[CSE4152] 고급소프트웨어 실습 I
Week 3

서강대학교 공과대학 컴퓨터공학과
교수 임 인 성

공개 소프트웨어를 통한 문제 해결

- **6주차 실습의 목표**
  - Netlib와 같은 수치 계산용 SW 공개 사이트의 프로그램을 사용하여 응용 문제를 해결하여 본다.
  - C/C++ 프로그램에서 다른 언어(본 실습에서는 FORTRAN 언어)로 작성된 함수를 호출하여 문제를 해결하여 본다.

- **C/C++과 FORTRAN 함수의 혼용을 통한 문제 해결**
  - f2c와 코드 자동 변환 소프트웨어를 사용하는 방법
  - C/C++ 프로그램에서 FORTRAN 함수를 직접 호출하는 방법

✓LISP/PROLOG 프로그램에서 MATLAB에서 제공하는 C/C++ 함수를 호출할 수 있을까?

✓Python 프로그램에서 FORTRAN으로 작성한 math library의 함수를 호출할 수 있을까?

✓ https://en.wikipedia.org/wiki/List_of_numerical_analysis_software 참조

Netlib Repository 소개

From Wikipedia

- **A repository of software for scientific computing** maintained by AT&T, Bell Laboratories, the University of Tennessee and Oak Ridge National Laboratory.
- Netlib comprises **a large number of separate programs and libraries**.
- Most of the code is **written in C and Fortran**, with some programs in other languages.

➢ 2020년 10월 11일 현재까지 총 1,211,587,548의 코드 요청이 있었음.

URL: www.netlib.org

Main Index of Software Libraries

Search the Netlib Repository attribute/value database.

A more descriptive version of this list is also available.

Netlib Libraries:

a	fftpack	machines	random
access	fishpack	magic	research
aicm	fitpack	maspar	rib
alliant	floppy	math77	scalapack
amos	fmm	mds	sched
ampl	fn	microscope	scilib
anl-reports	fortran	minpack	seispack
apollo	fortran-m	misc	sequent
arpack	fp	mpfun	sfmm
atlas	gcv	mpi	slap
benchmark	gmat	mpicl	slatec
bib	gnu	na-digest-html	sminpack
bibnet	go	napack	sodepack
bihar	graphics	netsolve	sparse
blacs	harwell	news	sparse-blas
blas	hence	numeralgo	sparspak
blast	hompack	ode	specfun
bmp	hpf	odepack	spin
c	hypercube	odrpack	srwn
c++	ieeecss	opt	stoeplitz
cephes	ijsa	p4	stringsearch
chammp	image	paragraph	svdpack

BLAS (Basic Linear Algebra Subprograms)

Menu

- Presentation:
  - Acknowledgments:
  - History
- Software:
  - Licensing:
  - REFERENCE BLAS Version 3.8.0
  - CBLAS
  - Level 3 BLAS tuned for single processors with caches
  - Extended precision Level 2 BLAS routines
  - BLAS for windows
  - GIT Access
  - The netlib family and its cousins
- Support
- Documentation
  - BLAS Technical Forum
  - Optimized BLAS Library
- BLAS Routines
  - LEVEL 1
  - LEVEL 2
  - LEVEL 3
  - Extended precision Level 2 BLAS routines

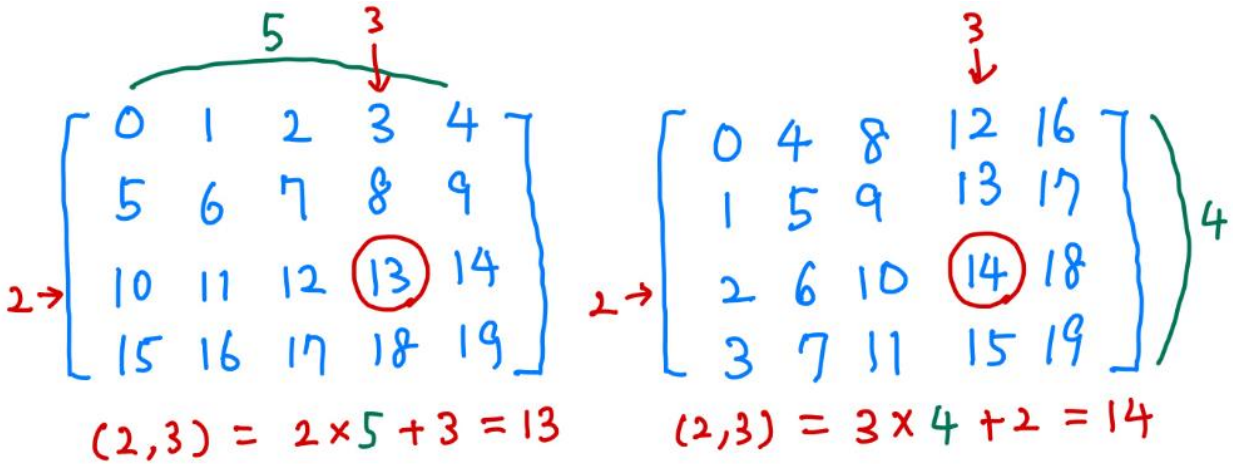
Questions/comments? lapack@icl.utk.edu

Contact us get the lastest news

BLAS(Legacy Website) FAQ

C/C++와 FORTRAN 언어 간의 차이

	C 언어	FORTRAN 언어
매개 변수 전달 방법	call by value	call by reference
행렬 저장 방법	row-major order	column-major order



Netlib 함수를 사용한 다항식의 근 구하기

- **문제:** 다음 n차 다항식의 모든 근을 구하라.
  - ✓ n차 다항식의 근은 실근, 허근, 중근을 포함하여 정확히 n개가 존재함

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$$

- **방법:** Netlib의 rpoly.f 함수를 사용하여 해결함.
  - rpoly.f의 RPOLY() 함수의 인자들의 의미를 이해해야 함. 자세한 방법은 강의자료 참조.

```
      SUBROUTINE RPOLY(OP, DEGREE, ZEROR, ZEROI,
     * FAIL)
C FINDS THE ZEROS OF A REAL POLYNOMIAL
C OP - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN
C      ORDER OF DECREASING POWERS.
C DEGREE - INTEGER DEGREE OF POLYNOMIAL.
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF
C      REAL AND IMAGINARY PARTS OF THE
C      ZEROS.
C FAIL - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF
```

RPOLY Function

```
      SUBROUTINE RPOLY(OP, DEGREE, ZEROR, ZEROI,
     * FAIL)
C FINDS THE ZEROS OF A REAL POLYNOMIAL
C OP - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN
C      ORDER OF DECREASING POWERS.
C DEGREE - INTEGER DEGREE OF POLYNOMIAL.
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF
C      REAL AND IMAGINARY PARTS OF THE
C      ZEROS.
C FAIL - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF
```

이 함수는 주어진 n차 다항식 $p_n(x)$에 대하여,

차수 n을 이름이 DEGREE인 int 타입의 변수에,

그리고 $n+1$개의 계수들을 고차항의 계수부터 차례대로 $(\{a_n, a_{n-1}, \cdots, a_1, a_0\})$ 이름이 OP인 double 타입의 배열을 통하여 넘겨 받아,

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0$$

```
      SUBROUTINE RPOLY(OP, DEGREE, ZEROR, ZEROI,
     * FAIL)
C FINDS THE ZEROS OF A REAL POLYNOMIAL
C OP - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN
C      ORDER OF DECREASING POWERS.
C DEGREE - INTEGER DEGREE OF POLYNOMIAL.
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF
C      REAL AND IMAGINARY PARTS OF THE
C      ZEROS.
C FAIL - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF
```

n개의 근 $\alpha_i + \beta_i \mathbf{i}$ $(i = 0, 1, 2, \cdots, n-1)$을 구한 후,

실수부 (real part)에 해당하는 값들 $(\{\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{n-1}\})$을 이름이 ZEROR인 double 타입의 배열에 담아,

그리고 허수부 (imaginary part)에 해당하는 값들 $(\{\beta_0, \beta_1, \beta_2, \cdots, \beta_{n-1}\})$을 이름이 ZEROI인 double 타입의 배열에 담아 넘겨주게 된다.

또한 이름이 FAIL인 long int 타입의 변수에 함수 수행 결과에 대한 정보를 담아 넘겨주는데, 이에 대한 설명을 발췌하면 다음과 같다.

```
#include <stdio.h>
#include <math.h>
#define DEGREE 4
#define NCOEF 5
extern "C"
{
   int rpoly_(double *, int *, double *, double *, long int *);
}
 void main(void) {
   double poly[NCOEF] = { 1.0, -11.0, 42.35, -66.55, 35.1384 };

   int degree = DEGREE;
   double zeror[DEGREE], zeroi[DEGREE];
   long int fail;
   int i;
   rpoly_(poly, &degree, zeror, zeroi, &fail);

   if (fail) { ··· }
   for (i = 0; i < degree; i++) printf("%10f ", zeror[i]); printf("\n");
   for (i = 0; i < degree; i++) printf("%10f ", zeroi[i]); printf("\n");
}
```

```
      SUBROUTINE RPOLY(OP, DEGREE, ZEROR, ZEROI,
     * FAIL)
C FINDS THE ZEROS OF A REAL POLYNOMIAL
C OP - DOUBLE PRECISION VECTOR OF COEFFICIENTS IN
C      ORDER OF DECREASING POWERS.
C DEGREE - INTEGER DEGREE OF POLYNOMIAL.
C ZEROR, ZEROI - OUTPUT DOUBLE PRECISION VECTORS OF
C      REAL AND IMAGINARY PARTS OF THE
C      ZEROS.
C FAIL - OUTPUT LOGICAL PARAMETER, TRUE ONLY IF
```

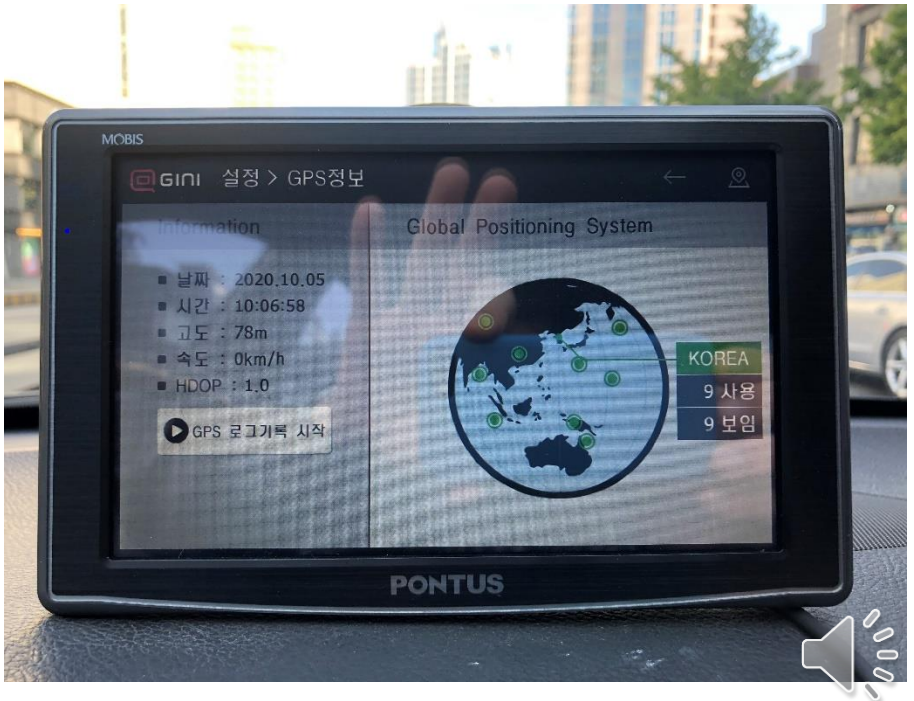
	C 언어	FORTRAN 언어
매개 변수 전달 방법	call by value	call by reference
행렬 저장 방법	row-major order	column-major order

$$p_4(x) = x^4 - 11.0x^3 + 42.35x^2 - 66.55x + 35.1384$$

GPS 수신기의 위치 계산 문제

- **문제: 다음의 System of Nonlinear Equations을 풀어라.**

$$
\begin{aligned}
f_1(x_1,x_2,x_3,x_4) &= (x_1-p_{11})^2+(x_2-p_{12})^2+(x_3-p_{13})^2-\{\,C(tr_1+x_4-t_1)\,\}^2 = 0\\
f_2(x_1,x_2,x_3,x_4) &= (x_1-p_{21})^2+(x_2-p_{22})^2+(x_3-p_{23})^2-\{\,C(tr_2+x_4-t_2)\,\}^2 = 0\\
f_3(x_1,x_2,x_3,x_4) &= (x_1-p_{31})^2+(x_2-p_{32})^2+(x_3-p_{33})^2-\{\,C(tr_3+x_4-t_3)\,\}^2 = 0\\
f_4(x_1,x_2,x_3,x_4) &= (x_1-p_{41})^2+(x_2-p_{42})^2+(x_3-p_{43})^2-\{\,C(tr_4+x_4-t_4)\,\}^2 = 0
\end{aligned}
$$



Newton-Raphson 방법

- **System of nonlinear equations**

$$
\begin{aligned}
f_1(x_1, x_2, \cdots, x_n) &= 0 \\
f_2(x_1, x_2, \cdots, x_n) &= 0 \\
&\vdots \\
f_n(x_1, x_2, \cdots, x_n) &= 0
\end{aligned}
$$

$$
\begin{aligned}
f_1(x_1, x_2, x_3, x_4) &= (x_1 - p_{11})^2 + (x_2 - p_{12})^2 + (x_3 - p_{13})^2 - \{ C(tr_1 + x_4 - t_1) \}^2 = 0 \\
f_2(x_1, x_2, x_3, x_4) &= (x_1 - p_{21})^2 + (x_2 - p_{22})^2 + (x_3 - p_{23})^2 - \{ C(tr_2 + x_4 - t_2) \}^2 = 0 \\
f_3(x_1, x_2, x_3, x_4) &= (x_1 - p_{31})^2 + (x_2 - p_{32})^2 + (x_3 - p_{33})^2 - \{ C(tr_3 + x_4 - t_3) \}^2 = 0 \\
f_4(x_1, x_2, x_3, x_4) &= (x_1 - p_{41})^2 + (x_2 - p_{42})^2 + (x_3 - p_{43})^2 - \{ C(tr_4 + x_4 - t_4) \}^2 = 0
\end{aligned}
$$

$$
\mathbf{F}(x_1, x_2, \cdots, x_n) = (f_1(x_1, x_2, \cdots, x_n),\ f_2(x_1, x_2, \cdots, x_n), \cdots,\ f_n(x_1, x_2, \cdots, x_n))^t
$$

$$
\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n \qquad \mathbf{x} = (x_1, x_2, \cdots, x_n),\ \mathbf{0} = (0, 0, \cdots, 0)^t
$$

$$
\mathbf{F}(\mathbf{x}) = \mathbf{0}
$$

- n = 1 $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = 0 \longrightarrow \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \ k = 0, 1, 2, \cdots$$

- n > 1

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad \longrightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \ k = 0, 1, 2, \cdots$$

$$\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})^t$$

$$J(\mathbf{x}) = \frac{\partial(f_1, f_2, \cdots, f_n)}{\partial(x_1, x_2, \cdots, f_n)}(\mathbf{x})$$

$$= \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{pmatrix}$$

야코비 행렬(Jacobian matrix)

minpack 함수를 사용한 비선형 방정식 시스템의 풀이

- **HYBRJ1 함수를 사용한 문제 풀이**
  - 변형된 **Powell 방법** 사용
  - Solver에게 **방정식 외에도 야코비 행렬을 함수 형태로 제공해야 함**.
  - ✓ 자세한 방법은 강의 자료 참조

$$\mathbf{F}(x_1, x_2, \cdots, x_n) = (f_1(x_1, x_2, \cdots, x_n),\ f_2(x_1, x_2, \cdots, x_n), \cdots,\ f_n(x_1, x_2, \cdots, x_n))^t$$

$$\mathbf{x} = (x_1, x_2, \cdots, x_n),\ \mathbf{0} = (0, 0, \cdots, 0)^t$$

$$J(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{pmatrix}$$

$$\begin{aligned} f_1(x_1, x_2, x_3, x_4) &= (x_1 - p_{11})^2 + (x_2 - p_{12})^2 + (x_3 - p_{13})^2 - \{ C(tr_1 + x_4 - t_1) \}^2 = 0 \\ f_2(x_1, x_2, x_3, x_4) &= (x_1 - p_{21})^2 + (x_2 - p_{22})^2 + (x_3 - p_{23})^2 - \{ C(tr_2 + x_4 - t_2) \}^2 = 0 \\ f_3(x_1, x_2, x_3, x_4) &= (x_1 - p_{31})^2 + (x_2 - p_{32})^2 + (x_3 - p_{33})^2 - \{ C(tr_3 + x_4 - t_3) \}^2 = 0 \\ f_4(x_1, x_2, x_3, x_4) &= (x_1 - p_{41})^2 + (x_2 - p_{42})^2 + (x_3 - p_{43})^2 - \{ C(tr_4 + x_4 - t_4) \}^2 = 0 \end{aligned}$$

HYBRJ1 Function

```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

n: 변수의 개수 (즉 방정식의 개수)를 나타내는 int 타입의 양의 정수.

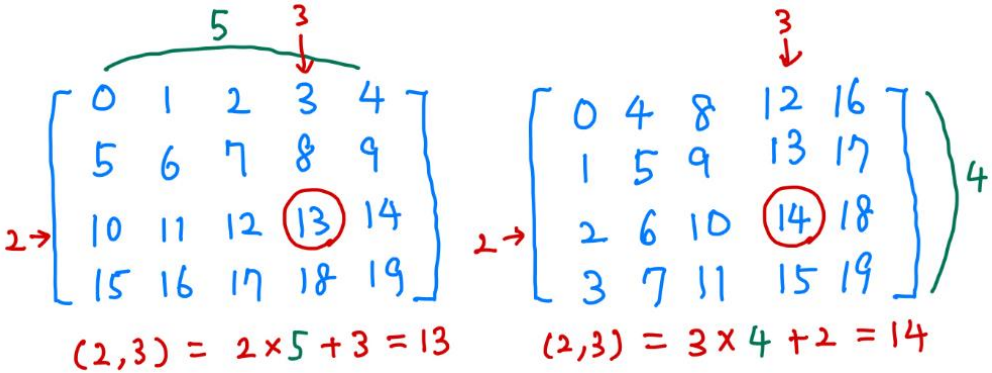
x: n개의 double 타입의 원소를 가지는 배열. 함수 호출 시에는 초기값 $\mathbf{x}^{(0)}$을 저장해 주어야 하며, 함수 리턴 시 이 함수가 구한 근 (정확히 말해서 근에 대한 추정값) $\bar{\mathbf{x}}$를 저장하고 있음.

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad \longrightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \ k = 0, 1, 2, \cdots$$

$$\mathbf{F}(x_1, x_2, \cdots, x_n) = (f_1(x_1, x_2, \cdots, x_n), \ f_2(x_1, x_2, \cdots, x_n), \cdots, \ f_n(x_1, x_2, \cdots, x_n))^t$$

$$\mathbf{x} = (x_1, x_2, \cdots, x_n), \ \mathbf{0} = (0, 0, \cdots, 0)^t$$

HYBRJ1 Function




```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

$$J(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \frac{\partial f_n}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{pmatrix}$$

ldfjac: 야코비 행렬 관련 정보 저장을 위한 공간인 2차원 배열 fjac의 실제 행의 개수로서 n보다 같거나 큰 int 타입의 양의 정수. C 언어와는 달리 FORTRAN에서는 열 우선 순서로 저장이 되기 때문에 실제 fjac 행렬이 double fjac[MATROWS][MATCOLS];와 같이 정의가 되었다면 MATROWS 값을 알아야 주소 계산이 가능하므로, 함수 호출 시 이 값을 ldfjac 변수를 통하여 넘겨주어야 함. 가장 간단한 방법은 double fjac[N][N]; 과 같이 n행 n열 행렬 공간을 잡은 후 n 값을 넘기는 것이다.

HYBRJ1 Function

```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

fvec: 함수 리턴 시 $\bar{\mathbf{x}}$에서의 함수값 $\mathbf{F}(\bar{\mathbf{x}})$를 저장하고 있는 n개의 double 타입의 원소를 가지는 배열.

fjac: 함수 리턴 시 $\bar{\mathbf{x}}$에서의 야코비 행렬 $J(\bar{\mathbf{x}})$와 관련된 정보를 저장하고 있는 double 타입의 배열. 그 크기는 최소한 n행 n열 행렬을 저장할 수 있을 정도로 커야 함.

tol: 함수 호출시 넘겨주어야 하는 0보다 같거나 큰 double 타입의 값으로서, 현재까지 구한 근 추정값 $\bar{\mathbf{x}}$와 정확한 근과의 상대 오차가 이 값보다 같거나 작다고 판단될 때에 계산을 멈춤.

HYBRJ1 Function

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad \longrightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \; k = 0, 1, 2, \ldots.$$

```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

fcn: 이 함수는 자신이 C 언어로 작성하여 제공하는 함수로서 임의의 $\mathbf{x}$에 대한 다변수 함수값 $\mathbf{F}(\mathbf{x})$와 야코비 행렬값 $J(\mathbf{x})$를 계산해주는 것을 목적으로 한다. 이 함수는 FORTRAN에서는 다음과 같이 정의가 되는데,

The value of iflag should not be changed by fcn.

```
subroutine fcn(n, x, fvec, fjac, ldfjac, iflag)
integer n,ldfjac,iflag
double precision x(n),fvec(n),fjac(ldfjac,n)
```

이에 대응하여 C에서는 다음과 같은 프로토타입의 함수를 작성하면 된다.

```
void fcn(int *n, double *x, double *fvec, double *fjac, int
*ldfjac, int *iflag);
```

HYBRJ1 Function

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad \longrightarrow \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J(\mathbf{x}^{(k)})^{-1}\mathbf{F}(\mathbf{x}^{(k)}), \; k = 0, 1, 2, \ldots.$$

```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

wa: 이 함수를 호출하는 함수에서 다음 인자 lwa가 저장하고 있는 양의 정수 개 만큼의 원소를 가지는 double 타입의 배열에 대한 메모리 영역을 확보한 후, 함수 호출 시 이 배열을 넘김. 이 영역은 근을 구하는 계산 시 임시 데이터를 저장하기 위한 영역으로 쓰임.

lwa: 함수 호출 시 wa 배열의 길이를 지정하는 int 타입의 변수로서, (n*(n+13))/2보다 같거나 큰 양의 정수값을 가져야 함.

HYBRJ1 Function

```
subroutine hybrj1(fcn,n,x,fvec,fjac,ldfjac,tol,info,wa,lwa)
integer n,ldfjac,info,lwa
double precision tol
double precision x(n),fvec(n),fjac(ldfjac,n),wa(lwa)
external fcn
```

`info`: 함수 리턴 시 함수 수행 결과에 대한 정보를 저장하는 `int` 타입의 정수 변수로서 그에 대한 설명을 발췌하면 다음과 같음.

`info` is an integer output variable. If the user has terminated execution, info is set to the (negative) value of iflag (see description of fcn). Otherwise, `info` is set as follows.

a) `info` = 0: Improper input parameters.

b) `info` = 1: Algorithm estimates that the relative error between x and the solution is at most `tol`.

c) `info` = 2: Number of calls to `fcn` with `iflag` = 1 has reached 100*(n+1).

d) `info` = 3: `tol` is too small. No further improvement in the approximate solution x is possible.

e) `info` = 4: Iteration is not making good progress.