# Jacobian 변환으로 다변수 함수 미분계수 구하기

김도윤

## Go Code

```go
package main

import (
    "fmt"
    "math"
)

// 편미분을 계산하는 인터페이스 정의
type PartialDifferential interface {
    Function1(x, y float64) float64
    Function2(x, y float64) float64
    Differential1(func(float64, float64) float64, string) float64
    Differential2(func(float64, float64) float64, string) float64
    Partial1() (float64, float64)
    Partial2() (float64, float64)
}

// 다변수 함수와 편미분 계산을 위한 구조체 정의
type MultiVarFunction struct {
    h float64 // 편미분 계산을 위한 작은 값
}

// Function1 정의
func (mvf MultiVarFunction) Function1(x, y float64) float64 {
    return x * x * y
}

// Function2 정의
func (mvf MultiVarFunction) Function2(x, y float64) float64 {
    return y * math.Cos(x/y)
}

// Differential1 정의 - 편미분 계산 함수
func (mvf MultiVarFunction) Differential1(f func(float64, float64) float64,
varName string) float64 {
    var x, y float64 = 1, 2 // 편미분 계산을 위한 변수 값 설정
    h := mvf.h

    switch varName {
    case "x":
        return (f(x+h, y) - f(x, y)) / h
    case "y":
        return (f(x, y+h) - f(x, y)) / h
    default:
```

```
        return 0
    }
}

// Differential2 정의 - 편미분 계산 함수
func (mvf MultiVarFunction) Differential2(f func(float64, float64) float64,
varName string) float64 {
    var x, y float64 = 1, 2 // 편미분 계산을 위한 변수 값 설정
    h := mvf.h

    switch varName {
    case "x":
        return (f(x+h, y) - f(x, y)) / h
    case "y":
        return (f(x, y+h) - f(x, y)) / h
    default:
        return 0
    }
}

// Partial1 정의 - 편미분 결과 반환 함수
func (mvf MultiVarFunction) Partial1() (float64, float64) {
    dfdx := mvf.Differential1(mvf.Function1, "x")
    dfdy := mvf.Differential1(mvf.Function1, "y")
    return dfdx, dfdy
}

// Partial2 정의 - 편미분 결과 반환 함수
func (mvf MultiVarFunction) Partial2() (float64, float64) {
    dfdx := mvf.Differential2(mvf.Function2, "x")
    dfdy := mvf.Differential2(mvf.Function2, "y")
    return dfdx, dfdy
}

// 2x2 행렬의 행렬식을 계산하는 함수
func determinant2x2(a11, a12, a21, a22 float64) float64 {
    return a11*a22 - a12*a21
}

func main() {
    mvf := MultiVarFunction{h: 1e-5} // 편미분 계산을 위한 작은 값 설정

    dfdx1, dfdy1 := mvf.Partial1()
    dfdx2, dfdy2 := mvf.Partial2()

    // 2x2 야코비안 행렬 출력
    fmt.Printf("Jacobian matrix:\n")
    fmt.Printf("[[%f, %f],\n [%f, %f]]\n", dfdx1, dfdy1, dfdx2, dfdy2)

    // 2x2 행렬의 행렬식 계산
    det := determinant2x2(dfdx1, dfdy1, dfdx2, dfdy2)
    fmt.Printf("Determinant: %f\n", det)
}
```

```
----------------------------------------------------------------------
-실행결과
Jacobian matrix:
[[4.000020, 1.000000],
 [-0.479428, 1.117295]]
Determinant: 4.948629
----------------------------------------------------------------------
```