```
> restart
> with(LinearAlgebra) :
> with(CodeGeneration) :
> with(CurveFitting) :
> with(plots) :
```

KINEMATICS

Defining known lengths

```
> a1 := 0.07 :; a2 := 0.36 :; a3 := 0.445 :; d1 := 0.352 :
```

Joint definitions

```
> q := Vector([[q1(t)], [q2(t)], [q3(t)]]) :
> Dq := map(diff, q, t) :
> DDq := map(diff, Dq, t) :
```

Calculating the translation matrix

$$> A1 := \begin{bmatrix} \cos(q[1]) & 0 & -\sin(q[1]) & a1\cdot\cos(q[1]) \\ \sin(q[1]) & 0 & \cos(q[1]) & a1\cdot\sin(q[1]) \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$> A2 := \begin{bmatrix} \sin(q[2]) & \cos(q[2]) & 0 & a2\cdot\sin(q[2]) \\ -\cos(q[2]) & \sin(q[2]) & 0 & -a2\cdot\cos(q[2]) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

$$> A3 := \begin{bmatrix} -\sin(q[3]) & -\cos(q[3]) & 0 & -a3\cdot\sin(q[3]) \\ \cos(q[3]) & -\sin(q[3]) & 0 & a3\cdot\cos(q[3]) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} :$$

```
> T03 := MatrixMatrixMultiply(MatrixMatrixMultiply(A1, A2), A3) :
```

Calculating end effector coordinates

> $Ident1 := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$:

> $Ident2 := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$:

> $Ident3 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$:

```
> P0 := MatrixMatrixMultiply(MatrixMatrixMultiply(Ident1, T03), Ident2) :

> #Matlab(P0[1], resultname = "p0x");
> #Matlab(P0[2], resultname = "p0y");
> #Matlab(P0[3], resultname = "p0z");
```

Calculating rotation matrices

```
> R01 := MatrixMatrixMultiply(MatrixMatrixMultiply(Ident1, A1), Ident3) :
> R12 := MatrixMatrixMultiply(MatrixMatrixMultiply(Ident1, A2), Ident3) :
> R23 := MatrixMatrixMultiply(MatrixMatrixMultiply(Ident1, A3), Ident3) :
> R02 := MatrixMatrixMultiply(R01, R12) :
> R03 := MatrixMatrixMultiply(R02, R23) :
```

Calculating rotation axis of each joint expressed in its own frame

\>

> $z0 := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$:

```
> b1 := MatrixMatrixMultiply(Transpose(R01), z0) :
> b2 := combine(MatrixMatrixMultiply(Transpose(R02), MatrixVectorMultiply(R01, z0)),
      trig) :
> b3 := combine(MatrixMatrixMultiply(Transpose(R03), MatrixVectorMultiply(R02, z0)),
      trig) :
```

DYNAMICS

Defining the initial gravity vector

$$> g0 := \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} :$$

Defining the link masses (in kg) with values from SolidWorks

```
> m1 := 34.655 :
> m2 := 15.994 :
> m3 := 20.862 :
```

Defining the Inertia Matrices with values from SolidWorks

\>

$$> I1 := \begin{bmatrix} 0.51205253974 & -0.00385971 & 0.051305 \\ 0.00136134 & 0.46407468859 & 0.0703356 \\ 0.051305 & -0.0627205 & 0.55411384358 \end{bmatrix} :$$

$$> I2 := \begin{bmatrix} 0.0948179 & -0.00385971 & 0.037932 \\ -0.00385971 & 0.32860416324 & -0.00108897 \\ 0.037932 & -0.00108897 & 0.27746300488 \end{bmatrix} :$$

$$> I3 := \begin{bmatrix} 0.50006091595 & -0.00186325 & 0.000934876 \\ -0.00186325 & 0.0751527 & -0.0152041 \\ 0.000934876 & -0.0152041 & 0.51542475434 \end{bmatrix} :$$

Defining mass center vectors with values from SolidWorks

```
> r0c1 := Vector([[0.08903], [-0.02789], [0.04312]]) :
> r1c2 := Vector([[0.19829], [-0.09243], [0.00973]]) :
> r2c3 := Vector([[0.07996], [0.00456], [0.00586]]) :
> r01 := Vector([[a1], [-d1], [0]]) :
> r12 := Vector([[a2], [0], [0]]) :
> r23 := Vector([[a3], [0], [0]]) :
> r1c1 := r0c1 − r01 :
> r2c2 := r1c2 − r12 :
> r3c3 := r2c3 − r23 :
```

Newton − Euler recursions

Forward Recursion for Link 1:

```
> ω1 := b1·Dq[1]:
> α1 := b1·DDq[1] + CrossProduct(ω1, b1·Dq[1]) :
> Dω1 := map(diff, ω1, t) :
> ae1 := CrossProduct(Dω1, r01) + CrossProduct(ω1, CrossProduct(ω1, r01)) :
> ac1 := CrossProduct(Dω1, r0c1) + CrossProduct(ω1, CrossProduct(ω1, r0c1)) :
> g1 := MatrixMatrixMultiply(Transpose(R01), g0) :
```

Forward Recursion for Link 2:

```
> ω2 := combine(MatrixVectorMultiply(Transpose(R12), ω1) + b2·Dq[2], trig) :
> α2 := combine(MatrixVectorMultiply(Transpose(R12), α1) + b2·DDq[2]
        + CrossProduct(ω2, b2·Dq[2]), trig) :
> Dω2 := map(diff, ω2, t) :
> ae2 := combine(MatrixVectorMultiply(Transpose(R12), ae1) + CrossProduct(Dω2, r12)
        + CrossProduct(ω2, CrossProduct(ω2, r12)), trig) :
> ac2 := combine(MatrixVectorMultiply(Transpose(R12), ae1) + CrossProduct(Dω2, r1c2)
        + CrossProduct(ω2, CrossProduct(ω2, r1c2)), trig) :
> g2 := combine(MatrixVectorMultiply(Transpose(R02), g0), trig) :
```

Forward Recursion for Link 3:

```
> ω3 := combine(MatrixVectorMultiply(Transpose(R23), ω2) + b3·Dq[3], trig) :
> α3 := combine(MatrixVectorMultiply(Transpose(R23), α2) + b3·DDq[3]
        + CrossProduct(ω3, b3·Dq[3]), trig) :
> Dω3 := map(diff, ω3, t) :
> ae3 := combine(MatrixVectorMultiply(Transpose(R23), ae2) + CrossProduct(Dω3, r23)
        + CrossProduct(ω3, CrossProduct(ω3, r23)), trig) :
> ac3 := combine(MatrixVectorMultiply(Transpose(R23), ae2) + CrossProduct(Dω3, r2c3)
        + CrossProduct(ω3, CrossProduct(ω3, r2c3)), trig) :
> g3 := combine(MatrixVectorMultiply(Transpose(R03), g0), trig) :
```

Backward Recursion for Link 3

```
> f3 := Add(m3·ac3, -m3·g3) :
> τ3 := -CrossProduct(f3, r2c3) + MatrixVectorMultiply(I3, α3) + CrossProduct(ω3,
        MatrixVectorMultiply(I3, ω3)) :
> τ3z := collect(combine(MatrixVectorMultiply(Transpose(b3), τ3), trig), {DDq[1], DDq[2],
        DDq[3], Dq[1], Dq[2], Dq[3]}) :
```

Backward Recursion for Link 2

```
> f2 := MatrixVectorMultiply(R23, f3) + Add(m2·ac2, -m2·g2) :
> τ2 := MatrixVectorMultiply(R23, τ3) − CrossProduct(f2, r1c2)
      + CrossProduct(MatrixVectorMultiply(R23, f3), r2c2) + MatrixVectorMultiply(I2, α2)
      + CrossProduct(ω2, MatrixVectorMultiply(I2, ω2)) :
> τ2z := collect(combine(MatrixVectorMultiply(Transpose(b2), τ2), trig), {DDq[1], DDq[2],
      DDq[3], Dq[1], Dq[2], Dq[3]}) :
```

Backward Recursion for Link 1

```
> f1 := MatrixVectorMultiply(R12, f2) + Add(m1·ac1, -m1·g1) :
> τ1 := MatrixVectorMultiply(R12, τ2) − CrossProduct(f1, r0c1)
      + CrossProduct(MatrixVectorMultiply(R12, f2), r1c1) + MatrixVectorMultiply(I1, α1)
      + CrossProduct(ω1, MatrixVectorMultiply(I1, ω1)) :
> τ1z := collect(combine(MatrixVectorMultiply(Transpose(b1), τ1), trig), {DDq[1], DDq[2],
      DDq[3], Dq[1], Dq[2], Dq[3]}) :
```

Defining dynamic system matrix elements

Inertia-matrix

```
> m11 := eval(τ1z, {DDq[1] = 1, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m12 := eval(τ1z, {DDq[1] = 0, DDq[2] = 1, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m13 := eval(τ1z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 1, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m21 := eval(τ2z, {DDq[1] = 1, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m22 := eval(τ2z, {DDq[1] = 0, DDq[2] = 1, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m23 := eval(τ2z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 1, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m31 := eval(τ3z, {DDq[1] = 1, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m32 := eval(τ3z, {DDq[1] = 0, DDq[2] = 1, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
> m33 := eval(τ3z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 1, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0, g
      = 0}) :
```

Gravity vector

```
> g1 := eval(τ1z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0}) :
> g2 := eval(τ2z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0}) :
```

```
> g3 := eval(τ3z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, Dq[1] = 0, Dq[2] = 0, Dq[3] = 0}) :
```

```
> cM1 := eval(τ1z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, g = 0}) :
> cM2 := eval(τ2z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, g = 0}) :
> cM3 := eval(τ3z, {DDq[1] = 0, DDq[2] = 0, DDq[3] = 0, g = 0}) :
```

```
> #Matlab(cM1, resultname = "c1");
> #Matlab(cM2, resultname = "c2");
> #Matlab(cM3, resultname = "c3");
```

Setting up the dynamic system on matrix form

```
> M0 := Matrix([[m11, m12, m13], [m21, m22, m23], [m31, m32, m33]]) :
> G0 := Vector([[g1], [g2], [g3]]) :
> τ0 := Vector([[τ1z], [τ2z], [τ3z]]) :
> C0Dq := combine(τ0 − MatrixVectorMultiply(M0, DDq) − G0, trig) :
```

Setting up the expression for kinetic energy

> Ek := $\frac{1}{2}$ · VectorMatrixMultiply(Transpose(Dq), MatrixVectorMultiply(M0, Dq)) :

Code to convert matrix elements to matlab code

```
> #Matlab(m11, resultname = "m11");
> #Matlab(m12, resultname = "m12");
> #Matlab(m13, resultname = "m13");
> #Matlab(m21, resultname = "m21");
> #Matlab(m22, resultname = "m22");
> #Matlab(m23, resultname = "m23");
```

```
> #Matlab(m31, resultname = "m31");
> #Matlab(m32, resultname = "m32");
> #Matlab(m33, resultname = "m33");
```

```
> #Matlab(g1, resultname = "g1");
> #Matlab(g2, resultname = "g2");
> #Matlab(g3, resultname = "g3");
```

```
> #Matlab(C0Dq(1), resultname = "cdq1");
> #Matlab(C0Dq(2), resultname = "cdq2");
> #Matlab(C0Dq(3), resultname = "cdq3");
```

PATH AND TRAJECTORY PLANNING

Defining virtual holonomic constraints

Horizontal circular motion

> $\Phi := \text{Vector}([[\phi1(\theta1)], [\phi2(\theta1)], [\phi3(\theta1)]]) :$

> $D\Phi := \text{map}(\text{diff}, \Phi, \theta1) :$

> $DD\Phi := \text{map}(\text{diff}, D\Phi, \theta1) :$

Vertical circular motion

> $\sigma := \text{Vector}([[\sigma1(\theta2)], [\sigma2(\theta2)], [\sigma3(\theta2)]]) :$

> $D\sigma := \text{map}(\text{diff}, \sigma, \theta2) :$

> $DD\sigma := \text{map}(\text{diff}, D\sigma, \theta2) :$

Straight line motion

> $\epsilon := \text{Vector}([[\epsilon1(\text{xsl})], [\epsilon2(\text{xsl})], [\epsilon3(\text{xsl})]]) :$

> $D\epsilon := \text{map}(\text{diff}, \epsilon, \text{xsl}) :$

> $DD\epsilon := \text{map}(\text{diff}, D\epsilon, \text{xsl}) :$

Transition segment

> $\lambda := \text{Vector}([[\lambda1(\theta t)], [\lambda2(\theta t)], [\lambda3(\theta t)]]) :$

> $D\lambda := \text{map}(\text{diff}, \lambda, \theta t) :$

> $DD\lambda := \text{map}(\text{diff}, D\lambda, \theta t) :$

Inverse Kinematic Calculations

Horizontal circular motion

> $\text{xc1} := 0.65 :; \text{yc1} := 0 :; \text{zc1} := 0.3 :; \text{R1} := 0.2 :$

> $\text{l1} := \sqrt{(\text{d1} - \text{zc1})^2 + (\text{xc1} - \text{a1} + \text{R1}\cdot\cos(\theta1))^2} :$

> $\theta m := \arccos\left(\frac{(\text{a2}^2 + \text{a3}^2 - \text{l1}^2)}{2\cdot\text{a2}\cdot\text{a3}}\right) :$

> $\theta f := \arctan\left(\frac{(\text{d1} - \text{zc1})}{\text{xc1} - \text{a1} + \text{R1}\cdot\cos(\theta1)}\right) :$

> $\theta b := \arcsin\left(\frac{\text{a3}\cdot\sin(\theta m)}{\text{l1}}\right) - \theta f :$

> $\phi1 := \text{evalf}(\arctan(\text{R1}\cdot\sin(\theta1), \text{xc1} + \text{R1}\cdot\cos(\theta1))) :$

> $\phi2 := \text{evalf}\left(\frac{\pi}{2} - \theta b\right) :$

$> \phi 3 := \text{evalf}\left(\frac{\pi}{2} - \theta m\right):$

Vertical circular motion

$> xc2 := 0.45 :; yc2 := 0 :; zc2 := 0.5 :; R2 := 0.2 :$

$> l2 := \sqrt{(xc2)^2 + (R2\cdot\sin(\theta 2))^2} - a1 :$

$> h := R2\cdot\cos(\theta 2) + zc2 - d1 :$

$> \theta m2 := \arccos\left(\frac{(a2^2 + a3^2 - l2^2 - h^2)}{2\cdot a2\cdot a3}\right):$

$> \theta b2 := \arctan\left(\frac{h}{l2}\right):$

$> \theta f2 := \arcsin\left(\frac{a3\cdot\sin(\theta m2)}{\sqrt{l2^2 + h^2}}\right):$

$> \sigma 1 := \text{evalf}(\arctan(R2\cdot\sin(\theta 2), xc2)) :$

$> \sigma 2 := \text{evalf}\left(\frac{\pi}{2} - (\theta b2 + \theta f2)\right):$

$> \sigma 3 := \text{evalf}\left(\frac{\pi}{2} - \theta m2\right):$

Straight line motion

$> l3 := \sqrt{(d1 - zc1)^2 + (x - a1)^2} :$

$> \theta m3 := \arccos\left(\frac{(a2^2 + a3^2 - l3^2)}{2\cdot a2\cdot a3}\right):$

$> \theta f3 := \arctan\left(\frac{(d1 - zc1)}{x - a1}\right):$

$> \theta b3 := \arcsin\left(\frac{a3\cdot\sin(\theta m3)}{l3}\right) - \theta f3 :$

$> \epsilon 1 := \text{evalf}(0) :$

$> \epsilon 2 := \text{evalf}\left(\frac{\pi}{2} - \theta b3\right):$

$> \epsilon 3 := \text{evalf}\left(\frac{\pi}{2} - \theta m3\right):$

Transition segment

$> rt := 0.02 :; c := 2\cdot rt :; \beta := 2\cdot\alpha :;$

$> hts := R1 - \sqrt{R1^2 - \frac{c^2}{4}} :$

$> lt := \sqrt{(d1 - zc1)^2 + (xc2 + hts + rt - a1 + rt\cdot\cos(\theta t))^2} :$

> $\theta m4 := \arccos\left(\frac{(a2^2 + a3^2 - lt^2)}{2 \cdot a2 \cdot a3}\right)$:

> $\theta f4 := \arctan\left(\frac{(d1 - zc1)}{xc2 + hts + rt - a1 + rt \cdot \cos(\theta t)}\right)$:

> $\theta b4 := \arcsin\left(\frac{a3 \cdot \sin(\theta m4)}{lt}\right) - \theta f4$:

>

> $\lambda 1 := \mathrm{evalf}(\arctan(rt \cdot \sin(\theta t), xc2 + hts + rt + rt \cdot \cos(\theta t)) + \arctan(rt, xc2 + hts + rt))$:

> $\lambda 2 := \mathrm{evalf}\left(\frac{\pi}{2} - \theta b4\right)$:

> $\lambda 3 := \mathrm{evalf}\left(\frac{\pi}{2} - \theta m4\right)$:

Plotting the curves for the target trajectories

```
> hc1 := plot(ϕ1(θ1), θ1 =-3.14..3.14, color = "Black") :
> hc2 := plot(ϕ2(θ1), θ1 =-3.14..3.14, color = "Green") :
> hc3 := plot(ϕ3(θ1), θ1 =-3.14..3.14, color = "Blue") :
> vc1 := plot(σ1(θ2), θ2 =-3.14..3.14, color = "Black") :
> vc2 := plot(σ2(θ2), θ2 =-3.14..3.14, color = "Green") :
> vc3 := plot(σ3(θ2), θ2 =-3.14..3.14, color = "Blue") :
> sl1 := plot(ϵ1(x), x = 0.45..0.65, color = "Black") :
> sl2 := plot(ϵ2(x), x = 0.45..0.65, color = "Blue") :
> sl3 := plot(ϵ3(x), x = 0.45..0.65, color = "Green") :
> ts1 := plot(λ1(θt), θt =-3.14..3.14, color = "Black") :
> ts2 := plot(λ2(θt), θt =-3.14..3.14, color = "Green") :
> ts3 := plot(λ3(θt), θt =-3.14..3.14, color = "Blue") :
```

```
> display({hc1, hc2, hc3}) :
> display({vc1, vc2, vc3}) :
> display({sl1, sl2, sl3}) :
> display({ts1, ts2, ts3}) :
```

Generating upper boundary curves

Horizontal circle:

```
> Dϕ1 := diff(ϕ1, θ1) :
> Dϕ2 := diff(ϕ2, θ1) :
> Dϕ3 := diff(ϕ3, θ1) :
```

```
> DDϕ1 := diff(Dϕ1, θ1) :
```

> $DD\phi 2 := \mathrm{diff}(D\phi 2, \theta 1):$

> $DD\phi 3 := \mathrm{diff}(D\phi 3, \theta 1):$

> $absD\phi 1 := \mathrm{abs}(D\phi 1):$

> $absD\phi 2 := \mathrm{abs}(D\phi 2):$

> $absD\phi 3 := \mathrm{abs}(D\phi 3):$

> $s1hc := \left(\frac{3.4907}{absD\phi 1}\right):$

> $s2hc := \left(\frac{3.4907}{absD\phi 2}\right):$

> $s3hc := \left(\frac{4.5379}{absD\phi 3}\right):$

Vertical circle:

> $D\sigma 1 := \mathrm{diff}(\sigma 1, \theta 2):$

> $D\sigma 2 := \mathrm{diff}(\sigma 2, \theta 2):$

> $D\sigma 3 := \mathrm{diff}(\sigma 3, \theta 2):$

> $DD\sigma 1 := \mathrm{diff}(D\sigma 1, \theta 2):$

> $DD\sigma 2 := \mathrm{diff}(D\sigma 2, \theta 2):$

> $DD\sigma 3 := \mathrm{diff}(D\sigma 3, \theta 2):$

> $absD\sigma 1 := \mathrm{abs}(D\sigma 1):$

> $absD\sigma 2 := \mathrm{abs}(D\sigma 2):$

> $absD\sigma 3 := \mathrm{abs}(D\sigma 3):$

> $s1vc := \left(\frac{3.4907}{absD\sigma 1}\right):$

> $s2vc := \left(\frac{3.4907}{absD\sigma 2}\right):$

> $s3vc := \left(\frac{4.5379}{absD\sigma 3}\right):$

Straight line:

> $D\epsilon 1 := \mathrm{diff}(\epsilon 1, x):$

> $D\epsilon 2 := \mathrm{diff}(\epsilon 2, x):$

> $D\epsilon 3 := \mathrm{diff}(\epsilon 3, x):$

> $DD\epsilon 1 := \mathrm{diff}(D\epsilon 1, x):$

> $DD\epsilon 2 := \mathrm{diff}(D\epsilon 2, x):$

> $DD\epsilon 3 := \mathrm{diff}(D\epsilon 3, x):$

> $absD\epsilon 1 := \mathrm{abs}(D\epsilon 1):$

> absDϵ2 := abs(Dϵ2) :

> absDϵ3 := abs(Dϵ3) :

> $s2sl := \left(\frac{3.4907}{absD\epsilon 2}\right)$:

> $s3sl := \left(\frac{4.5379}{absD\epsilon 3}\right)$:

Transition segment:

> Dλ1 := diff(λ1, θt) :

> Dλ2 := diff(λ2, θt) :

> Dλ3 := diff(λ3, θt) :

> DDλ1 := diff(Dλ1, θt) :

> DDλ2 := diff(Dλ2, θt) :

> DDλ3 := diff(Dλ3, θt) :

> absDλ1 := abs(Dλ1) :

> absDλ2 := abs(Dλ2) :

> absDλ3 := abs(Dλ3) :

> $s1ts := \left(\frac{3.4907}{absD\lambda 1}\right)$:

> $s2ts := \left(\frac{3.4907}{absD\lambda 2}\right)$:

> $s3ts := \left(\frac{4.5379}{absD\lambda 3}\right)$:

Plotting boundary curves used to specify interpolation points

> HCboundry1 := plot(s1hc(θ1), θ1 =−π..π, 3 ..12, color = "Black") :

> HCboundry2 := plot(s2hc(θ1), θ1 =−π..π, 3 ..12, color = "Green") :

> HCboundry3 := plot(s3hc(θ1), θ1 =−π..π, 3 ..12, color = "Blue") :

> VCboundry1 := plot(s1vc(θ2), θ2 =−π..π, 3 ..16, color = "Black") :

> VCboundry2 := plot(s2vc(θ2), θ2 =−π..π, 3 ..16, color = "Green") :

> VCboundry3 := plot(s3vc(θ2), θ2 =−π..π, 3 ..16, color = "Blue") :

> SLboundry2 := plot(s2sl(x), x = 0.45 ..0.65, 0 ..3, color = "Green") :

> SLboundry3 := plot(s3sl(x), x = 0.45 ..0.65, 0 ..3, color = "Blue") :

> TSboundry1 := plot(s1ts(θt), θt =−π..π, 3 ..100, color = "Black") :

> TSboundry2 := plot(s2ts(θt), θt =−π..π, 3 ..100, color = "Green") :

> TSboundry3 := plot(s3ts(θt), θt =−π..π, 3 ..100, color = "Blue") :

```
> display( {HCboundry1, HCboundry2, HCboundry3} ) :
> display( {VCboundry1, VCboundry2, VCboundry3} ) :
> display( {SLboundry2, SLboundry3} ) :
> display( {TSboundry1, TSboundry2, TSboundry3} ) :
>
```

Defining points and generating interpolating curve

(change the values of the points to change velocity profiles)

Horizontal circle:

```
> pointsHC := [[-3.14159, 7.6], [-2.7, 8.7], [-2.3, 8.5], [-1.8, 6.6], [-1, 5.4], [-0.35, 7.1],
      [0, 10], [0.35, 7.1], [1, 5.4], [1.8, 6.6], [2.3, 8.5], [2.7, 8.7], [3.14159, 7.6]] :
> InterpolyHC := CurveFitting[Spline](pointsHC, θ1, endpoints='natural') :

> HCbound1 := plot(s1hc(θ1), θ1 =-π..π,  3 ..12, color = "DarkCyan") :
> HCbound2 := plot(s2hc(θ1), θ1 =-π..π, 3 ..12, color = "DarkCyan") :
> HCbound3 := plot(s3hc(θ1), θ1 =-π..π, 3 ..12, color = "DarkCyan") :
```

Vertical circle:

```
> pointsVC := [[-3.14159, 7.55], [-2.6, 9], [-1.8, 11.5], [-1.3, 12.3], [-0.6, 9.6], [0, 7.7],
      [0.6, 9.6], [1.3, 12.3], [1.8, 11.5], [2.6, 9], [3.14159, 7.55]] :
> InterpolyVC := CurveFitting[Spline](pointsVC, θ2, endpoints='natural') :

> VCbound1 := plot(s1vc(θ2), θ2 =-π..π,  3 ..12, color = "DarkCyan") :
> VCbound2 := plot(s2vc(θ2), θ2 =-π..π, 3 ..12, color = "DarkCyan") :
> VCbound3 := plot(s3vc(θ2), θ2 =-π..π, 3 ..12, color = "DarkCyan") :
```

Straight line:

```
> pointsSLv := [[0.45, 0.001], [0.48, 1.4], [0.5, 1.5], [0.65, 1.22]] :
> pointsSLf := [[0.45, 1.55], [0.55, 1.42], [0.65, 1.22]] :
> pointsSLs := [[0.45, 0.001], [0.48, 1.4], [0.5, 1.5], [0.55, 1.4], [0.6, 1.35], [0.62, 1.18],
      [0.65, 0.001]] :
> pointsSLr := [[0.45, 1.55], [0.55, 1.42], [0.6, 1.35], [0.62, 1.18], [0.65, 0.001]] :
> InterpolySL := CurveFitting[Spline](pointsSLf, x, endpoints='natural') :

> SLbound2 := plot(s2sl(x), x = 0.45 ..0.65, 0 ..3, color = "DarkCyan") :
> SLbound3 := plot(s3sl(x), x = 0.45 ..0.65, 0 ..3, color = "DarkCyan") :
```

Transition segment:

```
> pointsTS := [[-3.14, 0], [-2.7, 8.7], [-2.3, 8.5], [-1.8, 6.6], [-1, 5.4], [-0.35, 7.1], [0, 10],
      [0.35, 7.1], [1, 5.4], [1.8, 6.6], [2.3, 8.5], [2.7, 8.7], [3.14, 0]] :
> InterpolyTS := CurveFitting[Spline](pointsTS, θt, endpoints='natural') :

> TSbound1 := plot(s1ts(θt), θt =-π..π,  3 ..12, color = "DarkCyan") :
```

```
> TSbound2 := plot(s2ts(θt), θt =-π..π, 3 ..12, color = "DarkCyan") :
> TSbound3 := plot(s3ts(θt), θt =-π..π, 3 ..12, color = "DarkCyan") :
```

Plotting velocity profiles with boundary curves

```
> HCSplineplot := plot(InterpolyHC, θ1 =-3.14 ..3.14, 3 ..12, color = "DarkRed") :
> VCSplineplot := plot(InterpolyVC, θ2 =-3.14 ..3.14, 7 ..17, color = "DarkRed") :
> SLSplineplot := plot(InterpolySL, x = 0.45 ..0.65, 0 ..3, color = "DarkRed") :
> TSSplineplot := plot(InterpolyTS, θt =-3.14 ..3.14, 3 ..12, color = "DarkRed") :

> display( {HCbound1, HCbound2, HCbound3, HCSplineplot}) :
> display( {VCbound1, VCbound2, VCbound3, VCSplineplot}) :
> display( {SLbound2, SLbound3, SLSplineplot}) :
> display( {TSbound1, TSbound2, TSbound3, TSSplineplot}) :
```

Calculating planned period of motion along the circle trajectory

Horizontal circle:

> HCTfShading := plot$\left(\frac{1}{\text{InterpolyHC}}\right.$, θ1 =-π..π, 0 ..0.2, filled = true, color = orange, transparency = 0.8$\left.\right)$:

> HCTfBoarder := plot$\left(\frac{1}{\text{InterpolyHC}}\right.$, θ1 =-π..π, 0 ..0.2 , color = black, thickness = 2$\left.\right)$:

> display([HCTfBoarder, HCTfShading]) :

> TfHC := evalf$\left(\text{int}\left(\frac{1}{\text{InterpolyHC}}, θ1 = -3.14159 ..3.14159\right)\right)$

$$TfHC := 0.9178941914 \quad \textbf{(1)}$$

Vertical circle:

> VCTfShading := plot$\left(\frac{1}{\text{InterpolyVC}}\right.$, θ2 =-π..π, 0 ..0.2, filled = true, color = orange, transparency = 0.8$\left.\right)$:

> VCTfBoarder := plot$\left(\frac{1}{\text{InterpolyVC}}\right.$, θ2 =-π..π, 0 ..0.2 , color = black, thickness = 2$\left.\right)$:

> display([VCTfBoarder, VCTfShading]) :

> TfVC := evalf$\left(\text{int}\left(\frac{1}{\text{InterpolyVC}}, θ2 = -3.14159 ..3.14159\right)\right)$

$$TfVC := 0.6407211965 \quad \textbf{(2)}$$

Straight line:

> $SLTfShading := plot\left(\frac{1}{InterpolySL}, x = 0.45..0.65, filled = true, color = orange, transparency = 0.8\right):$

> $SLTfBoarder := plot\left(\frac{1}{InterpolySL}, x = 0.45..0.65, color = black, thickness = 2\right):$

> display([SLTfBoarder, SLTfShading]) :

> $TfSL := evalf\left(int\left(\frac{1}{InterpolySL}, x = 0.45..0.65\right)\right)$

$$TfSL := 0.1428386517 \quad (3)$$

Transition segment:

> $TSTfShading := plot\left(\frac{1}{InterpolyTS}, \theta t = -\pi..\pi, 0..0.2, filled = true, color = orange, transparency = 0.8\right):$

> $TSTfBoarder := plot\left(\frac{1}{InterpolyTS}, \theta t = -\pi..\pi, 0..0.2, color = black, thickness = 2\right):$

> display([TSTfBoarder, TSTfShading]) :

> $TfTS := evalf\left(int\left(\frac{1}{InterpolyTS}, \theta t = -3.141592654..3.141592654\right)\right):$

Converting results to matlab code

Horizontal circle:

> #Matlab(ϕ1, resultname = "phi1");

> #Matlab(ϕ2, resultname = "phi2");

> #Matlab(ϕ3, resultname = "phi3");

> #Matlab(Dϕ1, resultname = "dphi1");

> #Matlab(Dϕ2, resultname = "dphi2");

> #Matlab(Dϕ3, resultname = "dphi3");

> #Matlab(DDϕ1, resultname = "ddphi1");

> #Matlab(DDϕ2, resultname = "ddphi2");

> #Matlab(DDϕ3, resultname = "ddphi3");

Vertical circle:

> #Matlab(σ1, resultname = "sigma1");

> #Matlab(σ2, resultname = "sigma2");

> #Matlab(σ3, resultname = "sigma3");

> #Matlab(Dσ1, resultname = "dsigma1");

> #Matlab(Dσ2, resultname = "dsigma2");

> #Matlab(Dσ3, resultname = "dsigma3");

> #Matlab(DDσ1, resultname = "ddsigma1");

> #Matlab(DDσ2, resultname = "ddsigma2");

```
> #Matlab(DDσ3, resultname = "ddsigma3");
```

Straight line:

```
> #Matlab(ϵ1, resultname = "epsilon1");
> #Matlab(ϵ2, resultname = "epsilon2");
> #Matlab(ϵ3, resultname = "epsilon3");
> #Matlab(Dϵ1, resultname = "depsilon1");
> #Matlab(Dϵ2, resultname = "depsilon2");
> #Matlab(Dϵ3, resultname = "depsilon3");
> #Matlab(DDϵ1, resultname = "ddepsilon1");
> #Matlab(DDϵ2, resultname = "ddepsilon2");
> #Matlab(DDϵ3, resultname = "ddepsilon3");
```

Transition segment:

```
> #Matlab(λ1, resultname = "lambda1");
> #Matlab(λ2, resultname = "lambda2");
> #Matlab(λ3, resultname = "lambda3");
> #Matlab(Dλ1, resultname = "dlambda1");
> #Matlab(Dλ2, resultname = "dlambda2");
> #Matlab(Dλ3, resultname = "dlambda3");
> #Matlab(DDλ1, resultname = "ddlambda1");
> #Matlab(DDλ2, resultname = "ddlambda2");
> #Matlab(DDλ3, resultname = "ddlambda3");
```