Run

Run

```
BeginPackage["OneCyclesNew`"];

MC::usage = "the code for a singular knot";
Pp::usage = "Pp[a,b] is a positive crossing from point a to point b along the knot";
Pm::usage = "Pm[a,b] is a negative crossing from point a to point b along the knot";
Po::usage = "Po[a,b] is a singular (double) point from point a to point b along the knot"
Crossings::usage = "returns the sum of the number of multiple points and crossings";
IsCrossing::usage = "IsCrossing[C,a,b] tells if (a,b) or (b,a) is a chord in C and gives i
IsDirectedCrossing::usage = "IsDirectedCrossing[C,a,b] tells if (a,b) is a chord in C";
CrossingDirection::usage = "CrossingDirection[C,a,b] returns
+1 if (a,b) is a chord in C
-1 if (b,a) is a chord in C
0 if neither (a,b) nor (b,a) are chords in C";
CrossingPosition::usage = "CrossingPosition[C,a,b] returns the position of given crossing
CrossingType::usage = "returns 0 or 1 according to direction of chord as in Thomas Fiedle
0 if i<j in crossing [i,j]
1 if j>i in crossing [i,j] ";
W1::usage="return the linear weight of a crossing as in Thomas Fiedler's definition";
W2::usage="return the quadratic weight of a crossing as in Thomas Fiedler's definition";
FindSingR3ThruArc::usage = "for each triple point gives the numbers of the three chords i
i.e. gives all i<j<k that form a R3 and its global type, e.g.{{{2,7,11},{4,9,10}},{\"R1\"
FindSingR3::usage = "Gives the positions in mc array, the l,m,t strands, the global and lo
FindR2::usage = "Gives the numbers and the parallel/crossing type";

EndPackage[]
```

```
BeginPackage["OneCyclesNew`"];
Begin["`Private`"]

MC[mc_MC] := mc

Crossings[mc_MC] := Max @@ Max @@@ mc;

IsCrossing[mc_MC, p1_, p2_] := MemberQ[mc, (Pp | Pm | Po)[p1, p2] | _[p2, p1]];

IsDirectedCrossing[mc_MC, p1_, p2_] := MemberQ[mc, (Pp | Pm | Po)[p1, p2]];

CrossingDirection[mc_MC, p1_, p2_] := Which[
   IsDirectedCrossing[mc, p1, p2], 1,
   IsDirectedCrossing[mc, p2, p1], -1,
   True, 0];

CrossingPosition[mc_MC, p1_, p2_] :=
  Position[mc, (Pp | Pm | Po)[p1, p2] | _[p2, p1]]⟦1, 1⟧;

CrossingType[mc_MC, p1_, p2_] := Module[
 (*TO DO: give name to crossings and invoke this with crossing name*)
 (*TO DO: choose output for "not a crossing" *)
 {a, b, i, outp},
 If[p1 > p2, b = p1; a = p2;, a = p1; b = p2;];
 i = 0;
 While[i++ < Length[mc], (*Print[i,mc[[i]][[1]]];*)
  Which[mc[[i]][[1]] === a && mc[[i]][[2]] === b, (*Print["** 1 **"];*)outp = 1;
      i = 2 * Length[mc]; Break[];,
  mc[[i]][[1]] === b && mc[[i]][[2]] === a, (*Print["** 0 **"];*)outp = 0;
      i = 2 * Length[mc]; Break[];,
  True,];
 ];
 outp
];

W1[mc_MC, p1_, p2_] := Total[
   Cases[mc, (P_)[i_, j_] /; i > p2 && p1 < j ≤ p2 ⧴ P] /. {Pp → 1, Pm → -1, Po → 0}
  ];

W2[mc_MC, p1_, p2_] := Module[
(*TO DO: give name to crossings and invoke this with crossing name*)
 {outp},
 outp = 0;
 If[CrossingType[mc, p1, p2] == 1, ,
  Module[
   {a, b, i},
   If[p1 > p2, b = p1; a = p2;, a = p1; b = p2;];
   i = 0;
   While[i++ < Length[mc],
    If[(mc[[i]][[1]] < a) && (mc[[i]][[2]] != b) &&
          (CrossingType[mc, mc[[i]][[1]], mc[[i]][[2]]] == 1), (*check BBB*)
(*last condition not needed since will have W1=0*)
     If[Head[mc[[i]]] === Pp,
      outp = outp + W1[mc, mc[[i]][[1]], mc[[i]][[2]]];,
       outp = outp - W1[mc, mc[[i]][[1]], mc[[i]][[2]]];
      ]
(*Print["i:",i," out:",outp];*)
```

```
    ](*end if*)
(*Print[outp];*)
   ];(*end while*)
  ];(*end module*)
 ]; (*end if*)outp
];

End[];
EndPackage[];
```

```
OneCyclesNew`Private`
```

```
BeginPackage["OneCyclesNew`"];
Begin["`Private`"]

FindSingR3ThruArc[mc_MC] := Module[
 (*gives all i<j<k that form a R3 and its global type,
   e.g.{{{2,7,11},{4,9,10}},{"R1","L2"}}*)
 {i, j, k, cr, outp, outtypes},
 outp = {};
 outtypes = {};
 (*Print[outp===Null];*)
 cr = Crossings[mc];
 i = 0;
 While[i ++ < cr, j = i;
  While[j ++ < cr, k = j;
   While[k ++ < cr,
    Which[CrossingDirection[mc, i, j] == 1 && CrossingDirection[mc, j, k] == -1 &&
          CrossingDirection[mc, k, i] == 1, outp = Append[outp, {i, j, k}];
     outtypes = Append[outtypes, "R1"];, CrossingDirection[mc, i, j] == -1 &&
          CrossingDirection[mc, j, k] == 1 &&
          CrossingDirection[mc, k, i] == 1, outp = Append[outp, {i, j, k}];
     outtypes = Append[outtypes, "R2"];, CrossingDirection[mc, i, j] == 1 &&
          CrossingDirection[mc, j, k] == 1 &&
          CrossingDirection[mc, k, i] == -1, outp = Append[outp, {i, j, k}];
     outtypes = Append[outtypes, "R3"];, CrossingDirection[mc, i, j] == 1 &&
          CrossingDirection[mc, j, k] == -1 &&
          CrossingDirection[mc, k, i] == -1, outp = Append[outp, {i, j, k}];
     outtypes = Append[outtypes, "L1"];, CrossingDirection[mc, i, j] == -1 &&
          CrossingDirection[mc, j, k] == -1 &&
          CrossingDirection[mc, k, i] == 1, outp = Append[outp, {i, j, k}];
     outtypes = Append[outtypes, "L2"];, CrossingDirection[mc, i, j] == -1 &&
          CrossingDirection[mc, j, k] == 1 &&
          CrossingDirection[mc, k, i] == -1, outp = Append[outp, {i, j, k}];
```

```
      outtypes = Append[outtypes, "L3"];, True,
     ];(*end which*)
    ](*end while*)
   ](*end while*)
  ]; (*end while*)
  outp = DeleteCases[outp, Null, 3];
  {outp, outtypes}
 ];

 FindSingR2[mc_MC] := Module[
  {i, j, cr, outp},
  outp = {};
  (*Print[outp===Null];*)cr = Crossings[mc];
  i = 0;
  While[i++ < cr, j = i;
   While[j++ < cr,
    Which[IsCrossing[mc, i, j] && IsCrossing[mc, i + 1, j + 1],
        (*and none is a triple pt!*)
       outp = Append[outp, {i, j, i + 1, j + 1}];
       Print["crossing at ", i, j];,
       IsCrossing[mc, i, j] && IsCrossing[mc, i + 1, j - 1],
         (*and none is a triple pt!!*)
       outp = Append[outp, {i, j, i + 1, j - 1}];
       Print["-crossing at ", i, j];, True,
       ];
   ](*end while*)
  ]; (*end while*)
  (*outp=DeleteCases[outp,Null,3];*)
  outp
 ];

 FindSingR3[mc_MC] := Module[
  (*gives all i<j<k that form a R3 and its global type,
    local type, low, middle and top strand*)
  {leng, i, j, k, l, m, t, globaltype, localtype},
  leng = Length[mc];
  i = 0;
  While[i++ < leng,
     j = 0;
     While[j++ < leng,
        If[mc[[i]][[2]] == mc[[j]][[1]],
          k = 0;
          While[k++ < leng,
If[mc[[j]][[2]] == mc[[k]][[2]] && mc[[i]][[1]] == mc[[k]][[1]],
  l = mc[[i]][[1]];
  m = mc[[i]][[2]];
  t = mc[[k]][[2]];
  If[Head[mc[[i]]] == Pp,
   If[Head[mc[[j]]] == Pp,
     If[Head[mc[[k]]] == Pp,
      localtype = 1, (* +++ *)
      localtype = 6 (* ++- *)
      ],
     If[Head[mc[[k]]] == Pp,
      localtype = 7, (* +-+ *)
      localtype = 4 (* +-- *)
      ]
     ],
   If[Head[mc[[j]]] == Pp,
     If[Head[mc[[k]]] == Pp,
      localtype = 5, (* -++ *)
      localtype = 3 (* -+- *)
      ],
     If[Head[mc[[k]]] == Pp,
      localtype = 2, (* --+ *)
      localtype = 8 (* --- *)
      ]
     ]
   ];
  (*If[mc[[k]][[1]]<mc[[k]][[2]] &&
  mc[[i]][[2]]>mc[[k]][[1]] && mc[[i]][[2]]<mc[[k]][[2]]||
       mc[[k]][[1]]>mc[[k]][[2]] && (mc[[i]][[2]]<mc[[k]][[2]] ||
    mc[[i]][[2]]>mc[[k]][[1]]), (* global type R *)
       globaltype = "R",
       globaltype = "L"
       ];*)
  If[mc[[k]][[1]] < mc[[k]][[2]],
   If[mc[[i]][[1]] < mc[[i]][[2]], globaltype = "R3",

If[mc[[j]][[1]] > mc[[j]][[2]], globaltype = "L1", globaltype = "L3"]
     ],
   If[mc[[i]][[1]] < mc[[i]][[2]], globaltype = "R2",

If[mc[[j]][[1]] < mc[[j]][[2]], globaltype = "R1", globaltype = "L2"]
     ]
   ];
```

```
              Print["Position in mc: (", i, ",", j, ",", k, "), l,m,t:(", l,
            ",", m, ",", t, "), global ", globaltype, " local ", localtype];
              ](*if i,j,k*)
            ](*while k*)
          ](*if i,j*)
        ](*while j*)
      ] (*while i*)
];

FindR2[mc_MC] := Module[
 {i, j, leng},
 leng = Length[mc];
 i = 0;
 While[i++ < leng - 1,
    j = 0;
    While[j++ < leng - 1,
      If[mc[[i]][[1]] + 1 == mc[[j]][[1]],
        Which[
          mc[[i]][[2]] == mc[[j]][[2]] + 1, Print["parallel ", mc[[i]][[1]],
        ",", mc[[i]][[2]], ",", mc[[j]][[1]], ",", mc[[j]][[2]] ],
          mc[[i]][[2]] + 1 == mc[[j]][[2]], Print["crossing ", mc[[i]][[1]],
        ",", mc[[i]][[2]], ",", mc[[j]][[1]], ",", mc[[j]][[2]] ],
          True,]
        ]
      ]
    ]
];

End[];
EndPackage[];
OneCyclesNew`Private`

meight := MC[Pp[6, 1], Pp[2, 5], Pm[4, 7], Pm[8, 3]];
mtest1 := MC[Pp[1, 3], Pm[5, 2], Pp[8, 3], Pm[6, 4], Pp[7, 9]];
mtest2 := MC[Pp[3, 1], Pp[2, 7], Pp[10, 4], Pp[9, 5], Pp[6, 8], Pp[11, 7], Pp[10, 9]];
mtest3 := MC[Pp[3, 1], Pp[2, 7], Pp[10, 4],
   Pp[9, 5], Pp[6, 8], Pp[11, 7], Pp[10, 9], Pp[11, 2], Pp[4, 9]];
mtest4 := MC[Pp[3, 1], Pp[2, 7], Pp[10, 4], Pp[9, 5], Pp[6, 8],
   Pp[11, 7], Pp[10, 9], Pp[11, 2], Pp[9, 4]];
mtest5 := MC[Pp[1, 3], Pp[2, 7], Pp[10, 4], Pp[9, 5], Pp[8, 6],
   Pp[11, 7], Pp[10, 9], Pp[11, 2], Pp[9, 4]];
```

```
mtest4

MC[Pp[3, 1], Pp[2, 7], Pp[10, 4], Pp[9, 5],
 Pp[6, 8], Pp[11, 7], Pp[10, 9], Pp[11, 2], Pp[9, 4]]

FindSingR3[mtest4]

Position in mc: (7,9,3), l,m,t:(10,9,4), global L2 local 1

Position in mc: (8,2,6), l,m,t:(11,2,7), global R1 local 1

mtest2

MC[Pp[3, 1], Pp[2, 7], Pp[10, 4], Pp[9, 5], Pp[6, 8], Pp[11, 7], Pp[10, 9]]

Crossings[mtest2]

11

CrossingSign[mc_MC, p1_, p2_] := Module[
   {i, outp},
   outp = 0;
   i = 0;
   While[i++ < Length[mc], (*Print[i,mc[[i]][[1]]];*)
    If[mc[[i]][[1]] === p1 && mc[[i]][[2]] === p2 ||
       mc[[i]][[1]] === p2 && mc[[i]][[2]] === p1,
      If[Head[mc[[i]]] === Pp, outp = 1;
        Break[];, outp = -1; Break[];];
     ];
   ];
   outp
  ];

CrossingSign[mtest1, 2, 5]

-1

Head[mtest1[[1]]] === Pp

True

FindSingR3[mtest4]

Position in mc: (7,9,3), l,m,t:(10,9,4), global L2 local 1

Position in mc: (8,2,6), l,m,t:(11,2,7), global R1 local 1

FindR2[mtest5]

parallel 9,5,10,4

parallel 8,6,9,5
```

```
FindR3[mc_MC] := Module[
   {i, j, k},
   i = 0;
   While[i++ < Length[mc],
    j = i;
    While[j++ < Length[mc],
     k = j;
     While[k++ < Length[mc],
      Which[mc[[i]][[1]] + 1 == mc[[j]][[1]],
       Which[mc[[i]][[2]] + 1 == mc[[k]][[1]],
        Which[mc[[j]][[2]] + 1 == mc[[k]][[2]], Print["ij-ik-jk"];,
         mc[[j]][[2]] - 1 == mc[[k]][[2]], Print["ij-ik-kj"];], (*end which*)
        mc[[i]][[2]] - 1 == mc[[k]][[1]],
        Which[mc[[j]][[2]] + 1 == mc[[k]][[2]], Print["ij-ki-jk"];,
         mc[[j]][[2]] - 1 == mc[[k]][[2]], Print["ij-ki-kj"];](*end which*)
       ], (*end which*)
       mc[[i]][[1]] - 1 == mc[[j]][[1]],
       Which[mc[[i]][[2]] + 1 == mc[[k]][[1]],
        Which[mc[[j]][[2]] + 1 == mc[[k]][[2]], Print["ji-ik-jk"];,
         mc[[j]][[2]] - 1 == mc[[k]][[2]], Print["ji-ik-kj"];], (*end which*)
        mc[[i]][[2]] - 1 == mc[[k]][[1]],
        Which[mc[[j]][[2]] + 1 == mc[[k]][[2]], Print["ji-ki-jk"];,
         mc[[j]][[2]] - 1 == mc[[k]][[2]], Print["ji-ki-kj"];](*end which*)
       ](*end which*)
      ](*end which*)
      (*If[CommonElement[mc[[i]],mc[[j]]] , Print[i,j],]*)

     ](*while k*)
    ](*while j*)
   ](*while i*)
  ]; (* only for all forward!*)
```

```
FindR3[mtest1]
```

```
CommonElement[p1_, p2_] :=
  If[p1[[1]] == p2[[1]] || p1[[1]] == p2[[2]] ||
    p1[[2]] == p2[[1]] || p1[[2]] == p2[[2]], True,];
```

```
Intersection @@@ {mtest3[[3]], mtest3[[7]]}
```

```
Intersection::normal : Nonatomic expression expected at position 1 in 10 ∩ 4. ≫

Intersection::normal : Nonatomic expression expected at position 1 in 10 ∩ 9. ≫

{10⋂4, 10⋂9}
```

```
Intersection @@ {mtest1[[1]], Pm[1, 4]}

Intersection::heads : Heads Pm and Pp at positions 2 and 1 are expected to be the same. >>

Pp[1, 3] ⋂ Pm[1, 4]
```