

大学入試センター試験数学 1・A を用いた数式処理システムの 性能評価

プロジェクトマネジメントコース
ソフトウェア開発管理グループ
矢吹研究室
1242116
森谷慧士

謝辞

本研究を進めるにあたり，矢吹研究室矢吹太郎准教授には，多くの時間をご指導にさいて頂きました．また矢吹研究室の皆様には，多くの知識や示唆を頂きました．協力していただいた皆様に感謝の気持ちと御礼を申し上げます．

目次

第 1 章	序論	7
第 2 章	背景	13
2.1	第 4 の産業革命	13
2.2	ソフトバンクの Pepper	15
2.3	コールセンターに Watson を導入	18
第 3 章	目的	19
第 4 章	プロジェクトマネジメントとの関連	21
第 5 章	手法	23
5.1	コンピュータシステム	23
5.2	解答方法	24
5.3	研究過程	25
5.4	問題処理の例	26
5.5	集計	27
第 6 章	大学入試センター試験	29
6.1	大学入試センター試験とは	29
第 7 章	数式処理システム	31
7.1	Mathematica とは	31
7.2	Wolfram 言語とは	32
7.3	Wolfram 言語の使用例	33
7.4	Mathematica の使用例	34
第 8 章	調査	37
8.1	調査方法	37
8.2	第一工程	37
8.3	第二工程	38
第 9 章	結果	39
9.1	使用したシンボル	41
9.2	利用した数学的知識	59
第 10 章	考察	61
第 11 章	結論	65

参考文献

67

第 1 章

序論

東京大学の入試問題を全自動で解くプロジェクト（東ロボプロジェクト）が進められている [1] .

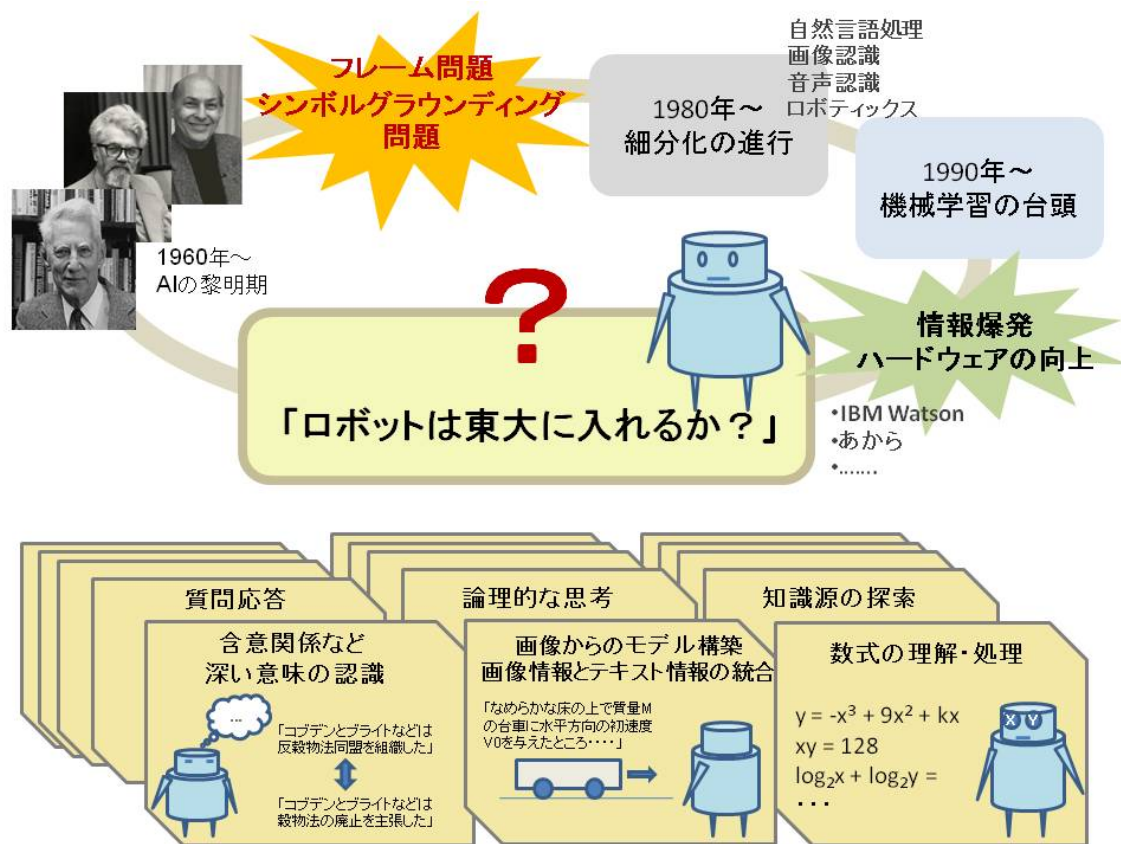


図 1.1 東ロボプロジェクト

このプロジェクトは、国立情報学研究所（大学共同利用機関法人 情報・システム研究機構）が、中心となって 1980 年以降細分化された人工知能分野を再統合することで新たな地平を切り拓くことを目的に、若い人たちに夢を与えるプロジェクトとして発足した [2] .

このプロジェクトの具体的なベンチマークとして、2016 年度までに大学入試センター試験で高得点をマークすること、また 2021 年度に東京大学入試を突破することを目標に研究活動を進めている。

これまで蓄積された人工知能の各要素技術の精度を高め、情報技術分野の未来価値創成につなげるとともに、人間の思考に関する包括的な理解を内外の研究者とともに深めることを目標としている。また、本プロジェクトでは、日本における学際的な知識・先端技術を集積するだけでなく、国際的な連携も視野に入れ、研究活動を進めている。

ここでは、東ロボプロジェクトの中でも特に数学について紹介する。

東ロボは、情報処理技術の観点から人間の知能に対するアプローチの一つとして取り組んでおり、数学入試問題で東大入試に合格できるレベルの技術を作ることを目的としている。

東ロボでは、NII と富士通研究所が共同で、問題文を認識・解釈しコンピュータが理解できるデータの形にする数式認識や、数式処理ソルバが理解できる式表現を立式するための自然言語処理、立式された問題を高速に正確に解くための数式処理技術など、人間中心の IT を実現するために必要な技術の開発を行い、目標達成を目指している。このプロジェクトで開発された技術により、高度な数理解析技術が誰にでも容易に使えるようなものになることが期待される。

東ロボプロジェクトでは、数学の問題を解く過程を以下の3つの手順にしている。

1. 人間にとって理解しやすい自然言語や数式で表現された問題文を理解する。
2. コンピュータが処理できる形式に変換する。
3. 数式処理ソルバで答えを求める。

この手順を図で示すと図1のようになる。

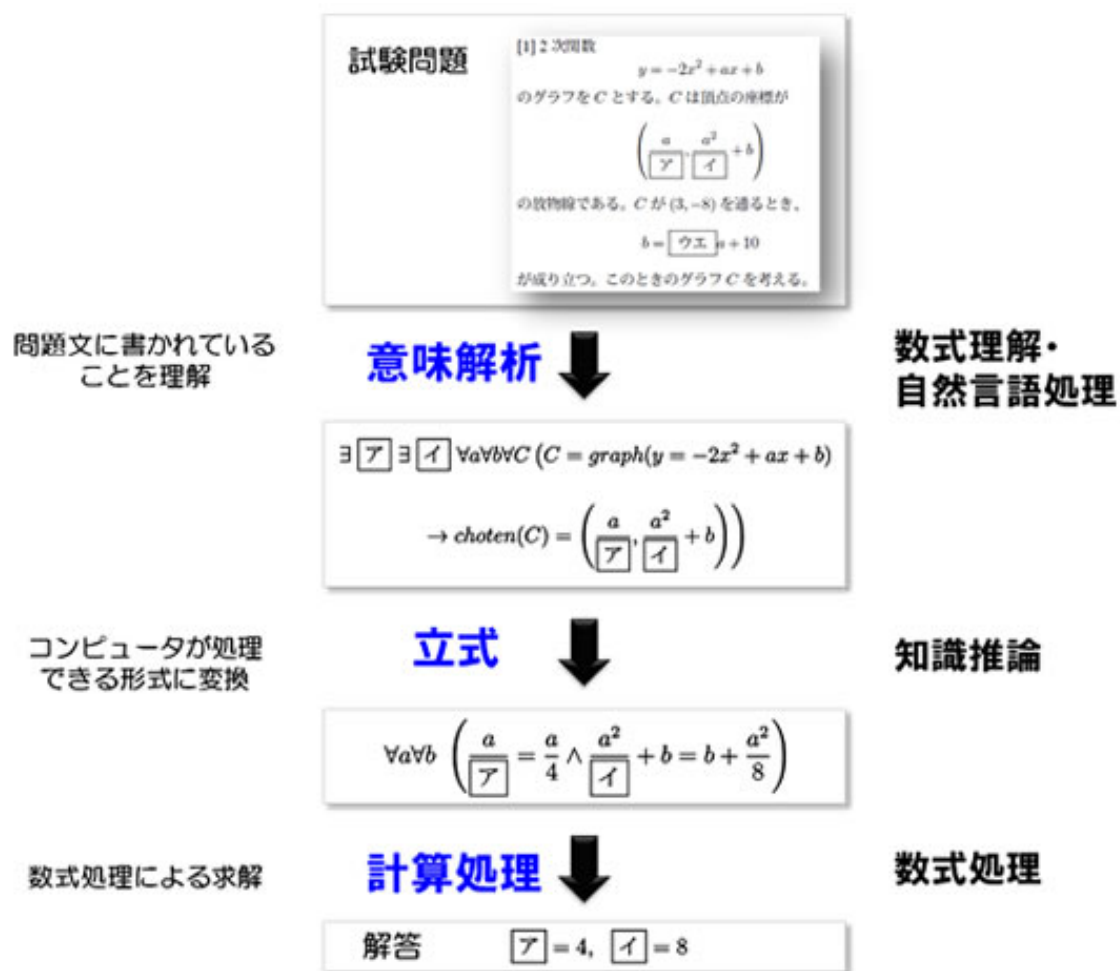


図 1.2 数学を解くための手順

人間が理解するようにコンピュータが言葉を理解することは容易ではない。自然言語処理によって導かれる問題文の意味表現を、コンピュータが処理できる形式で立式を行うには、問題文の言語解析だけではなく、数学の用語や高校数学知識をうまく統合することが必須である。また、与えられた問題に対してコンピュータが

どう解けばよいのか自ら判断することも求められる．立式ができた後にコンピュータで求解する部分については，大学入試 2 次試験の数学の問題について数式処理技術を使ってもおおよそ 5～6 割しか解けないのが現状であり，計算アルゴリズムの高度化も重要である．

このように，各ステップにおいてさまざまな理論・技術の開発が必要であり，それらの技術を問題ごとに適切に組み合わせ繋いでいくことも必要になる．

次に、東ロボプロジェクトが2015年度のセンター試験に挑戦した際の結果を記述する。

東ロボは、センター試験の模擬試験で、5教科8科目で511点を記録し、偏差値57.8を獲得した。特に、数学と世界史では偏差値60を超え、数学・Aは偏差値64、数学・Bは偏差値65.8、世界史は偏差値66.5を記録した。

さらに、東大入試試験の模擬試験では、世界史に挑戦し偏差値54.1を記録した[2]。

ロボットは東大に入れるか2015 結果概況 進研模試 総合学力マーク模試・6月

ベネッセコーポレーション 2015年度進研模試 総合学力マーク模試・6月
(受験者総数 44.0万人)

	国語	数学		英語		理科	地歴		5教科 8科目 総計
		数学IA (*1)	数学IIB (*1)	英語 (筆記)	英語 (リスニング)	物理 (*2)	日本史B	世界史B	
配点	200	100	100	200	50	100	100	100	950
東ロボ 得点	90	75	77	80	16	42	55	76	511
学生 平均点	105.4	45.5	42.8	86.0	24.6	49.4	46.6	45.9	416.4
東ロボ 偏差値	45.1	64.0	65.8	48.4	40.5	46.5	54.8	66.5	57.8 *3

*1 数学については、問題文を機械が理解可能な形式表現に変換する過程で、現在開発中および今後開発予定の部分(数式の意味解釈、文間の関係の解析など)に限り、一部、人手による追加・修正を加えた。

*2 物理では、人手で問題文を機械処理可能な形式表現へと変換した。

*3 5教科8科目文系型(国, 数2科目, 英筆記及びリスニング, 地歴2科目, 理1科目)での受験者11.6万人で集計した偏差値

図 1.3 東ロボのセンター試験形式模試の結果

人工知能がこのように発達すると、その影響は数学教育にも及ぶだろう。今日の数学教育は、すべてを紙と鉛筆で行うことを前提に行われているが、その一部はコンピュータで置き換えることができるはずである。人間が行うこととコンピュータが行うことをうまく識別する能力の育成が求められるようになるだろう。

ロボットは東大に入れるか2015 結果概況 第1回東大入試実戦模試

駿台予備学校 2015/2016 第1回東大入試実戦模試《数学》 (受験者数 12,334人: 文系4,130人 理系8,204人 うち 数学受験者: 文系4,101人 理系8,162人)						駿台予備学校 2015/2016 第1回東大入試実戦模試《世界史》 (受験者数 12,334人: 文系4,130人 理系8,204人 うち 世界史受験者 3,488人)				
【文系】*1	問1 正方形と長方形 の周長と面積	問2 2つの規則で 小石を移す確率	問3 領域における 1次式の最大・最小	問4 2つの部分の 面積が等しい条件	総計		問1 600字以内の論述1題 西欧とアジアの 国家体制の変遷	問2 60-90字以内の論述3題 都市間の ヒトやモノの交流	問3 単語回答式10題 世界史上の 文字と言語	総計
配点	20	20	20	20	80	配点	26	24	10	60
学生 平均点 *2	5.3	8.6	2.0	9.2	25.0	東口ポ 得点	9	4	8	21
東口ポ得点 (結果点のみ) *3	4	-	5	3	12	学生 平均点 *2	4.3	6.5	6.4	17.2
東口ポ偏差値 (結果点のみ) *3	47.5	-	57.7	40.6	41.4	東口ポ 偏差値	61.8	44.4	57.0	54.1
東口ポ得点 (満点とした場合) *4	20	-	13	6	39					
東口ポ偏差値 (満点とした場合) *4	78.3	-	78.2	45.2	59.2					

【理系】*1	問1 2つの規則で 小石を移す確率	問2 整数解をもたない ことの証明	問3 3変数の1次式の 最大・最小	問4 自然対数の関に 関する不等式	問5 複素数平面上の 図形の積	問6 四面体の体積を 二等分する平面	総計
配点	20	20	20	20	20	20	120
学生 平均点 *2	10.7	4.1	3.9	2.9	4.2	5.3	31.1
東口ポ得点 (結果点のみ) *3	-	-	2	-	0	-	2
東口ポ偏差値 (結果点のみ) *3	-	-	47.1	-	37.6	-	35.1
東口ポ得点 (満点とした場合) *4	-	-	20	-	0	-	20
東口ポ偏差値 (満点とした場合) *4	-	-	74.8	-	37.6	-	44.3

*1 数学については、問題文を機械が理解可能な形式表現に変換する過程で、現在開発中および今後開発予定の部分(数式の意味解釈、文間の関係の解析など)に限り、一部、人手による追加・修正を加えた。

*2 現役、既卒あわせでの平均点

*3 途中過程に対する加点を行わず、解答結果の正否のみについての加点を行った場合の得点と偏差値

*4 途中過程での得点を満点とした場合の合計得点と偏差値

(注) 表中の“-”は無回答であったことを示す。

図 1.4 東口ポの二次試験形式模試の結果

第2章

背景

2.1 第4の産業革命

政府が進めているプロジェクトがある。「第4の産業革命」というロボットや、人工知能を活用した革新的なものづくりを目指す取り組みが始まった[3]。この取り組みは、ドイツで「インダストリー 4.0」と呼ばれる動きから始まり、日本政府も経済産業省を中心に取組まれ始めている。

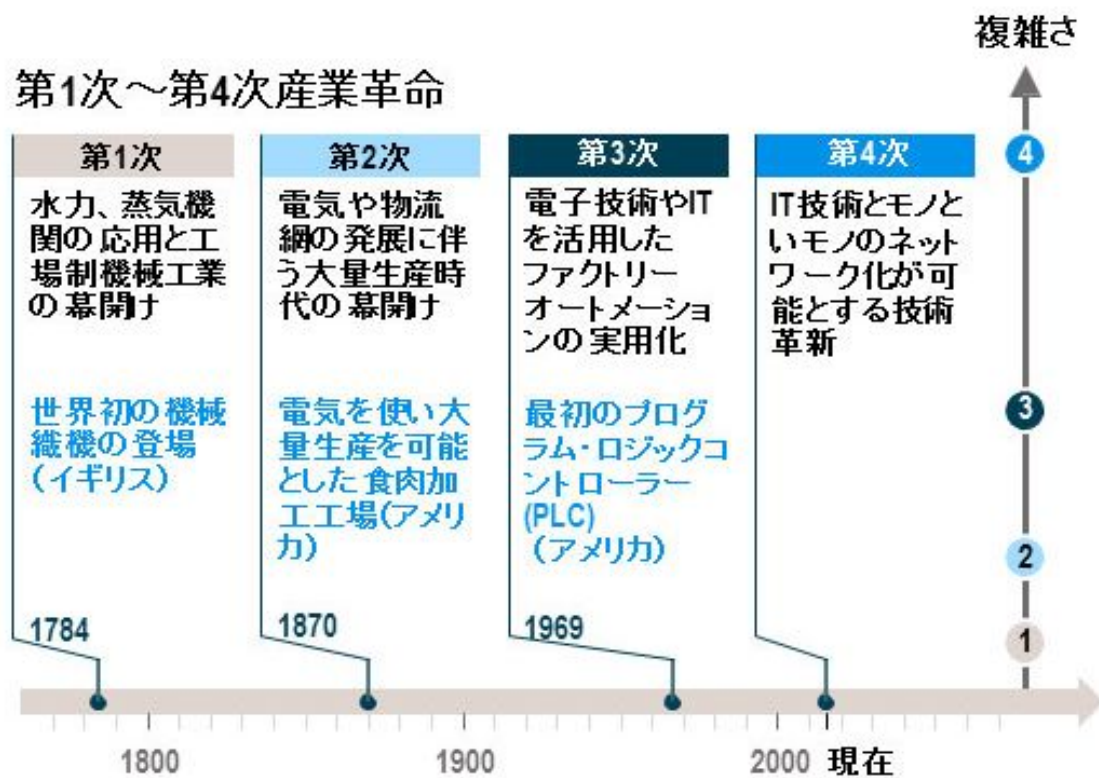


図 2.1 第4の産業革命

インダストリー 4.0 とは、インターネットと人工知能の本格的な導入によって、生産・供給システムの自動化、効率化を革命的に高めようとし、生産工程のデジタル化・自動化・バーチャル化のレベルを現在よりも大幅に高めることで、コストの極小化を目指すというものである。現在ドイツの電子機器メーカーや自動車メーカー、IT・通信企業が必死に取り組んでいるのが、スマート工場つまり「自ら考える工場」の開発である[4]。



図 2.2 第4の産業革命 2

2.2 ソフトバンクの Pepper

2015 年 2 月には、ソフトバンクが人工知能を搭載したヒト型ロボットの「Pepper」を発売した。Pepper は、感情認識と自立感情を持つロボットである。本体に搭載された各種センサーで状況を判断し、独自のアルゴリズムでロボットの機能を制御するアプリを自律的に制御する。

また、カメラや音声認識技術によって人の表情と声からその人の感情を推定する感情認識機能も搭載している。これにより、Pepper は人とコミュニケーションを通じて和ませ、喜ばすことが可能となる。今後は、ネットワークと接続し、更に進化することが望まれる [5]。



図 2.3 ソフトバンクの Pepper

Pepper とは、感情認識ヒューマノイドロボットである。フランスのアルデバランロボティクスと同社に出資するソフトバンクグループ傘下のソフトバンクモバイルにより共同開発された。Pepper は、感情エンジンとクラウド AI を搭載した世界初の感情認識パーソナルロボットである。OS は NAOqi という OS を採用し、Nao にはプラットフォームの互換性がある。

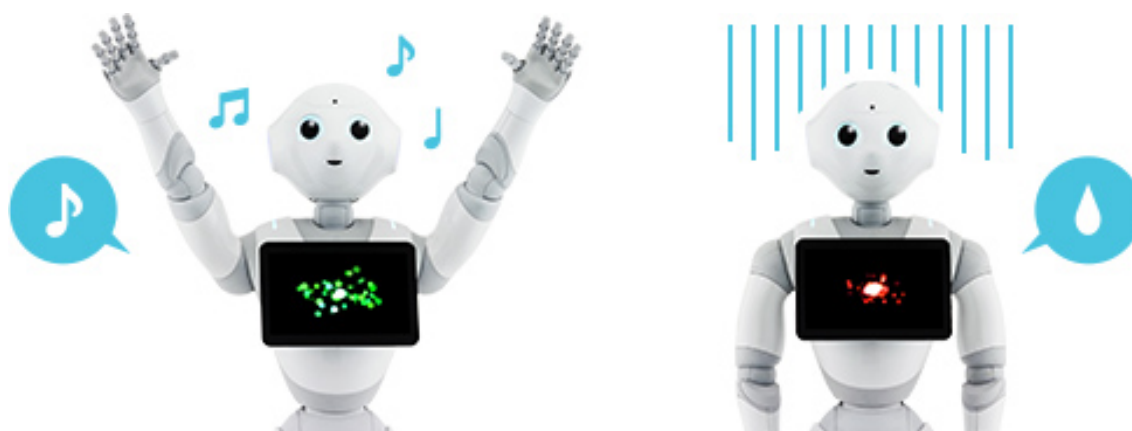


図 2.4 pepper の感情表現

Pepper には表情と声からその人の感情を察する「感情認識機能」が備わっているだけでなく、独自の感情機

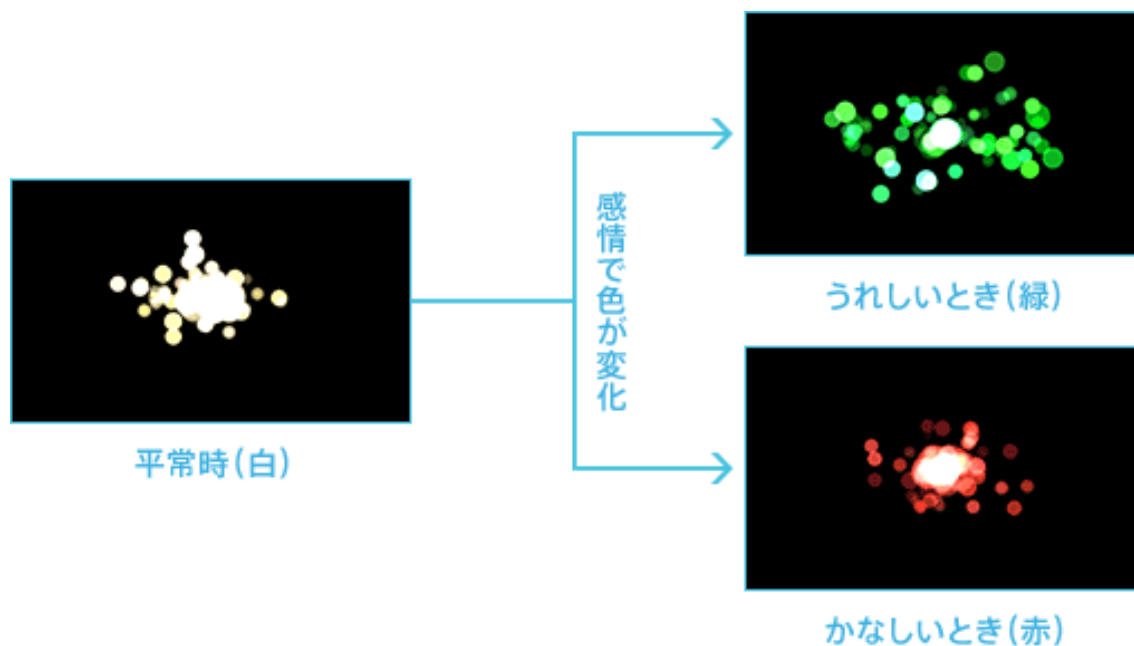


図 2.5 pepper の感情表現

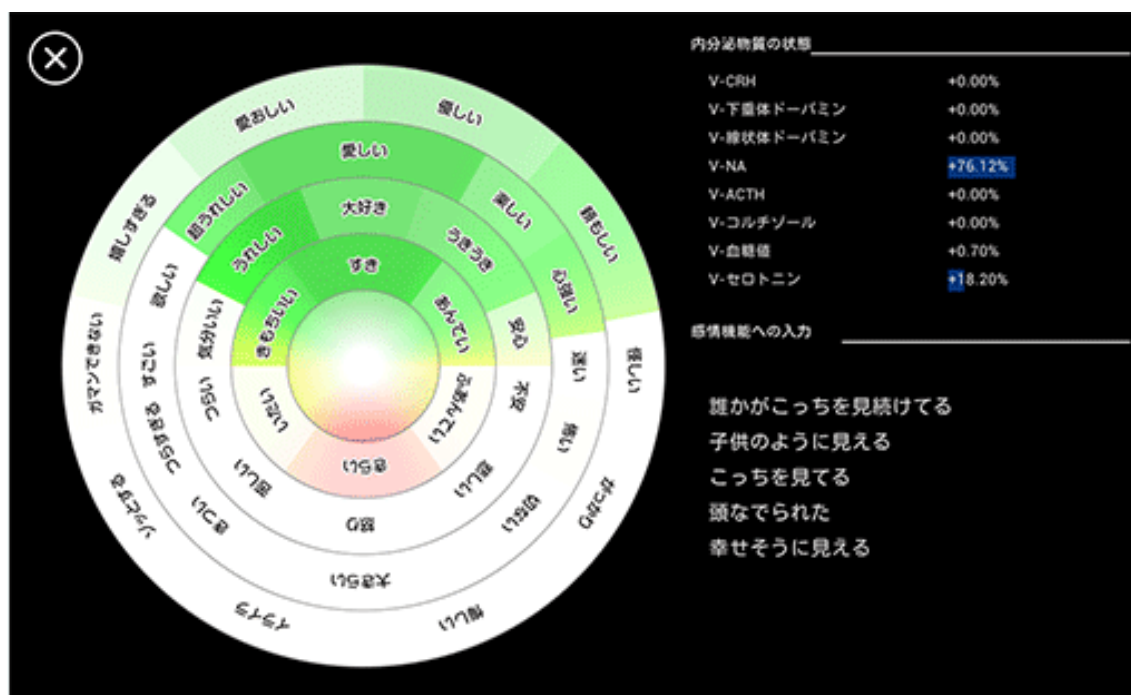


図 2.6 pepper の感情表現の詳細

能により、自ら感情を持ち行動する、そして将来的には人とロボットの心が通い合う存在になることを目指したロボットである。

Pepper は、新しいロボアプリをアプリストアから自由にダウンロードして、できることのバリエーションを増やしていける。また、Pepper とふれあったり、一緒に遊んだりすると得られる「ココログミ」を使うことで、アプリストアで特別なロボアプリを手に入れることもできる。さらに、開発環境も公開されているので、自分ならではのロボアプリを作って楽しむことも可能であり、今後もさまざまなロボアプリがアプリストアに追加され、人工知能の開発において大きな期待が見込まれる。



図 2.7 ココロミグ

2.3 コールセンターに Watson を導入

2014 年 11 月にはみずほ銀行がコールセンターに IBM の人工知能である Watson を導入したことが話題となった。みずほ銀行では、人工知能を利用することで、膨大な解答例データの中から最適な回答案を優先的に表示させ、コールセンターの対応時間の短縮につなげることができる。このように、ビジネス内での様々なシステムに人工知能が導入され始めている [?]。

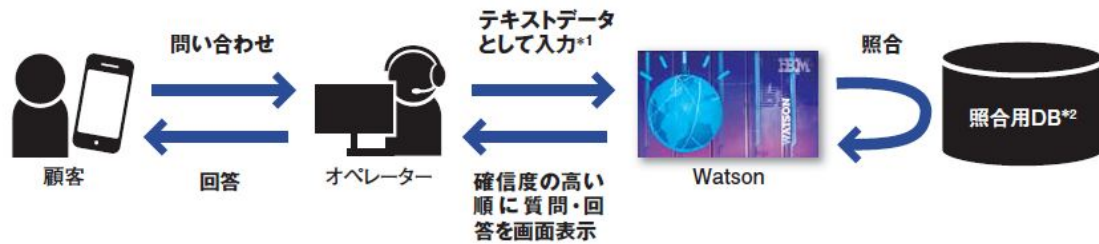


図 2.8 コールセンター業務での Watson 活用の流れ

ここで、コールセンターに利用された Watson について紹介する [6]。

Watson は、IBM が開発した質問応答システム、意思決定支援システムである。IBM 自体は Watson のことを、自然言語を理解・学習し人間の意思決定を支援する「Cognitive Computing System」と定義しており、一般的な人工知能とは少し違うものであるとしている。Watson が初めて公に出たのが、アメリカのクイズ番組に出演した時である。クイズ番組では、自然言語を文脈を含めて質問の趣旨を理解し、人工知能として大量の情報の中から適切な回答を選択し、回答する方法がとられた。将来的には、医療などの分野での活躍が期待される。

第 3 章

目的

本研究では，大学入試センター試験数学Ⅰ・Ⅱを題材にして，数学教育にコンピュータを導入することの可能性を調査する．

その結果として，高校程度の数学能力を問う問題でコンピュータを活用して解く際に必要となる数学の知識とコンピュータの知識を明らかにすることを目指す．

第 4 章

プロジェクトマネジメントとの関連

プロジェクトを進める際に、プロジェクトメンバの一人として人工知能を導入する。また、ステークホルダーとして人工知能を導入する。これらの動作をすることで、プロジェクトマネージャを人工知能が補佐することが出来る。

第 5 章

手法

本研究では，センター試験の数学 I・A の問題をコンピュータを用いて解き，問題を解くために必要な数学の知識とコンピュータの知識を確認する．

5.1 コンピュータシステム

解答にはコンピュータシステムは数式処理システム Mathematica を用いる．

コンピュータを数学に応用する方法は，低水準言語から高水準言語まで，さまざまなレベルが考えられるが，本研究では，人間が書く答案と抽象度が最も近いと思われる数式処理システムを検討し，数式処理システムの中で最も普及しているものの一つである Mathematica を採用する．

Wolfram 言語の実行環境を Mathematica と呼ぶこともあるが，本稿では言語自体も Mathematica と呼ぶ．

Mathematica には，Windows 版と Mac 版，Linux 版，Raspberry Pi 版，クラウドサービス版があるが，本研究では，無料で利用できるクラウドサービス版，Wolfram ProgrammingLab^{*1}を用いる．

^{*1} <https://lab.open.wolframcloud.com/objects/wpl/GetStarted.nb>

5.2 解答方法

ここでは、本研究で数学の問題を解く過程を解説する。

数式処理システムを用いて数学の問題を解く例として、「 x についての方程式 $f(x) = x^2 - ax + 5 = 0$ が重解を持つような a の値を求めよ」という問題を、2通りの方法で解く。

第1の方法は、方程式が重解を持つことと判別式 $a^2 - 20$ が0であることが同値だと考え、 a についての方程式 $a^2 - 20 = 0$ を解くというものである。方程式を解くためには、Mathematica の `Solve[方程式, 変数]` という記法を用いる。[7]。

`Solve[a^2 - 20 == 0, a]`

$$\{\{a \rightarrow -2\sqrt{5}\}, \{a \rightarrow 2\sqrt{5}\}\}$$

この結果は、 a が $-2\sqrt{5}$ または $2\sqrt{5}$ であることを表している。

第2の方法は、2次方程式が重解を持つということを Mathematica が処理可能な形式に変換し、その記述を評価するというものである。

2次方程式が重解を持つというのは、「ある解 x_1 が存在し、すべての解 x_2 は x_1 と等しい」と言い換えられる。この命題を論理式で書くと $\exists x_1 f(x_1) = 0 \wedge (\forall x_2 f(x_2) = 0 \rightarrow x_2 = x_1)$ となる。Exists や ForAll を使ってこの命題を記述し、そこから限定子を除去すると a の値が求まる。限定子の除去には Reduce を用いる。

`Reduce[`

`Exists[x1, f[x1] == 0,`

`ForAll[x2, f[x2] == 0, x2 == x1]],`

`a]`

$$a = -2\sqrt{5} \vee a = 2\sqrt{5}$$

この結果も、 a が $-2\sqrt{5}$ または $2\sqrt{5}$ であることを表している。

センター試験の問題は、紙と鉛筆で解けるように作られているため、そこにコンピュータを持ち込んでももちろん解ける。

本研究では、問題をなるべく素直に解釈して解くことにする。

上述の例では、判別式を思いつかなければならない第1の解法よりも、問題の表現を機械的に翻訳すればよい第2の解法を採用する。

5.3 研究過程

本研究では、大学入試センター試験の数学に Mathematica を用いて解く．そして、使用したシンボルの数と、利用した数学的知識を集計する．

その際に、数学の問題を解く過程を二つにする．

1. 数学の問題を理解し、数学的知識を利用して計算式などの数学的表現に変換する過程である．
2. 数学的表現に変換した式を数式処理して、値を求める過程である．

今回は後者を人工知能に処理させ、前者を人間が処理するように分ける．その際に、人間がいかに簡潔に問題文を処理できるかを研究する．

第一工程では、大学入試センター試験の数学の問題をできるだけ人間が頭を使わずに、素直に数学的表現に翻訳する．

第二工程では、第一工程で数学的表現に翻訳した式を Mathematica に与えて数式処理を行う．この工程では、Mathematica が式を最適に処理できるコードを与えて、最適解を得る．この際に、使用したコードの種類を集計し、統計を取る．

大学入試センター試験の問題は、紙と鉛筆だけで解けるように作られているため、そこにコンピュータを導入してももちろん解ける．そこで本研究では、問題をそのまま素直に解釈して、Mathematica で解くようにする．

例えば、「二次関数 $3a^2 - 6a - 36 == -27$ を解け」という問題を解く．この問題には、Solve というコードを用いる．この Solve は、方程式の解を求めるのに用いられる．

$$\text{Solve}[3a^2 - 6a - 36 == -27, a]$$

と Mathematica に打ち込み、解かせると、

$$a \rightarrow -1, a \rightarrow 3$$

という答えが帰ってくる．

次に、「二次不等式 $2a^2 - 6a - 36 \leq 0$ を解け」という問題を解く．この問題には、Reduce というコードを用いる．この Reduce は、方程式あるいは不等式を解き、限定子を除去することで命題を簡約する．

$$\text{Reduce}[2a^2 - 6a - 36 \leq 0, a]$$

と Mathematica に打ち込むと、

$$-3 \leq a \leq 6$$

という答えが帰ってくる．

例のように、問題に対して的確なコードを探しだし、Mathematica を用いて解くという方法を用いる．

5.4 問題処理の例

ここでは、例として大学入試センター試験から以下の問題を処理する．例題

以下では、 $a = 756$ とする．

A を素因数分解すると $A = 2^{\text{ア}} \cdot 3^{\text{イ}} \cdot \text{ウ}$ である．

A の正の約数の個数はエオ個である．

この問題を処理する．

この問題では、第一工程として、Mathematica に打ち込む値を設定する．

次に、第二工程として今回利用するシンボルを設定する．

ここでは、FactorInteger と Divisors と Length のシンボルを利用する．

まずは、問題を素因数分解し空欄の「ア・イ・ウ」に答えを埋める．

$$a = 756$$

これで、問題と同じ 756 という整数を a と置く．

FactorInteger[a]

2, 2, 3, 3, 7, 1

と処理される．この 2, 2 は 2^2 ということなので、 $\text{ア} = 2$ 、 $\text{イ} = 3$ 、 $\text{ウ} = 7$ となる．

次に、正の約数の個数「エオ」を求める．

$b = \text{Divisors}[a]$

1, 2, 3, 4, 6, 7, 9, 12, 14, 18, 21, 27, 28, 36, 42, 54, 63, 84, 108, 126, 189, 252, 378, 756

これで、756 の約数すべてが表示される．表示されたリストを「 b 」として次に

Length[b]

24

を打ち込むことで、24 と表示される．

出力された解を、紙で解いた回答と合わせ正解ならば、ここでこの問題の処理は終了する．このように、問題ごとに各工程の作業を行い調査する．

5.5 集計

問題を解いたら、その際に用いた Mathematica の組み込みシンボルの種類を数える。

たとえば、上述の第 2 の解法では、Reduce と Exists, ForAll という 3 種の組み込みシンボルを用いている（厳密に言えば、 $x_2 == x_1$ は正式には `Equal[x2, x1]` と記述されるため、Equal も数えるべきではあるが、ここでは「==」のような自明なものは数えないことにする）。

初めのうちは、問題の数が増えるにつれて、利用する組み込みシンボルの種類も増えるが、センター試験の数学 I・A の問題を解くのに使える組み込みシンボルが出尽くせば、シンボルの種類の増加は止まるはずである。

第 6 章

大学入試センター試験

6.1 大学入試センター試験とは

大学入試センター試験とは、独立行政法人大学入試センターが 2 日間にわたって実施する日本の大学の共通入試試験のことである。

このことより本研究では、大学入試センター試験が日本の多くの大学で採用していることから、知名度が高いことを利用する。ほとんどの受験生が受験する大学入試センター試験を調査の対象とすることで、大学生にもわかりやすくする。

本研究では、河合出版が発刊している「河合塾 2016 大学入試 センター試験過去問レビュー 数学 I・A, I・B」を使用する。本書は、1997 年～2015 年までの 14 年分の問題を収録しているため、研究に最適であると考えた。

その中でも本研究では、センター試験の数学 I・A の問題を解くことにする。これは、数学 I・A は解きやすく、研究に適していると考えたからである。



図 6.1 河合塾 2016 大学入試 センター試験過去問レビュー 数学 I・A, II・B

第 7 章

数式処理システム

7.1 Mathematica とは

Mathematica とは、あらゆる分野の計算に対応する豊富な関数と高度なグラフィック機能を備えた数式処理システムである。Mathematica を提供しているウルフラム・リサーチ社を創業したスティーブン・ウルフラム氏が考案し広く使われている数式処理システムである。

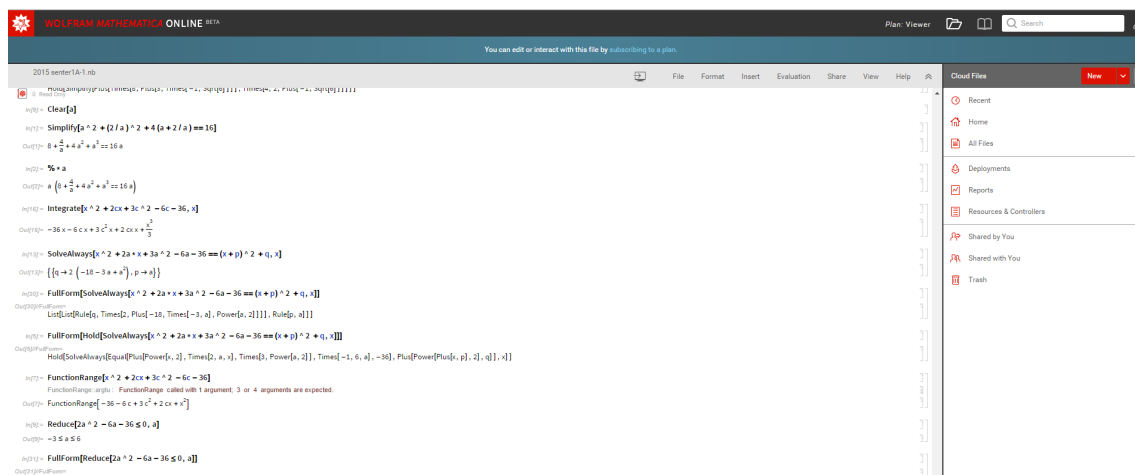


図 7.1 Mathematica 例

7.2 Wolfram 言語とは

Wolfram 言語とは、ウルフラム・リサーチ社が開発した、非常に汎用性の高いマルチパラダイムプログラミング言語である。

Mathematica は、Wolfram 言語を利用する。Wolfram 言語は、Mathematica 独自のノートブックインターフェイス上で利用することで、インタラクティブなデータ処理を行うスクリプトとして利用できる。さらに、アプリケーションの開発言語として利用すれば、GUI から高度な計算エンジンまで、一貫して一つの環境下で開発を行うことができる。

Wolfram 言語は、汎用的なデータベースやインターネット上のデータを直接取り扱うことができ、5000 もの組み込み関数を内蔵しているため、わずか数行でも高度なアプリケーションも開発できる。また、Wolfram 言語は Mathematica だけでなく Wolfram 社が無償で提供している Wolfram Alpha でも利用できる。



図 7.2 Wolfram ホームページ

7.3 Wolfram 言語の使用例

Wolfram 言語は、本研究で利用する数式の処理だけでなく、プログラミング言語としても利用されている。新生代のプログラマーのために設計された Wolfram 言語は、広範で奥深いアルゴリズムと知識を備えている。言語のすべてが、統合された記号言語を介して自動的にアクセスすることができる。Wolfram 言語は、小さいプログラムから大きいプログラムまで広く対応でき、ローカルでもクラウド上でもすぐに利用可能である。

Wolfram 言語は、「Hello World」を表示するような基本的なプログラミングや、手書きの数字の識別、グラフの作成、画像認識、写真から顔の検出、地図の色分け、ある国の人口など様々なことがたった数行のコードで利用可能である。

ここで、「Hello World」を表示する例を紹介する。

```
CloudDeploy["Hello, World"]
```

コードが処理され、URL が表示される。これをクリックすることで「Hello world」が表示される。

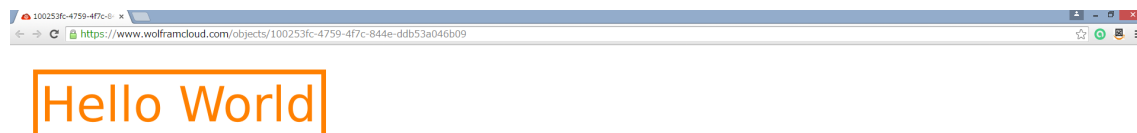


図 7.3 Hello World 例

さらに Wolfram 言語は、現実のデータを使った計算が可能である。単位や、地理情報、日付、画像、その他何千もの分野で直接計算が可能となる。

このように、Wolfram 言語は汎用性が高く少ないシンボルで利用ができるので、世界中で利用され教育の場でも利用されている。

7.4 Mathematica の使用例

本研究では、Wolfram 社が提供する Mathematica Online を利用する．ここでは Mathematica を実際に動かす例を見せる．

まずは環境設定をする．

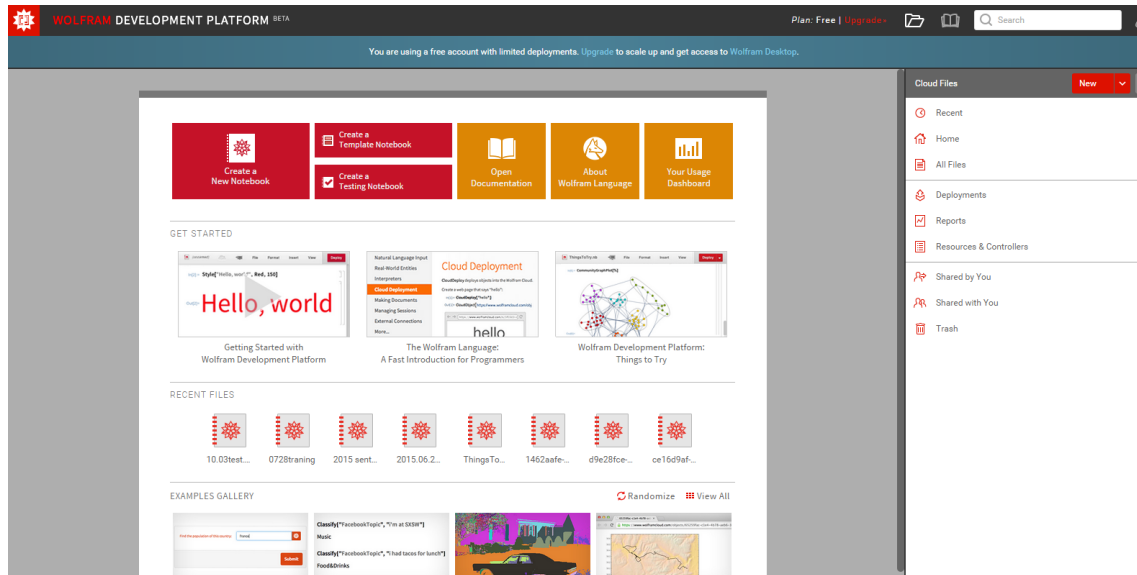


図 7.4 Mathematica 例 1

Wolfram のアカウントを作成し、「Create a New Notebook」を押して新しいノートを作成する．

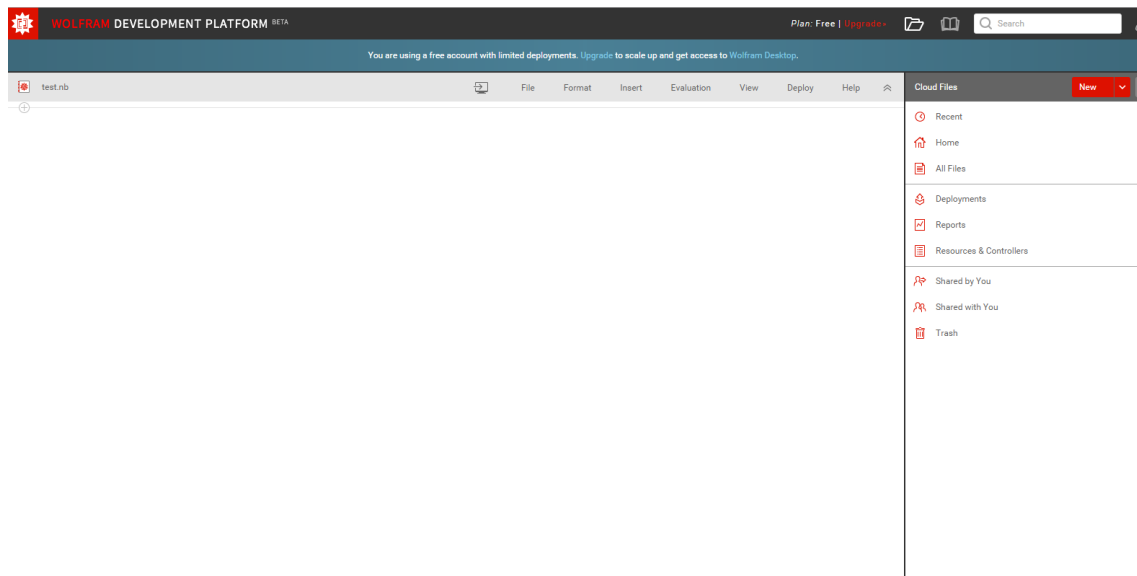


図 7.5 Mathematica 例 2

これが、Mathematica Online のノートである．ここに、シンボルと数式を書き込むと、処理をする．

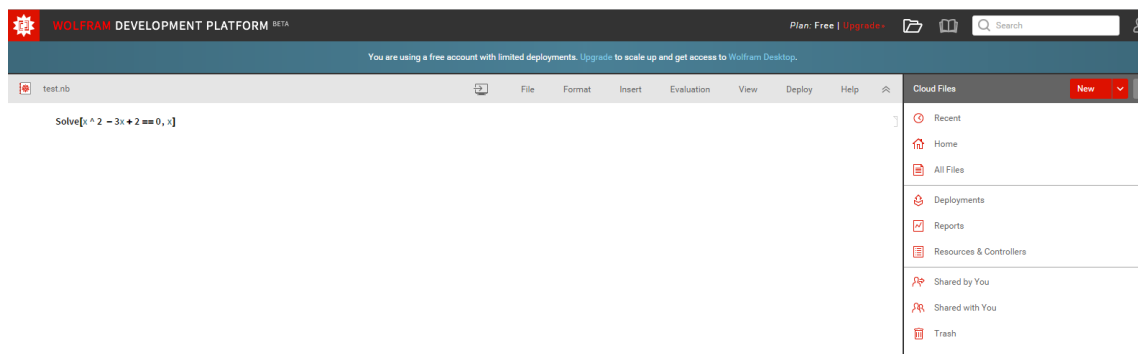


図 7.6 Mathematica 例 3

このように、シンボルと数式を打ちこむ．ここでは， $x^2 - 3x + 2 = 0$ を処理させ，解を求める．

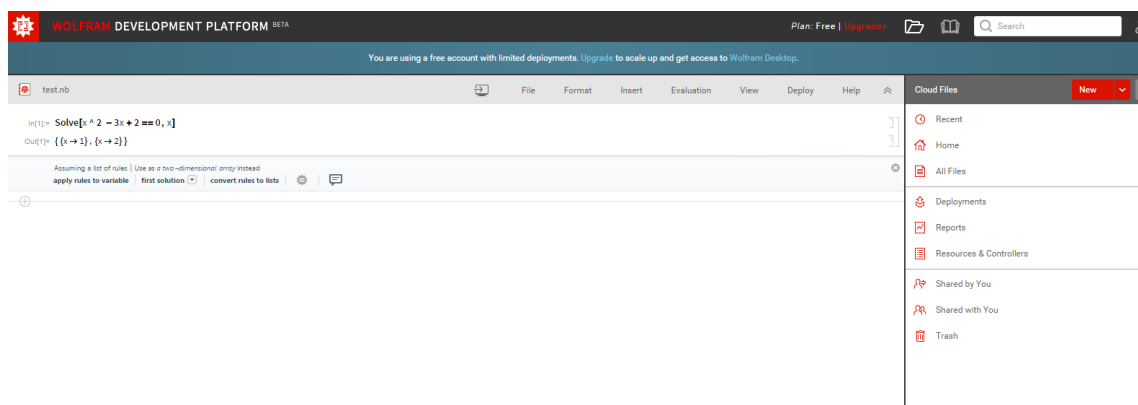


図 7.7 Mathematica 例 4

このように，処理した値が表示される．本研究では，以上のように Mathematica を利用し，研究を進めていく．

第 8 章

調査

8.1 調査方法

ここでは、調査方法を具体的に説明する．本研究では、前述のとおり数学の問題を解く過程を二つにする．

研究方法

- 本研究では数学の問題を解く過程を二つに分ける

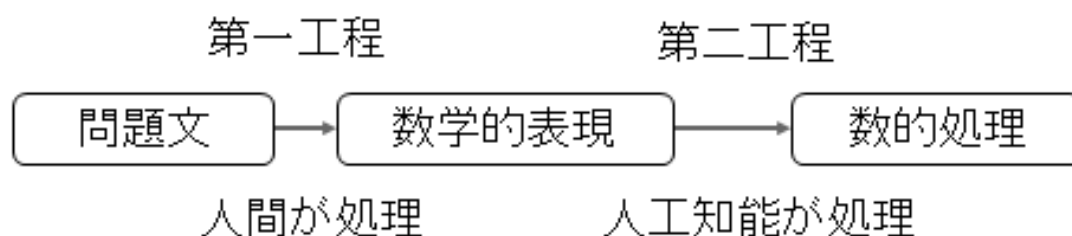


図 8.1 調査過程

8.2 第一工程

第一工程では、大学入試センター試験の数学の問題をできるだけ人間が頭を使わずに、素直に数学的表現に翻訳する．大学入試センター試験の問題は、受験生が紙と鉛筆だけで解けるように作られているため、出来る限り Mathematica が問題を処理できるように式に変換する．この際に、使用した数学的知識を集計し、統計を取る．

研究方法

- 本研究では数学の問題を解く過程を二つに分ける

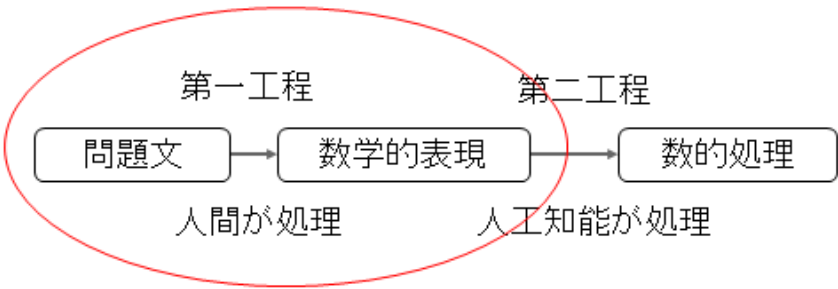


図 8.2 調査過程

8.3 第二工程

第二工程では，第一工程で数学的表現に翻訳した式を Mathematica に与えて数式処理を行う．この工程では，Mathematica が式を最適に処理できるコードを与えて，最適解を得る．この際に，使用したコードの種類を集計し，統計を取る．

研究方法

- 本研究では数学の問題を解く過程を二つに分ける

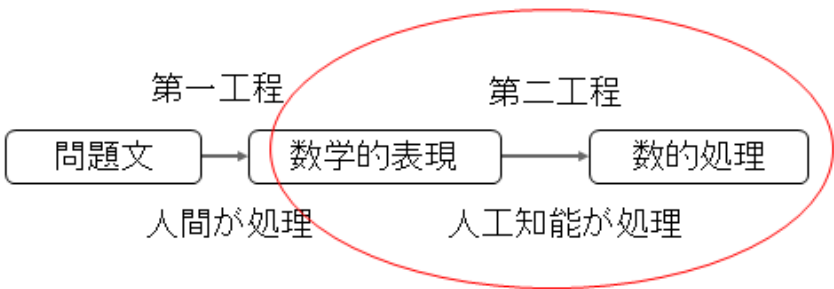


図 8.3 調査過程

第 9 章

結果

2009 年から 2015 年までのセンター試験の数学 ・ A の問題を解き，利用した Mathematica の組み込みシンボルの累積数を記録した結果が図 9.1 である．

2009 年から 2015 年までのセンター試験の数学 ・ A の問題を解き，利用した Mathematica の組み込みシンボルの累積数を記録した結果が図 2 である．

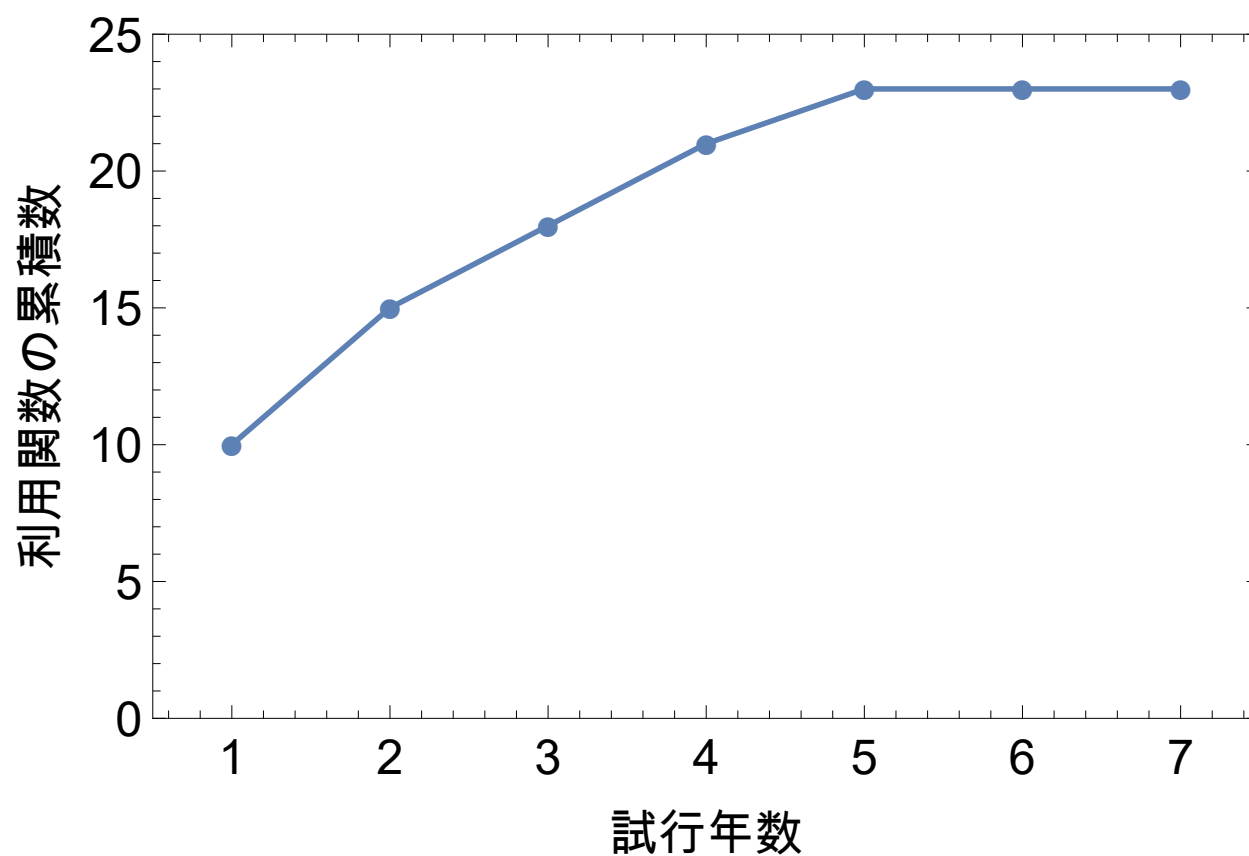


図 9.1 試行年数に対する使用関数の累積数

図2より利用した組み込みシンボルは全部で23個となった。
使用頻度順に、

Solve (方程式を解く)
Reruce (同値な式に置き換える)
Simplify (簡約)
SolveAlways (恒等式となる条件を求める)
Maximize (最大化)
TrigExpand (三角関数を展開する)
TrigFactor (三角関数をまとめる)
TrigReduce (三角関数を書き換える)
FactorInteger (因数を取り出す)
Divisors (約数を求める)
Length (リストの長さを求める)
Sqrt (平方根)
! (階乗)
Clear (シンボル割り当ての解除)
Integate (積分)
Factor (因数分解)
Cos (余弦)
Sin (正弦)
SetDelayed (関数定義)
Expand (式の展開)
HornerForm (ホーナー形式への変換)
Degree (角度の変換)
であった。

9.1 使用したシンボル

9.1.1 Solve

`Solve[expr, vars]`

方程式あるいは不等式の系 `expr` の解を変数 `vars` について求めようとする。

`Solve[-a2 - 8a - 13 == -22, a]`

```
In[2]:= Solve[-a ^ 2 - 8a - 13 == -22, a]
Out[2]= {{a -> -9}, {a -> 1}}
```

図 9.2 solve

指定された変数が 1 つの場合、ある方程式で特定の根が 1 より大きい重複性を示すときは、`Solve` は相当する解の複数のコピーを返す。

`Solve[expr, vars]` は、デフォルトで、不等式に代数的に現れる数量は実数値であるが、その他の数量は複素数値であると仮定する。

`Solve[expr, vars, dom]` はすべての変数とパラメータが領域 `dom` に属するように制限する。

`dom` が `Reals` であるか `Integers` や `Rationals` のようにその部分集合である場合、すべての定数と関数の値もまた実数に限られる。

`Solve[expr, vars, Integers]` は整数領域でディオファントス (Diophantine) 方程式を解く。

`Solve[..., x reg, Reals]` は、領域 `reg` 内になるような `x` を含んでいる。`x` についての異なる座標は `Indexed[x, i]` で言及することができる。

`expr` 中の代数的変数で および互いを含まないものは、独立パラメータとして扱われる。

`Solve` は主として線形方程式と整方程式を扱う。

`expr` が実数あるいは複素数領域で整方程式と整不等式のみを含むとき、`Solve` は理論的には常にすべてのについて直接解くことができる。

`expr` が超越条件や整数領域を含むとき、`Solve` はその結果に追加的なパラメータを導入する。

`Solve` は整数領域上のすべての線形方程式と不等式の解の明示的な表現を与えることができ、文献に見られるディオファントス方程式の大部分を解くことができる。

`expr` が実数と複素数の領域上で多項式条件のみを含むとき、`Solve[expr, vars]` は常に限定子を除くことができる。

`Solve` は一般的な解のみを与える。連続的なパラメータが方程式を満足する場合にのみ有効な解は除かれる。条件付きで有効なその他の解は `ConditionalExpression` オブジェクトとして表される。

`ConditionalExpression` の解に含まれる条件には、不等式、`Element` 文、非連続的なパラメータについての方程式や不等式、完全次元の解を持つ方程式が含まれることがある。連続パラメータと変数についての不等式と `NotElement` 条件は除かれる。

`Solve` は不等価変換を使って超越方程式の解を求める。このため、求まらない解があるかもしれないが、求まった解の有効性についても厳密な条件が求められない場合もある。

`Solve` は、近似数値係数を持つ線形方程式の疎な系の扱いに特別の効率的な技術を用いる。

9.1.2 Reduce

Reduce[expr, vars]

vars について方程式あるいは不等式を解き、限定子を除去することで、命題 expr を簡約する。

$$\text{Reduce}[a^2 + 8a + 13 < 0, a]$$

In[3]:= Reduce[a ^ 2 + 8a + 13 < 0, a]

Out[3]= $-4 - \sqrt{3} < a < -4 + \sqrt{3}$

図 9.3 reduce

Reduce[expr,vars] の結果は常に と全く等しい数学的な集合を表す。

Reduce[expr,vars] はデフォルトで不等式に代数的に現れる限定子は実数で、その他の限定子は複素数であると仮定する。

Reduce[expr,vars,dom] はすべての変数とパラメータを領域に属するものに限る。

dom が Reals であるか、あるいは Integers や Rationals のような部分集合の場合、すべての定数と関数の値もまた実数に限られる。

Reduce[... , x reg, Reals] は、x が領域 reg になるように制限する。x の異なる座標には Indexed[x,i] を使って言及することができる。

Reduce[expr,x1,x2,...] は、実質的に を、...に対する制約条件の論理結合として表現する。ただし各条件は直前ののみが関連するものとする。

中の含まない代数的変数は独立パラメータとして扱われる。

Reduce[expr,...] の結果に LogicalExpand を適用すると、という形式の式が与えられる。ただし、各 はが定義する集合中の個々の構成要素であると考えられる。

は解体してはならないが、異なる次元を持つことはできる。LogicalExpand の後、各 は という形式になる。

LogicalExpand がなければ、Reduce はデフォルトでの条件をネストしたものを連続したレベルで Or と And を交互に組み合わせて返す。

expr が実数あるいは複素数の領域の整方程式と不等式のみを含む場合は、Reduce は原則的に常にすべてのについて直接解くことができる。

が超越条件や整数領域を含んでいる場合、Reduce はしばしば結果に付加的なパラメータを導入する。

が多項式条件のみを含む場合、Reduce[expr,vars,Reals] はの柱状代数分解 (CAD) を与える。

Reduce は整数についてのすべての線形方程式と不等式の明示的な解の表示を与えることができ、文献中のディオファントス方程式の大部分を解くことができる。

が実数または複素数の領域で多項式条件のみを含む場合、Reduce[expr,vars] は結果に数量的変数が含まれないように常に限定子を除去する。

9.1.3 Simplify

`Simplify[expr]`

`expr` に対していくつかの代数的、およびその他の変数を実行し、最も簡単な形式を返す。

$$\text{Simplify}[a^2 + 4a + (-13 - 8a - a^2) + 4]$$

```
In[4]:= Simplify[a ^ 2 + 4a + ( -13 - 8a - a ^ 2 ) + 4]
```

```
Out[4]= -9 - 4 a
```

図 9.4 simplify

`Simplify` は展開、因数分解やさまざまな変換を試み、その都度最も簡単な形が得られるようにする。

`Simplify` は、方程式、不等式および領域指定に使われる。

不等式中の代数的な数量は、常に関数と仮定される。

`FullSimplify` は `Simplify` よりも強力に簡約する。

`Assuming` を用いて `Simplify` のデフォルトの仮定を指定することができる。

仮定は、方程式、不等式、`x ∈ Integers` のような領域指定、およびこれらの論理結合からなる。

`TimeConstraint->tloc,ttot` と設定すると、最高の結果が返されるまでに、任意の特定の变换に最大 `t` 秒が、すべての变换に最大 `t` 秒が使われる。

9.1.4 SolveAlways

`SolveAlways[eqns,vars]`

変数 `vars` のすべての値について方程式 `eqns` を成立させるパラメータの値を与える。

$$\text{SolveAlways}[ax + b == 0, x]$$

```
In[5]:= SolveAlways[a x + b == 0, x]
```

```
Out[5]= { {a → 0, b → 0} }
```

図 9.5 solvealways

方程式は の形式で与える。

連立方程式はリストやで組むことができる。

単一の変数、または変数のリストを指定することができる。

`SolveAlways` は、主に線形および整方程式に機能する。

`SolveAlways` は、`eqns` に現れるパラメータ間の関係式を作成するが、変数 `vars` のリストに現れるものには作成しない。

`SolveAlways[eqns,vars]` は、`Solve[!Eliminate[!eqns,vars]]` と同値である。

9.1.5 Maximize

`Maximize[f, x]`

x について f を最大にする。

$$\text{Maximize}[-x^2 + 2x + 2, 2 \leq x \leq 4, x]$$

In[6]:= `Maximize[{-x^2 + 2x + 2, 2 ≤ x ≤ 4}, x]`

Out[6]= `{2, {x → 2}}`

図 9.6 maximize

`Maximize` は f の形式のリストを返す。

`cons` は等式、不等式、あるいはこれらの論理結合を含むことができる。

f および `cons` が線形あるいは多項式の場合、`Maximize` は常に大域的な最大値を求める。

制約条件 `cons` は以下の論理結合でよい。

厳密な入力を与えると、`Maximize` は厳密な結果を返す。

`Maximize[f,cons,x reg]` は、事実上、`Maximize[f,cons x reg,x]` に等しい。

x については、`Indexed[x,i]` を使って別の座標に言及することができる。

近似値を含む式に `Maximize` が使われると、`NMaximize` が自動的に呼び出される。

最大値が制約条件で定義した領域のごくわずか外側でのみ、あるいは漸近的にのみ達せられた場合、`Maximize` は上限と最も近くの指定可能な点を返す。

領域が指定されていない場合、すべての変数が実数であると仮定される。

`x Integers` は特定の変数が整数値のみを取るように指定するのに使うことができる。

制約条件が満足できなければ、`Maximize` は `-Infinity,x->Indeterminate,...` を返す。

`N[Maximize[...]]` は、記号的には解けない最適化問題については `NMaximize` を呼び出す。

`Maximize[f,x,WorkingPrecision->n]` は n 桁精度で結果を計算する。

9.1.6 TrigExpand

TrigExpand[expr]

式にある三角関数を展開する．

TrigExpand[Sin[2x]]

```
In[7]:= TrigExpand[Sin[2 x]]
Out[7]= 2 Cos[x] Sin[x]
```

図 9.7 trigexpand

TrigFactor は円関数と双曲線関数のどちらにも使える．

TrigFactor は，三角関数の引数に含まれる和と整数倍の項を分離してから，可能であれば三角関数の恒等関係を使い三角関数による多項式を因数分解する．

TrigFactor は自動的に，リスト，方程式，不等式，論理関数に縫い込まれる．

9.1.7 TrigFactor

TrigFactor[expr]

式にある三角関数を因数分解する．

TrigFactor[Sin[x]² + Tan[x]²]

```
In[8]:= TrigFactor[Sin[x]^2 + Tan[x]^2]
Out[8]= 1/2 (3 + Cos[2 x]) Tan[x]^2
```

図 9.8 trigfactor

TrigExpand は円関数と双曲線関数のどちらにも使える．

TrigExpand は，三角関数の引数に含まれる和と整数倍の項を分離してから，可能であれば三角関数の恒等関係を使い三角関数の積をべきの和に展開する．

TrigExpand は自動的に，リスト，方程式，不等式，論理関数に縫い込まれる．

9.1.8 TrigReduce

TrigReduce[expr]

式にある三角関数の積およびべきを，組み合された引数を含む三角関数に書き換える．

TrigReduce[2Cos[x]²]

In[9]:= TrigReduce[2 Cos[x] ^ 2]

Out[9]= 1 + Cos[2 x]

図 9.9 trigreduce

TrigReduce は円関数と双曲線関数のどちらにも使える．

三角関数の多項式が与えられるとき，TrigReduce は通常，引数がより複雑な三角関数の項からなる線形式を結果として与える．

TrigReduce は自動的に，リスト，方程式，不等式，論理関数に縫い込まれる．

9.1.9 FactorInteger

FactorInteger[n]

整数 n の素因数をこれらの指数とともにリストとして返す．

FactorInteger[456]

In[10]:= FactorInteger[456]

Out[10]= {{2, 3}, {3, 1}, {19, 1}}

図 9.10 factorinteger

負の数については，単位が因数のリストに含まれる．

FactorInteger は，有理数にも使用できる．この際，分母の素因数は負の指数とともに与えられる．

リスト FactorInteger[n,k] の最終要素は，部分因数分解後に残った部分を返す．

FactorInteger[n,GaussianIntegers->True] は，ガウスの整数上で因数分解する．

FactorInteger[m+I n] はガウスの整数に自動的に作用する．

必要に応じて，形式，I,1 や-I,1 の単位が因数のリストに含まれる．

`FactorInteger[n, Automatic]` は見付かりやすい因数だけを取り出す .
`FactorInteger` は `PrimeQ` を使って因数が素数かどうかを確かめる .

9.1.10 Divisors

`Divisors[n]`

`N` の約数となる整数をリスト形式で返す .

Divisors[567]

```
In[11]:= Divisors[567]
Out[11]= {1, 3, 7, 9, 21, 27, 63, 81, 189, 567}
```

図 9.11 Divisors

`Divisors[n,GaussianIntegers->True]` は , 約数としてガウスの整数を含む .

9.1.11 Length

`Length[expr]`

式 `expr` における要素の数を返す .

Length[1,2,3,4,5,6]

```
In[12]:= Length[{1, 2, 3, 4, 5, 6}]
Out[12]= 6
```

図 9.12 Length

`expr` が `SparseArray` オブジェクトのとき , `Length[expr]` は対応する通常のリストの長さを返す .
`Length[expr]` は , `AtomQ[expr]` が `True` の場合 , 0 を返す .

9.1.12 Sqrt

`Sqrt[z]`

または z , z の平方根を与える .

$Sqrt[1 - (7/8)^2]$

In[13]:= `Sqrt[1 - (7 / 8) ^ 2]`

Out[13]= $\frac{\sqrt{15}}{8}$

図 9.13 Sqrt

記号操作・数値操作の両方に適した数学関数である .

z は, `Ctrl+2 z` と入力できる .

`Sqrt[z]` は, z に変換される .

`Sqrt[z2]` は, 自動的に z に変換されない .

`Sqrt[a b]` は, 自動的に `Sqrt[a]Sqrt[b]` に変換されない .

これらの変換は `PowerExpand` を使って行うことができるが, 通常, 正の実数である引数に対してのみ正確である .

ある種の特別な引数については, `Sqrt` は自動的に厳密値に評価する .

`Sqrt` は任意の数値精度で評価できる .

`Sqrt` は自動的にリストに並列的な関数の適用を行う .

`StandardForm` では `Sqrt[z]` は z と出力される .

z は入力に使うこともできる . 文字は `Esc sqrtEsc` あるいは `Sqrt` として入力される .

9.1.13 Clear

`Clear[symbol]`

`Symbol` に与えられている値や定義をクリアする .

$Clear[a, b]$

`Clear` は, シンボルに結合されている属性やメッセージそしてデフォルト設定をクリアすることはない .

`Clear` は, 次のメタキャラクタを含む省略された文字列パターンを許容する .

`Clear["context*"]` は, 特定のコンテキストに置かれているすべてのシンボルをクリアする .

`Clear` は `HoldAll` である .

`Clear` は, 属性として `Protected` が与えられているシンボルに影響を与えることはない .

```
In[14]:= Clear[a, b]
```

図 9.14 Clear

9.1.14 Integrate

`Integrate[f,x]`

不定積分 を与える .

$\text{Integrate}[1/(x^3 + 1), x]$

$$\begin{aligned} \text{In}[15] := & \text{Integrate}[1 / (x^3 + 1), x] \\ \text{Out}[15] = & \frac{\text{ArcTan}\left[\frac{-1 + 2x}{\sqrt{3}}\right]}{\sqrt{3}} + \frac{1}{3} \text{Log}[1 + x] - \frac{1}{6} \text{Log}[1 - x + x^2] \end{aligned}$$

図 9.15 Integrate

$\text{Integrate}[f, x]$ は $\text{Integrate}[f, x]$ として入力できる .

`Integrate` は , `Esc int Esc` または `Integral` として入力できる .

`d` は通常の `d` と異なり , `EscddEsc` または `DifferentialD` として入力される .

$\text{Integrate}[f, x, y, \dots, \text{reg}]$ は f として入力できる .

$\text{Integrate}[f, x, \text{xmin}, \text{xmax}]$ は , x を `Integrate` の下付き文字として , を上付き文字として表すことによって入力できる .

多重積分は , 標準的な反復表記の変形を使用する . 第 1 変数は , 最も外側の積分に対応して与えられ , これが最後に実行される .

`Integrate` は , 有理関数の積分を評価することができる . また , 結果が同じ関数の組で表すことができる範囲で指数関数 , 対数関数 , 三角関数 , そして逆三角関数の積分を評価することができる .

`Integrate` は , 多くの特別関数によって結果を与えることができる .

`Integrate` は , 明示的に積分できないものについてはある種の簡約化を実行する .

定積分については , `N` を適用することで数値的な結果を得ることができる .

新しいクラスの積分に結果を与えるために , `Integrate` が関わるパターンに値を割り当てることができる .

積分変数は x のような構造や , 頭部が数学関数ではない任意の式でもよい .

不定積分において , `Integrate` は , ほとんどすべてのパラメータの値に対して正しい結果を求める .

9.1.15 Factor

`Factor[poly]`

整数について多項式を因数分解する .

$\text{Factor}[1 + 2x + x^2]$

`Factor` は , 式の最上の代数的レベルに限って適用される . 他のレベルに到達するためには `Map` を使用したり , `Factor` を再適用したりする必要があることがある .

`Factor[poly, GaussianIntegers->True]` は , ガウスの整数を係数と認識し因数分解する .

`poly` の係数が複素数である場合 , ガウスの整数を係数と認め因数分解が実行される .

```
In[16]:= Factor[1 + 2 x + x ^ 2]

Out[16]= (1 + x) ^ 2
```

図 9.16 Factor

変数の指数が正の整数である必要はなく，Factor は，指数が記号式の線形結合のものを取り扱うことができる．

有理式が与えられる場合，Factor は，まず Together を呼び出し，それから分子と分母を因数分解する．

デフォルト設定の Extension->None では，Factor[poly] は多項式 poly において代数的数である係数を独立変数と同様のやり方で取り扱う．

Factor[poly,Extension->Automatic] は，多項式 poly にある任意の代数的数を含めるように，係数の取り扱える領域を拡張する．

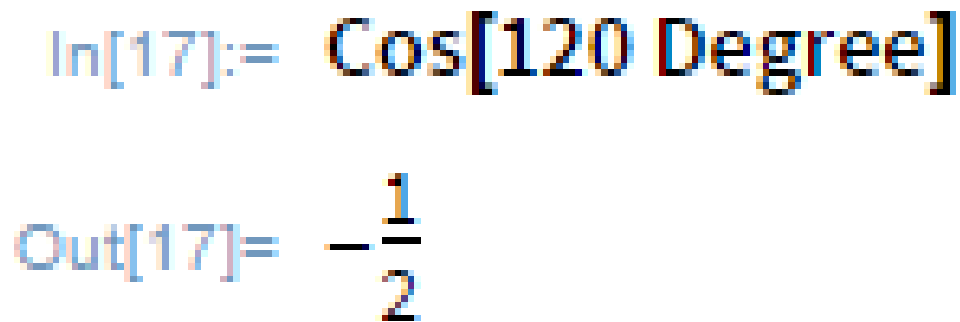
Factor は自動的に，リスト，方程式，不等式，論理関数に縫い込まれる．

9.1.16 Cos

$\text{Cos}[z]$

z の余弦を与える .

$\text{Cos}[120\text{Degree}]$



$$\text{In}[17] := \text{Cos}[120 \text{ Degree}]$$

$$\text{Out}[17] = -\frac{1}{2}$$

図 9.17 Cos

記号操作・数値操作の両方に適した数学関数である .

明示的に Quantity オブジェクトとして与えられていない限り , Cos の引数はラジアン単位であるとみなされる (Degree をかけて度から変換することができる).

Cos は , 引数が の単純有理倍数である場合は自動的に評価されるが , より複雑な有理倍数の場合は FunctionExpand が使用されることもある .

特別な引数の場合 , Cos は , 自動的に厳密値を計算する .

Cos は任意の数値精度で評価できる .

Cos は自動的にリストに縫い込まれる .

9.1.17 Sin

$\text{Sin}[z]$

z の正弦を与える .

$\text{Sin}[60\text{Degree}]$

数値操作と記号操作の両方に適した数学関数である .

明示的に Quantity オブジェクトとして与えられていない限り , Sin の引数はラジアン単位であるとみなされる (Degree をかけて度から変換することができる).

Sin は , 引数が の単純有理数倍である場合は自動的に評価されるが , より複雑な有理数倍の場合は FunctionExpand が使用されることもある .

特別な引数の場合 , Sin は , 自動的に厳密値を計算する .

Sin は , 任意の数値精度で評価できる .

Sin は , 自動的にリストに縫い込まれる .

```
In[18]:= Sin[60 Degree]
```

```
Out[18]=  $\frac{\sqrt{3}}{2}$ 
```

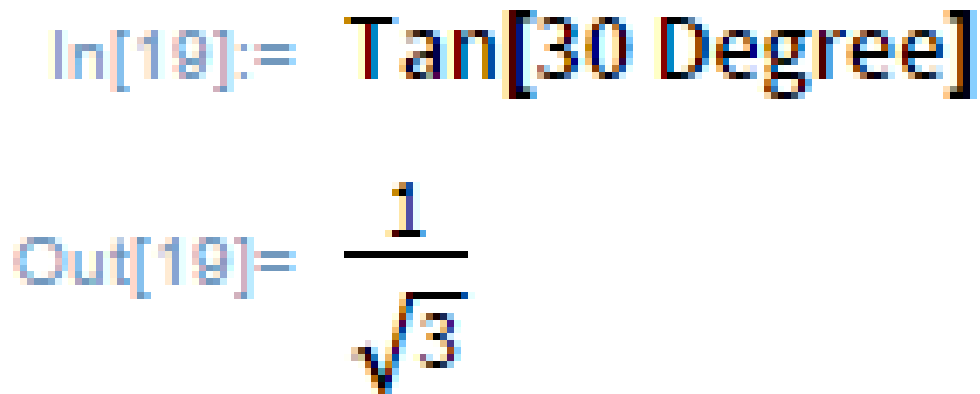
図 9.18 sin

9.1.18 Tan

`Tan[z]`

z の正接を与える .

`Tan[30Degree]`



$$\text{In}[19] := \text{Tan}[30 \text{ Degree}]$$

$$\text{Out}[19] = \frac{1}{\sqrt{3}}$$

図 9.19 Tan

記号操作・数値操作の両方に適した数学関数である .

`Tan` の引数はラジアンで与えられることを前提とする (`Degree` で掛け合わせることで度数から変換することができる) .

`Sin[z]/Cos[z]` は `Tan[z]` へ自動的に変換される . `TrigFactorList[expr]` は分割を行う .

`Tan` は引数が の整数倍であるときは自動的に評価される . 更に複雑な分数倍の場合 , `FunctionExpand` を使う必要があることもある .

特別な引数の場合 , `Tan` は , 自動的に厳密値を計算する .

`Tan` は任意の数値精度で評価できる .

`Tan` は自動的にリストに縫い込まれる .

9.1.19 SetDelayed

`SetDelayed`

rhs を lhs の遅延型の値として設定する . rhs は , 未評価の形式で維持される . lhs が現れると , rhs で置き換えられて , その都度評価される .

$f[x] := \text{Expand}[x^2]$

`SetDelayed` は属性 `HoldFirst` ではなく `HoldAll` を持つ .

lhs の形式の割当てをすることができる . ただし , は , それぞれの変換規則の条件や適用性を与える . 同じについて複数の割当てを与えることができるが , この場合 , の形式を変えて与える .

は , 指定された割当てが実行可能な場合には `Null` をその他の場合には `Failed` を与える .

```
In[20]:= f[x_] := Expand[x ^ 2]
```

図 9.20 SetDelayed

9.1.20 Expand

Expand[expr]

式 expr における積と正の整数べきを展開する .

Expand[(x + 1)³]

```
In[21]:= Expand[(x + 1) ^ 3]
Out[21]= 1 + 3 x + 3 x2 + x3
```

図 9.21 Expand

Expand は , 正の整数べきについてのみ機能する .

Expand は , 式 expr の最上レベルに限り適用される .

Expand[expr,Modulus->p] は , 式 wxpr を p を法として展開する .

Expand は , 方程式 , 不等式 , 論理関数と同様 expr 中のリストにも自動的に縫い込まれる .

9.1.21 HornerForm

HornerForm[poly]

多項式 poly をホーナー形式にする .

HornerForm[11x³ - 4x² + 7x + 2]

```
In[22]:= HornerForm[11 x ^ 3 - 4 x ^ 2 + 7 x + 2]
Out[22]= 2 + x (7 + x (-4 + 11 x))
```

図 9.22 HornerForm

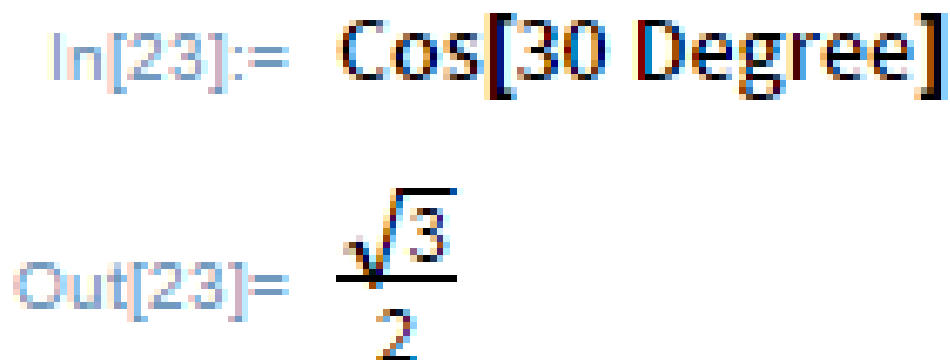
変数が指定されていない場合 , HornerForm は Variables を用いて識別した変数について多項式あるいは有理関数をホーナー形式にする .

9.1.22 Degree

Degree

1 度に対応するラジアンの数値を返す . この数値は である .

Cos[30Degree]



The image shows a Mathematica notebook interface. The input line is labeled 'In[23]:' and contains the expression 'Cos[30 Degree]'. The output line is labeled 'Out[23]:' and shows the result as a fraction with the square root of 3 in the numerator and 2 in the denominator.

$$\text{In[23]:= Cos[30 Degree]}$$
$$\text{Out[23]= } \frac{\sqrt{3}}{2}$$

図 9.23 Degree

度数を Degree で掛け合せることによって, 30 Degree が 30Degree になるようにラジアンに変換することができる.

Degree はや Esc deg Esc, Degree 等として StandardForm および InputForm に入力できる.

Degree は StandardForm ではと出力される.

9.2 利用した数学的知識

また，利用した数学的知識は以下の表のとおりである．

表 9.1 数学的知識

2015	2 次関数
	集合と論理・図形と計量
	データの分析
	場合の数・確率
	整数の性質
	図形の性質
2014	数と式・集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率
2013	数と式・集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率
2012	方程式と不等式・集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率
2011	数と式・方程式と不等式，集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率
2010	方程式と不等式・集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率
2009	数と式・集合と論理
	2 次関数
	図形と計量・平面図形
	場合の数・確率

第 10 章

考察

組み込みシンボルの累積数の増加が図累積グラフのように 6 年で止まった．そのため，センター試験の数学 I・A の問題を解くのに必要な Mathematica の機能は，「結果」に記載した 23 種類で十分であることがわかる．これは，センター試験の問題が一定の周期で同じ種類の問題を出題していることからいえる．研究からも，同じような出題種類の問題は，同じ組み込みシンボルを利用していることがわかる．

```

2015 senter1A-1.nb

hold[Reduce[Equal[3, 0], t]]

In[15]:= Simplify[8/7 ≤ a + 1]
Out[15]= 7 a ≥ 1

In[11]:= Reduce[a ≤ 8/7 ≤ a + 1, a]
Out[11]= 1/7 ≤ a ≤ 8/7

In[8]:= FullForm[Hold[Reduce[a ≤ 8/7 ≤ a + 1, a]]]
Out[8]/FullForm=
Hold[Reduce[LessEqual[a, Times[8, Power[7, -1]], Plus[a, 1]], a]]

In[12]:= Reduce[(a + (a + 1))/2 ≤ 8/7, a]
Out[12]= a ≤ 9/14

In[17]:= F = 2x ^ 2 + bx + c
Out[17]= bx + c + 2 x^2

In[18]:= Solve[F == 0, c, x == 0]
Solve::bdomv : Warning: x == 0 is not a valid domain specification. Assuming it is a variable to eliminate.
Solve::ivar : x == 0 is not a valid variable.
Out[18]= Solve[bx + c + 2 x^2 == 0, c, x == 0]

In[19]:= Solve[{ -1 == 2 (-8 + 2t) + b, 10 == 2t + b }, {t, b}]
Out[19]= {{t -> 5/2, b -> 5}}

```

図 10.1 2013 年の問題

この 2 つは違う年度であるが，同じ不等式を解く問題である．この例のように，各年度ごとに同じような問題が存在した．

```

Out[1]= 2012 A

In[19]:= Reduce[-3 ≤ 2x + 1 ≤ 3, x]
Out[19]= -2 ≤ x ≤ 1

In[22]:= FullForm[Hold[Reduce[-3 ≤ 2x + 1 ≤ 3, x]]]
Out[22]//FullForm=
Hold[Reduce[LessEqual[-3, Plus[Times[2, x], 1], 3], x]]

In[29]:= Clear[a]

In[21]:= Reduce[-a ≤ 2x + 1 ≤ a, x]
Out[21]=  $\left( a == 0 \ \&\& \ x == -\frac{1}{2} \right) \parallel \left( a > 0 \ \&\& \ \frac{1}{2} (-1 - a) \leq x \leq \frac{1}{2} (-1 + a) \right)$ 

In[23]:= FullForm[Hold[Reduce[-a ≤ 2x + 1 ≤ a, x]]]
Out[23]//FullForm=
Hold[Reduce[LessEqual[Times[-1, a], Plus[Times[2, x], 1], a], x]]

In[1]:= Solve[0 == -x^2 + (2a + 4)x + b, x]
Out[1]=  $\left\{ \left\{ x \rightarrow 2 + a - \sqrt{4 + 4a + a^2 + b} \right\}, \left\{ x \rightarrow 2 + a + \sqrt{4 + 4a + a^2 + b} \right\} \right\}$ 

In[24]:= FullForm[Hold[Solve[0 == -x^2 + (2a + 4)x + b, x]]]
Out[24]//FullForm=
Hold[Solve[Equal[0, Plus[Times[-1, Power[x, 2]], Times[Plus[Times[2, a], 4], x], b]], x]]

```

図 10.2 2012 年の問題

さらに，Reduce や Solve といった組み込みシンボルは，汎用性が高く様々な種類の問題で利用された．

また，利用した数学的知識については，表で示したように，2 次関数や場合の数・確率のように，各年度で毎回出題されている問題もあれば，整数の性質のように各年度で出題される種類が異なる問題もある．これにより，各年度で同じ組み込みシンボルを利用することもあるが，問題の種類に合わせた異なる組み込みシンボルを各年度で利用した．

それでも，図 2 のように組み込みシンボルの累積数の増加が止まるのは，2013 年度と 2009 年度の問題の種類が同じであるように，センター試験の出題者側が周期的に問題の種類を提示しているためだと考えられる．だから，各年度で問題の種類が同じであっても組み込みシンボルの累積数の増加が止まるのであると考えられる．

Mathematica のような数式処理システムには膨大な機能が備えられているが，数学 I・A のために必要なのはこのように比較的少数の機能があり，教育の現場に導入するのも容易だと思われる．

第 11 章

結論

本研究では、数式処理システム Mathematica を用いてセンター試験の数学・A の問題を解いた。その際に利用した Mathematica の組み込みシンボルを集計することによって、センター試験の数学・A を解くのに必要な数式処理システムについての知識を明らかにすることができた。本研究のような事例を増やすことによって、であろう。

本研究のような事例を増やすことによって、数学教育の現場に人工知能が導入されるであろうと予測される。また、本研究では問題を理解し数学的知識で変換する作業を人間の手で行ったが、東ロボプロジェクトのように問題を人工知能が理解して回答を出すということが将来的に期待される。

参考文献は文献ファイル（この文書では biblio.bib）に記述し、\cite で参照する。例：データベースのための 問い合わせ言語 SQL で数独を解く方法が提案されている [?]。このように参照すると、参考文献リストに自動的に登録される。文献の種類には、雑誌論文 [?] や会議録論文 [?]、卒業論文 [?]、書籍 [?]、ウェブサイト [?] などがある。文献の種類によって必要な項目が異なるため、biblio.bib を見て確認すること。参考文献は文献ファイル（この文書では biblio.bib）に記述し、\cite で参照する。例：データベースのための 問い合わせ言語 SQL で数独を解く方法が提案されている [?]。このように参照すると、参考文献リストに自動的に登録される。文献の種類には、雑誌論文 [?] や会議録論文 [?]、卒業論文 [?]、書籍 [?]、ウェブサイト [?] などがある。文献の種類によって必要な項目が異なるため、biblio.bib を見て確認すること。

参考文献

- [1] 新井紀子. ロボットは東大に入れるか. イースト・プレス, 2014.
- [2] ITmedia ニュース. 人工知能「東ロボくん」、センター試験模試で「偏差値 57.8」 数学と世界史は偏差値 60 超え. <http://www.itmedia.co.jp/news/articles/1511/16/news061.html> (2015.08.10 閲覧).
- [3] 経済産業省. 我が国ものづくり産業の競争力向上への課題と対応に関する調査. http://www.meti.go.jp/meti_lib/report/2014fy/E004105.pdf (2015.08.10 閲覧).
- [4] 日経ビジネス ONLINE. インダストリー 4.0 とは何か? ドイツが官民一体で進める「第 4 の産業革命」(1). <http://business.nikkeibp.co.jp/article/world/20140717/268842/?rt=nocnt>(2015.10.01 閲覧).
- [5] ソフトバンク. 製品情報 pepper. <http://www.softbank.jp/robot/consumer/products/>(2015.09.30 閲覧).
- [6] IBM Watson (ワトソン)Japan. Watson. <http://www.ibm.com/smarterplanet/jp/ja/ibmwatson/>(2015.10.01 閲覧).
- [7] Wolfram. Wolfram 言語 & システム ドキュメントセンター. <http://reference.wolfram.com/language/> (2014.11.25 閲覧).