Development and application of high speed and high accuracy cutting simulator using Vatti clipping

博　士　論　文

Vatti クリッピングを用いた高速高精度切削シミュレータ開発とその応用

金沢大学大学院　自然科学研究科
機械科学専攻

学　籍　番　号　1824032004
氏　　名　鬼頭　亮太
主任指導教員名　浅川　直紀
提　出　年　月　令和　3 年　1 月

―目次―

第 5 章　切削シミュレータの応用例

第 6 章　結論

謝辞

第 1 章　緒論

1.1　研究背景

コンピュータの発展に伴い，さまざまな分野でコンピュータを用いた自動化が行われている．これはものづくりも同様であり，1952 年に MIT において，世界初の NC 工作機械が開発されたのが，ものづくりの自動化の始まりである [1]．それまで，人間が工作機械を扱い, 製品を手動で作成していたのが, NC 工作機械を用いて自動化できるようになった．一方，この NC 工作機械を動作させる NC プログラムの作成は人間が行っており，非常に時間が掛かったため，1956 年に工具経路の計算を自動化するためのプログラミング言語として CAM（Computer Aided Manufacturing）の前身である APT（Automatically Programmed Tool）が開発された [2]．そして, その後 CAD（Computer Aided Design）や CAE（Computer Aided Engineering）など自動化のための多くのソフトウェアが開発されている．そして，2000 年ごろに多くの多軸加工機や複合加工機が開発され，自由曲面の数値制御加工が実用化，多軸加工などに対応した CAM ソフトウェアが開発されてきた [3]．このようにものづくりにおいて，ソフトウェアなどのデジタル技術は非常に重要かつ切り離すことのできない関係となっている．

そして, 近年では IoT (Internet of Things) によって多くの物がインターネットにつながり, センシング技術の発達によりさまざまなデータを測定することができ，コンピュータの高性能化やビッグデータ解析，AI（Artificial Intelligence）技術などの発達により，大量に測定されたデータが活用されている．そして，その次のステップとして，デジタルツインが近年注目されている．これは現実世界のモノをデジタル世界で再現し，センシングにより測定したデータなどを用いて，そのモノをシミュレートする技術である．あらゆる問題をデジタル世界で先に知ることによって, 問題が発生する前に対応を可能とする [4]．例えば, 高炉の外部に取り付けた 1000 個以上のセンサから温度や圧力を測定，高炉内部の状況をシミュレートし，異常が起きる予兆を通知できるシステムの構築が行われている [5]．このデジタルツインは生産加工分野でも議論されており，例えば，切削工具のデジタルツインのデータ構造などの提案 [6] や，多軸工作機械の誤差を含む加工運動機能を表現できるデジタルツインを開発できれば，この多軸工作機械の特性を生かした工程設計を行うことができると提案している [7]．また，KOM-MICS では，社内及びその協力企業にある産業用ロボットや工作機械のデータを測定，収集し，稼働状況や切削力などの見える化を行っている．また，その測定したデータを用いて，切削力の一定化やエアカットの削減を行った加工プログラムの自動修正ができ, 改善後の NC プログラムのシミュレーションを行い, 切削力や切削時間の確認もシステム上で行うことができる [8]．このように多くの研究者や企業がデジタルツインに関する研究開発を行っており，これらを応用することによって工作機械が故障する前に故障を予知したり，工具寿命の把握，治具と工具などの干渉回避や，びびり振動の回避，寸法誤差や表面粗さの把握などを実時間より短い時間で行うことができ，生産加工分野への影響は多大なものになると考えられる．

このデジタルツインで特に重要になってくるのが，シミュレーションに関する部分だと考えられる．シミュレーションの精度が現実世界と一致しなければ，現実世界と同じものをデジタル世界で表現することができない．また，現実世界で掛かっている時間より短い時間でシミュレーションを行うことができなければ，デジタル世界で現実世界より先に問題を把握することができない．よって，デジタルツインにとって，シミュレーションは高精度かつ高速であることが重要となる．

一方，エッジコンピューティングという新しい情報処理の枠組みが注目されている [9]．これは，サーバーに任せていたデータ処理をユーザに近い端末，例えば小型 PC やネットワーク対応型シーケンサ，工作機械で言えばパネルコンピュータ上でデータ処理を行い，上位システムへの負荷や通信遅延を解消するものである [10]．上述したデジタルツインを工作機械のパネルコンピュータ内で構築可能であれば，加工と同時進行でシミュレーションを行い，工具経路などに問題がある場合，現実空間より先に動作を止めることができる．また，動作を止めるだけでなく，最適な工具経路の自動生成や修正を加工中に動的に行うこともできると考えられる．デジタルツインのシミュレーションは計算コストが非常に大きいと予想されるが，工作機械のパネルコンピュータは高性能な PC を使用していないため，大量のマルチコアや GPU などの演算ユニットに頼らない計算方法でシミュレーションの高速化を行う必要がある．

そこで, 本研究はデジタルツインやエッジコンピューティングへの発展を視野に入れた，切削シミュレータ開発を行おうと考えた．そのためには，上述したように，PC のスペックに頼らず，アルゴリズムやデータ構造を工夫することによって，高精度かつ高速な切削シミュレータを開発する必要がある．

1.2　切削シミュレータ

切削シミュレータの研究例として，Inui らは工作物形状をデクセルモデルで表現し，工具と工作物の接触領域を GPU を用いて計算している [11]．本田らはいくつかの工具形状について FEM 解析を行い，切り屑形状や残留応力などの解析を行っている [12]．金子らは工作機械の主軸モータの切削トルクと切削力モデルを用いてシミュレーションした切削トルクを比較し，工具の欠損を検知するシステムを構築している [13]．このように，さまざまな切削シミュレーションに関する研究が行われている．この切削シミュレータの機能を大きく分類すると

(1) 工具，治具，工作物の干渉検出
(2) 切削過程の確認
(3) 切削体積，切削力解析
(4) 切りくず流出解析
(5) 切削温度解析

などが挙げられ，解析内容によって計算方法が変わる．その計算方法を示すと，

（ⅰ）切削係数法
（ⅱ）エネルギ解析法
（ⅲ）有限要素法

が挙げられる．これらの特徴を表 1.1 に示す．図 1.1 に示すように，有限要素法やエネルギ解析法では切り屑や切削温度など，切削係数法よりさまざまなことをシミュレーション

表 1.1　切削シミュレータの計算方法 [14]

	解析手法	出力	特徴（長所・短所）
切削係数法	切削力＝比切削抵抗 × 切削面積＋刃先の押込み力	切削力	【長所】計算時間が早いため，びびり振動の解析まで容易に可能 【短所】工具形状に依存する比切削抵抗のデータベースの管理が問題
エネルギ解析法	3 次元の切りくず形状を 2 次元切削の重ね合わせとしてモデル化．切りくず流出方向はエネルギ最小の方向	切削力 切りくず流出方向 切削エネルギ	【長所】①切りくず流出方向が解析的に得られる．②二次元切削データのみで任意の切れ刃形状に対応するため，工具形状設計に利用できる．③データベースの管理が容易． 【短所】比切削抵抗法に比べて解析時間がかかる．
有限要素法	塑性力学に基づく数値解析．流動応力特性により材料の変形を逐次解析	切削力 材料と工具の応力・ひずみ・温度	【長所】①切りくず形状が得られるため，切りくず処理の具体化が可能．②材料内部の応力，ひずみ，温度から加工変質層，残留応力等が評価できる． 【短所】①解析時間がかかるため，三次元の複雑な工具形状に対する解析が困難．②流動応力特性の完備，管理が困難．

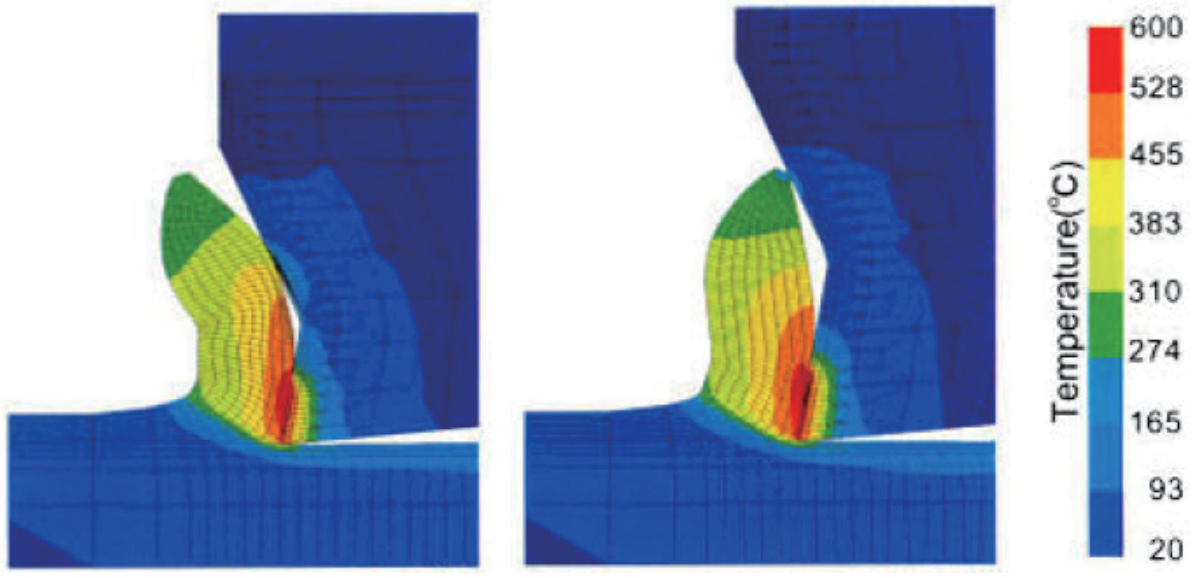


図 1.1　有限要素法でのシミュレーション例 [12]

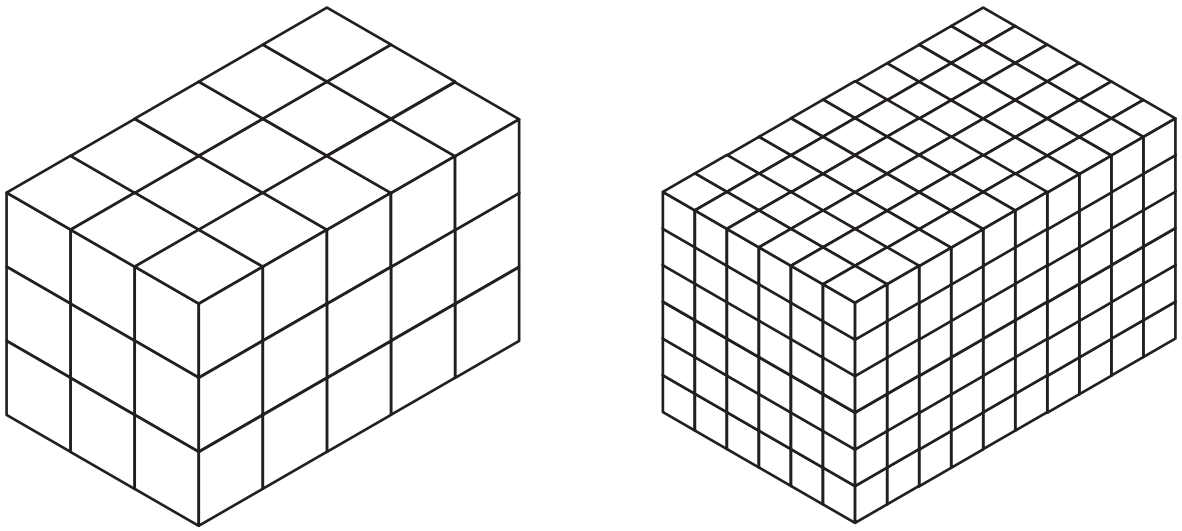


図 1.2　ボクセル表現

することができるが，計算時間が多く掛かってしまう．切削係数法は切削領域の幾何学的な解析に基づくため，他の手法より計算時間が短く，切削厚さの計算が複雑なエンドミルの切削力解析に適用されている [15]．しかし，最も計算時間が短い切削係数法であっても，空間分解能と時間分解能を上げると計算時間が増加してしまうため，高速性は低いのが現状であり，本研究では，切削過程の確認と切削係数法に必要な切削体積を解析することができる切削シミュレータ開発を行う．

切削係数法を用いている切削シミュレータに関する研究の多くは図 1.2 に示すような，ボクセル表現と呼ばれる微小な立方体形状を基本概念とする表現方法を用いている．ボクセル表現はこの立方体の集合で工作物を表現し，ブール演算を実施することで，切削過程を表現する．このボクセル表現には，単純なデータ構造であるため，データ構造への対策が容易であったり，微小立方体の内外判定でブール演算や切削体積計算を行うため，非常に単純であり，GPU を用いた演算に最適である特徴がある．ボクセル表現における空間分解能はボクセルサイズそのものとなるため，例えば，空間分解能 0.1 mm で 10 mm^3 の立方体を表現する場合，100 万個のボクセルが必要となる．また，空間分解能を 0.01 mm に変更すると，10 億個のボクセルが必要になり，空間分解能を増加させると，空間分解能の増分比の 3 乗倍でボクセル数が増加していく．ボクセル表現を使用している研究はさまざまあり [16] 〜 [26]，例えば，Wou らはボクセル表現と CG 分野のレイキャスティングを使用して切削力シミュレーションを開発している [22]．Noguchi らは機械構造の数学モデルとスピンドル駆動の数学モデルを組み合わせたボクセル切削力シミュレータを開発している [23]．Nishida らはボクセルを用いて静的な工具のたわみを考慮したエンドミル加工の切削シミュレータを開発している [24]．Joy らはボクセル表現を基本としたフレームスライスボクセル表現を開発し，ボクセル表現より高い精度を実現した [25].

他にも，ボクセルを使用した切削シミュレータの高速化やメモリ消費量を抑える研究として Octree 構造が提案されている [26]．この Octree 構造は図 1.3 に示すように，1 個のボクセルを 8 等分に分割して小さなボクセルとし，これを階層構造にすることで，微細な形状は小さなボクセルを用いて，単純な形状では大きなボクセルを使用して，計算コストや

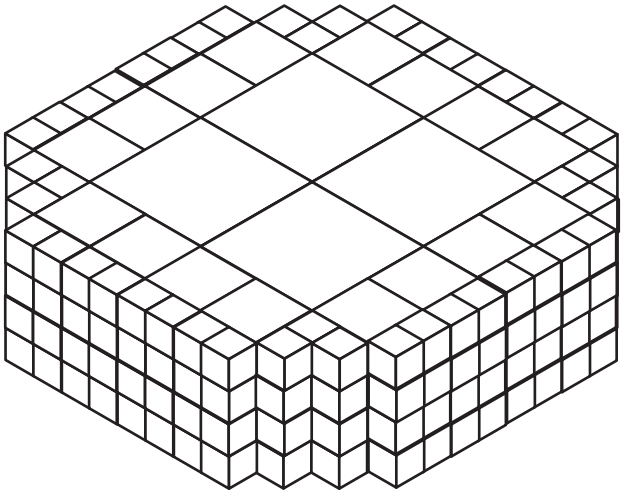


図 1.3　Octree 構造を用いたボクセル表現

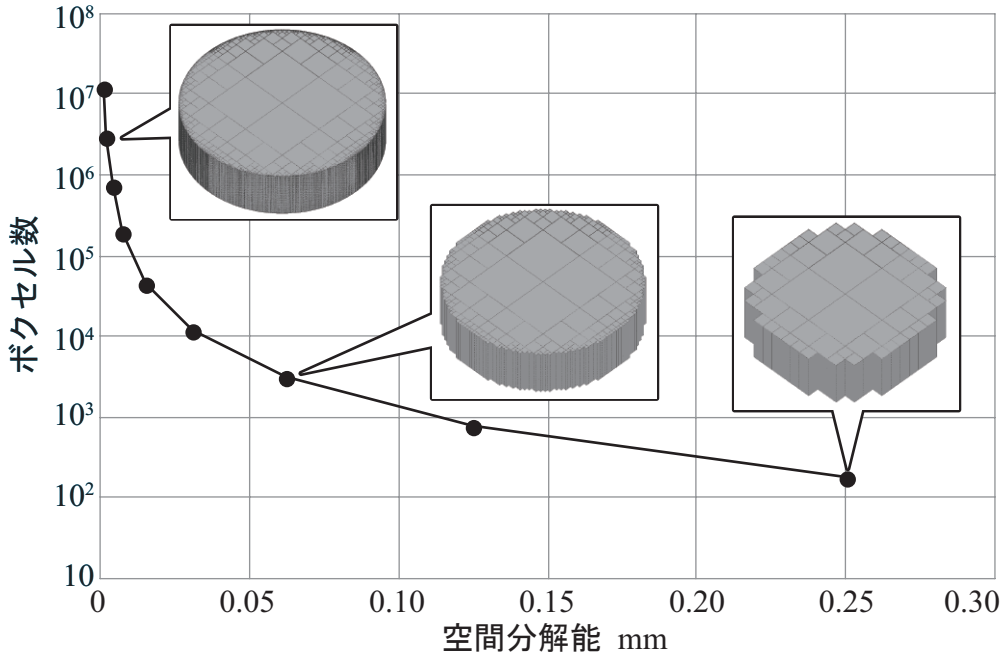


図 1.4　Octree 構造を使用したときの空間分解能とボクセル数の関係

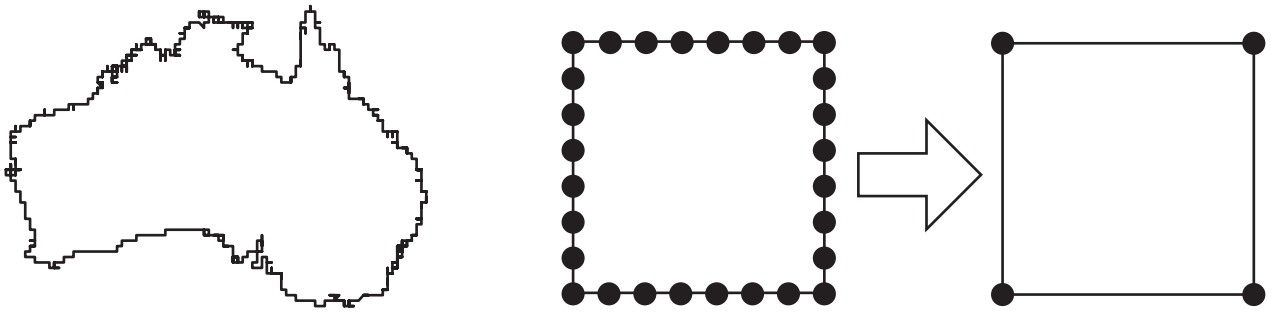
メモリ消費量を減らす技術である．しかし，ボクセル表現は，Octree 構造を使用しても，空間分解能を上げる際に最小ボクセルのサイズを小さくしなければいけないため，メモリ消費量や計算時間がどうしても増加してしまうという問題点が存在する．例えば，最大ボクセルサイズを 1 mm として，直径 4 mm，高さ 1 mm の円筒を表現した場合，図 1.4 のように空間分解能を上げるとボクセル数が増加してしまう．また，空間分解能が 0.25 mm の場合，総ボクセル数は 180 個だが，空間分解能を約 1 μm に変更すると，総ボクセル数は 1167 万個になり，空間分解能によって指数関数的にボクセル数が増加することがわかる．よって，ボクセル表現の空間分解能には限界がある．

1.3 研究目的

従来のボクセル表現では空間分解能を増加させると，工作物を表現するのに使用するボクセル数が増加してしまうため，本研究では，従来のボクセル表現を使用せず，新しい方法で高速高精度の新しい切削シミュレータ開発を目的とする．

そこで着目したのが，多角形の面で表現されるポリゴン表現である．上述したように，ボクセル表現では，ボクセルのサイズが空間分解能を決めるため，単純な形状であっても空間精度の高いモデルを表現する場合大量のボクセルが必要になる．いわば，ボクセルの個数が空間分解能を決め，3 次元空間を微小立方体で満たす必要がある．これに対して，ポリゴン表現を用いたとき，図 1.5(a) のように，複雑な形状を表現する場合，多くの頂点が必要になるが，これらの頂点自体はプログラム上では実数型で保持でき，隣り合う点間距離に制約はないため，図 1.5(b) に示すように，空間上を頂点で満たす必要がなく，頂点数が空間分解能を左右することはない．よって，最低限の頂点数で形状を表現できるため，無駄な演算が発生しない．また，頂点を実数型で保持できるため，空間分解能も実数型の精度となる．

また，本研究ではシミュレーション中の途中形状の計算方法として，ポリゴンクリッピングに着目した．ポリゴンクリッピングは CG 分野で発達しており，さまざまなアルゴリズムがある．例えば，Liang-Barsky アルゴリズム [27] は単純な長方形のみに対してのみクリッピングを行うことができ，Sutherland–Hodgman アルゴリズム [28] や Cyrus–Beck アルゴリズム [29] は凸ポリゴンに対してのみクリッピング可能である．これらのアルゴリズムのように，ポリゴンの形状に制限があると，さまざまな形状に変化する切削シミュレータに応用することができない．そこで，本研究では Vatti クリッピング [30] に着目した．Vatti クリッピングは高速性，汎用性を兼ね備えており [31]，アルゴリズムも単純であるため改良を行うことも可能であると考え，採用した．Vatti クリッピングは様々な分野で使用されている技術である [32] 〜 [34]．例えば，Surmann らは室内環境を 3D レーザ測定し，その測定データの処理に使用している [32]．Jones らは，CAD モデルの自動変換を行う際に使用している [33]．また，Tymstra らは，火災成長シミュレーションの開発に使用している [34]．本研究ではこの Vatti クリッピングを応用することによって，高速高精度の切



(a) 複雑形状と頂点数　　(b) 特徴のある箇所にのみ頂点生成

図 1.5 ポリゴン表現の特徴

削シミュレータの開発を行う．

しかし，2 章で詳しく説明するが，ポリゴン表現と Vatti クリッピングを切削シミュレータに応用する上で以下に示す 3 つの問題点が存在する．

（ｉ）頂点数が増加すると計算時間が増加する
（ii）円の表現は多角形になる
（iii）Vatti クリッピングは 2 次元形状にのみ対応したアルゴリズムである

ポリゴン表現で複雑な形状を表現する際には大量の頂点が必要となり，頂点数が増加すると，高速にブール演算を行える Vatti クリッピングであっても，計算時間が増加してしまう．例えば，単純な立方体に対して加工を行うシミュレーションを実施した際，加工序盤では頂点数が少ないため高速に計算できるが，加工終盤では，加工によって工作物形状が複雑になり頂点数が増加しているため，計算に時間が掛かると考えられる．また，円の表現に関しては，ポリゴン表現は頂点を直線でつなげるため，大量の頂点を使用した多角形で表現する．よって，精度を増加させるために頂点数を増やすと計算時間が増加し，計算時間を短くするために頂点数を減らすと精度が低下してしまい，精度と計算時間にトレードオフの関係がある．最後に，Vatti クリッピングは 2 次元形状にのみ対応したアルゴリズムであり，3D モデルに対してブール演算を行えないため，Vatti クリッピングをそのまま使用して，3 次元切削シミュレータを開発できない．

そこで，本研究はこれら 3 つの問題を解決し，高速高精度の切削シミュレータ開発を行う．具体的な空間分解能と時間分解能の目標値は，x，y 方向の空間分解能は 1 μm，ボクセル表現を用いて，GPU と Octree 構造を応用した先行研究 [35] では空間分解能が最大でも 0.3 mm となっていることから，z 方向はこの空間分解能を超える 0.1 mm 以下とし，工具が 1 回転するのに掛かる時間を時間分解能の目標値とする．高速性の目標としては，加工途中形状の算出と切削体積計算を実際の加工時間と同等または，より短い時間でシミュレーション可能な切削シミュレータ開発を行う．

また，開発した切削シミュレータの高速性を利用したアプリケーション開発を行う．具体的な説明は 5 章で行うが，工具微小移動ごとに，開発した切削シミュレータを用いて切削体積を算出する．そして，その切削体積を用いてびびり安定限界解析を行い，びびり振動の有無を判断，最適な加工点生成し，びびり振動が発生しない工具経路生成を行う．そして，実機による加工実験を行い，提案手法の有効性を示す．

1.4 開発環境

1.4.1 Personal Computer（PC）

本システムを開発し，計算時間を測定した PC のスペックは，CPU：Intel(R) Xeon(R) Silver 4110 2.10GHz，メモリ：32GB，OS：Windows10 Pro である．

1.4.2 開発言語

開発言語には汎用プログラミング言語である C/C++ を使用した．また，統合開発環境は Microsoft Visual Studio Community 2017 Version 15.9.16 を使用した．

1.4.3 使用ライブラリ

本システムを開発するにあたって，使用したライブラリを以下に示す．

・GLEW/GLFW

GLEW（OpenGL Extension Wrangler Library）[36] はクロスプラットフォームでオープンソースの C/C++ 拡張ライブラリである．OpenGL 4.6 をサポートしており，グラフィックスハードウェアの拡張機能のためのライブラリである．

GLFW（Graphics Library Framework）[37] はオープンソースのマルチプラットフォームライブラリであり，ウィンドウなどを作成し，キーボードなどの入力を処理できる API である．

本システムでは，これらのライブラリをウィンドウの作成，工具，工作物の描画などで使用している．また, 使用した GLEW はバージョン 2.1.0, GLFW はバージョン 3.3.2 である．

・Eigen

Eigen[38] は線形代数（行列, ベクトルなどに関するアルゴリズム）用の C++ テンプレートライブラリである．

本システムでは，複素数を含む行列の固有値，固有ベクトルを求める際に使用している．また，使用したバージョンは 3.3.7 である．

・Dear ImGui

Dear ImGui[39] は C++ 用のグラフィックユーザーインターフェイスライブラリであり，本研究では，開発した切削シミュレータの GUI に使用した．図 2.1 に Dear ImGui のウィジェットのサンプルを示す．さまざまな機能を持ったウィジェットが用意されている．

図 1.6　Dear ImGui のウィジェットの例

1.5　本論文の構成

本論文は第 1 章の緒論から第 6 章の結論までの全 6 章で構成されている．以下に各章の概要を示す．

第 1 章「緒論」では，切削シミュレータの現状と課題，本研究の目的，本研究で用いた PC や開発言語，使用したライブラリについて述べる．

第 2 章「ポリゴン表現と Vatti クリッピング」では，ポリゴン表現と Vatti クリッピングに関する詳しい説明と問題点を示す．

第 3 章「切削シミュレータの開発」では，2 章で述べた Vatti クリッピングを応用する上での問題点の解決方法を説明する．

第 4 章「切削シミュレータの高速性の検証」では，3 章で開発した切削シミュレータを用いて 3 種類のケーススタディを行い，高速性の検証を行う．

第 5 章「切削シミュレータの応用」では，開発した切削シミュレータを用いたびびり振動が発生しない工具経路生成方法を提案し，実機による加工実験を行い，提案した工具経路生成の有効性を示す．

第 6 章「結論」では，本論文の結論を述べる．

参考文献

[1] 雨宮好文 , 安田仁彦：CAD/CAE/CAM 入門，オーム社，2001

[2] 白瀬敬一：変種変量生産に求められる CAM 技術，計測と制御，46，7(2007)，505

[3] 松村隆：平成の精密加工を振り返る ～切削加工を取り巻く環境の変化と動向～，精密工学会誌，86，1(2020)，15

[4] 日経 BP 社：製造業革新を加速するデジタルツイン，日経ものづくり，755，(2017)，34

[5] 日経 BP 社：JFE スチール 高炉の「デジタルツイン」開発 数億円の損害出るトラブルを回避，日経コンピュータ，1029，(2020)，66

[6] D.Botkina, M.Hedlind, B.Olsson, J.Henser, T.Lundholm: Digital twin of a cutting tool, Procedia CIRP, 2018;72;215–218.

[7] 田中文基：個別多軸工作機械における加工機能のデジタルツイン，日本機械学会第 13 回生産加工・工作機械部門講演会講演論文集，(2019)，287

[8] 齋藤尚登，坪井啓介：生産現場 " つながる化 "KOM-MICS，Komatsu technical report，62，169(2016)，9

[9] 日経 BP 社：クラウド偏重からの脱却、まずは特定用途から始動 (Breakthrough エッジコンピューティングの破壊力)，日経エレクトロニクス，1161，(2015)，46

[10] キーエンス :IoT 用語辞典，エッジコンピューティング , https://www.keyence.co.jp/ss/general/iot-glossary/edge-computing.jsp, (2020/12/22 アクセス)

[11] M.Inui, M.Kobayashi, N.Umezu: Cutter Engagement Feature Extraction Using Triple-Dexel Representation Workpiece Model and GPU Parallel Processing Function, Computer-Aided Design and Applications, 2018;16;1:89-102.

[12] 本田考耶，笹原弘之：有限要素法による切削シミュレータの開発―GUI の開発と工具形状の影響の検討―，精密工学会誌，71，1(2005)，115

[13] 金子和暉，西田勇，佐藤隆太，白瀬敬一:エンドミル加工における切削力シミュレーションを利用した加工状態のスマートモニタリング，2020 年度精密工学会春季大会学術講演会講演論文集，(2020)，736

[14] 松村隆：切削加工シミュレーション (特集 切削加工と特殊鋼)，特殊鋼，63，3(2014)，6

[15] 松村隆：切削シミュレーションの現状と課題，精密工学会誌，80，9(2014)，803

[16] K.Watanabe , J.Kaneko, K.Horio: Development of high-speed automatic planning method of tool posture considering shape change of workpiece during machining process, Proc. of the 9th Int. Conf. on Leading Edge Manufacturing in 21th Century 2017.

[17] T.Kobayashi, T.Hirooka, A.Hakotani, R.Sato, K.Shirase: Tool Motion Control Referring to Voxel Information of Removal Volume Voxel Model to Achieve Autonomous Milling Operation, International Journal of Automation Technology, 2014;8;6:792-800.

[18] D.Jang, K.Kim, J.Jung: Voxel-Based Virtual Multi-Axis Machining, The International Journal of Advanced Manufacturing Technology, 2000;16;10:709-713.

[19] G.J.Jense: Voxel-based methods for CAD, Computer-Aided Design, 1989; 21;8:528-533.

[20] W.H.Walstra, W.F.Bronsvoort, J.S.M.Vergeest: Interactive simulation of robot milling for rapid shape prototyping, Computers & Graphics, 1994;18;6:861-871.

[21] K.Shirase, K.Nakamoto: Simulation Technologies for the Development of an Autonomous and Intelligent Machine Tool, Int. J.Automation Technol, 2013;7;1:6-15.

[22] S.J.Wou, Y.C.Shin, H.El-Mounayri: Ball end milling mechanistic model based on a voxel-based geometric representation and a ray casting technique, Journal of Manufacturing Processes, 2013;15:338-347.

[23] S.Noguchi, R.Sato, I.Nishida, K.Shirase: Coupled Simulation between Machine Tool Behavior and Cutting Force using Voxel Simulator, Proc. of the 9th Int. Conf. on Leading Edge Manufacturing in 21th Century 2017.

[24] I.Nishida, R.Okumura, R.Sato, K.Shirase: Cutting Force Simulation in Minute Time Resolution for Ball End Milling Under Various Tool Posture, Journal of Manufacturing Science and Engineering, Transactions of the ASME, 2018;77:571-577.

[25] J.Joy, H.Y.Feng. Frame-sliced voxel representation: An accurate and memory-efficient modeling method for workpiece geometry in machining simulation, Computer-Aided Design, 2017;88:1-13.

[26] 中本圭一，河野智之，小山智，阪口龍彦，白瀬敬一:ボクセルモデルを用いたヴァーチャルマシニングシミュレータの開発，精密工学会誌，74，12(2008)，1308

[27] Y.Liang, B.A.Barsky: A new concept and method for line clipping, ACM Transactions on Graphics, 1984;3;1:1-22.

[28] Sutherland.I.E, Hodgman.G.W: Reentrant Polygon Clipping, Communications of the ACM,1974; 17; 1; 32-42.

[29] M.Cyrus, J.Beck: Generalized two- and three-dimensional clipping, Computers & Graphics, 1978; 3; 1; 23-28.

[30] R.B.Vatti: A Generic Solution to Polygon Clipping. In Communications of the ACM 1992;35:7:56-63

[31] M.K.Agoston: Computer Graphics and Geometric Modeling -Implementation and Algorithms, Springer, 2005.

[32] H.Surmann, A.Nuchter, J.Hertzberg: An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments, Robotics and Autonomous Systems, 2003;45:3:181-198.

[33] N.Jones, B.McCrone, B.Walter, K.Pratt, D.Greenberg: Automated translation and thermal zoning of digital building models for energy analysis, Building Simulation 2013. Proceedings, Thirteenth International Conference of International Building Performance Simulation Association, 2013;202-209.

[34] C.Tymstra, R.W.Bryce, B.M.Wotton, S.W.Taylor, O.B.Armitage: Development and structure of Prometheus, the Canadian wildland fire growth simulation model, Natural Resources Canada Information Rep, 2010; NOR-X-417.

[35] 土棚善貴，金子順一，堀尾健一郎 : Voxel 表現に基づく多軸制御加工切削シミュレーションの大規模並列処理手法ーグラフィックハードウェアのアーキテクチャを考慮した状態記述 / 干渉判定法ー，精密工学会誌，79，5(2013)，467
[36] GLEW: The OpenGL Extension Wrangler Library, The OpenGL Extension Wrangler Library, http://glew.sourceforge.net/, (2020/11/04 アクセス)
[37] GLFW - An OpenGL library, GLFW, https://www.glfw.org/index.html, (2020/11/04 アクセス)
[38] Eigen, Main page, http://eigen.tuxfamily.org/index.php?title=Main_Page, (2020/11/04 アクセス)
[39] GitHub, Dear ImGui, https://github.com/ocornut/imgui, (2020/11/04 アクセス)

第2章　ポリゴン表現とVattiクリッピング

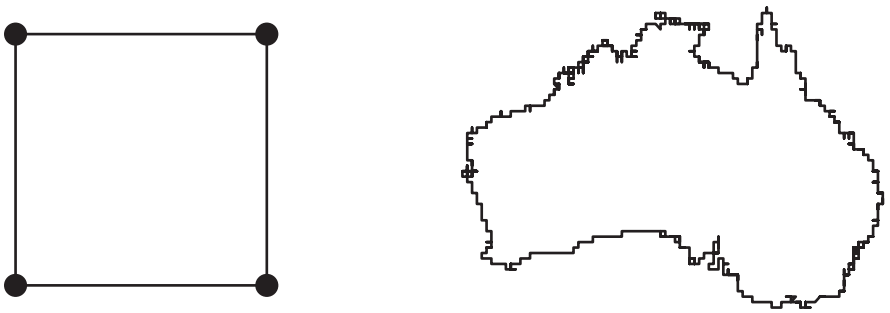
2.1　ポリゴン表現

本研究は従来のボクセル表現でなく，ポリゴン表現を使用して，切削シミュレータを開発する．このポリゴン表現の特徴やポリゴンの種類，本研究が対象とするポリゴンに関して説明する．

1章で簡単に説明したが, 多角形の面で表現されるポリゴン表現は, 図2.1に示すように, 少ない頂点数で単純な形状を表現できる．また，ポリゴン表現は頂点の集合で表現されるため，ポリゴンの空間分解能は頂点の座標値精度に依存し，この頂点の座標値精度はプログラム上では実数型で保存することができるため，非常に精度が良い．よって，単純な形状であれば，非常に高い空間分解能，少ない頂点数で形状を表現することができる．しかし，オーストラリアの海岸線のような複雑な形状を表現する場合は，大量の頂点数を必要とするため，描画などのさまざまな処理に時間が掛かる．

ポリゴン表現は図2.2に示すように，単純ポリゴンと複合ポリゴンに分けられる [1]．単純ポリゴンは，1つの頂点の羅列のみで表現されるが，複合ポリゴンは複数の頂点の羅列で表現され，穴の表現を行うことができる．単純ポリゴンは穴の表現を行うことができないが，処理が簡単であるため，複合ポリゴンより使用されている．本研究では，単純ポリゴンを対象として，開発を行った．他にも，凸ポリゴンと凹ポリゴンの2つに分けることができる．凸ポリゴンにはくぼみがなく，反対に凹ポリゴンには1つ以上のくぼみが存在する．Vattiクリッピングは凸ポリゴン，凹ポリゴンどちらも対応している．

また，ポリゴンは図2.3に示すような自己交差が発生する場合がある．この自己交差が発生しているポリゴンのことを自己交差ポリゴンと呼ぶ．また，水平のエッジがあるポリゴンを水平エッジポリゴンと呼ぶ．これら自己交差ポリゴンと水平エッジポリゴンどちらに対してもVattiクリッピングはブール演算を実施することができるが，切削シミュレータの特性上，自己交差が発生することはなく，水平エッジに対しては，水平エッジに対応したVattiクリッピングはアルゴリズムが複雑になるため，自己交差ポリゴン，水平エッジポリゴンを許容したVattiクリッピングの実装を行わなかった．また，水平エッジに関しては，水平エッジを回避するため，ポリゴン全体を微小に回転させ，水平エッジが発生しないようにし，対策を行った．



単純な形状（四角形）
頂点数：4

複雑な形状（オーストラリア）
頂点数：36111

図 2.1　単純な形状と複雑な形状の例

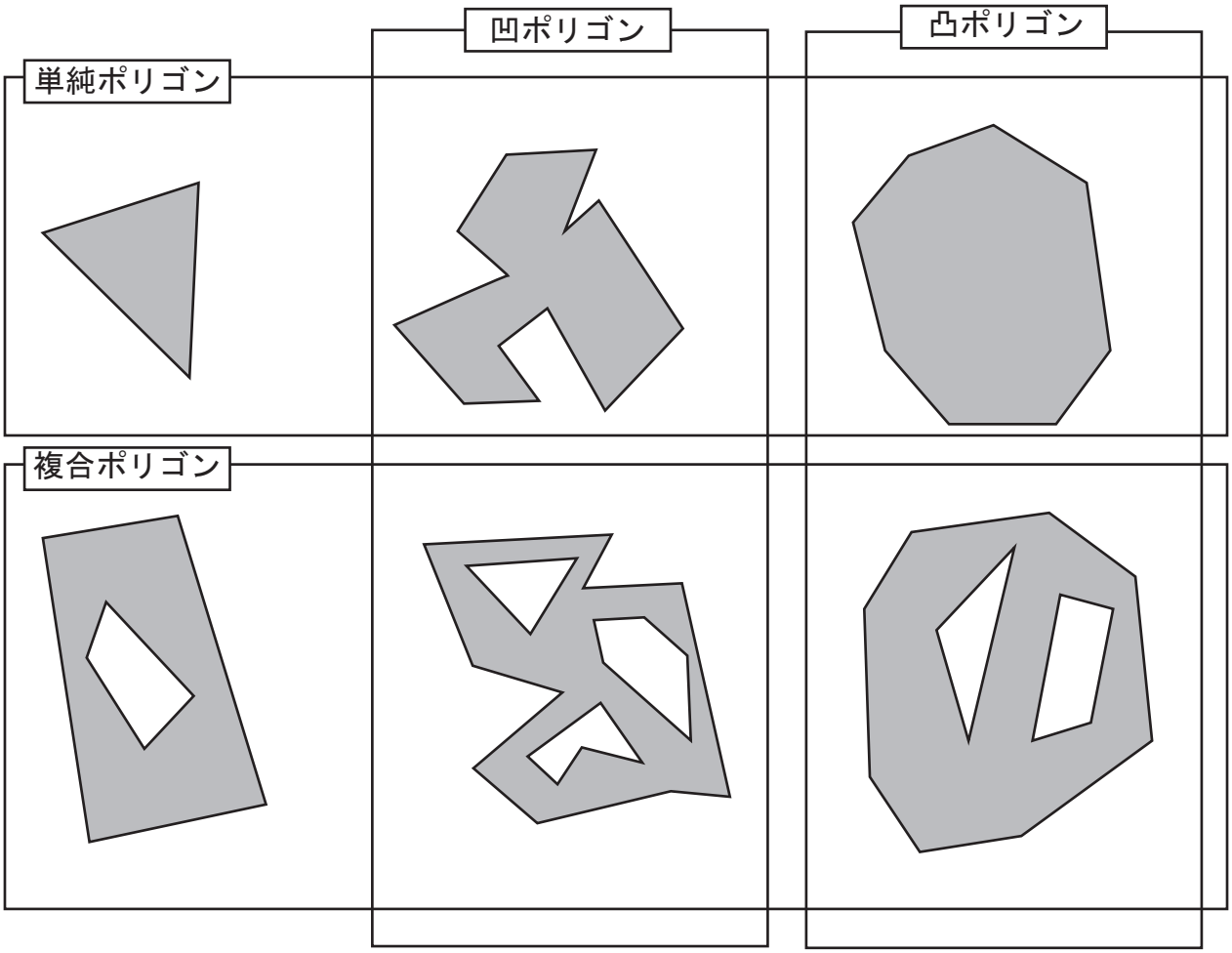


図 2.2　単純ポリゴンと複合ポリゴン

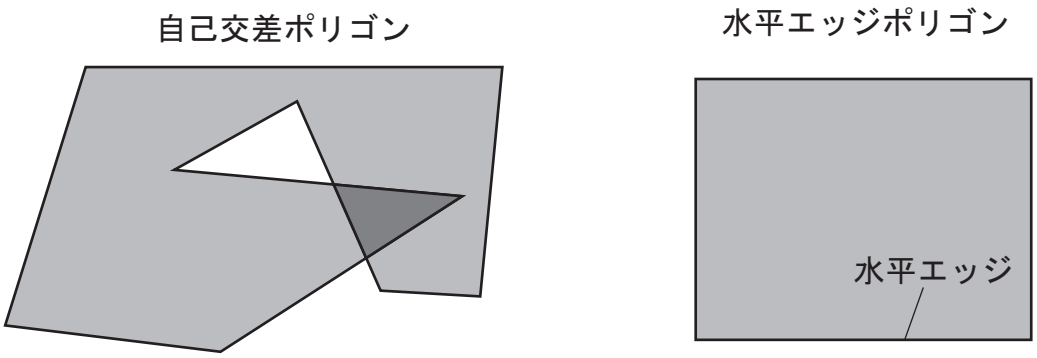


図 2.3　自己交差ポリゴンと水平エッジポリゴン

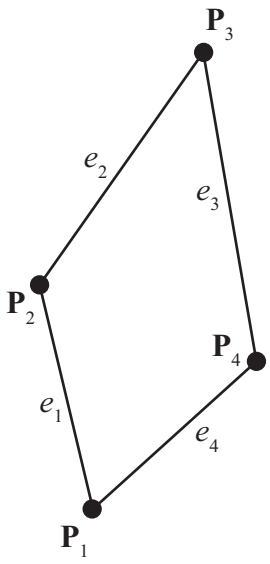
2.2　Vatti クリッピング

2.2.1　概要

Vatti クリッピングとは 2 次元ポリゴン同士のブール演算を高速に行うことができるアルゴリズムであり，ブール演算の和，差，積全てに対応することができる．Vatti クリッピングはポリゴンに対してブール演算を行う場合，スキャンビームと呼ばれるポリゴンの頂点を通る 2 本の走査線の間の水平部分を最下部から最上部まで移動し，徐々に出力ポリゴンを構築していく．交差探索等を行う際に，ポリゴンのエッジ全てに対して行わず，現在のスキャンビームにあるエッジにのみ行うため，処理を行うエッジ数が少なく，高速に行うことができる．また，Vatti クリッピングではポリゴンのエッジを LEFT，RIGHT と分類している．この分類によって，交差形態を区別している．この交差形態のことを交差タイプと呼び，交差タイプに応じた交差処理を行い，ブール演算後のポリゴンを出力している．

2.2.2　データ構造

Vatti クリッピングは，入出力データとして頂点のリスト構造であるポリゴンを使用しているが，内部ではこのポリゴンをエッジ構造に変換しており，エッジ構造を図 2.4 に示す．エッジは，サブジェクトポリゴンのエッジかクリップポリゴンのエッジかがわかるようになっており，上述した LEFT と RIGHT の分類の 2 つを用いて交差タイプを分類している．また，エッジ 1 つずつにこのエッジが出力ポリゴンに必要なエッジかどうかがわかるフラグと出力ポリゴンのポインタを保存しており，ブール演算の種類ごとにこのフラグと出力ポリゴンのポインタを用意する．今回は差と積のブール演算を使用するため，差と積のみ用意した．



Edge 構造

Coord	StartPoint	// 始点
Coord	EndPoint	// 終点
bool	Type	// サブジェクト or クリップ
bool	Side	//LEFT or RIGHT
bool	contribNot	// 出力ポリゴンに必要かどうか（差）
bool	contribAnd	// 出力ポリゴンに必要かどうか（積）
Edge	*Succ	// 次のエッジのポインタ
Edge	*Pair	// ペアのエッジのポインタ
Polygon	*OutputPolygonNot	// 出力ポリゴンのポインタ（差）
Polygon	*OutputPolygonAnd	// 出力ポリゴンのポインタ（積）

e_1
- StartPoint = $\mathbf{P}_1$
- EndPoint = $\mathbf{P}_2$
- Type = Subject or Clip
- Side = LEFT
- contribNot = true or false
- contribAnd = true or false
- Succ = e_2
- Pair = e_4
- OutputPolygonNot = デフォルトは NULL
- OutputPolygonAnd = デフォルトは NULL

e_3
- StartPoint = $\mathbf{P}_4$
- EndPoint = $\mathbf{P}_3$
- Type = Subject or Clip
- Side = RIGHT
- contribNot = true or false
- contribAnd = true or false
- Succ = NULL
- Pair = NULL
- OutputPolygonNot = デフォルトは NULL
- OutputPolygonAnd = デフォルトは NULL

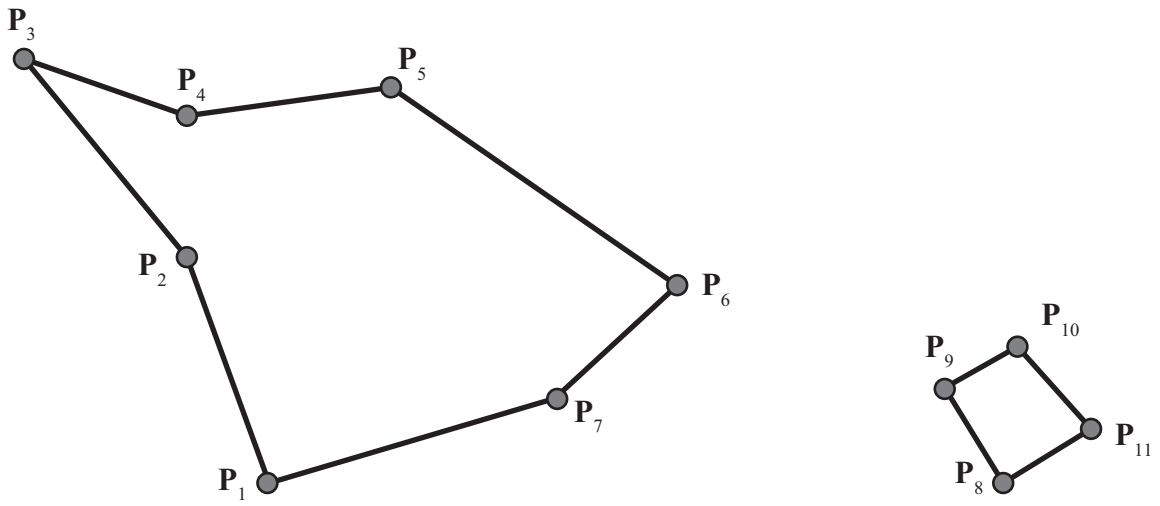
図 2.4　エッジ構造

2.2.3　アルゴリズムの詳細

図 2.5 に示すポリゴンでブール演算を行ったときの Vatti クリッピングの処理に関して説明する．このサブジェクトポリゴンとはブール演算の差の処理を行った際にくり抜かれるポリゴンであり，クリップポリゴンは逆にくり抜くポリゴンであり，切削シミュレータに応用する際には工作物がサブジェクトポリゴンで，工具がクリップポリゴンである．Vatti クリッピングではポリゴンを頂点のリストで表現しており，例えば．図 2.5 のサブジェクトポリゴンだと, 頂点を時計回りに羅列した $\mathbf{P}_1$,$\mathbf{P}_2$,$\mathbf{P}_3$,$\mathbf{P}_4$,$\mathbf{P}_5$,$\mathbf{P}_6$,$\mathbf{P}_7$ でポリゴンを表現している．これはブール演算後のポリゴン（以下出力ポリゴンと呼ぶ）でも同じである．

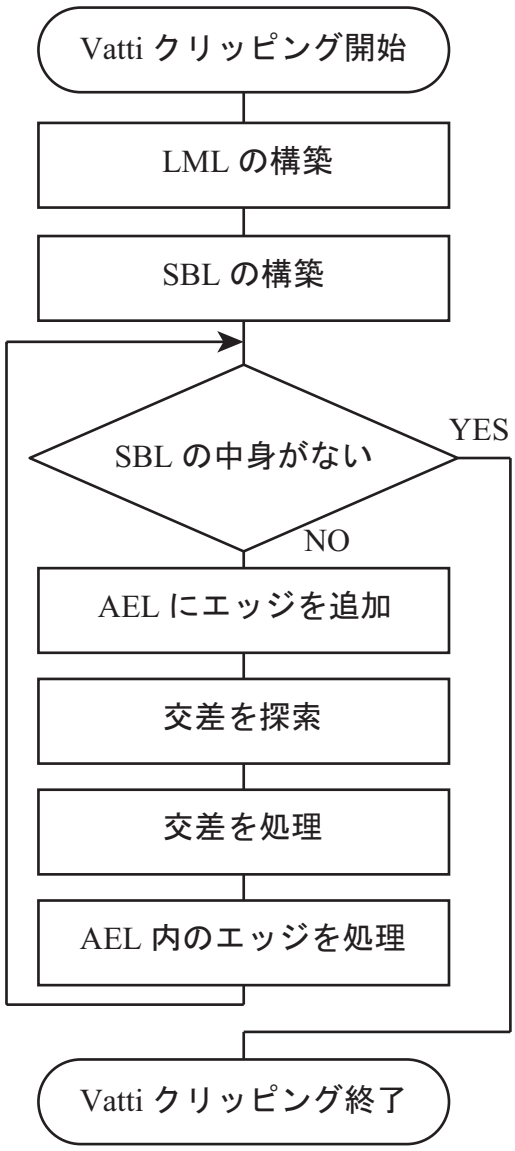
図 2.6 に Vatti クリッピングの大まかな処理のフローチャートを示す．このフローチャートに沿って説明を行う．

最初に LML と SBL の構築を行う．頂点の羅列で表現されたポリゴンを線分のエッジを用いた表現方法に変換する．このとき上述したように，エッジを LEFT，RIGHT に分類する．これは時計回りにポリゴンのエッジが並んでおり，最初のエッジの始点が y 座標値最小の頂点の場合，最初のエッジが LEFT になる．そして，エッジが上向きから下向きに変化すると LEFT から RIGHT へ変わる．例えば図 2.7 では，最初のエッジは $\mathbf{P}_1$-$\mathbf{P}_2$ のエッジ e_1 になりこのエッジは LEFT である．そして，$\mathbf{P}_3$-$\mathbf{P}_4$ のエッジ e_3 では $\mathbf{P}_4$ は $\mathbf{P}_3$ より y 座標が小さい下向きエッジであるため，このエッジは RIGHT になる．このようにして，エッジを構築していき，図 2.7 のように必要な情報をエッジに保存する．Type はサブジェクトポリゴンまたは，クリップポリゴンかを判断するためのパラメータである．この Succ は，このエッジの終点を始点とするエッジを保存している．例えば，e_1 の場合終点 $\mathbf{P}_2$ を始点とする e_2 が e_1 の Succ のエッジである．また，Pair はこのエッジの始点を始点とする他のエッジである．例えば，e_1 の場合終点 $\mathbf{P}_1$ を始点とする e_7 が e_1 の Pair のエッジである．そして, LML は Pair の組み合わせ（e_1 と e_7 や e_4 と e_3, この Pair の組み合わせを Bound と呼ぶ）を始点の y 座標値の小さい順にリスト化したものである．また，SBL は LML に保存され



(a) サブジェクトポリゴン　　(b) クリップポリゴン

図 2.5　Vatti クリッピングを行うポリゴンの例



LML：Local Minima List

SBL：ScanBeam List

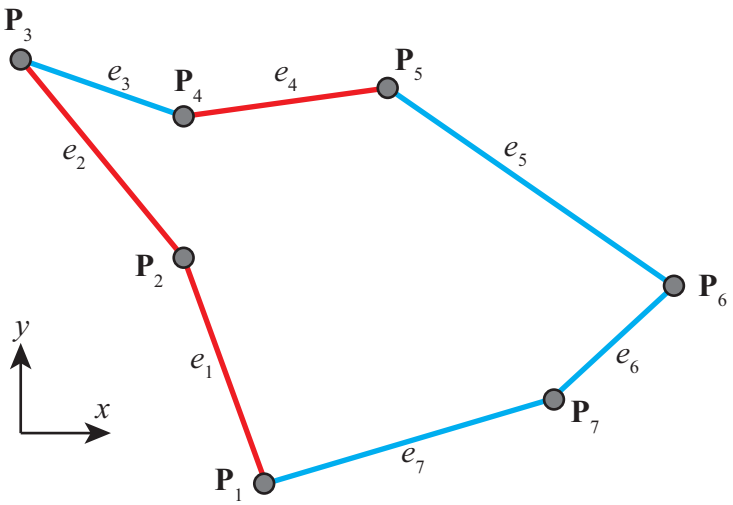
AEL：Active Edge List

図 2.6　Vatti クリッピングのフローチャート

ているエッジの始点と終点の y 座標値を昇順にソートしたリストである．これを全てのポリゴンで行う．

ここから，図 2.8 のように，サブジェクトポリゴンとクリップポリゴンが配置されている場合を考える．このとき，LML には（e_1，e_7），（e_8，e_{11}），（e_4，e_3）が保存され，SBL には $\mathbf{P}_{1y}$，$\mathbf{P}_{8y}$，$\mathbf{P}_{11y}$，$\mathbf{P}_{7y}$，$\mathbf{P}_{9y}$，$\mathbf{P}_{2y}$，$\mathbf{P}_{4y}$，$\mathbf{P}_{5y}$，$\mathbf{P}_{3y}$ が保存されている．まず，SBL の先頭から 2 つデータを取り出し，y_{bottom}，y_{top} とする．よって，$y_{\text{bottom}} = \mathbf{P}_{1y}$，$y_{\text{top}} = \mathbf{P}_{8y}$ となる．この y_{bottom} と y_{top} の幅がスキャンビームである．そして，データを取り出した後の SBL は SBL = $\mathbf{P}_{11y}$，$\mathbf{P}_{7y}$，$\mathbf{P}_{9y}$，$\mathbf{P}_{2y}$，$\mathbf{P}_{4y}$，$\mathbf{P}_{5y}$，$\mathbf{P}_{3y}$ となる．

次に AEL にエッジを追加する．AEL（Active Edge List）とは現在のスキャンビーム内にあるエッジを x 座標値が小さい順に保存されているリストである．この AEL に LML からエッジを追加する．LML 内の Bound の始点の y 座標値が y_{bottom} が同じ場合，Bound のエッジを AEL に追加する．よって，$y_{\text{bottom}} = \mathbf{P}_{1y}$，$y_{\text{top}} = \mathbf{P}_{8y}$ の場合，AEL = e_1，e_7 となる．この AEL にエッジを追加した場合，このエッジが出力ポリゴンに必要なエッジかどうかの判断を行い，この判定をエッジ構造体内の contrib に保存する．この判定は，実施するブール演算の種類ごとに行う必要があり，差の場合エッジの Type によっても判断方法が変わる．



e_1 { StartPoint = $\mathbf{P}_1$　Side = LEFT ; EndPoint = $\mathbf{P}_2$　Succ = e_2 ; Type = Subject　Pair = e_7 }

e_2 { StartPoint = $\mathbf{P}_2$　Side = LEFT ; EndPoint = $\mathbf{P}_3$　Succ = NULL ; Type = Subject　Pair = NULL }

e_3 { StartPoint = $\mathbf{P}_4$　Side = RIGHT ; EndPoint = $\mathbf{P}_3$　Succ = NULL ; Type = Subject　Pair = e_4 }

e_4 { StartPoint = $\mathbf{P}_4$　Side = LEFT ; EndPoint = $\mathbf{P}_5$　Succ = NULL ; Type = Subject　Pair = e_3 }

e_5 { StartPoint = $\mathbf{P}_6$　Side = RIGHT ; EndPoint = $\mathbf{P}_5$　Succ = NULL ; Type = Subject　Pair = NULL }

e_6 { StartPoint = $\mathbf{P}_7$　Side = RIGHT ; EndPoint = $\mathbf{P}_6$　Succ = e_5 ; Type = Subject　Pair = NULL }

e_7 { StartPoint = $\mathbf{P}_1$　Side = RIGHT ; EndPoint = $\mathbf{P}_7$　Succ = e_6 ; Type = Subject　Pair = e_1 }

LML = (e_1 , e_7) , (e_4 , e_3)

SBL= $\mathbf{P}_{1y}$, $\mathbf{P}_{4y}$

図 2.7　LML と SBL の構築

今回は本研究で使用する差と積のブール演算の判定方法を以下に示す.

・差の場合

エッジの Type が Subject

AEL の中でこのエッジより前に Clip のエッジが奇数個ある

contrib = false（このエッジは出力ポリゴンに不必要）

AEL の中でこのエッジより前に Clip のエッジが偶数個ある

contrib = true（このエッジは出力ポリゴンに必要）

エッジの Type が Clip

AEL の中でこのエッジより前に Clip のエッジが奇数個ある

contrib = false（このエッジは出力ポリゴンに必要）

AEL の中でこのエッジより前に Clip のエッジが偶数個ある

contrib = true（このエッジは出力ポリゴンに不必要）

・積の場合

AEL の中でこのエッジより前にこのエッジの Type でないエッジが奇数個ある

contrib = true（このエッジは出力ポリゴンに必要）

AEL の中でこのエッジより前にこのエッジの Type でないエッジが偶数個ある

contrib = false（このエッジは出力ポリゴンに不必要）

このようにして，AEL に追加したエッジが出力ポリゴンに必要か否か判断を行う．現在の AEL = e_1，e_7 を判断すると，e_1，e_7 共に Subject で Clip のエッジが AEL の中に無いため，差の contrib は true になり，積の contrib は false になる．このとき，contrib が true の場合，AddLocalMin の処理を行う．これは出力ポリゴンのリストを生成し，頂点をそのリストに追加する処理である．このとき，追加する頂点は始点の頂点の $\mathbf{P}_1$ となる．この出力ポリゴンのリストのポインタをエッジに保存する．よって，e_1，e_7 内に $\mathbf{P}_1$ が保存されている出力ポリゴンが保存されている．この出力ポリゴンを OP_1 とする．

そして，次の処理である交差判定，交差処理に移るが，この交差はスキャンビーム内で発生している交差のみ処理を行うので，現在のスキャンビームでは交差が発生していない

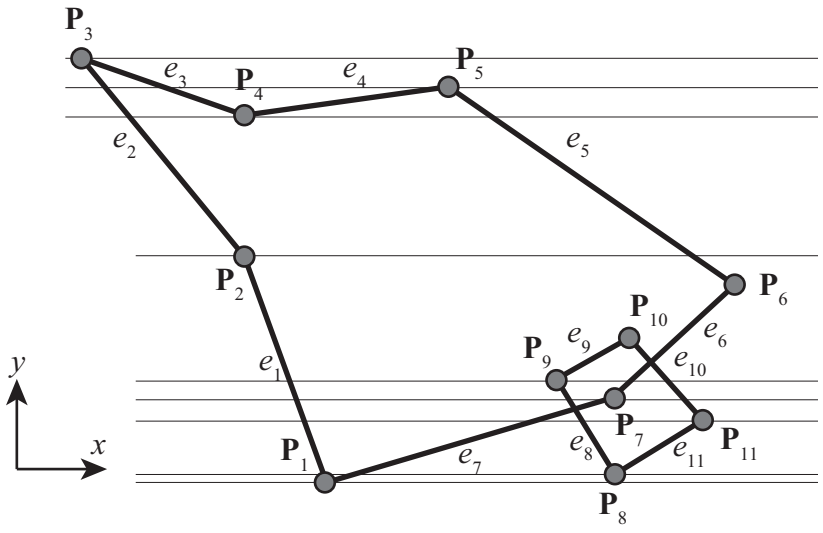


図 2.8　クリッピングの初期状態

ため，次の処理に移る．

次に AEL 内のエッジを処理を行うが，AEL 内に保存されているエッジの中で，終点の y 座標値が y_{top} の場合処理を行うため，何も行わない．ここまでが 1 回の処理になり，この処理を最上部まで行う．

スキャンビームが $y_{\text{bottom}} = \mathbf{P}_{11y}$，$y_{\text{top}} = \mathbf{P}_{7y}$ の場合を考える．このときの AEL などの情報をまとめたものを図 2.9 に示す．LML から AEL に追加するエッジはないが，e_7 と e_8 が交差していることがわかる．この交差処理に関して説明する．交差処理は交差タイプごとに処理を行っており，RIGHT，LEFT の分類を使用して交差タイプを構築できる．例えば，エッジの始点の x 座標値が小さい順にクリップポリゴンの LEFT とサブジェクトポリゴンの RIGHT が交差する場合，LC-RS といった交差タイプになる．この交差タイプごとに交差処理を行っている．交差タイプごとの交差処理を図 2.10 に示す．e_7 と e_8 の交差は RS-LC であるため，差の場合 RI，積の場合 MX となる．LI，RI，MN，MX のそれぞれの処理を以下に示す．このとき，エッジの始点の x 座標値が小さい順に Edge_1，Edge_2 が交差しているとする．

$\text{AEL} = e_1, e_7, e_8, e_{10}$　　$\text{OP}_1 = \mathbf{P}_1$

$\text{SBL} = \mathbf{P}_{9y}, \mathbf{P}_{10y}, \mathbf{P}_{2y}$　　$\text{LML} = (e_4, e_3)$

e_1 {
- contribNOT = true
- contribAND = false
- OutputPolygonNOT = OP_1
- OutputPolygonAND = NULL

}

e_7 {
- contribNOT = true
- contribAND = false
- OutputPolygonNOT = OP_1
- OutputPolygonAND = NULL

}

e_8 {
- contribNOT = false
- contribAND = false
- OutputPolygonNOT = NULL
- OutputPolygonAND = NULL

}

e_{10} {
- contribNOT = false
- contribAND = false
- OutputPolygonNOT = NULL
- OutputPolygonAND = NULL

}

図 2.9　$y_{\text{bottom}} = \mathbf{P}_{11y}$，$y_{\text{top}} = \mathbf{P}_{7y}$ のときの AEL などの情報

差の場合	積の場合
(RC-LS) or (LS-RC) → LI	(LC-LS) or (LS-LC) → LI
(RS-LC) or (LC-RS) → RI	(RC-RS) or (RS-RC) → RI
(RS-RC) or (LC-LS) → MN	(LS-RC) or (LC-RS) → MN
(RC-RS) or (LS-LC) → MX	(RS-LC) or (RC-LS) → MX

図 2.10　交差タイプと交差処理の分類

LI の処理→ AddLeft（$Edge_2$）

Edge2 の Output Polygon の末尾に交点を追加

RI の処理→ AddRight（$Edge_1$）

Edge1 の Output Polygon の先頭に交点を追加

MN の処理→ AddLocalMin（$Edge_1$，$Edge_2$）

新しい Output Polygon を作成し，交点を追加

MX の処理→ AddLocalMax

差の場合

Edge1 が AEL で保存されている位置

奇数番目→ AddLeft（$Edge_1$）

偶数番目→ AddRight（$Edge_1$）

積の場合

Edge1 が AEL で保存されている位置

奇数番目→ AddRight（$Edge_1$）

偶数番目→ AddLeft（$Edge_1$）

$Edge_1$ と $Edge_2$ の出力ポリゴンが異なる場合

$Edge_1$ の出力ポリゴン（a b c），$Edge_2$ の出力ポリゴン（x y z）の場合

$Edge_1$ が AEL で保存されている位置によって出力ポリゴンを変更

差の場合

奇数番目→ a b c x y z

偶数番目→ x y z a b c

積の場合

奇数番目→ x y z a b c

偶数番目→ a b c x y z

この処理を行ったあと，AEL 内のエッジの位置と出力ポリゴンを入れ替え，contrib を変更する．よって，交点を $\mathbf{P}_{12}$ として，e_7 と e_8 の交差の処理を行うと図 2.11 のようになる．e_7 と e_8 の contrib が true から false，false から true へ変更されており，各 Output Polygon も入れ替わっている．

次に AEL 内のエッジの処理に関して説明する．スキャンビームを 1 つ進めた $y_{\mathrm{bottom}} = \mathbf{P}_{7y}$，$y_{\mathrm{top}} = \mathbf{P}_{9y}$ のときを考える．このとき，e_8 の終点は $\mathbf{P}_9$ であるため，次のスキャンビームに移動する際に，次のエッジ e_9 へ変更する必要がある．また，このエッジ e_8 は出力ポリゴンに必要なエッジであるため，出力ポリゴンに頂点 $\mathbf{P}_9$ を追加する必要がある．この処理するエッジを Edge として，この処理を以下に示す．

$\mathrm{AEL} = e_1, e_8, e_7, e_{10}$　　$\mathrm{OP}_1 = \mathbf{P}_{12}, \mathbf{P}_1$　　$\mathrm{OP}_2 = \mathbf{P}_{12}$

$\mathrm{SBL} = \mathbf{P}_{9y}, \mathbf{P}_{10y}, \mathbf{P}_{2y}$　　$\mathrm{LML} = (e_4, e_3)$

e_1 { contribNOT = false; contribAND = false; OutputPolygonNOT = OP_1; OutputPolygonAND = NULL }

e_7 { contribNOT = false; contribAND = true; OutputPolygonNOT = NULL; OutputPolygonAND = OP_2 }

e_8 { contribNOT = true; contribAND = true; OutputPolygonNOT = OP_1; OutputPolygonAND = OP_2 }

e_{10} { contribNOT = false; contribAND = false; OutputPolygonNOT = NULL; OutputPolygonAND = NULL }

図 2.11　交差処理後の AEL などの情報

Succ が NULL（MX）
　　AddLocalMax（Edge，AEL での Edge の次のエッジ）
　　Edge，AEL での Edge の次のエッジを AEL 内から削除する
Edge が AEL 内で奇数番目（LI）
　　差の場合
　　　　AddLeft（Edge）
　　積の場合
　　　　AddRight（Edge）
　　Edge を Succ のエッジに変更する
Edge が AEL 内で偶数番目（RI）
　　差の場合
　　　　AddRight（Edge）
　　積の場合
　　　　AddLeft（Edge）
　　Edge を Succ のエッジに変更する

この処理を行い，Edge を Succ のエッジに変更した場合は，Succ のエッジに Edge の contrib や OutPut Polygon などの情報を引き継がせる必要がある．そして，Succ のエッジの終点の y 座標値を SBL に追加し，SBL のソートを行う．

以上の処理を行い，ブール演算を実施する．

2.2.4　Vatti クリッピングの問題点

この Vatti クリッピングを切削シミュレータに応用する場合，図 2.12 のようにサブジェクトポリゴンで工作物，クリップポリゴンで工具を表現し，クリップポリゴンを一定量動かし，ブール演算を実施することによって，加工形状や切削体積を計算する．ここで，Vatti クリッピングは 2 次元でのみ使用できるアルゴリズムであるため，本研究では工具軸方向から見た形状で，エンドミルの形状を 1 回転した円形状として考える．よって，ブール演算の差によって計算できる形状が加工途中形状であり，積によって，切削面積が計算できる．しかし，Vatti クリッピングを切削シミュレータに実装するにあたって，下記に示す 3 つの課題が存在する．

(1) ブール演算に掛かる計算時間は頂点数に依存し，シミュレーションが進むにつれて，頂点数が増えるため，ブール演算時間も同時に増加してしまう．
(2) 円を表現する場合大量の頂点を用いた多角形近似が必要になる．
(3) Vatti クリッピングを 3 次元シミュレータへと拡張する必要がある

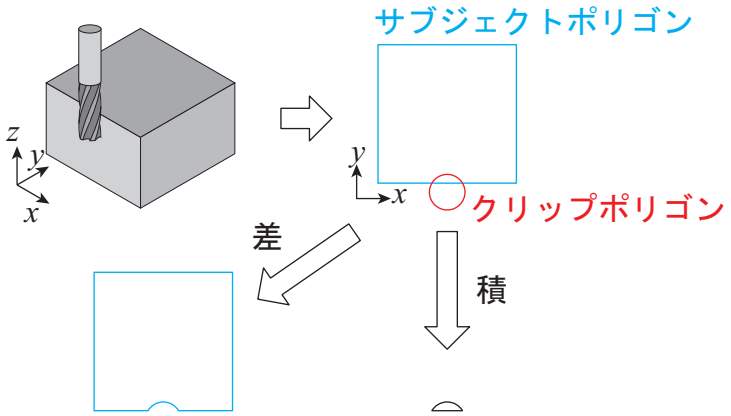


図 2.12　Vatti クリッピングを切削シミュレータに応用する方法

(1) の問題点を確認するために，実際に Vatti クリッピングを実装し計算時間を測定した．サブジェクトポリゴンは四角形，クリップポリゴンは円を模した 10 角形で表現し，クリップポリゴンをサブジェクトポリゴンの左側から横方向に微小移動とブール演算を繰り返し行った．ブール演算の計算時間とサブジェクトポリゴンの頂点数の関係を図 2.13 に示す．横軸にサブジェクトポリゴンの頂点数，縦軸にブール演算に要した計算時間を示す．今回使用した CPU コア数は 1 コアである．計算時間を見てみると，サブジェクトポリゴンの頂点数の増加によって，計算時間が増加していることがわかる．これは，頂点数が多いとエッジが多くなり，1 つのスキャンビーム内に大量のエッジが存在するケースが発生する．この場合，スキャンビーム内のエッジのみ交差探索などを行っていて高速にブール演算できる Vatti クリッピングであっても，スキャンビーム内に大量のエッジが存在すると探索に計算時間が多く掛かる．また，スキャンビームは全ての頂点の y 座標値で作られ，最下部から最上部までスキャンビームを移動させ出力ポリゴンを生成するため，頂点が多いと何度もスキャンビームを移動させる必要があり，一連の処理を何度も行い計算時間が多く掛かる．

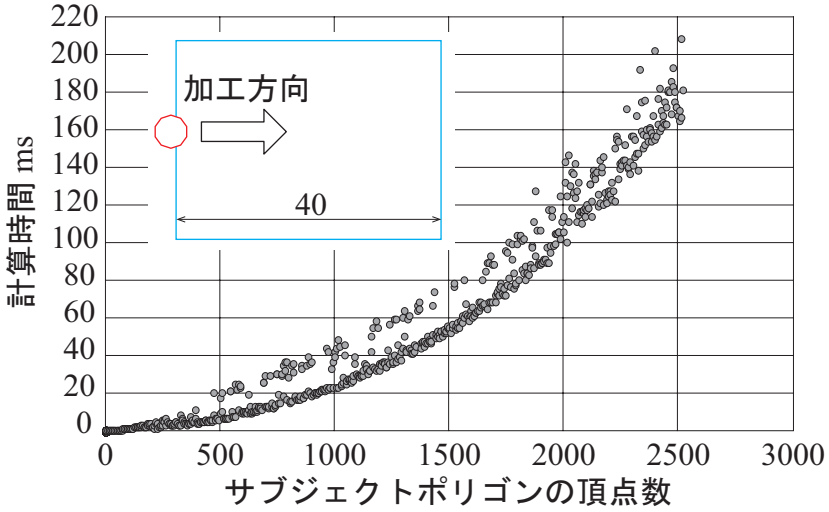


図 2.13　Vatti クリッピングの計算時間

次に，Vatti クリッピングは直線のエッジで構成されているポリゴンに対して使用できるアルゴリズムのため，円の表現は多角形近似になる．しかし，工具は円で表現されるため多角形で表現すると精度に問題がある．そこで，精度を良くするために大量の頂点を使用して表現した場合，図 2.14 に示すように形状誤差は小さくなっているが，計算時間は多く掛かってしまう．この図 2.14 は四角形のサブジェクトポリゴンを 4 回ブール演算した合計の計算時間である．例えば，頂点数 30 の場合，計算時間は 0.35 ms と高速だが，直径 5 mm の円との誤差は約 10 μm となる．また，頂点数 180 の場合，誤差は 0.3 μm となるが，計算時間は約 5 倍の 1.55 ms となる．よって，上述したように，計算時間は頂点数に依存しており，精度に関しても頂点数に依存しているため，直線のみで工具，工作物を表現し，精度と計算時間を両立した切削シミュレータ開発は難しいことがわかる．

最後に，Vatti クリッピングは 2 次元でのみ行えるアルゴリズムであるため，3 次元に対応させるためのアルゴリズムが必要である．

これら 3 つの問題を解決しなければ，高速で高精度の切削シミュレータを開発することができない．これらに対する解決策を次章で述べる．

参考文献

[1] Fletcher Dunn，Ian Parberry，（松田晃一 訳）：実例で学ぶゲーム 3D 数学，オーム社，2011

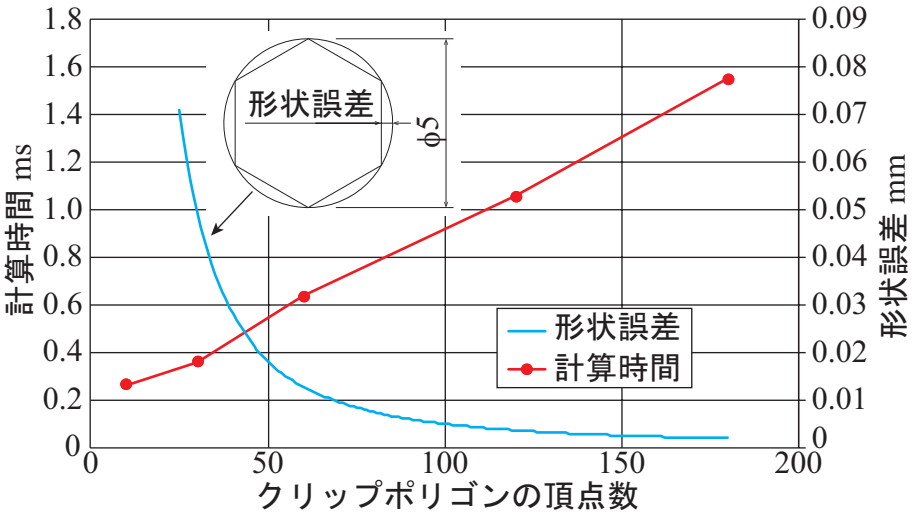


図 2.14　クリップポリゴンの頂点数と計算時間，形状誤差の関係

第 3 章　切削シミュレータ開発

3.1　はじめに

2 章で Vatti クリッピングのアルゴリズムと Vatti クリッピングを切削シミュレータに応用する上での 3 つの問題点を説明した．本章では，問題点の解決方法を提案し，その解決方法の有効性を示す．

3.2　頂点数増加による計算時間の増加

2 章で述べたように，Vatti クリッピングは頂点数が多いポリゴンに対してブール演算を行う場合，多くの計算時間を必要とする．工作物を表現するサブジェクトポリゴンに対して，工具を表現するクリップポリゴンでブール演算の差を行うと，サブジェクトポリゴンにクリップポリゴンが転写されたようになるため，頂点数が増え，切削シミュレーションを行うと徐々にサブジェクトポリゴンの頂点数増加につれ，計算時間が徐々に増加していく．よって，複雑な形状の加工を行うシミュレーションや加工距離が長いシミュレーションを行う場合は計算時間が増加し易く，高速な切削シミュレータ開発が難しい．

この問題を解決する方法として，例えば頂点と頂点の間隔が微小であった場合，頂点の削除を行ったり，サブジェクトポリゴンの近似を行い，大量の頂点で表現されていたサブジェクトポリゴンを少ない頂点数で表現する方法が考えられる．しかし，この頂点の削除による問題解決方法では工作物形状の変更になるため，現実とは異なったシミュレーション結果になる可能性がある．よって，本研究では，頂点の削除以外の方法で頂点数増加による計算時間の増加の問題点を解決する．

そこで，本研究が着目したのが工作物と工具の大きさの比率である．一般的な加工では，図 3.1 に示すように，工具と工作物の大きさを比較すると，工作物の方が工具より大きく

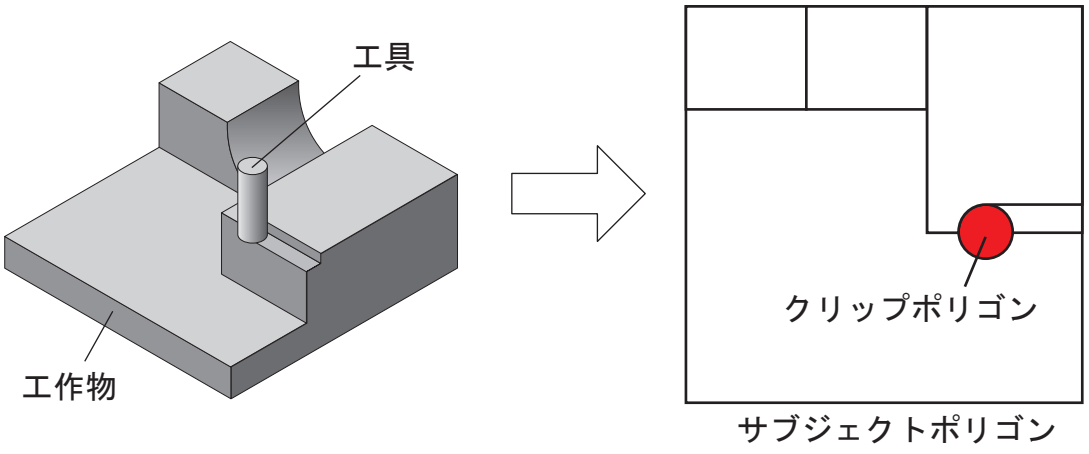


図 3.1　工具と工作物のサイズ

なる．工具と工作物を表現するクリップポリゴン，サブジェクトポリゴンも同様で，サブジェクトポリゴンの方がクリップポリゴンより大きくなる．よって，ブール演算を行う際には，クリップポリゴン付近のサブジェクトポリゴンのみブール演算の影響を受け，他の大部分ではブール演算を行っても変化がなく，言わば，無駄な計算を行っている．そこで，本研究では，ブール演算に必要な部分を的確に取り出し，その部分のみブール演算を行うことができれば，無駄な計算をすることなく，より高速に計算を行えると考えた．

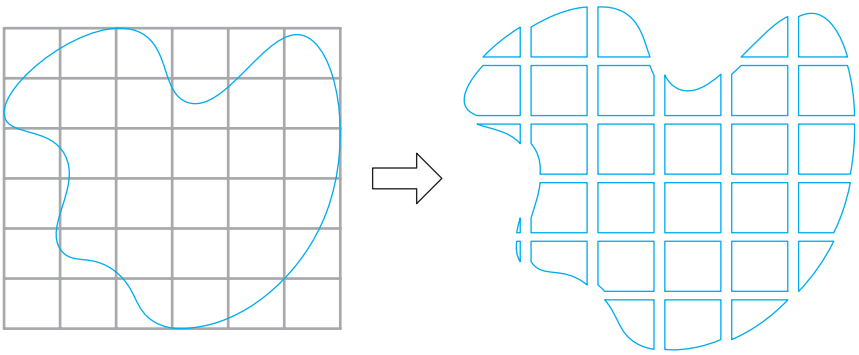
そこで，本研究では，サブジェクトポリゴンを格子状に分割する方法を考えた．これは下記に示すような方法でブール演算に必要な部分を判別し，その部分のみブール演算を行い計算時間の削減を行う．

(1) サブジェクトポリゴンを格子状に分割する．（図 3.2(a)）
(2) クリップポリゴンを囲む四角形を使用し，AABB(Axis-Aligned Bounding Box) を用いてクリップポリゴンの周辺部分をピックアップする．（図 3.2(b)）
(3) ピックアップされたサブジェクトポリゴンごとにブール演算を実施する．
(4) ピックアップされたサブジェクトポリゴンを差の出力ポリゴンで更新する．（図 3.2(c)）

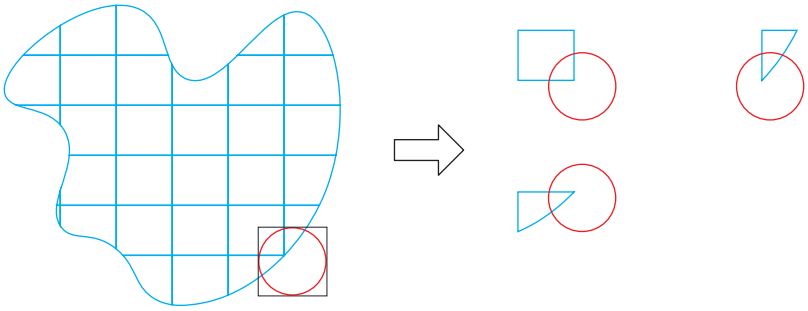
この AABB とは，バウンディングボリュームと呼ばれる交差判定のしやすい形状の一種で，軸方向に整列した四角形のバウンディングボリュームであり，単純な計算によって，干渉判定が行える CG 分野やゲーム分野で使用されている技術である [1]．この技術を使用して，クリップポリゴン周辺のサブジェクトポリゴンを取り出し，必要な部分のみブール演算の対象とし，ブール演算を行う頂点数を削減することができる．

図 3.3 に示すサブジェクトポリゴン，クリップポリゴンを使用して計算時間を測定した結果を図 3.4 に示す．四角形のサブジェクトポリゴンに対して，10 角形のクリップポリゴンでブール演算を行い，サブジェクトポリゴンは 5 mm で格子状に分割した．この結果を見てみると，頂点数が増加しても計算時間が一定以上増加しなくなった．また，最大で約 200 ms 掛かっていた計算時間が 9 ms になり，約 96 % 計算時間を削減することに成功した．

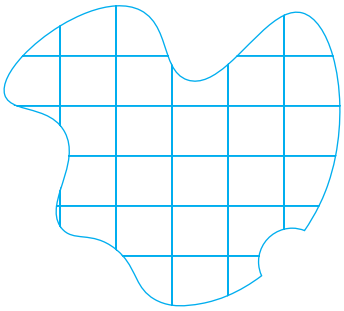
今回クリップポリゴンと格子のサイズを同じにした．これは 2 章で説明したように，図 3.5 に示すようなポリゴンの中にポリゴンがあり，穴が空いているような複合ポリゴンに許容して実装していないため，クリップポリゴンより大きくすることができないためである．また，小さくしすぎると一度にピックアップされる格子が多くなり，それぞれに対して Vatti クリッピングを行うため，却って計算時間が掛かる可能性があるため，今回はクリップポリゴンと同じサイズで格子状にサブジェクトポリゴンを分割した．



(a) サブジェクトポリゴンを格子状に分割



(b) AABB によるサブジェクトポリゴンの選択



(c) サブジェクトポリゴンの更新

図 3.2　頂点数増加による計算時間増加の対策

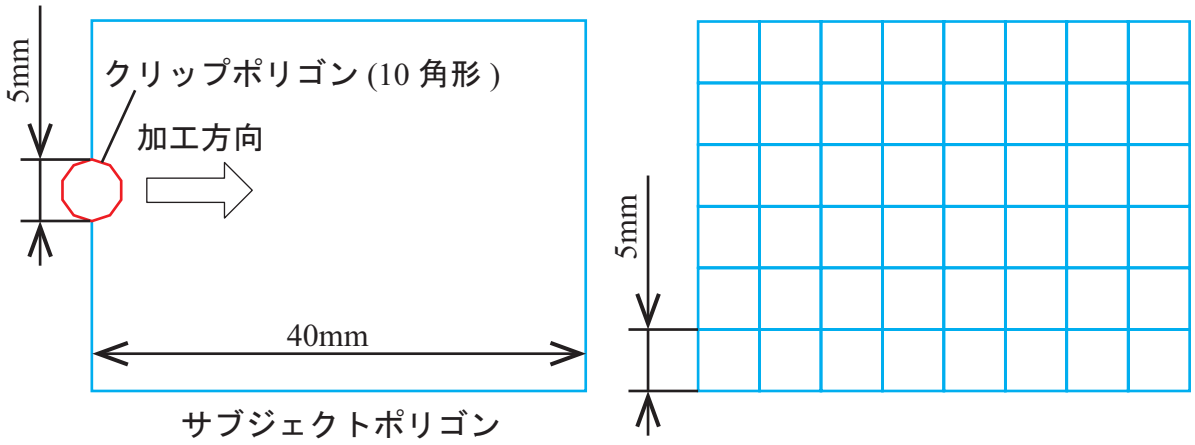


図 3.3　計算時間測定に使用したサブジェクトポリゴンとクリップポリゴン

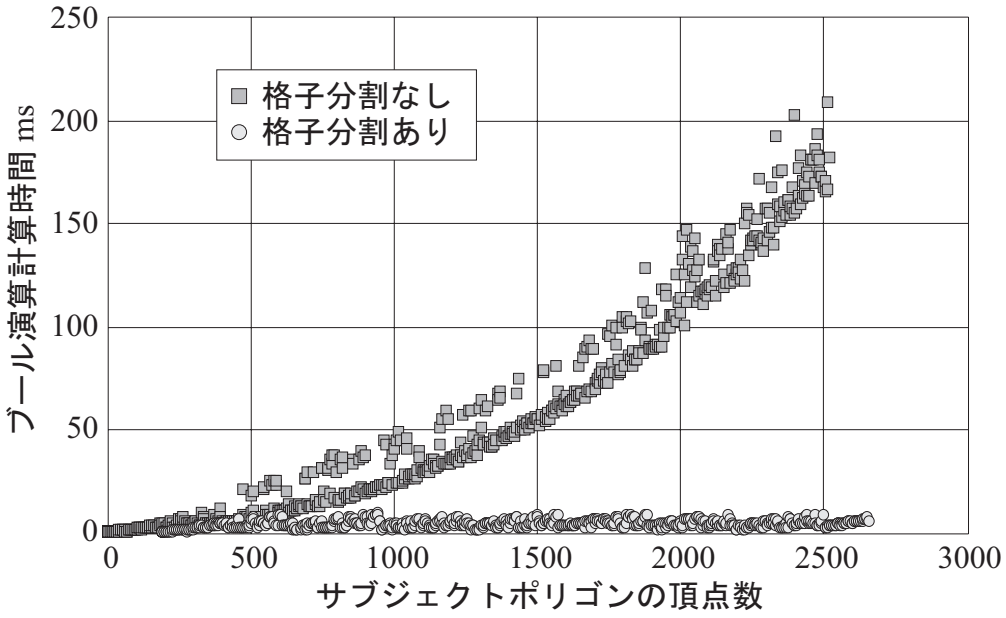


図 3.4　格子分割ありなしによるブール演算の計算時間比較

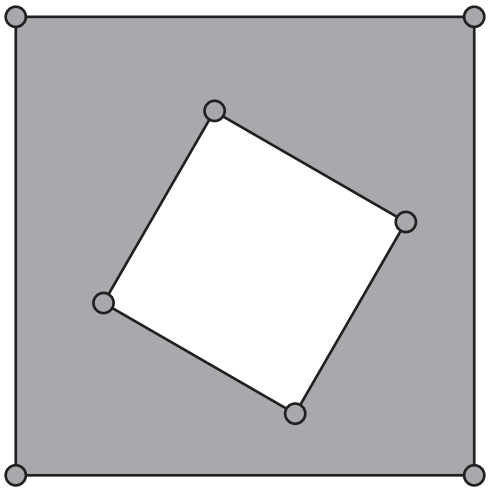


図 3.5　許容しないポリゴン形状

3.3　円弧に対応した改良型 Vatti クリッピング

切削シミュレータにおいて, 円の表現は非常に重要である．例えば, 図 3.6 に示すように，エンドミルを工具軸方向から見て，エンドミルの回転掃引断面形状を考えてみると円形状になる．よって，主軸 1 回転ごとに工具を移動させる場合の切削シミュレータでは，エンドミル形状を円形状に近似できるためである．

2 章でも述べたが，直線を用いた多角形の円近似では，近似精度を高くすると大量の頂点を用いる事になり，計算時間が多く掛かってしまう．一方，頂点数を少なくすると計算時間は短いが，精度が悪くなってしまう．そこで，本研究では Vatti クリッピングを円弧に対応させることによって，精度と計算時間の両立を目指す．

最初に円弧に関するデータ構造の変更点を説明する．円弧は図 3.7 にあるように，x，y 座標最大，最小値に頂点を用意する．これは，Vatti クリッピングではスキャンビームを最下部から最上部まで順々に移動して，徐々に出力ポリゴンを構築するため，エッジの途中で y 座標値が上下するエッジを処理することができないためである．よって，円を表現する際には，4 つの頂点が必要になる．Vatti クリッピングでは 2 章で説明したように，頂点のリストでポリゴンを表現しており，この頂点のリストで入出力を行っている．直線のみでの Vatti クリッピングでは頂点座標値のみで問題なかったが，円弧に対応させる場合は頂点座標値だけでなく，円弧か直線か（以下 Form と呼ぶ）を判断できるようにしなければならない．そこで，本研究では図 3.8 のように，頂点の座標値と共にその頂点を始点と

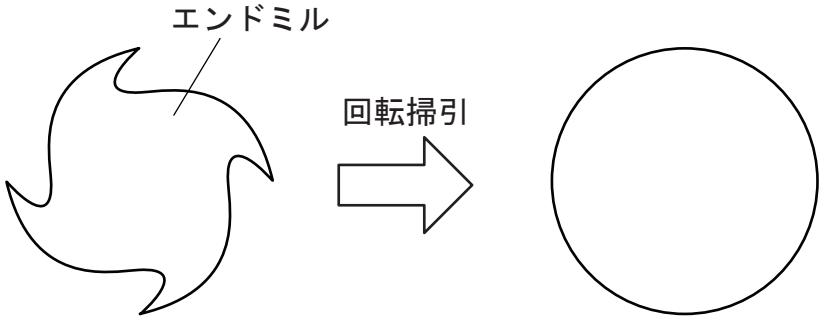


図 3.6　エンドミルの回転掃引形状

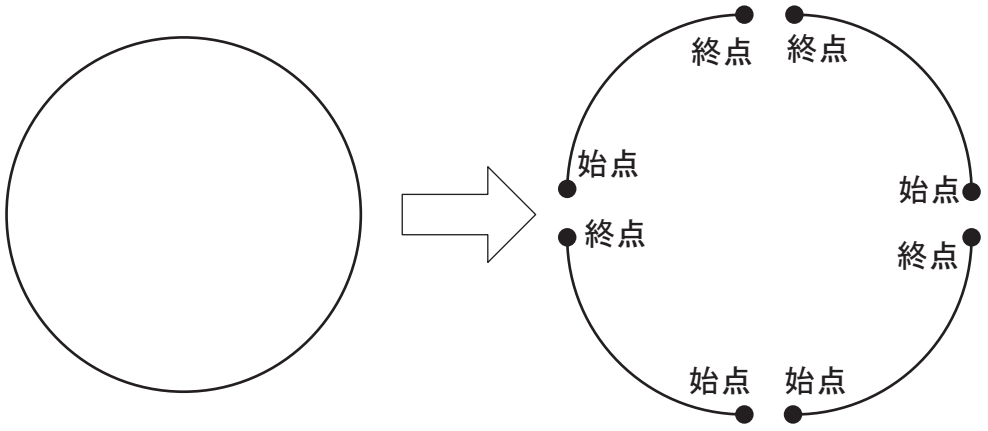


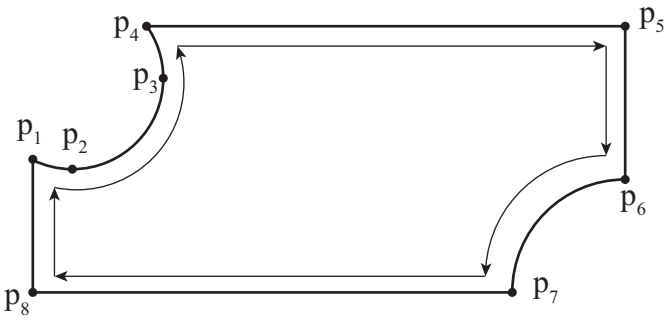
図 3.7　エッジの定義

するエッジが直線か円弧か判断できるプロパティ（Edge Form と呼ぶ）を保存する．また，円弧の場合は，中心座標値と円弧の半径値を保存する．これらの条件とデータ構造で Vatti クリッピングを円弧に対応させる．

Vatti クリッピングに円弧を対応させるためには既存の Vatti クリッピングに下記に示す 2 つの処理を追加する必要がある．

（1）AEL 内のエッジの処理部分で出力ポリゴンに追加する頂点の Form を決める
（2）交差処理部分で出力ポリゴンに追加する交点の Form を決める

どちらも，頂点を出力ポリゴンに追加するときの Form の決定方法に関する部分である．また，円弧の交差判定や交点座標の計算をもちろん追加する必要がある．



Polygon = p_1 - p_2 - p_3 - p_4 - p_5 - p_6 - p_7 - p_8

p_1 頂点座標値 = (p_{1x}, p_{1y}) Edge Form = ARC	p_5 頂点座標値 = (p_{5x}, p_{5y}) Edge Form = LINE
p_2 頂点座標値 = (p_{2x}, p_{2y}) Edge Form= ARC	p_6 頂点座標値 = (p_{6x}, p_{6y}) Edge Form= ARC
p_3 頂点座標値 = (p_{3x}, p_{3y}) Edge Form = ARC	p_7 頂点座標値 = (p_{7x}, p_{7y}) Edge Form = LINE
p_4 頂点座標値 = (p_{4x}, p_{4y}) Edge Form = LINE	p_8 頂点座標値 = (p_{8x}, p_{8y}) Edge Form= LINE

図 3.8　Edge Form の追加

最初に（1）の AEL 内のエッジの処理部分で行う Form の決め方を説明し，以下に Form の決め方を示す.

LI，RI の場合

処理するエッジが LEFT

出力ポリゴンに追加する頂点の Form，中心座標値，半径値は処理するエッジの Succ と同じにする

処理するエッジが RIGHT

出力ポリゴンに追加する頂点の Form，中心座標値，半径値は処理するエッジと同じにする

MX の場合

処理するエッジが LEFT

出力ポリゴンに追加する頂点の Form，中心座標値，半径値は処理するエッジの AEL 内で次のエッジと同じにする

処理するエッジが RIGHT

出力ポリゴンに追加する頂点の Form，中心座標値，半径値は処理するエッジと同じにする

図 3.9 にこの処理に関する模式図を示す．例えば，e_1 のエッジを処理する場合，LI, LEFT であるため，出力ポリゴンに追加する頂点は，座標値 p_2 で，e_1 の Succ である e_2 の EdgeForm が円弧であるため，Form を円弧にし，e_2 の中心座標値と半径値を使用する．一方，e_{10} を処理する場合は RI，RIGHT であるため，出力ポリゴンに追加する頂点は，座標値 p_{10} で, e_{10} の EdgeForm が直線であるため, Form を直線とする．MX の場合である e_6 のエッジを処理する場合，出力ポリゴンに追加する頂点は，座標値 p_7 で，e_6 の AEL 内で次のエッ

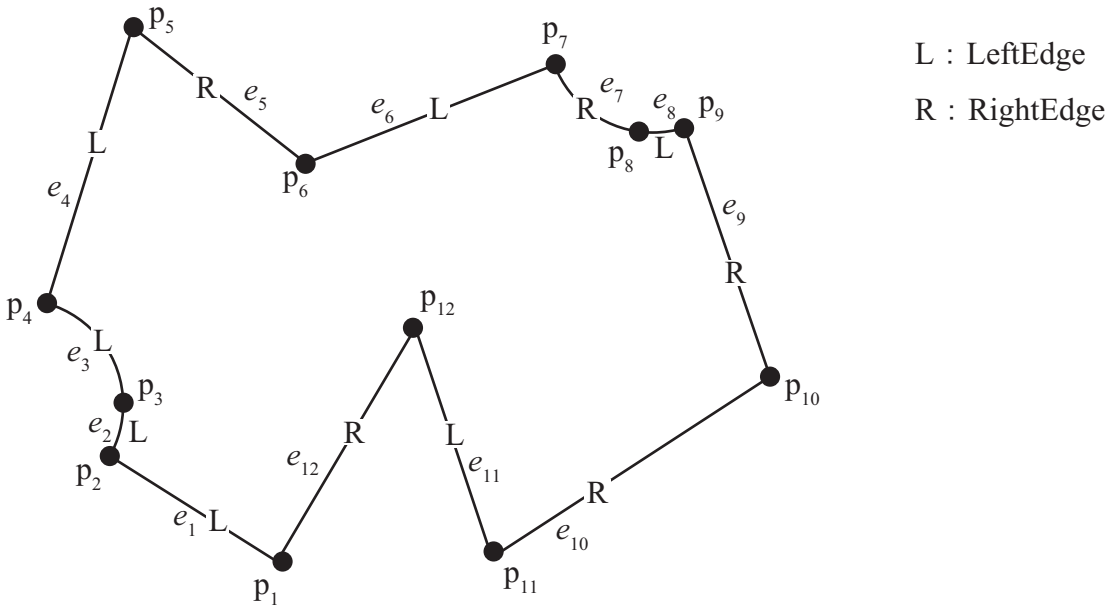


図 3.9　円弧エッジを許容したときの模式図

ジである e_7 の EdgeForm が円弧であるため，Form を円弧にし，e_7 の中心座標値と半径値を使用する．このようにして AEL 内のエッジを処理する．

次に，(2) の交差処理部分での Form を決める方法を説明する．交差処理では，交差タイプによって，交差処理の内容を変更していたが，この Form を決定する際も同様に交差タイプによって Form が変わる．また，ブール演算する際に，直線と円弧が交差した場合，交点の Form が直線か円弧かを決める．また，円弧と円弧が交差した場合に関しては，交点の Form は基本的には円弧になるが，交差しているどちらの円弧の中心座標値を使用するのかを交差タイプを用いて決定する．この交差タイプと Form の選択，中心座標値の選択を表 3.1 に示す．このルールに基づいて交差処理を行う．例えば，サブジェクトポリゴンが直線，クリップポリゴンが円弧で LS-LC の交差のときは，ブール演算の NOT では交点の Form は円弧になる．他には，同じ円弧と円弧，もしくは円弧と直線が交点を 2 つ持つ場合に関しては，オリジナルの Vatti クリッピングと同様に y 座標値が小さい交点から交差処理を行う．また通常，交差処理を行う場合は，AEL でのエッジの位置を入れ替える必要があるが，エッジ同士が接して交差している場合は AEL のエッジを入れ替える処理を行わない．

この 2 つの処理を Vatti クリッピングに追加し，円弧に対応させた．これによって，ブール演算後のサブジェクトポリゴンの形状と計算時間がどのように変わるかを確認した．ブール演算後の形状を図 3.10，計算時間を図 3.11 に示す．使用したサブジェクトポリゴンは図 3.3 と同じで，クリップポリゴンを頂点 4 点で表現した円とした．比較対象としては，図 3.4 の格子分割ありの結果と比較を行う．もちろん，円弧対応した計算時間結果もサブジェクトポリゴンの格子分割を行っている．また，クリップポリゴンは 0.065 mm ずつ移動させブール演算を行っており，このときの理論表面粗さは 0.2 μm となる．形状を確認した結果を見てみると, 10 角形のクリップポリゴンを使用したときは, 表面粗さが 0.01 mm，100 角形のクリップポリゴンを使用しても 1 μm となり，理論表面粗さと異なる結果となっている．一方，円弧を使用した場合は，0.2 μm と理論表面粗さと同じ結果となり，円弧対応によって，精度向上を確認することができた．また，計算時間は 最大で約 9 ms から約 4 msec に減少し，55 % 計算時間の減少に成功した．円弧対応のために 2 つの処理を追加したが，計算時間に大きく影響せず，クリップポリゴンの頂点数が 10 点から 4 点になった事による計算時間の短縮の方が大きいと考えられる．

表 3.1　円弧の交差による交点の Form と中心座標値の決定

（a）　差

Subject: Line　Clip: Arc

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
Form	Arc	Line	Line	Arc
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
Form	Line	Arc	Arc	Line

Subject: Arc　Clip: Line

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
Form	Line	Arc	Arc	Line
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
Form	Arc	Line	Line	Arc

Subject: Arc　Clip: Arc

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
中心座標値	Clip	Subject	Subject	Clip
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
中心座標値	Subject	Clip	Clip	Subject

（b）　積

Subject: Line　Clip: Arc

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
Form	Line	Arc	Arc	Line
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
Form	Arc	Line	Line	Arc

Subject: Arc　Clip: Line

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
Form	Arc	Line	Line	Arc
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
Form	Line	Arc	Arc	Line

Subject: Arc　Clip: Arc

交差タイプ	LS-LC	RS-LC	LS-RC	RS-RC
中心座標値	Subject	Clip	Clip	Subject
交差タイプ	LC-LS	LC-RS	RC-LS	RC-RS
中心座標値	Clip	Subject	Subject	Clip

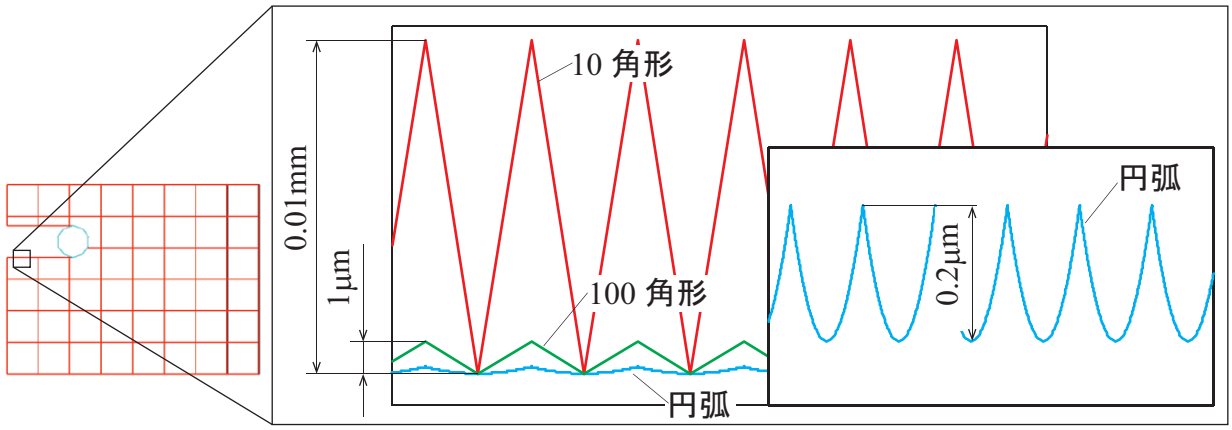


図 3.10　円弧対応前後によるシミュレーション後の形状の比較

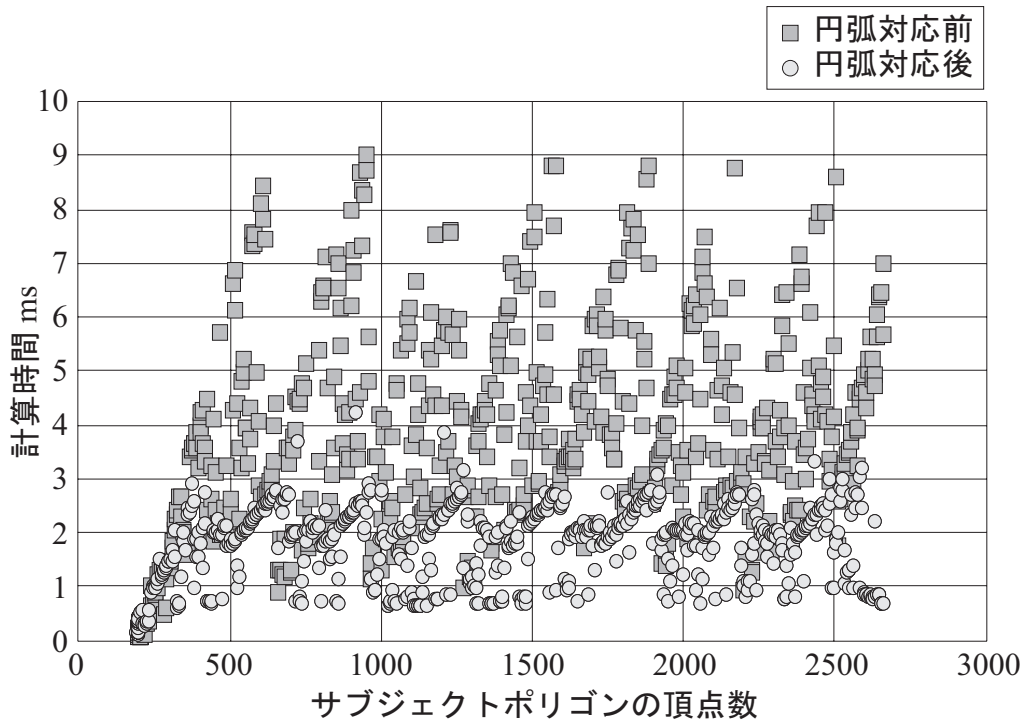


図 3.11　円弧対応前後による計算時間の比較

3.4　3 次元化

最後の問題点である 2 次元を対象とした Vatti クリッピングを 3 次元切削シミュレータへ応用する方法を考案した．本研究では 2 次元ポリゴンを積層することによって，3 次元化を実現した．図 3.12 に示すような 3 次元モデル形状を表現するとき，z 方向に厚みを持った複数枚のレイヤに保存されたポリゴンを積層することによって，3 次元化する．ブール演算を行う際には，工具位置からブール演算に関係のあるレイヤを取り出し，レイヤごとにブール演算を行う．そして，3 次元モデルに再構築を行う．

しかし，大量のレイヤが選択された場合何度もブール演算を行わなければならず，計算時間が非常に増大してしまう問題点が存在する．この問題点はこの積層による 3 次元化を行っている限り発生する問題であるが，各レイヤにインデックスを用意する事によって，ブール演算を行うレイヤを的確に選択するし，計算時間の削減を行った．

このインデックスはブール演算を行うたびに更新し，同じインデックスのレイヤがブール演算される場合，1 つのレイヤのみブール演算を実施し，その結果を他のレイヤに反映する．よって，同じサブジェクトポリゴンでのブール演算を何度も行う必要がなくなり，1 度のブール演算によって，多数のレイヤに計算結果を反映できる．しかし，切削シミュレーションが進み，レイヤ同士のインデックスが異なっても同じポリゴンを使用している可能性があり，このとき，インデックスを同じにする必要がある．この同じポリゴンかどうかを判断する方法として，以下に示す方法を使用した．また，この 3 次元化に関するフローチャートを図 3.13 に示す．

(1) レイヤごとのポリゴン同士の頂点数を確認する
(2) (1) で頂点数が同じ場合，ポリゴンを構成する頂点の積集合を確認する
(3) 積集合とポリゴンの頂点数が一致しているか確認する

(1) から (3) の計算を行い，積集合とポリゴンの頂点数が一致している場合，インデックスを同じにする．

この方法で，3 次元切削シミュレータを開発した．この切削シミュレータの性能を確認

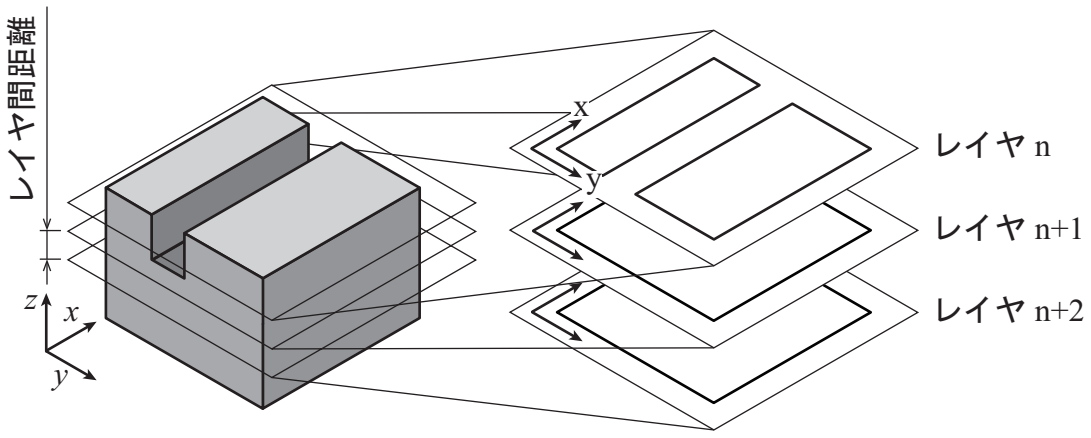


図 3.12　レイヤ積層による 3 次元化

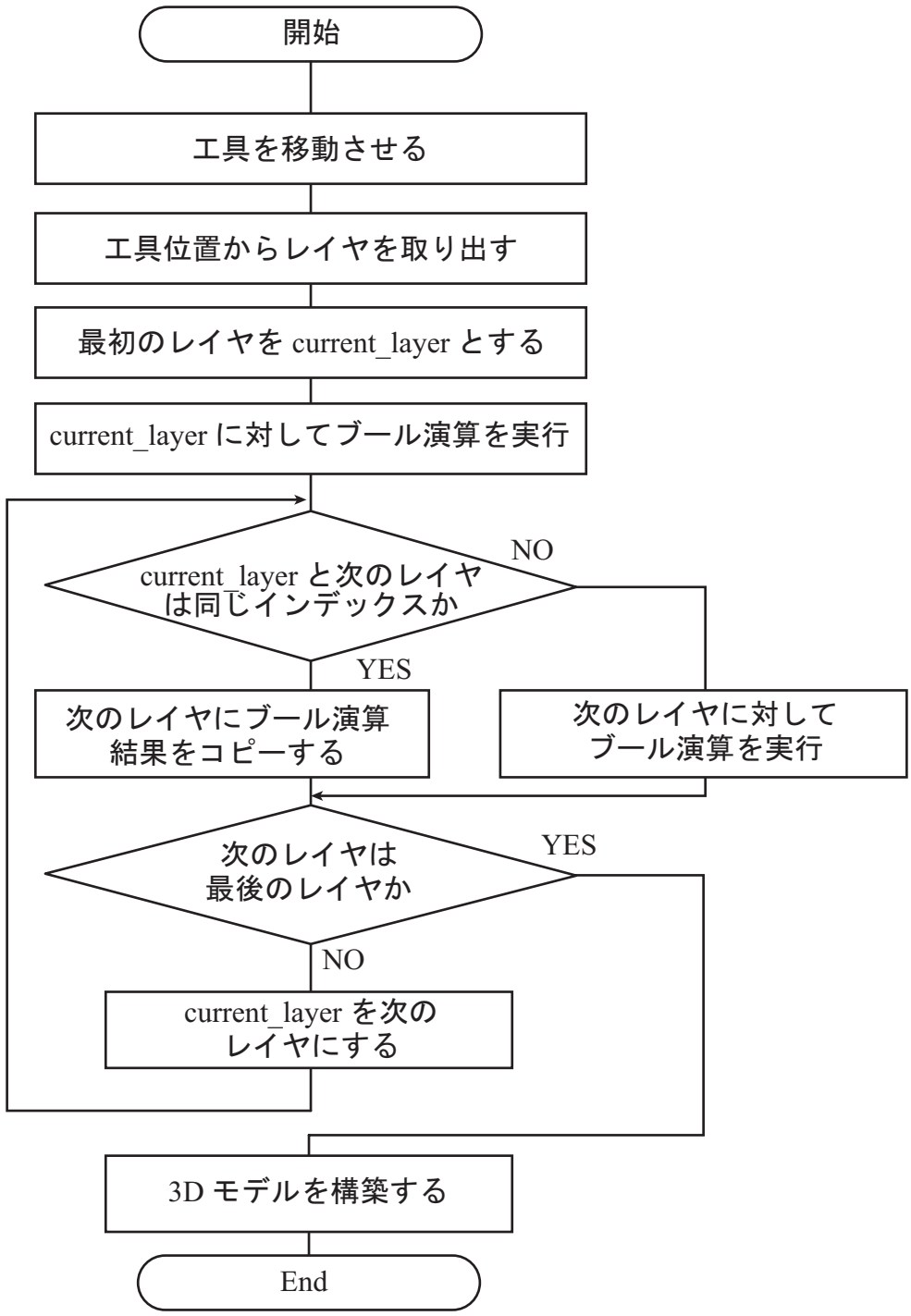


図 3.13　3D モデル構築のフローチャート

するために，図 3.14 に示すようなエンドミルを上下に移動させながら，y 方向に移動させシミュレーションを行った．エンドミルの動作などの条件は表 3.2 に示す．この 1 ステップとは，工具を微小移動させ，ブール演算を行う処理を 1 ステップと呼ぶ．この 1 ステップの処理を何度も行うことによってシミュレーションを行う．今回対象としたシミュレーションは実際には存在しないような加工であるが，このシミュレーションでは，各レイヤが異なるポリゴンで，レイヤのインデックスが異なるレイヤを大量にピックアップすることができ，計算負荷が大きいと考えられる．図 3.15（a）に実行結果を示し，（b）にブール演算に掛かった計算時間を示す．また，今回はレイヤごとにコアを割り当て，マルチコアを使用して演算を行った．

計算時間結果を見てみると，工具の移動と同じ様に，計算時間が山形状になっていることがわかる．これは，z 座標値が大きいときは，対象となるレイヤが少なく計算時間が短くなっているが，工具の z 座標値が小さくなったときには多くのレイヤが計算の対象となり，計算時間が多く掛かっているが，最大でも約 18 ms でブール演算の計算を行うことができた．一方シミュレーション後の工作物形状を見てみると，レイヤを積層することによって，3 次元化を行っているため，レイヤ間距離が z 方向の精度になる．よって，本来ならなめらかな面になる部分が図 3.15(a)のように，階段状になっていることがわかる．これは，レイヤ間距離を細かくすることによって，改善することができるが，レイヤ間距離を細かくすることは，レイヤ数が多くなるため，計算時間は増加すると考えられる．

次に切削体積が精度良く計算できているかを確認した．図 3.16 に示すように立方体形

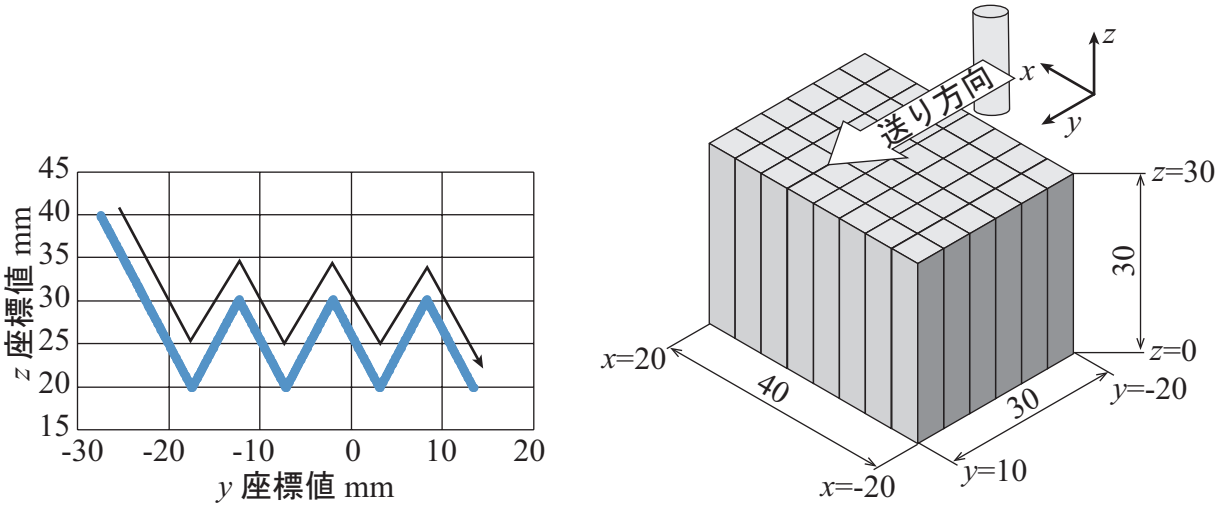
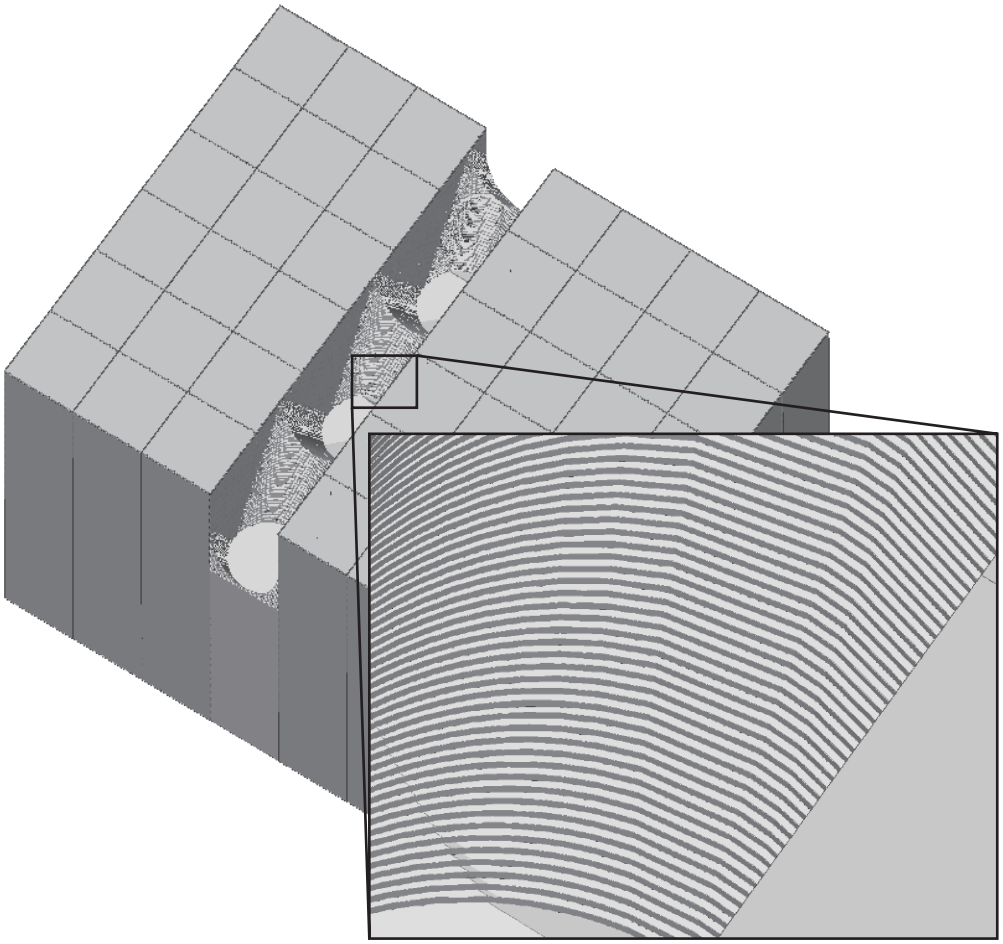


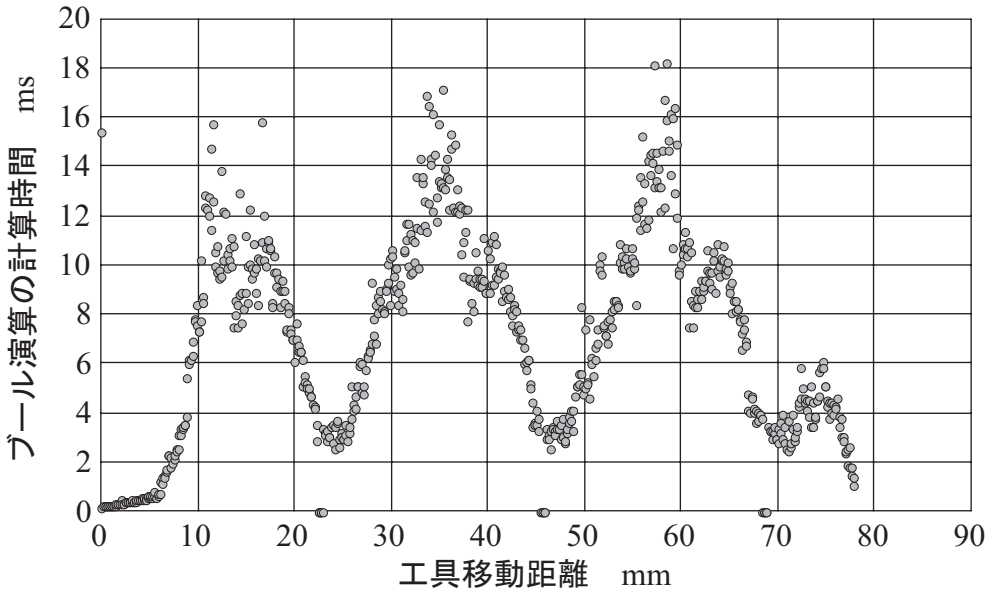
図 3.14　工具の移動方向

表 3.2　シミュレーションの条件

エンドミル	円柱形状 $\phi 5$
1 ステップの y 方向移動量	0.05 mm
1 ステップの z 方向移動量	0.1 mm
レイヤ間距離	0.1 mm



(a)　外観図



(b)　計算時間測定結果

図 3.15　シミュレーション結果

状の工作物に直径 2 mm の円柱形状の工具を z 方向に 1 mm 切り込ませ, ブール演算を行い, 切削体積を計算した．理論上この切削体積は π mm^3（3.14159265358979... mm^3）になるが, 切削シミュレータを用いて計算した結果は 3.14159265358983... mm^3 となり，小数点第 12 位まで理論値と一致した．よって，非常に高い精度で切削体積を計算できていることがわかる．

3.5 まとめ

本章では，Vatti クリッピングを切削シミュレータに応用する上での 3 つの問題点の解決方法を提案し，この解決方法の有効性を検証し以下の結論を得た．

（1）サブジェクトポリゴンを格子状に分割することによって計算時間が一定以上増加しないことを確認した．
（2）Vatti クリッピングを円弧に対応させ，計算時間の減少に成功した．
（3）レイヤを積層させることによって，3 次元切削シミュレータを開発した．

参考文献

[1] Fletcher Dunn，Ian Parberry,（松田晃一 訳）: 実例で学ぶゲーム 3D 数学，オーム社, 2011

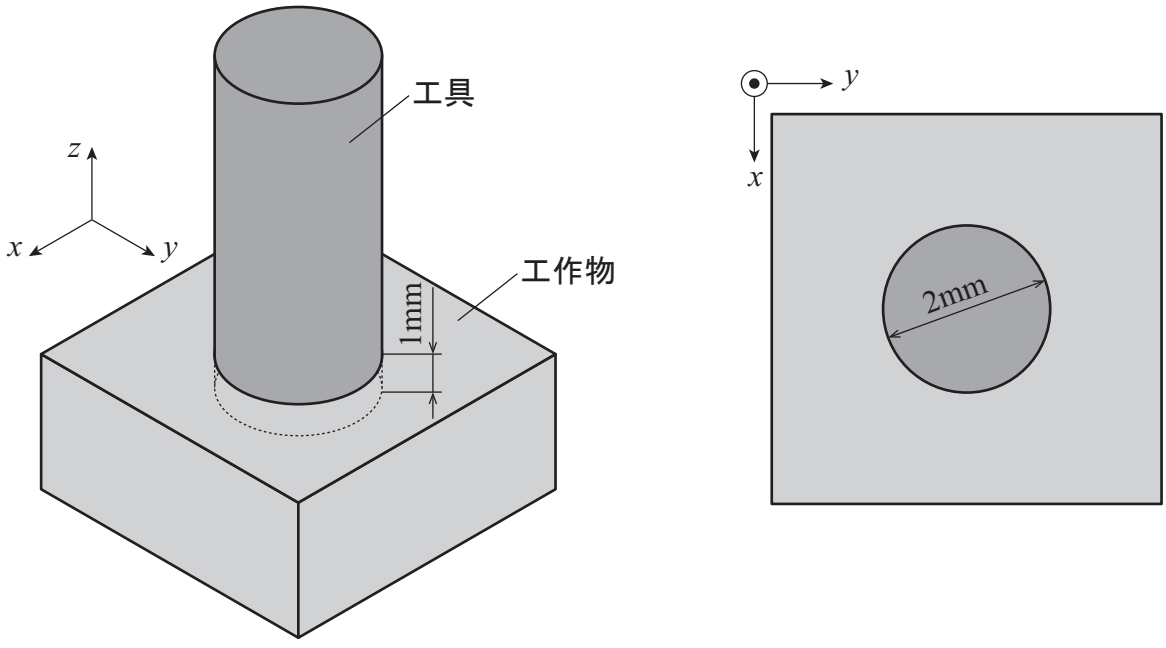


図 3.16　切削体積計算方法

第 4 章　切削シミュレータの高速性の検証

4.1　はじめに

2 章で述べた 3 つの問題点の解決方法を 3 章で説明し，切削シミュレータを開発した．この開発した切削シミュレータの高速性を検証するために，2 種類の工具経路を使用して，シミュレーションを行った．開発した切削シミュレータは 3 章でレイヤを z 方向に積層することによって，3 次元化を実現しているため，z 方向の移動を行う工具経路では，z 方向の移動を行わない工具経路より計算時間が多く掛かると考えられる．そこで，z 方向の移動がない工具経路と z 方向の移動がある工具経路の 2 種類で開発した切削シミュレータの高速性の検証を行う．

4.2　ポケット加工

最初のケーススタディとしてポケット加工のシミュレーションを行った. 図 4.1 にポケット加工の動作を示す．この工具経路は z 方向の移動で工作物に進入し．その後 x，y 方向の移動により，ポケット加工を行う．よって，この加工では，z 方向を移動しながら x または y 方向に移動しないため, レイヤのインデックスが同じになり, 計算量が小さいシミュレーションになると考えられる．そこで，計算量の小さい工具経路でのシミュレーション時間の検証と切削体積がどのように変化していくかを検証する．

シミュレーションの条件を表 4.1 に示す．x，y 方向の移動は 1 ステップ当たり 0.05 mm，レイヤ間距離を 0.1 mm としたため，z 方向は同じ 0.1 mm とした．また，計算はシングルコアで計算を行った．

シミュレーションを行った計算時間と切削体積の計算結果を図 4.2 に示す．計算時間を見てみると，1 ステップの計算時間は最大でも約 5 ms で非常に高速である．また，切削体積を見てみると，ポケットのコーナー部分に近づくと，切削体積が増加していく様子がわかり，一般的に知られているようなポケット加工において，コーナー部分で切削体積が増加することがシミュレートできていると言える．

実際の加工では 1 回転あたりの送り量と主軸回転数によって計算時間は変化する．一方シミュレーションでは主軸 1 回転ごとに演算を行っている場合，1 ステップの工具移動量によってシミュレーション時間が変化する．言わば，1 ステップの工具移動量は工具 1 回転あたりの送り量と言える．この 1 ステップの工具移動量を変化させ，シミュレーションし, その総計算時間を測定した結果と実際の回転数と 1 回転あたりの送り量の関係を図 4.3 に示す. どの主軸回転数, 送り速度であっても予想加工時間より短い時間でシミュレーションが行えていることがわかる．なお，今回図 4.3 にプロットした加工は最大で主軸回転数 5 万回転で送り速度が 1.0 mm/rev と実際の加工にはありえない加工条件となっており，開発した切削シミュレータは一般的な実加工より短い時間でシミュレーションを行うことができた.

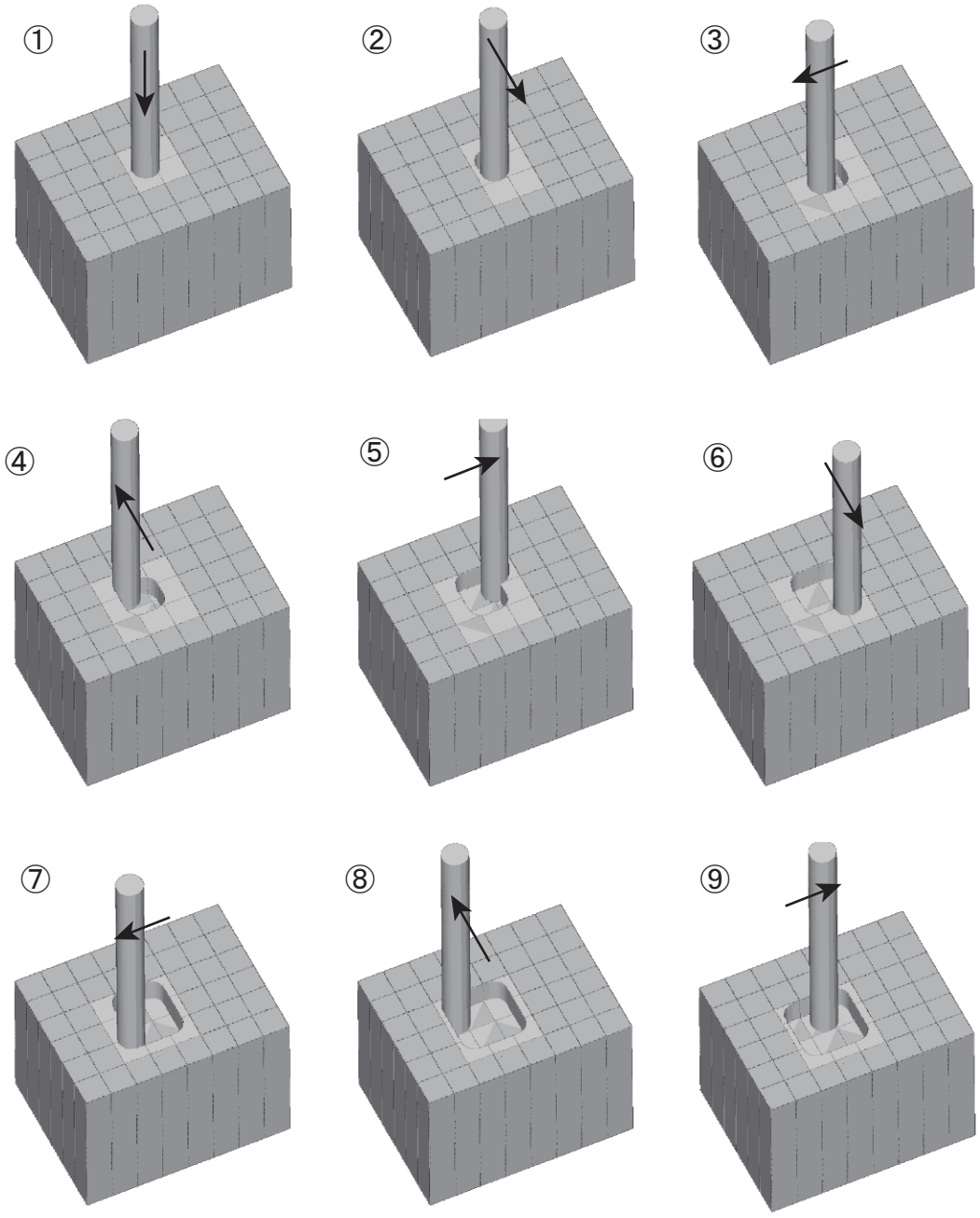


図 4.1　ポケット加工の動作

表 4.1　ポケット加工のシミュレーション条件

エンドミル	円柱形状 $\phi 5$
1 ステップの x, y 方向移動量	0.05 mm
1 ステップの z 方向移動量	0.1 mm
レイヤ間距離	0.1 mm

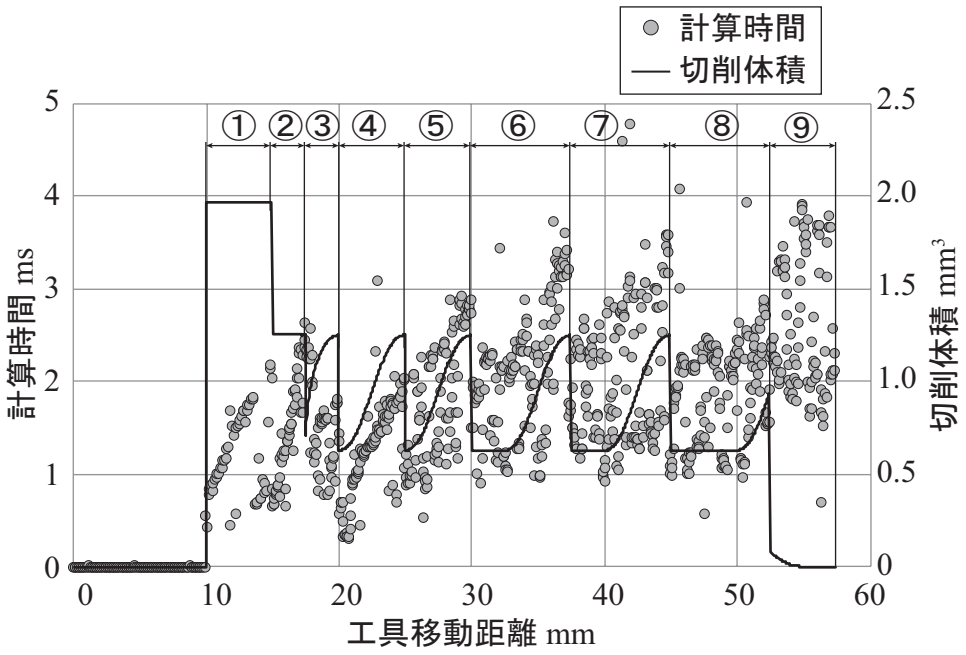


図 4.2　ポケット加工における計算時間と切削体積の測定結果

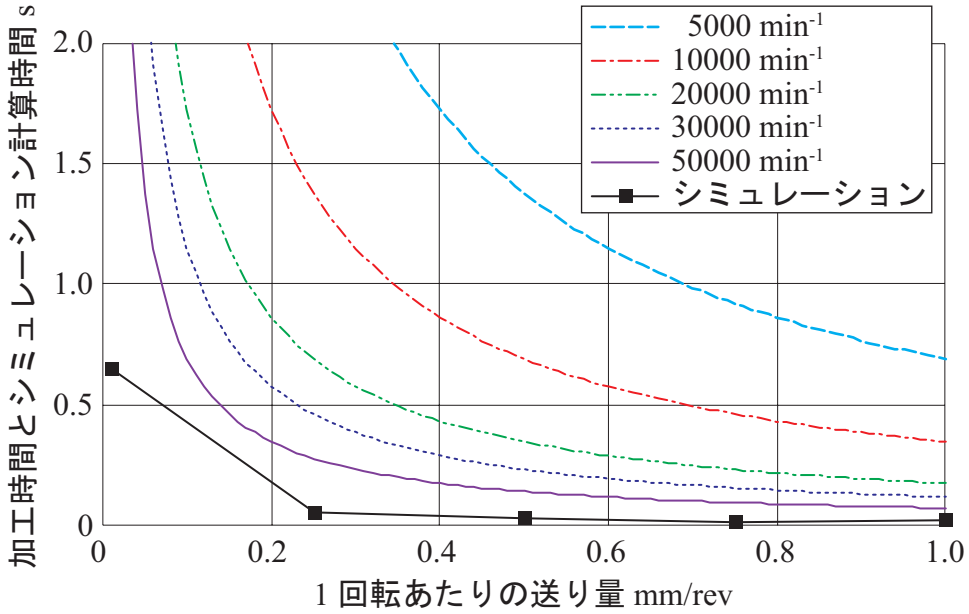


図 4.3　シミュレーションの計算時間と実加工時間の関係

4.3　スパイラル加工

次のケーススタディとして，スパイラル加工のシミュレーションを行った．図 4.4 にスパイラル加工の動作を示す．このように，工具を x，y 方向に移動しながら，z 方向にも移動するため，レイヤごとに異なるポリゴンになり，それぞれのレイヤごとにブール演算を実施する必要があり，計算量が多いシミュレーションになると考えられる．よって，開発した切削シミュレータが不向きな加工での高速性の評価を行う．

図 4.5 にシミュレーション結果を示す．このときのレイヤ間距離は 0.02 mm であり，1 ステップのエンドミル移動量は約 0.05 mm とし，z 方向の移動量はレイヤ間距離と同じ 0.02 mm とした．また，このスパイラル加工は約 1.5 周しながら，z 方向に 10 mm 加工し，その後 z 方向に移動せずに x，y 方向のみで 1 周加工を行う．

ブール演算に掛かった計算時間と切削体積の計算結果を図 4.6 に示す．また，今回マルチスレッドを使用し，計算を行った．計算時間は 1 ステップ当たり最大で約 60 ms となり，ポケット加工と比べると 12 倍計算時間が掛かったが，レイヤ数が最大で 500 枚になるシミュレーションに対して，最大 60 ms であるため，非常に高速だと考えられる．また，ポケット加工と同様にさまざまな回転数と送り速度で加工時間を計算，シミュレーション時間を測定した結果を図 4.7 に示す．ポケット加工では主軸回転数が 50000 $\mathrm{min^{-1}}$ の加工より短い時間でシミュレーションすることができていたが，スパイラル加工では，1 回転あたりの送り量が小さいときは，5000 $\mathrm{min^{-1}}$ の加工と同等の計算時間となり，送り量が大きいときは 20000 $\mathrm{min^{-1}}$ の加工と計算時間が同等となった．よって，対象としたスパイラル加工のような大量のレイヤに対してブール演算を行う必要のある z 方向に移動する加工は，ポケット加工のような z 方向に移動しない加工より計算時間時間が多くかかり，高速性が低下するという結果となったが，ポケット加工でシミュレート可能だった条件に比べ，一般的な切削条件で，加工時間と同等の時間でシミュレーションを行うことができた．

このケーススタディでは，大量のレイヤに対してそれぞれブール演算を行いながら，高速にシミュレーションが可能であることがわかった．今回計算コストが増加することから，工具をスクエアエンドミルとし，工具にレイヤを用意しないことによって，計算コストの小さいシミュレーションを対象としていた．しかし，このケーススタディの結果から工具にもレイヤを用意し，ボールエンドミルのような工具の径が工具軸方向で変化する工具を対象としても，高速にシミュレーション可能であると考えられる．

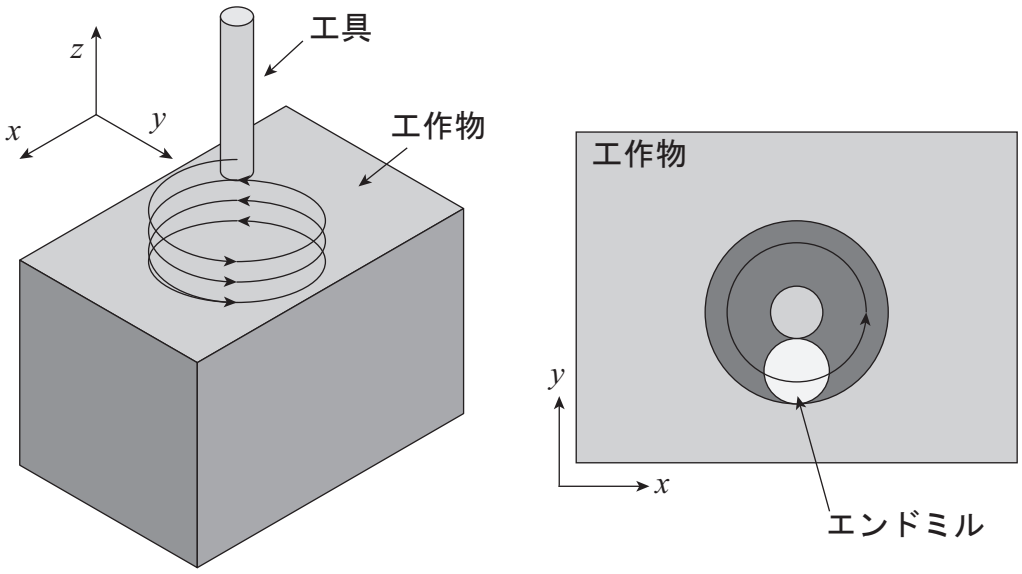


図 4.4　スパイラル加工の動作

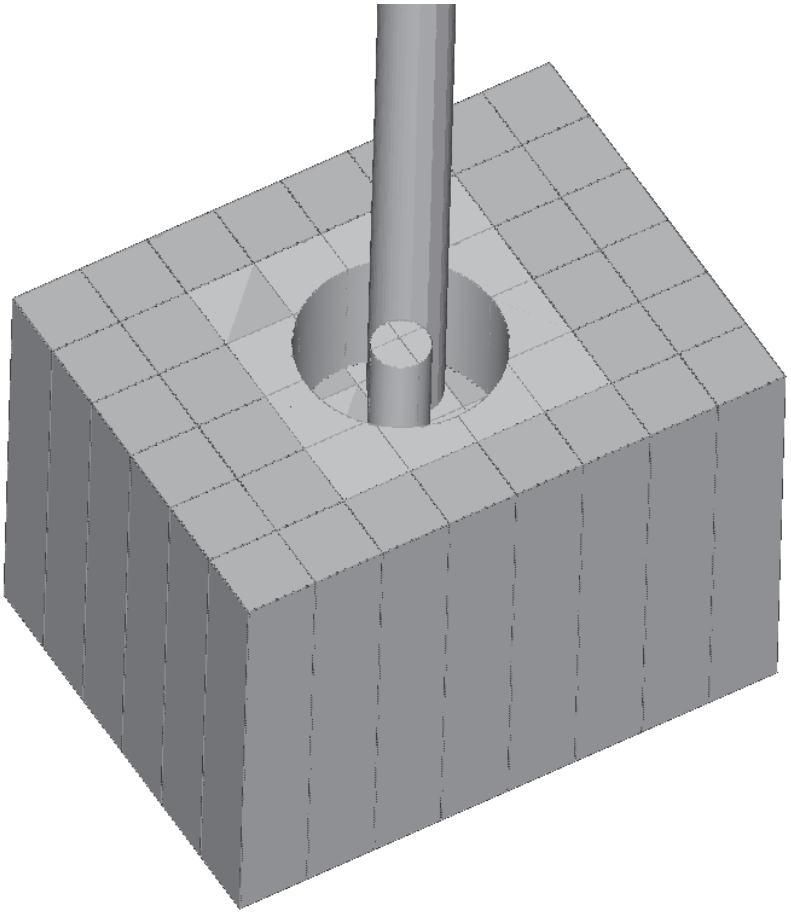


図 4.5　スパイラル加工のシミュレーション結果

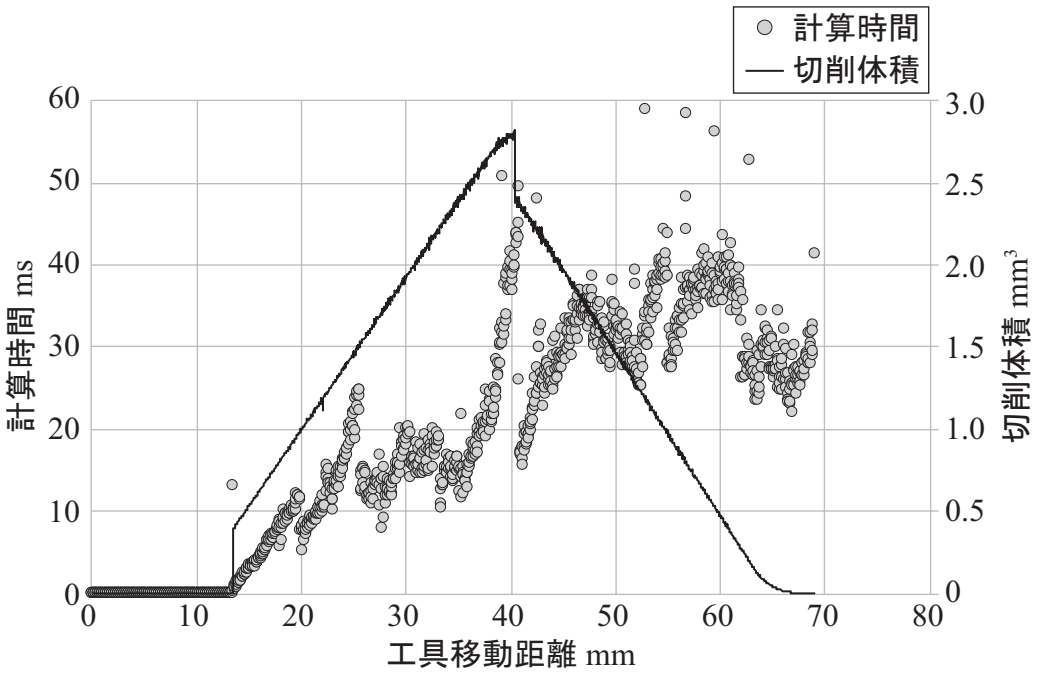


図 4.6　スパイラル加工における計算時間と切削体積の測定結果

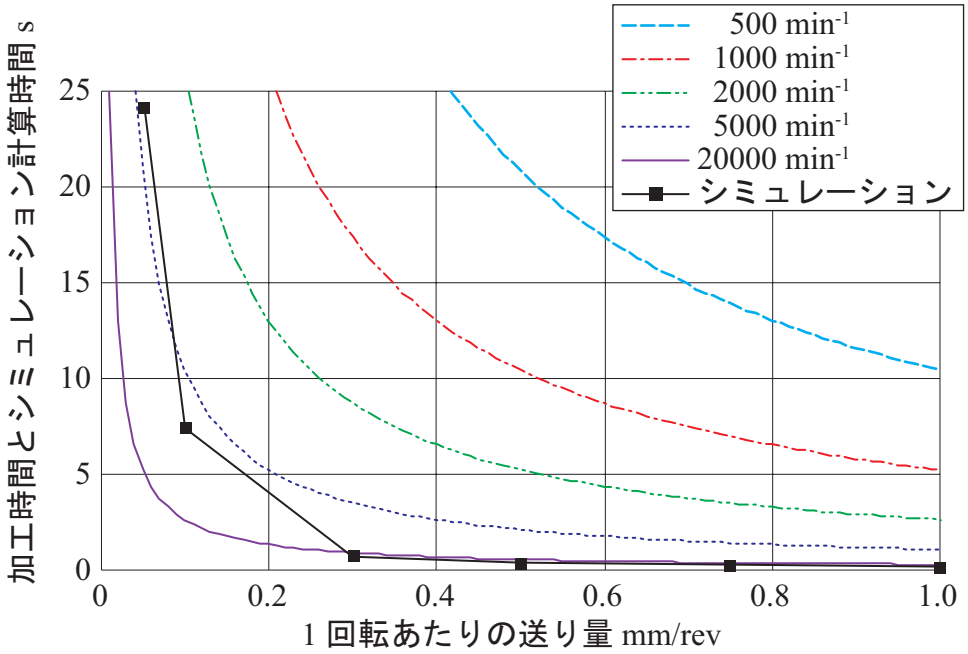


図 4.7　スパイラル加工におけるシミュレーションの計算時間と実加工時間の関係

4.4 まとめ

本章では 3 章で開発した切削シミュレータの高速性をポケット加工とスパイラル加工の 2 種類で検証し，以下の結論を得た．

(1) ポケット加工のシミュレーションでは予想加工時間より短い時間でシミュレーションを行うことができた．また，ポケットの隅部で切削体積が増加することを確認した．
(2) 計算負荷が高いと考えられるスパイラル加工のシミュレーションを行った結果，大量のレイヤに対して処理を行うシミュレーションでも一般的な加工条件と同等の時間でシミュレーション可能であることを確認した．
(3) 工具にもレイヤを用意することによって，ボールエンドミルなどのさまざまな形状の工具に対応させる必要がある．

第 5 章　切削シミュレータの応用例

5.1　はじめに

4 章で開発した切削シミュレータの高速性の検証を行い，高い高速性を示すことができた．そこで，この高速性を生かした応用例を考えた．

工作機械における切削加工中に発生する問題の 1 つにびびり振動が挙げられる．このびびり振動は図 5.1 のように面粗さや加工精度，工具寿命に悪影響を及ぼすため，その抑制は非常に重要な課題となっており，さまざまな研究が行われている．例えば，発生メカニズムの研究例として，赤澤らは低剛性工作物をボールエンドミルで加工する場合における工具傾斜を考慮した数値解析モデルを構築している [1]．また，廣垣らはエンドミル加工での側面切削時のびびり振動に関して，加工面に生じるびびり模様からその振動周波数の推定手法を開発している [2]．他にも，びびり振動抑制方法として，内海らは，薄板加工において動吸振器を設置することでびびり振動の抑制を目的とし，動吸振器の設計，構造に関して検討している [3]．社本らや Kojima らは不等ピッチ，不等リードエンドミルによりびびり振動の成長を阻害する方法 [4][5] を提案している．小池らはびびり振動周波数から最もびびり振動に対して安定な主軸回転数を算出する手法 [6] を提案している．このように，びびり振動の発生メカニズムの解析や工具，工作物に対して対策を行いびびり振動を抑制する研究などが行われている．他にもオークマでは工作機械に搭載されたセンサで振動状態を監視し，びびり振動の種類を判別し，最適な主軸回転数などの提示を行っている [7]．これらのびびり振動抑制手法はびびり振動が発生するかどうかの境界での対策であり，工具経路自体にびびり振動が発生する要因がある場合，これらの対策では対応することができない．そこで，工具経路を生成する段階，すなわち CAM 上において，びびり振動を抑制した工具経路生成を行う必要性があり，実現できれば，実機によるトライアンドエラーの削減，生産効率の向上に繋がる．しかし，工具経路生成への対策によってびびり振動を抑制した研究は非常に少ない．これは，びびり振動を抑制した工具経路生成に 2 つの問題点が存在するためだと考えられる．

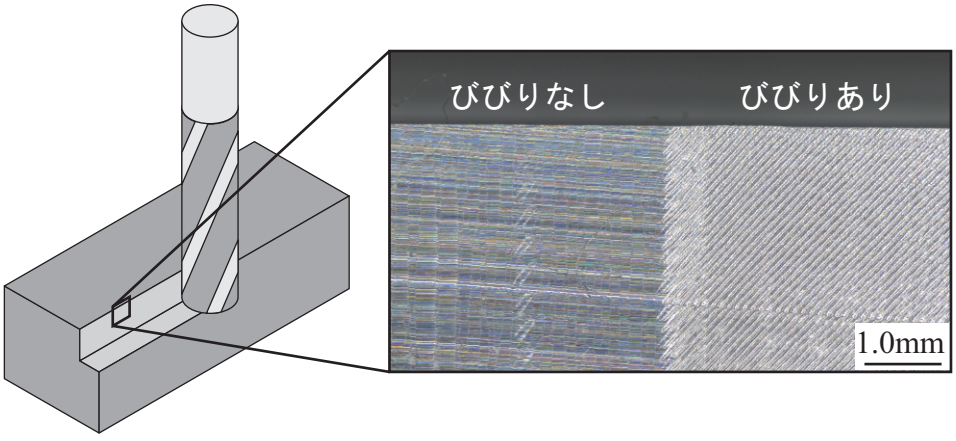
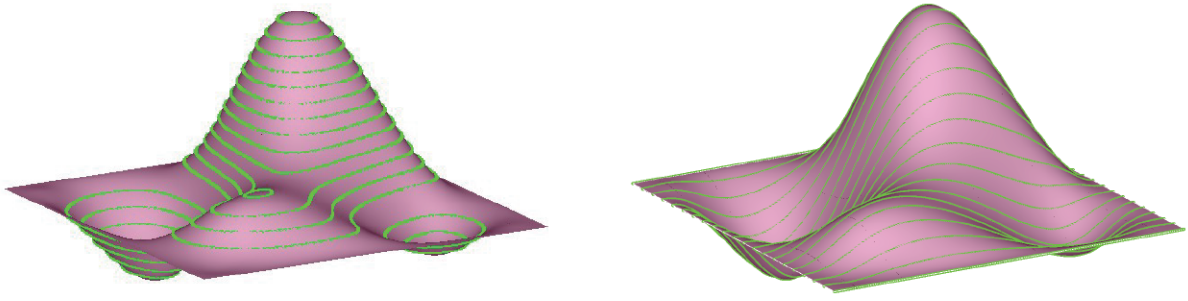


図 5.1　びびり振動による面粗さの影響

1 つ目の問題点を説明する．一般的に工具経路生成の 1 つのステップとして，CAD モデルや幾何学的ルールを用いて経路が生成されており，この際の幾何学的ルールを加工モードと称されている．例えば図 5.2 に示すような等高線や走査線のような加工モードがあり，これらの加工モードでは，ある軸方向の座標値一定にして加工点群を生成している．他にも直線や曲線，CAD モデルの情報を用いる加工モードがある．これらの加工モードでは大量の加工点を一度に生成する．よって，加工点ごとにびびり振動の有無を判断できた場合，図 5.3 に示すように，加工点の位置を移動することによって，びびり振動を回避したしても，他の加工点でのびびり振動を誘発する可能性がある．よって，何度も加工点ごとのびびり振動の判定と加工点位置を変更する必要があり，計算時間が多く掛かるのはもちろん，この処理が終わらない可能性もあり，加工点を生成した後にびびり振動の有無を判断する方法は困難であると考えられる．そこで，本研究では，加工点生成後にびびり振動の有無を判断するのではなく，びびり振動の有無を判断しながら加工点生成を行う必要があると考え，加工モードにびびり振動の有無という，従来使用されていた幾何学的ルール以外のルールを加える．

次に 2 つ目の問題点を説明する．1 つ目の問題点の説明のときに，加工点ごとにびびり振動の有無を判断すると述べたが，びびり振動の有無を判断する方法として，びびり安定限界線図 [8] が挙げられる．びびり安定限界線図の詳細は 5.2 章で説明するが, この手法は, 工作機械の動特性と切削条件などからびびりの有無を判定できる．このびびり安定限界解析には，図 5.4 に示すような工具 1 回転分の切削形状から切削開始角度，終了角度を求める必要がある．（切削開始 / 終了角度の幅を切削関与角と呼ぶ.）よって，無数にある加工点候補に対して切削形状をシミュレーションする必要性があり，計算時間が非常に多く掛かり，高速な切削シミュレータが必要になる．

そこで，本研究では，開発した切削シミュレータを用いることで，高速に切削形状をシミュレーションし，自励振動のびびり振動を対象としたびびり安定限界線図を生成，びびり振動の有無を判断しながら，加工点を生成する CAM ソフトウェアを開発した．



(a) 等高線　　(b) 走査線

図 5.2　加工モードの種類

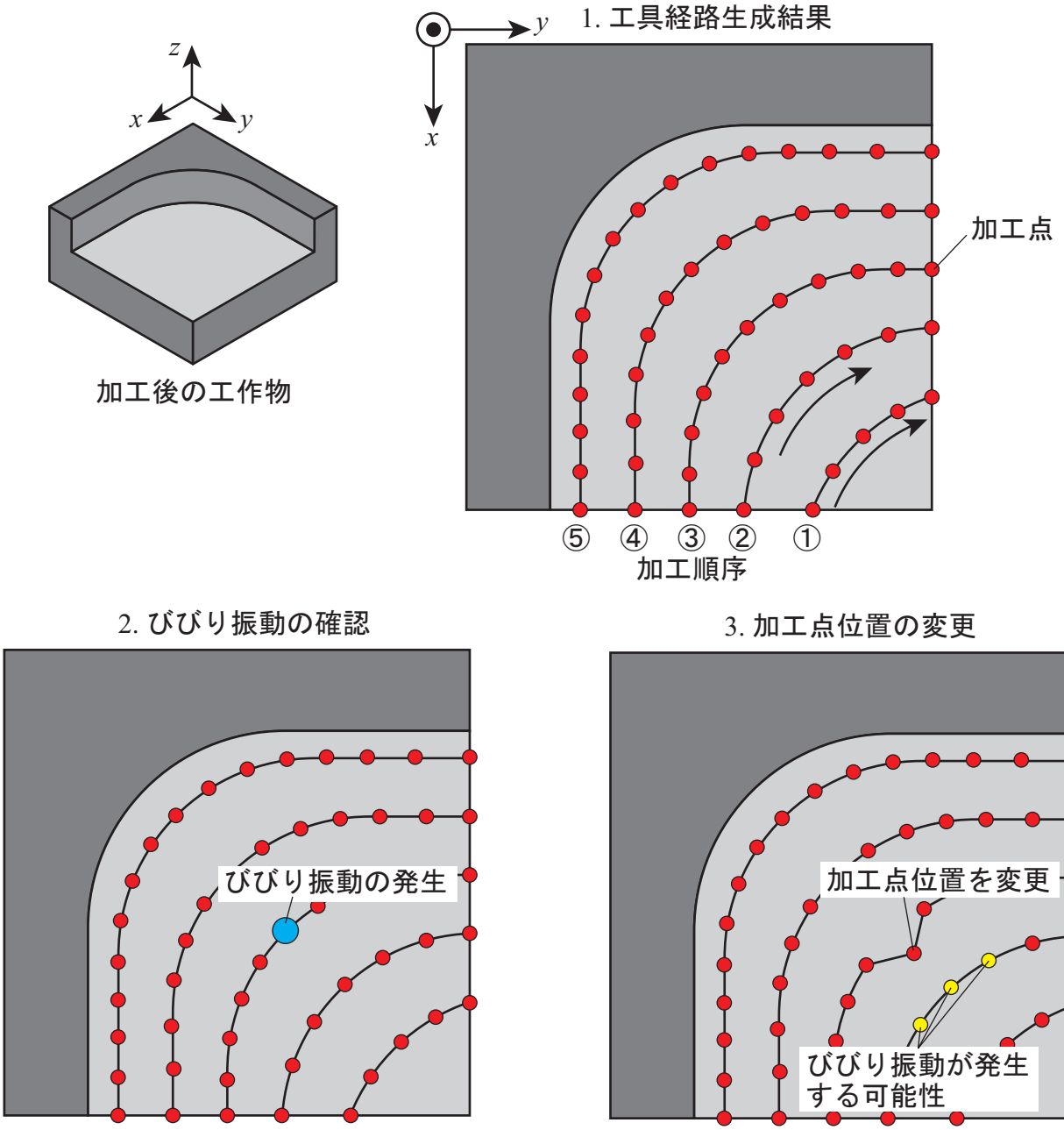


図 5.3　びびり振動の抑制工具経路生成方法の例

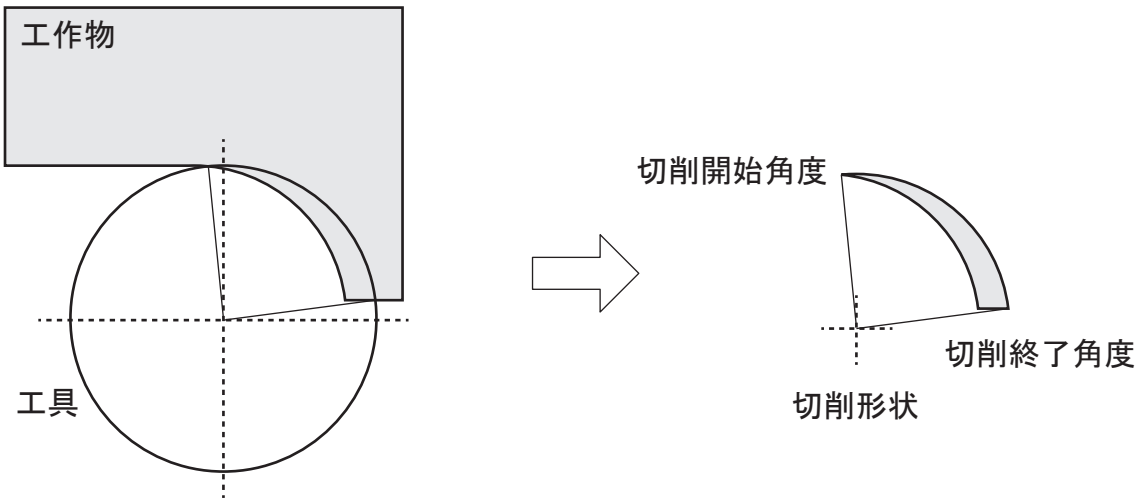


図 5.4　切削形状と切削開始 / 終了角度

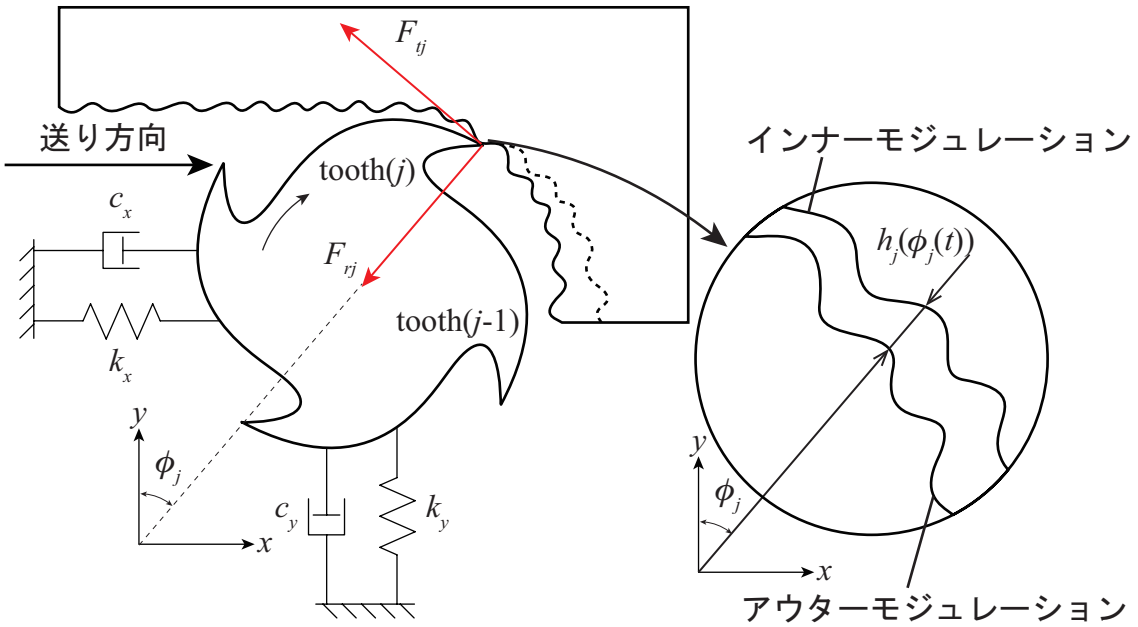


図 5.5　エンドミル加工における解析モデルの概要図

5.2　びびり安定限界線図

本研究では Altintas ら [7] によって提案された方法でびびり安定限界解析を行った．図 5.5 にエンドミル加工を工具軸方向から見た模式図を示す．びびり振動は切削時の切り取り厚さの変動が引き起こす自励振動である．エンドミル加工における切り取り厚さとは現在切削中の切れ刃 j が振動を伴って描く軌跡（インナモジュレーション）と前回切削に関与した切れ刃 j-1 による軌跡（アウタモジュレーション）の差 $h_j(t)$ といえる．この切り取り厚さ $h_j(t)$ は次式のように表せる．

$$h_j(\phi_j) = \{\Delta x \sin\phi_j + \Delta y \cos\phi_j\} \cdot g(\phi_j) \tag{1}$$

このとき，Δx と Δy は各方向の 1 刃前の振動変位と現在の振動変位との差である．また，$g(\phi_j)$ は切削中かどうかを示すステップ関数である．次に半径方向の切削反力を F_{rj}，接線方向の切削反力を F_{tj} とすれば

$$\begin{aligned} F_{tj} &= aK_t h(\phi_j) \\ F_{rj} &= K_r F_{tj} \end{aligned} \tag{2}$$

と表せる．ここで，K_t は接線方向の比切削抵抗，K_r は分力比であり，a は軸方向切り込みである．この切削反力を x，y 方向に分割すると，次式になる．

$$\begin{cases} F_{xj} = -F_{tj}\cos\phi_j - F_{rj}\sin\phi_j \\ F_{yj} = +F_{tj}\sin\phi_j - F_{rj}\cos\phi_j \end{cases} \tag{3}$$

さらに，刃数を N として，エンドミル 1 回転分の積算切削反力を求めると，

$$\begin{Bmatrix} F_x \\ F_y \end{Bmatrix} = \frac{1}{2} aK_t \begin{bmatrix} \alpha_{xx} & \alpha_{xy} \\ \alpha_{yx} & \alpha_{yy} \end{bmatrix} \begin{Bmatrix} \Delta x \\ \Delta y \end{Bmatrix} \tag{4}$$

となる．ただし，

$$\begin{cases} \alpha_{xx} = \sum_{j=0}^{N-1} -g(\phi_j)\{\sin 2\phi_j + K_r(1-\cos 2\phi_j)\} \\ \alpha_{xy} = \sum_{j=0}^{N-1} -g(\phi_j)\{(1+\cos 2\phi_j) + K_r \sin 2\phi_j\} \\ \alpha_{yx} = \sum_{j=0}^{N-1} g(\phi_j)\{(1-\cos 2\phi_j) - K_r \sin 2\phi_j\} \\ \alpha_{yy} = \sum_{j=0}^{N-1} g(\phi_j)\{\sin 2\phi_j - K_r(1-\cos 2\phi_j)\} \end{cases} \tag{5}$$

である．ここで，切削力係数行列 $[A(t)]$ を使用すると，式 (4) は，次のように表せる．

$$\{F(t)\} = \frac{1}{2} aK_t [A(t)]\{\Delta(t)\} \tag{6}$$

そして，式 (6) をフーリエ変換すると，

$$\{F(j\omega)\} = \frac{1}{2} aK_t \{[A(j\omega)]^* \{\Delta(j\omega)\}\} \tag{7}$$

となり，[$A(j\omega)$] は以下のように表せる．ここで，ω_T は 1 刃通過周波数，T は切れ刃間の周期である．

$$\begin{aligned}[A(j\omega)] &= \sum_{r=-\infty}^{\infty}[A_r]\exp(jr\omega_T t)\\ [A_r] &= \frac{1}{T}\int_0^T [A(t)]\exp(-jr\omega_T t)dt\end{aligned} \tag{8}$$

ここで，r=0 のみの近似を考えると，

$$\begin{aligned}[A_0] &= \frac{1}{T}\int_0^T [A(t)]dt\\ &= \frac{N}{2\pi}\int_{\phi_{st}}^{\phi_{ex}} [A(\phi)]d\phi = \frac{N}{2\pi}\begin{bmatrix}\alpha_{xx} & \alpha_{xy}\\ \alpha_{yx} & \alpha_{yy}\end{bmatrix}\end{aligned} \tag{9}$$

$$\begin{cases}\alpha_{xx} = \dfrac{1}{2}\left[\cos 2\phi - 2K_r\phi + K_r \sin 2\phi\right]_{\phi_{st}}^{\phi_{ex}}\\ \alpha_{xy} = \dfrac{1}{2}\left[-\sin 2\phi - 2\phi + K_r \cos 2\phi\right]_{\phi_{st}}^{\phi_{ex}}\\ \alpha_{yx} = \dfrac{1}{2}\left[-\sin 2\phi + 2\phi + K_r \cos 2\phi\right]_{\phi_{st}}^{\phi_{ex}}\\ \alpha_{yy} = \dfrac{1}{2}\left[-\cos 2\phi - 2K_r\phi + K_r \sin 2\phi\right]_{\phi_{st}}^{\phi_{ex}}\end{cases} \tag{10}$$

となる．ϕ_{st}，ϕ_{ex} は切れ刃が切削を開始，終了する角度である．

次に，式 (7) によって，発生した切削力変動が機械構造の振動に変化を与える場合に関して考えてみる．現在と 1 刃前の振動変位を式 (11)，機械構造系のコンプライアンスを式 (12) に示す．

$$Q(t) = \begin{Bmatrix}x(t)\\ y(t)\end{Bmatrix} \quad , \quad Q_0(t) = \begin{Bmatrix}x(t-T)\\ y(t-T)\end{Bmatrix} \tag{11}$$

$$[\Phi(j\omega)] = \begin{bmatrix}\Phi_{xx}(j\omega) & \Phi_{xy}(j\omega)\\ \Phi_{yx}(j\omega) & \Phi_{yy}(j\omega)\end{bmatrix} \tag{12}$$

このとき，周波数領域における振動変位は，

$$\begin{aligned}\{Q(j\omega)\} &= [\Phi(j\omega)]\{F(j\omega)\}\\ \{Q_0(j\omega)\} &= e^{-j\omega T}\{Q(j\omega)\}\end{aligned} \tag{13}$$

となり，振動変位差 Δ は次式になる．

$$\begin{aligned}\{\Delta(j\omega)\} &= \{Q(j\omega)\} - \{Q_0(j\omega)\} \\ &= (1-e^{-j\omega T})[\Phi(j\omega)]\{F(j\omega)\}\end{aligned} \tag{14}$$

式 (9) と式 (14) を式 (7) に代入し，びびり振動は固有周波数近傍の単一のびびり周波数 ω_c で成長することが知られており，$\omega = \omega_c$ とすると，

$$\{F(j\omega_c)\} = \frac{1}{2} a_{\lim} K_t (1-e^{-j\omega_c T})[A_0][\Phi(j\omega_c)]\{F(j\omega_c)\} \tag{15}$$

となる．このとき，$a_{\lim}$ は臨界切り込み深さである．この式 (15) が $F = 0$ の自明解以外の解を得るためには，

$$\det\left[[I] - \frac{1}{2} a_{\lim} K_t (1-e^{-j\omega_c T})[A_0][\Phi(j\omega_c)]\right] = 0$$

$$\det[[I] - \Lambda[A_0][\Phi(j\omega_c)]] = 0 \tag{16}$$

$$\Lambda = \Lambda_R + j\Lambda_I = -\frac{1}{2} a_{\lim} K_t (1-e^{-j\omega_c T}) \tag{17}$$

となる．さらに式 (17) はオイラーの公式より，

$$\begin{aligned}\Lambda_R &= -\frac{1}{2} a_{\lim} K_t (1-\cos\omega_c T) \\ \Lambda_I &= -\frac{1}{2} a_{\lim} K_t \sin\omega_c T\end{aligned} \tag{18}$$

となり，この式 (18) より，臨界切り込み深さ $a_{\lim}$，位相差 ε，主軸回転数 n が以下のように求められる．

$$a_{\lim} = -\frac{\Lambda_R}{K_t}\left\{1 + \left(\frac{\Lambda_I}{\Lambda_R}\right)^2\right\} \tag{19}$$

$$\varepsilon = \pi - 2\tan^{-1}\frac{\Lambda_I}{\Lambda_R} \tag{20}$$

$$n = \frac{60\omega_c}{N(\varepsilon + 2k\pi)} \tag{21}$$

この式 (19) から式 (21) を使用して，びびり安定限界線図を作成する．

開発した切削シミュレータを使用し，びびり安定限界線図を作成する方法を説明する．

図 5.6 に示すように，円で表現された工具と工作物の切削シミュレーションを行うとブール演算の積を使用して切削形状を表現できる．この切削形状の角度を計算することによって，切れ刃の切削開始，終了角度である ϕ_{st}，ϕ_{ex} を求めることができる．そして，工作機械から測定したコンプライアンスを使用して，びびり安定限界線図を作成することができ，開発した切削シミュレーションは非常に高速であるため，エンドミルの微小移動ごとにびびり安定限界線図の計算を行うことができる．

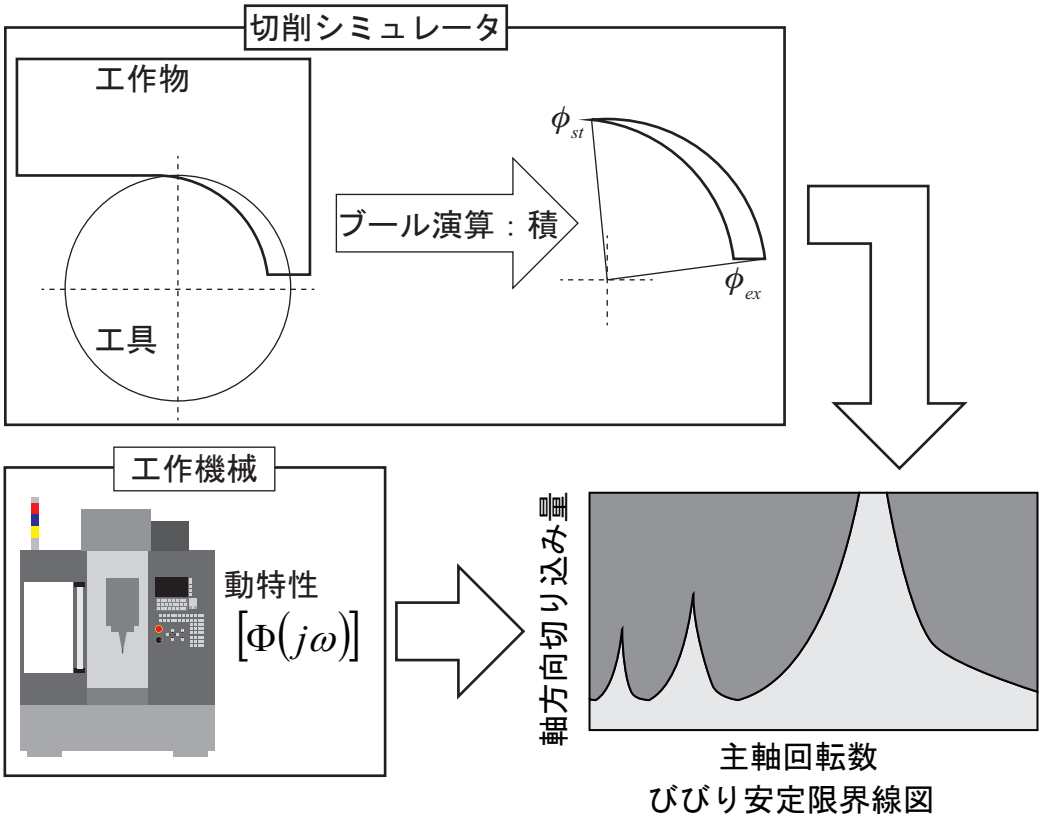


図 5.6　本研究におけるびびり安定限界線図の生成フロー

5.3　工具経路生成方法

本手法では，スクエアエンドミルを使用した軸方向切り込み量が変化しない加工を対象とし，主軸回転数，送り速度，軸方向切り込み量，工具初期位置，加工領域，工具刃先のコンプライアンスを入力パラメータとして工具経路生成を行う．前述した安定限界解析は，工具 1 回転分の積算切削力からびびり振動を解析しているため，びびり安定限界解析は 1 回転当たりの送り量ごとに行いびびりが発生しない加工点を生成することによって，工具経路全体通じてびびり振動が発生しない工具経路が得られる．

そこで本手法では，下記に示すルールで加工点を生成する．

- 現在の工具位置から 1 回転当たりの送り量だけ離れた場所にある
- ユーザーが指定する大まかな工具の移動方向内にある
- 指定した加工条件がびびり安定限界線図の安定領域内にある

このルールでは，従来の加工モードのような幾何学的な手法を用いていないことがわかる．具体的な計算方法としては，工具初期位置から 1 回転当たりの送り量だけ離れた位置に次の加工点候補を生成し，その加工点候補ごとに開発した切削シミュレータを使用して，切削関与角を計算する．この切削関与角を用いて安定限界解析を行い，指定した主軸回転数での安定限界切り込み量を計算する．そして，加工点候補の中から最適な加工点を選択し，この計算を繰り返し行うことで，工具経路生成を行う．以下に実装フロー，図 5.7 に模式

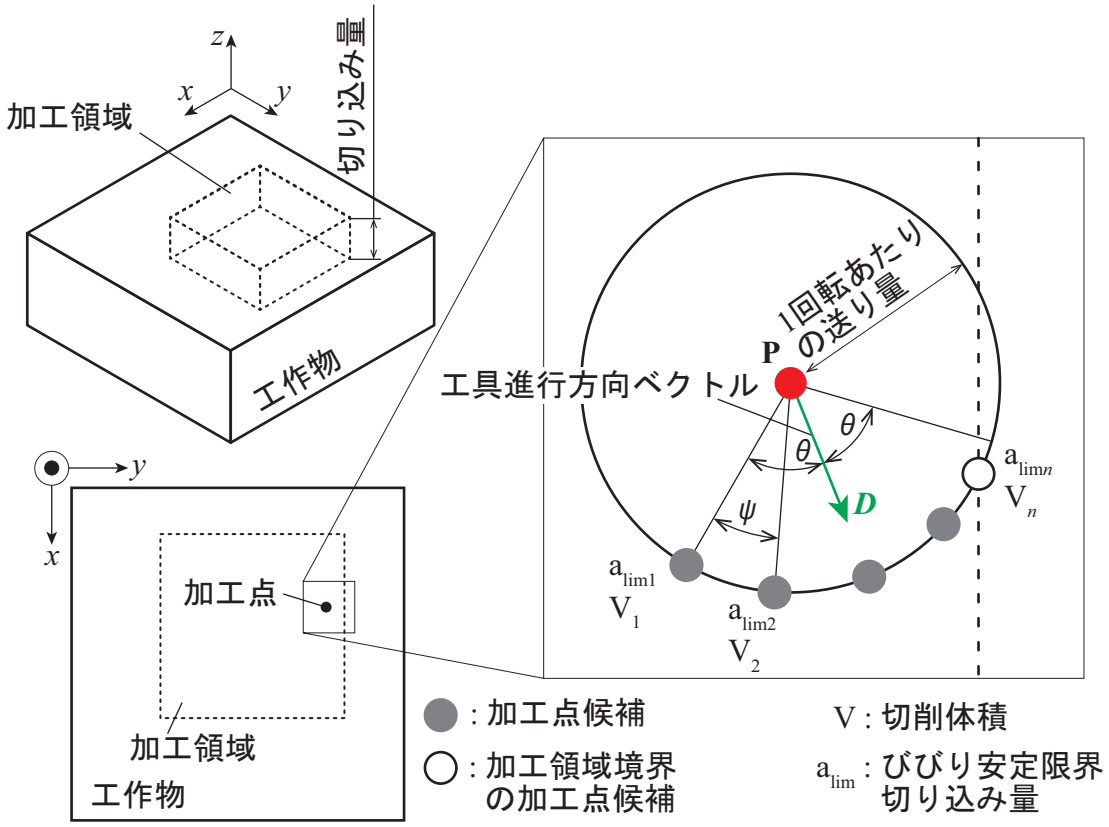


図 5.7　提案する工具経路生成方法

図を示す.

(1) 初期加工点 **P** および工具進行方向ベクトル $\boldsymbol{D}$ を定める.
(2) 1 回転の送り量を半径とする円周上かつ $\boldsymbol{D}$ の傾角 $\pm\theta$ の範囲で，任意の角度 ψ ごとに加工点候補を生成する．ただし加工領域外の点は除外し，加工領域の境界には加工点候補を生成する.
(3) 加工点候補全てに対してシミュレーションを行い，ϕ_{st}，ϕ_{ex} 及び安定限界切り込み量を計算し，安定限界内かつ最も切削体積が大きい加工点候補を次の加工点とする.
(4) (3) で求まった加工点に工具を移動させ，工作物形状をシミュレーションから求める.
(5) (1) に戻り，繰り返し加工点を決定する.

安定限界切り込み量と実際の切り込み量が同じ場合びびり振動が発生する可能性があるため，算出された安定限界切り込み量に安全率を掛けて計算する．また，加工領域は工具中心の移動範囲とする.

通常びびり安定限界線図の使用方法としては，切削関与角を用いて生成した安定限界線図を用いて，主軸回転数や軸方向切り込み量を変化させて，びびり振動を回避する．一方，本手法では主軸回転数や軸方向切り込み量を一定にし，加工点の座標を変化させることによって，切削関与角を変更，びびり安定限界線図を変化させ，その主軸回転数と軸方向切り込み量でびびり振動なく加工可能な加工点を探索する．これは高速に切削形状，切削関与角を計算できるため，使用することができる手法である．また，工具初期点と工具進行方向ベクトル及び主軸回転数や送り速度などの切削条件を与えるだけで，びびりがなくかつ切削体積が最大の加工点が選択され，最適な工具経路が自動で生成可能である．5.1 章で述べたように，従来工具経路を生成する方法としては，工作物の CAD モデルや幾何計算を用いて工具経路を生成するのが一般的であるが，本手法は，幾何計算などから加工点を生成せずに，びびり安定性を拘束条件として工具経路生成を行っている．よって，従来とは全く異なる工具経路生成方法であり，工具進行方向ベクトルを用いてある程度の工具の移動方向に制限を設けているが，基本的には工具がどのように移動するかはびびり安定性と切削体積が決定しているため，ある種のトポロジー最適化のような手法であるといえる.

5.4　工具経路生成結果

本手法を使用し，ケーススタディとして側面加工，溝加工，ポケット加工の 3 種類の工具経路生成を行った．生成条件と計算に掛かった計算時間を説明する．

5.4.1　側面加工

側面加工を対象として本手法の適用例を示す．図 5.8 に加工領域と初期加工点を示す．θ は 90 度，ψ は 10 度とした．また，工具進行方向ベクトル $\boldsymbol{D}$ は，図 5.9 に示すように，最初は y 軸方向である (0，1) とし，加工点生成を行い，$\mathbf{P}_2$ の加工点から次の y 座標値が 60 mm の加工点である $\mathbf{P}_3$ を生成した場合，次の加工点から工具進行方向ベクトルを y 軸マイナス方向である (0，-1) として加工点生成を行う．そして，$\mathbf{P}_6$ の加工点から次の y 座

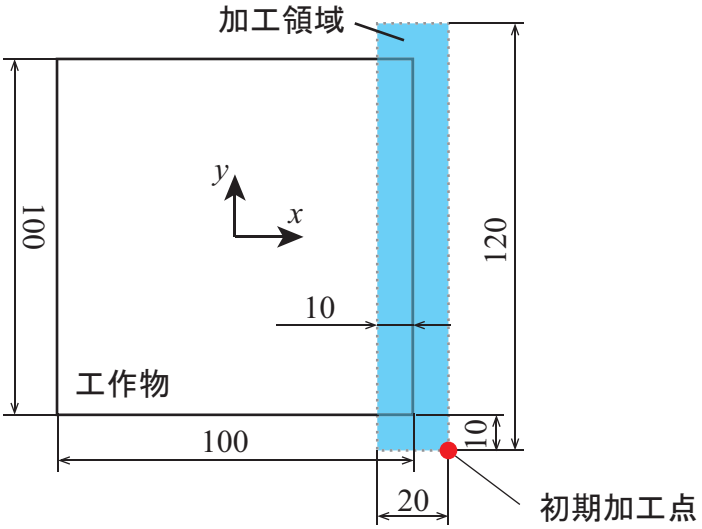


図 5.8　側面加工における加工領域と初期加工点

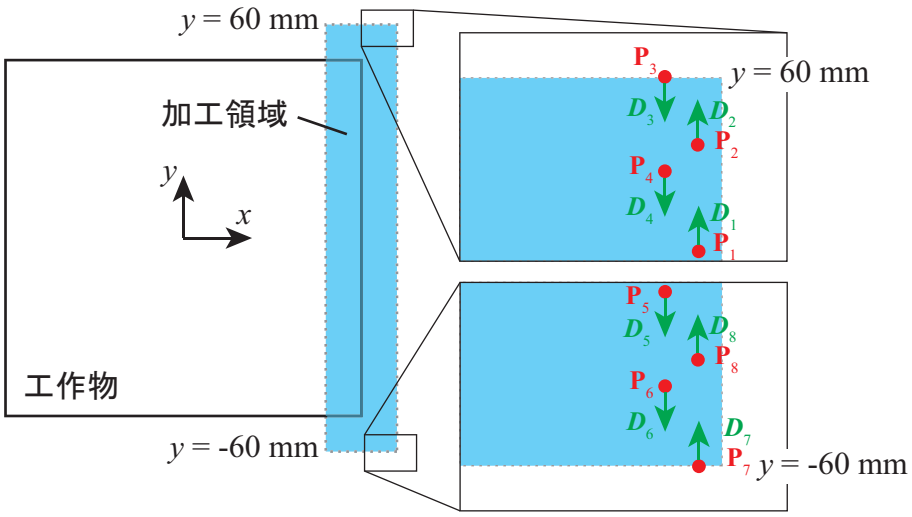


図 5.9　側面加工における工具進行方向ベクトル

表 5.1　工具経路生成条件

切削条件	主軸回転数	5500 min^{-1}
	送り速度	1650 mm/min
	軸方向切り込み量	8 mm
	分力比	0.3
	安全率	25 %
エンドミル	刃数	3
	直径	ϕ20
工作物	材質	A7075
	比切削抵抗	880 MPa

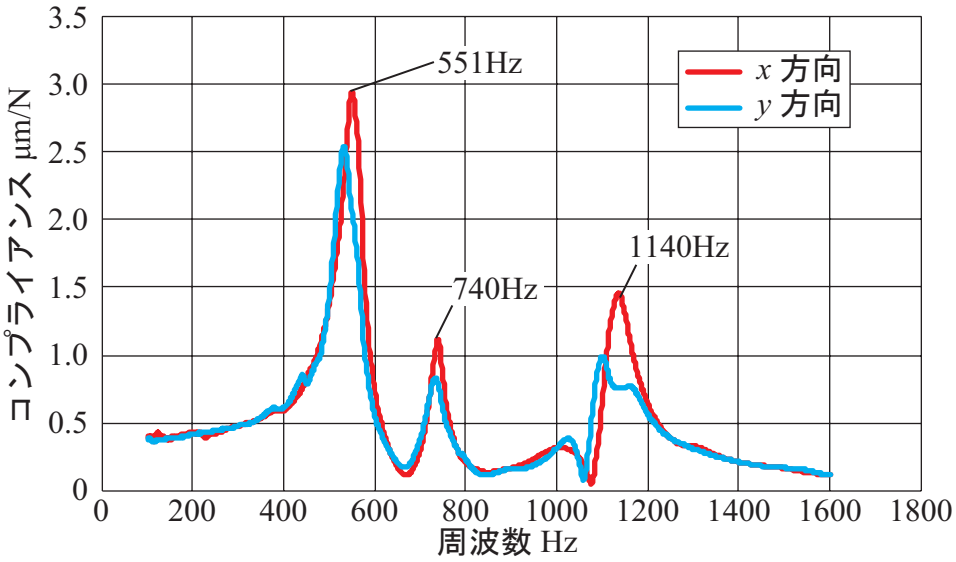


図 5.10　工具経路生成に使用した工具刃先のコンプライアンス

標値が -60 mm の $\mathbf{P}_7$ を生成した場合場合, 60 mm の時と同様に工具進行方向ベクトルを (0, 1) に変更する．また，複数の加工点候補でびびりが発生せず，切削体積が同じ場合，工具進行方向ベクトル $\boldsymbol{D}$ の方向に最も近い加工点候補を選択する．

その他のシミュレーション条件等を表 5.1 に示す．3 枚刃エンドミルでアルミ合金である A7075 の加工を対象とし，今回軸方向切り込み量は 8 mm，安全率は 25 %とした．また，ハンマリング試験によって得られた工具先端のコンプライアンスを図 5.10 に示す．このコンプライアンスを見てみると，x，y 方向によって，多少のコンプライアンスの大小はあるが，ほぼ同じ周波数でピーク値があり，値も大きく変化しないため，今回は主軸の異方性は無いものとして，x 方向のコンプライアンスのみ使用し，工具経路生成を行った．しかし，主軸の異方性があり，コンプライアンスが異なる場合は，両方向のコンプライアンスを使用することによって，異方性を考慮したびびり安定限界線図が生成され，工具経路も異方性に考慮される結果となる．また，モードカップリングの影響などもびびり安定限

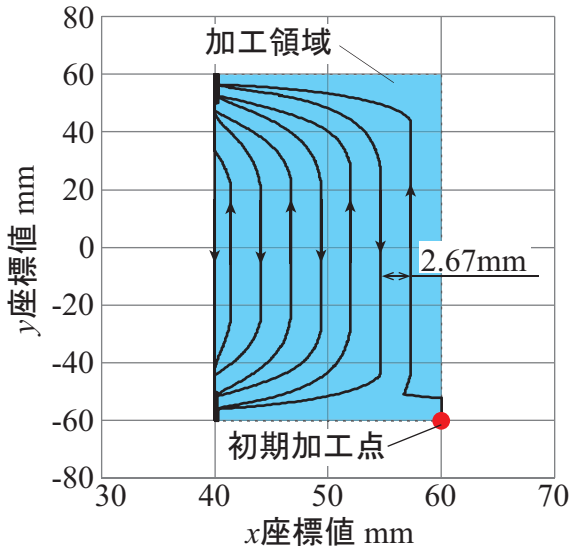


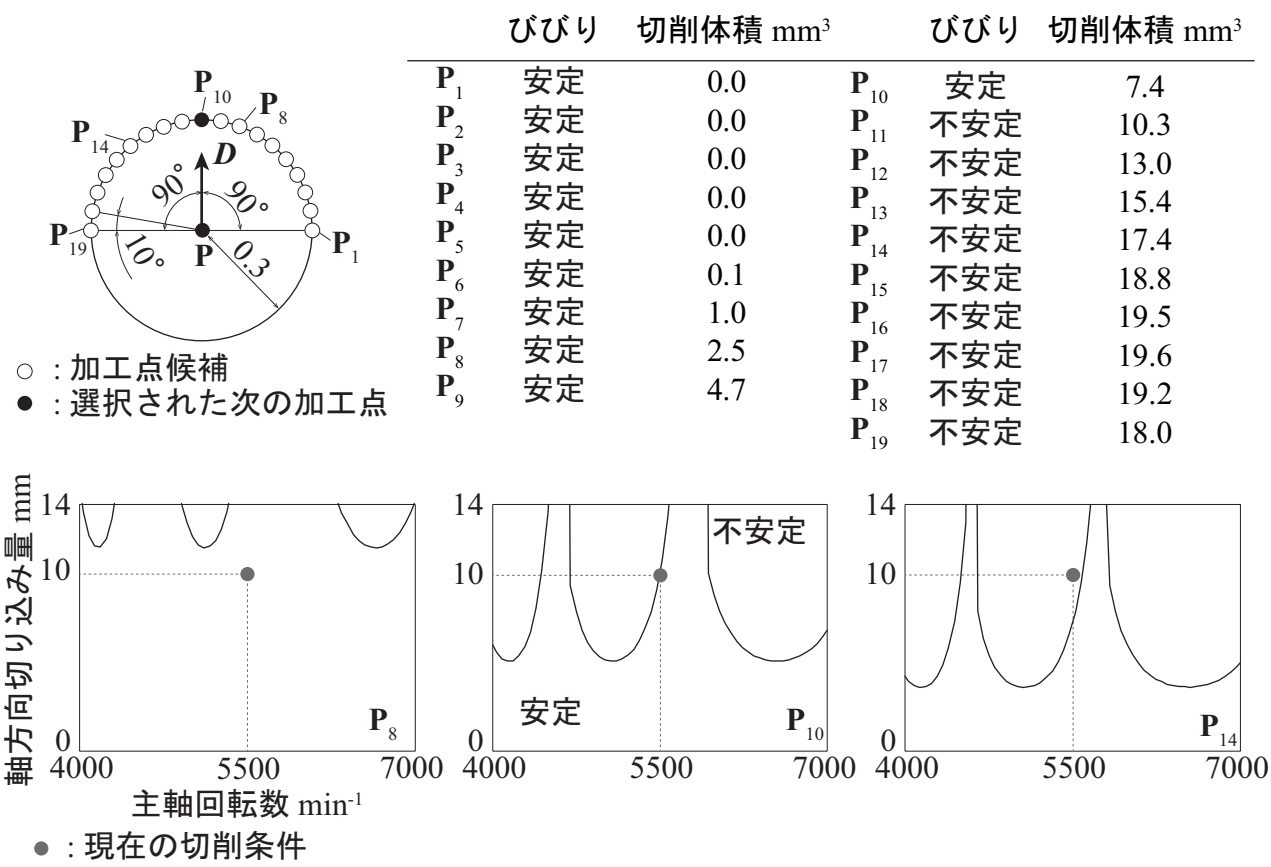
図 5.11　側面加工の工具経路生成結果

界解析に反映することで，工具経路生成にも反映できる．

工具経路生成結果を図 5.11 に示す，この工具経路の直線部分におけるある加工点生成の計算結果を図 5.12 に示す．現在工具がいる加工点 **P** から加工条件である 1 回転当たりの送り量である 0.3 mm を半径とする円周上に 19 点加工点候補を生成した．そして，各加工点候補でシミュレーションを行い，切削体積とびびり振動の有無を計算する．例えば，$\mathbf{P}_8$ は安定領域内にあるためびびりが発生しないが，切削体積が小さい．一方 $\mathbf{P}_{14}$ は不安定領域内であるため，びびりが発生してしまう．$\mathbf{P}_{10}$ は線図上にあるが安全率を掛けているため，問題なく加工ができ最も切削体積が大きい．よって，$\mathbf{P}_{10}$ を次の加工点 **P** として計算を繰り返し工具経路生成を行う．

生成された工具経路を見てみると，直線部分は径方向切り込み量 2.67 mm で加工が行われている．これは指定した軸方向切り込み量で加工ができる最大の径方向切り込み量と考えられる．そして，工作物の端である y 座標値が 50 mm，-50 mm に近づくと，びびり振動が発生しない範囲でより切削体積を得られる工作物内側へ徐々に移動している．一方，単純な直線のみで生成される工具経路では，工作物端付近に近づくと切削体積が徐々に小さくなり，最終的には加工を行わない工具経路になるが，本手法で生成した工具経路は常にびびり振動が発生しない中で，最も大きい切削体積の加工点を選択するため，加工能率が高いと考えられる．

この工具経路生成では，加工点 3281 点を生成し，切削シミュレーションを 63469 回行っており，このときの計算時間は 643.7 秒掛かった．これを加工点 1 点あたりの計算時間で表すと，約 0.2 秒で加工点を生成できており，非常に高速である．この際，加工点 3000 点の工具経路生成に対して，6 万回のシミュレーションを行っているため，高速な切削シミュレータの有効性がわかる．



	びびり	切削体積 mm³		びびり	切削体積 mm³
P_1	安定	0.0	P_{10}	安定	7.4
P_2	安定	0.0	P_{11}	不安定	10.3
P_3	安定	0.0	P_{12}	不安定	13.0
P_4	安定	0.0	P_{13}	不安定	15.4
P_5	安定	0.0	P_{14}	不安定	17.4
P_6	安定	0.1	P_{15}	不安定	18.8
P_7	安定	1.0	P_{16}	不安定	19.5
P_8	安定	2.5	P_{17}	不安定	19.6
P_9	安定	4.7	P_{18}	不安定	19.2
			P_{19}	不安定	18.0

図 5.12　最適な加工点の探索例

5.4.2　溝加工

次に溝加工での工具経路生成を行った．加工領域と初期工具位置を図 5.13 に示す．このとき θ は側面加工と同じ 90 度とし，ψ は 2.5 度とした．工具進行方向ベクトル $\boldsymbol{D}$ は最初 x 軸方向である (1，0) とし，側面加工と同じように x 座標値が 2.5 mm の加工点を生成した場合，工具進行方向ベクトルを (-1，0) に，x 座標値が -2.5 mm の加工点を生成した場合，工具進行方向ベクトルを (1，0) に変更する．その他の条件は側面加工のパス生成と同じである．

図 5.14 に示す生成されたパスは，円弧を描きながら，徐々に加工を進めていることがわかる．これは切削関与角が一番大きくなると考えられる隅部では，小さな切り込みで加工し，びびりなく加工の行える中央部分では大きく切り込み加工を行っている．よって，本手法は, 自動的に加工可能な箇所を判定し, その加工できる箇所を徐々に広げる事によって，びびりにより加工できなかった箇所を加工できるようにして，指定された加工領域の加工を行っている．

また，この工具経路生成では，加工点 1801 点を生成し，切削シミュレーションを 112775 回行っており，このときの計算時間は 821.7 秒掛かった．これを加工点 1 点あたりの計算時間で表すと，約 0.46 秒で加工点を生成できた．ψ を 2.5 度と側面加工のときより小さくしたため，加工点候補が多く生成され，シミュレーションの回数などが増加し，側面加工に比べ加工点数が少ないが計算時間が多く掛かった．

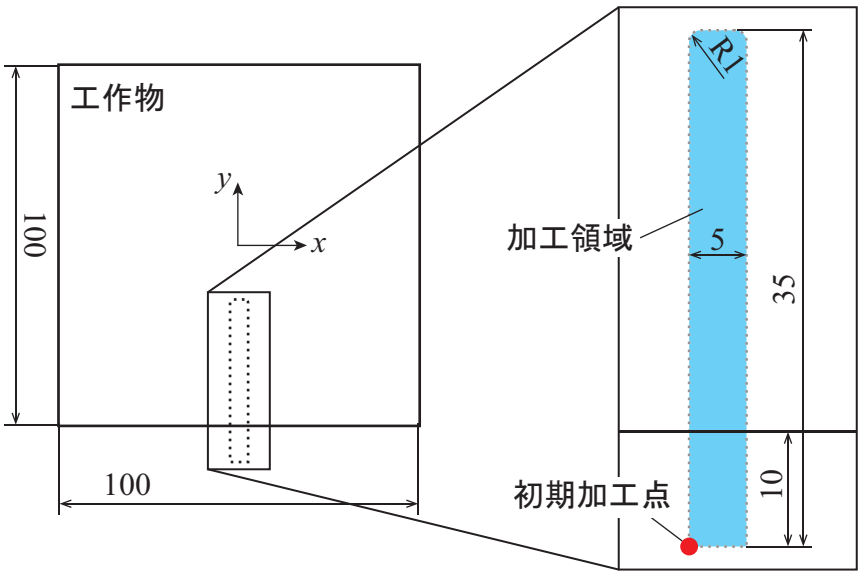


図 5.13　溝加工における加工領域と初期加工点

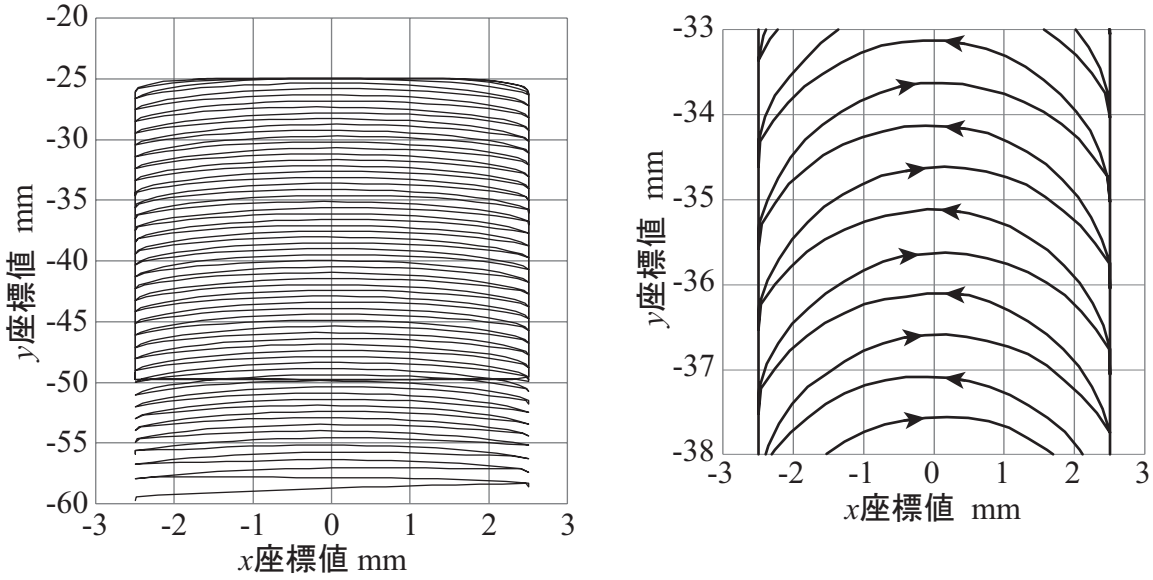


図 5.14　溝加工の工具経路生成結果

5.4.3　ポケット加工

次にポケット加工の工具経路生成を行った．図 5.15 に加工領域を示す．また，スクエアエンドミルの z 方向のアプローチ時の切削は考慮しないものとして，ポケット中心には予め工具の進入のため予備加工を行っている．この予備加工範囲も工具中心の移動範囲である．ψ は 10 度，工具進行方向ベクトル $\boldsymbol{D}$ は，図 5.16 に示すように，加工領域の中心に原点を設定し，加工点の位置ベクトルを 90 度回転させた方向とした．また，複数の加工点候補で切削体積が同じ場合，原点からの距離が一番遠い加工点候補を選択した．その他の条件は側面加工の時と同じである．

図 5.17(a) に本手法を用いて生成した工具経路を示す. 生成された工具経路を見てみると，ポケット加工に有効であることが知られている渦巻を描く工具経路を自動的に生成することができている．また，図 5.17(b) に示すような幾何計算などによる従来の手法で生成された単純な渦巻経路では，渦巻 1 回転ごとの径方向切り込み量が一定になっているが，本手法で生成したパスは渦巻の半径が大きくなるに連れて徐々に径方向の切り込み量が大きくなる経路になっていることがわかる．これは，渦巻の半径が小さいところでは，切削体積が大きくなりやすいため, 径方向切り込み量を小さくし, びびりを回避している. よって，単純な渦巻経路では，工作物中心に近いところの加工では許容する切削体積以上に加工を行ってしまい，工作物中心から離れると許容する切削体積より遥かに小さい加工を行ってしまうが，本手法では，許容する切削体積での加工が可能となり，びびり振動が発生しない範囲で最も加工能率が高い工具経路が生成されていると考えられる．

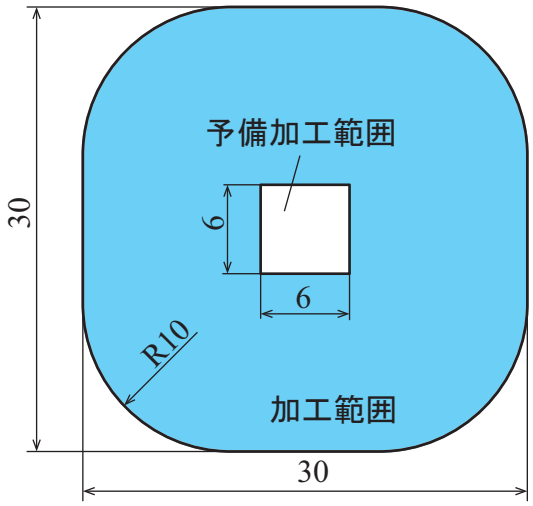


図 5.15　ポケット加工における加工範囲と予備加工範囲

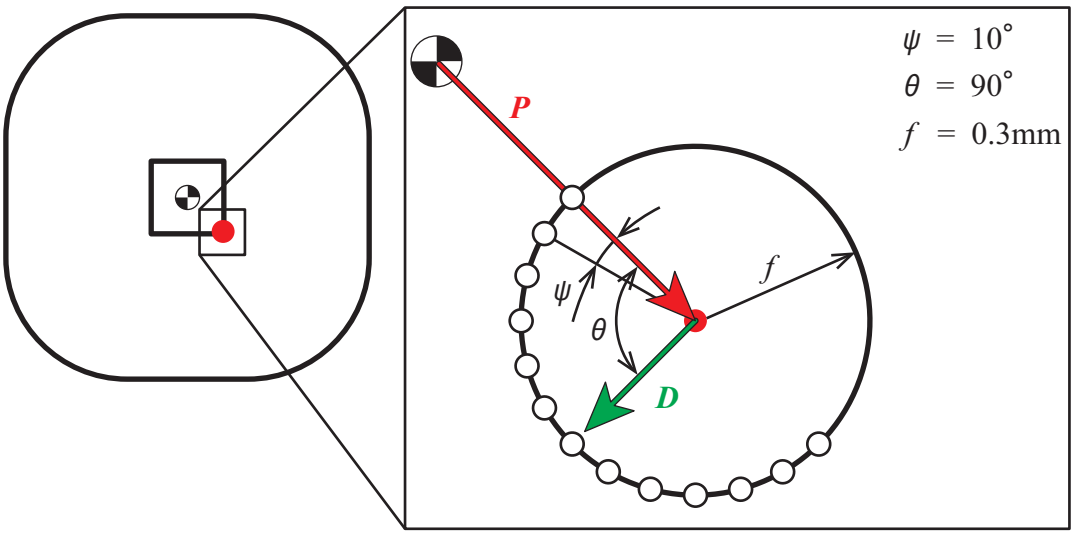
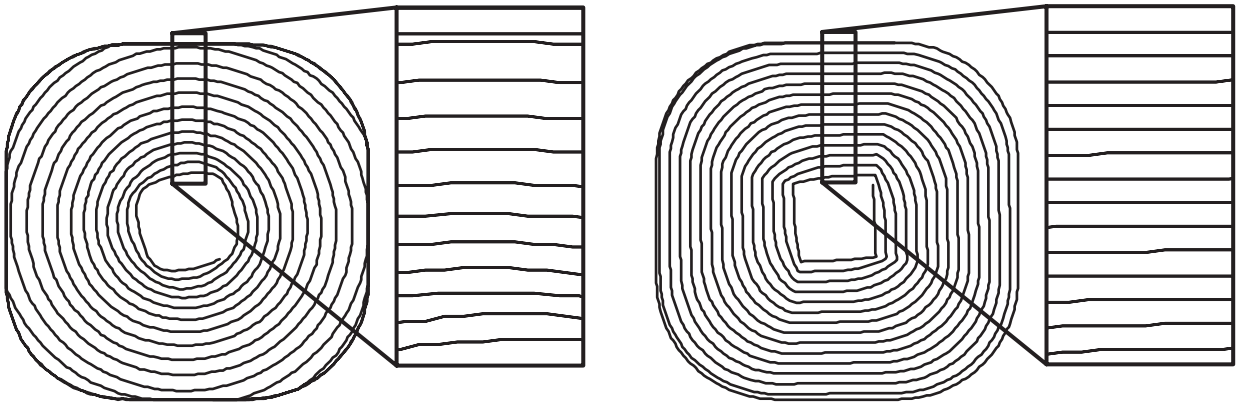


図 5.16　ポケット加工の工具進行方向ベクトルの生成方法



(a) 提案手法　　(b) 従来手法

図 5.17　ポケット加工の工具経路生成結果

5.5　加工実験

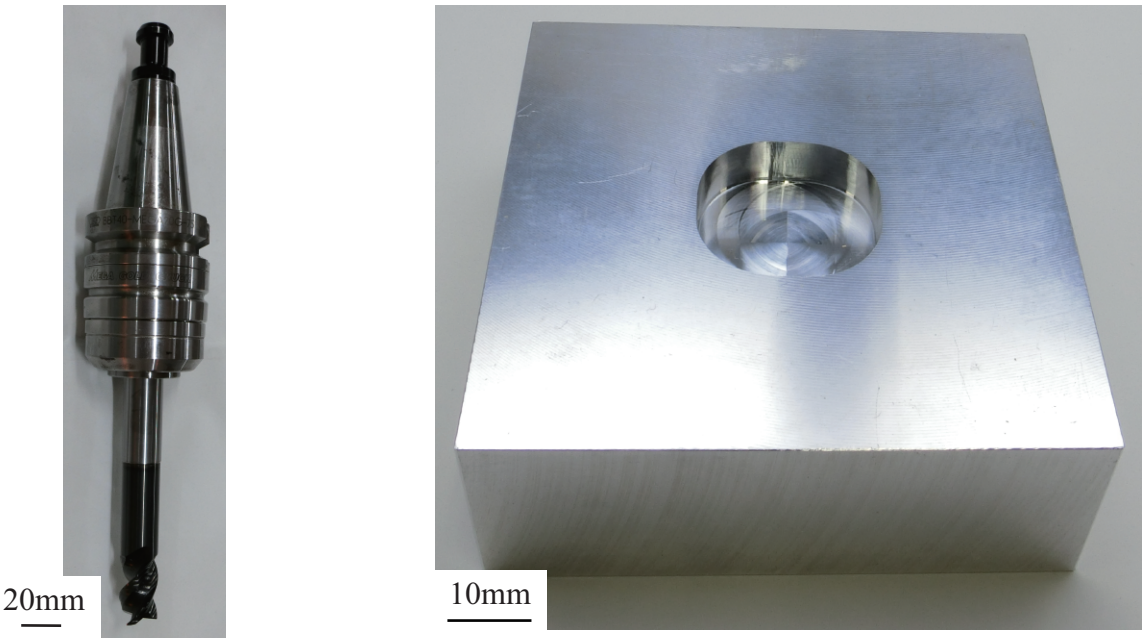
5.5.1　実験条件

本手法で生成した工具経路がびびり振動抑制できるか確認するため，実機によるポケット加工を対象とした加工実験を行った．ポケット加工を対象とした理由として，ポケット加工では，4.2 章のシミュレーション結果からもわかるように，隅部分で切削体積の増加からびびり振動が発生する可能性が高く，本手法の効果が出やすいと考えたためである．以下に実験の条件を説明する．

表 5.1 に示す加工条件で実験を行った．使用した工具，工作物の写真を図 5.18 に示す．工作物はアルミ合金である A7075 で，工具経路生成時に行っていたように，z 方向のアプローチ時の切削は考慮しないものとして，ポケット中心には予め工具の進入のため予備加工をしている．また, 工具は DLC コーティングがされている 3 枚刃スクエアエンドミル（不二越製:DLCSLTLS20）を使用した．工作機械は立形マシニングセンタ（松浦機械製作所製: FX-5G）を使用し, 工具を工作機械に取り付け, 図 5.19 に示すように工具刃先に加速度ピックアップを取り付け，ハンマリング試験を行った結果が，図 5.10 である．

使用した工具経路は 5.4.3 章で生成した図 5.17(a) の工具経路を使用した．また，比較のために図 5.17(b) の従来手法で生成した工具経路でも加工実験を行った．本手法で生成した工具経路をパス 1，従来手法で生成した工具経路をパス 2 と呼ぶ．パス 1 とパス 2 は同等の巻数に設定している．

図 5.20 に実験の概要図を示す．動力計（KISTLER 社製：9272）上に工作物を取り付け，切削力を測定し，工作機械のテーブル上に取り付けた加速度ピックアップで加工中の加速



(a) 工具　　(b) 工作物

図 5.18　加工実験に使用した工具と工作物

度を測定し，FFT を行った．この測定する切削力と加速度の 2 つからパス 1，パス 2 でびびり振動の有無を判断する．また，開発した切削シミュレータを用いて，切削体積などのシミュレーション結果を比較する．

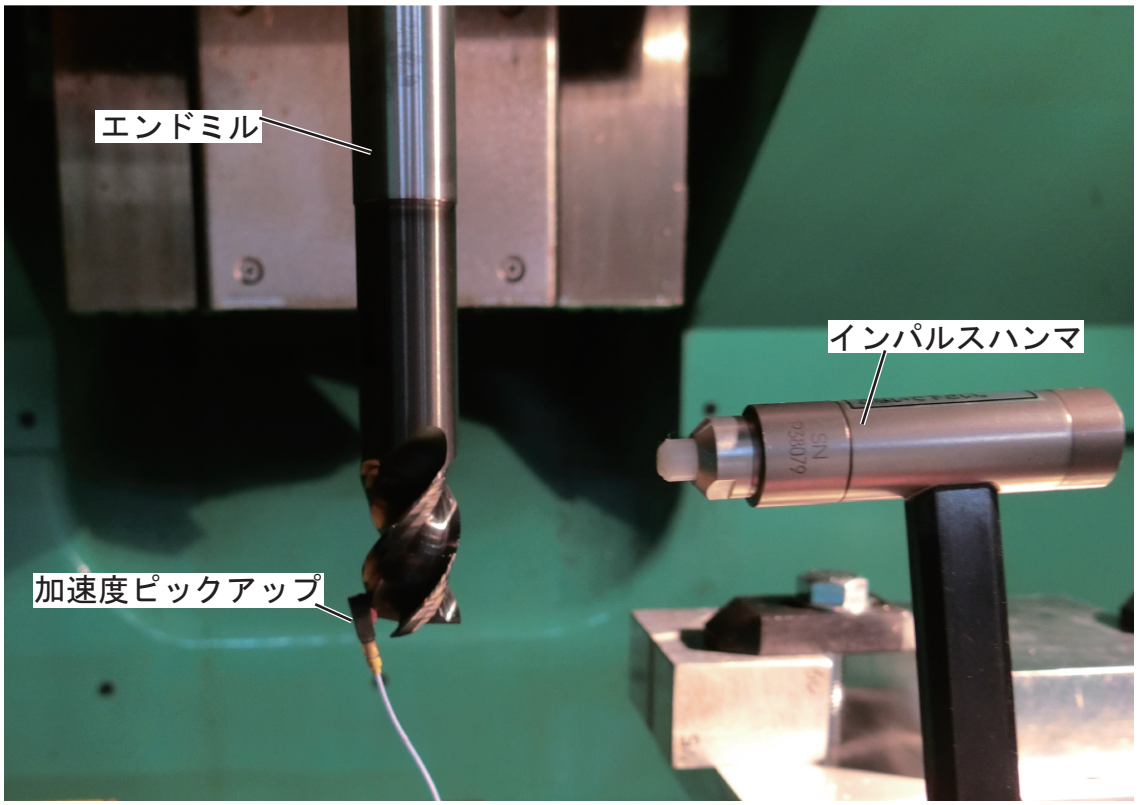


図 5.19　ハンマリング試験の様子

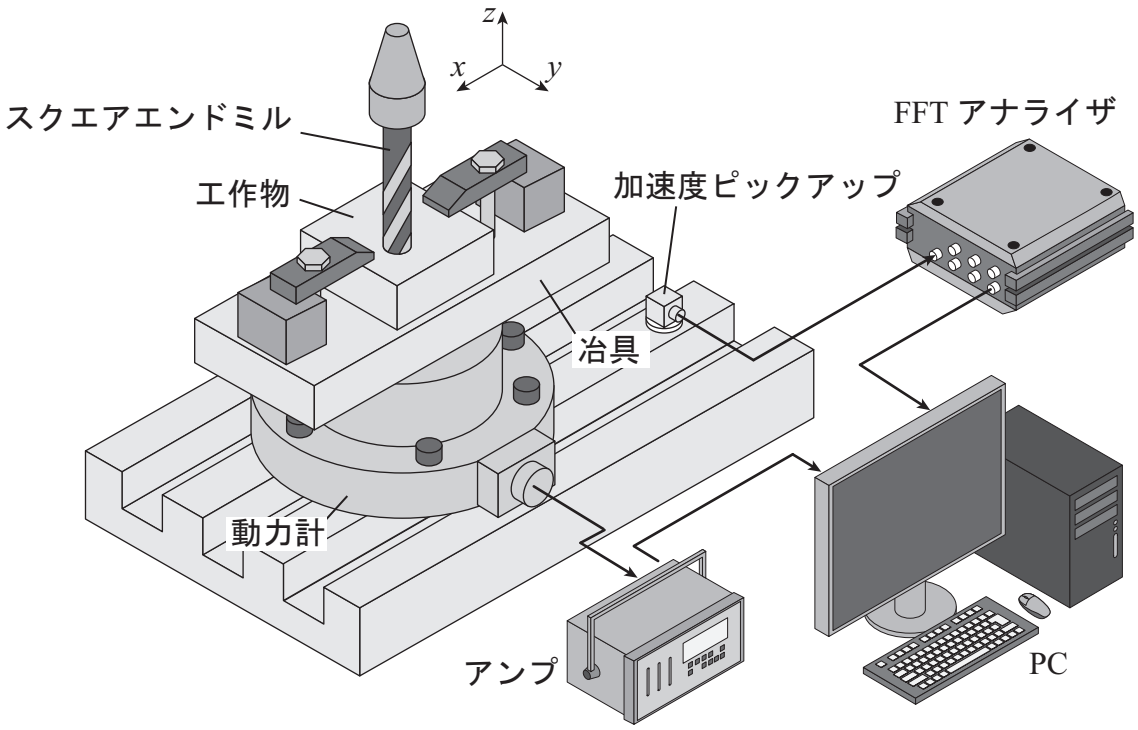


図 5.20　実験模式図

5.5.2　シミュレーション結果

パス 1，パス 2 のシミュレーション結果を図 5.21 ～図 5.24 に示す．パス 1，パス 2 それぞれ工具経路全体と加工開始から加工距離 100 mm までで，工具経路の x，y 座標値と切削体積，切削関与角，びびり振動なく加工可能な軸方向切り込み量である安定限界切り込み量のシミュレーション結果を示している．

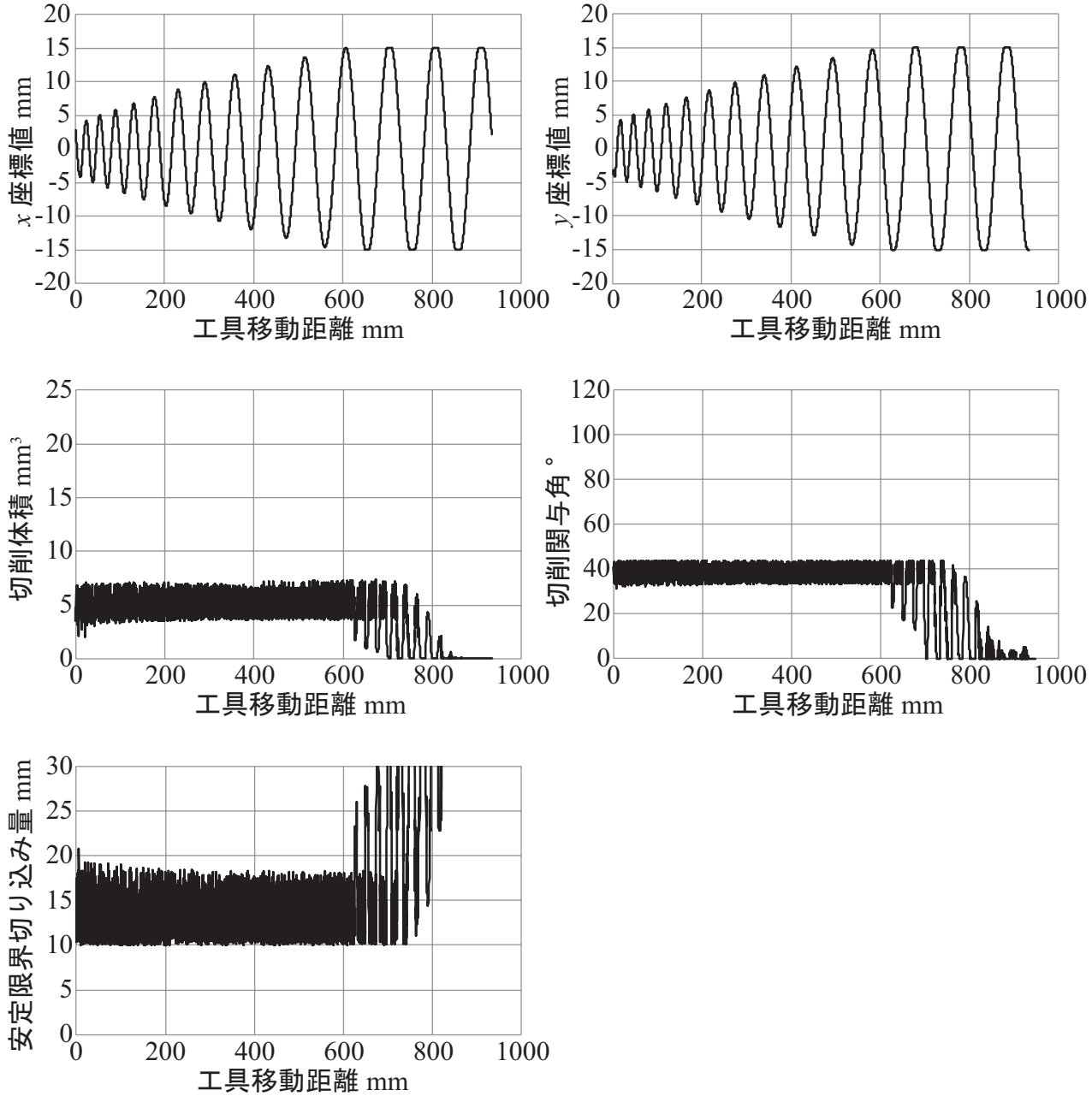


図 5.21　パス 1 のシミュレーション結果（工具経路全体）

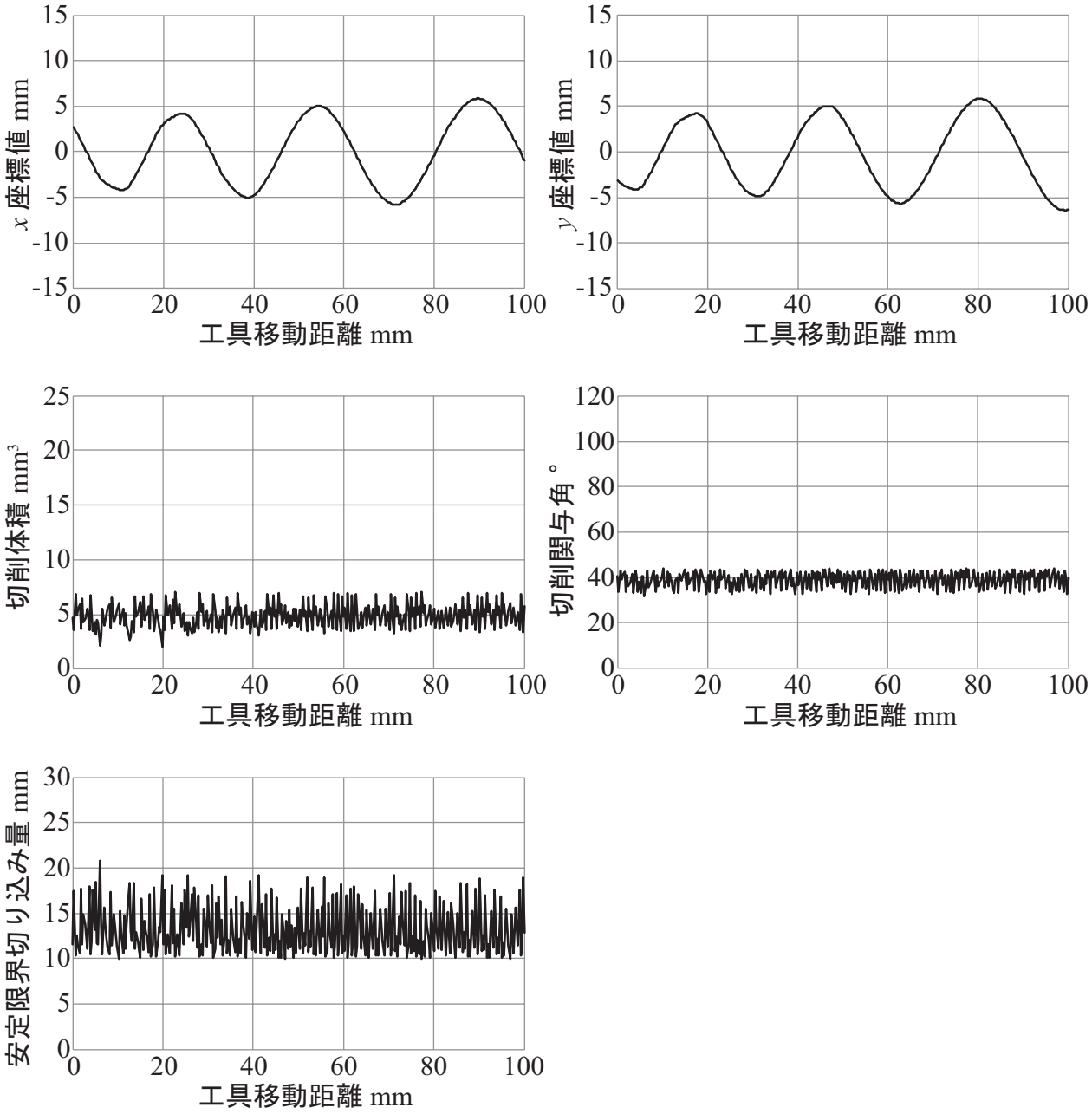


図 5.22　パス 1 のシミュレーション結果（工具移動距離 100 mm）

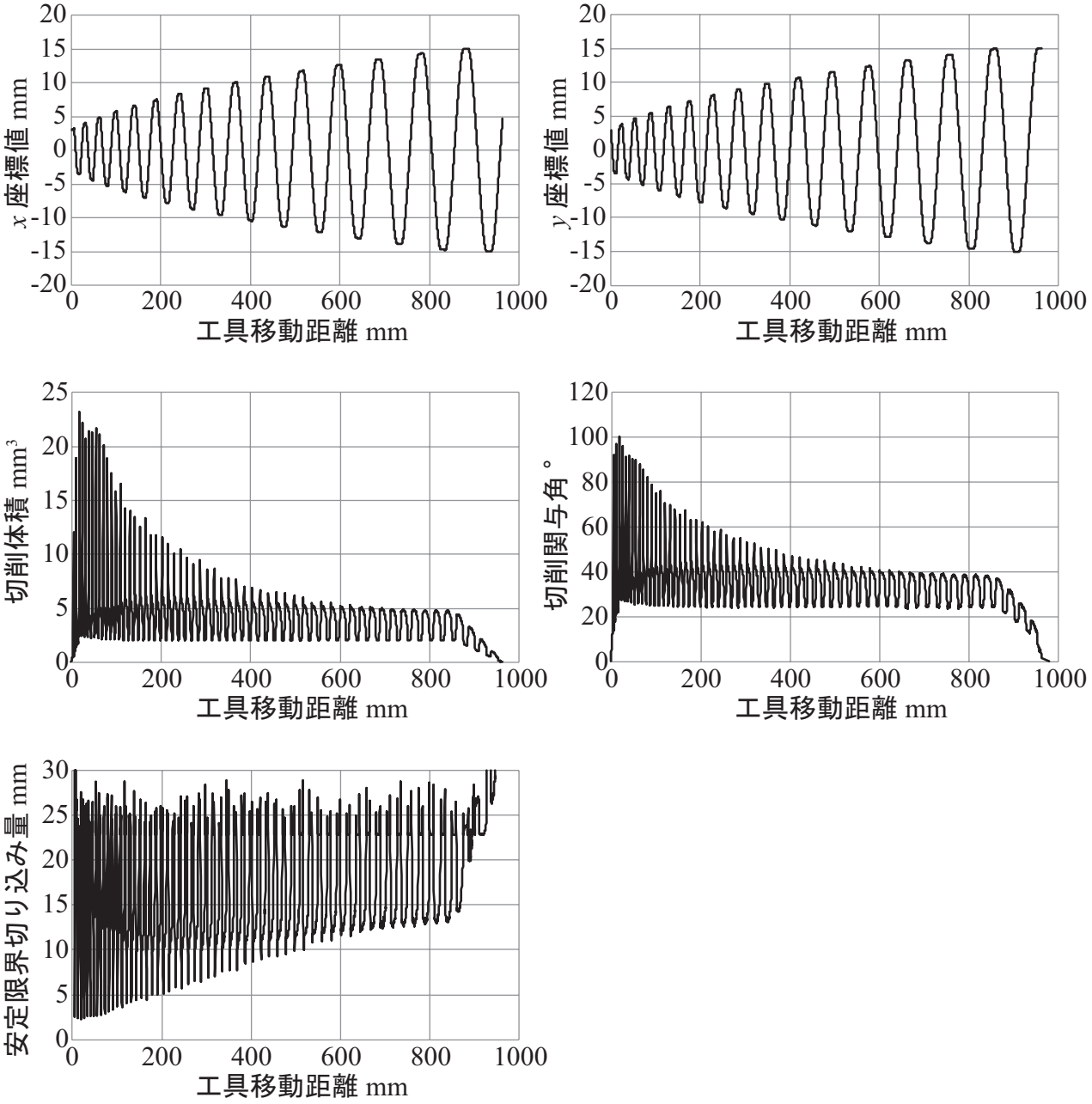


図 5.23　パス 2 のシミュレーション結果（工具経路全体）

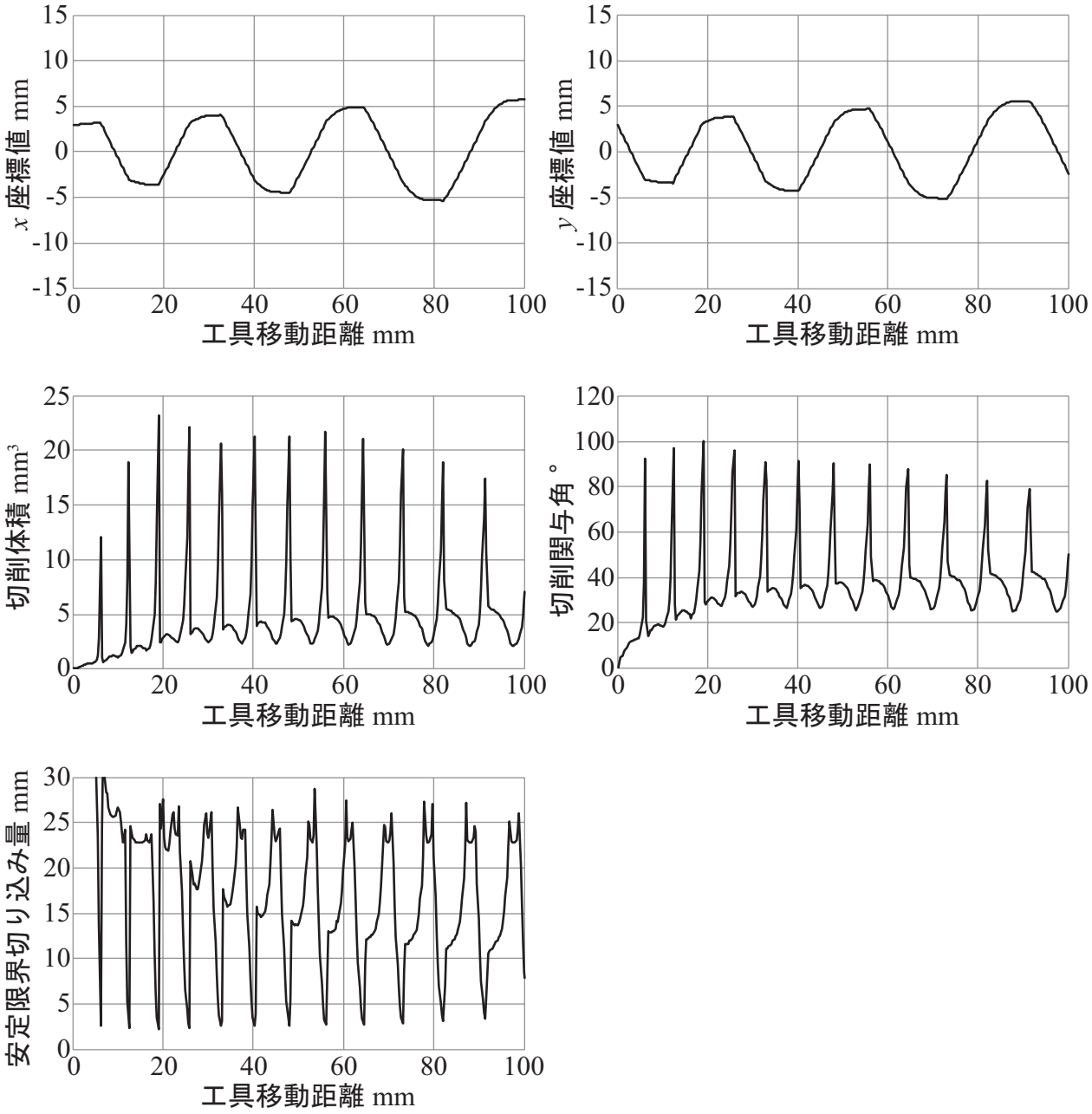


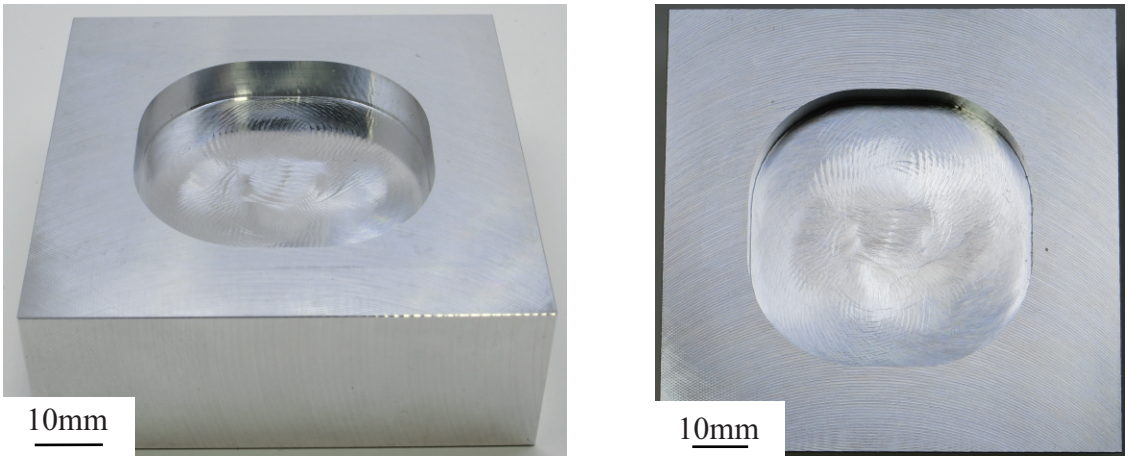
図 5.24　パス 2 のシミュレーション結果（工具移動距離 100 mm）

パス 1 のシミュレーション結果を見ると，切削体積と切削関与角が一定になっていることがわかる．工具移動距離 600 mm 以降で切削体積，切削関与角どちらも減少している箇所があるが，これはポケット加工の直線部分のエアカットによるものだと考えられる．また，安定限界切り込み量を見ると，指定した軸方向切り込み量の 8 mm に安全率 25 %掛けた 10 mm 以上に常になっており，8 mm で加工を行ってもびびり振動が発生せずに加工を行うことができる．一方，パス 2 の結果を見てみると，加工開始直後に非常に大きな切削体積, 切削関与角になっており, 安定限界切り込み量は最低で約 2 mm ほどになっている．これは隅部分で方向転換を行う際に，加工開始時ではほぼ 90 度で方向転換を行い，切削体積が急激に増加するためだと考えられるが，工具移動距離 600 mm 以降は，切削体積などはパス 1 とほぼ同じになっている．よって，加工開始直後でパス 1，パス 2 の加工結果に大きな差が発生すると考えられ，パス 2 では，びびり振動が発生すると考えられる．

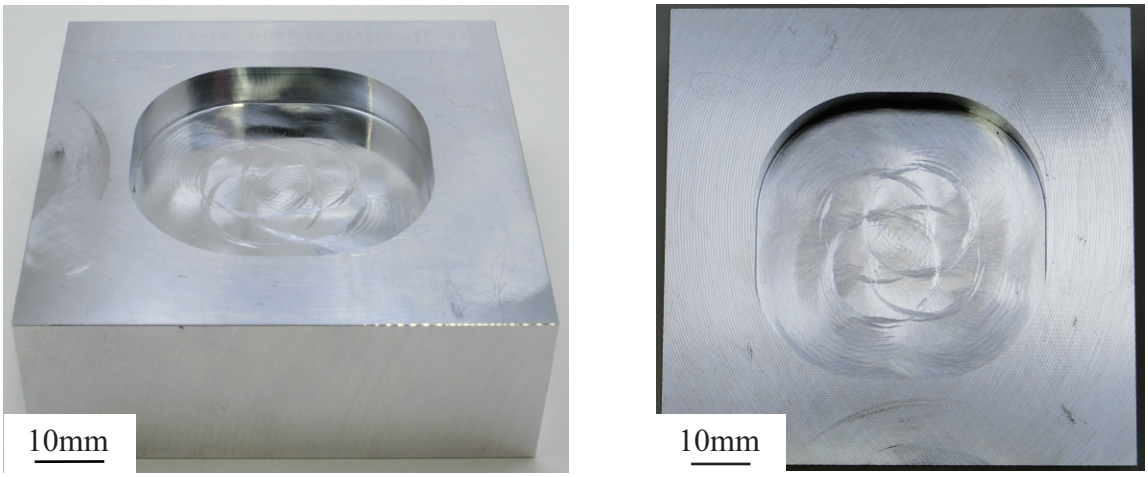
5.5.3　加工結果

パス 1，2 を用いて，実機による加工を行い，加速度，切削力の測定を行った．図 5.25 に加工後の工作物を示す．ほぼ切削後の側面の表面に違いは生じなかった．これは，上述した切削シミュレーション結果を見てわかるように，加工開始直後では切削体積などに大きく差が生じているが，加工中盤以降では切削体積などに差がないことがわかる．よって，加工開始直後では表面性状に違いが出ていたと考えられるが，その加工面を加工してしまったため，表面性状に違いが出なかったと考えられる．一方，底面の切削痕を見てみると，パス 2 のみ 4 箇所大きな切削痕が見られる．これは，ほぼ 90 度で方向転換を行っている角部分の切削痕だと考えられる．

パス 1，2 の加工中の加速度の FFT 結果を図 5.26，振動変位を図 5.27 に示す．この測定結果は加工開始から終了までの測定結果である．また，工作物に対してハンマリング試験を行い，コンプライアンスを測定した結果を図 5.28 に示す．加工中の加速度の測定結果は加速度に差が見られるが，ピークがでている周波数など，傾向は x 軸方向，y 軸方向ど

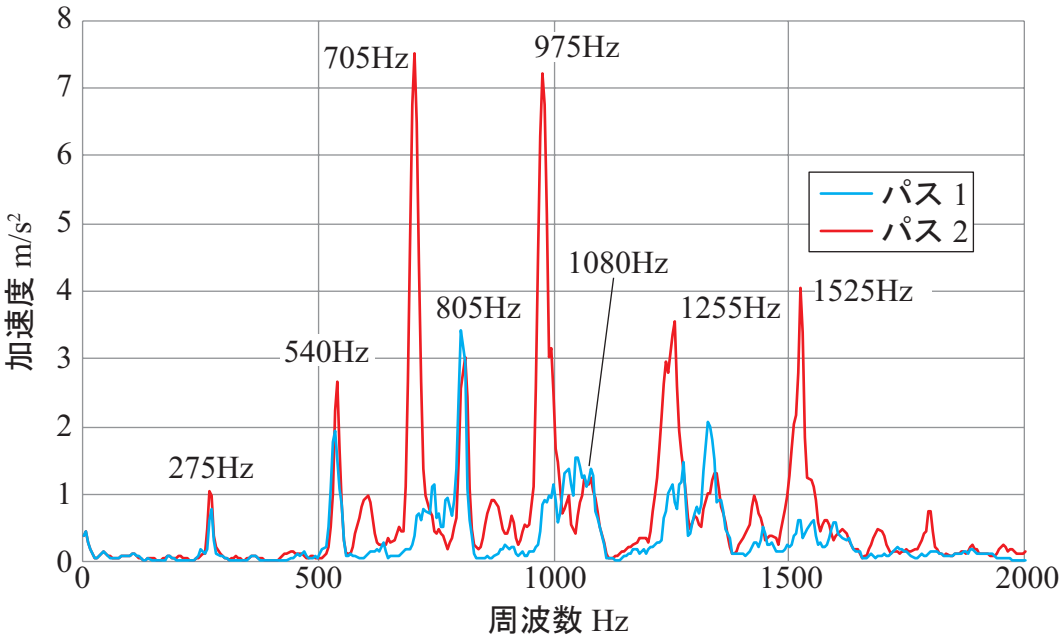


(a)　パス 1

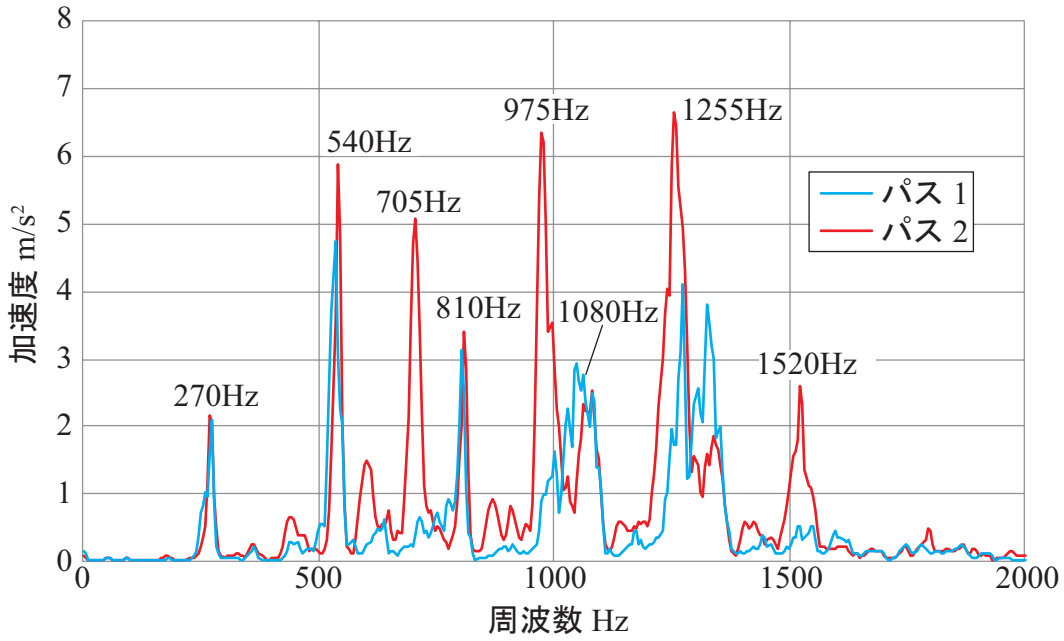


(b)　パス 2

図 5.25　加工後の工作物

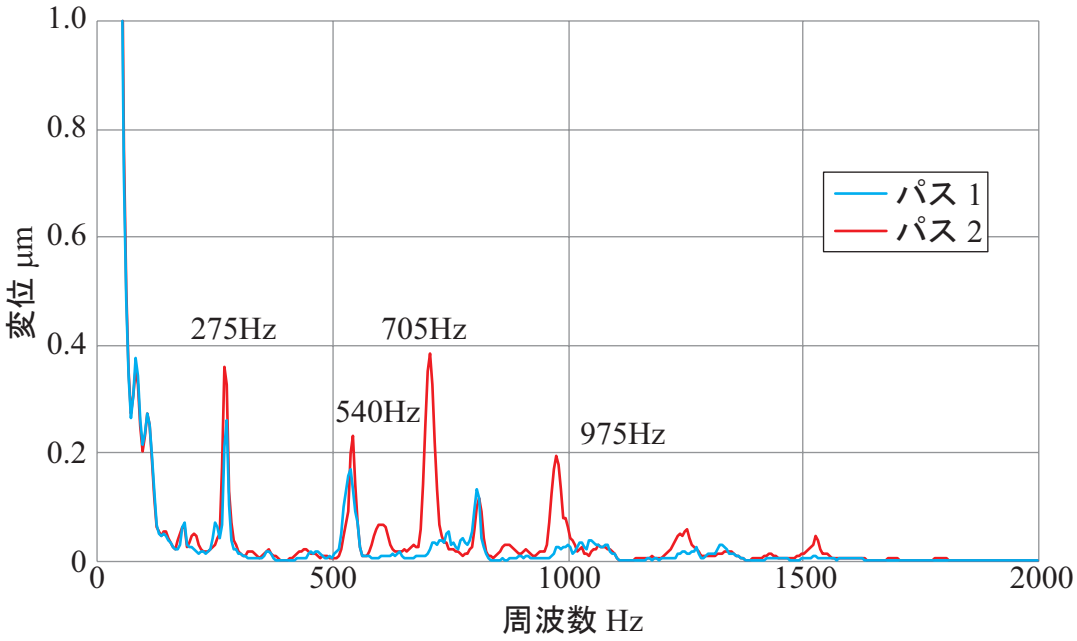


(a)　x 軸方向

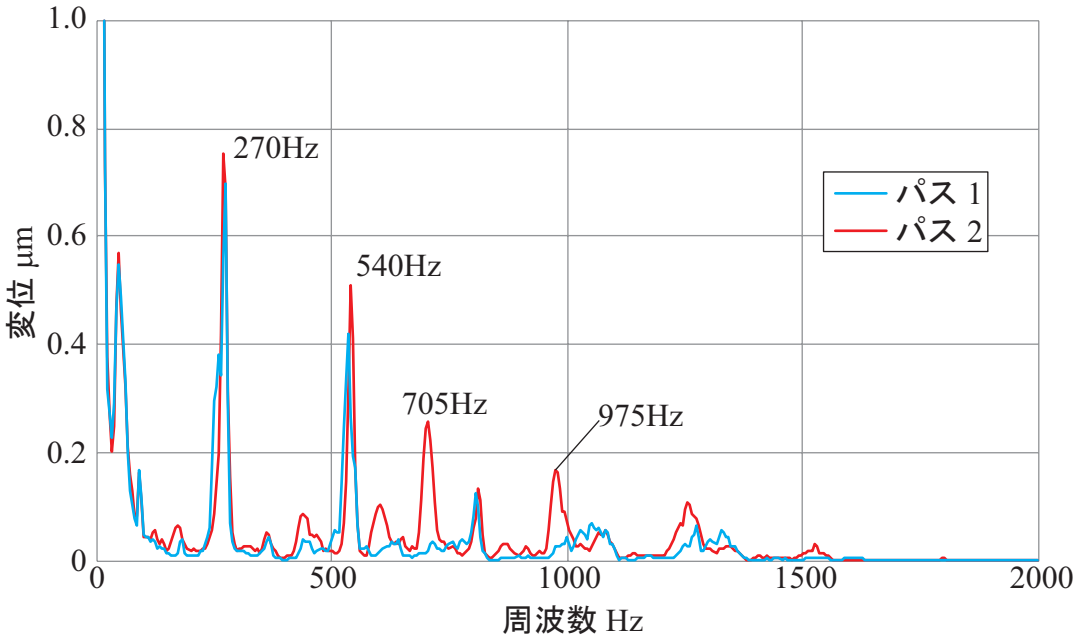


(b)　y 軸方向

図 5.26　加工中の加速度のパワースペクトル



(a)　x 軸方向



(b)　y 軸方向

図 5.27　加工中の振動変位のパワースペクトル

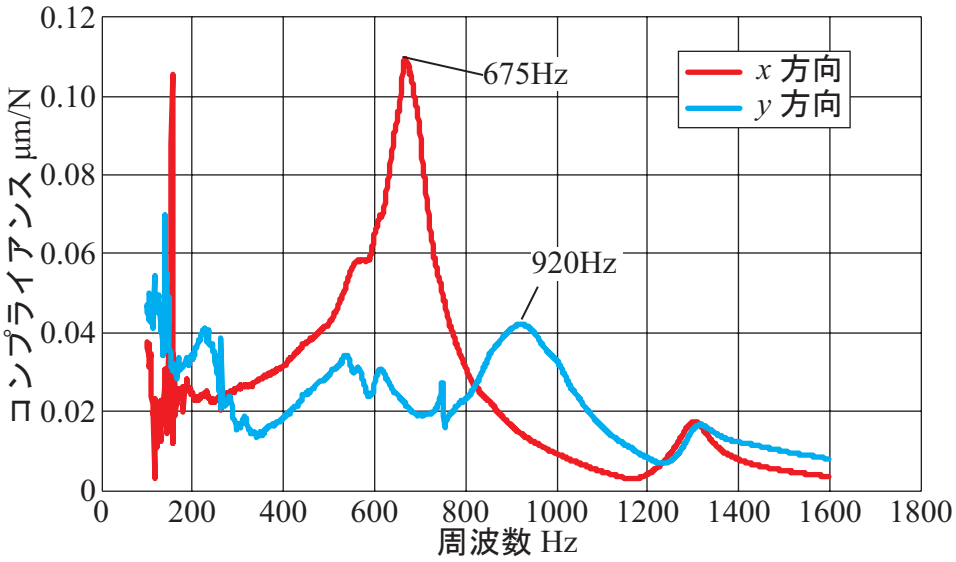
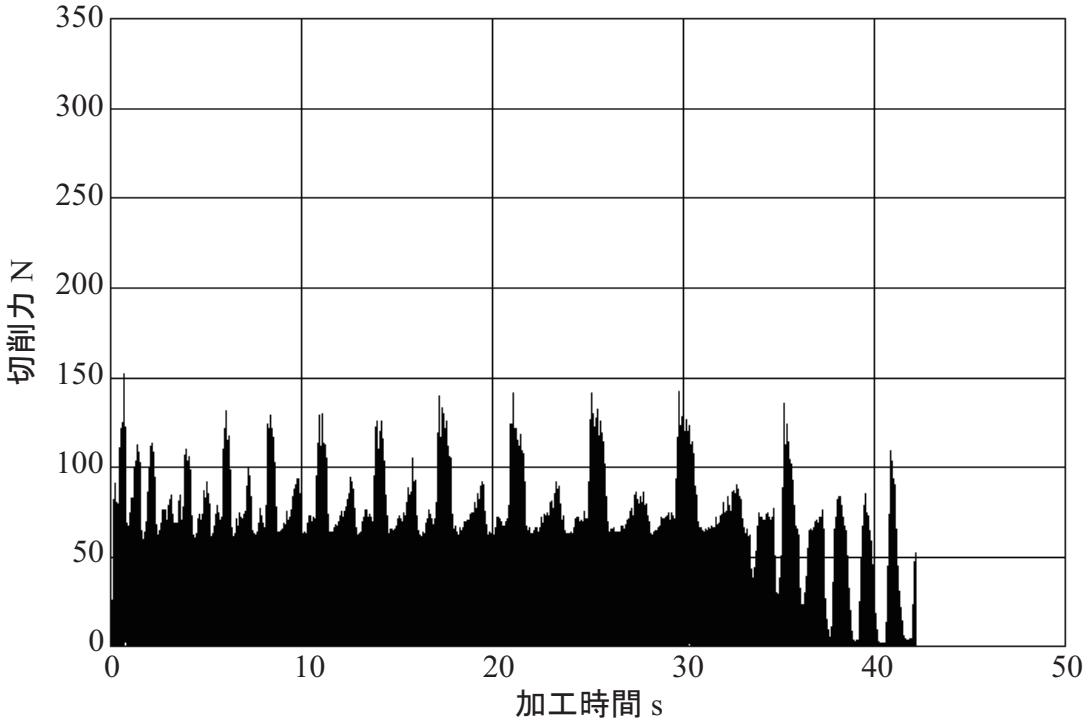


図 5.28　工作物のコンプライアンス

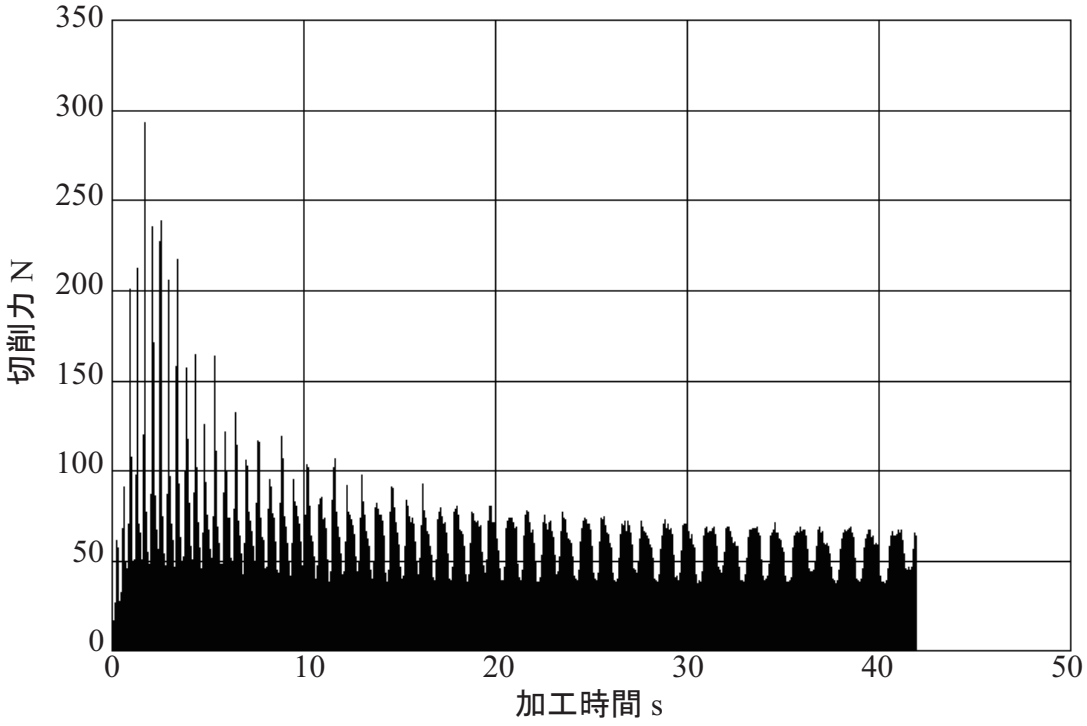
ちらもほぼ同じ結果となった．測定結果を見ると，工具の断続切削による周波数の 270 Hz およびそのハーモニクスはどちらの工具経路においても見られる．しかし，パス 2 のみ，約 705 Hz，975 Hz，1255 Hz，1520 Hz の 4 箇所に大きな振動が確認できる．また，工作物のコンプライアンスの共振周波数である 675 Hz，920 Hz 以外の部分で振動が発生していることから，パス 2 のみ発生している 4 箇所の振動がびびり振動だと考えられる．そして，振動変位の結果を見てみると，工具の断続切削による振動変位と比べ，びびり周波数の振動変位の大きさがほぼ等しいか，やや小さいことから，びびり発生の境界付近での切削だったと思われる．ハンマリングによる工具の固有振動数が 550 Hz 近辺であったのに対し，びびり周波数が 700 Hz 近辺であったことからもそれがうかがえる．よって，本手法を用いた場合，びびり周波数でのピークが抑えられており，びびり振動を低減する効果は十分確認できる．

次に切削力の測定結果を図 5.29 に示す．また，切削シミュレーション結果が大きく異なった加工開始 5 秒間の切削力測定結果を図 5.30 に示す．また，図 5.31 に図 5.30 に示す部分の切削体積のシミュレーション結果を示す．この切削力の測定結果は，x 方向，y 方向の切削力の合力の結果である．パス 2 はスパイク状の切削力が増加が見られる．これは，角部分の切削の際に急激に切削体積が大きくなり，切削力増加したと考えられる．一方，パス 1 では，パス 2 ほどの急激な切削力の増加が見られず，隅部分において切削体積が緩和されるようなパス生成が行われたことがわかる．また，シミュレーション結果と比較するとパス 2 のスパイク形状が切削体積でも確認することができ，発生している間隔も一致していることから，開発した切削シミュレータの有効性が確認できる．

加工中の加速度測定と切削力測定から，本手法を用いて生成した工具経路は，急激な切削力増加を抑え，びびり振動を低減する効果が確かめられた．

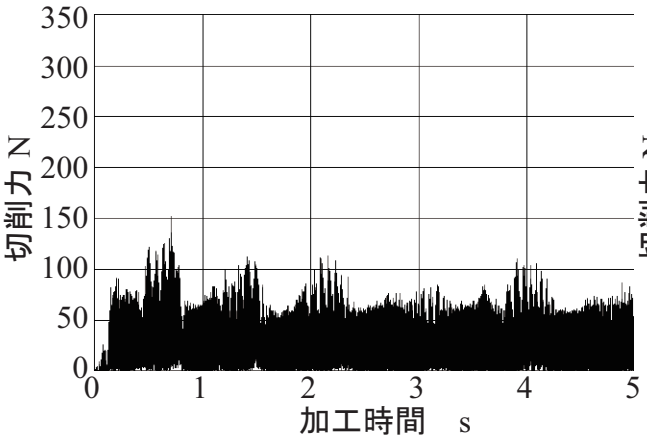


(a)　パス 1

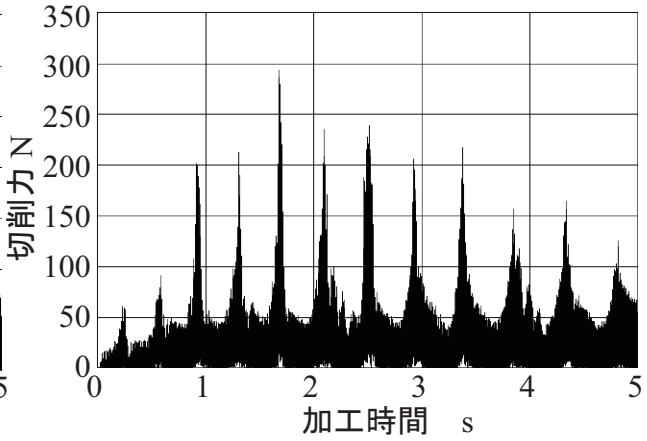


(b)　パス 2

図 5.29　切削力測定結果

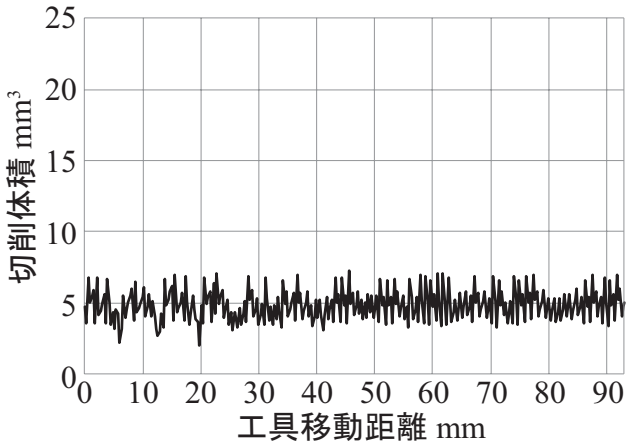


(a) パス 1

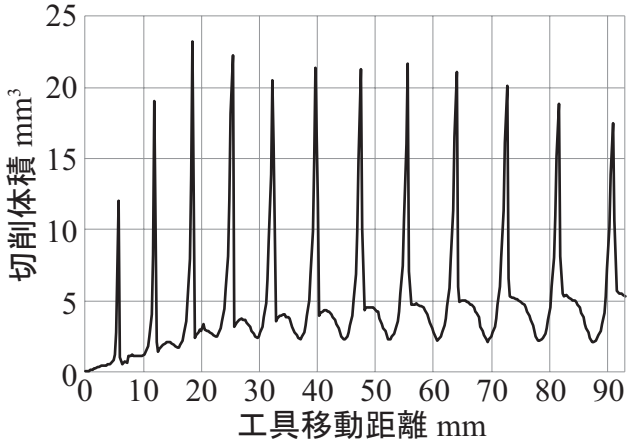


(b) パス 2

図 5.30 加工開始 5 秒間の切削力測定結果



(a) パス 1



(b) パス 2

図 5.31 切削体積のシミュレーション結果

5.6 まとめ

開発した切削シミュレータの応用例として，びびり安定限界解析を用いたびびり振動を抑制した工具経路生成方法を開発した．また，ポケット加工を対象とした実切削によって，一般的な渦巻経路との比較試験を行い，本手法の有効性を示した．以下に本章のまとめを示す．

(1) 本手法を使用して側面加工，溝加工，ポケット加工の 3 種類のパスを生成し，1 つの加工点当たり最大でも約 0.46 秒で高速に加工点を生成することができた．
(2) ポケット加工の工具経路に対してシミュレーションを行い, 切削体積がほぼ一定になっていることを確認した．
(3) ポケット加工の実加工を行い，加工中の振動を測定した結果，渦巻経路で発生していた工具の刃先が工作物に接触する周波数とは異なる振動が本手法の工具経路では発生しないことを確認した．
(4) 切削力を測定した結果，本手法の工具経路では渦巻経路のように急激に切削力が変化しないことを確認した．
(5) 振動測定，切削力測定から，本手法の工具経路では切削力を一定にし，びびり振動を低減することができた．

参考文献

[1] 赤澤浩一 , 社本英二 : 低剛性工作物のボールエンドミル加工における再生型びびり振動に関する研究 - 切削送り方向周りの工具傾斜を考慮した解析モデルの構築 - , 精密工学会誌 ,75,8(2009),984

[2] 廣垣俊樹 , 青山栄一 , 塩田亮祐 , 青谷凱斗 : エンドミル加工面のびびり振動模様に基づく現象の逆解析と安定ポケットの探索方法 , 日本機械学会論文集 ,83,848(2017),DOI:10.1299/transjsme.16-00362

[3] 内海幸治 , 河野一平 , 小野塚英明 , 加藤吐夢 , 笹原弘之 : 薄板加工におけるびびり振動抑制用動吸振器の設計方法に関する研究 , 精密工学会誌 ,81,2(2015),187

[4] 社本英二 , 影山和宏 , 森脇俊道 : 不等ピッチエンドミルによる再生型びびり振動の抑制 : 解析モデルの構築とピッチ角の最適化 , 日本機械学会関西支部講演論文集 , 024-1(2002),3-5

[5] T.Kojima, N.Suzuki, R.Hino, E.Shamoto: A novel design method of variable helix cutters to attain robust regeneration suppression, Procedia CIRP 8,(2013), 363

[6] 小池綾 , 柿沼康弘 , 青山藤詞郎 , 大西公平 : サーボ情報を応用したびびり振動に対する安定主軸回転数同定 , 日本機械学会論文集 ,81,830(2015),DOI: 10.1299/transjsme.15-00387

[7] 上野浩 : 機械加工における " びびり振動 " と制御技術～「加工ナビ」による加工効率向上の事例 , 機械技術 ,68,15(2020),76

[8] Y.Altintas, E.Budak: Analytical Prediction of Stability Lobes in Milling, Annals of the CIRP, 44,1(1995),357

第 6 章　結論

6.1　結論

本研究では高速高精度の切削シミュレータ開発を目的として，ポリゴン表現と高速にブール演算が行える Vatti クリッピングを用いた切削シミュレータの開発を行った．また，開発した切削シミュレータの高速性を利用し，びびり振動を抑制する工具経路生成を行い，実際に加工実験を行い，本手法の有効性の検証を行った．

本研究で得られた結論を各章ごとにまとめると以下の通りである．

第 1 章では，本研究の背景及び目的を明らかにし，本研究で用いた PC やライブラリの説明を行った．

第 2 章では，ポリゴンの種類や本研究で対象とするポリゴンの説明を行い，本研究で使用する Vatti クリッピングのアルゴリズムについて説明し，Vatti クリッピングを切削シミュレータに応用する上での 3 つの問題点に関して明らかにした．

第 3 章では，第 2 章で述べた 3 つの問題点の解決方法を提案した．頂点数増加による計算時間の増加に対しては工作物と工具の大きさの比率に着目し，工作物のポリゴンを格子状に分割することで，ブール演算を行う範囲を制限することによって頂点数が増加しても計算時間が一定以上に増加しない方法を提案した．円を表現する際には大量の頂点を使用した多角形で表現しており，計算時間と精度に問題があったが，Vatti クリッピングを円弧に対応させることによって，計算時間を削減しながら精度を向上させることに成功した．最後に，2 次元ポリゴンを積層し，各レイヤにインデックスを用意し，ブール演算を行うレイヤを最小限することで，3 次元切削シミュレータを開発した．

第 4 章では，第 3 章で開発した切削シミュレータに対して，ポケット加工とスパイラル加工の 2 種類でシミュレーションを行い，高速性の評価を行った．z 方向に移動が少なく，レイヤのインデックスが同じになり，計算量が小さいポケット加工では，実際の加工にはないような高速な加工条件より短い計算時間でシミュレーションを行うことができた．一方，スパイラル加工では，z 方向の移動が多く，各レイヤで異なるポリゴンになり，非常に計算量が多いシミュレーションであったため，ポケット加工での計算速度程ではないが，一般的な加工条件と同等の時間でシミュレーションを行うことができ，開発した切削シミュレータの高速性を示すことができた．

第 5 章では，開発した切削シミュレータの高速性を利用した応用例として，びびり振動を抑制する工具経路生成方法を提案した．提案した工具経路生成方法は，1 回転当たりの送り量ごとにびびり安定限界解析を使用して最適な加工点を生成し，この処理を繰り返し行うことで，びびり振動が発生しない工具経路生成を行った．この手法を用いて，側面加工，溝加工，ポケット加工の 3 種類の工具経路生成を行い，最大でも加工点 1 点あたりで 0.46

秒で高速に加工点を生成することができた．また，生成したポケット加工と従来の幾何計算等で生成した渦巻経路のシミュレーションと加工実験を行い，本手法は切削体積，切削力を一定にし，びびり振動を低減することができた．

6.2 今後の展望

本研究は高速高精度の切削シミュレータを開発し，切削シミュレータの応用例としてびびり振動を抑制した工具経路生成を行い，本研究の有効性を示すことができたが，以下に示す課題を残している．

(1) 3 次元化の改善

本研究では 3 次元化をレイヤの積層によって実現したが，z 方向の精度はレイヤ間距離に依存し，このレイヤ間距離を小さくし，大量のレイヤを使用した場合，z 方向の精度がよくなるが，計算時間が多く掛かってしまう．よって，本手法とは全く異なる方法で 3 次元化を行い，精度と計算時間の両立を図る必要がある．

(2) 5 軸工作機械用の切削シミュレータ開発

今回は 3 軸の工具経路を対象として切削シミュレータを開発した．しかし，近年では 5 軸工作機械を用いて，工具の姿勢を変化させ複雑な工作物の加工を行っているため，切削シミュレータの 5 軸対応が必要であると考えられる．

(3) スクエアエンドミル以外の工具を用いたシミュレーション

4 章で述べたように，複数のレイヤに対する処理でも高速に処理することができたため, 工具にもレイヤを用意し, ボールエンドミルなどを使用したシミュレーションを高速に行うことができると考えられ，実装，有効性の確認が必要である．

(4) 切れ刃形状の表現

今回は工具を回転掃引させ，円形状で表現したが，この方法では切れ刃を表現していないため，エンドミルのねじれ角の影響などに関して，シミュレーションすることができていない．そこで，切れ刃まで表現し，より現実に近づいたシミュレーションが行う必要がある．また，これによって，びびり振動抑制工具経路生成に関しても，今回使用したびびり振動解析以外のより高精度なびびり振動解析を行うことができ，工具経路生成の精度も向上すると考えられる．

(5) 3 軸以上の切削に対応したびびり振動抑制工具経路生成

今回提案した工具経路生成方法は軸方向切り込み量が変化しない 2 軸の加工での工具経路生成方法であり，軸方向切り込み量や工具の姿勢が変化する 3 軸以上の加工でも工具経路生成を行うことができれば，より多くの加工に対してびびり振動を抑制することができる．

謝辞

本論文を作成するにあたり，指導教員として多くのご指導ご助言を賜りました，金沢大学　浅川直紀教授，高杉敬吾助教に心より厚く御礼申し上げます．また，本論文の作成のみに留まらず，研究活動に対する姿勢とその楽しさを教えていただけたことは私の今後の糧となるものでした．ありがとうございました．

ご多忙の中，副査をお引き受け頂きました，金沢大学　関啓明教授，古本達明教授，細川晃教授，金沢工業大学　森本喜隆教授に心から感謝の意を表します．

公私にわたってさまざまなことに関して常に相談に乗って頂きました元金沢大学　浅野久志技術職員に心から感謝申し上げます．

実験治具の設計および加工，その他多方面におけるご指導を賜りました金沢大学　山岸大輔技術職員に心から感謝申し上げます．

また，本研究のソフトウェア開発にあたって，さまざまなご助言，ご指摘を賜りました株式会社ソフィックス　水谷貴彦様，佐藤大輔様に心から感謝申し上げます．

さらに，本研究を進めるにあたり，さまざまなご協力，アドバイスを頂いたマンマシン研究室の諸先輩方，諸同輩，諸後輩の存在は研究を進めていく上で大きな励みとなりました．心から感謝申し上げます．

日本学術振興会には，博士課程 2 年より特別研究員に採用頂き，2 年間の公私にわたる資金的な援助を頂きました．熱く御礼申し上げます．

最後に，今日に至るまでの学生生活を暖かく見守り続け，支援してくれた両親と家族に対して深く感謝の意を表して謝辞といたします．