筑波大学　情報学群　情報科学類・情報メディア創成学類

令和９年度　学群編入学試験

学力試験問題（専門科目）

[注意事項]

1. 試験開始の合図があるまで、問題の中を見てはいけません。
2. 解答用紙の定められた欄に、氏名、受験番号を記入すること。
3. この問題冊子は全部で９ページ（表紙、白紙を除く）です。
4. 問題１から問題４（数学、情報基礎）の計４問をすべて答えなさい。
5. 解答用紙は、４問に対して、各問１枚の合計４枚を用いること。
6. 解答用紙上部の［　　］欄に解答する問題番号を記入すること。
7. 解答用紙の裏面を使用する場合には、その旨を解答用紙の表面に示してください。

問題1　数学(1)

実数変数x, yに対して以下の等式で定まる曲線が囲む領域Dの面積Sを求めたい。

$$x^{\frac{2}{3}} + y^{\frac{2}{3}} = 1$$

このとき、次に示す変数変換により面積Sを求めることを考える。

$$x = r\cos^3\theta, \quad y = r\sin^3\theta \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{(a)}$$

面積Sは領域Dのうち第一象限に含まれる部分の面積S'を4倍すれば求まるので、以下では第一象限の面積S'を求めることにする。
以下の問いに答えなさい。

(1) $x^{\frac{2}{3}} + y^{\frac{2}{3}} \leqq 1$を満たす第一象限の領域の点に1対1対応する$r$の範囲を求め、その根拠を示しなさい。$\theta$については$0 \leqq \theta \leqq \frac{\pi}{2}$とする。
なお、境界では1対1対応しないがここでは無視してよい。

(2) 式(a)の偏導関数$\frac{\partial x}{\partial r}, \frac{\partial x}{\partial \theta}, \frac{\partial y}{\partial r}, \frac{\partial y}{\partial \theta}$を求めなさい。

(3) 式(a)の変数変換のヤコビ行列を示し、その行列式を計算しなさい。

(4) 式(a)の変数変換と(3)のヤコビ行列式を用いて面積S'を求めなさい。
導出の過程も解答に含めること。

問題2　数学 (2)

(1) 4次元ベクトル空間 $V = \mathbb{R}^4$ の次式で表される部分ベクトル空間を U，U の直交補空間を $U^{\perp}$ とする．以下の問いに答えなさい．

$$U = \left\{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \;\middle|\; 2x_1 + 2x_2 - x_3 - x_4 = 0,\ x_3 - x_4 = 0 \right\}$$

(1-1) 以下のベクトル $\boldsymbol{a}_1$ を含む U の正規直交基底，およびベクトル $\boldsymbol{b}_1$ を含む $U^{\perp}$ の正規直交基底を求めなさい．

$$\boldsymbol{a}_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \qquad \boldsymbol{b}_1 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 0 \\ -1 \end{pmatrix}$$

(1-2) 次式を満たすベクトル $\boldsymbol{u} \in U$ とベクトル $\boldsymbol{w} \in U^{\perp}$ を求めなさい．

$$3\boldsymbol{u} + 2\boldsymbol{w} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

(2) 以下の行列 A が対角化可能かを判定しなさい．可能な場合は対角化し，不可能な場合はその理由を説明しなさい．

$$A = \begin{pmatrix} -1 & 0 & -\sqrt{2} \\ 0 & -2 & 0 \\ -\sqrt{2} & 0 & 0 \end{pmatrix}$$

問題3 情報基礎（1）

電卓には逆ポーランド記法（Reverse Polish Notation, RPN）という動作モードを持つものがある。本問題ではRPN電卓をプログラミングすることを考える。一般的な電卓は、演算子を項の間に置く中置記法を採用している。例えば `1 + 2 =` と入力すれば計算結果 `3` が得られる。RPN電卓では、演算子を項の後に置く後置記法を採用する。例えば `1 2 +` と入力すれば計算結果 `3` が得られる。表1にRPNを用いる計算例を示す。ただし、計算順序を明示するために、冗長な括弧を数式に加えている場合がある。

表1　RPNを用いた計算例

	数式	RPN入力	出力
例1	$(1+2)+3$	1 2 + 3 +	6
例2	$1+(2+3)$	1 2 3 + +	6
例3	$(1+2)\times 3$	1 2 + 3 *	9
例4	$(1+2)\times(3+4)$	1 2 + 3 4 + *	21

RPN電卓の動作は、演算子を入力すると直近2つの項を演算し、その結果を新しい項として置き換える動作と言い換えられ、スタックを用いることで表現できる。RPN電卓をC言語で実装したプログラム1について、以下の問いに答えなさい。ただし、対応している演算は加算（+）と乗算（*）のみとする。扱える値は0以上の整数のみであり、`int`型は少なくとも32ビットの大きさがあると仮定する。

(1) プログラム1の空欄（ア）～（オ）を適切に埋めて、プログラムを完成させなさい。ただし、（ア）と（イ）は順不同である。

(2) 関数`term`の動作を文章で説明しなさい（最大120文字程度）。

(3) `1 2 + 3` を入力してプログラム1を実行したときの出力を答えなさい。

(4) `1 a +` を入力してプログラム1を実行したときの出力を答えなさい。

(5) `10 20 2 * 50 + +` を入力してプログラム1を実行したときの出力を答えなさい。

(6) `1 2 3 4 5 + + + +` を入力してプログラム1を実行したときの出力を答えなさい。

次のページに続く

プログラム 1　RPN 電卓

```
#include <stdio.h>
#include <stdlib.h>

static int S[4];
static int p = 0;

static void op_plus() {
  if (p < 2) {
    printf("Error: stack underflow\n");
    exit(1);
  }
  S[p - 2] = S[ (ア) ] + S[ (イ) ];
  p = (ウ) ;
}

static void op_mul() {
  if (p < 2) {
    printf("Error: stack underflow\n");
    exit(1);
  }
  S[p - 2] = S[ (ア) ] * S[ (イ) ];
  p = (ウ) ;
}

static char* term(char* inp) {
  int value = 0;
  while (*inp != ' ' && *inp != '\n') {
    if ('0' <= *inp && *inp <= '9') {
      value = value * 10 + *inp - '0';
    } else {
      printf("Error: not a number: %c\n", *inp);
      exit(1);
    }
    inp++;
  }

  if (p >= 4) {
    printf("Error: stack overflow\n");
    exit(1);
  }
  S[ (エ) ] = value;

  return inp;
}
```

次のページに続く

```
static char* op(char* inp) {
  switch (*inp) {
  case '+':
    op_plus(); break;
  case '*':
    op_mul(); break;
  default:
    printf("Error: not an op: %c\n", *inp);
    exit(1);
  }

  inp++;
  return inp;
}

static void compute_rpn(char* inp) {
  while (*inp) {
    if ('0' <= [(オ)] && [(オ)] <= '9') {
      inp = term(inp);
    } else {
      inp = op(inp);
    }
    while (*inp == ' ' || *inp == '\n') {
      inp++;
    }
  }
}

int main() {
  char inp[128];
  fgets(inp, sizeof(inp), stdin);

  compute_rpn(inp);

  if (p == 0) {
    printf("Error: empty stack\n");
    exit(1);
  } else if (p > 1) {
    printf("Error: not completed\n");
    exit(1);
  }
  printf("%d\n", S[p - 1]);
  return 0;
}
```

問題4 情報基礎（2）

現在の機械学習において広く活用されている技術として誤差逆伝播法がある。誤差逆伝播法は複数の関数の合成で表現される関数の微分を効率的に計算可能なアルゴリズムである。ここで関数 f を以下のように定める。

$$
\begin{aligned}
f(z_0, w_1, \ldots, w_k) = z(g(z(g(\ldots z(g(z_0, w_1)) \ldots, w_{k-1})), w_k)), \\
g(x, w) = xw, \\
z(x) = \cos(x).
\end{aligned}
$$

関数 f はフィードフォワードニューラルネットワークを簡易的にしたものであり、変数 $w_1, \ldots, w_k$ は重みパラメーター、関数 g は線形層、関数 z は活性化関数に対応する。関数 f の変数 $w_1, \ldots, w_k$ に対する偏微分を誤差逆伝播法で求めたい。そのために図1に示す関数 f の計算過程を示すグラフを考える。誤差逆伝播法では、図1のノードを上から順番に評価する順伝播と下から順に評価する逆伝播を実行することによって偏微分を求める。具体的に、順伝播・逆伝播の手順を以下で説明する。

順伝播　$i = 1, \ldots, k$ について、以下の手順を行う。

（手順1）実数値 $g_i = g(z_{i-1}, w_i)$ を計算する。

（手順2）実数値 $z_i = z(g_i)$ を計算する。

上記で計算された実数値 z_k によって $f(z_0, w_1, \ldots, w_k) = z_k$ が成り立つ。

逆伝播

（手順1）初期化として、$dg_{k+1} = 1$ とする。

$i = k, \ldots, 1$ について、以下の手順を行う。

（手順2）実数値 $dz_i = dg_{i+1} \cdot \dfrac{dz}{dx}(g_i)$ を計算する。

（手順3）実数値 $\dfrac{\partial f}{\partial w_i} = dz_i \cdot \dfrac{\partial g}{\partial w}(z_{i-1}, w_i)$ を計算する。

（手順4）実数値 $dg_i = dz_i \cdot \dfrac{\partial g}{\partial x}(z_{i-1}, w_i)$ を計算する。

ここで、$\dfrac{dz}{dx}, \dfrac{\partial g}{\partial x}, \dfrac{\partial g}{\partial w}$ はそれぞれ関数 z の微分、関数 g の第1引数での偏微分、関数 g の第2引数での偏微分を表す。上記で計算された実数値 $\dfrac{\partial f}{\partial w_1}, \ldots, \dfrac{\partial f}{\partial w_k}$ が求めたい関数 f の変数 $w_1, \ldots, w_k$ に対する偏微分である。

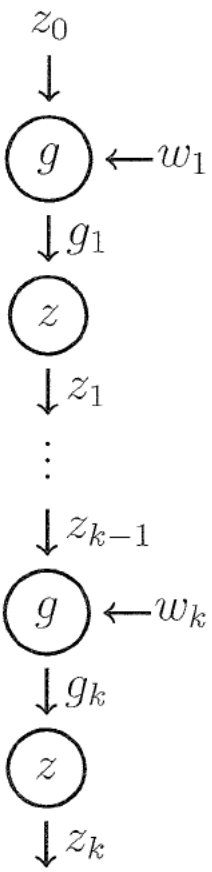

図1: 関数 f の計算グラフ

次のページに続く

プログラム1は上記の誤差逆伝播法の順伝播と逆伝播をC言語で実装したものである。プログラム1について以下の問いに答えなさい。ただし、math.hに定義されているsin、cos関数が使用できるとする。

(1) 関数 g、関数 z の微分・偏微分の実装に関する空欄（ア）、（イ）、（ウ）に当てはまるものをページ下部の選択肢リストから選択しなさい。

(2) 順伝播の実装の空欄（エ）、（オ）に当てはまるものをページ下部の選択肢リストから選択しなさい。

(3) 逆伝播の実装の空欄（カ）、（キ）、（ク）に当てはまるものをページ下部の選択肢リストから選択しなさい。

(4) プログラム1のmainを実行した際に、関数dg_dxが呼ばれる回数を k を使って記述しなさい。

(5) プログラム1を実行した結果、以下の出力が得られる。

```
順伝播
gi = 0.50, zi = 0.88
gi = 1.32, zi = 0.25
gi = 0.45, zi = 0.90
gi = 1.98, zi = −0.40
逆伝播
dzi = −0.92
dzi = 0.88
dzi = −1.54
dzi = 1.11
```

このとき、プログラム1実行後のdw[1]、dw[2]に格納されている値を、小数点3位で四捨五入した小数で答えなさい。

選択肢リスト

（A）x　（B）w　（C）2 * x　（D）2 * w　（E）x * w　（F）cos(x)

（G）sin(x)　（H）−cos(x)　（I）−sin(x)　（J）g(zi[i−1], w[i])

（K）g(zi[i], w[i])　（L）z(zi[i−1])　（M）z(zi[i])

（N）g(gi[i−1], w[i])　（O）g(gi[i], w[i])　（P）z(gi[i−1])

（Q）z(gi[i])　（R）dz(gi[i])　（S）dg_dx(zi[i−1], w[i])

（T）dg_dw(zi[i−1], w[i])　（U）dzi * dg_dx(zi[i−1], w[i])

（V）dzi * dg_dw(zi[i−1], w[i])　（W）dgi * dz(gi[i])

（X）dgi + dgi * dz(gi[i])　（Y）dgi * dg_dx(zi[i−1], w[i])

（Z）dgi * dg_dw(zi[i−1], w[i])

次のページに続く

プログラム1

```
#include <stdio.h>
#include <math.h>

#define K 4 // kの値

// 関数g
double g(double x, double w) {
    return x * w;
}

// 関数gの第1引数での偏微分
double dg_dx(double x, double w) {
    return （ア）;
}

// 関数gの第2引数での偏微分
double dg_dw(double x, double w) {
    return （イ）;
}

// 関数z
double z(double x) {
    return cos(x);
}

// 関数zの微分
double dz(double x) {
    return （ウ）;
}

void calc_grad(double z0, double* w, double* dw) {
    double gi[K+1];
    double zi[K+1];

    // 順伝播
    printf("順伝播\n");
    zi[0] = z0;
    for(int i = 1; i <= K; i++) {
        gi[i] = （エ）;
        zi[i] = （オ）;
        printf("gi = %.2f, zi = %.2f\n", gi[i], zi[i]);
    }
```

次のページに続く

```
    // 逆伝播
    printf("逆伝播\n");
    double dgi = 1.0;
    double dzi;
    for(int i = K; i >= 1; i--) {
        dzi  = [          (カ)          ];
        dw[i] = [          (キ)          ];
        dgi  = [          (ク)          ];
        printf("dzi = %.2f\n", dzi);
    }
}

int main() {
    double z0 = 0.5;
    double w[K+1] = {0.0, 1.0, 1.5, 1.8, 2.2};
    double dw[K+1];

    calc_grad(z0, w, dw);

    return 0;
}
```