

---

# Single Screw Simulator (Ver.12.0.0)

## SSSカスタマイズ機能 補足資料

・ユーザ定義ルーチン / Fortran compiler の利用方法	p. 1
・開発環境の設定例 ① Intel Fortran	pp. 2- 8
・開発環境の設定例 ② GFortran	pp. 9-17
・変数一覧	pp.18-27
・サンプルプログラム	pp.28-49

### ③ SSSカスタマイズ機能

#### ユーザ定義ルーチン / Fortran compiler の利用方法

SSS Ver.12.0.0 で利用可能な開発環境／下表2種類のいずれかを利用して下さい.

開発環境 (compiler)	使用PCのOS (Operating System)
① Intel Fortran	Windows
② GFortran (GNU Fortran)	Windows (MinGWを使用)

SingleScrewSimulatorVer12.0.0¥bin¥x86 内には3つのSystemフォルダが存在します.

PC > OS (C:) > SSS > SingleScrewSimulatorVer12.0.0 > bin > x86		
名前	更新日時	種類
Systemver.12.0.0	2022/11/07 10:41	ファイルフォルダー
Systemver.12.0.0_GFORT	2022/11/07 10:43	ファイルフォルダー
Systemver.12.0.0_IFORT	2022/11/07 11:09	ファイルフォルダー

#### フォルダの内容と対象ユーザ

Systemver.12.0.0	: 通常解析を実施されるユーザ	標準版	ユーザ定義ルーチン 利用ユーザ
Systemver.12.0.0_IFORT	: ①Intel Fortranを利用されるユーザ		
Systemver.12.0.0_GFORT	: ②GFortranを利用されるユーザ		

## 開発環境の設定例 ① Intel Fortran

### 1) Intel® oneAPI HPC Toolkit をダウンロードする.

入手可能なウェブサイト(下図に抜粋)

<https://www.intel.com/content/www/us/en/developer/articles/news/free-intel-software-developer-tools.html>

Free Intel® Software Development Tools

Published: 01/12/2021  
Last Updated: 02/01/2021

#### Intel® oneAPI Toolkits - Free for All Developers

Intel® oneAPI Toolkits are the next generation of standards-based Intel® Software Development Tools for building applications across diverse architectures. All Intel® oneAPI Toolkits products are available at no cost. The Intel oneAPI Toolkits do not require license files and the terms of use are based on the [End User License Agreement](#). Support is available via [Intel Developer Zone community forums](#).

#### Native Code Toolkits

##### Intel® oneAPI Base Toolkit

Get started with this foundational kit that enables developers of all types to build, test, and deploy performance-driven, data-centric applications across CPUs, GPUs, and FPGAs. For specialized workloads, use the Base Kit with one or more add-on toolkits.

[Get the Base Kit](#) + [Add a Domain-Specific Toolkit](#)

##### Add Domain-Specific Toolkits for Specialized Workloads

##### Intel® oneAPI HPC Toolkit

Deliver fast DPC++, C++, Fortran, OpenMP, and MPI applications that scale.

[Get the Base Kit](#) + [Get the HPC Kit](#)

##### Intel® oneAPI IoT Toolkit

Build high-performing, efficient, reliable solutions that run at the network's edge.

[Get the Base Kit](#) + [Get the IoT Kit](#)

##### Intel® oneAPI Rendering Toolkit

Accelerate High-Fidelity Rendering and Visualization Applications with Powerful Libraries.

[Get the Base Kit](#) + [Get the Render Kit](#)

コマンドプロンプト上でコンパイル  
する場合にはHPC Kit のみ  
インストールします(当社実施環境).

インテル社製品のダウンロードには、  
インテル社のサイトでアカウントを  
作成する必要があります。

## 開発環境の設定例 ① Intel Fortran

### 2) Intel® oneAPI HPC Toolkit をインストールする.

#### 【補足情報】

- ・ Intel® oneAPI HPC Toolkit を使用するためには、Microsoft社の Visual Studio をインストールする必要があります。  
(当社使用バージョン: Microsoft Visual Studio Community 2017)
- ・ インテルソフトウェア開発製品の正規販売代理店であるエクセルソフト株式会社のウェブサイトから評価版をインストールすることも可能です。30日間のサポートサービスを受けられます。

エクセルソフト株式会社のウェブサイト(下図に抜粋)

<https://www.xlsoft.com/jp/products/intel/download/eval.html>

The screenshot shows the Intel oneAPI HPC Toolkit evaluation page on the XLSoft website. The page has a blue header with the XLSoft logo and a search bar. Below the header is a navigation bar with links: ホーム (Home), 製品 (Products), 購入 (Purchase), 技術情報 (Technical Information), サポート (Support), 評価する (Evaluate), and お問い合わせ (Contact Us). The main content area has a breadcrumb trail: ホーム > ダウンロード > 評価版申し込み手順. The title is 'インテル® ソフトウェア開発製品 評価版/コミュニティー・ライセンスの申し込み手順'. Below the title is a paragraph explaining the evaluation version and license application process. A box contains a section titled '■ インテル® oneAPI ツールキット評価版サポートサービスに関して' with detailed information about the support service, including a 13-day trial period and a 30-day support period. At the bottom of the page is a blue bar with the text '評価版/コミュニティー・ライセンス申し込み手順'.

## 開発環境の設定例 ① Intel Fortran

3) Intel® oneAPI HPC Toolkit をインストール後, Windows 画面左下のスタートボタンから Intel oneAPI command prompt for Intel32 をクリックしてコマンドプロンプトを起動します.



2. Intel oneAPI command prompt for Intel32 をクリックする.  
(Intel64 も存在するため注意)

1. Windowsスタートボタン  
をクリックする.  
(当社実施環境 windows10)

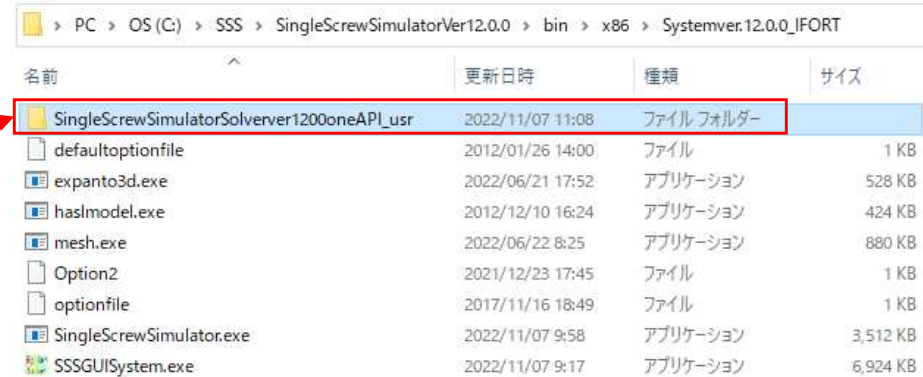
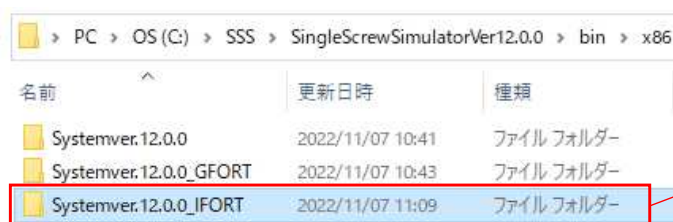
3. コマンドプロンプトが起動する. (x86の表記を確認)

```
Intel(r) oneAPI Tools
:: initializing oneAPI environment...
:: Initializing Visual Studio command-line environment...
Visual Studio version 15.9.22 environment configured.
"C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\
Visual Studio command-line environment initialized for: 'x86'
: compiler -- latest
: debugger -- latest
: dev-utilities -- latest
: inspector -- latest
: itac -- latest
: mpi -- latest
: tbb -- latest
:: oneAPI environment initialized ::
C:\Program Files (x86)\Intel\oneAPI>
```

## 開発環境の設定例 ① Intel Fortran

4) Intel oneAPI command prompt for Intel32 のコマンドプロンプト上で, ユーザーチンのソースコードが格納されているSSS内のフォルダに移動します.

(¥bin¥x86¥Systemver.12.0.0\_IFORT¥SingleScrewSimulatorSolver1200oneAPI\_usr)



```
Intel(r) oneAPI Tools
:: initializing oneAPI environment...
:: Initializing Visual Studio command-line environment...
Visual Studio version 15.9.22 environment configured.
"C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\"
Visual Studio command-line environment initialized for: 'x86'
: compiler -- latest
: debugger -- latest
: dev-utilities -- latest
: inspector -- latest
: itac -- latest
: mpi -- latest
: tbb -- latest
:: oneAPI environment initialized ::

C:\Program Files (x86)\Intel\oneAPI> cd C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewS
imulatorSolver1200oneAPI_usr
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>
```

cd (ディレクトリ移動コマンド) の後に,  
SingleScrewSimulatorSolver1200oneAPI\_usr  
までのパス(C:ドライブからの絶対パス)  
を入力して, キーボードのEnterキーを  
押すと, カレントディレクトリが変更されます.



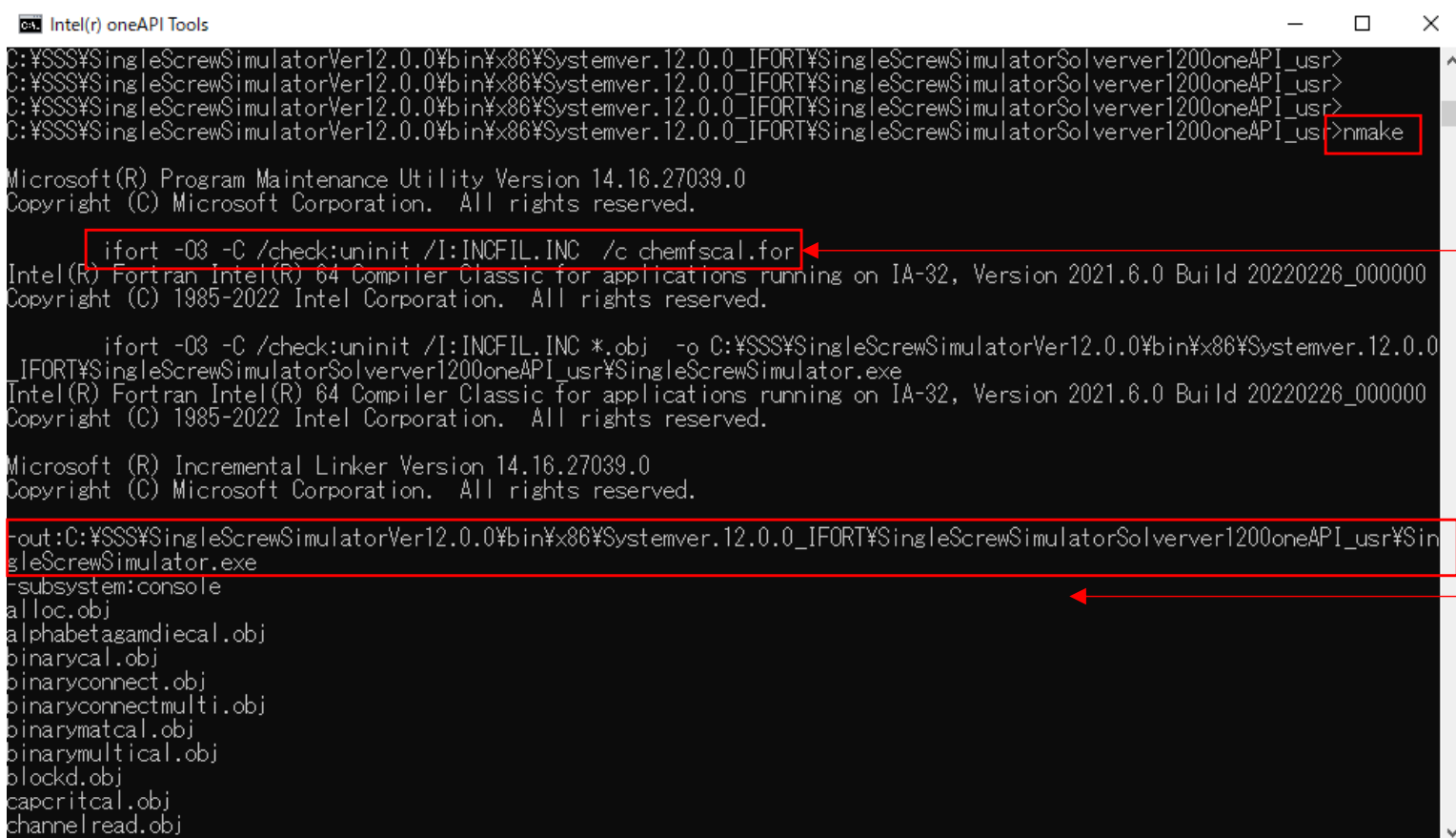
## 開発環境の設定例 ① Intel Fortran

5) ¥SingleScrewSimulatorSolverver1200oneAPI\_usr 内に存在するソースコードを用途向きに書き直します  
(任意のエディタを使用)。

« bin > x86 > Systemver.12.0.0_IFORT > SingleScrewSimulatorSolverver1200oneAPI_usr				
名前	更新日時	種類	サイズ	
viscal.for	2022/10/09 7:31	Fortran Source	10 KB	ユーザがカスタマイズ可能な サブルーチンのソースコード (ファイル名.for)
tempcal.for	2022/10/05 18:08	Fortran Source	15 KB	
initialsetforchem.for	2022/10/09 11:22	Fortran Source	5 KB	
chemwrite.for	2022/10/09 7:30	Fortran Source	7 KB	
chemvariable.for	2022/10/09 7:26	Fortran Source	1 KB	
chemfscal.for	2022/10/09 7:33	Fortran Source	2 KB	
vtkwite.obj	2022/10/12 18:21	3D Object	36 KB	(注)ソースコード以外に格納されている ファイル(.obj など)を変更したり 削除するとコンパイルができなく なりますので注意して使用ください。
voftraceimplicit.obj	2022/10/12 18:22	3D Object	94 KB	
voftrace.obj	2022/10/12 18:21	3D Object	116 KB	
vofconnectimplicit.obj	2022/10/12 18:22	3D Object	12 KB	
vofconnect.obj	2022/10/12 18:21	3D Object	11 KB	
viscaltad.obj	2022/10/12 18:21	3D Object	3 KB	
viscaldie.obj	2022/10/12 18:21	3D Object	16 KB	
viscal.obj	2022/10/12 18:21	3D Object	45 KB	
unfillcal1.obj	2022/10/12 18:21	3D Object	109 KB	
timetempset.obj	2022/10/12 18:21	3D Object	25 KB	
timepressset.obj	2022/10/12 18:21	3D Object	9 KB	
timefillset.obj	2022/10/12 18:21	3D Object	12 KB	
tempconnect.obj	2022/10/12 18:21	3D Object	19 KB	
tempcal.obj	2022/10/12 18:21	3D Object	178 KB	

## 開発環境の設定例 ① Intel Fortran

- 6) ソースコードを編集後、コマンドプロンプト上で `nmake` と入力してキーボードのEnterキーを押すと、`makefile` を利用したコンパイルが実行されます。



```
Intel(r) oneAPI Tools
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>
C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr>nmake

Microsoft(R) Program Maintenance Utility Version 14.16.27039.0
Copyright (C) Microsoft Corporation. All rights reserved.

ifort -03 -C /check:uninit /I:INCFIL.INC /c chemfscal.for
Intel(R) Fortran Intel(R) 64 Compiler Classic for applications running on IA-32, Version 2021.6.0 Build 20220226_000000
Copyright (C) 1985-2022 Intel Corporation. All rights reserved.

ifort -03 -C /check:uninit /I:INCFIL.INC *.obj -o C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr\SingleScrewSimulator.exe
Intel(R) Fortran Intel(R) 64 Compiler Classic for applications running on IA-32, Version 2021.6.0 Build 20220226_000000
Copyright (C) 1985-2022 Intel Corporation. All rights reserved.

Microsoft (R) Incremental Linker Version 14.16.27039.0
Copyright (C) Microsoft Corporation. All rights reserved.

-out:C:\SSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_IFORT\SingleScrewSimulatorSolver1200oneAPI_usr\SingleScrewSimulator.exe
-subsystem:console
alloc.obj
alphabetagamdiecal.obj
binarycal.obj
binaryconnect.obj
binaryconnectmulti.obj
binarymatcal.obj
binarymultical.obj
blockd.obj
capcritical.obj
channelread.obj
```

変更したソースコードのみ  
コンパイルされます。

コンパイルに成功すると、  
フォルダ内に実行プログラム  
SingleScrewSimulator.exe  
が作成されます。

コンパイルに失敗した場合は  
エラーメッセージが出力され  
るので、ソースコードを修正  
後に再度 `nmake` を実施して  
ください。



## 開発環境の設定例 ① Intel Fortran

7) ¥ SingleScrewSimulatorSolverver1200oneAPI\_usrフォルダ内の SingleScrewSimulator.exe の更新日時が、コンパイルした日時に変更されていることを確認後、Systemver.12.0.0\_IFORTフォルダ内に存在する SingleScrewSimulator.exeを上書き保存して更新すると、SSSのGUIから解析実行した際に、更新された SingleScrewSimulator.exe が実行されます。

名前	更新日時	種類
SingleScrewSimulator.exe	2022/11/07 11:29	アプリケーション
chemfscal.obj	2022/11/07 11:29	3D Object
chemfscal.for	2022/11/07 11:29	Fortran Source
Makefile	2022/11/07 11:29	ファイル
viscal.obj	2022/11/07 11:26	3D Object
viscal.for	2022/11/07 11:25	Fortran Source
heleshawcgsss.obj	2022/10/14 14:50	3D Object
smshset.obj	2022/10/12 18:23	3D Object
chemconnectv.obj	2022/10/12 18:22	3D Object
lhscgsss.obj	2022/10/12 18:22	3D Object
solvecgsss.obj	2022/10/12 18:22	3D Object
chemconnectc.obj	2022/10/12 18:22	3D Object

上書き保存  
して更新

(参考) 更新前のSingleScrewSimulator.exeを  
別名でコピーしておく、または別の場所に  
コピーしておく、更新前の実行プログラム  
を残しておくことができます。

名前	更新日時	種類
SingleScrewSimulatorSolverver1200oneAPI_usr	2022/11/07 11:29	ファイル フォルダ
defaultoptionfile	2012/01/26 14:00	ファイル
expanto3d.exe	2022/06/21 17:52	アプリケーション
haslmodel.exe	2012/12/10 16:24	アプリケーション
mesh.exe	2022/06/22 8:25	アプリケーション
Option2	2021/12/23 17:45	ファイル
optionfile	2017/11/16 18:49	ファイル
SingleScrewSimulator.exe	2022/11/07 11:29	アプリケーション
SingleScrewSimulator_buk.exe	2022/11/07 9:58	アプリケーション
SSSGUISystem.exe	2022/11/07 9:17	アプリケーション



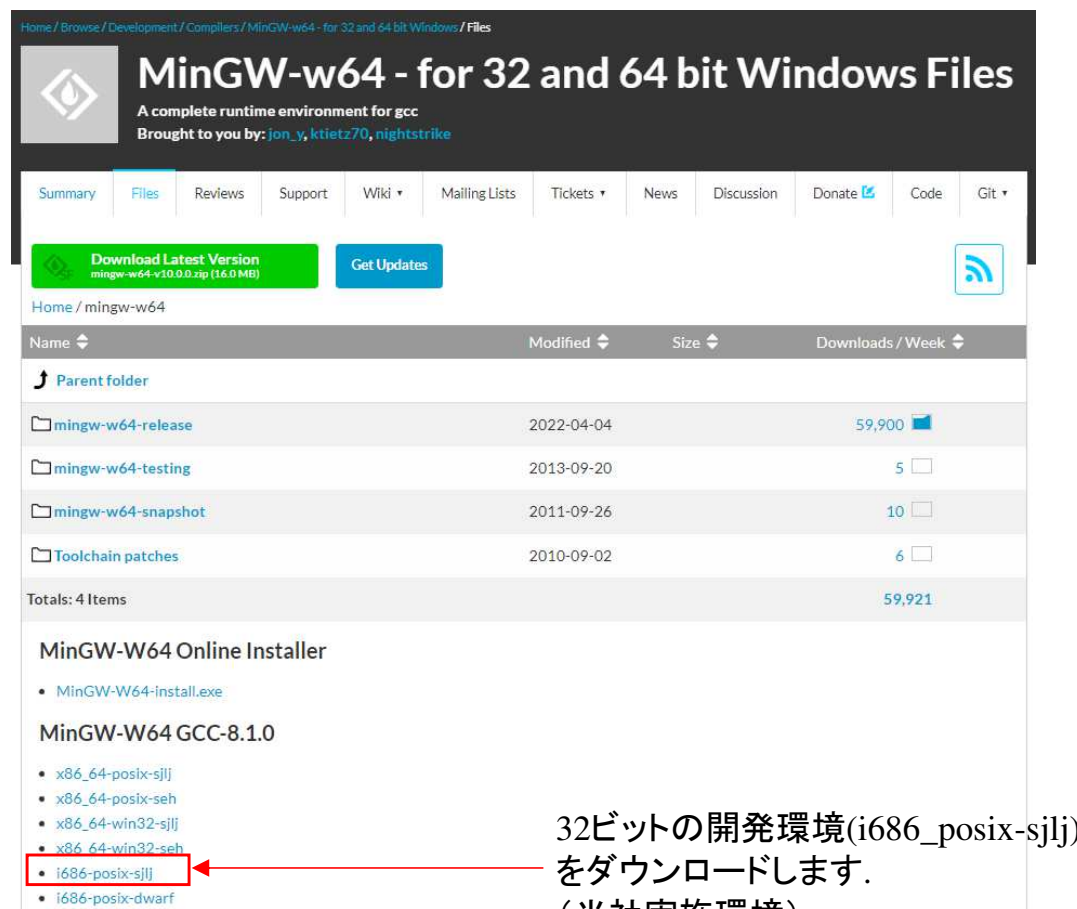
SSS(GUI)から解析実行すると、  
Systemver.12.0.0\_IFORT  
フォルダ内に存在する  
SingleScrewSimulator.exe  
が実行されます。

## 開発環境の設定例 ② GFortran

1) MinGW (Minimalist GNU for Windows, GUNコンパイラのWindows移植版) をダウンロードする.

入手可能なウェブサイト(下図に抜粋)

<https://sourceforge.net/projects/mingw-w64/files/mingw-w64/>



The screenshot shows the SourceForge page for MinGW-w64. The page title is "MinGW-w64 - for 32 and 64 bit Windows Files". Below the title, there is a navigation bar with links: Summary, Files, Reviews, Support, Wiki, Mailing Lists, Tickets, News, Discussion, Donate, Code, and Git. A green button labeled "Download Latest Version" and a blue button labeled "Get Updates" are visible. Below these buttons, there is a table listing files. The table has columns: Name, Modified, Size, and Downloads / Week. The table lists four items: "mingw-w64-release", "mingw-w64-testing", "mingw-w64-snapshot", and "Toolchain patches". Below the table, there is a section titled "MinGW-W64 Online Installer" with a link to "MinGW-W64-install.exe". Another section titled "MinGW-W64 GCC-8.1.0" lists several files: "x86\_64-posix-sjlj", "x86\_64-posix-seh", "x86\_64-win32-sjlj", "x86\_64-win32-seh", "i686-posix-sjlj", and "i686-posix-dwarf". A red arrow points to the "i686-posix-sjlj" file in the list.

Name	Modified	Size	Downloads / Week
Parent folder			
mingw-w64-release	2022-04-04		59,900
mingw-w64-testing	2013-09-20		5
mingw-w64-snapshot	2011-09-26		10
Toolchain patches	2010-09-02		6
Totals: 4 Items			59,921

MinGW-W64 Online Installer

- MinGW-W64-install.exe

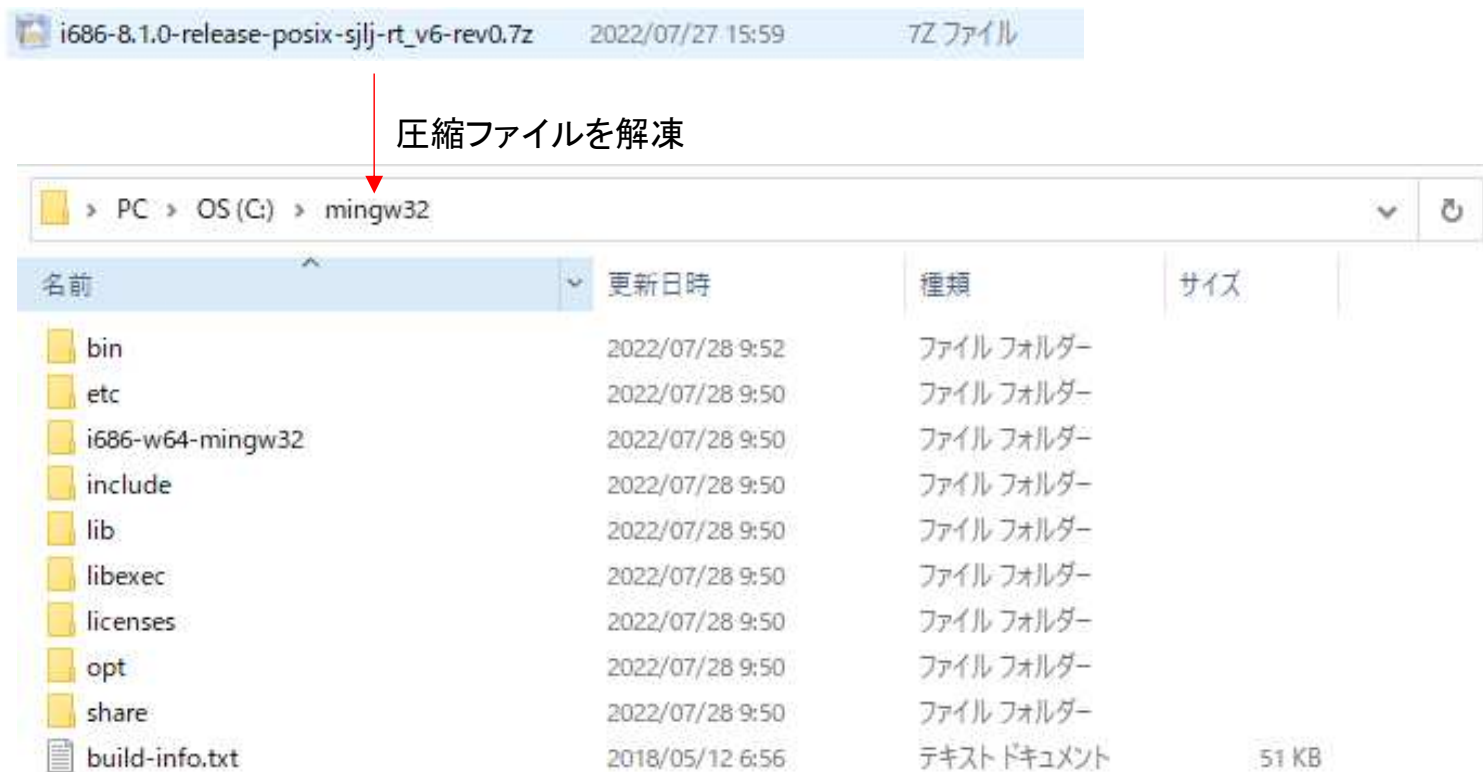
MinGW-W64 GCC-8.1.0

- x86\_64-posix-sjlj
- x86\_64-posix-seh
- x86\_64-win32-sjlj
- x86\_64-win32-seh
- i686-posix-sjlj
- i686-posix-dwarf

32ビットの開発環境(i686\_posix-sjlj)  
をダウンロードします。  
(当社実施環境)

## 開発環境の設定例 ② GFortran

2) ダウンロードした圧縮ファイル(.7z)を解凍すると, mingw32というフォルダが作成される.



## 開発環境の設定例 ② GFortran

- 3) mingw32 フォルダを任意の場所に移動し(注: 日本語を含むフォルダ名は不可), Windowsの環境変数の設定から, %mingw32%bin フォルダへの Path を設定する.

環境変数

yorif のユーザー環境変数(U)

変数	値
LIB	C:\Program Files (x86)\Microsoft Visual Studio .NET 2003\SDK\v1.1...
NETGEN_USER_DIR	C:\Users\yorif\AppData\Roaming\Netgen-5.0occ
NETGENDIR	C:\NETGEN_OCC\Netgen-5.0_x64\bin
OneDrive	C:\Users\yorif\OneDrive
OneDriveConsumer	C:\Users\yorif\OneDrive
Path	C:\Users\yorif\AppData\Local\Microsoft\WindowsApps;C:\VI;C:\m...
SMS_LICENSE_FILE	C:\SMSMFLICENSE\MF\license.txt

新規(N)... 編集(E)... 削除(D)

システム環境変数(S)

変数	値
ANS_OLD_ATTACH	1
ComSpec	C:\WINDOWS\system32\cmd.exe
CUDA_PATH	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2
CUDA_PATH_V10_2	C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.2
DriverData	C:\Windows\System32\Drivers\DriverData
I_MPI_ONEAPI_ROOT	C:\Program Files (x86)\Intel\oneAPI\mpi\2021.6.0
IDB_PATH	C:\Program Files (x86)\Intel\

新規(W)... 編集(I)... 削除(L)

OK キャンセル

環境変数名の編集

%USERPROFILE%\AppData\Local\Microsoft\WindowsApps

C:\VI

C:\mingw64\bin

C:\mingw32\bin

(参考) 当社実施環境での設定例  
Cドライブ直下にmingw32を移動

> PC > OS (C:) > mingw32

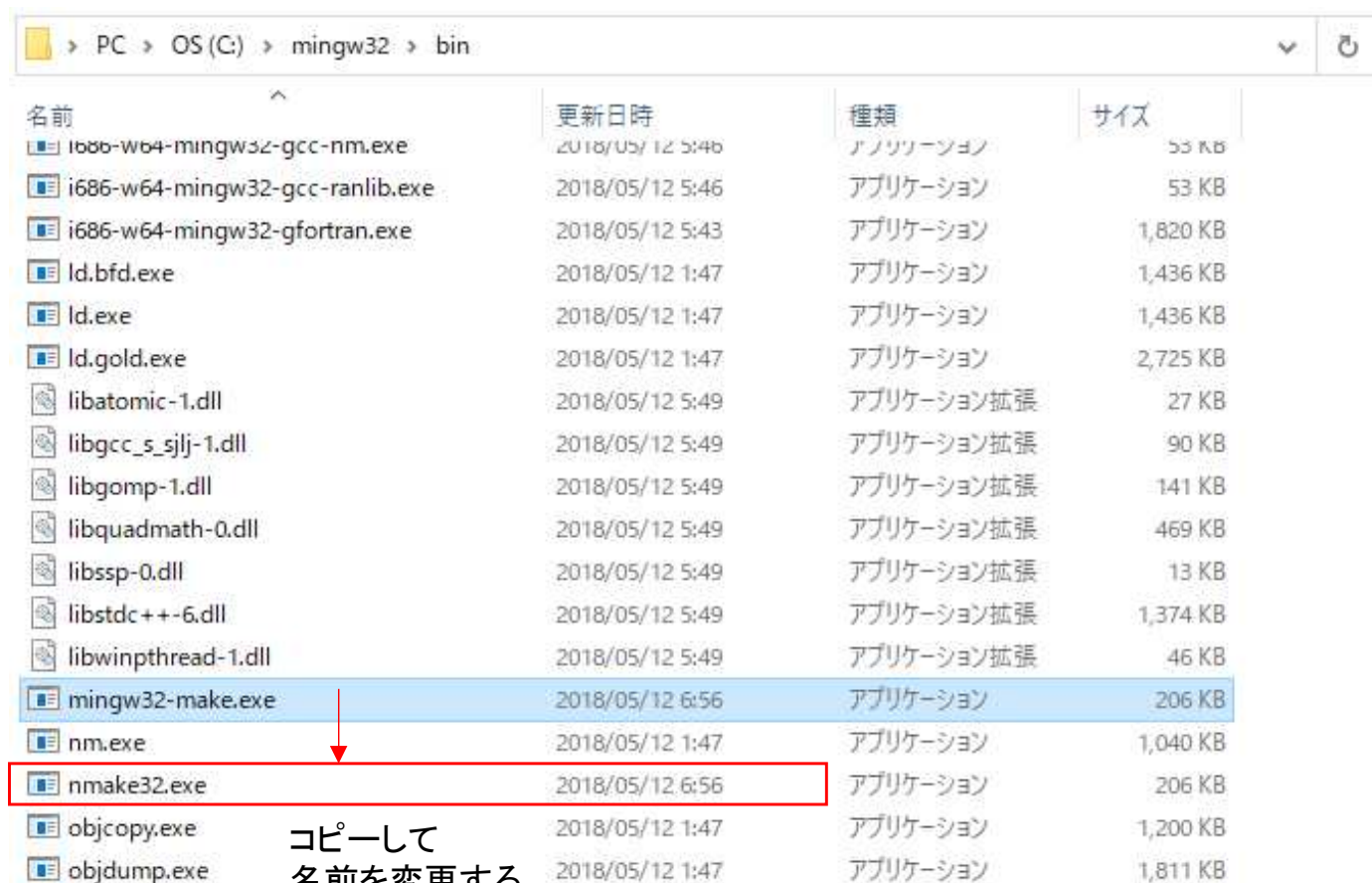
名前	更新日時
bin	2022/07/28 9:52
etc	2022/07/28 9:50
i686-w64-mingw32	2022/07/28 9:50

新規(N) 編集(E) 参照(B)... 削除(D) 上へ(U) 下へ(O) テキストの編集(T)...

OK キャンセル

## 開発環境の設定例 ② GFortran

- 4) ¥mingw32¥bin フォルダ内の makefile を用いたコンパイル実行プログラム mingw32-make.exe をフォルダ内にコピーし, nmake32.exe に名前を変更する. ここまでがMinGwのインストール手順例です.



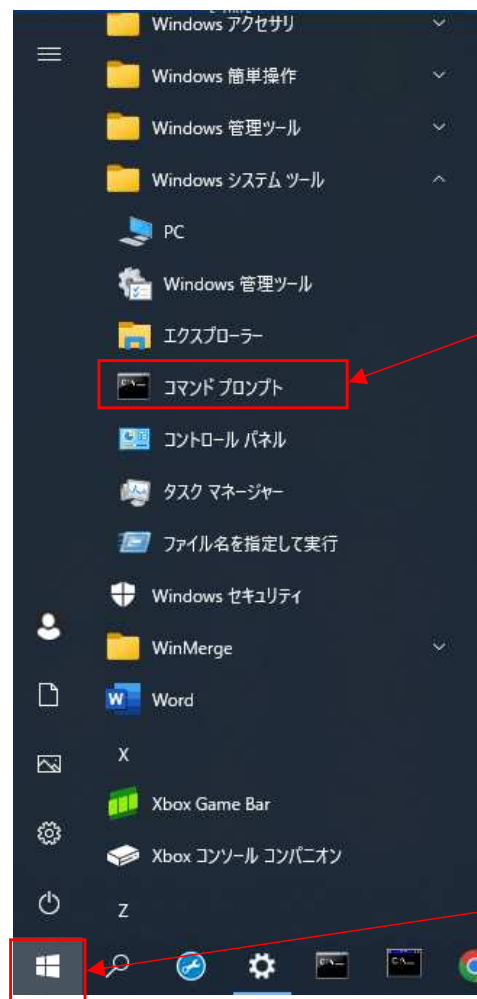
名前	更新日時	種類	サイズ
i686-w64-mingw32-gcc-nm.exe	2018/05/12 5:46	アプリケーション	53 KB
i686-w64-mingw32-gcc-ranlib.exe	2018/05/12 5:46	アプリケーション	53 KB
i686-w64-mingw32-gfortran.exe	2018/05/12 5:43	アプリケーション	1,820 KB
ld.bfd.exe	2018/05/12 1:47	アプリケーション	1,436 KB
ld.exe	2018/05/12 1:47	アプリケーション	1,436 KB
ld.gold.exe	2018/05/12 1:47	アプリケーション	2,725 KB
libatomic-1.dll	2018/05/12 5:49	アプリケーション拡張	27 KB
libgcc_s_sjlj-1.dll	2018/05/12 5:49	アプリケーション拡張	90 KB
libgomp-1.dll	2018/05/12 5:49	アプリケーション拡張	141 KB
libquadmath-0.dll	2018/05/12 5:49	アプリケーション拡張	469 KB
libssp-0.dll	2018/05/12 5:49	アプリケーション拡張	13 KB
libstdc++-6.dll	2018/05/12 5:49	アプリケーション拡張	1,374 KB
libwinpthread-1.dll	2018/05/12 5:49	アプリケーション拡張	46 KB
mingw32-make.exe	2018/05/12 6:56	アプリケーション	206 KB
nm.exe	2018/05/12 1:47	アプリケーション	1,040 KB
nmake32.exe	2018/05/12 6:56	アプリケーション	206 KB
objcopy.exe	2018/05/12 1:47	アプリケーション	1,200 KB
objdump.exe	2018/05/12 1:47	アプリケーション	1,811 KB

コピーして  
名前を変更する。  
(名前は任意)



## 開発環境の設定例 ② GFortran

5) MinGw をインストール後, Windows 画面左下のスタートボタンから Windows システムツール内のコマンドプロンプトを起動します.



2. コマンドプロンプト  
をクリックする.

1. Windowsスタートボタン  
をクリックする.  
(当社実施環境 windows10)

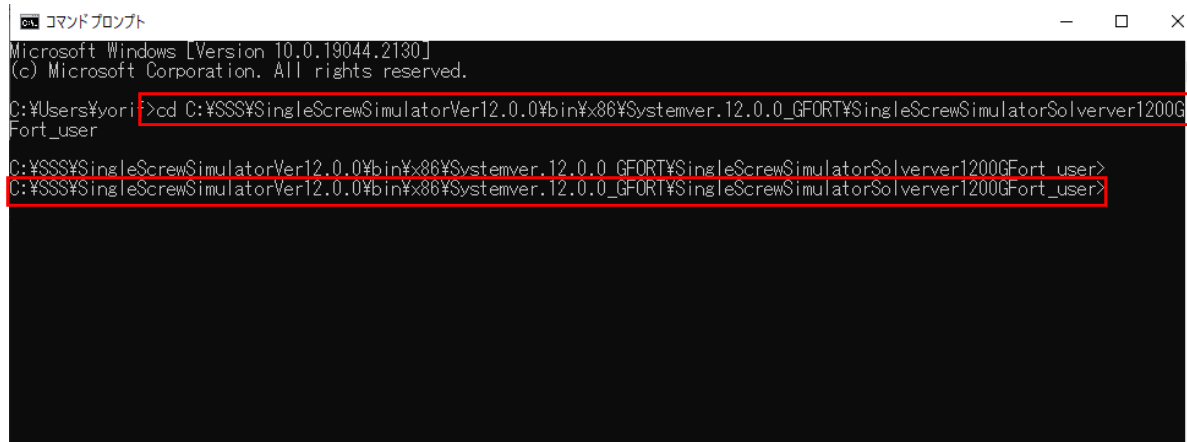
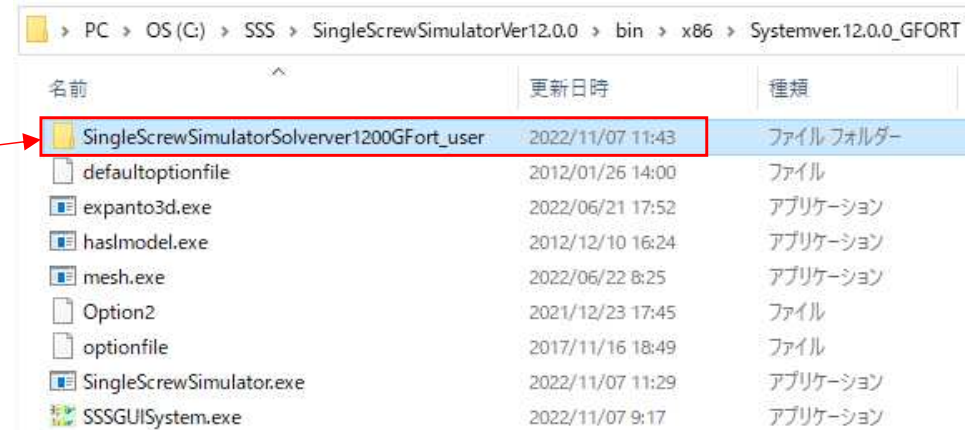
3. コマンドプロンプトが起動する.





## 開発環境の設定例 ② GFortran

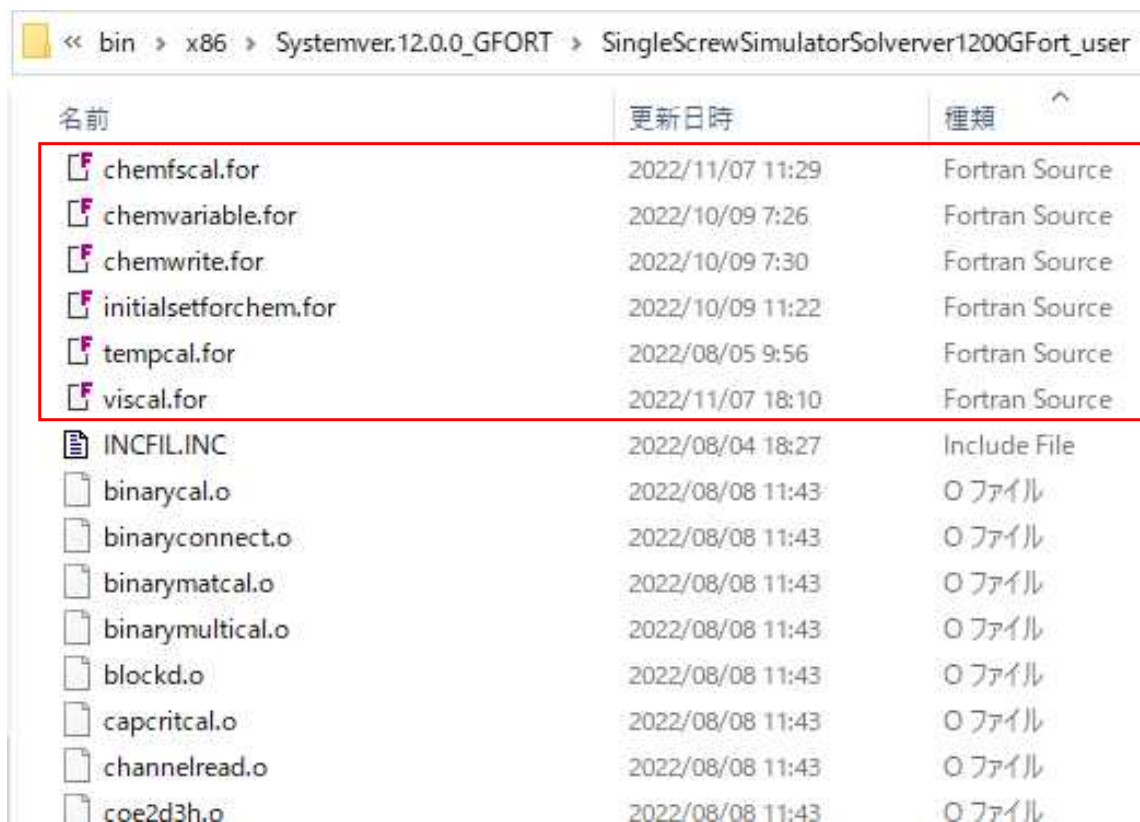
6) コマンドプロンプト上で, GFortran用ユーザーチンのソースコードが格納されているTSS内のフォルダ (¥bin¥x86¥Systemver.9.0.0\_GFORT¥ SingleScrewSimulatorSolverver1200GFort\_user) に移動します.



cd (ディレクトリ移動コマンド) の後に,  
¥SingleScrewSimulatorSolverver1200GFort\_user  
までのパス(C:ドライブからの絶対パス)  
を入力して, キーボードのEnterキーを  
押すと, カレントディレクトリが 変更されます.  
(注: 日本語を含むフォルダ名は不可)

## 開発環境の設定例 ② GFortran

- 7) ¥SingleScrewSimulatorSolver1200GFort\_user 内に存在するソースコードを用途向きに書き直します  
(任意のエディタを使用)。



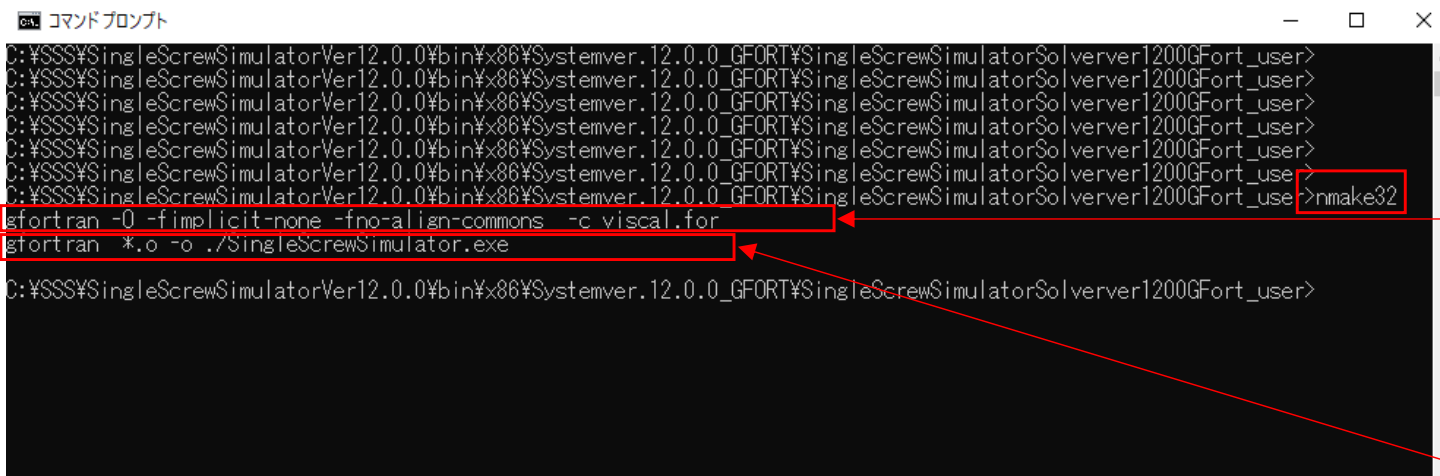
名前	更新日時	種類
chemfscal.for	2022/11/07 11:29	Fortran Source
chemvariable.for	2022/10/09 7:26	Fortran Source
chemwrite.for	2022/10/09 7:30	Fortran Source
initialsetforchem.for	2022/10/09 11:22	Fortran Source
tempcal.for	2022/08/05 9:56	Fortran Source
viscal.for	2022/11/07 18:10	Fortran Source
INCFILE.INC	2022/08/04 18:27	Include File
binarycal.o	2022/08/08 11:43	O ファイル
binaryconnect.o	2022/08/08 11:43	O ファイル
binarymatcal.o	2022/08/08 11:43	O ファイル
binarymultical.o	2022/08/08 11:43	O ファイル
blockd.o	2022/08/08 11:43	O ファイル
capcritical.o	2022/08/08 11:43	O ファイル
channelread.o	2022/08/08 11:43	O ファイル
coe2d3h.o	2022/08/08 11:43	O ファイル

ユーザがカスタマイズ可能な  
サブルーチンのソースコード  
(ファイル名.for)

(注)ソースコード以外に格納されている  
ファイル(.o など)を変更したり  
削除するとコンパイルができなく  
なりますので注意して使用ください。

## 開発環境の設定例 ② GFortran

- 8) ソースコードを編集後、コマンドプロンプト上で `nmake32` (p.12 の4) で変更した名前) と入力してキーボードの Enterキーを押すと、makefile を利用したコンパイルが実行されます。



```
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>nmake32
gfortran -O -fimplicit-none -fno-align-commons -c viscal.for
gfortran *.o -o ./SingleScrewSimulator.exe
C:\SSSS\SingleScrewSimulatorVer12.0.0\bin\x86\Systemver.12.0.0_GFORT\SingleScrewSimulatorSolver1200GFort_user>
```

変更したソースコードのみ  
コンパイルされます。

コンパイルに成功すると、  
フォルダ内に実行プログラム  
SingleScrewSimulator.exe  
が作成されます。

コンパイルに失敗した場合は  
エラーメッセージが出力される  
ので、ソースコードを修正  
後に 再度 `nmake32` を実施して  
ください。

## 開発環境の設定例 ② GFortran

- 9) ¥SingleScrewSimulatorSolverver1200GFort\_user フォルダ内の SingleScrewSimulator.exe の更新日時がコンパイルした日時に変更されていることを確認後, Systemver.12.0.0\_GFORT フォルダ内に存在する SingleScrewSimulator.exe を上書き保存して更新すると, SSSのGUIから解析実行した際に更新された SingleScrewSimulator.exeが実行されます。

名前	更新日時	種類
SingleScrewSimulator.exe	2022/11/09 10:05	アプリケーション
viscal.o	2022/11/09 10:05	オブジェクトファイル
viscal.for	2022/11/09 10:05	Fortran Source
viscal.for~	2022/11/09 10:04	FORTRAN ファイル
Makefile	2022/11/07 18:09	ファイル
Makefile~	2022/11/07 18:09	ファイル
chemfscal.for	2022/11/07 11:29	Fortran Source
initialsetforchem.for	2022/10/09 11:22	Fortran Source
chemwrite.for	2022/10/09 7:30	Fortran Source
chemvariable.for	2022/10/09 7:26	Fortran Source

名前	更新日時
SingleScrewSimulatorSolverver1200GFort_user	2022/11/09 10:05
defaultoptionfile	2012/01/26 14:00
expanto3d.exe	2022/06/21 17:52
haslmodel.exe	2012/12/10 16:24
mesh.exe	2022/06/22 8:25
Option2	2021/12/23 17:45
optionfile	2017/11/16 18:49
SingleScrewSimulator.exe	2022/11/09 10:05
SingleScrewSimulator_buk.exe	2022/11/07 11:29
SSSGUISystem.exe	2022/11/07 9:17

(参考) 更新前のSingleScrewSimulator.exe を別名でコピーしておく, または別の場所にコピーしておく, 更新前の実行プログラムを残しておくことができます。



SSS(GUI)から解析実行すると, Systemver.12.0.0\_IFORT フォルダ内に存在する SingleScrewSimulator.exe が実行されます。

表1 ユーザが自由にカスタマイズ可能なサブルーチン

ユーザ定義ルーチン名	機能
1) initialsetforchem	化学反応種の初期設定及び解析条件の設定
2) viscal	粘度計算
3) tempcal	温度計算
4) chemfscal	化学反応式の左辺及び右辺荷重ベクトルの設定
5) chemvariable	化学反応成分の代数的関係式の計算
6) chemwrite	化学反応解析結果のファイル出力

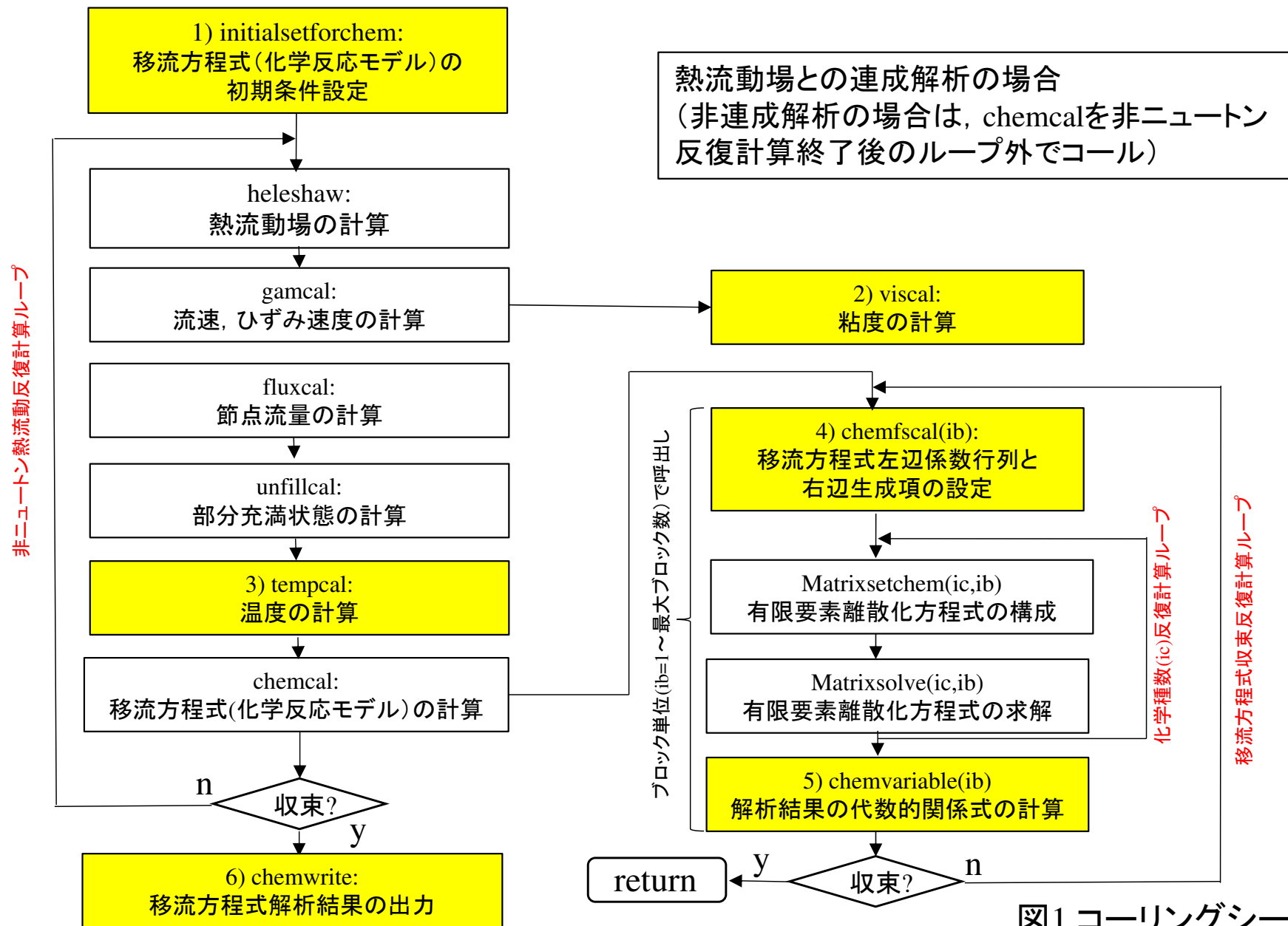


図1 コーリングシーケンス

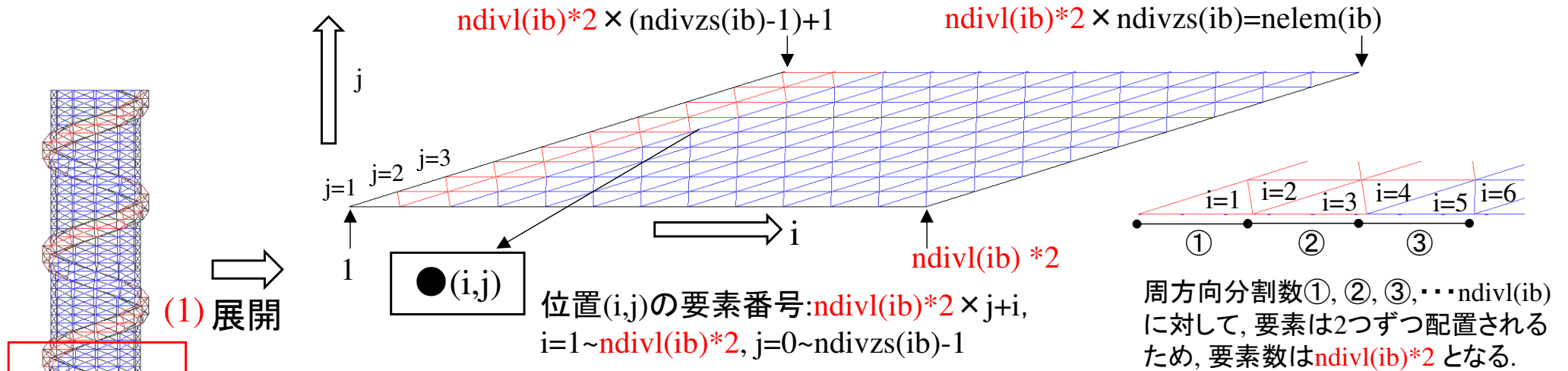


表2 解析モデル要素情報 (ib: ブロック数)

変数名	内容
nelem(ib)	要素数
ndivl(ib)	周方向要素分割数
ndivzs(ib)	軸方向要素分割数
ibtype(ib)	スクリュモデル定義 (0: 標準定義, 1: 矩形領域定義)
inqc(ie,ib)	要素構成節点数 (SSSは三角形要素で構成されるため 3 )
ndiv	肉厚方向の要素分割数
ientab(ie,ib)	要素ieを構成する節点番号リスト(ローカル)
entab(ientab(ie,ib),ib)	要素ieを構成する節点番号リスト(グローバル) (例) 要素ieを構成する節点番号は以下で抽出できる. entab(ientab(ie,ib)+ii-1,ib), ii=1 ~ 3(inqc(ie,ib))
height(ie,ib)	要素ieの肉厚(ie=1~nelem(ib),単位cm)
vol(ie,ib)	要素ieの体積(ie=1~nelem(ib),単位cm <sup>3</sup> )

## 要素情報

### (1) 標準定義(フライトあり)



### (2) 矩形領域定義

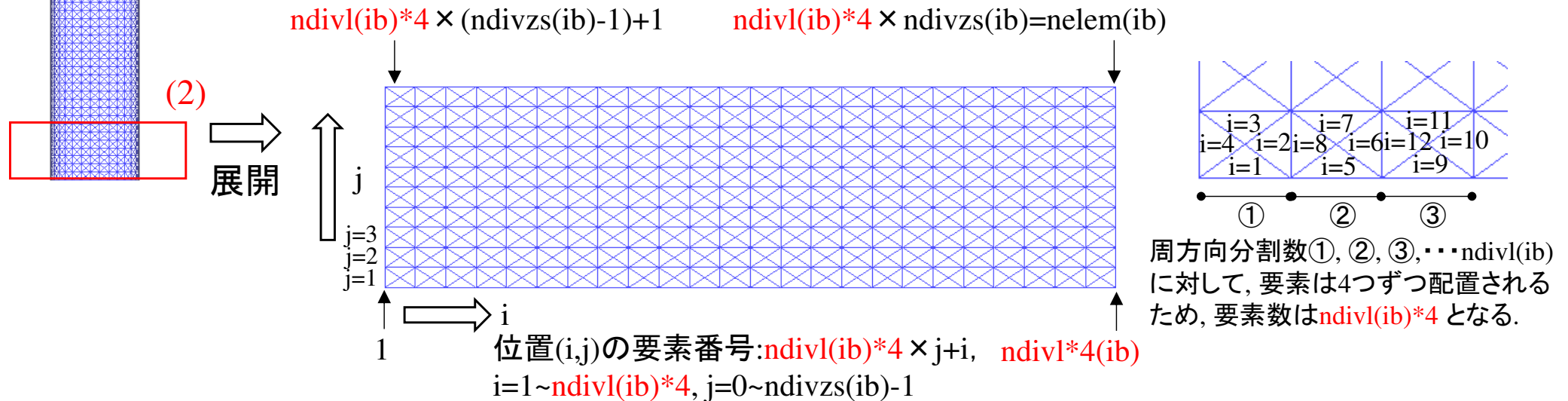
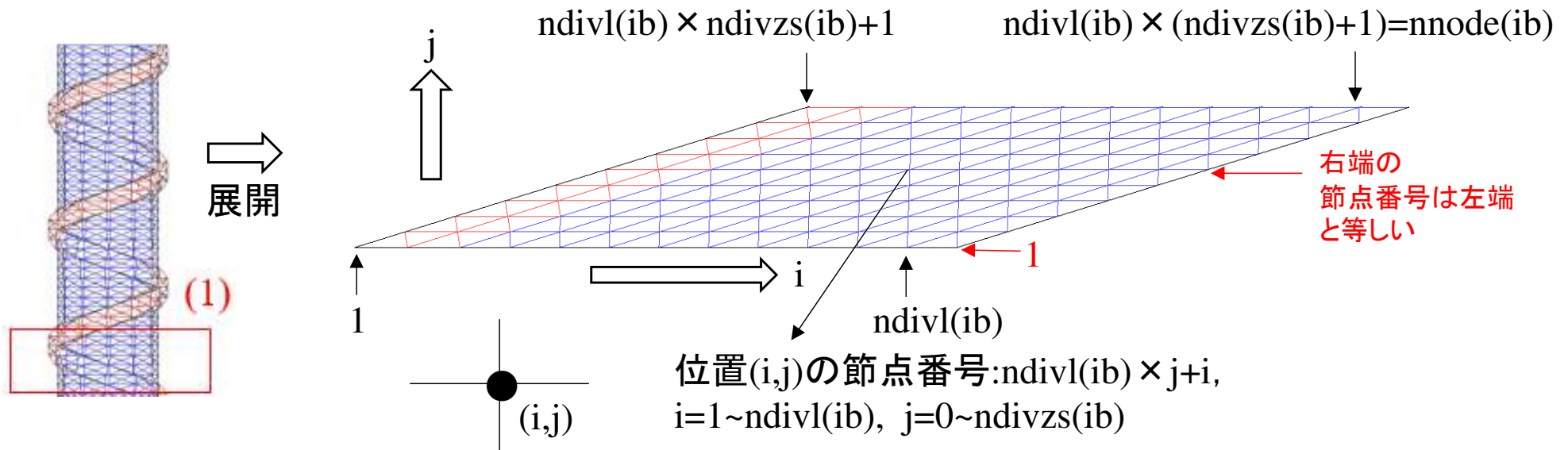


表3 解析モデル節点情報 (ib: ブロック数)

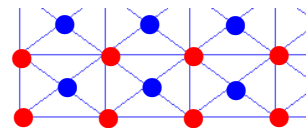
変数名	内容
nnode(ib)	節点数
xnode(in,ib),ynode(in,ib),znode(in,ib)	節点inのx,y,z座標(in=1~nnode(ib),単位cm)
nren(in,ib)	節点inを含む(隣接する)要素数
nrelem(inrelem(in,ib)+ii-1,1)	節点inを含む(隣接する)要素番号 (ii=1~nren(in,ib))

(1) 標準定義(フライトあり)



## 節点情報

### (2) 矩形領域定義



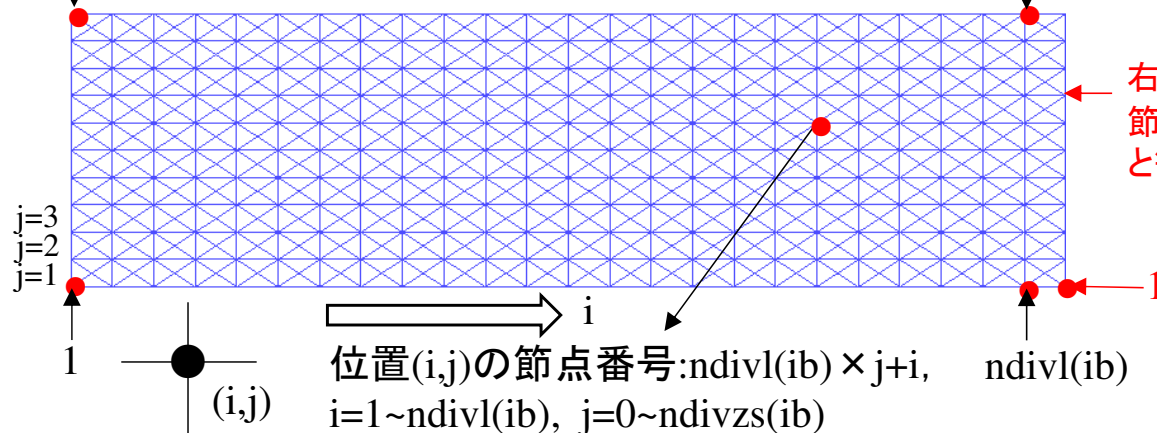
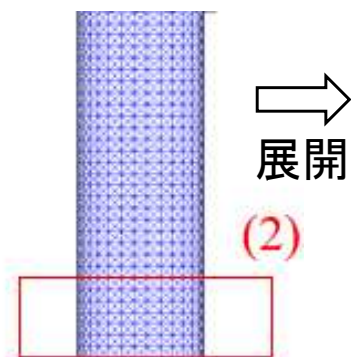
外周節点(●)と内部節点(●)に分けて、  
以下のように定義されます。

#### ・外周節点(●)

$$\text{ndivl(ib)} \times \text{ndivzs(ib)} + 1$$

外周節点の総数

$$\text{ndivl(ib)} \times (\text{ndivzs(ib)} + 1)$$



右端の  
節点番号は左端  
と等しい

位置(i,j)の節点番号:  $\text{ndivl(ib)} \times j + i$ ,  
 $i=1 \sim \text{ndivl(ib)}$ ,  $j=0 \sim \text{ndivzs(ib)}$

#### ・内部節点(●)

$$\text{ndivl(ib)} \times (\text{ndivzs(ib)} + 1) + \text{ndivl(ib)} \times (\text{ndivzs(ib)} - 1) + 1$$

外周節点の総数

内部節点の総数

$$\text{ndivl(ib)} \times (\text{ndivzs(ib)} + 1) + \text{ndivl(ib)} \times \text{ndivzs(ib)} = \text{nnode(ib)}$$

位置(i,j)の節点番号:  
 $\text{ndivl(ib)} \times (\text{ndivzs(ib)} + 1)$   
 $+ \text{ndivl(ib)} \times j + i$ ,  
 $i=1 \sim \text{ndivl(ib)}$ ,  $j=1 \sim \text{ndivzs(ib)}$

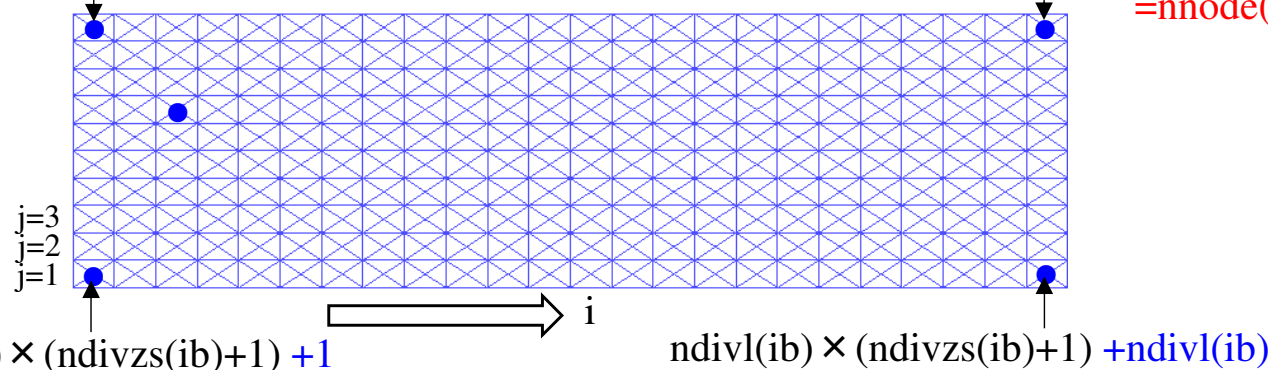


表4 境界条件,成形条件情報 (ib: ブロック数)

変数名	内容
tempinlet	流入温度(°C)
pprescribedinlet	流入口設定圧力(MPa)
pprescribed	流出口圧力(MPa)
qprescribed	押出量(kg/h)
flowrateinlet	流入口設定流量(cc/s)
rpm	スクリュ回転数(rpm)
href(1,ie,ib)	要素ieのスクリュ面熱伝達係数(W/cm <sup>2</sup> /K)
tbound(1,ie,ib)	要素ieのスクリュ面境界温度(°C)
href(2,ie,ib)	要素ieのバレル面熱伝達係数(W/cm <sup>2</sup> /K)
tbound(2,ie,ib)	要素ieのバレル面境界温度(°C)
iboundt(in,ib)	節点inの温度境界条件(0:拘束, 非0:自由)
ibound(in,ib)	節点inの圧力境界条件(0:拘束, 非0:自由)

表5 物性情報 (ib: ブロック数)

変数名	内容
rhoh(i,ie,ib)	要素ieの層(i=1~ndiv+1)の密度(g/cm <sup>3</sup> )
cph(i,ie,ib)	要素ieの層(i=1~ndiv+1)の比熱(J/g/K)
vish(i,ie,ib)	要素ieの層(i=1~ndiv+1)の粘度(Pa・s)

i=1 : スクリュ表面, i=ndiv+1 : バレル表面

表6 解析結果要素情報 (ib: ブロック数)

変数名	内容
temh(i,ie,ib)	要素ieの層(i=1~ndiv+1)の温度(°C)
gam(i,ie,ib)	要素ieの層(i=1~ndiv+1)のひずみ速度(J/g/K)
uh(i,ie,ib),vh(i,ie,ib),wh(i,ie,ib)	要素ieの層(i=1~ndiv+1)の流速ベクトル成分(cm/s)
tempe(ie,ib)	要素ieの温度
fille(ie,ib)	要素ieの充満率
filleavb(ie,ib)	要素ieの平均充満率(濃度計算用)



表7 解析結果節点情報 (ib: ブロック数)

変数名	内容
pres(in,ib)	節点inの圧力(Pa)
temp(in,ib)	節点inの温度(°C)
visn(in,ib)	節点inの粘度(Pa・s)
gamn(in,ib)	節点inのひずみ速度(s <sup>-1</sup> )

表8 ユーザ定義変数 (ib: ブロック数)

変数名	内容
chemcnumber	解析対象とする移流方程式の本数(化学種数)
chemvnumber	解析で考慮する配列変数の数
commonvnumber	ルーチン間で共用するスカラー変数の数
chemcname(i)	化学種の名称 (i=1~chemcnumber)
chemvname(i)	配列変数の名称 (i=1~chemvnumber)
chemc(i,ie,ib)	化学種の要素濃度 (i=1~chemcnumber, ie=1~nelem(ib))
chemcn(i,in,ib)	化学種の節点濃度 (i=1~chemcnumber, ie=1~nelem(ib))
chempar(i,ie,ib)	化学種依存の要素変数 (i=1~chemvnumber, ie=1~nelem(ib))
chemparn(i,in,ib)	化学種依存の節点変数 (i=1~chemvnumber, ie=1~nelem(ib))
commonvpar(i)	ルーチン間で共用するスカラー変数 (i=1~commonvnumber)

# サンプルプログラム

## ① 濃度計算(フォルダ名:conc)

### Initialsetforchem の内容

```
c+++++
c+   User define variable number
c+++++
c
c   Number of chemical species
c   chemcnumber=1  ← 単一成分濃度
c
c   Number of chemical variable
c   chemvnumber=0
c
c   Number of common variable
c   commonvnumber=0
c+++++
```

```
c+++++
c   User define initial/boundary condition
c+++++
c   chemcname(1)='test sample 1' ← ポストタイトル
c
c   do ib=1,iblock ← 全ブロックに設定
c
c   do ie=1,nelem(ib) } ← 要素濃度の初期設定
c   chemc(1,ie,ib)=0.0
c   end do
c
c   do in=1,nnode(ib) } ← 節点濃度の初期設定
c   chemcn(1,in,ib)=0.0
c   end do
c
c   if(ib.eq.1) then ← 1ブロック目に流入境界条件を設定
c
c   qsumtotal=0.0
c   do in=1,ndivl(ib) } ← 流入口に設定されている全流量の計算
c   qsumtotal=qsumtotal+qfluxb(in,ib)
c   end do
c
c   qsumhalf=0.0
c   do in=1,ndivl(ib)/2 } ← 流入口境界
c   qsumhalf=qsumhalf+qfluxb(in,ib) } (in=1~iinletnodelist/2)
c   end do } の半領域に設定されている流量の計算
c
c   do ii=1,ndivl(ib)/2 } ← 流入口境界
c   chemcn(1,ii,ib)=0.5*qsumtotal/qsumhalf } (in=1~iinletnodelist/2)の
c   end do } 半領域に濃度を設定, 残
c   end if } りの半領域の濃度は0
c   end do } (当設定によって濃度平均
c+++++ } 値は0.5に漸近)
```

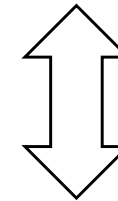
## chemfscal の内容

subroutine chemfscal(ib) ← ブロック毎に呼出される

```
c+++++
c+  User define left hand side coefficient & right hand source
c+++++
c  do ie=1,nlem(ib)
c
c    chemf(1,ie,ib)=0.0
c    chems(1,ie,ib)=0.0
c
c  end do
c+++++
```

← 比較

$$\mathbf{u} \cdot \nabla f = 0$$



$$(A_i + \mathbf{u} \cdot \nabla) f_i = B_i$$

TSSが解析対象とする移流方程式:

$$(chemf(ic, ie, ib) + \mathbf{u} \cdot \nabla) f(ic, ie, ib) = chems(ic, ie, ib)$$

$$ic=1, f(1, ie)=chemc(1, ie, ib)$$

$$chemf(1, ie, ib)=0.0$$

$$chems(1, ie, ib)=0.0$$

$A_i, B_i$ : ユーザ定義任意関数 ( $i=1 \sim n$ )

$f_i$ : ユーザ定義未知関数 ( $i=1 \sim n$ )

$n$ : ユーザ定義方程式数

$\mathbf{u}$ : 流速ベクトル

$\nabla$ : ナブラ演算子

## User define model タブメニューの設定

Single Screw Simulator Template

スクリュ形状 ダイ形状 ホッパー形状 押出機形状 解析プログラム実行 解析結果統括表 ユーザ定義解析

☒ ユーザ定義ルーチンの利用/RTD解析 (デフォルト) ← ユーザ定義ルーチンのコール

☐ 熱流動解析との連成 ← 当濃度計算は, 熱流動解析に影響を及ぼさないため, 非連成 (チェックボックスを非チェック)

計算パラメータ

反復計算回数	1	← 線形方程式の定常解析では, 反復計算を要しないため 1 に設定
反復計算の緩和係数	1	
マトリクスソルバ反復計算回数	50000	
マトリクスソルバの収束基準値	1E-06	
RTD解析の時間刻み	0.2	

## ② 滞留時間分布(RTD)計算(フォルダ名:rtd)

### Initialsetforchem の内容

```
c+++++
c+   User define variable number
c+++++
c   Number of chemical species
c
c   chemcnnumber=1 ← 単一成分濃度
c
c   Number of chemical variable
c
c   chemvnumber=0
c
c   Number of common variable ← 非定常計算刻みとシミュレ
c                               ション時間として使用する
c   commonvnumber=2 ← 変数の数
c+++++
```

### 各種配列変数のダイナミックアロケーション (ユーザ定義不要の固定構文)



```
c
c   Definition of common number variable
c
c   commonvpar(1)=rtdtime ← 時間刻み:0.2 sec, GUIから入力可能
c   commonvpar(2)=0.0 ← シミュレーション時刻
```

```
c+++++
c   User define initial/boundary condition
c+++++
c   chemcname(1)='test sample 1' ← ポストタイトル
c
c   do ib=1,iblock ← 全ブロックに設定
c
c   do ie=1,nelem(ib)
c   chemc(1,ie,ib)=0.0 } ← 要素濃度の初期設定
c   end do
c
c   do in=1,nnode(ib)
c   chemcn(1,in,ib)=0.0 } ← 節点濃度の初期設定
c   end do
c
c   if(ib.eq.1) then ← 1ブロック目を指定
c
c   do ii=1,ndivl(ib)
c   chemcn(1,ii,ib)=1.0 } ← 流入口に配置された節
c                           点に対する境界条件
c                           (濃度:1) の設定
c   end do
c
c   end if
c
c   end do
c+++++
```



## chemfscal の内容

subroutine chemfscal(ib) ← ブロック毎に呼出される

```

C+++++
C+   User define left hand side coefficient & right hand source
C+++++
      do ie=1,nelem(ib)
C
C→  chemf(1,ie,ib)=1.0/commonvpar(1)
      chems(1,ie,ib)=chemc(1,ie,ib)/commonvpar(1) ←
C
      end do
C+++++

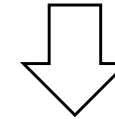
```

$ic=1, f(1,ie,ib)=chemc(1,ie,ib), \Delta t=commonvpar(1)$

$chemf(1,ie,ib)=1/\Delta t=1/commonvpar(1)$

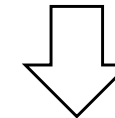
$chems(1,ie,ib)=f^{n-1}/\Delta t=chemc(1,ie,ib)/commonvpar(1)$

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = 0$$

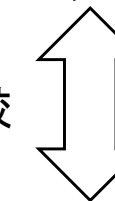


後退差分

$$\frac{f^n - f^{n-1}}{\Delta t} + \mathbf{u} \cdot \nabla f^n = 0$$



$$\left( \frac{1}{\Delta t} + \mathbf{u} \cdot \nabla \right) f^n = \frac{1}{\Delta t} f^{n-1}$$



比較

TSSが解析対象とする移流方程式:

$$(\text{chemf}(ic, ie, ib) + \mathbf{u} \cdot \nabla) f(ic, ie, ib) = \text{chems}(ic, ie, ib)$$

## chemvariable の内容

subroutine chemvariable(ib) ← ブロック毎に呼出される

```
volout=0.0
csum=0.0
```

```
iz=ndivzs(ib)-4 ← 流出口境界条件の影響を緩和するための境界層数: 4
```

```
if(ibtype(ib).eq.0) then ← ブロック ib のスクリュメッシュが標準定義の場合
```

```
do i=1,2*ndivl(ib) ← 周方向要素数分のループ
  ie=i+iz*2*ndivl(ib) ← 流出口境界位置の要素番号に変換
  volout=volout+filleavb(ie,ib)*vol(ie,ib)
  csum=csum+filleavb(ie,ib)*vol(ie,ib)*chemc(1,ie,ib)
end do
```

```
else ← ブロック ib のスクリュメッシュが矩形領域定義の場合
```

```
do i=1,4*ndivl(ib) ← 周方向要素数分のループ
  ie=i+iz*4*ndivl(ib) ← 流出口境界位置の要素番号に変換
  volout=volout+filleavb(ie,ib)*vol(ie,ib)
  csum=csum+filleavb(ie,ib)*vol(ie,ib)*chemc(1,ie,ib)
end do
```

```
end if
```

流出口境界で計算される濃度の周方向体積重み付け  
平均値:  $csum/volsum$  の計算

filleavb(ie,ib): ブロックibの要素ieの充満率

vol(ie,ib): ブロックibの要素ieの体積

chemc(1,ie,ib): ブロックibの要素ieの濃度

$$volsum = \sum_{ie=ns}^{ne} filleavb(ie,ib) * vol(ie,ib)$$

$$csum = \sum_{ie=ns}^{ne} filleavb(ie,ib) * vol(ie,ib) * chemc(1,ie,ib)$$

スクリュ形状    ダイ形状    ホッパー形状    押出機形状

ブロック数

削除対象番号

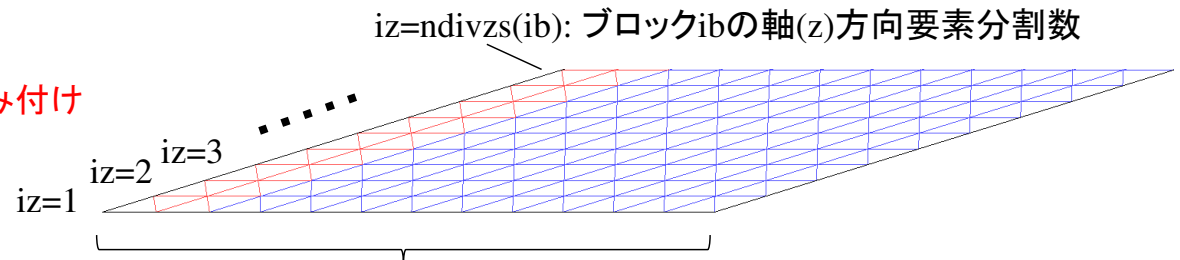
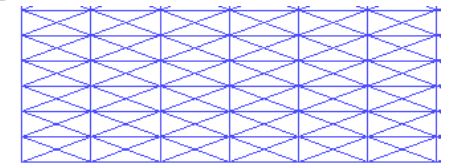
パレル直径  mm

スクリュ形状基本パラメータ

ゾーン数

フライト幅方向分割数

☐ 矩形領域定義    ☐ 逆ねじ



ndivl(ib): ブロックibの周方向要素分割数

標準定義の場合(ibtype(ib)=0), 周方向要素数は  $2*ndivl(ib)$

矩形領域定義の場合(ibtype(ib)=1), 周方向要素数は  $4*ndivl(ib)$

## chemvariable の内容(続き)

```
if(ib.eq.1) then  
commonvpar(2)=commonvpar(2)+commonvpar(1)  
end if
```

← 1ブロック目の場合  
←  $t=t+\Delta t$

```
if(ib.eq.iblock) then  
write(99,*) ib,',',commonvpar(2),',',csum/volout  
end if
```

← スクリュモデルの最終ブロック(iblock)の場合

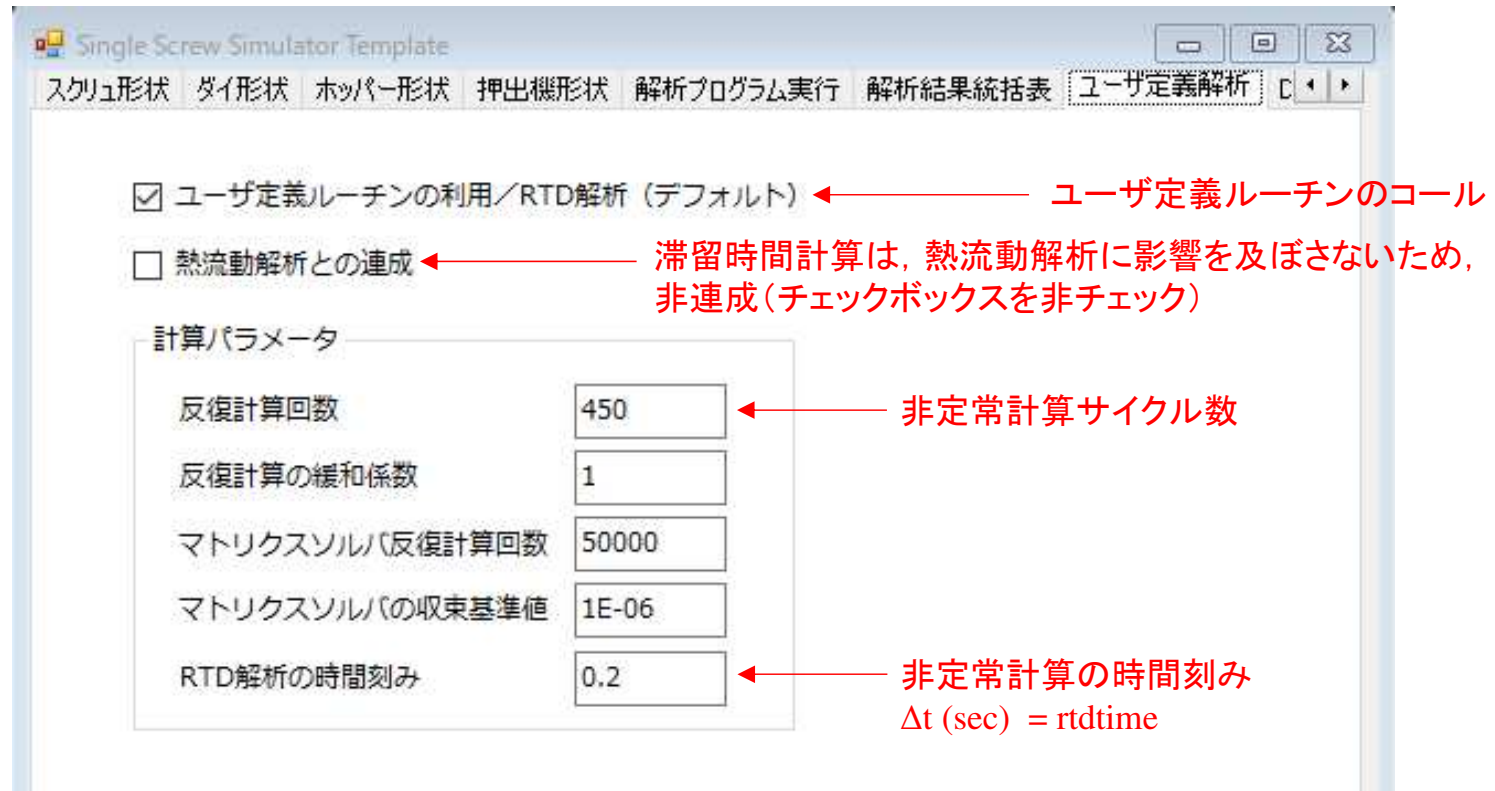
```
write(*,*) ib,',',commonvpar(2),',',csum/volout
```

↑  
実行ウィンドウ(出力番号:\*)へのシミュレーション時刻と流出口平均濃度のエコープリント

cheminf ファイル (出力番号:99)へのシミュレーション時刻と流出口平均濃度のエコープリント

cheminf ファイルの出力内容は, 解析終了後に解析結果ファイル名.rtd に自動コピーされます.

## User define model タブメニューの設定



## SSS出力ファイル

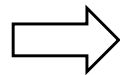
(xxx.rtd, xxx:プロジェクト名; rtd\*\*)

File Edit Format View Help

rtd\_straight.rtd - メモ帳

3	0.2000000	0.0000000E+00
3	0.4000000	0.0000000E+00
3	0.6000000	0.0000000E+00
3	0.8000000	1.4012985E-45
3	1.0000000	1.8216880E-44
3	1.2000000	2.9427268E-43
3	1.4000000	3.9698785E-42
3	1.6000000	4.6304506E-41
3	1.8000000	4.7561331E-40
3	2.0000000	4.3710381E-39
3	2.2000000	3.6392262E-38
3	2.4000000	2.7726071E-37
3	2.6000000	1.9490346E-36
3	2.8000000	1.2729759E-35
3	3.0000000	7.7706462E-35
3	3.200001	4.4559510E-34
3	3.400001	2.4109729E-33
3	3.600001	1.2356626E-32
3	3.800001	6.0194333E-32
3	4.000000	2.7956876E-31
3	4.200000	1.2413390E-30
3	4.400000	5.2824407E-30
3	4.600000	2.1592047E-29
3	4.800000	8.4947955E-29
3	5.000000	3.2227038E-28
3	5.199999	1.1809667E-27

コンマ区切りで  
エクセルに読み込み



流出口(スクリュ出口)  
のブロック番号

シミュレーション時刻

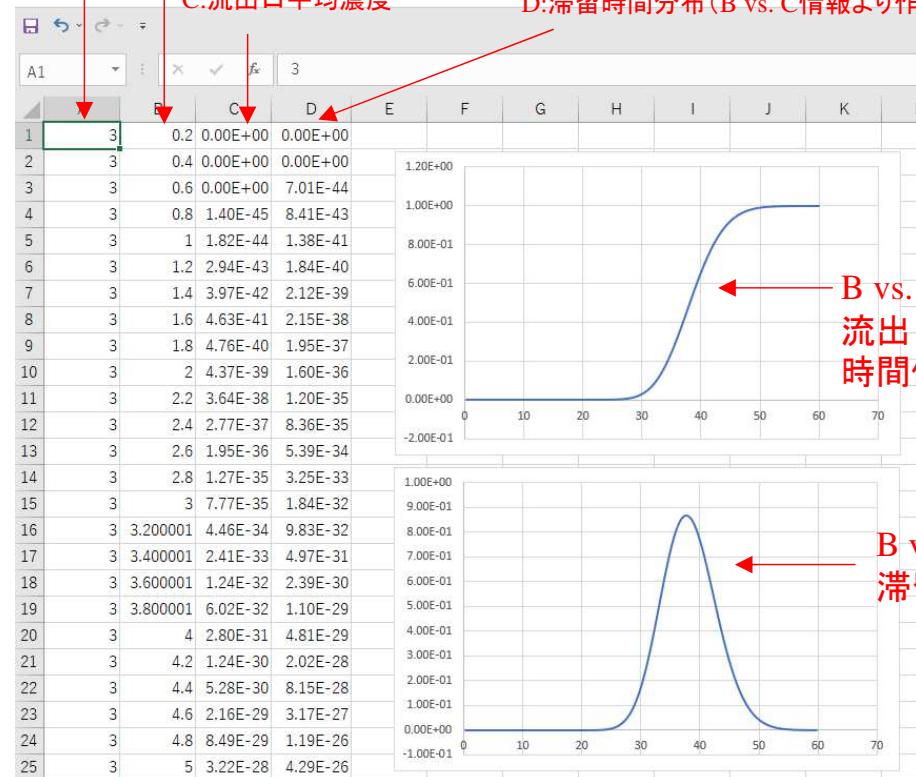
流出口平均濃度

A:流出口(スクリュ出口)のブロック番号

B:シミュレーション時刻

C:流出口平均濃度

D:滞留時間分布(B vs. C情報より作成)



B vs. C :  
流出口平均濃度の  
時間依存性

B vs. D :  
滞留時間分布

## 滞留時間分布情報の作成方法

1) Dカラムの1行目に数式 $=(C2-C1)/0.2$ を入力してリターンキーを押す. ((C2-C1)/計算時間刻み:  $\Delta t=0.2$ の場合)

	A	B	C	D	E
1	3	0.2	0.00E+00	0.2	
2	3	0.4	0.00E+00		
3	3	0.6	0.00E+00		
4	3	0.8	1.40E-45		
5	3	1	1.82E-44		
6	3	1.2	2.94E-43		
7	3	1.4	3.97E-42		

2) Dカラムの1行目をマウスクリックし, マウス右ボタンを押してコピーを選択.

	A	B	C	D
3	0.2	0.00E+00	0.00E+00	
3	0.4	0.00E+00		
3	0.6	0.00E+00		
3	0.8	1.40E-45		
3	1	1.82E-44		
3	1.2	2.94E-43		
3	1.4	3.97E-42		
3	1.6	4.63E-41		

3) Dカラムの2行目から末尾行までをマウスドラッグ選択し, 再度マウス右ボタンを押して形式の貼り付けをマウスクリック選択.

	A	B	C	D	E
1	3	0.2	0.00E+00	0.00E+00	
2	3	0.4	0.00E+00	0.00E+00	
3	3	0.6	0.00E+00	7.01E-44	
4	3	0.8	1.40E-45	8.41E-43	
5	3	1	1.82E-44	1.38E-41	
6	3	1.2	2.94E-43	1.84E-40	
7	3	1.4	3.97E-42	2.12E-39	
8	3	1.6	4.63E-41	2.15E-38	
9	3	1.8	4.76E-40	1.95E-37	
10	3	2	4.37E-39	1.60E-36	
11	3	2.2	3.64E-38	1.20E-35	
12	3	2.4	2.77E-37	8.36E-35	

4) Dカラムの一番最後の行のみ削除する

296	3	59.20015	0.999934	2.35E-04
297	3	59.40015	0.999938	2.00E-04
298	3	59.60015	0.999942	1.75E-04
299	3	59.80016	0.999946	1.55E-04
300	3	60.00016	0.999949	
301				



### ③ PP分解反応計算(フォルダ名:pp)

#### Initialsetforchemの内容

```

double precision dio,q0o,q1o,q2o,q3o,dm0
C
C+++++
C+   User define variable number
C+++++
C
C   Number of chemical species
C
chemnumber=4
C
C   Number of chemical variable
C
chemvnumber=6
C
C   Number of common variable
C
commonvnumber=2
C
C+++++

```

当ルーチン内で利用する変数の型宣言

```

C+++++
C   User define Initial/Boundary condition set
C+++++
dio=1.0d-03
q0o=1.0081d-02
q1o=1.7857d+04
q2o=1.5136d+11
q3o=2.0*q2o*(2.0*q2o*q0o-q1o**2.0)/q1o/q0o
dm0=4.2D-02

```

化学反応計算の初期値  
(境界値)設定

```

C
C   chemname(1)='Peroxide (moles/kg)',
C   chemname(2)='Q0 (moles/m3)',
C   chemname(3)='Q1 (moles/m3)',
C   chemname(4)='Q2 (moles/m3)',
C
C   chemvname(1)='Q3 (moles/m3)',
C   chemvname(2)='Mn (kg/m3)',
C   chemvname(3)='Mw (kg/m3)',
C   chemvname(4)='Mz (kg/m3)',
C   chemvname(5)='Mw/Mn (-)',
C   chemvname(6)='Mz/Mw (-)',
C
C   dmn=dm0*q1o/q0o
C   dmw=dm0*q2o/q1o
C   dmz=dm0*q3o/q2o
C
C   commonvpar(1)=dmw
C   commonvpar(2)=dmz/dmw
C

```

解析対象とする移流  
方程式の数=4

代数的関係式の数=6

コモン変数の数=2

ポスト項目の  
見出し

各種平均分子量の初期値設定

Viscalと共用する変数定義



```

c      do ib=1,iblock ← 全ブロックに設定
c
c      do ie=1,nelem(ib)
c
c      chemc(1,ie,ib)=dio
c      chemc(2,ie,ib)=q0o  化学種要素変数の初期設定
c      chemc(3,ie,ib)=q1o
c      chemc(4,ie,ib)=q2o
c
c      chempar(1,ie,ib)=q3o
c      chempar(2,ie,ib)=dmn
c      chempar(3,ie,ib)=dmw
c      chempar(4,ie,ib)=dmz
c      chempar(5,ie,ib)=chempar(3,ie,ib)/chempar(2,ie,ib)
c      chempar(6,ie,ib)=chempar(4,ie,ib)/chempar(3,ie,ib)
c      end do
c
c      do in=1,nnode(ib)
c      chemcn(1,in,ib)=dio
c      chemcn(2,in,ib)=q0o
c      chemcn(3,in,ib)=q1o  化学種節点変数の初期設定
c      chemcn(4,in,ib)=q2o
c
c      chemparn(1,in,ib)=q3o
c      chemparn(2,in,ib)=dm0*q1o/q0o
c      chemparn(3,in,ib)=dm0*q2o/q1o
c      chemparn(4,in,ib)=dm0*q3o/q2o
c      chemparn(5,in,ib)=chemparn(3,in,ib)/chemparn(2,in,ib)
c      chemparn(6,in,ib)=chemparn(4,in,ib)/chemparn(3,in,ib)
c
c      end do

```

```

c      if(ib.eq.1) then
c
c      do ii=1,ndivl(ib)
c      chemcn(1,ii,ib)=dio  ブロック1の流入口
c      chemcn(2,ii,ib)=q0o  (in=1~iinletnodelist)に
c      chemcn(3,ii,ib)=q1o  濃度境界値を設定
c      chemcn(4,ii,ib)=q2o
c      end do
c      end if
c
c      end do

```

## chemfscalの内容

```

c subroutine chemvariable(ib)  ← ブロック毎に呼出される
c
c double precision dk0,actv,dkd,th  ← 当ルーチン内で利用する変数の型宣言
c fc=1.0  ← Initiator efficiency
corg te=200.0+273.15
c dk0=1.98d+012  ← ペルオキシド分解速度定数
c actv=1.4947d+04  ← パラメータ
corg dkd=dk0*dexp(-actv/te)
c
c do ie=1,nelem(ib)
c
c te=tempe(ie,ib)+273.15  ← ペルオキシド分解速度定数の計算
c dkd=dk0*dexp(-actv/te)  ← tempe(ie, ib): ブロックibの要素ieの温度
c
c Peroxide implicit difference
c
c chemf(1,ie,ib)=dkd  ← 
c chems(1,ie,ib)=0.0  ← 
c
c th=2.0*fc*dkd*chemc(1,ie,ib)
c
c Moment a0,q1,q2 explicit difference
c
c q0
c chemf(2,ie,ib)=0.0
c chems(2,ie,ib)=th*(chemc(3,ie,ib)-3.0*chemc(2,ie,ib))
c & /((chemc(3,ie,ib)-chemc(2,ie,ib)))
c
c q1
c chemf(3,ie,ib)=0.0
c chems(3,ie,ib)=-2.0*th*chemc(2,ie,ib)
c & /((chemc(3,ie,ib)-chemc(2,ie,ib)))
c
c q2
c chemf(4,ie,ib)=0.0
c chems(4,ie,ib)=th*(-chempar(1,ie,ib)/3.0
c & +chemc(3,ie,ib)/3.0-2.0*chemc(2,ie,ib))
c & /((chemc(3,ie,ib)-chemc(2,ie,ib)))
c
c end do
c
c return
c stop
c end

```

TSSが解析対象とする移流方程式:

$$\left( chemf(ic,ie,ib) + \frac{D}{Dt} \right) f(ic,ie,ib) = chems(ic,ie,ib)$$

$$\frac{D[I]}{Dt} = -\kappa_d [I] \Rightarrow \left( \kappa_d + \frac{D}{Dt} \right) [I] = 0$$

$$\frac{DQ_0}{Dt} = 2f\kappa_d [I] \left( \frac{Q_1 - 3Q_0}{Q_1 - Q_0} \right)$$

$$\frac{DQ_1}{Dt} = -2f\kappa_d [I] \left( \frac{Q_0}{Q_1 - Q_0} \right)$$

$$\frac{DQ_2}{Dt} = 2f\kappa_d [I] \left( \frac{-\frac{1}{3}Q_3 + \frac{1}{3}Q_1 - 2Q_0}{Q_1 - Q_0} \right)$$

変数対応関係

$[I]$ : chemc(1,ie,ib)

$Q_0$ : chemc(2,ie,ib)

$Q_1$ : chemc(3,ie,ib)

$Q_2$ : chemc(4,ie,ib)

## chemvariableの内容

subroutine chemvariable(ib) ← ブロック毎に呼出される

double precision q0o,q1o,q2o,dm0 ← 当ルーチン内で利用する変数の型宣言

dm0=4.2D-02 ← プロピレンの分子量: 4.2D-02 kg/m<sup>3</sup>

do ie=1,nelem(ib)

q0o=chemc(2,ie,ib)  
q1o=chemc(3,ie,ib)  
q2o=chemc(4,ie,ib)

Closure relation

q3o=2.0\*q2o\*(2.0\*q2o\*q0o-q1o\*\*2.0)/q1o/q0o ← Closure relation

chempar(1,ie,ib)=q3o  
chempar(2,ie,ib)=dm0\*q1o/q0o  
chempar(3,ie,ib)=dm0\*q2o/q1o  
chempar(4,ie,ib)=dm0\*q3o/q2o  
chempar(5,ie,ib)=chempar(3,ie,ib)/chempar(2,ie,ib)  
chempar(6,ie,ib)=chempar(4,ie,ib)/chempar(3,ie,ib)

end do

return  
stop  
end

### 変数対応関係

Chempar(1,ie,ib) :  $Q_3$

Chempar(2,ie,ib) : 数平均分子量  $M_n$

Chempar(3,ie,ib) : 重量平均分子量  $M_w$

Chempar(4,ie,ib) : Z平均分子量  $M_z$

Chempar(5,ie,ib) :  $M_w/M_n$

Chempar(6,ie,ib) :  $M_z/M_w$

## viscalの内容

nonnew : 非ニュートン識別番号

1:Newtonian

2:Power law

3:Log polynomial

4:Carreau

5:Carreau Yasuda

6:Cross

```
c*** (6) Cross Model
```

```
c
c   if(nonnew.eq.6) then
c     eta0=bcoef*exp(tref/(temhi+273.15))
c     eta=eta0/(1.0+(eta0*gamm/rcoef)**(1.0-dn))
c     vish(i,ie,ib)=eta
c
c     Molecular weight average eta0 shift proposed by Fukuoka & Min
c
c     eta0=(chempar(3,ie,ib)/commonvpar(1))*3.7*eta0
c
c     MWD shift of shear thinning
c
c     dc=chempar(6,ie,ib)/commonvpar(2)
corg eta=eta0/(1.0+(eta0*gamm/rcoef)**(1.0-dn))
c     eta=eta0/(1.0+dc*(eta0*gamm/rcoef)**(1.0-dn))
c
c     vish(i,ie,ib)=omgc*eta+(1.0-omgc)*vishold
c   end if
c
```

$\eta_0 = a \exp\left(\frac{T_b}{T + 273.15}\right)$

$\eta_0(t) = \left(\frac{M_w(t)}{M_w(0)}\right)^{3.7} \eta_0(0)$

$dc = \frac{M_z(t)/M_w(t)}{M_z(0)/M_w(0)}$

$\eta = \frac{\eta_0}{1 + dc * \exp\left(\frac{\eta_0 * gamm}{rcoef}\right)^{1-dn}}$

$a \rightarrow bcoef$  (Material fit パラメータ),  
 $T_b \rightarrow tref$  (Material fit パラメータ),  
 $T \rightarrow temhi$ : 要素ieの層iの温度

## chemwriteの内容

subroutine chemwrite

```
dimension chemcon(4,20),chemparameter(6,20)
cccccccccccccccccccccccccccccccccccccccc
open(110,File='CHEMCON',access='sequential',status='unknown',
&    recl=220)
open(111,File='CHEMPAR',access='sequential',status='unknown',
&    recl=220)
cccccccccccccccccccccccccccccccccccccccc
do ib=1,iblock
  if(chemcnumber.gt.0) then
    do in=ndivl(ib)+1,nnode(ib)
      do ic=1,chemcnumber
        volsum=0.0
        csum=0.0
        do ii=1,nren(in,ib)
          ine=nrelem(inrelem(in,ib)+ii-1,ib)
          if(unfillmask.eq.1) then
            volsum=volsum+vol(ine,ib)*fille(ine,ib)
            csum=csum+vol(ine,ib)*chemc(ic,ine,ib)*fille(ine,ib)
          else
            volsum=volsum+vol(ine,ib)
            csum=csum+vol(ine,ib)*chemc(ic,ine,ib)
          end if
        end do
        chemcn(ic,in,ib)=csum/volsum
      end do
    end do
  end if
```

ローカル変数の定義: 化学種4成分, 代数関係式6成分

出力ファイルオープン

機番110: ファイル名CHEMCON, 化学種4成分出力用

機番111: ファイル名CHEMPAR, 代数関係式6成分

ブロックのループ

境界節点を除く, 節点番号ループ

化学種ループ

節点inを含む要素数ループ

ine: 節点inを含む要素番号

unfillmask: 0 ; 未充满領域も計算量を表示

: 1 ; 未充满領域の計算量を非表示

流入境界を除く節点の化学種濃度を体積重み付けの要素濃度より計算

unfillmask: 1 の場合は, 要素充满率を考慮した体積重み付け平均,

unfillmask: 0 の場合は, 要素充满率を無視した体積重み付け平均

nren(in,ib): ブロックibの節点inを含む要素数

nrelem(inrelem(in,1)+ii-1,ib): ブロックibの節点inを含むii番目の要素番号

fille(ine,ib) : ブロックibの要素ineの充满率 (0 or 1)

非チェック: unfillmask=0

チェック: unfillmask=1

☐ Result display considered with unfill

設定/開じる

\* Surface Renewal Model パラメータ設定フォーム下部

c

```

if(chemvnumber.gt.0) then
do in=ndivl(ib)+1,nnode(ib) ← 境界節点を除く, 節点番号ループ
do ic=1,chemvnumber ← 代数関係式数ループ
volsum=0.0
csum=0.0
do ii=1,nren(in,ib) ← 節点inを含む要素数ループ
ine=nrelem(inrelem(in,ib)+ii-1,ib) ← ine:節点inを含む要素番号
if(unfilmask.eq.1) then ← unfilmask:0 ; 未充満領域も計算量を表示
volsum=volsum+vol(ine,ib)*fille(ine,ib) :1 ; 未充満領域の計算量を非表示
csum=csum+vol(ine,ib)*chempar(ic,ine,ib)*fille(ine,ib)
else
volsum=volsum+vol(ine,ib)
csum=csum+vol(ine,ib)*chempar(ic,ine,ib)
end if
end do
chemparn(ic,in,ib)=csum/volsum
end do
end do
end if

```

流入口境界を除く節点の代数関係式  
を体積重み付けの要素濃度より計算

unfilmask:1の場合は, 要素充満率を考慮した体積重み付け平均(未充満領域を非表示),  
unfilmask:0の場合は, 要素充満率を無視した体積重み付け平均(未充満領域も表示)

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
if(chemnumber.ne.0) then
```

```
if(ibtype(ib).eq.0) then
```

```
do iz=0,ndivzs(ib)
```

```
ntops=ndivl(ib)*iz+1
```

```
ntope=ndivl(ib)*(iz+1)
```

```
do ic=1,chemnumber
```

```
chemcon(ic,ib)=0.0
```

```
end do
```

```
do ic=1,chemnumber
```

```
countn=0.0
```

```
do in=ntops,ntope
```

```
countn=countn+1.0
```

```
chemcon(ic,ib)=chemcon(ic,ib)+chemcn(ic,in,ib)
```

```
end do
```

```
chemcon(ic,ib)=chemcon(ic,ib)/countn
```

```
end do
```

```
write(110,*) znode(ntops,ib)*10.0,',',
```

```
&  
&  
&  
&  
&
```

```
chemcon(1,ib),',',  
chemcon(2,ib),',',  
chemcon(3,ib),',',  
chemcon(4,ib)
```

```
end do
```

```
else
```

```
cccccccccc ↑ ibtype(ib).eq.1
```

ブロック ib のスクリュメッシュが標準定義の場合

軸方向分割数ループ

軸方向分割数iz断面内の開始節点番号

軸方向分割数iz断面内の終了節点番号

化学種 ic の数平均値の計算

軸方向分割数iz断面のz座標と  
各種数平均化学種計算値の  
ファイルCHEMCON への出力

軸垂直断面内の化学種  
平均値の計算

以下の矩形領域定義のコードは省略(詳細は直接ソースコードを参照ください)

化学種変数と数平均  
値対応関係

[I]: chemcon(1)

Q<sub>0</sub>: chemcon(2)

Q<sub>1</sub>: chemcon(3)

Q<sub>2</sub>: chemcon(4)



```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```

if(chemvnumber.ne.0) then
  if(ibtype(ib).eq.0) then
    do iz=0,ndivzs(ib)
      ntops=ndivl(ib)*iz+1
      ntope=ndivl(ib)*(iz+1)

```

ブロック ib のスクリュメッシュが標準定義の場合  
 軸方向分割数ループ  
 軸方向分割数 iz 断面内の開始節点番号  
 軸方向分割数 iz 断面内の終了節点番号

```

do ic=1,chemvnumber
  chemparameter(ic,ib)=0.0
end do
do ic=1,chemvnumber
  countn=0.0
  do in=ntops,ntope
    countn=countn+1.0
    chemparameter(ic,ib)=chemparameter(ic,ib)+chemparn(ic,in,ib)
  end do
  chemparameter(ic,ib)=chemparameter(ic,ib)/countn
end do
write(111,*) znnode(ntops,ib)*10.0,',',
&
&
&
&
&
&
&
&
end do

```

代数関係式 ic の  
数平均値の計算

軸垂直断面内の代数的  
関係式平均値の計算

軸方向分割数 iz 断面の z 座標と  
各種数平均代数的関係式  
計算値のファイルCHEMPAR  
への出力

```

c
else
cccccccccc ↑ ibtype(ib).eq.1

```

ブロック ib のスクリュメッシュが  
矩形領域定義の場合

以下の矩形領域定義のコードは省略(詳細は直接ソースコードを参照ください)

数平均値と代数的関係式対応関係

chemparameter(1) :  $Q_3$   
 chemparameter(2) : 数平均分子量  $M_n$   
 chemparameter(3) : 重量平均分子量  $M_w$   
 chemparameter(4) : Z平均分子量  $M_z$   
 chemparameter(5) :  $M_w/M_n$   
 chemparameter(6) :  $M_z/M_w$

## User define model タブメニューの設定

Single Screw Simulator Template

スクリュー形状 ダイ形状 ホッパー形状 押出機形状 解析プログラム実行 解析結果統括表 ユーザ定義解析

☒ ユーザ定義ルーチンの利用/RTD解析 (デフォルト) ← ユーザ定義ルーチンのコール

☒ 熱流動解析との連成 ← 当化学反応計算は、熱流動解析に影響を及ぼすため、  
熱流動場との連成解析 (チェックボックスをチェック)

計算パラメータ

反復計算回数	10	← 非線形反復計算回数
反復計算の緩和係数	1	
マトリクスソルバ反復計算回数	20000	
マトリクスソルバの収束基準値	1E-06	
RTD解析の時間刻み	0.2	

## SSS出力ファイル (CHEMCON, プロジェクト名; pp\*\*)

CHEMCON - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

0.000000E+00	1.000000E-03	1.0081000E-02	17857.00	1.5136000E+11
3.175000	9.8075939E-04	1.0119480E-02	17857.00	1.4818532E+11
6.350000	9.6868741E-04	1.0143602E-02	17856.96	1.4625820E+11
9.525000	9.5648953E-04	1.0167991E-02	17856.96	1.4436337E+11
12.70000	9.4420608E-04	1.0192607E-02	17857.03	1.4250644E+11
15.87500	9.3183777E-04	1.0217361E-02	17857.05	1.4068644E+11
19.05000	9.1939484E-04	1.0242223E-02	17857.02	1.3890388E+11
22.22500	9.0689218E-04	1.0267206E-02	17856.99	1.3715976E+11
25.40000	8.9434226E-04	1.0292271E-02	17856.93	1.3545447E+11
28.57500	8.8176149E-04	1.0317413E-02	17856.91	1.3378898E+11
31.75000	8.6916424E-04	1.0342623E-02	17856.93	1.3216374E+11
34.92500	8.5655990E-04	1.0367843E-02	17856.95	1.3057835E+11
38.09999	8.4395875E-04	1.0393035E-02	17856.93	1.2903259E+11
41.27499	8.3137600E-04	1.0418214E-02	17856.95	1.2752682E+11
44.44999	8.1882108E-04	1.0443344E-02	17856.98	1.2606056E+11
47.62499	8.0630137E-04	1.0468346E-02	17856.92	1.2463292E+11
50.80000	7.9383160E-04	1.0493253E-02	17856.88	1.2324413E+11
53.97499	7.8142365E-04	1.0518066E-02	17856.87	1.2189402E+11
57.14999	7.6908356E-04	1.0542703E-02	17856.81	1.2058146E+11
60.32499	7.5682235E-04	1.0567162E-02	17856.71	1.1930616E+11
63.49999	7.4465183E-04	1.0591458E-02	17856.65	1.1806785E+11
66.67499	7.3258067E-04	1.0615569E-02	17856.60	1.1686593E+11
69.84999	7.2061876E-04	1.0639503E-02	17856.62	1.1570001E+11
73.02499	7.0877076E-04	1.0663196E-02	17856.62	1.1456886E+11
76.20000	6.9704576E-04	1.0686655E-02	17856.62	1.1347205E+11
79.37500	6.8545277E-04	1.0709904E-02	17856.72	1.1240918E+11
82.55000	6.7398982E-04	1.0732799E-02	17856.67	1.1137814E+11
85.72501	6.6266622E-04	1.0755382E-02	17856.58	1.1037875E+11
88.90001	6.5149489E-04	1.0777768E-02	17856.64	1.0941148E+11
92.07501	6.4047478E-04	1.0799855E-02	17856.71	1.0847457E+11
95.25002	6.2960945E-04	1.0821632E-02	17856.78	1.0756716E+11
98.42502	6.1889872E-04	1.0843026E-02	17856.74	1.0668778E+11
101.6000	6.0834904E-04	1.0864089E-02	17856.69	1.0583619E+11

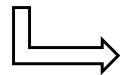
z座標

[I]

$Q_0$

$Q_1$

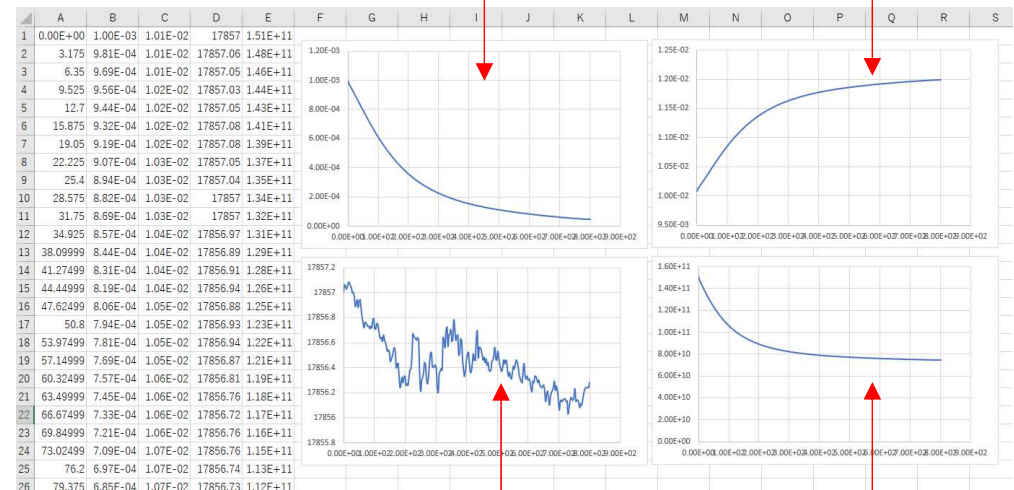
$Q_2$



コンマ区切りで  
エクセルに読み込み

A vs. B :  
z座標 vs. ペルオキ  
シド濃度

A vs. C :  
z座標 vs.  $Q_0$



A vs. D :  
z座標 vs.  $Q_1$

A vs. E :  
z座標 vs.  $Q_2$

## SSS出力ファイル (CHEMPPAR, プロジェクト名;pp\*\*)

CHEMPPAR - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

0.000000E+00	4.5956216E+18	74396.82	356001.8	1275213.	4.785177	3.582043
1.250000	4.5499855E+18	74334.25	354310.3	1268573.	4.766447	3.580399
2.500000	4.5320531E+18	74309.46	353643.6	1265956.	4.759066	3.579749
3.750000	4.5131555E+18	74283.17	352939.8	1263192.	4.751271	3.579059
5.000000	4.4937689E+18	74256.06	352215.5	1260350.	4.743255	3.578348
6.250000	4.4741599E+18	74228.47	351481.8	1257470.	4.735132	3.577625
7.500000	4.4544627E+18	74200.62	350743.0	1254571.	4.726953	3.576894
8.750000	4.4346146E+18	74172.35	349996.9	1251642.	4.718694	3.576152
10.00000	4.4145884E+18	74143.70	349242.2	1248682.	4.710341	3.575402
11.25000	4.3946020E+18	74114.91	348487.7	1245721.	4.701990	3.574647
12.50000	4.3746327E+18	74085.95	347731.9	1242755.	4.693624	3.573890
13.75000	4.3545924E+18	74056.73	346971.6	1239773.	4.685211	3.573125
15.00000	4.3344692E+18	74027.23	346206.4	1236771.	4.676743	3.572352
16.25000	4.3142481E+18	73997.35	345435.7	1233748.	4.668214	3.571570
17.50000	4.2938999E+18	73967.11	344658.2	1230699.	4.659612	3.570780
18.75000	4.2733976E+18	73936.45	343872.9	1227619.	4.650923	3.569977
20.00000	4.2526971E+18	73905.29	343078.0	1224502.	4.642128	3.569162
21.25000	4.2318685E+18	73873.72	342276.4	1221358.	4.633260	3.568337
22.50000	4.2110550E+18	73841.96	341473.1	1218209.	4.624375	3.567508

z座標

$Q_3$

Mn

Mw

Mz

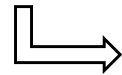
Mw/Mn

Mz/Mw

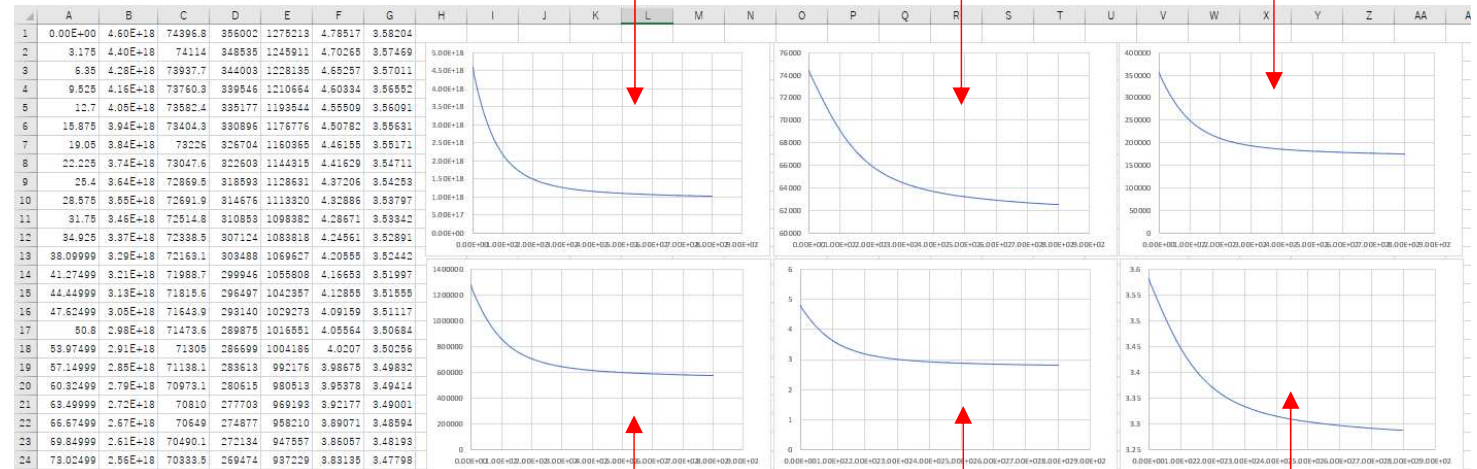
A vs. B :  
z座標 vs.  $Q_3$

A vs. C :  
z座標 vs. Mn

A vs. D :  
z座標 vs. Mw



コンマ区切りで  
エクセルに読み込み



A vs. E :  
z座標 vs. Mz

A vs. F :  
z座標 vs. Mw/Mn

A vs. G :  
z座標 vs. Mz/Mw