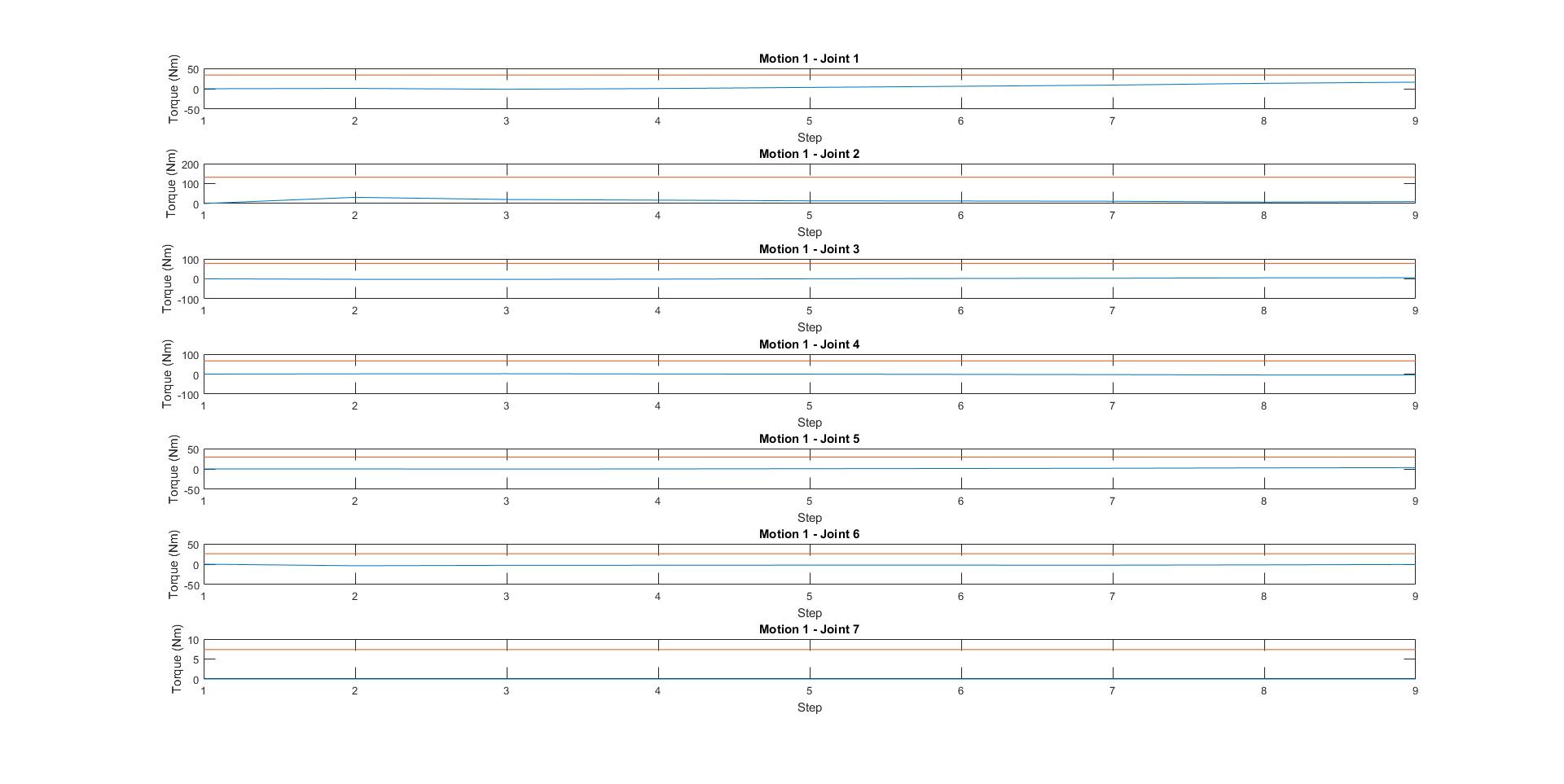


Figure 4: 2kg Load Motion 1

### Motion 2

Move along Y -3m from motion 1

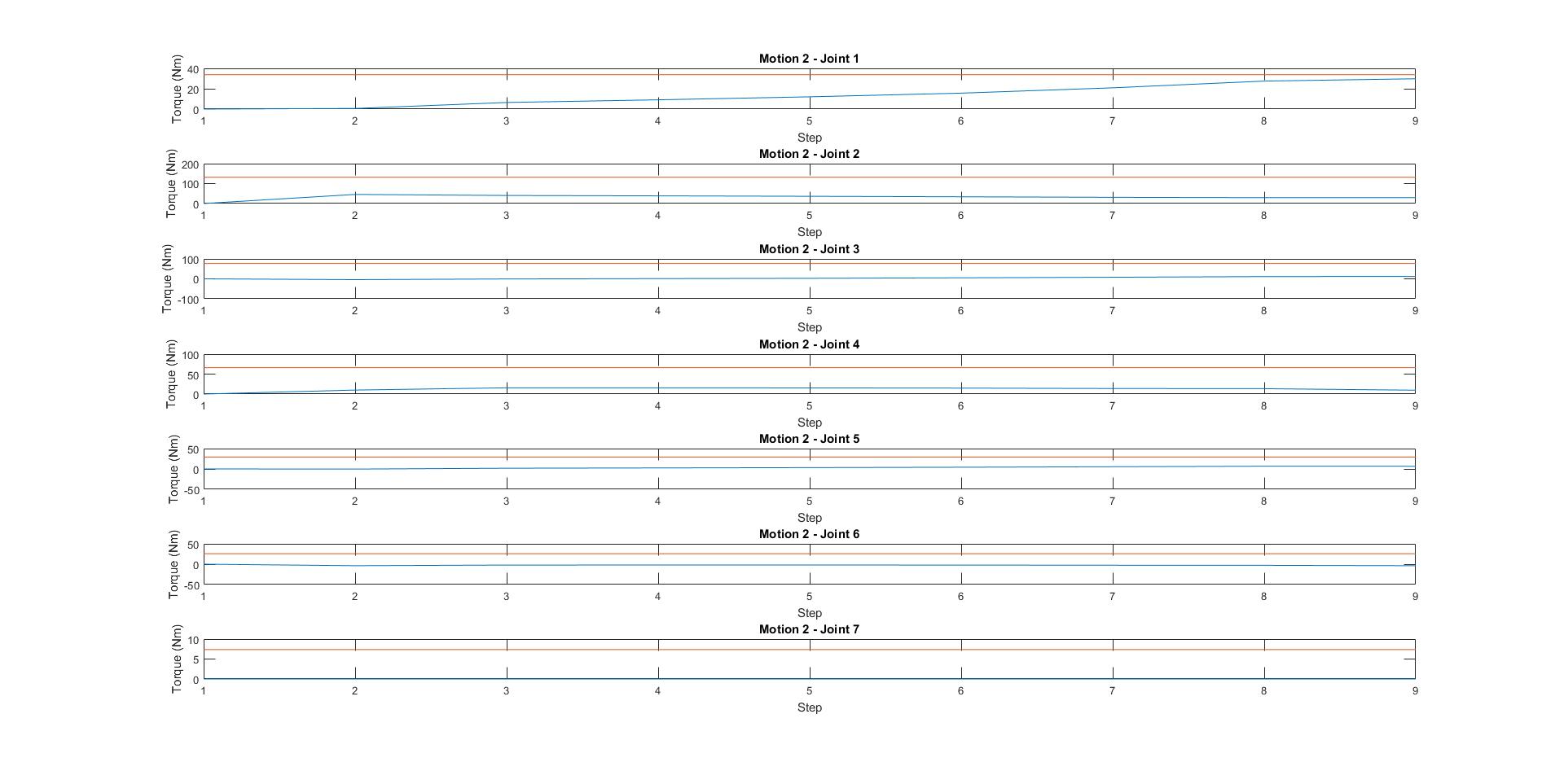


Figure 5: 2kg Load Motion 2

### Motion 3

Move down along Z 1m and along X 0.4m to place the object back down.

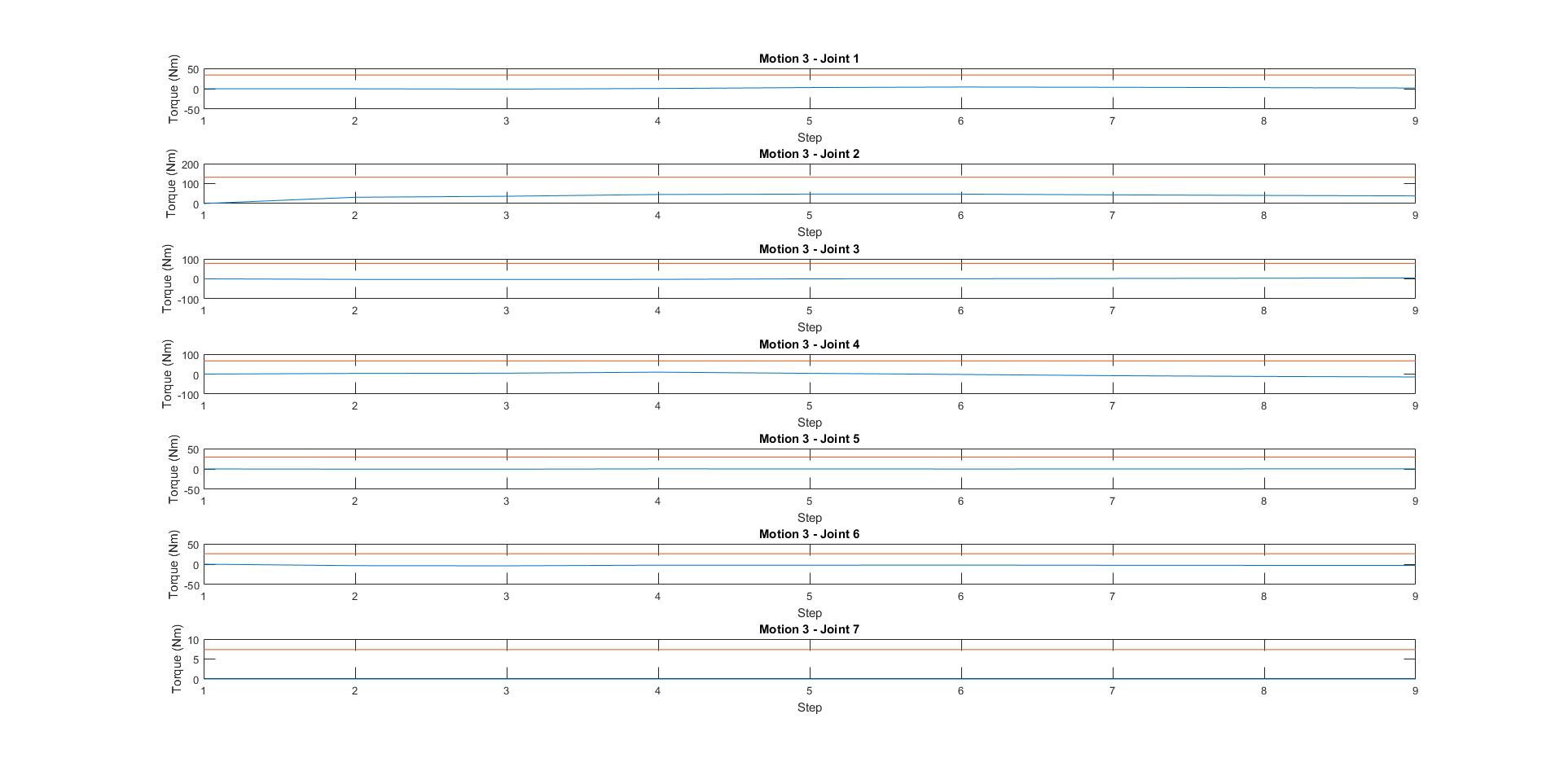


Figure 6: 2kg Load Motion 3

### Motion 4

Move up along Z 0.5m from where the object was released. The payload on the endeffector is now 0 aswell.

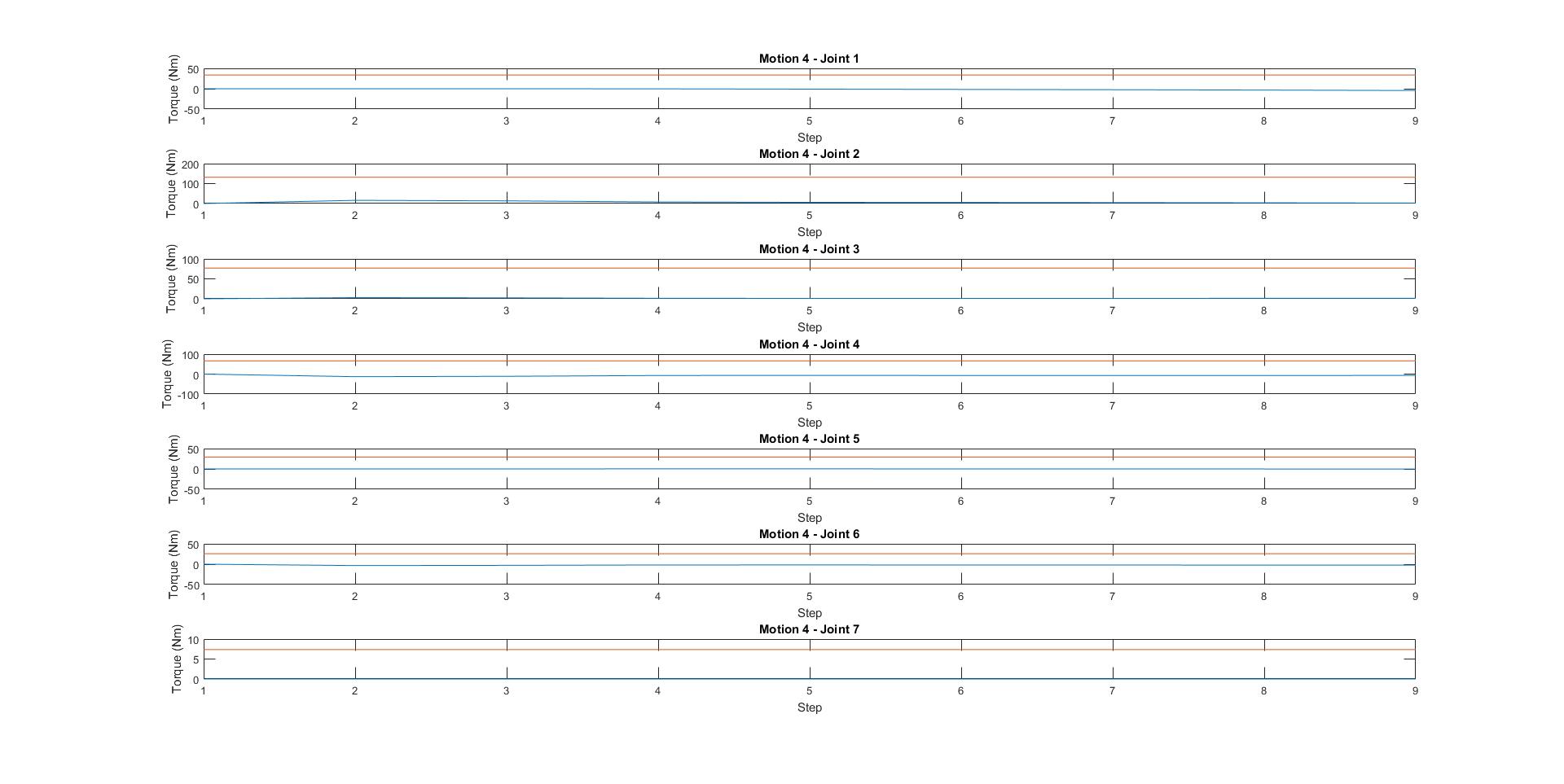


Figure 7: 2kg Load Motion 4

We can see that throughout the movement all the torques remained below the maximum joint torque. We however do see that at times it got fairly close. Motion 2, joint 1 was very close to the joint limit.

## 5kg Load

### Motion 1

Move up along Z 1m and along X -0.4m from where the object was grasped.

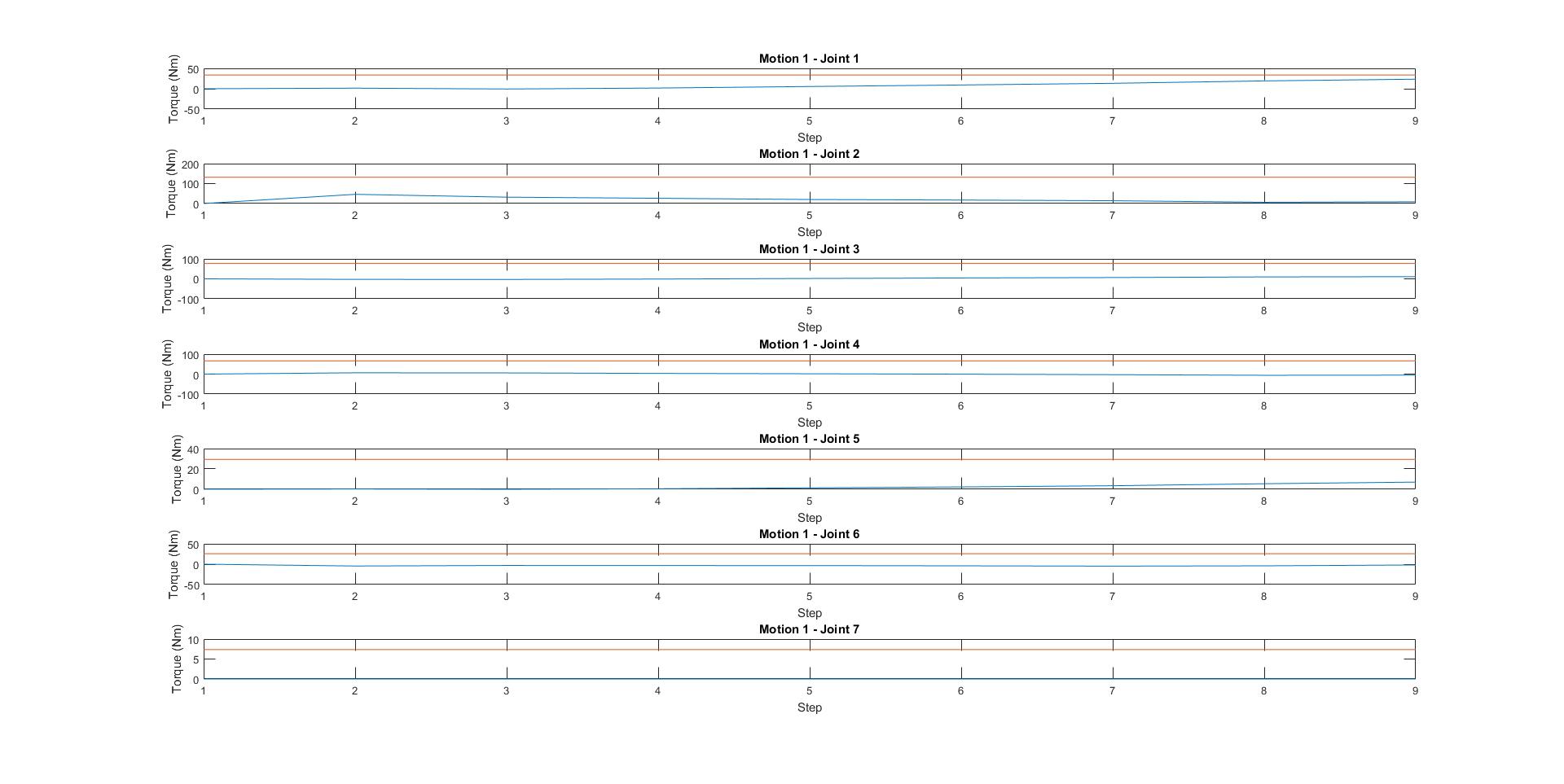


Figure 8: 5kg Load Motion 1

### Motion 2 (Joint Failure here)

Move along Y -3m from motion 1

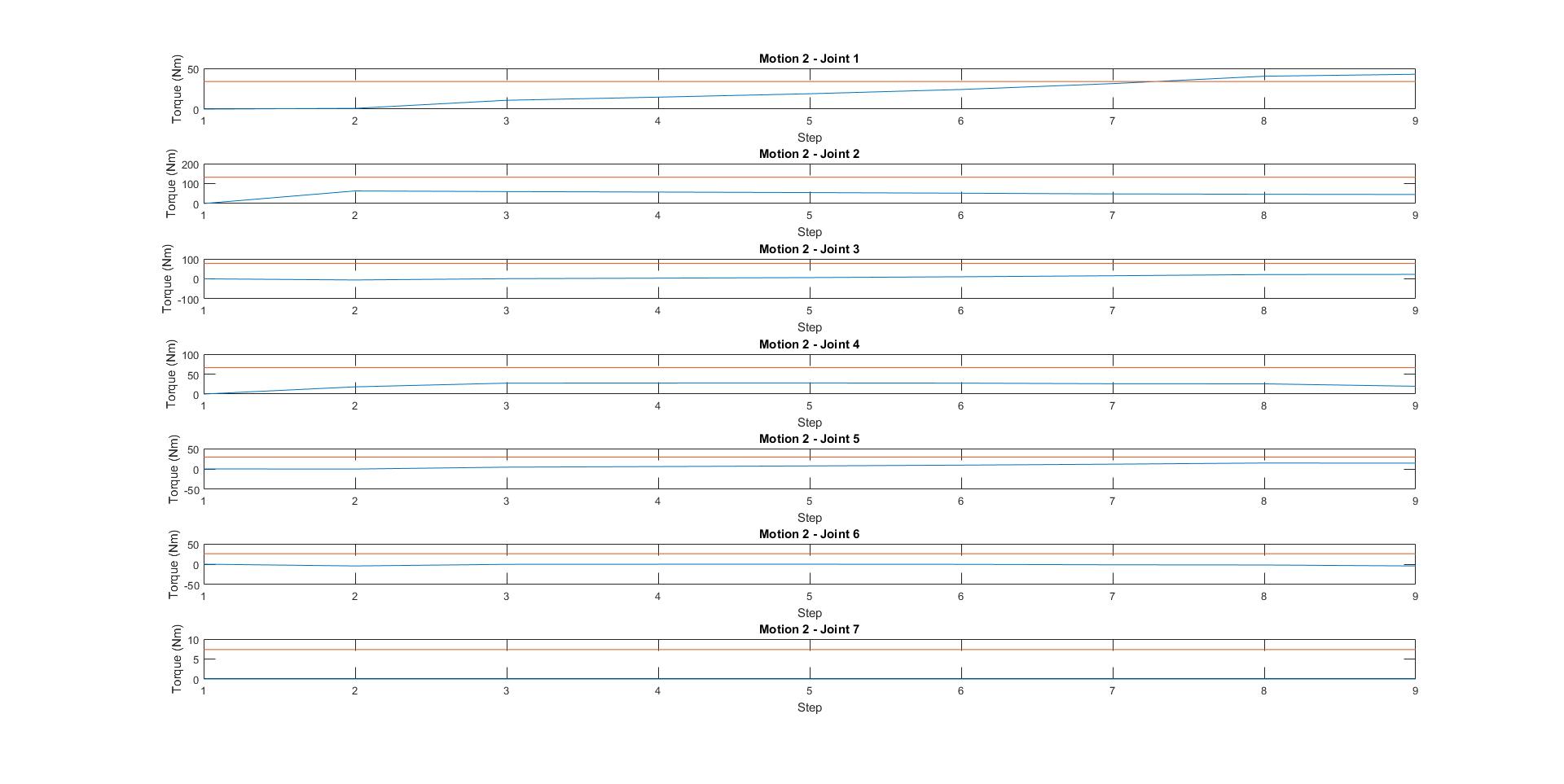


Figure 9: 5kg Load Motion 2

We can see above in motion 2, under joint 1, the arm torque exceeds the maximum joint torque, as such the fetch has failed to perform its task and all the motions have not been completed.

## Process

**Refer to Appendix for full code solution. This is simplified inline code.**

Firstly, the maximum torques are defined:

tau\_max = [33.82 131.76 76.94 66.18 29.35 25.70 7.36]';

Then we initialise the time derivative and the initial state:

dt = 1/10;

steps = time/dt;

q1 = bot.getpos;

q(1,:) = q1;

Create a trapezoidal velocity profile:

x1 = zeros(6,1);

x2 = zeros(6,1);

x1(1:3,1) = T1(1:3,4);

x2(1:3,1) = T2(1:3,4);

x = zeros(length(x1),steps);

s = lspb(0,1,steps); % Create interpolation scalar

for i = 1:steps

x(:,i) = x1\*(1-s(i)) + s(i)\*x2;

end

Initialise Arrays and set payload on end effector

qd = zeros(steps,7); % Array of joint velocities

qdd = nan(steps,7); % Array of joint accelerations

tau = nan(steps,7); % Array of joint torques % Payload mass (kg)

bot.payload(mass, massPos); % Mass and masspos are passed in arguments

Initialize 7DOF RMRC states

W = eye(7);

c = [1 1 1 1 1 1 1];

Loop through all states

Calculate joint velocities (For over actuated arm):

xdot = (x(:,i) - x(:,i-1))/dt;

J = bot.jacob0(q(i - 1,:));

jV = (inv(W)\*J')\*inv(J\*inv(W)\*J')\*xdot;

Limit velocities to maximums. To maintain correct trajectory, each joint will be reduced by the same ratio that the over speed joint was reduced by.

for z = 1:length(jV)

if(jV(z) > maxSpeed(z))

ratio = maxSpeed(z)/jV(z);

jV = jV\*ratio;

elseif (jV(z) < -maxSpeed(z))

ratio = -maxSpeed(z)/jV(z);

jV = jV\*ratio;

end

end

Calculate joint position

q(i,:) = q(i-1,:) + (jV\*dt)';

Update W (Refer to calc ‘W’ function in Appendix)

W = calcW(W, bot,q(i-1,:),c);

Calculate joint acceleration

qdd(i,:) = (1/dt)^2 \* (q(i,:) - q(i,:) - dt\*qd(i-1,:));

Calculate all affecting factors on joints

M = bot.inertia(q(i,:));

C = bot.coriolis(q(i,:),qd(i,:));

g = bot.gravload(q(i,:));

Calculate Torque on joints

tau(i,:) = (M\*qdd(i,:)' + C\*qd(i,:)' + g')';

If safe torque argument is passed in, limit torques to safe working limits

for j = 1:7

if abs(tau(i,j)) > tau\_max(j)

tau(i,j) = sign(tau(i,j))\*tau\_max(j);

end

end

# “Robots are increasingly impacting industry, our job and our daily lives in both positive and negative ways”

Unfortunately, when we look at the direction that robotics is taking we see a very bleak picture, particularly when we look at the effects that robotics has already had on society. We can see, particularly when looking at the US labour markets, a decline in employments and a reduction in total wages as more and more repetitive tasks are being automated (Acemoglu and Restrepo, 2017). As this is a continual trend we can expect that this amount of reduction of jobs will not only continue but increase in rate as technology grows exponentially.

However, the future doesn’t need to be that bleak. We currently from a world which didn’t have this level of technology, we come from a time of great change where we went from no TV’s to a world completely connected with the internet. This could be causing the fears and expectations of AI that we currently have. The new generation will grow up with all the wonders of modern technology. By Raising them to become AI literate we can expect them to better tackle all the issues that we are facing today in terms of AI. They will fear the technology less and be able to tackle it with a clear mind and come better equipped to deal with issues that arise with AI (Kandlhofer et al., 2016).

One aspect that the current and future generations will have to deal with is morality within technology. Something recent that will need to be dealt with is morality in autonomous vehicles. “If motor vehicles are to be truly autonomous and able to operate responsibly on our roads, they will need to replicate—or do better than—the human decision-making process” (Lin, 2016). Vehicles will need to make human decision based in morality, if someone must die, how do you decide? This will see an increase in morality considerations made by programmers and engineers. Moving from purely quantitative decision make to qualitative coupled with a deeper understanding of morality to allow systems to make life or death decisions.

Even with these grim considerations, that if dealt with properly will lead to much higher standards of living. We can see in the field of medicine an increase in surgical robots. These do not take over their human counterparts but add to their skills and capabilities. They have been able to improve operation success rates, reduce recovery time and reduce mistakes (Merrifield and Taylor, 2017). Overall, they have improved surgery and paint a bright future where robots work alongside humans improving their lives.

We have seen that the impact of robotics can be grim and there will be many hurdles that future designers will need to consider. But considering them early and not shying away from the issues will allow us to solve these problems. Humans have always had innovation at their door, we have always seen things relatively and have always striven for more. We have always wanted technology to improve and we have adapted to that technology. The same will happen with robotics and AI, we will adapt (Cueni, 2017). When we do, we can see a new era of prosperity and increased standard of living, leading to very positive changes because of robotics.

# “how my job may require me to create/integrate/install robots which may result in other people no longer being required to work due to no fault of their own”

Robotics is a fast-growing field that has the potential to do great, but it is already responsible and will continue being responsible for the loss of jobs. Robotics and AI are inherently skilled at doing a singular task very well, as such creative work has been untouched by robots. With the advances in machine learning we can expect robots to take over even those fields. We can see even in manufacturing, the use of robots has already taken jobs, but the act of programming a robot to do a task has been still left to a human worker. Gu ́erin et al show us the advances in manufacturing techniques that lead to robots learning and developing on their own. Essentially kicking out the human worker. As my field centres around robotics I can expect to use techniques as the one expressed by Gu ́erin et al to develop systems that will not require a human touch down the line.

Particularly in the field that I will be moving into, military, we see robotics integrating into the military as well. There has been a shift in the capabilities of robotics and as such has already reduced the operator to robot ratio (Jacoby and Chang, 2008). This is a welcome change as it will reduce the number of war fatalities, at least that of our soldiers. It will however displace a lot of the combat personnel, this is something that I will have to consider down the line. Developing robots that will take these jobs will save lives, but I will be taking jobs from people, can I justify it for the greater good. This is further reinforced by Sharkey (Sharkey, 2008), where particularly America is further expanding its robot combat force. With new and improved systems that will allow them to make kill decisions themselves. Other than the terrifying thought of killer robots, this further shows that even the military isn’t safe from automation.

Automation is taking over all aspects of the market, they are starting to infiltrate all aspects of manufacturing, design and creative works (Piva et al., 2017). No matter where you turn robotics and automation will play hand. It is inevitable that every job will be one day automated. This is where our roles play a big part. We are the primary developers of automated products. My field will see me taking jobs in whatever field I end up working in.

Frey and Osborne (Frey and Osborne, 2017) show us the potential different jobs have of being computerised, aka automated. The list is extensive and to no one’ surprise, telemarketers are ranked as the most likely to be automated. However, this list has a frightening amount of jobs that have more than 90 % chance of being automated (around 170 fall in this range) and the total amount that are more than 50% likely, 403. This is a staggering amount of potential computerised jobs, one of which being hospitality. Most developed nations move towards service type jobs, which is where we sit. Given the high probability of the hospitality industry being computerised, I can expect to play a role in the reduction of hospitality jobs and the eventually replacement of those already hired.

Given the large potential for automation in almost all field, we can further expect this list to grow and to have even more jobs fall under potential and actual computerisation. No job is safe and they will all fall to automation sooner or later. With that, it is inevitable that one day I will be responsible for the loss of someone’s job, through no fault on their own, due to something that I had created. It is inevitable that one day I will take jobs with my designs regardless of what field I work under.

# Bibliography

Acemoglu and Restrepo, 2017 Acemoglu, D. and Restrepo, P. 2017. Robotsand jobs: Evidence from us labor markets.

Kandlhofer et al., 2016] Kandlhofer, M., Steinbauer, G., Hirschmugl-Gaisch,S., and Huber, P. 2016. Artificial intelligence and computer science in edu-cation: From kindergarten to university. InFrontiers in Education Conference(FIE), 2016 IEEE, pages 1–9. IEEE.

Lin, 2016 Lin, P. 2016. Why Ethics Matters for Autonomous Cars, pages 69–85. Springer Berlin Heidelberg, Berlin, Heidelberg

Merrifield and Taylor, 2017 Merrifield, R. and Taylor, R. (2017). The surgi-cal robot challenge (from the guest editors). IEEE Robotics & AutomationMagazine, 24(2):22–23. Cueni, 2017 Cueni, R. (2017).

Robots Will Take All Our Jobs, pages 35–36.Springer International Publishing, Cham

Gu ́erin et al., 2016] Gu ́erin, J., Gibaru, O., Nyiri, E., and Thiery, S. (2016).Learning local trajectories for high precision robotic tasks: Application tokuka lbr iiwa cartesian positioning. InIECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, pages 5316–5321.

Jacoby and Chang, 2008 Jacoby, G. A. and Chang, D. J. (2008). Towards command and control networking of cooperative autonomous robotics for military applications (carma). InElectrical and Computer Engineering, 2008.CCECE 2008. Canadian Conference on, pages 000815–000820. IEEE.

Piva et al., 2017 Piva, M., Vivarelli, M., et al. (2017). Is innovation destroyingjobs? firm-level evidence from the eu. Technical report, United Nations University-Maastricht Economic and Social Research Institute on Innovationand Technology (MERIT).

Frey and Osborne, 2017 Frey, C. B. and Osborne, M. A. (2017). The future of employment: how susceptible are jobs to computerisation? Technological Forecasting and Social Change, 114:254–280.

Sharkey, 2008 Sharkey, N. (2008). Cassandra or false prophet of doom: Airobots and war.IEEE Intelligent Systems, 23(4).

# Appendix

## TorqueFun Function

function [ torques, qTotal ] = torqueFun( bot, T1, T2, time, mass, massPos, torqueSafe, maxSpeed )

tau\_max = [33.82 131.76 76.94 66.18 29.35 25.70 7.36]';

dt = 1/10;

steps = time/dt;

q1 = bot.getpos;

q(1,:) = q1;

x1 = zeros(6,1);

x2 = zeros(6,1);

x1(1:3,1) = T1(1:3,4);

x2(1:3,1) = T2(1:3,4);

x = zeros(length(x1),steps);

s = lspb(0,1,steps); % Create interpolation scalar

for i = 1:steps

x(:,i) = x1\*(1-s(i)) + s(i)\*x2; % Create trajectory in x-y plane

end

qd = zeros(steps,7); % Array of joint velocities

qdd = nan(steps,7); % Array of joint accelerations

tau = nan(steps,7); % Array of joint torques

bot.payload(mass, massPos); % Set payload mass in Puma 560 model: offset 0.1m in x-direction

W = eye(7);

c = [1 1 1 1 1 1 1];

for i = 2:steps-1

xdot = (x(:,i) - x(:,i-1))/dt;

J = bot.jacob0(q(i - 1,:));

jV = (inv(W)\*J')\*inv(J\*inv(W)\*J')\*xdot;

for z = 1:length(jV)

if(jV(z) > maxSpeed(z))

ratio = maxSpeed(z)/jV(z);

jV = jV\*ratio;

elseif (jV(z) < -maxSpeed(z))

ratio = -maxSpeed(z)/jV(z);

jV = jV\*ratio;

end

end

qd(i,:) = jV;

q(i,:) = q(i-1,:) + (jV\*dt)';

W = calcW(W, bot,q(i-1,:),c);

qdd(i,:) = (1/dt)^2 \* (q(i,:) - q(i,:) - dt\*qd(i-1,:)); % Calculate joint acceleration to get to next set of joint angles

M = bot.inertia(q(i,:)); % Calculate inertia matrix at this pose

C = bot.coriolis(q(i,:),qd(i,:)); % Calculate coriolis matrix at this pose

g = bot.gravload(q(i,:)); % Calculate gravity vector at this pose

tau(i,:) = (M\*qdd(i,:)' + C\*qd(i,:)' + g')'; % Calculate the joint torque needed

torques(i,:) = tau(i,:);

for j = 1:7

if abs(tau(i,j)) > tau\_max(j) % Check if torque exceeds limits

tau(i,j) = sign(tau(i,j))\*tau\_max(j); % Cap joint torque if above limits

end

end

if(torqueSafe)

torques(i,:) = tau(i,:);

end

qdd(i,:) = (inv(M)\*(tau(i,:)' - C\*qd(i,:)' - g'))';

end

qTotal = q;

end

## Plot Torques Function

function [] = plotTorques( overallTorque, tauMax )

for i = 1:length(overallTorque)

figure('Name', ['Motion ', num2str(i)]);

motionTorques = overallTorque{i};

for T = 1:length(motionTorques(1,:))

hold on;

subplot(length(motionTorques(1,:)),1,T);

plot(motionTorques(:,T));

hold on;

plot([1 length(motionTorques(:,1))], [tauMax(T), tauMax(T)]);

test = (motionTorques(:,T) > tauMax(T));

xlabel("Step");

ylabel("Torque (Nm)");

title(['Motion ', num2str(i), ' - Joint ', num2str(T)]);

end

end

end

## Dynamic Torque Function

function [ overallTorque ] = dynamicTorque( bot, helixModel, safeSteps, mass, time,tau\_max,maxSpeed)

masPos = [0 0 0.1];

% Move piece up

T1 = bot.fkine(bot.getpos);

T2 = T1 \* transl(-0.4,0,-1);

[torqueList, qMatrix] = torqueFun(bot, T1, T2, time, mass, masPos, safeSteps, maxSpeed);

overallTorque{1} = torqueList;

for q = 1:length(qMatrix(:,1))

Q = qMatrix(q,:);

T = torqueList(q,:);

for jT = 1:length(T)

if(T(jT) > tau\_max(jT))

disp("Joint Failed")

disp(jT);

disp(T(jT));

return;

end

end

bot.animate(Q);

helixModel.MovePart(bot.fkine(Q)\* troty(pi));

drawnow;

end

% Move along y

T1 = bot.fkine(bot.getpos);

T2 = T1 \* transl(0,-3,0);

[torqueList, qMatrix] = torqueFun(bot, T1, T2, time, mass, masPos, safeSteps,maxSpeed);

overallTorque{2} = torqueList;

for q = 1:length(qMatrix(:,1))

Q = qMatrix(q,:);

T = torqueList(q,:);

for jT = 1:length(T)

if(T(jT) > tau\_max(jT))

disp("Joint Failed")

disp(jT);

disp(T(jT));

return;

end

end

bot.animate(Q);

helixModel.MovePart(bot.fkine(Q)\* troty(pi));

drawnow;

end

% place piece down

T1 = bot.fkine(bot.getpos);

T2 = T1 \* transl(0.4,0,1);

[torqueList, qMatrix] = torqueFun(bot, T1, T2, time, mass, masPos, safeSteps,maxSpeed);

overallTorque{3} = torqueList;

for q = 1:length(qMatrix(:,1))

Q = qMatrix(q,:);

T = torqueList(q,:);

for jT = 1:length(T)

if(T(jT) > tau\_max(jT))

disp("Joint Failed")

disp(jT);

disp(T(jT));

return;

end

end

bot.animate(Q);

helixModel.MovePart(bot.fkine(Q)\* troty(pi));

drawnow;

end

%Move Away

T1 = bot.fkine(bot.getpos);

T2 = T1 \* transl(0,0,-0.5);

[torqueList, qMatrix] = torqueFun(bot, T1, T2, time, 0, masPos, safeSteps,maxSpeed);

overallTorque{4} = torqueList;

for q = 1:length(qMatrix(:,1))

Q = qMatrix(q,:);

T = torqueList(q,:);

for jT = 1:length(T)

if(T(jT) > tau\_max(jT))

disp("Joint Failed")

disp(jT);

disp(T(jT));

return;

end

end

bot.animate(Q);

drawnow;

end

end

## Draw Fetch Function

function [ robot ] = drawFetch( startPos )

robot = importrobot('fetch.urdf');

homeConfig = homeConfiguration(robot);

transform1 = getTransform(robot,homeConfig,'shoulder\_pan\_link');

transform2 = getTransform(robot,homeConfig,'shoulder\_lift\_link');

transform3 = getTransform(robot,homeConfig,'upperarm\_roll\_link');

transform4 = getTransform(robot,homeConfig,'elbow\_flex\_link');

transform5 = getTransform(robot,homeConfig,'forearm\_roll\_link');

transform6 = getTransform(robot,homeConfig,'wrist\_flex\_link');

transform7 = getTransform(robot,homeConfig,'wrist\_roll\_link');

transform8 = getTransform(robot,homeConfig,'gripper\_link');

transformList = [];

transformList(:,:,1) = transform1;

transformList(:,:,2) = transform2;

transformList(:,:,3) = transform3;

transformList(:,:,4) = transform4;

transformList(:,:,5) = transform5;

transformList(:,:,6) = transform6;

transformList(:,:,7) = transform7;

transformList(:,:,8) = transform8;

distanceList = [];

for i = 1:(length(transformList) - 1)

differenceTransform = transformList(:,:,i+1) - transformList(:,:,i);

distanceList(1,i) = differenceTransform(1,4);

end

body1 = getBody(robot,'shoulder\_pan\_link');

body2 = getBody(robot,'shoulder\_lift\_link');

body3 = getBody(robot,'upperarm\_roll\_link');

body4 = getBody(robot,'elbow\_flex\_link');

body5 = getBody(robot,'forearm\_roll\_link');

body6 = getBody(robot,'wrist\_flex\_link');

body7 = getBody(robot,'wrist\_roll\_link');

close all;

jointLim1 = deg2rad([-92 92]);

jointLim2 = deg2rad([-70 87]);

jointLim3 = [-2\*pi 2\*pi];

jointLim4 = deg2rad([-129 129]);

jointLim5 = [-2\*pi 2\*pi];

jointLim6 = deg2rad([-125 125]);

jointLim7 = [-2\*pi 2\*pi];

l1 = Link('d',0.06,'a',0.1170,'alpha',pi/2, ...

'offset', 0, 'qlim', jointLim1);

l1.I = body1.Inertia;

l1.r = body1.CenterOfMass;

l1.m = body1.Mass;

l2 = Link('d',0,'a',0,'alpha',-pi/2,...

'offset',-pi/2, 'qlim', jointLim2);

l2.I = body2.Inertia;

l2.r = body2.CenterOfMass + body1.CenterOfMass;

l2.m = body2.Mass;

l3 = Link('d',0.3520,'a',0,'alpha',pi/2,'offset',0, ...

'offset', 0, 'qlim', jointLim3);

l3.I = body3.Inertia;

l3.r = body3.CenterOfMass;

l3.m = body3.Mass;

l4 = Link('d',0,'a',0,'alpha',-pi/2,'offset',0, 'qlim', jointLim4);

l4.I = body4.Inertia;

l4.r = body3.CenterOfMass + body4.CenterOfMass;

l4.m = body4.Mass;

l5 = Link('d',0.3215,'a',0,'alpha',pi/2,'offset',0,...

'offset', 0, 'qlim', jointLim5);

l5.I = body5.Inertia;

l5.r = body5.CenterOfMass;

l5.m = body5.Mass;

l6 = Link('d',0,'a',0,'alpha',-pi/2,'offset',0, 'qlim', jointLim6);

l6.I = body6.Inertia;

l6.r = body6.CenterOfMass + body5.CenterOfMass;

l6.m = body6.Mass;

l7 = Link('d',0.3049,'a',0,'alpha',0,'offset',0, 'qlim', jointLim7);

l7.I = body7.Inertia;

l7.r = body7.CenterOfMass;

l7.m = body7.Mass;

links = [l1 l2 l3 l4 l5 l6 l7];

robot = SerialLink(links, 'name', 'test');

robot.base = robot.base \* transl(0,0,0.7260);

robot.plot(startPos);

end

**MAKE SURE THE FETCH.URDF IS INCLUDED IN YOUR PATH. IT WILL EXTRACT DETAILS FROM IT.**

## PerfromVS function

function [cam] = performVS(status)

%% Draw Fetch and staircase model

close all;

clc;

% clear;

tau\_max = [33.82 131.76 76.94 66.18 29.35 25.70 7.36]';

maxSpeed = [1.25 1.45 1.57 1.52 1.57 2.26 2.26];

startPos = [1.4772 1.0687 0.1256 -0.4053 -0.0001 -2.1380 1.1257];

bot = drawFetch(startPos);

modelLocation = transl(0.45, 0.5, 0);

hold on;

helixModel = PartLoader('helix3.ply', modelLocation);

mod1 = modelLocation \* transl(0.15,0,0);

mod2 = modelLocation \* transl(-0.15,0,0);

mod3 = modelLocation \* transl(0,0.15,0);

mod4 = modelLocation \* transl(0,-0.15,0);

pS1 = [(512 + 170);512];

pS2 = [(512 - 170);512];

pS3 = [512;(512 - 170)];

pS4 = [512;(512 + 170)];

pStar = [pS2 pS1 pS4 pS3];

P = [mod1(1:3,4) mod2(1:3,4) mod3(1:3,4) mod4(1:3,4)];

depth = mean (P(1,:));

T1 = bot.fkine(bot.getpos);

depth = norm(T1(1:3,4) - modelLocation(1:3,4));

axis([-1.5 1.5 -1.5 1.5 -0.5 1.5])

%% Setup Cam

cam = CentralCamera('focal', 0.08, 'pixel', 10e-5, ...

'resolution', [1024 1024], 'centre', [512 512],'name', 'FetchCam');

% frame rate

fps = 25;

%Define values

%gain of the controler

lambda = 1;

%% Draw inital State

Tc0= bot.fkine(bot.getpos);

cam.T = Tc0;

cam.plot\_camera(P, 'label','scale',0.15);

%% Projection

p = cam.plot(P, 'Tcam', Tc0);

%camera view and plotting

cam.clf()

cam.plot(pStar, '\*'); % create the camera view

cam.hold(true);

cam.plot(P, 'Tcam', Tc0, 'o'); % create the camera view

pause(2)

cam.hold(true);

cam.plot(P); % show initial view

%% Loop

ksteps = 0;

W = eye(7);

c = [1 1 1 1 1 1 1];

while true

ksteps = ksteps + 1;

% compute the view of the camera

uv = cam.plot(P);

% compute image plane error as a column

e = pStar-uv; % feature error

e = e(:);

Zest = [];

% compute the Jacobian

if isempty(depth)

% exact depth from simulation (not possible in practice)

pt = homtrans(inv(cam.T), P);

J = cam.visjac\_p(uv, pt(3,:) );

elseif ~isempty(Zest)

J = cam.visjac\_p(uv, Zest);

else

J = cam.visjac\_p(uv, depth );

end

% compute the velocity of camera in camera frame

try

v = lambda \* pinv(J) \* e;

catch

status = -1;

return

end

% fprintf('v: %.3f %.3f %.3f %.3f %.3f %.3f\n', v);

% The bottom weirdness (changing velocity directions) comes from testing and the initial camera pose. Only the values under else are used the top was left over.

if(length(pStar(1,:)) == 1)

v(1) = -v(1);

v(2) = v(2);

v(3) = v(3);

v(4) = -v(4);

v(5) = v(5);

v(6) = v(6);

else

v(1) = -v(1);

v(2) = v(2);

v(3) = -v(3);

v(4) = -v(4);

v(5) = v(5);

v(6) = -v(6);

end

J2 = bot.jacob0(bot.getpos);

jV = (inv(W)\*J2')\*inv(J2\*inv(W)\*J2')\*v;

slowDown = 0;

for x = 1:length(jV)

if(jV(x) > 20)

slowDown = 1;

end

end

if(slowDown)

jV = jV.\*0.01;

end

qp = jV;

q0 = bot.getpos;

% qp = bot.getPos + (jV\*deltaT)';

W = calcW(W, bot,bot.getpos,c);

%Update joints

q = q0' + (1/fps)\*qp;

bot.animate(q');

%Get camera location

Tc = bot.fkine(q);

cam.T = Tc;

drawnow

pause(1/fps)

test = (abs(e) < 3);

if(test)

e

test

break

end

end %loop finishese

%% Move along Z

%Camera is no longer needed, lets just move it away

cam.T = transl(0,0,10);

drawnow;

height = bot.fkine(bot.getpos);

height = (height(3,4));

W = eye(7);

c = [1 1 1 1 1 1 1];

fps = 25;

while(height ~= 0)

distance = bot.fkine(bot.getpos) - modelLocation;

height = bot.fkine(bot.getpos);

height = (height(3,4));

test = norm(distance(1:3,4));

if(test < 0.09)

break;

end

velocity = [0 0 -1 0 0 0]';

J2 = bot.jacob0(bot.getpos);

jV = (inv(W)\*J2')\*inv(J2\*inv(W)\*J2')\*velocity;

for z = 1:length(jV)

if(jV(z) > maxSpeed(z))

ratio = maxSpeed(z)/jV(z);

jV = jV\*ratio;

elseif (jV(z) < -maxSpeed(z))

ratio = -maxSpeed(z)/jV(z);

jV = jV\*ratio;

end

end

qp = jV;

q0 = bot.getpos;

% qp = bot.getPos + (jV\*deltaT)';

W = calcW(W, bot,bot.getpos,c);

%Update joints

q = q0' + (1/fps)\*qp;

bot.animate(q');

pause(1/fps);

end

%% Move piece to random spot

if(status == 1)

qMatrix = jtraj(bot.getpos, [0.9313 0.9591 0.2513 -0.4053 -0.1257 0 1],40);

for i = 1:length(qMatrix(:,1))

bot.animate(qMatrix(i,:));

helixModel.MovePart(bot.fkine(qMatrix(i,:)) \* troty(pi));

end

end

if(status == 2)

disp("Mass = 2kg");

mass = 2;

time = 1;

overallTorque = dynamicTorque(bot,helixModel,1, mass, time,tau\_max, maxSpeed);

disp("Sucessfully Completed Drop");

end

if(status == 3)

disp("Mass = 5kg");

mass = 5;

time = 1;

overallTorque = dynamicTorque(bot,helixModel,0, mass, time,tau\_max, maxSpeed);

end

try

plotTorques(overallTorque,tau\_max);

end

end

## Calculate ‘W’ Function

function [ W ] = calcW( WPrev, robot, pose, c)

W = eye(length(pose));

Links = robot.links;

for i = 1:length(pose)

try

qMin = Links(i).qlim(1);

qMax = Links(i).qlim(2);

q = pose(i);

value = ((qMax - qMin)\*(2\*q - qMax - qMin))/(c(i)\*((qMax - q)^2)\*((q - qMin)^2));

if(isnan(value))

c(i)

q

qMax

qMin

end

if(value - WPrev(i,i) > 0)

W(i,i) = 1 + abs(value);

else

W(i,i) = 1;

end

catch

W(i,i) = 1;

end

end

end

## RunMe Script

%% Only VisualServoing

%Will visual servo to the object, move down, pick the object up and move

%using jtraj to a joint position I randomly selected. No torque or torque

%plot will be checked here

clear all;

camObj = performVS(1);

%% Mass = 2kg

% Visual Servo to object, move down, pick up the object, then maintain a

% safe torque (and speed) throughout the process while trying to complete

% the task at the same time as the 5kg scenario

%Three motions once the object is grasped

% Move up 1 and along x -0.4

% move along y -3

% move down 1 and along x 0.4

% Torques will be graphed at the completion of all motions. Seperate figure

% for each motion. Each joint is on seperate subplot. Blue line indicates

% joint torque. Orange Line indicates max joint torque.

clear all;

camObj = performVS(2);

%% Mass = 5kg

% Visual Servo to object, move down, pick up the object, a safe speed will

% be maintained but not a safe torque while trying to complete the task at

%the same time as the 2kg scenario. It will fail while moving along y

%Three motions once the object is grasped

% Move up 1 and along x -0.4

% move along y -3

% move down 1 and along x 0.4

% Torques will be graphed at the completion of all motions. Seperate figure

% for each motion. Each joint is on seperate subplot. Blue line indicates

% joint torque. Orange Line indicates max joint torque.

clear all;

camObj = performVS(3);

## GitHub Clone Link

https://github.com/denis-draca/roboticsA4.git

Start with the RunMe Script