平成３０年度 共同利用研究報告書

平成　31 年　2 月　4 日

九州大学 マス・フォア・インダストリ研究所長 殿

所属・職名　株式会社エス・イー・エー創研 システム開発チーム 主任
提案者 氏名　山口 大介

下記の通り共同研究の報告をいたします.　記

	※整理番号	20180010		
1.研究計画題目	造船工学における曲面幾何			
2.種目（○で囲む）	a. プロジェクト研究　b. 若手研究　(c.)一般研究			
3.種別（○で囲む）	a. 研究集会Ⅰ　b. 研究集会Ⅱ　(c.)短期共同研究　d. 短期研究員			
4.研究代表者	氏 名	山口 大介		
	所 属 部局名	株式会社エス・イー・エー創研 システム開発チーム	職名	主任
	連絡先			
	e-mail		TEL	
5.研究実施期間	平成　30 年　12 月 25 日（ 火曜日）～平成 30 年 12 月 28 日（ 金曜日）			
6.キーワード （複数可）	曲面幾何 CAD フェアリング 曲面成形 設計最適化			
7.参加者数	17 人　＊1			

＊1　短期研究員は九大の共同研究者も含める.
Ⅰ, Ⅱ, 短期共同研究は事務局から送った参加者データを元に記入.

8.本研究で得られた成果の概要（**成果報告書を別途要添付**　枚数は次頁参照）

本研究期間においては, まず海上技術安全研究所松尾宏平氏らにより提案された新しい外板展開手法である「曲率線展開法」について, 松尾氏から解説を行っていただいた. この解説を元に, 曲率線展開法の評価のための Mathematica 関数群の試作を行い, それらを用いて展開法アルゴリズムの改良について議論した. 成果報告書には, 本研究の成果として, 試作した Mathematica 関数群についての解説を添付する.

九州大学IMI短期共同研究報告
「造船工学における曲面幾何」

山口 大介 * [1], 溝口 佳寛 [2], 濱田 裕康 [3], 松谷 茂樹 [3], 田中 和明 [4], and 松岡 和彦 [5]

[1](株) エス・イー・エー創研・船舶設計システム開発チーム
[2] 九州大学・マス・フォア・インダストリ研究所
[3] 佐世保工業高等専門学校
[4] 九州工業大学・情報工学部
[5] 長崎総合科学大学・工学部

2019 年 2 月 1 日

概 要

船舶は滑らかな曲面で構成される構造物である. そのため, 船舶の設計及び製造とは曲面構造物の設計及び製造と同義であり, 特に船舶にあっては曲面構造物の設計及び製造こそが製品仕様や生産性を決める大きな要因となっている. 近年, 3 次元 CAD 等の実用化により, 3 次元の曲面構造物を高度に取り扱えるようになっているが, これまで 2 次元で行っていた設計, 製造の手法を 3 次元 CAD 上にそのまま再現し業務の効率化を図ったもので, 設計法及び製造法に関するアーキテクチャの革新には至っていない. 本研究は, 3 次元 CAD 等による製品表現技術の高度化に応じて, 曲面構造物の設計と製造に関する数学的解釈の再構築を行うものである. 具体的には以下の課題を対象に, 数学的解釈の再構築テーマを検索する. (1) 曲面の表現方法 (曲面構造物の CAD モデリング手法, 幾何学的表現の高度化 (リーマン幾何学的表現方法) 等), (2) 曲面の設計, 取り扱い方法 (フェアリング, 最適な板割り, 平面展開), (3) 曲面の製造方法 (造船曲り外板加工の幾何学的解釈の再構築), (4) 曲面と物性の関係 (曲面の幾何学的特性と流体力学的, 構造力学的特性. 最適化). 本研究では, 上記の課題を中心に数学的解釈による再構築に適した課題を探求し, 曲面構造物 (特に船舶) の設計及び製造法に関する新アーキテクチャ開発に関する方向性を導くものである.

キーワード: 曲面幾何, CAD, フェアリング, 曲面成形, 設計最適化

1 はじめに

株式会社エス・イー・エー創研らは, 船舶を対象に曲面構造物の設計及び製造に関する研究開発に取り組んでいる. 例えば, 熟練の技と言われる外板曲げ加工 (船舶を構成する曲面外板を平面から工場で加工すること, 「ぎょう鉄」ともいう) に関して, いくつかの現図展開法及び曲げ加工法を提案し, 実際の造船現場にて実用化しているが, 一方で, それらの提案手法に関する数学的解釈については解決されていない課題も残っている (展開基線の最適な選択問題, 提案する成形手法と非可展面形状の一意性に関する問題等). この曲面曲げ加工に関する問題の他, 曲面の 3 次元表現やその取扱い手法に関する研究開発にも取り組んでおり, 数学的解釈の再構築とそれらの実装によって, 造船所の更なる生産性向上を図っていきたい.

*dyamaguchi@sea-soken.co.jp

2　詳細スケジュール

日時: 2018 年 12 月 25 日 (火)〜28 日 (金)

場所: 2018 年 12 月 25 日 (火)〜26 日 (水)
九州大学 伊都キャンパス ウエスト 1 号館 D 棟 4 階 IMI コンファレンスルーム (W1-D-414)
2018 年 12 月 27 日 (木)〜28 日 (金)
九州大学 伊都キャンパス ウエスト 1 号館 C 棟 5 階 C512 中講義室 (W1-C-512)

2018 年 12 月 25 日 (火) [公開プログラム]

14:00 〜 14:05　溝口 佳寛 (九州大学 マス・フォア・インダストリ研究所)
開会の挨拶
14:05 〜 14:35　鍛冶 静雄 (九州大学 マス・フォア・インダストリ研究所)
微分形式を用いたメッシュ形状処理の基本
14:35 〜 15:35　松尾 宏平 (海上・港湾・航空技術研究所 海上技術安全研究所)
造船における曲面幾何に関する研究活動と数学的課題の抽出
15:35 〜 15:50　休憩
15:50 〜 16:20　隅田 大貴 (流体テクノ株式会社)
CFD による船型評価と水槽試験結果の推定について
16:20 〜 16:50　田中 和明 (九州工業大学 情報工学部)
mruby のパフォーマンス向上の報告
16:50 〜 17:20　松岡 和彦 (長崎総合科学大学 工学部工学科 船舶工学コース)
ICT による船舶の外板曲面形状作製
17:20 〜 17:30　連絡

2018 年 12 月 26 日 (水) [非公開プログラム]

10:30 〜 12:00　ワークショップ (1)
12:00 〜 13:00　昼食
13:00 〜 15:00　ワークショップ (2)
15:00 〜 15:15　休憩
15:15 〜 15:35　村里 優太 (佐世保工業高等専門学校 専攻科 2 年)
パルスパワーによる大気圧プラズマの数理解析
15:35 〜 15:55　西山 輝 (佐世保工業高等専門学校 専攻科 2 年)
ロボティクスの物体拘束の数理
15:55 〜 16:15　松藤 ちひろ (九州大学 大学院数理学府修士 2 年)
ポリゴンモデルに対する最適補間アルゴリズムの実現方法について
16:15 〜 16:35　栗原 寛明 (九州大学 大学院数理学府博士 1 年)
松藤 ちひろ (九州大学 大学院数理学府修士 2 年)
Topological Mode Analysis Tool (ToMATo) によるクラスタリング
16:35 〜 16:55　立野 圭一 (九州工業大学 大学院情報工学府修士 2 年)
複数画像間マッチングによる土木施工図面作成に関する研究

2018 年 12 月 27 日 (木) [非公開プログラム]

10:30 〜 12:00　ワークショップ (3)
12:00 〜 13:00　昼食
13:00 〜 15:00　ワークショップ (4)
15:00 〜 15:15　休憩
15:10 〜 17:00　ワークショップ (5)

2018 年 12 月 28 日 (金) [非公開プログラム]

10:30 〜 12:00　まとめ

3　成果

本研究期間においては, まず海上技術安全研究所松尾宏平氏らにより提案された新しい外板展開手法である「曲率線展開法」について, 松尾氏から解説を行っていただいた. この解説を元に, 曲率線展開法の評価のための Mathematica 関数群の試作を行い, それらを用いて展開法アルゴリズムの改良について議論した. 以下には, 本研究の成果として, 試作した Mathematica 関数群についての解説を添付する.

曲率線展開法の評価のためのMathematica関数群の試作について

平成30年度九州大学マス・フォア・インダストリ研究所短期共同利用研究
「造船工学における曲面幾何」(研究代表者 山口大介 (株式会社エス・イー・エー創研))
2018年12月25日～2018年12月28日

2019/1/31 19:00

1. 概要

海上技術安全研究所松尾宏平氏らにより提案された新しい外板展開手法である「曲率線展開法」についての, 性能評価, 精度保証, 手法の改良, 新たな理論展開のために, 海上技術安全研究所よりサンプル外板データ(曲率線展開曲面座標等)を提供頂き, 共同利用研究期間中に討論を行なった. 本項では, そのときに作成した, 数学ソフトウェアMathematicaで実装したデータ処理関数たちを紹介する.

2. データ入力

外板データのうち, 曲率線展開された3次元データ(線分, および, 交差情報), 平面射影された2次元データ, ならびに, 測地線曲率データの読込み関数を実装した. ここでは, 曲率線展開において, 最大曲率の曲率線を「赤線(red)」, 最小曲率の曲率線を「青線(blue)」と呼ぶことにする. サンプルデータは, 25本の赤線と13本の青線からなる. 3次元データ red3d, blue3d, 2次元データ red2d, blue2d, 交差点情報 redx, bluex, および, 曲率線の測地線曲率 redk, bluek を用いる. 以下は, そのサンプルデータを変数に読込む手続きである.

2.1 入力データ変数定義

2.2 入力データの確認

1本の赤線は3次元点列のリストである. red3d は, 赤線たちのリストである. すなわち, red3d[[1]]が1番の赤線の点列リスト, Length[red3d[[1]]]が, 1番の赤線の点列の長さ, Length-[red3d]が, 赤線の本数となる. 赤線と青線の本数, それぞれの構成頂点数の最大数と最小数を以下で確認する.

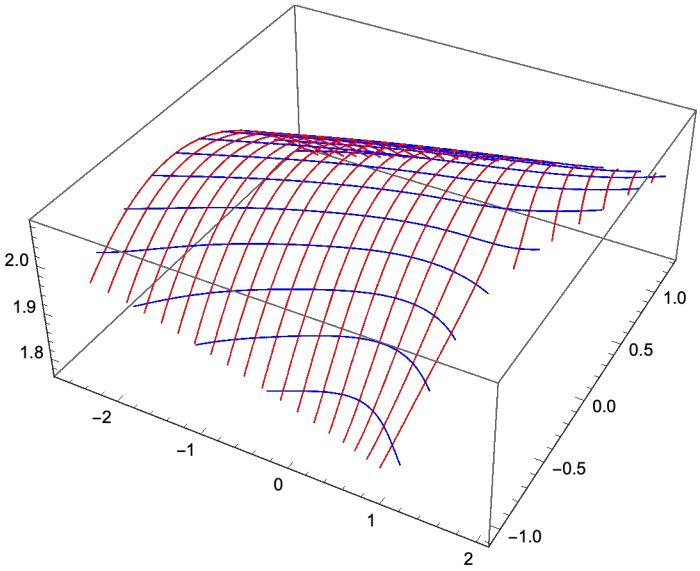
```
In[]:= Length /@ {red3d, blue3d}
Out[]= {25, 13}

In[]:= {Max[#], Min[#]} & [Length /@ #] & /@ {red3d, blue3d}
Out[]= {{614, 49}, {1182, 357}}
```

読み込んだ red3d と blue3d を3次元表示して確認する.

In[ ]:=
```
ListPointPlot3D[{Flatten[red3d, 1], Flatten[blue3d, 1]}, PlotStyle → {Red, Blue}]
```

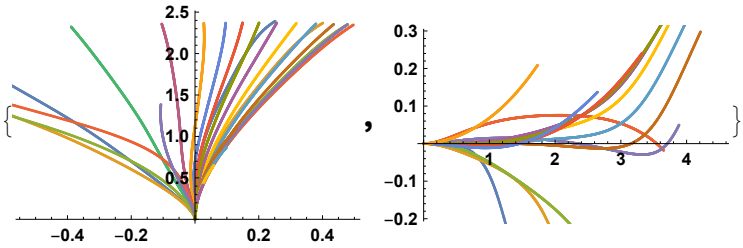
Out[ ]=



読み込んだ red2d と blue2d を重ねて2次元表示して確認する.

In[ ]:=
```
{ListPlot[red2d], ListPlot[blue2d]}
```

Out[ ]=



2. 基本関数

本項で利用する基本関数群を以下に定義する. その機能については, 本文中で説明する.

2.1 関数定義

2.2 基本関数の確認

交差点情報 redx, bluex の書く要素は, 対応する赤線と青線の交差点となる頂点に対応する要素が相手先の線番号になっている. 関数 FindCrossPoint は,交差点リストから必要な情報を引き出す関数である.

In[ ]:=
```
? FindCrossPoint
```

FindCrossPoint[l] 交差点リストに対して相手番号と位置の対のリストを返す.
FindCrossPoint[l,n] 相手番号nとの交差点リストの位置を返す.

1番の赤線(redx[[1]])の交差点の(相手の青線番号, 赤線の頂点番号)の対のリスト

In[ ]:=
```
FindCrossPoint[redx[[1]]]
```

Out[ ]=
```
{{4, 39}, {5, 103}, {6, 167}, {7, 228}, {8, 291},
 {9, 353}, {10, 407}, {11, 464}, {12, 520}, {13, 591}}
```

8番の青線(bluex[[8]])の交差点の(相手の青線番号, 青線の頂点番号)の対のリスト

```
In[]:= FindCrossPoint[bluex[[8]]]
Out[]= {{1, 5}, {2, 53}, {3, 103}, {4, 152}, {5, 199}, {6, 247}, {7, 294},
  {8, 341}, {9, 387}, {10, 434}, {11, 484}, {12, 535}, {13, 582},
  {14, 629}, {15, 676}, {16, 724}, {17, 768}, {18, 813}, {19, 857},
  {20, 900}, {21, 955}, {22, 1006}, {23, 1055}, {24, 1106}, {25, 1154}}
```

8番の青線(bluex[[8]])の1番の赤線との交差点である頂点番号

```
In[]:= FindCrossPoint[bluex[[8]], 1]
Out[]= 5
```

1番の赤線(red[[1]])の8番の赤線との交差点である頂点番号

```
In[]:= FindCrossPoint[redx[[1]], 8]
Out[]= 291
```

上記により, 1番の赤線の291番目の頂点と, 8番の青線の5番目の頂点が交差していることがわかる. 以下は, それぞれの頂点の平面曲線の座標を確認している. 平面曲線は空間曲線を長さを保存して平面内に射影した曲線で始点を原点にしているので, 交点の座標は一致しない.

```
In[]:= {red2d[[1, 291]], blue2d[[8, 5]]}
Out[]= {{0.0458695, 1.13954}, {0.0125344, 2.04957 × 10^-7}}
```

関数 PointArg は, 2次元の曲線点列データと頂点番号を引数とし, その頂点での接線の角度を返す関数である. 具体的には前後5頂点への直線の傾きの平均値を計算している.

```
In[]:= ? PointArg
```

PointArg[l,n] 2D点列リスト l の n 番目の点での接線の角度(ラジアン)を求める.

```
In[]:= ? RotateAngle
```

RotateAngle[x,y] y+a=x−$\frac{\pi}{2}$となる回転角aを求める.

以下は, 1番の赤線の291番目の頂点, 8番の青線の5番目の頂点での接線の傾きをPointArg関数で求め, さらに, この2つの曲線をこれらの頂点で直交させるために必要な回転角をRotateAngleで計算した例である. 確認のため, π/2 の数値も表示している.

```
In[]:= {PointArg[red2d[[1]], 291], PointArg[blue2d[[8]], 5],
  RotateAngle[PointArg[red2d[[1]], 291], PointArg[blue2d[[8]], 5]], π/2 // N}
Out[]= {1.46816, 0.00003919, -0.102672, 1.5708}

In[]:= PointArg[red2d[[1]], 291] - (PointArg[blue2d[[8]], 5] +
    RotateAngle[PointArg[red2d[[1]], 291], PointArg[blue2d[[8]], 5]]) - (π/2 // N)
Out[]= 0.
```

3. 曲率線展開法

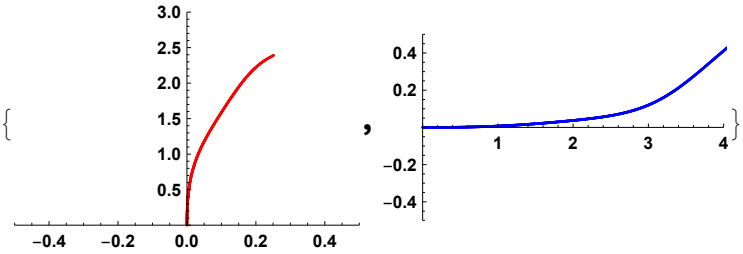
本節では実装関数とサンプルデータを用いて, 曲率線展開法の手順をひとつずつ実行する.

3.1 平面曲線の回転と接続

2.2節で確認した交差する1番の赤線と8番の青線を直交するように回転して, 交差点が一致するように平行移動する. まずは, 移動前の赤線と青線をグラフ表示する.

In[•]:=
```
{ListPlot[red2d[[1]], PlotRange → {{-0.5, 0.5}, {0, 3}}, PlotStyle → Red],
 ListPlot[blue2d[[8]], PlotRange → {{0, 4}, {-0.5, 0.5}}, PlotStyle → Blue]}
```

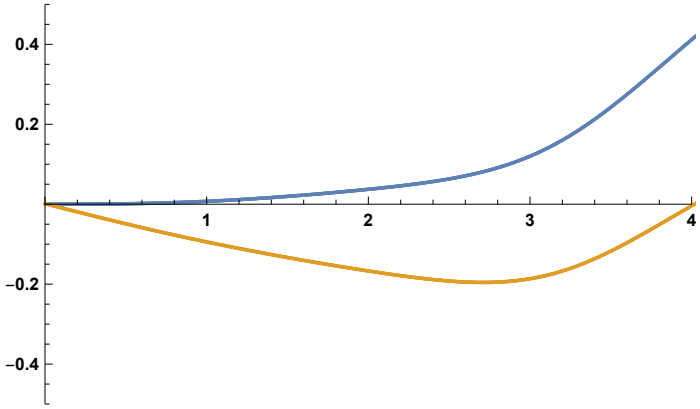
Out[•]=



8番の青線をRotateAngle関数で求めた直交させるために必要な角度だけ回転させる. 回転の中心は交点blue2d[[8,5]]になるので, 点列を平行移動して, 回転行列を掛けて回転する. ここでは, 再び, 平行移動して, 回転の様子を表示している.

In[•]:=
```
ListPlot[
 {blue2d[[8]], (RotationMatrix[RotateAngle[PointArg[red2d[[1]], 291], PointArg[
          blue2d[[8]], 5]]].(# - blue2d[[8, 5]]) + blue2d[[8, 5]] & /@
   blue2d[[8]])}, PlotRange → {{0, 4}, {-0.5, 0.5}}]
```

Out[•]=



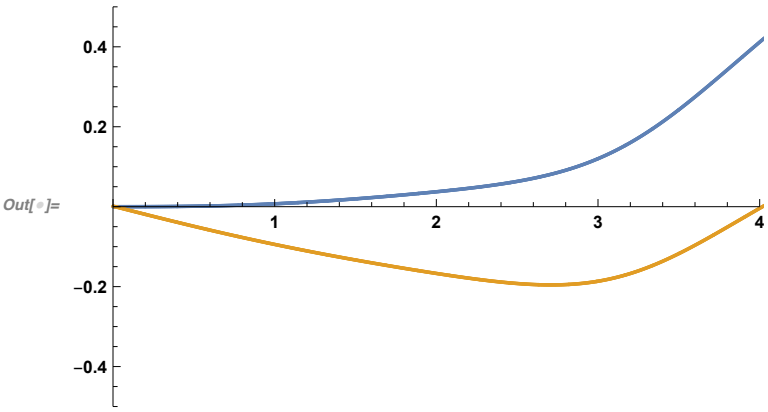
同様の操作を関数 RotateLine を使って実行すると以下のようになる.

In[•]:=
```
? RotateLine
```

RotateLine[l,n,a,p] 2D点列リストlのn番目の点を点pへ平行移動し角度aの回転を行う.

In[]:=
```
ListPlot[{blue2d[[8]], RotateLine[blue2d[[8]], 5,
    RotateAngle[PointArg[red2d[[1]], 291], PointArg[blue2d[[8]], 5]],
    blue2d[[8, 5]]]}, PlotRange → {{0, 4}, {-0.5, 0.5}}]
```

Out[]=



8番の青線をRotateAngle関数で求めた直交させるために必要な角度だけ回転させる. 回転の中心は交点blue2d[[8,5]]になるので, 点列を平行移動して, 回転行列を掛けて回転する. ここでは, 再び, 平行移動して, 回転の様子を表示している.

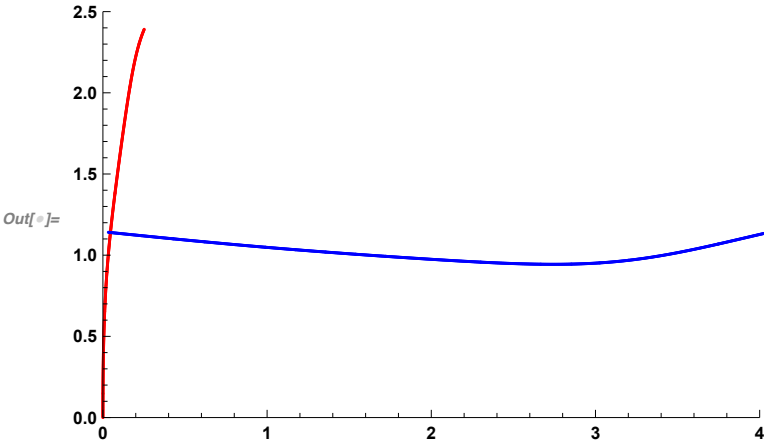
In[]:=
```
? RotateLine2
```

RotateLine2[l1,n1,l2,n2]
　2D点列リストl1のn1番目で2D点列リストl2のn2番目が直交するようにl2を回転した2D点列リストを返す.
RotateLine2[rn,bn,red2d,redx,blue2d,bluex]
　2D点列リストred2d[[rn]]とblue2d[[bn]]の交差点で直交するようにblue2d[[bn]]を回転し平行移動した2D点列リストを返す.

以下は, 1番の赤線と8番の青線を交点で直交するように回転して表示した例である. 関数RotateLine2 は, 交差する赤線と青線の番号を与えると交点で直交するように回転した青線の点列データを返す関数である.

In[]:=
```
ListPlot[{red2d[[1]], RotateLine2[1, 8, red2d, redx, blue2d, bluex]},
 PlotRange → {{0, 4}, {0, 2.5}}, PlotStyle → {Red, Blue}]
```

Out[]=



次は, 8番の青線の53番目の頂点が2番の赤線の298番の頂点と交差することを確認し, 必要な回

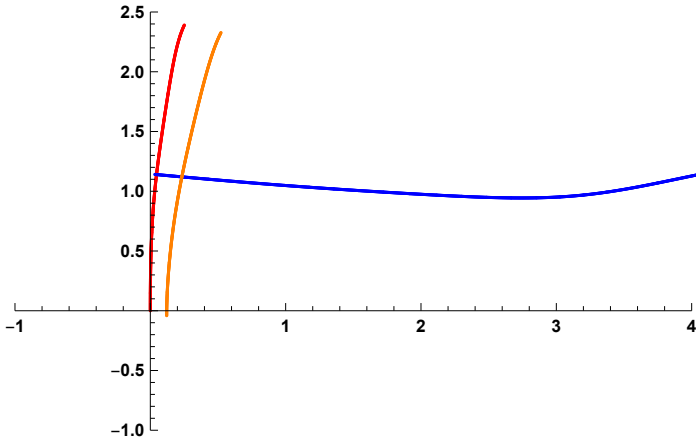
転と平行移動を施し, 8番の青線に2番の赤線を直交するように接続する.

In[•]:=
```
{FindCrossPoint[bluex[[8]], 2], FindCrossPoint[redx[[2]], 8]}
```

Out[•]=
```
{53, 298}
```

In[•]:=
```
ListPlot[{red2d[[1]], RotateLine2[1, 8, red2d, redx, blue2d, bluex],
  RotateLine[red2d[[2]], 298, 0,
   RotateLine2[1, 8, red2d, redx, blue2d, bluex][[53]]]},
 PlotRange → {{-1, 4}, {-1, 2.5}}, PlotStyle → {Red, Blue, Orange}]
```

Out[•]=



3.2 接続する曲率線の選択

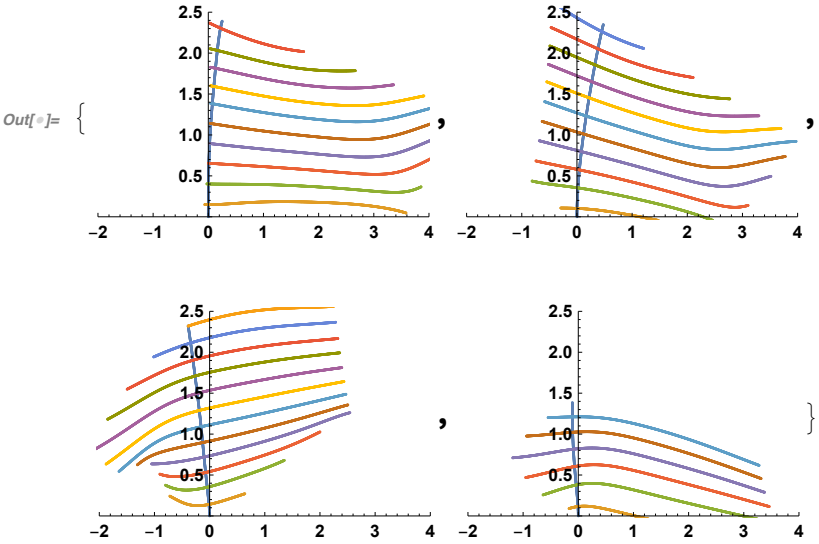
3.1節の操作を繰り返し, 赤線と青線を直交するように接続して, 曲率線展開が完成する. ここでの課題は, 2つの赤線の接続に利用する青線は複数あることである. どの青線で接続すると効率が良いか, 工程数を少なく曲面へのぎょう鉄作業が行えるかということである. そのための性能評価, 精度保証, 手法の改良, 新たな理論展開を行うことが本共同研究の主課題であり, そのための解析ツールとして, 本Mathematica関数群を開発した.

In[•]:=
```
Figure1[i_, red2d_, redx_, blue2d_, bluex_] :=
  ListPlot[
   {red2d[[i]]} ~Join~ (RotateLine2[i, #, red2d, redx, blue2d, bluex] & /@
      Transpose[FindCrossPoint[redx[[i]]][[1]]),
   PlotRange → {{-2, 4}, {0, 2.5}}];
Figure1::usage = "Figure1[i,red2d,redx,blue2d,bluex]
    i番目の赤線に青線を直交させて表示する";
```

まず, 1本の赤線に対して複数ある交差する青線を同時に表示する. その関数がFigure1である. 曲線データ, 交差点データに対して, 赤線の番号を与えるだけで, その線に交差する青線たちを直交するように回転及び平行移動を行い表示する. 以下の例では, 1番, 5番, 15番, 20番の赤線に対しての実行結果を表示している.

In[]:=
```
Table[Figure1[i, red2d, redx, blue2d, bluex], {i, {1, 5, 15, 20}}]
```

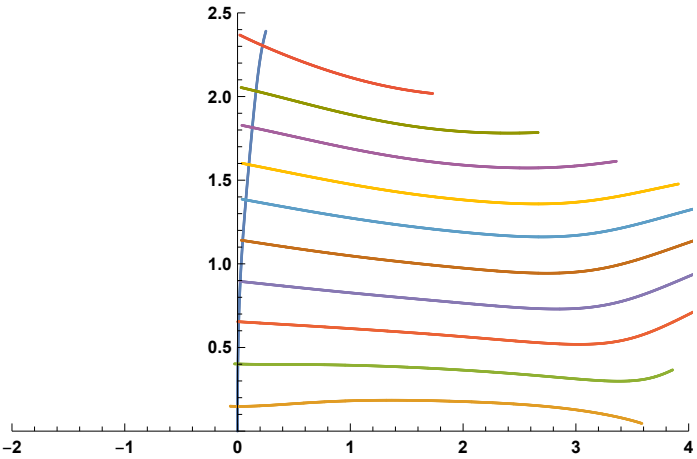
Out[]=



以降では, 最初のステップ, 1番の赤線と交差する青線の選択について, 詳細に考察する. 1番の赤線と交わる青線は10本存在する. それらの番号列とを blue2i に格納しておく. そして, それらを赤線と交差点で直交するように移動して表示する. いずれの青線を選択するにせよ, 次の赤線(2番)との交差点を先に求めておく. その頂点列を red2points に保存し, これらの頂点をもとのグラフに重ねて表示する.

In[]:=
```
Figure1[1, red2d, redx, blue2d, bluex]
```

Out[]=



In[]:=
```
blue2 = (RotateLine2[1, #, red2d, redx, blue2d, bluex] & /@
    Transpose[FindCrossPoint[redx[[1]]]][[1]]);
blue2i = Transpose[FindCrossPoint[redx[[1]]]][[1]]
```

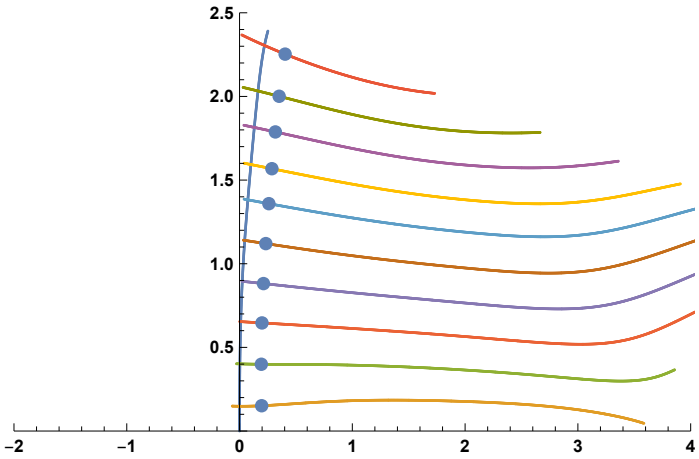
Out[]=
```
{4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
```

In[]:=
```
red2points = #[[1]][[#[[2]]]] & /@
  Transpose[{blue2, FindCrossPoint[bluex[[#]], 2] & /@ blue2i}]
```

Out[]=
```
{{0.19628, 0.151236}, {0.193292, 0.399208},
 {0.199004, 0.646218}, {0.212198, 0.881398},
 {0.233061, 1.12036}, {0.260017, 1.35905}, {0.286859, 1.56805},
 {0.317627, 1.78783}, {0.351194, 2.00096}, {0.404098, 2.25294}}
```

In[]:=
```
Show[{Figure1[1, red2d, redx, blue2d, bluex],
  ListPlot[red2points, PlotStyle → PointSize[Large]]}]
```

Out[]=

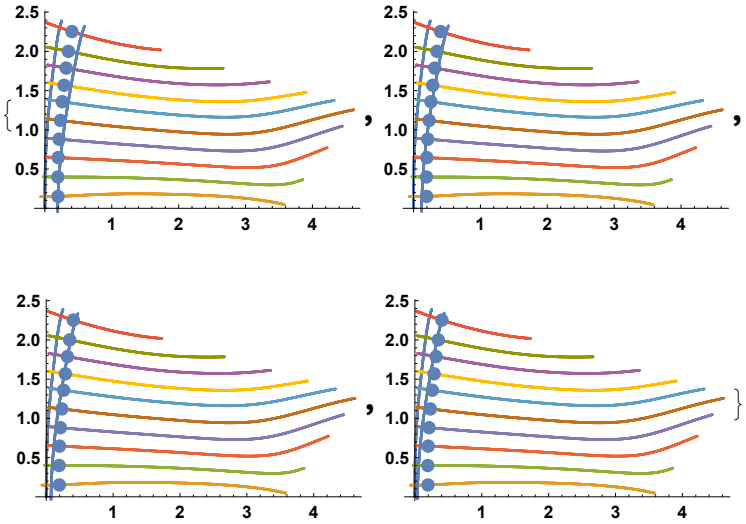


次の課題は, この10個の交差点のどれに2番の赤線を接続するのが効率が良いか, ということである. それを考察するために指定した青線を選択した場合の2番の赤線も一緒に表示する関数がFigure2である. 以下の例では, 4番, 8番, 11番, 13番の青線に対しての実行結果を表示している.

In[]:=
```
Figure2[i_, red2d_, redx_, blue2d_, bluex_] :=
  Show[{ListPlot[{red2d[[1]]}~Join~blue2],
    ListPlot[red2points, PlotStyle → PointSize[Large]], ListPlot[
     RotateLine[red2d[[2]], FindCrossPoint[redx[[2]], i], 0, RotateLine2[1,
        i, red2d, redx, blue2d, bluex][[FindCrossPoint[bluex[[i]], 2]]]]}];
Figure2::usage = "Figure2[i,red2d,redx,blue2d,bluex]
    i番の青線に赤線を直交させて表示する";
Red2Point[i_] :=
  RotateLine[red2d[[2]], FindCrossPoint[redx[[2]], i], 0, RotateLine2[1, i,
       red2d, redx, blue2d, bluex][[FindCrossPoint[bluex[[i]], 2]]][[#]] & /@
   (FindCrossPoint[redx[[2]], #] & /@ blue2i);
Red2Point::usage =
  "Red2Point[i] は, i番の青線を使って2番の赤線を引いたときの交点の座標たち.";
```

In[]:=
```
Table[Figure2[i, red2d, redx, blue2d, bluex], {i, {4, 8, 11, 13}}]
```

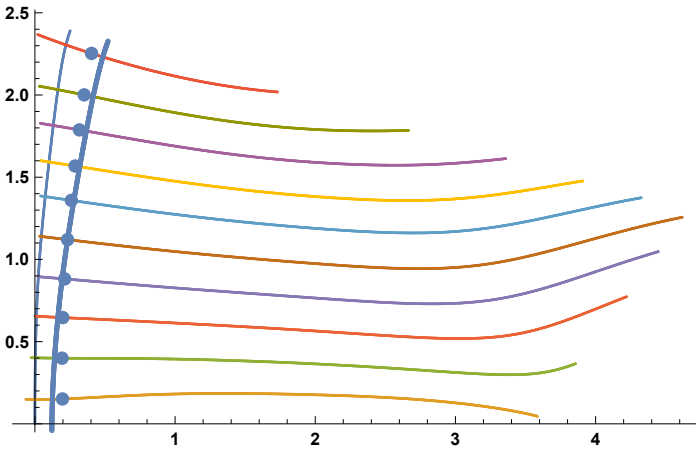
Out[]=



最後に, 8番の青線を選択した例について, 詳しく調べてみる. 関数Red2Point[i]はi番の青線を用いて接続したときの2番の赤線の交差点座標の列を返す関数である. これと正しい距離にある交差点座標red2pointsとのずれ, ベクトルとしての距離を指標として選択する青線を評価することを考える.

In[]:=
```
Figure2[8, red2d, redx, blue2d, bluex]
```

Out[]=



In[]:=
```
? Red2Point
```

Red2Point[i] は, i番の青線を使って2番の赤線を引いたときの交点の座標たち.

In[]:=
```
Red2Point[8]
```

Out[]=
```
{{0.125255, 0.156499}, {0.140487, 0.404034},
 {0.164321, 0.649973}, {0.19473, 0.883534},
 {0.233061, 1.12036}, {0.277427, 1.35643}, {0.319498, 1.56291},
 {0.366265, 1.77985}, {0.415328, 1.98996}, {0.486575, 2.23757}}
```

```
In[•]:= red2points
Out[•]= {{0.19628, 0.151236}, {0.193292, 0.399208},
  {0.199004, 0.646218}, {0.212198, 0.881398},
  {0.233061, 1.12036}, {0.260017, 1.35905}, {0.286859, 1.56805},
  {0.317627, 1.78783}, {0.351194, 2.00096}, {0.404098, 2.25294}}
```

最後に, 8番の青線を選択した例について, 詳しく調べてみる. 関数Red2Point[i]はi番の青線を用いて接続したときの2番の赤線の交差点座標の列を返す関数である. これと正しい距離にある交差点座標red2pointsとのずれ, ベクトルとしての距離を指標として選択する青線を評価することを考える. 以下は, 選択した青線番号と評価値のリストを評価値の小さい順に並べ替えて表示したものである. ここでは, 8番の青線を選択することが最も評価値の良い選択であることが確認できる.

```
In[•]:= Sort[Transpose[{blue2i, Norm[Red2Point[#] - red2points] & /@ blue2i}],
  #1[[2]] < #2[[2]] &]
Out[•]= {{8, 0.156414}, {9, 0.158185}, {7, 0.173513}, {10, 0.174955}, {11, 0.204023},
  {6, 0.204118}, {12, 0.239514}, {5, 0.245318}, {13, 0.287446}, {4, 0.292089}}
```

4. 曲率線の平面射影について

本節では曲率線展開法の要素である3次元空間での曲率線を長さを保存した平面曲率線へ変換する方法について述べる. まず, 事前に曲率線の測地線曲率は計算されているとする. これらの曲率を用いて2次元平面において, 原点(0,0)を始点として, 同じ曲率を持つ曲線を微分方程式を用いて求める.

4.1 赤1番の空間内曲率線データから平面射影曲率線データを作成する (1)

```
In[•]:= redt1 = FoldList[Plus, 0, Map[Norm, Drop[red3d[[1]], 1] - Drop[red3d[[1]], -1]]];
(* ↑ 各頂点に対して始点からの距離を計算する *)
rk1 = Interpolation[Transpose[{redt1, redk[[1]]}]];
(* ↑ 始点からの距離をパラメータとして 測地線曲率を関数として定義する. *)
```

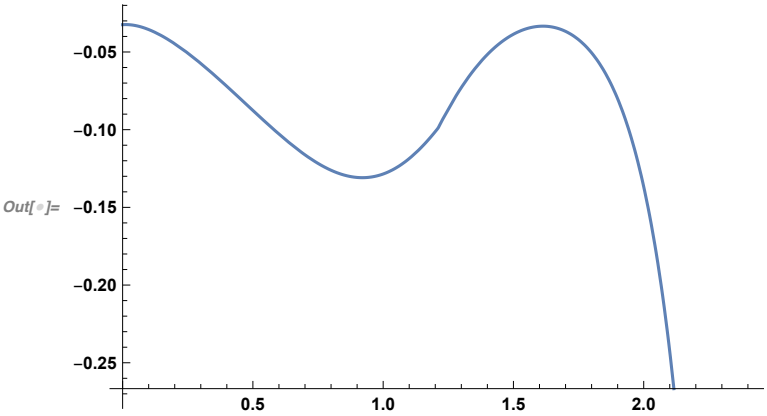
赤1番は, 曲率関数 k (u) = rk1[u] で, [0, 2.41144]上の関数である.

```
In[•]:= {redt1[[1]], redt1[[Length[redt1]]]}
(* ↑ 関数 rt の定義区間を確認する *)
Out[•]= {0, 2.41144}
```

曲率変化の確認

In[]:=
```
Plot[rk1[t], {t, redt1[[1]], redt1[[Length[redt1]]]}]
```

Out[]=



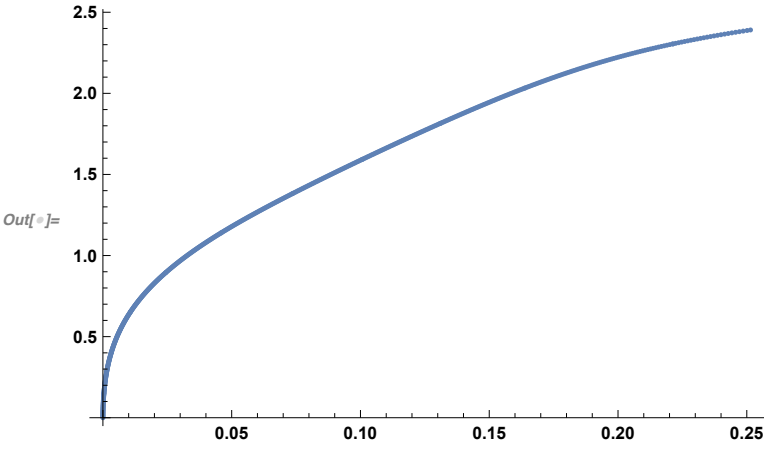
曲線 (x(t),y(t)) に対して, 曲率k(t)の定義式は, x' (t)^2 + y' (t)^2 == 1 のとき, k (t) = x' (t) y'' (t) - x'' (t) y' (t) である.

微分方程式を解いて平面曲線を求めて表示してみる
(注) NDSolveValueでの x, y は, 厳密には, x', y' のことである. したがって, 曲線の座標値を保存するリスト red2d21の定義では, これら(x,y)を原点から積分することによって座標値を求めている.

In[]:=
```
{xsol1, ysol1} = NDSolveValue[
   {rk1[t] == x[t] y'[t] - x'[t] y[t], x[t]^2 + y[t]^2 == 1, x[0] == 0, y[0] == 1},
   {x, y}, {t, redt1[[1]], redt1[[Length[redt1]]]}];
red2d21 = Table[{NIntegrate[xsol1[x], {x, 0, redt1[[i]]}],
    NIntegrate[ysol1[x], {x, 0, redt1[[i]]}]}, {i, 1, Length[redt1]}];
```

In[]:=
```
ListPlot[red2d21]
```

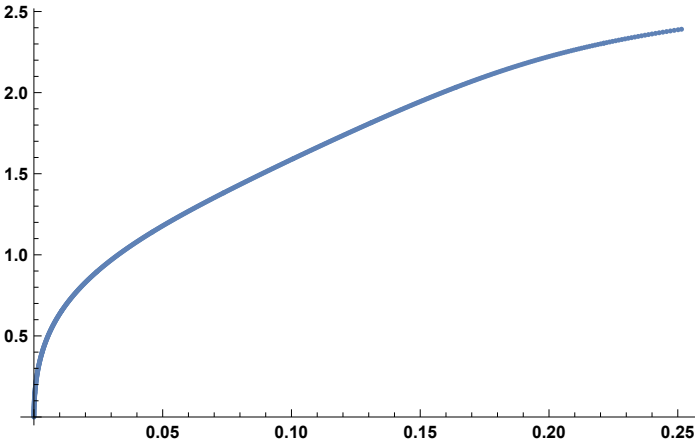
Out[]=



既存平面曲線データを表示して比較してみる

In[]:=
```
ListPlot[red2d[[1]]]
```

Out[]=



数値的にも, ほぼ同じ. 10^{-5} 以上の誤差はない.

In[]:=
```
Select[Norm /@ (red2d21 - red2d[[1]]), # > 10^-5 &]
```

Out[]=
```
{}
```

4.2 赤1番の空間内曲率線データから平面射影曲率線データを作成する (2)

微分方程式を解いて平面曲線を求めて表示してみる
(注) NDSolveValueでの x, y は, 厳密には, x', y' のことである. したがって, 曲線の座標値を保存するリスト red2d21の定義では, これら(x,y)を原点から積分することによって座標値を求めている.

ここでは微分方程式の解の積分計算を近似数値計算 (台形公式) により計算する
曲率関数 k (s) が与えられたとき, 求める平面曲線 γ (s) は

$$\gamma(s) = \int_0^s \left(\mathrm{Cos}\left[\int_0^t k(u)\,du\right], \mathrm{Sin}\left[\int_0^t k(u)\,du\right]\right) dt$$

In[]:=
```
{redt1[[1]], redt1[[Length[redt1]]], Length[redt1]}
```

Out[]=
```
{0, 2.41144, 614}
```

$$a0 = \int_0^t k(u)\,du$$

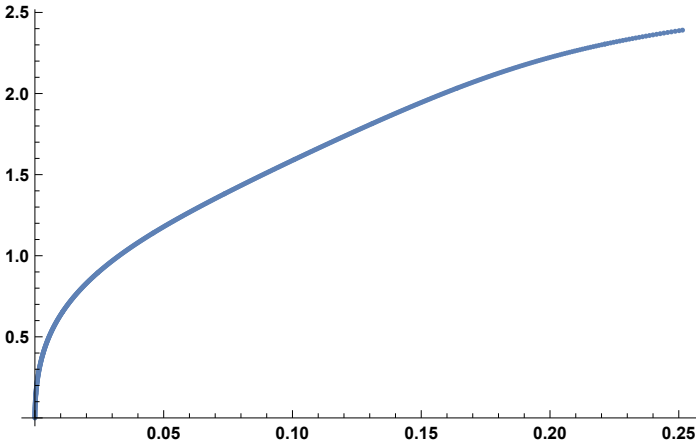
$$(cos0, sin0) = \int_0^s \left(\mathrm{Cos}\left[\int_0^t k(u)\,du\right], \mathrm{Sin}\left[\int_0^t k(u)\,du\right]\right) dt$$ **で計算し,**

解を赤のときは, $f = (-sin0, cos0)$ **, 青のときは,** $f = (cos0, sin0)$ **とする.**

```
In[•]:= a0 = FoldList[Plus, 0, Table[((redk[[1, i]] + redk[[1, i - 1]])/2) *
            (redt1[[i]] - redt1[[i - 1]]), {i, 2, Length[redt1]}]];
        cos0 = FoldList[Plus, 0, Table[((Cos[a0[[i]]] + Cos[a0[[i - 1]]])/2) *
            (redt1[[i]] - redt1[[i - 1]]), {i, 2, Length[redt1]}]];
        sin0 = FoldList[Plus, 0, Table[((Sin[a0[[i]]] + Sin[a0[[i - 1]]])/2) *
            (redt1[[i]] - redt1[[i - 1]]), {i, 2, Length[redt1]}]];
        f0 = Transpose[{-sin0, cos0}];
```

```
In[•]:= ListPlot[f0]
```

Out[•]=




```
In[•]:= Max[Norm /@ (red2d21 - f0)]
```

Out[•]= 3.82613×10^{-6}

微分方程式の解との差 10^{-5}未満

```
In[•]:= Max[Norm /@ (red2d[[1]] - f0)]
```

Out[•]= 1.08793×10^{-6}

既存ファイルデータとの差 10^{-5}未満

4.3 全ての最大曲率線(赤線)の平面射影曲率線データを作成する

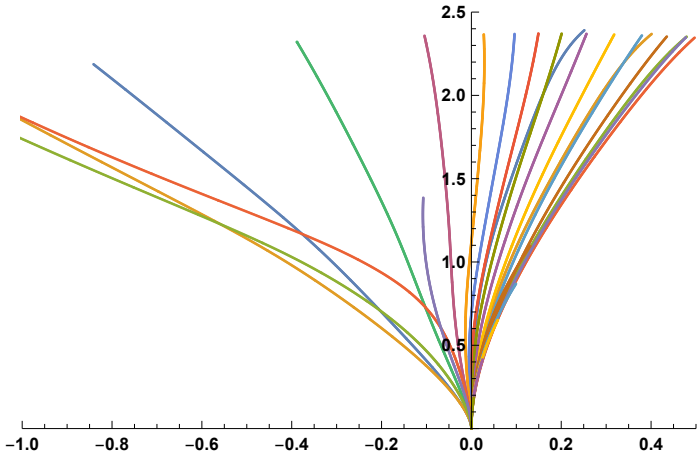
```
In[•]:= redt = FoldList[Plus, 0, Map[Norm, Drop[#, 1] - Drop[#, -1]] & /@ red3d;
        (* ↑ 各頂点に対して始点からの距離を計算する *)
```

```
In[ ]:= ra0 = FoldList[Plus, 0,
            Table[(#[[1]][[i]] + #[[1]][[i - 1]])/2 * (#[[2]][[i]] - #[[2]][[i - 1]]),
             {i, 2, Length[#[[2]]]}]] & /@ Transpose[{redk, redt}];
        rcos0 = FoldList[Plus, 0, Table[(Cos[#[[1]][[i]]] + Cos[#[[1]][[i - 1]]])/2 *
              (#[[2]][[i]] - #[[2]][[i - 1]]),
             {i, 2, Length[#[[2]]]}]] & /@ Transpose[{ra0, redt}];
        rsin0 = FoldList[Plus, 0, Table[(Sin[#[[1]][[i]]] + Sin[#[[1]][[i - 1]]])/2 *
              (#[[2]][[i]] - #[[2]][[i - 1]]),
             {i, 2, Length[#[[2]]]}]] & /@ Transpose[{ra0, redt}];
        rf0 = Transpose[{-#[[2]], #[[1]]}] & /@ Transpose[{rcos0, rsin0}];
```

全ての平面曲線(赤線)の一括表示

```
In[ ]:= ListPlot[rf0, PlotRange → {{-1, 0.5}, {0, 2.5}}]
```

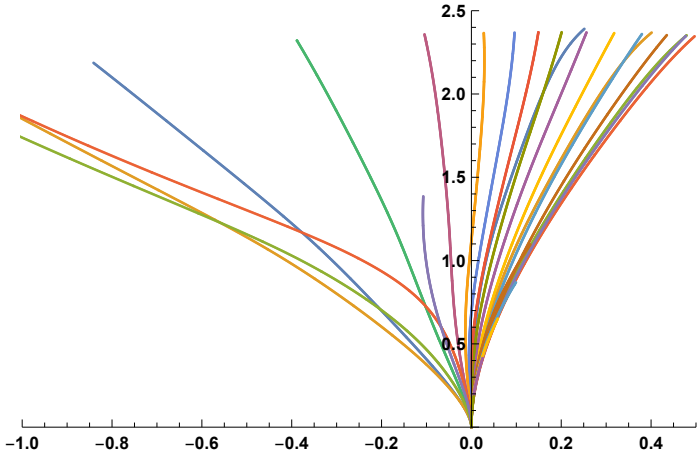
Out[ ]=



既存平面曲線データを表示して比較してみる (ほぼ同じ形!)

```
In[ ]:= ListPlot[red2d, PlotRange → {{-1, 0.5}, {0, 2.5}}]
```

Out[ ]=



4.4 全ての最小曲率線(青線)の平面射影曲率線データを作成する

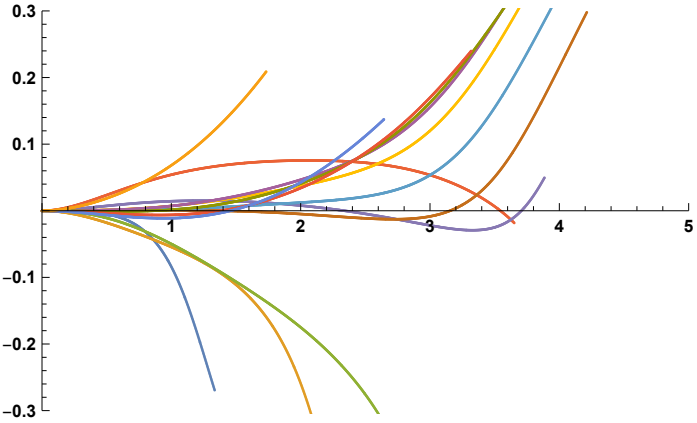
```
In[ ]:= bluet = FoldList[Plus, 0, Map[Norm, Drop[#, 1] - Drop[#, -1]] &] /@ blue3d;
        (* ↑ 各頂点に対して始点からの距離を計算する *)

In[ ]:= ba0 = FoldList[Plus, 0,
            Table[(#[[1]][[i]] + #[[1]][[i - 1]])/2 * (#[[2]][[i]] - #[[2]][[i - 1]]),
              {i, 2, Length[#[[2]]]}]] & /@ Transpose[{bluek, bluet}];
        bcos0 = FoldList[Plus, 0, Table[(Cos[#[[1]][[i]]] + Cos[#[[1]][[i - 1]]])/2 *
              (#[[2]][[i]] - #[[2]][[i - 1]]),
            {i, 2, Length[#[[2]]]}]] & /@ Transpose[{ba0, bluet}];
        bsin0 = FoldList[Plus, 0, Table[(Sin[#[[1]][[i]]] + Sin[#[[1]][[i - 1]]])/2 *
              (#[[2]][[i]] - #[[2]][[i - 1]]),
            {i, 2, Length[#[[2]]]}]] & /@ Transpose[{ba0, bluet}];
        bf0 = Transpose[{#[[1]], #[[2]]}] & /@ Transpose[{bcos0, bsin0}];
```

全ての平面曲線(青線)の表示

```
In[ ]:= ListPlot[bf0, PlotRange → {{0, 5}, {-0.3, 0.3}}]
```

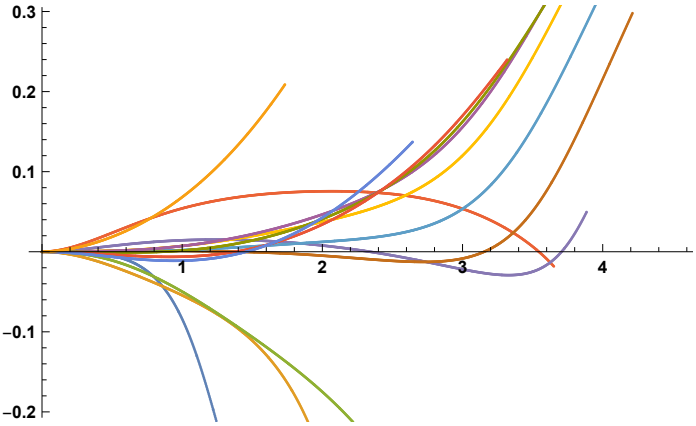
Out[ ]=



既存平面曲線データを表示して比較してみる (ほぼ同じ形!)

In[ ]:= ListPlot[blue2d]

Out[ ]=



5. 参考文献

[1] 松尾宏平他, 曲率展開による新外板展開法とそのソフトウェア・パッケージ化について, 日本船舶海洋工学会講演会論文集, 第1号,pp.285-286, 2005.
[2] 松尾宏平他, 曲率線を用いた新しい外板ランディング手法について, 日本船舶海洋工学会講演会論文集, 第3号,pp.525-526, 2006.
[3] 海上技術安全研究所, IT活用で測地線から「曲率線」展開へ (工数3割以上削減, 造船所の効率化支援), 海技研ニュース「船と海のサイエンス」, 2010年冬号, pp.4-8, 2010.
[4] 住友重機械マリンエンジニアリング, 曲率線展開プログラムの現場適用, 海技研ニュース「船と海のサイエンス」, 2010年冬号, pp.9-11, 2010.
[5] 松尾宏平, 藤本修平, 曲率線展開システム及びプレス施工支援システム, 海上技術安全研究所報告, Vol.15, No.4, pp.77-91, 2015.