Scheme-Programmierung in FLUENT 5 & 6

Mirko Javurek
Institut für Strömungsprozesse und Wärmeübertragung,
Johannes Kepler Universität Linz, Österreich
http://fluid.jku.at

Begonnen im September 2000, ergänzt: 2003, 10-2004

Inhalt

Vorwort

Scheme bietet sehr viele Möglichkeiten, Prozesse in Fluent automatisiert ablaufen zu lassen. Leider gibt es bis heute von Fluent so gut wie keine Dokumentationen zu diesem Thema. Dybviks Scheme-Buch (siehe Abschnitt Literatur) hat mir sehr geholfen, Scheme verstehen zu lernen. Zur Anwendung in Fluent bedarf es aber einiger über das Standard-Scheme hinausgehende Kenntnisse. Um meine Erfahrungen mit Fluent-Scheme auch anderen zukommen zu lassen, habe ich im September 2000 begonnen, dieses Skriptum zu schreiben. Das Skript ist etwas rudimentär, aber immer noch besser als nichts.
In der Zwischenzeit sind ein paar Erweiterungen dazugekommen, ich danke an dieser Stelle Zoran Carija für seinen Tipp mit der with-output-to-file Funktion und Angelo Sozzi dafür, dass er sich die Mühe macht, dieses Skript auf englisch zu übersetzen.
Es freut mich, immer wieder positive Rückmeldungen zu diesem Skript zu bekommen, und dass sogar FLUENT Deutschland das Skriptum seinen Kunden empfiehlt. FLUENT selbst wird nach Auskünften von FLUENT Deutschland keine offizielle Scheme-Dokumentation mehr herausbringen, da Scheme in Zukunft durch die Skriptsprache Python ersetzt werden soll.

Mirko Javurek, Linz am 27. 10. 2004

Einleitung

Scheme ist LISP-Dialekt;
Einheitliches Befehlsformat:

```
(befehlsname argument1 argument2 ...)
```

Jeder Befehlsaufruf ist ein Funktionsaufruf und liefert daher ein Ergebnis.
Befehls- und Variablennamen sind *nicht* case-sensitive (sollten nur Kleinbuchstaben enthalten), müssen mit einem Buchstaben beginnen, und dürfen sonst neben `a-z` und `0-9` auch die Sonderzeichen `+-*/<>=?.:%$!~^_` enthalten.
Kommentare werden mit `;;` eingeleitet und enden am Zeilenende.

Schnittstellen Fluent-Scheme

Aufruf von Scheme-Befehlen in Fluent:

- Befehl im Fluent-Textinterface eingeben (auch mit der Maus kopieren der Fluent-Befehle aus anderen Fenstern - z.B. Editor - über X-Selection in Unix möglich), **oder**
- Scheme-Programm mit Texteditor schreiben, speichern (.scm-Endung) und in Fluent-Menü mit "File/Read/Scheme" einlesen;
- Wenn sich im home-Verzeichnis eine Scheme-Datei namens `.fluent` befindet, wird sie beim Starten von Fluent automatisch ausgeführt.
- Im Menü "Solve/Monitor/Commands/Command" können im Textinterface- und Scheme-Befehle eingegeben werden, die dann jede Iteration oder jeden Zeitschritt ausgeführt werden;

Aufruf von Fluent-Befehlen in Scheme:

- Textinterface-Befehl:
  ```
  (ti-menu-load-string "display/contour temperature 30 100")
  ```
  Rückgabewert: #t wenn erfolgreich, #f wenn Fehler oder Abbruch durch Ctrl-C; Ctrl-C hält Fluent-Befehl, nicht aber Scheme-Programm an!
- GUI-Befehl: Journal mit den gewünschten GUI-Aktionen aufzeichnen, Journal enthält direkt Scheme-Befehle, z.B.:
  ```
  (cx-gui-do cx-activate-item "Velocity Vectors*PanelButtons*PushButton1(OK)")
  ```

Textinterface-Befehle sind schneller, kompakter und vielseitiger verwendbar. GUI-Kommandos sind langsam, unübersichtlich und schlecht anpassbar (Referenz auf Listeneintrag z.B. nicht nach Name des Eintrags, sondern Nummer). Textinterface-Kommandos sind demnach GUI-Kommandos vorzuziehen; GUI-Kommandos nur dann verwenden, wenn für gesuchte Aktion kein Textinterface-Kommando verfügbar ist.
Textinterface Kommandos sind noch nicht kommentiert (Ab Fluent 5.5?). Vorgehensweise: Gewünschtes Textkommando suchen, ausprobieren und alle gemachten Eingaben zusammenfassen.

Ausgabe von Scheme ins Fluent-Textinterface:

```
(display object)
(newline)
```

Dateizugriff (lesen/schreiben) in Scheme (siehe Beispiele).

RP-Variablen

Variable auslesen, z.B. aktuelle Simulationszeit:

```
> (rpgetvar 'flow-time)
0.1
```

Variable setzen:

```
> (rpsetvar 'flow-time 0)
```

Alle RP-Variablen sind im Case-File definiert (Section "Variables").

CX-Variablen

Variable auslesen, z.B.: Farbverlauftabellen:

```
> (cxgetvar 'cmap-list)
0.1
```

Variable setzen:

```
> (cxsetvar 'def-cmap "rgb")
```

Alle CX-Variablen sind (nur teilweise?) im Case-File definiert (Section "Cortex Variables").

Schnittstelle Fluent-Scheme-UDFs

Datenaustausch

Es können eigene RP-Variablen definiert werden, die in Fluent über das Textinterface und in UDFs über spezielle Funktionen angesprochen werden können.
Definition einer eigenen RP-Variable:

```
(rp-var-define name default-and-init-value type #f)
types: 'int 'real 'boolean 'string ...?
```

zum Beispiel:

```
> (rp-var-define 'udf/var1 0 'real #f)
```

Info über Variable:

```
> (rp-var-object 'udf/var1)
(udf/var1 0 real #f 0)

> (rp-var-object 'udf/var2)
#f
```

Ändern und Abfragen wie oben mit rpsetvar und rpgetvar.
Wenn eine RP-Variable einmal definiert ist, bleibt sie bis zum Beenden von Fluent erhalten (!), wird in jedem Case-File mit abgespeichert, und beim Hereinladen eines solchen Case-Files - falls nicht definiert - erzeugt und auf den abgespeicherten Wert gesetzt.
In UDFs können die RP-Variablen mit den C-Funktionen (deklariert in `Fluent.Inc/fluentX.Y/src/var.h`)

```
real RP_Get_Real(char *s);
long RP_Get_Integer(char *s);
char *RP_Get_String(char *s);
boolean RP_Get_Boolean(char *s);

void RP_Set_Real(char *s, real v);
void RP_Set_Integer(char *s, long v);
void RP_Set_Boolean(char *s, boolean v);
void RP_Set_String(char *s, char *v);
void RP_Set_Symbol(char *s, char *v);
```

abgefragt bzw. gesetzt werden, z.B.:

```
var1 = RP_Get_Real("udf/var1");
RP_Set_Real("udf/var1", 3.2);
```

Bei UDFs im Parallelmodus sind beim Zugriff auf RP-Variablen spezielle Regeln zu beachten, siehe dazu FLUENT-UDF Handbuch.

Aufruf von Funktionen

UDFs vom Type EOD können aus Scheme über den Befehl

```
(%udf-on-demand "udf-eod-name")
```

aufgerufen werden.
Um Scheme-Funktionen aus einer UDF aufzurufen, ist zur Zeit keine Möglichkeit bekannt; die C-Funktion `CX_Interpret_String("scheme-command-string")` - deklariert in `Fluent.Inc/fluentX.Y/cortex/src/cx.h` - interpretiert zwar den "scheme-command-string", hat aber keinen Zugriff auf das Environment.

Arithmetische Funktionen

Grundfunktionen `+ - * /` , entspricht UPN, mehr als 2 Argumente möglich:

```
> (+ 2 4 5)
11

> (/ 6 3)
2

> (/ 2)          ;; entspricht (/ 1 2)
0.5
```

Weiters `(abs x), (sqrt x), (expt x y)` [$= x^y$], `(exp x)` [$= e^x$], `(log x)` [$= \ln x$], `(sin x), (cos x), (atan x), (atan x y)` [$= \arctan(x/y)$], …

Integer(!)-Funktionen:

```
> (remainder 45 6)
3

> (modulo 5 2)
1

(truncate x), (round x), (ceiling x), (floor x), ...
```

weiters

```
(max x y ...), (min x y ...)
```

z.B. um aus Liste Maximum suchen:

```
> (apply max '(1 5 8 3 4))
8
```

und einige weitere (siehe Scheme-Literatur).

Globale Scheme-Variablen

Definieren mit:

```
> (define x 3)
> (+ x 1)
4
```

Keine Unterscheidung in Variablentypen (Integer, Real, String, ...) – Jede Variable kann Wert von jedem Typ annehmen.
Wert ändern mit erneuter Definition (nicht innerhalb von Funktionen möglich, dort gilt ein lokaler Variablenbereich, sodass die Variable mit define lokal neu definiert wird) oder besser

```
(set! x 1)
```

Wert darstellen mit

```
(display x)
```

oder

```
(write x)
```

Write sollte nur verwendet werden, wenn Fluent-Variablen in eine Datei abgespeichert und später wieder eingelesen werden sollen; Write stellt z.B. Strings mit Anführungszeichen dar.
Konstanten: Integer (2), Float (2.5), Boolean (`#t` für true, `#f` false) Strings `("this is a text string"`) und Symbole: `'symbol`, z.B.:

```
 (define x 'this-is-a-symbol)
```

Spezielle Zeichen für String-Definitionen:

```
\"   "
\n   neue Zeile
```

Globale Variablen und selbst definierte Scheme-Funktionen bleiben bis zum Beenden von Fluent erhalten.

Lokale Scheme-Variablen

```
(let ((var1 value1) (var2 value2) ...)
   ... Kommandos im Gültigkeitsbereich...
)
```

Listen

Definition z.B.:

```
> (define my-surfaces '(wall-top wall-bottom symmetry))
```

Länge beliebig, dynamische Verwaltung, Schachtelung möglich.
Listen Definieren mit ‘(elements ...) :

```
> (define l '(a b c))
```

Erstes Element einer Liste

```
> (car l)
a
```
"Rest" einer Liste (Liste ohne erstes Element)
```
> (cdr l)
(b c)
```
Anzahl der Listenelemente
```
> (length l)
3
```
i-tes Element einer Liste
```
(listref liste i)
```
Element in Liste suchen:
```
> (member 'd '(a b c d e f g))
(d e f g)
```
Funktion auf Liste(n) anwenden:
```
> (map (lambda (x) (* x x)) '(1 2 3))
(1 4 9)
> (apply + '(2 4 5))
11
> (apply max '(1 5 8 3 4))
8
```

if-Befehl

if-Befehl ist eine Funktion:
```
(if cond true-value false-value)
```
cond ist ein boolscher Ausdruck, der entweder `#t` (true) oder `#f` (false) ist.
Vergleichsoperationen:
Gleichheit:
```
(= a b)  ;; Zahlen
(eq? a b)  ;; Objekte
(eqv? a b) ;; Objekte gleichen Wert
```
Relationen:
```
(positive? x)
(negative? x)
(< a b)
(> a b)
(<= a b)
(>= a b)
```
Boolsche Funktionen:
```
(not a)
(and a b c ...)
(or  a b c ...)
```
Erweiterung des "if"- und "else"-Zweiges für mehrere Kommandos mit Block-Befehl "begin" ("sequencing", allgemein anwendbar):
```
(if cond
   (begin  ;; if
      ...
      true-value
   )
   (begin  ;; else
      ...
      false-value
   )
)
```
Wenn der Ergebniswert des if-Befehls nicht benötigt wird, können "else"-Zweig und die Ergebniswerte weggelassen werden.

Komplexere Befehle für bedingte Ablaufsteuerung (z.B. für stückweise definierte Funktionen):
```
(cond (test1 value1) (test2 value2) ... (else value))
```
und für diskrete Werte einer Variable
```
(case x ((x11 x12 x13 ...) value1) ((x21 x22 x23 ...) value2) ... (else value))
```
Wird x in einer der Listen gefunden (z.B. in (x11 x12 x13 ...)), so wird der entsprechende Wert zurückgegeben (value1).

do-Schleife

Einfachste Form (Variable, Startwert, Wert der der Schleifenvariable nach jedem Schleifendurchgang zugewiesen werden soll, Abbruchbedingung):
```
(do ((x x-start (+ x delta-x))) ((> x x-end))  ...loop-body... )
```
Mehrere oder auch keine Schleifenvariablen möglich.

Beispiel Iso-Surfaces erzeugen: mehrere Iso-Surfaces sollen in gleichmäßigen Abständen von Iso-Values generiert und automatisch benannt werden. Zuerst muss der Dialog im TUI für das Erzeugen einer Iso-Surface zusammengestellt werden:

```
>
adapt/                  grid/                   surface/
display/                plot/                   view/
define/                 report/                 exit
file/                   solve/

> surface

/surface>
delete-surface          mouse-line              point-array
surface-cells           mouse-plane             rake-surface
iso-surface             mouse-rake              rename-surface
iso-clip                partition-surface       sphere-slice
list-surfaces           plane-slice             zone-surface

/surface> iso-surface

iso-surface of>
pressure                entropy                 x-surface-area
pressure-coefficient    total-energy            y-surface-area
dynamic-pressure        internal-energy         z-surface-area
...
rel-total-temperature   x-coordinate            dp-dx
wall-temp-out-surf      y-coordinate            dp-dy
wall-temp-in-surf       z-coordinate            dp-dz

iso-surface of> x-coordinate
new surface id/name [x-coordinate-31] testname
range [-10.0131, 4.8575001]
from surface [()] ()
()
iso-value(1) (m) [()] 1.234
iso-value(2) (m) [()] ()
```

Einzeiliger TUI-Befehl lautet also (alle Eingaben in eine Zeile zusammengefasst, "nur Return" durch , (Beistrich) ersetzen):

```
surface/iso-surface x-coordinate testname () 1.234 ()
```

Daraus "parametrisierte" Scheme-Schleife:

```
(do ((x 0 (+ x 0.2)) ) ((> x 3.1))
  (ti-menu-load-string
    (format #f "surface/iso-surface x-coordinate x-~3.1f () ~a ()" x x))
)
```

Erzeugt folgende Textinterface-Befehle:

```
surface/iso-surface x-coordinate x-0.0 () 0 ()
surface/iso-surface x-coordinate x-0.2 () 0.2 ()
surface/iso-surface x-coordinate x-0.4 () 0.4 ()
...
surface/iso-surface x-coordinate x-3.0 () 3 ()
```

Verfeinerung: bessere Namen für positive und negative Koordinaten:

```
(do ((z -1 (+ z 0.25))) ((> z 1))
    (ti-menu-load-string
      (format #f "surface/iso-surface z-coordinate z~a~05.3f () ~a ()"
         (if (>= z 0) "+" "") z z))
)

surface/iso-surface z-coordinate z-1.000 () -1 ()
surface/iso-surface z-coordinate z-0.750 () -0.75 ()
surface/iso-surface z-coordinate z-0.500 () -0.5 ()
surface/iso-surface z-coordinate z-0.250 () -0.25 ()
surface/iso-surface z-coordinate z+0.000 () 0 ()
surface/iso-surface z-coordinate z+0.250 () 0.25 ()
surface/iso-surface z-coordinate z+0.500 () 0.5 ()
surface/iso-surface z-coordinate z+0.750 () 0.75 ()
surface/iso-surface z-coordinate z+1.000 () 1 ()
```

Abänderung: 2 Schleifenvariablen:

```
(do ((x 0 (+ x 0.2)) (i 1 (+ i 1))) ((> x 3.1))
  (ti-menu-load-string
    (format #f "surface/iso-surface x-coordinate x-~02d () ~a ()" i x))
)
```

```
surface/iso-surface x-coordinate x-01 () 0 ()
surface/iso-surface x-coordinate x-02 () 0.2 ()
surface/iso-surface x-coordinate x-03 () 0.4 ()
...
surface/iso-surface x-coordinate x-16 () 3 ()
```

format-Befehl

```
(format #f "Formatstring wie in C bei printf mit patterns für var1, var2, ..."  var1 var2 ... )
```

Statt dem %-Zeichen in C leitet hier die Tilde (~) ein Pattern ein; Patternbeispiele:

~a beliebige Variable in allgemeinem Format (Strings ohne "")
~d Integer-Zahl
~04d Integer mit Nullen vorne immer auf 4 Stellen anfüllen (5 wird zu 0005), z.B. für Dateinamen wichtig.
~f Fließkommazahl
~4.2f Fließkommazahl, 4 Zeichen insgesamt lang, 2 Stellen nach dem Komma: 1.2 wird zu 1.20
~s String unter "" einbauen: aus `(format #f "string: ~s !" "text")` wird `string: "text" !`
... und andere???

Spezialzeichen:
\n Zeilenvorschub
\" "

Der format-Befehl und seine Patterns gehören nicht zum Scheme-Standard, sind also von der in Fluent verwendeten Scheme-Implementierung abhängig; diese ist leider nicht dokumentiert....

for-each Schleife

Führt eine selbst zu definierende Funktion für jedes Element einer oder mehrerer Listen aus:

```
(for-each function list1 list2  ...)
```

Die Anzahl der Funktionsargumente von "function" muss der Anzahl Listen entsprechen.
Verwendbar z.B. für: Fluent-Zonennamen oder –Ids, Dateinamen (wenn sie keine Großbuchstaben enthalten),
Beispiel Temperatur und Wandgeschwindigkeit bei den BCs für mehrere Wandzonen setzen:

```
(define velocity 0.1)
(for-each
  (lambda (zone)
    (ti-menu-load-string
      (format #f "def/bc/wall ~a 0 0 yes giesspulver yes temperature no 1800 yes no no ~a  0 -1
0  no 0 0.5" zone velocity)
    )
    (newline) (display "  ")
  )
  '(
    kok_li kok_re
    kok_innen kok_aussen
    bieg_li bieg_re
    bieg_aussen bieg_innen
    kreis_li      kreis_re
    kreis_aussen  kreis_innen
  )
)
```

Lambda-Befehl zum Definieren von "lokalen" Funktionen:

```
(lambda (arg1 arg2 ...)  ... Funktionswert)
```

Aliases im TUI

Im TUI könne Abkürzungen kreiert werden:

```
(alias 'name scheme-function)
```

Zum Beispiel:

```
(alias 'time (lambda () (display (rpgetvar 'flow-time))))
```

Aufruf im Textinterface:

```
> time
 0.1
```

Argumente können nicht direkt der Scheme-Funktion übergeben werden (immer null Argumente, also lambda ()), sondern müssen durch folgende Funktionen vom Textinterface eingelesen werden:

```
(read-real prompt default)
(read-integer prompt default)
(ti-read-unquoted-string prompt default)
(yes-or-no? prompt default)
```

prompt ist ein String, und *default* der Default-Wert, der zurückgegeben wird, wenn der User nur Return drückt.

Aliases stehen immer automatisch zur Verfügung, wenn ihre Definitionen ins .fluent-file geschrieben werden (siehe oben).

Beispiel: Animation erstellen

Aus den Datenfiles einer instationären Rechnung werden die Einzelbilder für eine Animation erstellt. Die Namen der Datenfiles sind durchnumeriert, mit Anfangs-, Endwert und bestimmter Schrittweite. Fehler, die während der Ausführung eines Fluent-Befehls auftreten, oder ein Abbruch durch Ctrl-C soll auch das Scheme-Programm beenden.

```
(define datfilename "test-")   ;; -> test-0010.dat, test-020.dat, ...
(define first-index  10)
(define last-index  110)
(define delta        10)
(define imagefilename "image-")   ;; -> image-01.bmp, ...

(define (time) (rpgetvar 'flow-time))
(define t0 0)


;;------------------------------------------------------------------------
;; funktion, die die einzelbilder fuer den film erstellt
;;------------------------------------------------------------------------
(define (pp)
  (let
    (
      (break #f)
    )
    (ti-menu-load-string "display/set/hardcopy/driver/tiff") ;; TIFF-Format einstellen
    (ti-menu-load-string "display/set/hardcopy/color-mode/color") ;; Default ist "grey"
    (do   ((j first-index (+ j delta))   ;; datfile   startwert und delta
           (i  1 (+ i  1)))              ;; imagefile startwert und delta
          ((or (> j last-index) break))  ;; datfile   endwert
      (set! break (not (and
        (ti-menu-load-string
          (format #f "file/read-data ~a~04d.dat" datfilename j))
        (begin (if (= i 1) (set! t0 (time))) #t)
        (disp)
        (system "rm temp.tif") ;; hardcopy funktioniert nicht wenn file schon existiert
        (ti-menu-load-string "display/hardcopy temp.tif")
        (system
           (format #f "convert temp.tif ~a~02d.bmp &" imagefilename i))
           ;; convert-Befehl von www.imagemagick.com
      )))
    )
    (if break (begin (newline)(newline)(display "scheme interrupted!")(newline)))
  )
)
```

Beispiel einfache (disp)-Funktion: contour-plot:

```
(define (disp)
  (ti-menu-load-string "display/contour/temperature 290 1673")
)
```

Beispiel (disp)-Funktion: overlay contours/velocity-vectors, eigene Zeit einblenden:

```
(define (disp)
  (and
    (ti-menu-load-string
      (format #f "display set title \"Time = ~5.1f s\"" (- (time) t0))
    (ti-menu-load-string "display/set/overlays no")
    (ti-menu-load-string "display/contour temperature 290 1673")
    (ti-menu-load-string "display/set/overlays yes")
    (ti-menu-load-string "display/velocity-vectors velocity-magnitude 0.0 1.0 5 0")
                                              ;; colored by         min max scale skip
  )
)
```

Beispiel (disp)-Funktion: Iso-Surface generieren, hier Phasengrenze aus VOF-Rechnung mit y-Koordinate (=Höhe) eingefärbt:

```
(define (disp)
  (and
    (ti-menu-load-string "display/surface/iso-surface vof-steel interface-1 , 0.5 ,")
    (ti-menu-load-string "display/set/contours/surfaces interface-1 ()")
    (ti-menu-load-string "display/contour y-coordinate 2.755 2.780")
```

```
      (ti-menu-load-string "display/surface/delete interface-1")
    )
  )
```

Aufruf der (disp)-Funktion zum Testen:

```
> (disp)
```

Aufruf der Funktion zum Erzeugen der Bilder:

```
> (pp)
```

Beispiel: Reportdaten aus Datenfiles

Report-Daten müssen in eine transcript-file geschrieben werden:

```
(ti-menu-load-string "file/start-transcript temp.trn")
(ti-menu-load-string "report/cell-average fluid , temperature")
(ti-menu-load-string "file/stop-transcript")
```

Dazu gibt es alternative einen eigene Scheme-Funktion:

```
(with-output-to-file "temp.trn"
  (lambda ()
    (ti-menu-load-string "report/cell-average fluid , temperature")))
```

Dabei erfolgt keine Ausgabe am Bildschirm.

Transcript-file "temp.trn":

```
report/cell-average fluid , temperature
 volume-average of temperature on cell zones (fluid)
 Volume-weighted average = 300
file/stop-transcript
```

transcript-file in Scheme als Objekte einer Liste ("data") einlesen, nach "=" suchen, das folgende Element stellt den gesuchten Zahlenwert dar:

```
(let
    (
      (data
        (let ((p (open-input-file "temp.trn")))
          (let f ((x (read p)))
            (if (eof-object? x)
              (begin
                (close-input-port p)
                '())
              (cons x (f (read p)))
            )
          )
        )
      )
      (value 0)
    )
    (ti-menu-load-string "! rm temp.trn") (newline)

    (do ((i 0 (+ i 1)) ) ((>= i (length data)))
      (if (eq? (list-ref data i) '=)
        (set! value (list-ref data (+ i 1)))
      )
    )
    value
  )
```

Elegantere und kürzere Fassung der do-Schleife, die keine Variable value benötigt:

```
(cadr (member '= data))
```

Geschachtelte car-cdr Befehle:

```
(cadr x) = (car (cdr x))
```

Aus Wärmeflussbilanz (wird im TUI für alle Surfaces erstellt) Wärmefluss für bestimmte Surfaces herausschreiben:

Format der Bilanz:

```
...
zone 15 (stahl-bodenplatte): 11.2
zone 5 (stahl-kokille): 53.5
zone 6 (schlacke-aussen): 32.4
zone 14 (haube-schlacke): 26.9
...
```

Scheme-Programm:

```
(let
  (
    (p (open-output-file "fluxes.txt")) ;; Ausgabe-Textdatei öffnen
    (n 0)
```

```
    (surfaces '(
      stahl-bodenplatte
      stahl-kokille
      stahl-schlacke
      haube-schlacke
      elektrode-schlacke
      schlacke-innen
      schlacke-aussen
      aufsatz-schlacke
      haube-kuehlung
    ))

  )
  (for-each
    (lambda (filename)
      (if (zero? (modulo n 2))  ;; nur jedes zweite Datenfile nehmen
        (begin
          (ti-menu-load-string
            (format #f "file read-data ~a" filename))

          (ti-menu-load-string "file/start-transcript temp.trn")
          (ti-menu-load-string "report/heat-transfer")
          (ti-menu-load-string "file/stop-transcript")
          (define data                          ;; transcriptfile in "data" laden
            (let ((p (open-input-file "temp.trn")))
              (let f ((x (read p)))
                (if (eof-object? x)
                  (begin
                    (close-input-port p)
                    '())
                  (cons x (f (read p)))
                )
              )
            )
          )
          (ti-menu-load-string "! rm temp.trn")

          (display (time) p)
          (display "   " p)
          (for-each
            (lambda (zone)
              (begin
                (display (list-ref (member (list zone) data) 2) p)
                 ;; fluxwert von zone ermitteln
                (display " " p)
              )
            )
            surfaces
          )
          (newline p)
        )
      )
      (set! n (+ n 1))
    )
    '(
       best-0060.dat   best-0120.dat   best-0132.dat   best-0144.dat   best-0156.dat
       best-0168.dat   best-0180.dat   best-0192.dat   best-0204.dat   best-0216.dat
       best-0228.dat   best-0240.dat   best-0252.dat   best-0264.dat   best-0276.dat
       best-0288.dat   best-0300.dat   best-0312.dat   best-0324.dat   best-0336.dat
     )
  )
  (close-output-port p)
)
```

Data-file-Liste kann in UNIX mit `ls -x *.dat` in Shell erstellt werden.

Beispiel: Werte aus Data- oder Case-files holen

Format von Fluent Files sind geschachtelte Scheme-Listen: z.B. Datenfile:

```
(0 "fluent5.3.18")

(0 "Machine Config:")
(4 (23 1 0 1 2 4 4 4 8 4 4))

(0 "Grid size:")
(33 (10540 21489 10947))
```

```
(0 "Variables:")
(37 (
      (flow-time 3.7)
      (time-step 0.1)
      (periodic/pressure-derivative 0)
      (number-of-samples 0)
      (dpm/summary ())))

(0 "Data:")
(2300 (1 1 1 0 0 1 430)
...
```

Können daher sehr einfach als Scheme-Objekte eingelesen werden. Zum Beispiel Zeit aus Datenfiles lesen und Datenfiles mit Timecode hh:mm:ss umbenennen:

```
(let ((p (open-input-file filename)) (found #f) (t -1))
   (do ((x (read p) (read p))) ((or found (eof-object? x)) (close-input-port p) )
      (if (eqv? (car x) 37) ;; variables
          (begin
            (for-each
              (lambda (y)
                (if (eqv? (car y) 'flow-time)
                  (begin
                    (set! found #t)
                    (set! t (cadr y))
                  )
                )
              )
              (cadr x))
            (newline)
          )
      )
   )
   (ti-menu-load-string (format #f "!mv ~a ~a~a.dat"  filename newdatname (sec->hms t)))
)
```

Funktion sec->hms wandelt Sekunden in hh:mm:ss-Format um:

```
(define (sec->hms t)
  (let*
    (
      (h (truncate (/ t 3600)))  (t1 (- t  (* h 3600)))
      (m (truncate (/ t1 60)))
      (s (truncate (- t1 (* m 60))))
    )
    (format #f "~02d:~02d:~02d" h m s )
  )
)
```

Beispiel: Fluent Zonennamen für UDF exportieren

In UDFs kann über THREAD_ID(t) und THREAD_TYPE(t) zwar ID und Typ einer BC-Zone, nicht jedoch ihr Name angesprochen werden. Die einfachere Variante erzeugt mit der folgenden Scheme-Funktion eine Datenstruktur, die dann in den UDF-Code kopiert werden kann:

```
(define (export-bc-names)
  (for-each
    (lambda (name)
      (display
        (format #f "  {~a, \"~a\", \"~a\"},\n"
          (zone-name->id name)
          name
          (zone-type (get-zone name))
    )))
    (inquire-zone-names)
  )
)
```

In Fluent ausführen:

```
(export-bc-names)
  {26, "wall-wehr-l-shadow", "wall"},
  {2, "fluid", "fluid"},
  {29, "wall-damm-l-shadow", "wall"},
  {15, "wall-damm-l", "wall"},
  {17, "inlet", "mass-flow-inlet"},
  {25, "default-interior", "interior"}
  ...
```

dieser Text muss in den folgenden UDF-Code kopiert werden:

```
#define nc 100
typedef struct zone_info_struct
```

```
{
  int id;
  char name[nc];
  char type[nc];
}
zone_info;

zone_info zone[]={
/*** ab hier aus Fluent-Textinterface kopiert ***/
  {26, "wall-wehr-l-shadow", "wall"},
  {2, "fluid", "fluid"},
  {29, "wall-damm-l-shadow", "wall"},
  {15, "wall-damm-l", "wall"},
  {17, "inlet", "mass-flow-inlet"},
  {25, "default-interior", "interior"}
  ...
};
#define n_zones (sizeof(zone)/sizeof(zone_info))
```

Nun können die Zonennamen im UDF-Code über zone[i].name angesprochen werden.

Die Alternative ist eine Scheme-Funktion, die die Zonen als String in ein RP-Variable schreibt. Der Vorteil ist, dass diese RP-Variable im Case-File mitgespeichert wird, und so mehrere verschiedene Case-files mit der selben UDF funktionieren, ohne diese neu zu compilieren. Die Scheme Funktion muss also nur einmal beim Aufsetzen des Case-Files aufgerufen werden. Nachteil: funktioniert wegen RP-Variablen nicht so einfach im Parallel-Solver.

```
(define (bc-names->rpvar)
  (let ((zone-data ""))
    (for-each
      (lambda (name)
        (set! zone-data
          (format #f "~a ~a ~a ~a  " zone-data
            (zone-name->id name)
            name
            (zone-type (get-zone name))
      )))
      (inquire-zone-names)
    )
    (display zone-data)
    (rpsetvar* 'zone-names 'string zone-data)
  )
)
```

Dabei wird folgende Funktion verwendet:

```
(define (rpsetvar* var type value) ;; create cortex variable if undefined
  (if (not (rp-var-object var))
    (rp-var-define var value type #f)
    (rpsetvar var value)
  )
)
```

UDF-Code:

```
#define max_n_zones 200
#define max_zonenamechars 200

/* globale Variablen */
char zone_name[max_n_zones][max_zonenamechars];
char zone_type[max_n_zones][max_zonenamechars];
#define THREAD_NAME(t) zone_name[THREAD_ID(t)]

/* lokale Variablen */
char *rps,s[1000];
int i,n;

for(i=0; i<max_n_zones; i++) zone_name[i][0]=0; /* initialisieren */
rps = RP_Get_String("zone-names");
while(rps[0])
{
  sscanf(rps,"%s%n", s,&n); rps += n;
  i = atoi(s);
  sscanf(rps, "%s%s %n", zone_name[i], zone_type[i],&n); rps += n;
}
```

Hier gibt es sogar ein Makro THREAD_NAME(t), mit dem die Zonennamen angesprochen werden können.

Iterationssteuerung

Vor allem für instationäre Rechnungen interessant;
Cortex-Variablen (können auch gesetzt werden!):
Zeit (*t*):

```
flow-time
```

Nummer des aktuellen Zeitschritts (*N*):

```
time-step
```

Größe des Zeitschritts (Δ*t*):

```
physical-time-step
```

Liste der gespeicherten Iterationen (aktuelle Iteration zuerst, dann die vorhergehende etc.)

```
(residual-history "iteration")
```

daraus die aktuelle Iterationszahl:

```
(car (residual-history "iteration"))
```

Listen der Residuen (Einträge entsprechend der "iteration"-Einträge):

```
(residual-history "continuity")
(residual-history "x-velocity")
(residual-history "temperature")
...
```

TUI-Kommandos zum Iterieren (stationär oder wenn instationär am aktuellen Zeitschritt weitergerechnet werden soll):

```
solve/iterate number-of-iterations
```

Instationär:

```
solve/dual-time-iterate number-of-timesteps max-iterations
```

Schrittweite muss z.B. mit

```
(rpsetvar 'physical-time-step 0.1)
```

gesetzt werden.

Mittels dieser Kommandos und Variablen können komplexe Iterationssteuerungen programmiert werden, z.B.

- Instationär: Angabe von zu rechnendem Zeitintervall statt Anzahl der Zeitschritte;
- automatische Fortsetzung der instationären Rechnung nach Abbruch
- veränderliche Schrittweite nach Tabelle (z.B. 10 s lang dt=0.1 s, dann 20 s lang dt = 0.2 s usw.);
- Auto-Save zu bestimmten Zeitpunkten der Rechnung mit Zeitkode im Dateinamen (data-00:10.dat), konstante Zeitabstände trotz variabler Schrittweite
- eigene adaptive Schrittweitensteuerung
- mitführen eines Sicherungsdatenfiles für langwierige, absturzgefährdete Rechnungen: alle x Iterationen Datenfile abspeichern, und dann das vorher abgespeicherte Datenfile wieder löschen.

Besonderheiten des Fluent-Schemes

eval-Befehl und environment

```
(eval expression environemnt)
```

Das Standard Scheme-Environment (the-environment) beinhaltet sämtliche Symbole, die zusätzlich zu den Scheme-Standard Symbolen definiert sind, also alle vom User über (define ...) und auch von Fluent(!) definierten Variablen und Funktionen. Wenn diese beim Auswerten von *expression* nicht benötigt werden, kann stattdessen die leere Liste '() oder #f verwendet werden. Beispiel:

```
(define x 3)
(define y '(+ x 2)) ;; y ist eine Liste mit Elementen +, x, 2
(eval y (the-environment)) ;; Liste y wird als Scheme-Befehl interpretiert; Ergebnis: 5
```

Mit den folgenden Funktionen kann überprüft werden, ob ein Symbol definiert ist (bound?) und ob ihm ein Wert zugewiesen ist (assigned?):

```
(symbol-bound? 'symbol (the-environment))
(symbol-assigned? 'symbol (the-environment))
```

Listen-Befehle

```
(list-head list n)
(list-tail list n)
```

format-Befehl

Siehe Abschnitt „format-Befehl“ Seite 7.

system-Befehl

Ausführen von Shell-Kommandos, z.B.:

```
(system "rm temp.jou")
```

Fluent-Variablen und Funktionen

Sämtliche Fluent-Variablen und Funktionen sind im environment (the-environment) definiert, wenn auch nicht dokumentiert. Siehe Abschnitt „Fluent-Scheme Environment“.

Scheme-Literatur

Leider gibt es keine Literatur, die speziell auf Fluent-Scheme eingeht. Da Scheme eine sehr mächtige Sprache ist, mit der anspruchsvolle Anwendungen wie Künstliche Intelligenz realisiert werden können, gehen die meisten Scheme-Bücher viel weiter in die Tiefe, als es für Fluent notwendig ist. Ich verwende das Buch:

- R. Kent Dybvig, The Scheme Programming Language, Second Edition, 1996 Prentice Hall PTR.

Fluent empfielt folgende Scheme-Links im Web:

- http://www.swiss.ai.mit.edu/projects/scheme
- http://www.schemers.org/

Einige Scheme-Beispiele zu FLUENT sind mittlerweile im FLUENT User Service Center im „Online Technical Support“ zu finden:

- http://www.fluentusers.com/

Fluent-Scheme Standardfunktionen

Die folgenden Funktionen gehören zum Fluent-Scheme-Standard und scheinen nicht im Environment auf. Ich habe diese Liste aus der Fluent-Programmdatei extrahiert, musste aber feststellen, dass sie leider unvollständig ist: es gibt Funktionen, die weder in dieser Liste noch im Environment enthalten sind (z.B. list-head, list-tail und list-remove). Inzwischen habe ich ein paar dieser fehlenden Funktionen hinzugefügt (10-2004).

```
<
<=
=
>
>=
-
/
*
+
abs
access
acos
and
append
append!
apply
asin
assq
assv
atan
atan2
begin
bit-set?
boolean?
call/ccinput-port?
car
cdr
ceiling
char<?
char=?
char>?
char?
char-alphabetic?
char-downcase
char->integer
char-lower-case?
char-numeric?
char-ready?
char-upcase
char-upper-case?
char-whitespace?
chdir
clear-bit
close-input-port
close-output-port
closure?
closure-body
cond
cons
continuation?
copy-list
cos
cpu-time
debug-off
debug-on
define
display
do
dump
echo-ports
env-lookup
eof-object?
eq?
equal?
eqv?
error
error-object?
err-protect
err-protect-mt
eval
exit
exp
expand-filename
expt
fasl-read
file-directory?
file-exists?
file-modification-time
file-owner
float
floor
flush-output-port
for-each
foreign?
foreign-data
foreign-id
format
format-time
gc
gc-status
general-car-cdr
getenv
hash-stats
if
int
integer?
integer->char
interrupted?
lambda
length
let
let*
list
list-head
list->string
list-tail
list->vector
local-time
log
log10
logical-and
logical-left-shift
logical-not
logical-or
logical-right-shift
logical-xor
machine-id
make-foreign
make-string
make-vector
map
max
member
memq
memv
min
mod
newline
not
nt?
null?
number?
number->string
oblist
open-file
open-input-file
open-input-string
open-output-file
open-output-string
or
output-port?
pair?
peek-char
port-echoing?
port-name
procedure?
procedure-name
putenv
quotient
read
read-char
real?
remainder
remove-file
rename-file
reverse
set!
set-bit
set-car!
set-cc
set-cdr!
set-echo-ports!
sin
sqrt
stack-object
stack-size
string<?
string=?
string>?
string?
string-append
string-ci<?
string-ci=?
string-ci>?
string-length
string->list
string->number
string-ref
string-set!
string->symbol
substring
substring-fill!
substring->list
subvector-fill!
subvector->list
symbol?
symbol-assigned?
symbol-bound?
symbol->string
system
system
tan
the-environment
time
toggle-bit
trace-ignore
trace-off
trace-on
truncate
unix?
valid-continuation
vector?
vector-length
vector-ref
vector-set!
vms?
write
write-char
write-string
```

Fluent-Scheme Environment

Im folgenden sind alle Elemente des Fluent-Scheme Environments aufgelistet. Wenn es sich um Funktionen handelt, ist der Name in eigeklammert – eventuell erforderliche Parameter sind nicht angegeben. Bei Listen ist nur *list* vermerkt, weil die Listen teilweise sehr umfangreich sind, ansonsten ist der Wert der Variablen angegeben, oder *n/a* falls die Variable keinen Wert hat.

```
*solver-command-name*  fluent
(client-file-version)
(gui-get-selected-thread-ids)
(gui-show-partitions)
(gui-memory-usage)
(grid-show)
(rampant-menubar)
(gui-reload)
(ti-avg-xy-plot)
(ti-2d-contour)
(ti-avg-contour)
(ti-set-turbo-topo)
(gui-turbo-twod-contours)
(gui-turbo-avg-contours)
(gui-turbo-xyplots)
(gui-turbo-report)
(gui-set-topology)
(ti-turbo-define)
(ti-write-turbo-report)
(ti-compute-turbo-report)
(write-turbo-data)
(gui-turbo-define)
(correct-turbo-defenition)
(delete-turbo-topology)
(define-turbo-topology)
(setturbovar)
(getturbovar)
(add-turbo-post-menu)
(solve-controls-summary)
(gui-solve-iterate)
(gui-solve-controls-mg)
(gui-solve-controls-solution)
(order/scheme-name->type)
(order/scheme-type->name)
order/schemes   list
(name-list)
(print-name->attribute-list)
(print-name->pick-name)
(print-name)
(get-eqn-units-patch)
(get-eqn-units-default)
(get-eqn-var-default)
(get-eqn-var)
(set-eqn-var)
(get-eqn-index)
(symbol->rpvar)
(inquire-equations)
(gui-solve-set-limits)
(gui-solve-set-ms)
(gui-patch)
(gui-init-flow)
(gui-solar-calculator)
(gui-particle-summary)
(models-summary)
(gui-dpm-sort)
(gui-models-dpm)
(gui-models-viscous)
(gui-user-memory)
(gui-udf-on-demand)
(gui-udf-hooks)
(gui-uds)
(update-uds-domain-id-list)
(gui-models-soot)
(gui-models-nox)
(gui-vf-para)
(gui-surface-glob)
(gui-ray-trace)
(gui-models-radiation)
(set-radiation-model)
(gui-models-species)
(gui-models-multiphase)
(new-phase-name)
(multiphase-model-changed)
(gui-periodic-settings)
(gui-models-solidification)
(gui-models-energy)
(gui-operating-conditions)
(gui-models-solver)
cxsweep.provided       #t
(cx-delete-keyframe)
(cx-insert-keyframe)
(cx-display-frame)
keyframes       list
(create-mpeg)
(mpeg-open)
(play-mpeg)
(cx-set-mpeg-compression)
*mpeg-options*
*mpeg-qscale*   8
*mpeg-bsearch*  CROSS2
*mpeg-psearch*  EXHAUSTIVE
*mpeg-range*    8
*mpeg-pattern*  IBBPBB
*mpeg-compression?*    #f
*mpeg-command*  mpeg_encode
(cx-animate)
(cx-gui-animate)
(video-picture-summary)
(video-summary)
(cx-video-use-preset)
(cx-video-show-picture)
(cx-video-set-picture)
cx-video         n/a
(cx-video-show-options)
(cx-video-set-options)
(cx-video-close)
(cx-video-open)
(cx-video-panel)
(cx-video-enable)
*cx-video*      #t
cxvideo.provided       #t
cxanim.provided        #t
(ti-del-hxg)
(ti-set-hxg)
hxc-eff-vector  #f
(get-avail-zones)
(remove-frm-list)
(get-thread-id)
(get-center)
(set-porous-res)
(set-porous-dirs)
(update-hxc-model)
(initialize-hxc-model)
(free-hxc-model)
(heat-exchanger?)
(ti-get-res-hxc-input)
(get-hxc-opr)
(get-v)
(set-drop-down-widgets)
(set-integer-widgets)
(set-real-widgets)
(one-zone-group)
(%init-hxc-model)
(%free-hxc-model)
```

```
alstom  #f
heatxc-models   list
heatxc-groups   list
heatxc-geom     list
(get-group-id)
(get-hxg-id)
(new-hxg-id)
(ti-get-model-input)
(ti-get-hxc-input)
(ti-set-heatxc-model)
(ti-set-hxc)
(ti-hxc-report)
auto-set-porous?        #f
(update-model-list)
(update-heatxc-list)
(draw-all-macros)
(gui-heatxc-groups)
(gui-heatxc-models)
(gui-heat-exchanger)
(pdf-init)
(inquire-species-names)
(inquire-n-species)
(surface-species-number)
(surface-species-names)
(get-residual-norms-at)
(residual-default-setting)
(ti-client-residuals-reset)
(client-residuals-reset)
(client-support-residuals-reset?)
(residual-set)
(residual-history)
(unset-residual-norms!)
(set-residual-norms-by-max!)
(set-residual-norms-at!)
(gui-monitor-residuals)
clres.provided  #t
(gui-solution-animation)
(aniseq->path)
(aniseq->window)
(aniseq->storage)
(aniseq->monitor)
(aniseq->display)
(aniseq->name)
(ani-monitor-delete)
(remove-ani-sequence)
(change-storage-type)
(replace-sequence-by-name)
(change-sequence-window)
(sequence-path-rename)
(sequence-rename)
(add-animation-monitor)
(ani-monitor-update)
(set-animon-active?)
(animon->active?)
(animon->cmdstr)
(animon->when)
(animon->freq)
(animon->seqnum)
(animon->name)
(build-ani-monitor-list-element)
(remove-ani-xy-vars)
xy-vars-list    list
(set-xy-vars)
(ani-save-xy-vars)
(show-ani-monitors)
(show-one-ani-monitor)
(get-ani-monitors)
(ani-render-var-rename)
(ani-show-thunk-titles)
(ani-restore-thunk+title)
(ani-save-thunk+title)
(ani-remove-thunk+title)
(ani-rename-monitor-thunk+title)
(ani-restore-render-vars)
(ani-save-render-vars)
(ani-monitor-active?)
(ani-monitor-name->seq)
(ani-monitor-seq->name)
(ani-monitor-deactivate)
(ani-monitor-activate)
(ani-monitor-change-freq)
(ani-monitor-rename)
(remove-ani-monitor)
(add-ani-monitor-command)
(run-ani-monitors)
(animation-init)
(monitor-statistics-init)
solve/monitors/statistic-menu  list
(init-stats)
(gui-monitor-statistics)
(monitor-statistics)
(gui-monitor-forces)
(clear-monitor-forces)
(monitor-forces-init)
(monitor-forces)
(monitor-execute-at-end-transient)
(monitor-execute-at-end)
execute-string
(gui-monitor-commands)
(monitor-command-init)
(multiphase-surf-mon-compat)
(gui-monitor-surface)
(monitor-surface-init)
(multiphase-vol-mon-compat)
(gui-monitor-volume)
(monitor-volume-init)
main-menu       list
grid-menu       list
turbo-menu      list
(turbo-set-current-topology)
(turbo-avg-xy-plot)
(turbo-2d-contours)
(turbo-avg-contours)
(write-turbo-report)
(compute-turbo-report)
adapt-menu      list
adapt/set-menu  list
reorder-menu    list
reorder-method-menu     list
(ti-reorder-using-cell-functions)
(ti-reorder-using-cell-distance)
report-menu     list
report/reference-menu  list
report/reference/compute-menu  list
display-menu    list
plot-menu       list
define-menu     list
(ti-define-turbo-topology)
ud-menu  list
(ti-udm)
(ti-execute-at-end)
(ti-udf-on-demand)
(ti-ud-hooks)
profile-menu    list
operating-conditions-menu      list
(udf-models-changed)
(set-uds-defaults)
models-menu     list
solver-menu     list
radiation-menu  list
heat-exchanger-menu     list
(ti-uds)
(update-pollutant-solve)
soot-menu       list
nox-menu        list
(check-fuel-name-soot)
(check-species-name2)
(check-species-name1)
(check-fuel-name)
s2s-menu        list
dtrm-menu       list
(multiphase-menu)
turbulence-menu         list
turbulence-expert-menu  list
multiphase-turbulence-menu     list
near-wall-treatment-menu       list
```

```
species-menu    list
(ti-read-pdf-helper)
(enable-mixture-matl)
solve-menu      list
initialize-menu         list
initialize/compute-menu         list
(set-domain-averaged-derived-flow-inits!)
solve/set-menu  list
(set-eqn-vars)
(set-eqn/mg-controls)
(set-eqn/scheme)
(set-eqn/solve)
(set-eqn/relax)
(set-eqn/default)
monitors-menu   list
solve/monitors-menu     list
phase-menu      list
pc-menu  list
bc-menu  list
modify-zones-menu       list
display/set-menu        list
file-menu       list
interpolate-menu        list
surface-cluster-menu    list
file/export-menu        list
file/import-menu        list
file/import/cgns-menu   list
file/import/partition-menu      list
file/autosave-menu      list
(allow-v2f-model)
allow-v2f-model?        #f
cx-scene-menu   list
(ti-set-geometry)
(delete-cb)
(update-indices)
(ti-color-def)
(ti-transform)
(ti-time-step)
(ti-path-attr)
(ti-iso-sweep)
(ti-select-box-edge)
get-index       #f
ti-num-geoms    0
ti-selected-index       list
ti-selected-geom        list
ti-selected-type        list
ti-selected-segment     list
(cx-scene-update-geoms)
(cx-scene-default-value)
(scene-insert-order)
(cx-scene-insert-geoms)
(update-all-graphics)
(cx-scene-update-graphics)
(recreate-geom?)
(restore-cx-globals)
(save-cx-globals)
(close-gr-segments)
(open-gr-segments)
(cx-scene-draw-cmap)
(redisplay-all)
(cx-scene-set-iso-surf)
(cx-get-scene-update)
(cx-set-scene-update)
(cx-scene-list-geometry)
(scene-get-string)
(cx-show-user-option)
(cx-transform-highlight)
(cx-draw-bbox)
(cx-flush-bbox)
(cx-scene-show-bbox)
(cx-set-vv-attr)
(cx-set-profile-attr)
(cx-set-dpm-attr)
(cx-set-path-attr)
(cx-set-contour-attr)
(get-viz-iso-surf-id)
(iso-surface-ancestor)
(derived-from-iso-surface)
(show-surface-units)
(show-surface-quantity)
(show-surface-type)
*cx-scene-panel-present*       #f
(cx-gui-scene)
(cx-gui-bbox-frame)
(insert-projections)
(inc-geoms)
(insert-planes)
(add-delta)
(cx-display-bnd-frame)
cx-frame-growth-factor 0.01
cx-frame-domain?        #t
cxticks.provided        #t
cxscene.provided        #t
ti-non-reflecting-menu list
(remesh-local-prism-faces)
(print-remesh-cell-marks)
(draw-remesh-cell-marks)
(mark-remesh-cells)
(refine-coarsen-on-skewness-size)
(remesh-local-cells)
(repartition-remeshing-zones)
(print-volume-skewness-limits-per-zone)
(check-dynamic-mesh)
(print-dynamic-forces-moments)
dynamic-mesh-menu       list
dynamic-bc-menu         list
(eval-udf)
debug-dynamic-functions         #f
(clear-dynamic-functions)
(update-dynamic-functions)
(cancel-dynamic-function)
(register-dynamic-function)
(ti-position-starting-mesh)
(get-remesh-cell-threads)
(update-solver-thread-names)
(set-dt-mask)
(mask-names->mask)
(mask->mask-names)
cell-element-type-alist         list
(ti-modify-lift)
(ti-print-plot-lift)
(plot-valve-lift)
(print-valve-lift)
(ti-delete-internal-layer)
(ti-insert-internal-layer)
(ti-remove-layer)
(ti-insert-layer)
(remove-layer)
(insert-layer)
(delete-internal-layer)
(insert-internal-layer)
(cleanup-thread-list)
(update-in-cylinder-monitors)
(monitor-crank-angle)
(gui-ic-event-playback)
(ic-event-playback)
(gui-ic-events)
(ic-event-hook)
(handle-ic-event)
(inside-range)
(shift-down)
(shift-up)
angle-tol       1e-05
event-callback-alist    list
(remove-internal-layer-callback)
(insert-internal-layer-callback)
(remove-boundary-layer-callback)
(insert-boundary-layer-callback)
(change-motion-attr-callback)
(change-time-step-callback)
(delete-si-callback)
(create-si-callback)
(copy-bc-callback)
(change-zone-type-callback)
remove-internal-layer-event     9
insert-internal-layer-event     8
```

```
remove-layer-event      7
insert-layer-event      6
change-time-step-event 5
change-motion-attr-event       4
delete-sliding-interface-event         3
create-sliding-interface-event         2
copy-zone-event         1
change-zone-event       0
(lift->angle)
(crank-angle->time)
(crank-angle->absolute-time)
(time->crank-angle)
(nth-ic-cycle)
(time->absolute-crank-angle)
(fmod)
(gui-dynamic-zone-preview)
(display-surfaces)
(advance-mesh)
(auto-hardcopy)
(animate-motion)
(gui-motion-preview)
(preview-motion)
(update-all-geom-positions)
(update-one-geom-position)
(v3-rsub)
dz-ani-storage-name     dynamesh_preview
dz-ani-storage-type     2
dz-ani-sequence         -1
dz-animate?     #f
dz-nstep        1
dz-display-frequency    1
dz-display?     #t
dz-hardcopy?    #f
(contour-node-displacement)
(compute-cg-from-profile)
(zone-selection)
(disable-items)
(enable-items)
(hide-items)
(show-items)
(gui-dynamic-zones)
(gui-models-moving-grid)
(ti-list-dynamic-threads)
(ti-reset-dynamic-thread)
(delete-adjacent-dz)
(delete-dynamic-thread)
(ti-set-dynamic-thread)
(ti-enable-dynamic-mesh)
(enable-dynamic-mesh)
(change-dz-attr)
(new-copy-of-dz)
(set-dz-attr)
(get-dz-attr)
(get-dynamic-thread)
deforming-motion        3
user-defined-motion     2
solid-body-motion       1
no-motion       0
(motion-type-name->id)
(motion-type-id->name)
(new-dz)
(read-reference-vol)
(write-reference-vol)
(set-remesh-repartition-thresholds)
(set-sizing-function-defaults)
(set-sizing-function-dimension-defaults)
(update-dynamesh-hooks)
(allow-dynamic-mesh)
allow-dynamic-mesh?     #t
preview-auto-save       #f
execute-ic-event        #t
debug-ic-event  #f
(ti-export-event)
(ti-import-event)
(free-dynamic-mesh)
(dynamic-thread-vars-compat)
(dynamic-thread-list)
(unthread-dynamic-thread)
(thread-dynamic-thread)
(download-dynamic-threads)
(create-case-dynamic-threads)
(gui-s2s)
(gui-dtrm)
(display-sample-points)
(gui-samples-manage)
(pick-sample)
(sample-name)
(delete-sample)
(list-sample-fields)
(list-samples)
(plot-sample-histograms)
(update-sample-list-cache)
(sample-list)
(ti-dpm-sample-report)
(dpm-parallel-shared-memory)
(dpm-parallel-message-passing)
(ti-particle-tracks)
(pathline-summary)
(dpm-summary)
(dpm-iteration)
(pick-particle-cell-function)
(inquire-particle-cell-functions)
(dpm-display-path-lines)
(injection-types)
law-list        list
property-list   list
(pick-law)
(custom-laws?)
(ti-recover-law)
(ti-convert-law)
(ti-dpm-law-options)
(dpm-law-options)
(dpm-default-laws)
(dpm-material-type)
(combusting-not-multi-surface?)
(multi-surface?)
(comb-method)
(pick-particle-type)
(pick-injection-type)
(pick-dpm-material)
(pick-stream)
(pick-species)
(pick-generic)
(dpm-is-cloud-on?)
gui-law-define   n/a
(pick-injection)
(inquire-injection-names)
dpm-menu        list
dpm-injections-menu     list
(dpm-bcs-available?)
(dpm-material-types)
(activate-injection-surfaces)
(dpm-change-material-name)
(dpm-used-material-names)
gui-dpm-display-particle-traces         n/a
(get-all-dpm-material-names)
(download-injections)
(create-case-injections)
(gui-manage-injections)
(reset-injections)
(free-injections)
acoustics-menu  list
(ti-receivers-specification)
(ti-source-specification)
(ti-read-and-compute)
(gui-models-acoustics)
(acoustic-reader-gui)
(receivers-gui)
(acoustics-write-sound-gui)
(thread-acoustic-gui)
(append-ac-file)
(write-next-acoustic-timestep)
(open-acoustic-file-for-write)
(acoustics-trans-freq-proc)
(acoustics-file-write-and-compute)
(append-file-name-to-index-file)
```

```
(get-receiver)
exporting-data-only    #f
(acoustics-model-changed)
(allow-acoustics-model)
allow-acoustics-model? #t
(ti-crevice-menu)
(ti-make-crevice-threads)
(new-crevice-case)
(crevice-summary)
(valid-flmon-chkpnt-dir?)
flmon-chkpnt-dir       list
(flmon-init)
flmon-running?  #f
(valid-sge-chkpnt-dir?)
sge-chkpnt-dir  list
(sge-init)
(lsf-spawn)
(valid-lsb-chkpnt-dir?)
lsb-chkpnt-dir  list
(lsf-init)
(monitor-send-exit-sig)
(monitor-new-procs)
(contact-monitor)
(monitor-config-file)
(write-kill-script)
(exit-restart-file)
(restart-commands)
(checkpoint)
(create-checkpoint-fnbase)
(create-checkpoint-filename)
(remove-checkpoint-files)
(set-checkpoint-variable)
(chkpnt-dir)
(valid-chkpnt-dir?)
(valid-exit-file?)
(valid-check-file?)
(create-exit-file)
(create-check-file)
last-data-filename     #f
last-case-filename     #f
save-rc-filename       #f
checkpoint/exit-filename       /tmp/exit-
fluent
checkpoint/check-filename      /tmp/check-
fluent
checkpointed?   #f
(benchmark)
(debug-node)
(debug-all)
(debug-client)
(attach-debugger)
(attach-ibm-dbx)
(attach-sgi-dbx)
(attach-sun-dbx)
(attach-ladebug)
(attach-gdb)
(repartition-by-zone)
(gui-load-balance)
(gui-show-virtual-machine)
(gui-network-configure)
(gui-hosts-data-base)
(gui-auto-partition-grid)
(gui-partition-grid)
parallel-menu   list
partition-menu  list
partition/auto-menu    list
(ti-partition-auto)
partition/set-menu     list
(ti-partition)
(pick-partition-function)
parallel/network-menu  list
parallel/timer-menu    list
parallel/set-menu      list
(gui-reorder-domain)
(ok-to-invalidate-case?)
(spawn-from-file)
(spawn-from-list)
(spawn-compute-node)
(add-to-available-hosts)
(first-word)
(make-hosts-db-available)
(read-partition-id)
(read-partition-id-list)
(disable-load-balance-after-adaption)
(enable-load-balance-after-adaption)
(check-partition-encapsulation)
use-default-auto-partition?    #t
case-file-partition-pretest?    n/a
case-file-partition-method      n/a
(ti-migrate-marked-cells)
(migrate-marked-cells)
(list->user@host-dot)
(user@host->list)
(create-hosts-db)
(delete-from-available-hosts)
list-compute-nodes     list
list-of-available-hosts        list
list-hosts      list
(mkill?)
(valid-partition-id?)
(ti-translate-grid)
(ti-scale-grid)
(translate-grid)
(gui-translate-grid)
(scale-grid)
(gui-scale-grid)
cxgrid.provided        #t
(ti-summary)
(gui-summary)
(gui-volume-integrals)
(ti-cell-thread-integral)
(ti-cell-thread-volume)
(gui-reference-values)
(ti-plot-histogram)
(ti-print-histogram)
(gui-histogram)
(plot-histogram)
(print-histogram)
(client-histogram-bins)
(client-histogram-max)
(client-histogram-min)
*cx-histogram-location*        cells
cxreports.provided     #t
(gui-wall-reports)
(print-wall-moments)
(print-wall-forces)
(wall-moments)
(wall-forces)
(viscous-moment)
(viscous-force)
(pressure-moment)
(pressure-force)
(iprint-wall-moments)
(iprint-wall-forces)
(gui-thread-reports)
(thread-integrals)
(sort-threads-by-name)
(shell-heat-transfer)
(rad-heat-transfer)
(heat-transfer)
phase-mass-flow         n/a
phase-volume-flow       n/a
(uds-flow)
(species-mass-flow)
(mass-flow)
(cur-to-si-unit)
(si-to-cur-unit)
(ti-set-reference-var)
(advance-oned-solution)
(update-wave-bcs)
(start-oned-library)
(free-oned-library)
(draw-oned-cell)
library-list    list
(oned-library-gui)
(display-profile-points)
```

```
(ti-write-profiles)
(gui-write-profiles)
(gui-profiles-manage)
(gui-profile-orient)
(axial-profile-to-xy)
(radial-profile-to-xy)
(ti-conserve-total-enthalpy)
(ti-create-mixing-plane)
(ti-conserve-swirl)
(ti-set-pressure-level)
ti-mixing-plane-menu   list
(gui-mixing-plane)
(initialize-mixing-plane-profiles)
(update-mixing-planes)
(write-profiles)
(pick-profile)
(profile-name)
(update-profiles)
(delete-profile)
(list-profile-fields)
(list-profiles)
(update-user-function-list-cache)
(user-function-list)
(update-profile-list-cache)
(profile-list)
ti-real-gas-init       list
(ti-ud-real-gas)
(open-udrg-library)
(ti-nist-real-gas)
(ti-real-gas-dataname)
(open-rgas-library)
(default-property-constant)
(update-species-variable-lengths)
(properties-changed)
(species-changed)
(client-list-reactions)
(client-material-name-changed)
(client-set-material!)
(client-used-material-names)
(client-material-types)
(client-property-names)
(client-property-list)
(ti-import-chemkin)
materials-menu  list
(ti-copy-material-by-formula)
(ti-copy-material-by-name)
materials/database-menu        list
(import-chemkin-mechanism)
(load-pdf-species)
(property-method-assq)
(property-method-name-prmx)
(property-method-name)
(default-property-data)
(add-mixture-name)
(qlist-)
(total-reaction-list)
(mlist-)
(alist-)
(material-site-nspecies)
(material-site-species-names)
(material-surface-nspecies)
(material-surface-species-names)
(material-volumetric-nspecies)
(material-volumetric-species-names)
(material-nspecies)
(material-species-names)
(material-species-names-pair)
(material-profiles)
(reaction-list-for-material)
(material-prop)
(material-types)
(material-mixture-pair)
(material-type)
(material-formula)
(mixture-material-name)
(material-name)
(material-in-use-by-mixture)
(dpm-check-material-in-use)
*max-material-name-len*        25
all-material-types     list
(materials-summary)
(client-property-methods)
(reset-prop-methods-cache)
(update-case-material-properties)
(ti-delete-material)
ti-create-material      n/a
(ti-copy-material)
(ti-change-material)
(gui-materials-manage)
(list-database-materials)
(database-property-methods)
(get-database-material-copy)
(get-database-material-copy-by-formula)
(get-database-material-by-formula)
(get-database-material)
(property-units)
(property-name)
(inquire-database-material-names)
(inquire-material-names-all)
(inquire-material-names)
(get-default-material-name)
(get-names-of-type)
(get-first-material-name)
(pick-material-type)
(pick-database-material-by-formula)
(pick-database-material-by-name)
(pick-database-material)
(pick-material-all-with-prompt)
(pick-material-with-prompt)
(pick-material-all)
(pick-material)
(list-database-properties)
(list-properties)
(list-materials)
(get-material-copy)
(get-materials-of-type)
(get-material)
(set-material-properties!)
(set-material!)
(set-all-materials!)
(get-n-materials)
(get-all-materials)
(delete-material)
(update-material-properties)
(update-case-materials)
(create-case-materials)
(free-materials)
(create-mixture-material)
(copy-database-material)
(create-material)
(upload-materials)
(download-materials)
cxprop.provided        #t
(draw-bridge-nodes)
(free-bridge-nodes)
(fill-bridge-nodes)
(gui-smooth-grid)
(ti-make-hanging-interface)
(ti-swap-mesh-faces)
(swap-mesh-faces)
(ti-smooth-mesh)
(smooth-mesh)
(gui-yplus-adapt)
(gui-volume-adapt)
(gui-region-adapt)
(gui-isovalue-adapt)
(gui-gradient-adapt)
(gui-boundary-adapt)
(gui-manage-mark)
(gui-display-mark-options)
(gui-adapt-controls)
(gui-display-contours-or-error)
(create-new-interior-threads)
(set-register-ncrsn!)
(set-register-nrefn!)
(set-register-cbit!)
```

```
(set-register-rbit!)
(set-register-type!)
(set-register-name!)
(set-register-id!)
(register-ncrsn)
(register-nrefn)
(register-cbit)
(register-rbit)
(register-type)
(register-name)
(register-id)
(mark-percent-of-ncells)
(ti-mark-percent-of-ncells)
mask-register   #f
refn-register   #f
(ti-free-parents)
(refine-mesh)
(ti-refine-mesh)
(adapt-mesh)
(ti-adapt-mesh)
(register-invert)
(ti-register-invert)
(ti-mask-invert)
(swap-refn-crsn)
(ti-swap-refn-crsn)
(draw-marked-cells)
(draw-node-flags)
(ti-draw-marked-cells)
(mark-inout-iso-range)
(ti-mark-inout-iso-range)
(mark-inout-shape)
(ti-mark-inout-shape)
(ti-shape-define)
(mouse-shape-define)
(auto-refine-level)
(mark-with-refine-level)
(ti-mark-with-refine-level)
(mark-with-volume-change)
(ti-mark-with-volume-change)
(mark-boundary-cells-per-thread)
(ti-mark-boundary-cells-per-thread)
(mark-with-volume)
(ti-mark-with-volume)
(mark-with-boundary-volume)
(ti-mark-with-boundary-volume)
(mark-with-yplus-per-thread)
(ti-mark-with-ystar-per-thread)
(ti-mark-with-ystar)
(ti-mark-with-yplus-per-thread)
(ti-mark-with-yplus)
(mark-with-gradients)
(ti-mark-with-gradients)
(ti-adapt-to-boundary-cells)
(ti-adapt-to-refine-level)
(ti-adapt-to-volume-change)
(ti-adapt-to-ystar-per-thread)
(ti-adapt-to-ystar)
(ti-adapt-to-yplus-per-thread)
(ti-adapt-to-yplus)
(ti-adapt-to-gradients)
(ti-adapt-to-default-register)
(adapt-to-register)
(ti-adapt-to-register)
(list-registers)
(combine-list-of-registers)
(combine-registers)
(ti-combine-registers)
(get-mask-registers)
(get-refn-registers)
(get-all-registers)
(get-register)
(ti-read-mask-register)
(ti-read-refn-register)
(ti-read-register-list)
(ti-read-register)
(toggle-register-type)
(ti-toggle-register-type)
(fill-crsn-register)
(ti-fill-crsn-register)
(limit-marked-cells)
(ti-limit-marked-cells)
(ti-count-marked-cells)
(replace-register)
(free-registers)
(ti-free-registers)
(destroy-register)
(ti-destroy-register)
(get-def-register)
(create-copy-of-register)
(create-register)
adapt-env       list
(surface-ids->name-pair)
(thread-surface)
(ithread-surface)
(thread-coordinates)
(thread-values)
(inquire-zone-names)
(zone-name->id)
(zone-id->name)
(zone-var)
(zone-type)
(zone-name)
(read-zone-id-list)
(read-zone-id)
(get-zone)
(zone-id)
cx-surface-menu        list
(ti-surface-projected-area)
(pick-surface-group)
(read-new-surface-id.name)
(iline-surface)
(irename-surface)
(idelete-surface)
(icell-surface)
(ipoint-surface)
(ipoint-array)
(iiso-clip)
(iiso-surface)
(izone-surface)
(ipartition-surface)
(isphere-slice)
(iplane-slice)
(irake-surface)
(isurface-cells)
(isurface-grid)
(gui-surface-projected-area)
(default-minimum-feature-size)
(gui-surface-integrals)
(ti-surface-vertex-average)
(ti-surface-facet-average)
(ti-surface-sum)
(ti-surface-massavg)
(ti-surface-mass-average)
(ti-surface-flow-rate)
(ti-surface-mass-flow-rate)
(ti-surf-vertex-max)
(ti-surf-vertex-min)
(ti-surf-facet-max)
(ti-surf-facet-min)
(ti-surf-min-max)
(ti-surface-integral)
(ti-surface-average)
(ti-surface-area)
(print-one)
(unit-label)
(unit-value)
(print-header)
(gui-fill-face-zone-values)
(cx-fill-face-zone-values)
(surface-vertex-max)
(surface-vertex-min)
(surface-vertex-average)
(surface-facet-max)
(surface-facet-min)
(surface-facet-average)
(surface-sum '(7) "velocity-magnitude")
```

```
(surface-massavg)
(surface-mass-average)
(surface-mass-flow-rate)
(surface-flow-rate)
(surface-integral)
(surface-average)
(surface-area '(7))
(surface-integrate)
(cx-delete-srf-ref-in-grp)
(surface-ids->surface-groups)
(surface-id->surface-group)
(cx-reset-surface-groups)
(cx-init-surface-groups)
(make-surface-groups)
(inquire-surface-line-names)
(inquire-surface-plane-names)
(inquire-surface-group-names-of-type)
(inquire-surface-group-names)
(grp->srf)
(cx-rename-srf-group)
(cx-get-group-srfs)
(srf-grp?)
(cx-delete-group)
(cx-add-new-srf-group)
(cx-ungroup)
(remove-from-grp-list)
(cx-group-grps)
(flatten-surface-groups)
(surface-values)
(surface-coordinates)
(make-pairs)
(pair-coords)
(cx-surface-face-list)
(cx-surface-uniq-node-coords)
(cx-surface-uniq-node-values)
(surface-velocity-vectors)
(cx-surface-coordinates)
(cx-surface-values)
(zone-coordinates)
(zone-values)
(apply-slice)
(surface-facets)
(rename-surface)
(temp-surface?)
(surface?)
(list-surfaces)
(surface-area-vectors)
(surface-name)
(get-surface)
(new-surface-id)
(get-mouse-point)
(fill-cx-tmp-array)
(gui-surfaces-creation-failure)
(gui-add-named-surface)
(gui-transform-surface)
(cx-rotate-3d)
(cx-scale-mat)
(gui-iso-clip)
(gui-iso-surface)
(gui-quadric-surface)
(gui-plane-surface)
(cx-show-plane-tool-normals)
(gui-line/rake-surface)
(cx-show-line-tool-normals)
(gui-point-surface)
(gui-partition-surface)
(gui-zone-surface)
(gui-manage-surfaces)
(cx-remove-tmp-geom)
(cx-display-tmp-surf)
(cx-surface-get-min-max)
(cx-surface-fill-temp)
(rem-quote-sym)
(cx-get-surf-def-attr)
(cx-set-surf-def-attr)
(cx-get-surf-def-attr-pos)
cx-surf-interp-attr-list      list
(cx-copy-surface)
*cx-surface-list-width*       25
(read-surface-id-list)
(read-surface-id)
(read-surface-list)
(read-surface)
(cx-get-surface-ids)
(surface-name/id?)
(valid-surf-name?)
(rake-surface)
(mrake-surface)
(line-surface)
(mline-surface)
(plane-surface)
(mplane-surface)
(quadric-surface)
(sphere-slice)
(ibounded-plane)
(iplane-surface-align)
(point-normal-surface)
(plane-slice)
(point-array)
(sphere-coeff)
(plane-coeff)
(iso-clip)
(iso-clip-new)
(partition-surface)
(sample-plane-points)
(planar-point-surface)
(point-surface)
(transform-surface)
(cell-surface)
(zone-surface)
(iso-surface)
(surface-append!)
(delete-surfaces)
(surface-grid)
(suspend-surfaces)
(free-surfaces)
(iso-srf-chk)
(cx-restart-fast-iso)
(cx-end-fast-iso)
(cx-start-fast-iso)
(cx-destroy-surface-all)
(cx-destroy-surface)
(cx-delete-zone-surface)
(surface-id->zone-id)
(zone-id->surface-id)
(cx-create-boundary-zone-surfaces)
(create-zone-surface)
(add-zone-surface-defs)
(client-inquire-fast-iso)
(cutting-plane-off)
(create-cutting-plane)
(cutting-plane-hook)
(cx-activate-plane-tool)
cxplane.provided       #t
(ti-custom-field-function/load)
(ti-custom-field-function/save)
(cf-code)
(pp-cfd)
(cx-get-obj-desc-attr)
(cx-get-obj-attr)
(cx-get-obj-id)
(cx-store-obj-all-attr)
(cx-store-obj-desc-attr)
(cx-store-obj-attr)
(cx-get-obj-by-attr)
(cx-get-obj)
(cx-delete-obj)
(cx-add-obj)
(cx-new-obj-id)
(cx-cf-name->id)
(cx-cf-id-attr)
(cx-inquire-cf-ids)
(cx-inquire-user-def-cf-names)
(cx-get-cf-desc-attr)
(cx-get-cf-attr)
(cx-set-cf-attr)
```

```
(cx-get-cf-by-attr)
(cx-rename-cf)
(cx-delete-cf)
(cx-get-cf)
(cx-add-cf)
(cx-new-cf-id)
(cx-initialize-cell-functions-client)
(cx-eval-cf)
(code-gen)
(d/dz)
(d/dy)
(d/dx)
(gui-manage-cf)
(custom-field-function/load)
(custom-field-function/save)
(custom-field-function/define)
(err-reduction-fail)
(match-all-productions)
(match-production)
(push-sym)
(pop-sym-till-less-prec)
(inc-parse)
(token-value)
(token-syn-cat)
(top-terminal)
(set-preceedence!)
(preceedence)
(sym-value)
(end-parser)
(init-parser)
parse-stack     list
(value-prod)
production-list         list
(shift-error-check)
(err-table-ref)
(parse-table-ref)
col-entries     list
parse-table     list
(parse)
(lex)
(init-number)
(init-lexer)
(init-parser/lexer)
prev-token      #f
prev-number?    #f
cur-number-str
(gui-user-def-cf)
(but-name->display)
(bin-op?)
(un-op?)
(trig-op?)
trig-ops        list
un-ops  list
bin-ops list
calc-display-map        list
(var-name->display)
calc-display    list
calc-inp-list   list
cf-str
*cx-cell-function-length*       75
*cx-max-cf-name*        25
(cx-field-rename)
(cx-field-eval)
(cx-field-define)
cxcf.provided   #t
cxsurf.provided         #t
(surf-inits!)
(pp-list)
(v3-interpolate)
(v3-unit)
(v3-magnitude)
(v3-cross)
(v3-dot)
(v3->z)
(v3->y)
(v3->x)
(dot)
(row-mat)
(mm-aux)
(aref)
(mat-mult)
(cx-identity-mat)
(cx-translate-mat)
(cx-rotate-x)
(cx-rotate-y)
(cx-rotate-z)
(max-list)
(min-list)
(transpose)
(transform)
(cx-get-trans-pts-min-max)
(cx-update-surface-attr)
(cx-update-all-surface-attr)
(list-union)
(cx-ancestor-surfaces-id)
(cx-ancestor-surfaces-id-list)
(cx-purge-surface-def-list)
(iso-func)
(cx-create-surface-from-def)
(cx-generate-susp-surface-defs)
(cx-add-surface-def)
(cx-get-def-coarse-surface)
(virt2real)
(vt2rl)
(real2virt)
(rl2vt)
(cx-delete-virtual-id)
(cx-get-virtual-index)
(cx-delete-map-entry)
(cx-list-surfaces)
(cx-surface-area-vectors)
(surface-id->name)
(surface-ids->names)
(surface-name->id)
(inquire-point-surface-names)
(inquire-surface-names)
(inquire-surface-ids)
(cx-suspend-all-surfaces)
(cx-update-surface)
(cx-create-surface)
(cx-rename-surface)
(cx-delete-surface)
(cx-add-surface)
(cx-new-temp-surface-index)
(set-next-surface-index)
(new-surface-index)
(cx-active-surface-ids)
(cx-store-surface-all-attr)
(cx-store-surface-desc-attr)
(cx-store-surface-attr)
(cx-get-surface-attr)
(cx-get-surface-desc-attr)
(surface-id)
(cx-active-surface)
(cx-set-surface)
(cx-get-surface)
(cx-set-surface-lists)
(set-surface-version)
(surface-version)
(cx-save-surface-lists)
(surf-set-list-size)
(surf-list-bnds-chk)
surfaces-groups         list
cx-max-surf-id-ref      21
cx-surface-list         #f
cx-temp-surfaces?       #f
cx-temp-surface-list    list
first-virtual-id        4196
*cx-max-surface-num*    4096
*cx-fast-iso-info*      #f
*cx-big-neg*    -1e+20
*cx-big-pos*    1e+20
surf.provided   #t
(set-cx-field-render-vars)
(fill-face-thread-values)
(fill-face-values)
```

```
(fill-cell-values)
(fill-node-values)
(inquire-cell-functions-sectioned)
(inquire-cell-functions)
(%client-inquire-cell-vector-functions)
(%client-inquire-cell-functions-sectioned)
(%client-inquire-cell-functions)
(client-support-symmetry?)
(display-grid-partition-boundary)
(client-draw-grid-partitions)
(display-grid-outline)
(display-grid)
(idraw-thread-grid)
(grid-internal)
(grid-outline)
(thread-grid)
(gui-animation-control)
(animation-name-list)
(cxg-ani-hardcopy-frames-cb)
(create-mpeg-animation)
(cx-set-hardcopy-options-for-mpeg)
(get-node-field-list-seq)
(get-node-field-list-win)
(show-node-field-list)
(cxg-copy-win-field-to-seq-field)
(cxg-restore-seq-node-field)
(cxg-update-active-window-node-field)
(cxg-save-node-field)
(cxg-ani-last-hardcopy-frame-list)
(cxg-ani-create-hardcopy-frames)
(cxg-ani-hardcopy-callback)
(cxg-ani-hardcopy-filename)
(cxg-ani-replay)
(cxg-update+xy-animation)
(cxg-ani-xy-plot)
(cxg-update+snap-animation)
(cxg-snap-animation)
(cxg-ani-check-path)
(cxg-ani-get-seq-basename)
(seqlist->winid)
(seqlist->frames)
(seqlist->storetype)
(seqlist->name)
(ti-fft-plot-file)
(ti-xy-plot-radial-band-averages)
(radial-band-average)
(ti-xy-plot-axial-band-averages)
(axial-band-average)
(band-average)
(scalar-bands)
(band-clip)
(ti-xy-plot)
(ti-xy-plot-zone/surface)
(ti-xy-set-surface-scale)
plot/set-menu   list
(ti-surface-plot)
(ti-zone-plot)
(ti-solution-plot)
(ti-xy-set-scale)
(ti-xy-plot-files)
(ti-xy-plot-file)
(xy-plot-zones+surfaces)
(cx-xy-plot-buffer-data)
(gui-xy-plot-ffts)
(gui-xy-plot-files)
(gui-xy-plot-zone/surface)
(cx-xy-plot-to-file)
(cx-xy-plot-to-port)
(cx-xy-plot-data)
(cx-solution-plot)
(cx-surface-plot)
(cx-zone-plot)
(surface-positions)
(gui-xy-plot-curves)
(ti-set-xy/scale/fmt)
(gui-xy-plot-axes)
(cx-xy-plot-files)
(xy-plot-file)
(xy-read-file)
(xy-read-port)
my-multi?       #f
(xy-read-columns)
(xy-read-curves)
(xy-read-particle)
(xy-build-particle-curves)
(xy-read-particle-header)
(xy-plot-list)
(end-plot)
(start-title-plot)
*cx-xy-multiple-files* #t
cxxy.provided   #t
cxganim.provided        #t
(iread-bc)
(iwrite-bc)
(downcase)
(process-thread)
(set-dv-variables)
(set-rp-variables)
(set-bc)
(list-bc)
(read-bc)
(write-bc)
(has-wild-card?)
(get-thread-list)
(gui-autosave-files)
(autosave-case-data)
(write-transient-ensight-explicit)
(write-transient-ensight-auto-append)
(write-transient-ensight-case)
(append-transient-export)
(write-transient-ensight-scalar)
(write-transient-ensight-vel)
(write-transient-ensight-geo)
(autosave-freq-proc)
(gui-write-export)
(ti-write-gambit)
(ti-write-fast)
(ti-write-es-transient)
(ti-write-es-gold-transient)
(ti-write-es-gold)
(ti-write-es)
(ti-write-fv-data)
(ti-write-fv)
(ti-write-avs)
(ti-write-fv-uns)
(ti-write-tecplot)
(write-radtherm)
(write-patran)
(ti-write-cgns)
(write-cgns)
(write-gambit)
(write-flpost)
(write-fast)
(write-engold-gui)
(write-engold-ascii)
(write-engold-binary)
(write-engold)
(write-es-gold-transient)
(write-es-transient)
(write-es-gold)
(write-es)
(write-fv-data)
(write-fv)
(write-fv-uns)
(ti-write-dx)
(write-dx)
(write-avs)
(write-tecplot)
(write-icepak-results)
(ti-write-icemcfd)
(write-icemcfd)
(ti-write-flux-profile)
(write-radiation-export)
(ti-write-radiation)
(enable-radtherm)
(ti-write-radtherm)
```

```
(ti-write-ansys)
(write-ansys)
(ti-write-ascii)
(write-ascii)
(ti-write-ideas)
(write-ideas)
(ti-write-abaqus)
(write-abaqus)
(ti-write-nastran)
(write-nastran)
(ti-write-patran-cell-temperature)
(ti-write-patran-nodal-results)
(ti-write-patran-neutral-file)
(write-patran-nodal-results)
(write-patran-result-template)
(write-patran-neutral-file)
(write-patran-cell-temperature)
*enable-radtherm-export?*      #t
(ti-write-fast-solution)
(ti-write-fast-scalar)
(ti-write-fast-velocity)
(ti-write-fast-grid)
(write-fast-solution)
(write-fast-scalar)
(write-fast-velocity)
(write-fast-grid)
*fast-binary-files?*   #f
(ti-write-mpgs-scalar)
(ti-write-mpgs-velocity)
(ti-write-mpgs-geometry)
(write-mpgs-gold-scalar)
(write-mpgs-scalar)
(write-mpgs-gold-velocity)
(write-mpgs-velocity)
(write-mpgs-gold-geometry)
(write-mpgs-geometry)
(write-export)
(ti-open-oned-library)
(ti-open-udf-library)
(ti-udf-compile)
(gui-udf-compile)
(open-isat-library)
(open-udf-library)
(load-udf-scm-file)
(udf-compile)
(compare-case/solver)
(gui-import-patran)
(ti-import-patran)
(import-patran)
(gui-import-nastran)
(ti-import-nastran)
(import-nastran)
(gui-import-ideas-universal)
(ti-import-ideas-universal)
(import-ideas-universal)
(gui-import-gambit)
(ti-import-gambit)
(import-gambit)
(gui-import-fidap)
(ti-import-fidap)
(import-fidap)
(gui-import-cgns)
(ti-import-cgns)
(import-cgns)
(gui-import-ansys)
(ti-import-ansys)
(import-ansys)
(gui-import-fluent4-case)
(ti-import-fluent4-case)
(import-fluent4-case)
(gui-import-metis-zone-case)
(gui-import-metis-case)
(ti-import-metis-zone-case)
(ti-import-metis-case)
(import-metis-zone-case)
(import-metis-case)
(client-write-data)
(client-read-data)
(client-append-data)
(client-write-case)
(client-read-case)
(client-read-zone)
(reading-case-file)
(force-metis-method-compatibility)
(rpsetvar-to-default)
(particle-history-open?)
(stop-particle-history)
start-particle-history-write    n/a
(start-particle-history)
(gui-start-particle-history)
(ti-start-particle-history)
(write-solar-pos)
(gui-write-sglobs)
(ti-write-sglobs)
(write-sglobs)
(ti-write-viewfac)
(write-viewfac)
(gui-read-sglobs-vf)
(ti-read-sglobs-vf)
(read-sglobs-vf)
(gui-read-sglobs)
(ti-read-sglobs)
(read-sglobs)
(gui-write-rays)
(ti-write-rays)
(write-rays)
(gui-read-rays)
(ti-read-rays)
(read-rays)
(gui-read-pdf)
(ti-read-pdf)
(read-pdf)
(gui-interp-data)
(ti-write-interp-data)
(ti-read-interp-data)
(interp-data)
(gui-import-chemkin)
(gui-import-cgns-data)
(ti-read-cgns-data)
(cgns-data-read)
(update-solver-threads)
(ti-write-fan-profile)
(write-fan-profile)
(gui-write-hosts)
(ti-write-hosts)
(write-hosts)
(gui-read-hosts)
(ti-read-hosts)
(read-hosts)
(read-isat-table)
(ti-read-isat-table)
(gui-read-isat-table)
(write-isat-table)
(ti-write-isat-table)
(gui-write-isat-table)
(gui-write-surface-globs)
(ti-write-surface-globs)
(write-surface-globs)
(gui-write-boundary-grid)
(ti-write-boundary-grid)
(write-boundary-grid)
(gui-reread-grid)
(ti-reread-grid)
(reread-grid)
(gui-read-sample)
(read-sample)
(gui-write-existing-profile)
(ti-write-existing-profile)
(write-existing-profile)
(gui-read-profile)
(ti-read-transient-table)
(ti-read-profile)
(read-transient-table)
(read-profile)
export-file-types       n/a
 (write-transient-ensight)
```

```
pload   #f
loc     #f
filename         n/a
frequency-entry          n/a
htc-flux?       #t
htc-wall?       #f
htc-wall        #f
(ensight-frequency)
transient-on     n/a
transient?       n/a
htc      n/a
loads    n/a
transient        n/a
location         n/a
delimiter        n/a
surfaces         n/a
binary? #t
node     n/a
space    n/a
cell-centered    n/a
comma    n/a
(check-data)
(check-grid)
cl-file-package         list
(suffix-expand-filename)
(gui-import-data)
(ti-import-data)
(gui-import-case)
(ti-import-case)
(%import-by-filter)
(gui-write-case-data)
(ti-write-case-data)
(write-case-data)
(read-zone-grid-data)
(ti-read-zone-grid-data)
(gui-read-case-data)
(ti-read-case-data)
(read-case-data)
(gui-write-data)
(ti-write-data)
(write-data)
(append-data)
(gui-read-data)
(ti-read-data)
(read-data)
(gui-write-case)
(ti-write-case)
(write-case)
(read-zone)
(ti-read-zone)
(gui-read-case)
(ti-read-case)
(read-case)
(client-case-data-pattern)
(client-case-pattern)
(ti-set-file-format)
(client-show-configuration)
*client-run-time-release*      3
(ok-to-discard-data?)
(ok-to-discard-case?)
(canonicalize-filename)
(strip-version)
(string-suffix-ci?)
(client-check-grid)
(client-check-case)
clrelease.provided      #t
*client-library-run-time-release*     3
clfiles.provided        #t
(gui-open-udf-library)
(ti-compile-now)
(bc-summary)
(domainvars-rpvars-compat)
(initialize-domain-menus)
(ti-set-type-domain-vars)
(read-domain)
(gui-domains-manage)
(clear-and-hide-panel)
(download-domains)
(get-all-domain-vars)
(create-case-domains)
(create-case-domains-of-type)
(default-domain-name)
(%get-domain-by-name)
(%get-domain-by-id)
(get-domain)
(domain?)
(inquire-domain-names)
(domain-type)
(domain-id)
(domain-name)
(domains-compat)
(set-sfc-values)
mphase-zone-vars-compat        list
(get-phase-threads-with-id)
(next-domain-id)
(delete-phase-domain)
(add-phase-domain)
(update-domains)
(domains-changed)
(get-domains-manage-panel)
(get-all-domains)
(get-interaction-domains)
(get-phase-domains)
(get-geom-domains)
(get-domains-of-type)
reorder-domains          n/a
(free-domains)
(delete-domain)
(create-domain)
(free-virtual-cells)
(rehide-skewed-cells)
(ti-unhide-skewed-cells)
(ti-hide-skewed-cells)
(draw-shell-junction)
(draw-shell)
(free-shells)
(create-case-shells)
si-menu  list
(ti-make-periodic-interface)
(ti-draw-zones)
(ti-list-sliding-interfaces)
(list-sliding-interface)
(ti-delete-all-si)
(delete-all-sliding-interfaces)
(ti-delete-si)
(ti-create-si)
(si-thread?)
(gui-grid-interfaces)
(inquire-si-threads)
(sliding-interface-thread?)
(interface-in-use?)
(update-sliding-interface-write-case)
(case-has-si-face-periodics?)
(delete-sliding-interface)
(recreate-sliding-interfaces)
(delete-sliding-interface-case-threads)
(create-sliding-interface)
(update-sliding-interfaces)
(update-si)
(create-sliding-interface-threads)
(create-si-threads)
(build-sliding-interface-grids)
(build-si-grids)
(build-si-compatibility)
(gui-user-fan-model)
(ti-user-fan-model)
(register-user-fan-monitor)
(user-fan-monitor)
(update-user-fans)
(user-fan-command)
(user-fan-input-name)
(user-fan-output-name)
(list-flow-init-defaults)
(thread-name-default)
(inquire-grid-threads)
(grid-check)
```

```
ti-target-mfr-menu     list
threads-package        list
(move-from-nosolve-threads)
(move-to-nosolve-threads)
(delete-all-threads-of-phase)
(get-nosolve-phase-threads)
(get-nosolve-face-cell-threads)
(get-phase-threads)
(get-all-threads)
(get-face-threads)
(get-cell-threads)
(reorder-threads)
(free-threads)
(delete-single-thread)
(delete-thread)
(%create-thread)
(copy-thread)
(create-thread)
(create-thread-for-domain)
(gui-update-wall-thread-list)
(gui-threads-copy)
(ti-copy-bc)
(copy-thread-bc)
(set-thread-vars)
(get-thread-vars)
(gui-threads-manage)
cell-type-menu  list
external-type-menu     list
internal-type-menu     list
periodic-type-menu     list
(pick-thread-type)
(initialize-thread-menus)
(initialize-thread-lists)
(get-all-thread-types)
(get-cell-types)
(get-external-types)
(get-internal-types)
(get-periodic-types)
(thread-type-ignore?)
(thread-type-index->name)
(thread-type-name->rename)
(%thread-type-name->index)
(thread-type-name->index)
thread-type-list       list
(replace-zone)
(ti-replace-zone)
(ti-activate-cell-threads)
(ti-deactivate-cell-threads)
(update-scheme-threads)
(gui-activate-cell-threads)
(gui-deactivate-cell-threads)
(deactivate-cell-thread)
(ti-delete-cell-threads)
(gui-delete-cell-threads)
(activate-cell-thread)
(delete-cell-thread)
(ti-extrude-face-thread-parametric)
(ti-extrude-face-thread-delta)
(extrude-thread)
(gui-merge-threads)
(ti-merge-threads)
(merge-like-threads)
(merge-threads)
(gui-separate-cell-thread)
(ti-separate-cell-thread-by-region)
(ti-separate-cell-thread-by-mark)
(gui-separate-face-thread)
(ti-separate-face-thread-by-region)
(ti-separate-face-thread-by-angle)
(ti-separate-face-thread-by-face)
(ti-separate-face-thread-by-mark)
(ti-slit-periodic)
(slit-periodic)
(ti-repair-periodic)
(ti-create-periodic)
(create-periodic)
(gui-fuse-threads)
(ti-fuse-threads)
(fuse-threads)
(ti-orient-face-thread)
(orient-face-thread)
(sew-all-two-sided-walls)
(sew-two-sided-wall)
(slit-all-two-sided-walls)
(slit-two-sided-wall)
(ti-slit-face-thread)
slit-face-thread        n/a
(thread-prop)
(thread-materials)
(default-material-name)
(read-thread-names)
(read-cell-thread-id-list)
(read-boundary-thread-id-list)
(read-thread-id-list)
(read-thread-id)
(read-cell-thread-list)
(read-boundary-thread-list)
(read-wall-thread-list)
(read-boundary-thread)
(read-thread-list)
(read-thread)
(ti-set-type-thread-reference-values)
(ti-set-type-thread-flow-inits)
(ti-set-type-thread-vars)
(set-thread-type!)
(ti-set-thread-type)
(ti-per-pg-bc)
(ti-per-mfr-bc)
(read-flow-dir)
fl-dir-z         n/a
fl-dir-y         n/a
fl-dir-x         n/a
(ti-set-thread-name)
(get-fluid-thread-material)
(set-fluid-thread-material)
(thread-type-name->object)
(inquire-thread-names)
(thread-name->id)
(thread-id->name)
(thread-var)
(thread-kind)
(thread-type)
(thread-name)
(thread-domain-id)
(thread-id)
(cell-thread?)
(non-periodic-boundary-thread?)
(boundary-thread?)
(wall-thread?)
(get-all-thread-vars)
(list-threads)
(thread?)
(get-boundary-threads)
(get-threads-of-type)
(get-phase-thread)
(%get-thread-by-name)
(%get-thread-by-id)
(get-thread)
(cleanup-case-surfaces)
(create-case-threads)
(mp-vars-compat)
(update-thread-materials)
common-phase-bc-types? #t
(sort!)
(sort)
qsort.provided  #t
(rampant-repl)
(rampant-initialize)
(exit-rampant)
(shutdown-lam)
(check-lam-mpi-tasks)
list-compute-nodes-process-count       n/a
wipe-compute-node?      n/a
(isetvar)
(client-ti-set-var)
(ti-set-var)
```

```
(client-ti-set-window-var)
(ti-set-window-var)
(inquire-plot-info)
(update-physical-time)
(physical-time-steps)
(unsteady-iterate-hook)
(init-flow)
(init-s2s)
(init-dtrm)
(ti-patch)
(ti-iterate)
(iterate)
(dynamic-mesh-suspend-surfaces)
(set-config)
(dpm-cache?)
(rp-thread?)
(rp-host?)
(rp-graphics?)
(rp-double?)
(rp-3d?)
(solar?)
(sg-vfr?)
(sg-uds?)
(sg-udm?)
(sg-swirl?)
(sg-soot?)
(sg-s2s?)
(sg-rsm?)
(sg-rosseland?)
(sg-pull?)
(sg-premixed?)
(sg-pollut?)
(sg-pdf-transport?)
(sg-pdf?)
(sg-par-premix?)
(sg-p1?)
(sg-mphase?)
(sg-melt?)
(sg-network?)
(sg-dynmesh?)
(sg-dtrm?)
(sg-dpm?)
(sg-crev?)
(sg-bee-gees?)
(sg-disco?)
(sg-cylindrical?)
(rp-v2f?)
(rp-visc?)
(rp-absorbing-media?)
(rp-turb?)
(rp-trb-scl?)
(rp-spe-surf?)
(rp-spe-site?)
(rp-spe-part?)
(rp-spe?)
(rp-sge?)
(rp-seg?)
(rp-sa-des?)
(rp-sa?)
(rp-react?)
(rp-net?)
(rp-lsf?)
(rp-les?)
(rp-lam?)
(rp-kw?)
(rp-ke?)
(rp-inviscid?)
(rp-hvac?)
(rf-energy?)
(rp-amg?)
(rp-dual-time?)
(rp-unsteady?)
(rp-dpm-cache?)
(rp-axi?)
(rp-atm?)
(rp-acoustics?)
(update-cx-client-information)
(update-bcs)
(bcs-changed)
(models-changed)
(inquire-version-command)
(inquire-version)
(rf-cache-config)
(inquire-config)
(client-monitor-freq-proc)
(rp-2d?)
(object-parent)
(object?)
(environment-parent)
(environment?)
system-global-syntax-table    list
(syntax-table-define)
object.provided        #t
(update-menubar)
(client-update-menubar)
(gui-reset-symbol-list-items)
(gui-menu-insert-subitem!)
(gui-menu-insert-item!)
(gui-not-yet-implemented)
clgui.provided  #t
(client-set-var)
(client-get-var)
(client-var-define)
(domainsetvar)
(domaingetvar)
(rpsetvar)
(rpgetvar)
(inquire-all-versions)
(gui-rampant-run)
(choose-version)
(cx-gui-rsf)
(inquire-option)
communicator    #f
(rampant-file-version)
(rampant-run)
(fluent)
(cx-transform-turbo-surf)
(cx-create-turbo-surf)
(cx-set-turbo-axis)
(%cx-set-average-direction)
(cx-surface-write-values)
(%surface-integrate)
(%iso-zone)
(%iso-surface)
(%planar-point-surface)
(%point-surface)
(%cx-is-arc-surface)
(cx-exchange-node-values)
(%cx-surface-fill-temp)
(%cx-surface-get-min-max)
(cx-activate-fast-iso)
(cx-suspend-fast-iso)
(%cx-end-fast-iso)
(%cx-start-fast-iso)
(probe-surface-info)
(track-surface-on-zone)
(track-surface)
(%display-surface-elements)
(%iso-clip)
(%transform-surface)
(%cell-surface)
(%zone-surface)
(%surface-append!)
(%delete-surface)
(%free-surfaces)
(%domain-var-value-set!)
(%domain-var-value)
(%rp-var-value-set!)
(%rp-var-value)
(%rpgetvar)
(rp-var-clear-cache)
(inquire-sub-threads)
(%separate-skewed-cells)
(%unhide-cells)
(%hide-cells)
(%fast-io-transfer-dumps)
```

```
(%write-network-history)
(%network-temperature)
(%init-crevice-memory)
(%free-crevice-memory)
(%initialize-sound-array)
(%process-and-plot)
(%finish-fft-process)
(%write-sound-pressure)
(%set-adjacent-cell-zone)
(%extract-acoustics-signals)
(%compute-acoustics-sound-pressure)
(%set-acoustic-read-threads)
(%initialize-acoustics-file-for-read)
(%set-acoustics-receivers)
(%read-acoustic-next-timestep)
(%write-acoustic-data)
(%which-domain)
(%domain-super-domain)
(%domain-sub-domain)
(%set-domain-variables)
(%free-phase-domain)
(%create-phase-domain)
(test-migrate-shell)
(%delete-shell)
(%draw-shell-junction)
(%report-shell-status)
(%free-shell)
(%create-shell)
(%create-all-shells)
(%check-coupled-thread)
(hanging-or-sliding-mesh?)
(host-domain-filled?)
(solver-cpu-time)
(report-connectivity)
(delete-hxg)
(over-ride-zone)
(get-hxc-info)
(set-effectiveness-table)
(set-hxc-enthalpy)
(get-hxc-dimension)
(%draw-macro)
(init-hxg-model)
(%allocate-hxg-memory)
(%free-all-hxgs)
(phase/total-volume)
(initialize-unsteady-statistics)
(%init-disco)
(%stop-particle-history)
(%start-particle-history)
(pathline-print-summary)
(dpm-print-cache-report)
(%dpm-suggest-nthreads)
(%dpm-get-nthreads)
(%dpm-set-nthreads)
(%open-isat-library)
(%dpm-particle-summary-all)
(%dpm-particle-summary)
(dpm-clear-all-particles)
(dpm-inject-particles)
(dpm-print-summary)
(dpm-get-summary)
(dpm-inquire-summary-names-sectioned)
(dpm-inquire-summary-names)
(dpm-inquire-particle-functions-sectioned)
(dpm-inquire-particle-functions)
(dpm-inquire-particle-types)
(dpm-list-injections)
(dpm-get-min-max-units)
(dpm-compute-pathlines)
(dpm-flush-sources)
(%dpm-free-injections)
(%dpm-delete-injection)
(%dpm-set-injection)
(dpm-parameters-changed)
(who-am-i)
(%contact-monitor)
(%isat-table-size)
(%kill-isat-table)
(%write-isat-table)
(%read-isat-table)
(clear-pdf-particles)
(init-pdf-particles)
(%free-pdf-memory)
(inquire-pdf-species)
(%pdf-init)
(pdf-type)
(%read-pdf)
(%write-pdf)
(%delete-all-sgroups)
(%group-s-globs)
(%write-surface-globs)
(%compute-solar-pos)
(s2s-glob-ok?)
(s2s-set-glob-ok)
(s2s-globs-done?)
(s2s-set-globs-done)
(%read-sglobs-vf)
(%read-sglobs)
(%sglob-info)
(update-storage-before-s2s)
(%read-s2s-cells)
(%group-s2s-cells)
(%delete-all-s2s-groups)
(dtrm-rays-ok?)
(dtrm-set-rays-ok)
(dtrm-globs-done?)
(dtrm-set-globs-done)
(%read-rays)
(%ray-trace)
(update-storage-before-dtrm)
(%delete-all-groups)
(%group-cells)
(display-globs)
(%insert-rays)
(%display-curr-display-glob)
(%change-curr-display-glob)
(execute-udf-eval)
(%repartition-remeshing-zones)
(%remesh-local-prism-faces)
(%print-remesh-cell-marks)
(%draw-remesh-cell-marks)
(%mark-remesh-cells)
(%remesh-local-cells)
(%print-forces-moments)
(modify-lift)
(compute-lift)
(inquire-motion)
(%find-cell-at-location)
(%prismatic-layer)
(%insert-cell-layer)
(remesh-cell-region)
(subdivide-mesh-layer)
(%refine-mesh-by-mark)
(%contour-node-displacement)
(%get-max-skewness-on-zone)
(%get-min-max-volume-on-zone)
(%check-dynamic-mesh)
(%update-dynamic-mesh)
(%update-dynamic-threads)
(%free-dynamic-mesh)
(%delete-dynamic-thread)
(%create-dynamic-thread)
(%download-dynamic-threads)
(%init-dynamic-mesh)
(%create-dynamesh-node-grids)
(%inquire-si-parents)
(%inquire-si-mperiodics)
(%list-sliding-interfaces)
(%free-sliding-interfaces)
(%clear-sliding-interfaces)
(%delete-sliding-interface)
(%update-sliding-interfaces)
(%update-sliding-interface)
(%create-sliding-interface)
(%clean-up-sliding-interfaces)
(%sliding-mesh?)
```

```
(%non-conformal-mesh?)
(%non-conformal-count)
(set-relaxation-method)
(%invalidate-storage)
(set-residual-history-size)
(%open-udrg-library)
(%open-rgas-library)
(%open-udf-library)
(%user-function-list)
(%chip-exec)
(%chip-link)
(%execute-at-end)
(%udf-on-demand)
(%chip-listing)
(%chip-compile-file)
(%draw-oned-cell)
(%advance-oned-solution)
(%update-oned-bcs)
(%create-oned-udfs)
(%start-oned-library)
(%close-oned-library)
(%open-oned-library)
(%initialize-storage)
(%sample-min-max)
(%sample-bins)
(%delete-sample)
(%sample-list)
(%read-sample-file)
(%get-zone-heatflux)
(%report-zones-torque)
(%get-zone-torque)
(%initialize-nr-bcs)
(%display-profile-points)
(%write-fan-profile)
(%profile-list)
(%delete-profile)
(%create-oriented-profile)
(%any-thread-has-profile?)
(%update-mixing-plane-profile)
(%create-mixing-plane-profile)
(%update-dynamic-profiles)
(%free-profiles)
(%write-cgns-export)
(%write-profile-section)
(%read-transient-table)
(%read-profile-file)
(%read-profile-section)
(%inquire-interp-field-names)
(%interpolate-cell-thread-data)
(%write-interp-data)
(%filter-data)
(%update-storage-phase-walls)
(%properties-need-update)
(%property-methods)
(%list-reactions)
(%free-materials)
(%delete-material)
(%create-material)
(%set-material!)
(inquire-timers)
(print-flow-timer)
(print-data-timer)
(print-case-timer)
(end-write-data-timer)
(start-write-data-timer)
(end-write-case-timer)
(start-write-case-timer)
(end-read-data-timer)
(start-read-data-timer)
(end-read-case-timer)
(start-read-case-timer)
(clear-flow-timer)
(clear-data-timer)
(clear-case-timer)
(print-timer-history)
(clear-timer-history)
(init-timer-history)
(clear-domain-timers)
(%write-hosts-file)
(%read-hosts-file)
(%list-hosts)
(%delete-host)
(%delete-all-hosts)
(%add-host)
(get-parallel-communicator)
(check-compute-nodes-unique?)
(%list-compute-nodes-process-count)
(%list-compute-nodes)
(update-after-migration)
(%internet-dot-address)
(prf-update-all-rpvars)
(%prf-spawn)
(d-prf-set-var)
(prf-set-var)
(prf-set-partition-mask)
(%set-check-partition-mismatch)
(%prf-flush-message)
(prf-exit)
(prf-command-finished)
(fill-any-storage-allocated)
(%allocate-parallel-io-buffers)
(%query-parallel-io-buffers)
(%limit-parallel-io-buffer-size)
(%free-parallel-io-buffers)
(%resolve-duplicate-hanging-nodes)
(%dpm-full-locate-particles)
(%dpm-node-locate-particles)
(%dpm-host-locate-particles)
(%dpm-node-to-host-particles)
(%dpm-host-to-node-particles)
(%dpm-host-to-node-source)
(%node-to-host-solution)
(%free-host-domain)
(%fill-host-domain)
(%dpm-fill-host-domain)
(%create-neighborhood)
(kill-zero-cell-compute-nodes)
(kill-compute-node)
(%get-process-ids)
(show-virtual-machine)
(print-time-stamps-to-file)
(print-global-timer-offsets)
(calculate-global-timer-offset)
(mp-clear-send-recv-time-stamps)
(mp-clear-timer)
(mp-wall-time)
(mp-get-tcp-chunk-size)
(mp-set-tcp-chunk-size)
(mp-set-comm-timer)
(%mp-set-exchange-size)
(%mp-set-socket-size)
(mp-debug)
(mp-trace)
(mp-allow-suspend)
(mp-set-suspend)
(mp-set-generic-gop)
(%mp-set-time-out)
(prf-print-vars)
(compute-node-count)
(rp-host-id)
(rp-host-set-var-cache)
(rp-host-function)
(probe-all-marked-cell-info)
(probe-marked-cell-info)
(probe-thread-info)
(upload-thread-vars)
(download-thread-vars)
(%write-data)
(%write-surface-grid)
(%write-case)
(%read-sectioned-file)
(initialize-domain-vars)
(domain-var-default)
(domain-var-units)
(domain-var-value-define)
(domain-var-value-set!)
```

```
(domain-var-value)
(domain-var-define)
(domain-var-object)
(rp-var-default)
(rp-var-units)
(rp-var-value-define)
(rp-var-value-set!)
(rp-var-value)
(rp-var-define)
(rp-var-object)
(inquire-release)
(%residual-history)
(thread-normal)
(volume-integrals)
(domain-thread-integrals)
(inquire-grids)
(debug-cell-and-face)
(print-model-timers)
(%grid-debug)
(have-read-partitioned-case?)
(%grid-check-duplicate-nodes)
(%grid-check-partition-encapsulation)
(%grid-check)
(%convert-data-to-absolute)
(%convert-data-to-relative)
(%reset-thread-element-type)
(%translate-grid)
(%scale-grid)
(reset-residuals)
(%repair-translational-periodic)
(%repair-rotational-periodic)
(%merge-periodic)
(%split-periodic)
(%inquire-equations)
(%inquire-patch-variable-names)
(patch)
(get-thread-derived-variables)
(set-thread-variables)
(%set-thread-type)
(thread-empty?)
(thread-exists?)
(update-before-data-write)
(update-after-data-read)
(update-before-data-read)
(%build-grid)
(%init-grid)
(misc-mem-stats)
(mem-stats)
(display-thread-sv-data)
(display-thread-sv)
(display-memory-on-thread)
(display-memory)
(dump-memory)
(display-infinite-storage)
(display-constant-storage)
(display-untouched-storage)
(delete-untouched-storage)
(reorder-stores)
(display-storage)
(minimize-storage)
(display-tuples)
(garbage-collect-tuples)
(fluent-exec-name)
(fluent-arch)
(%clear-domain)
(inquire-adjacent-threads)
(%reset-inquire-all-adjacent-threads)
(%inquire-all-adjacent-threads)
(inquire-nosolve-face-threads)
(inquire-network-face-threads)
(inquire-network-cell-threads)
(inquire-face-threads)
(inquire-cell-threads)
(inquire-mesh-type)
(%inquire-cell-vector-functions)
(%inquire-cell-functions-sectioned)
(%inquire-cell-functions)
(solver-statistics)
(solver-residuals)
(%repair-face-handedness)
(%repair-face-to-cell-threads)
(%advance-particles)
(%iterate-time-step)
(%reset-vof-solution)
(%update-vof-solution)
(%update-physical-time)
(%iterate)
(flow-init)
(fill-face-threads)
(%get-min-max-info)
(%cell-only-field?)
(%fill-face-thread-values)
(%fill-cell-values)
(%fill-node-values)
(thread-extents)
(domain-extents)
(%reset-node-list)
(%set-machine-valid)
(%set-grid-valid)
(%set-data-valid)
(%set-case-valid)
(data-modified?)
(case-modified?)
(grid-modified?)
(machine-valid?)
(data-valid?)
(case-valid?)
(grid-valid?)
(%save-data-id)
(%save-case-id)
(update-case-id)
(print-bandwidth)
(%reorder-threads)
(%reorder-domain)
(reorder-cells-by-position)
(%remove-orphan-bridge-faces)
(print-partitions)
(print-domain)
(%merge-like-threads)
(%free-changed-id-list)
(update-thread-ids)
(%scan-sectioned-file)
(%read-sectioned-zone)
(%finish-read-zone)
(%create-phase-threads)
(%get-changed-ids)
(%init-read-zone)
(%create-face-and-shadow-pair)
(%activate-cell-thread)
(%deactivate-cell-thread)
(%delete-cell-thread)
(%merge-threads)
(%fuse-threads)
(%extrude-face-thread)
(%separate-cell-thread-by-region)
(%separate-cell-thread-by-mark)
(%separate-face-thread-by-region)
(%separate-face-thread-by-angle)
(%separate-face-thread-by-face)
(%separate-face-thread-by-mark)
(%sew-two-sided-wall)
(%sew-all-two-sided-walls)
(%slit-two-sided-wall)
(%slit-all-two-sided-walls)
(%slit-face-thread)
(%orient-face-thread)
(%smooth-partition)
(%reorder-partitions)
(%merge-partition-clusters)
(%copy-active-partition-to-stored)
(%combine-partition)
(%inquire-must-pretest-partition-functions)
(%inquire-pretest-partition-functions)
(%inquire-partition-picks)
(%inquire-partition-functions)
(%repartition-by-zone)
```

```
(set-partition-in-marked-region)
(partition-count)
(partition)
(%auto-partition)
(%auto-encapsulate-si)
(encapsulate-si)
(set-case-buffer-size)
(get-case-buffer-size)
(print-neighborhood)
(sync-solution)
(%migrate-partitions)
(%print-migration-statistics)
(%copy-dest-part-to-stored)
(%migrate-cells-before-slide)
(%migrate-cells-after-slide)
(%migrate-cells)
(%relabel-entities)
(%balance-before-kill-node)
(%calculate-balanced-partitions)
(%migrate-shell-back)
(%valid-partition-id?)
(generate-fpe)
(cell-bins)
(%swap-mesh-faces)
(%smooth-mesh)
(%compute-sizing-function-defaults)
(%free-sizing-functions)
(%draw-bridge-nodes)
(%fill-bridge-nodes)
(%free-parents)
(%make-hanging-interface)
(%refine-mesh-hang)
(%adapt-mesh-hang)
(%refine-mesh)
(%adapt-mesh)
(%mark-percent-of-ncells)
(%mark-inside-cylinder)
(%mark-inside-sphere)
(%mark-inside-hex)
(%mark-inside-iso-range)
(%mark-boundary-cells-per-thread)
(%mark-with-yplus-per-thread)
(%mark-with-refine-level)
(%mark-with-volume-change)
(%mark-with-volume)
(%mark-with-boundary-volume)
(%reset-boundary-proximity)
(%mark-with-gradients)
(%not-bit-in-marked-cells)
(%xor-bits-in-marked-cells)
(%or-bits-in-marked-cells)
(%and-bits-in-marked-cells)
(%copy-pair-bits-in-marked-cells)
(%copy-bits-in-marked-cells)
(%toggle-bit-in-marked-cells)
(%set-bit-in-marked-cells)
(%clear-pair-bits-in-marked-cells)
(%clear-bit-in-marked-cells)
(%limit-marked-cells)
(%count-marked-cells)
(%clear-marked-cells)
(%clear-pair-bits-in-all-marked-cells)
(%clear-bit-in-all-marked-cells)
(%clear-all-marked-cells)
(%clear-cell-function-names)
(%models-changed)
(materials-require-model-change?)
(%rp-config)
(update-node-flags)
(init-node-flags)
(%delete-turbo-topology)
(%define-turbo-topology)
(xy-plot-turbo-avg)
(calc-turbo-avg)
(display-turbo-avg)
(display-path-cone)
(display-domain-labels)
(display-node-flags)
(display-marked-cells)
(mouse-split)
(fill-cx-array)
(contour-surface)
(vector-function-surface)
(velocity-vector-surface)
(%thread-grid)
(grid-partition-boundary)
(draw-symmetries)
(%inquire-periodic-transform)
(draw-periodics)
(%partition-surface)
(%apply-slice)
(%read-cgns-data)
(update-cell-function-lists)
client-monitor-solution-done   #f
*remain-timestep/iter* list
*time/iteration*       0
(set-monitor-transient)
(monitor-transient-solution)
(get-monitor-frequency)
(show-solution-monitors)
(clear-solution-monitors)
(cancel-solution-monitor)
(register-solution-monitor)
(monitor-solution-done)
(monitor-solution-message)
(monitor-solution)
(client-set-version-info)
(client-exit)
(upload-zone-vars)
(download-zone-vars)
(upload-cx-vars)
(download-cx-vars)
cx-windows/hardcopy-menu       list
cx-windows/hardcopy/driver-menu       list
cx-windows/hardcopy/ps-format-menu    list
cx-windows/hardcopy/color-menu        list
cx-windows-menu        list
cx-windows/xy-menu     list
cx-windows/video-menu  list
cx-windows/text-menu   list
cx-windows/scale-menu  list
cx-windows/main-menu   list
cx-windows/axes-menu   list
(ti-set-window-pixel-size)
(ti-set-window-aspect)
(ti-hardcopy-window)
(ti-set-window)
(ti-close-window)
(ti-open-window)
(cx-display-ppm)
(cx-display-image)
(cx-gui-hardcopy-options)
(cx-gui-hardcopy)
(cx-hardcopy-suffix)
(cx-hardcopy-file-filter)
(cx-panelfig)
(cx-graphicsfig)
(cx-set-small-window)
(cx-preview-hardcopy)
(cx-set-window-size)
(cx-set-window)
(cx-close-window)
(cx-display-geom-window)
(cx-open-window)
(cx-use-new-window?)
(client-valid-window-owner?)
(client-assign-window-owner)
cxwindows.provided     #t
(ti-button-functions)
(read-function)
(ti-display-label)
(open-segments)
(delete)
*cx-render/cell/max*    0
*cx-render/cell/min*    0
*cx-render/node/max*    0
```

```
*cx-render/node/min*    0
*cx-render/units*       #f
*cx-render/domain*
*cx-render/name*
(cx-save-layout)
(cx-save-case-state)
(cx-set-case-defaults)
(cx-inquire-colors)
(cx-inquire-marker-symbols-nt)
(cx-inquire-marker-symbols)
(cx-inquire-line-patterns)
(cx-gui-button-functions)
(cx-add-probe-function)
(cx-update-probe-function)
(cx-current-probe-function-name)
cx-button-functions     list
cx-probe-functions      list
cx-null-probe-name      off
(handle-selection)
(popup-button)
(resize-other-windows)
(zoom-3d-window)
(cx-add-button-to-toolbar)
(cx-refresh-toolbar)
(handle-key)
*cx-key-map*    list
client-update-title-hook        #f
(cx-update-button-functions)
(right-button-function)
(middle-button-function)
(left-button-function)
(orbit-axes)
(print-marked-cell-info)
(cx-show-probe-hooks)
(cx-remove-probe-hook)
(cx-install-probe-hook)
(cx-set-default-probe-hook)
(cx-mouse-probe)
(cx-mouse-orbit)
(cx-mouse-dolly)
(cx-remove-text)
(cx-display-label)
(cx-mouse-annotate)
(cx-interrupt-mouse-point)
(cx-get-mouse-point)
(cx-highlight-selection)
(cx-single-send)
(cx-sendq)
(cx-set-io-busy-cursor)
(cx-get-kill-notification-method)
(cx-set-kill-notification-method!)
(cx-is-process-running?)
(cx-get-current-process)
(cx-set-current-process)
(cx-command-port)
(cx-trace)
(cx-kill-current-process)
(cx-kill)
(cx-send)
(cx-client-run)
(cx-listen)
(cx-run)
*cx-timeout*    300
*cx-trace*      #f
(cx-interrupt-client)
(cx-interrupt)
(menu-repl)
*cx-startup-file*       #f
*main-menu*     list
(cx-repl)
(%checkpoint)
(cx-install-checkpoint-hook)
(%emergency)
(cx-install-emergency-hook)
(cx-exit)
kill-script-filename    /home/mirko/kill-
fluent11300
(exit)
(cx-dialog-done)
(cx-multiple-file-dialog)
(cx-file-dialog)
(chdir)
(cx-select-dialog)
(cx-prompt-dialog)
(cx-working-dialog)
(cx-yes-no-dialog)
(cx-ok-cancel-dialog)
(cx-info-dialog)
(cx-warning-dialog)
(cx-error-dialog)
client-support-field-derivatives        #f
(client-free-host-domain)
(client-host-domain-filled?)
(client-host?)
(client-max-partition-id)
(client-set-cx-vars)
(client-get-coarse-surfaces)
(client-set-coarse-surfaces)
client-support-coarse-surfaces?         #f
(client-delete-surface)
(client-create-point-surface)
(client-activate-injection-surfaces)
(client-display-dpm-pathlines)
(client-compute-dpm-pathlines)
(client-vector-function-surface)
(client-relative-vector-in-surface-plane)
(client-relative-vector-surface)
(client-vector-in-surface-plane)
(client-vector-surface)
(client-contour-surface)
(client-draw-grid-zones)
(client-draw-grid-outline)
(client-draw-grid)
(client-draw-symmetries)
(client-set-symmetry)
(client-selected-symmetry-planes)
(client-all-symmetry-planes)
(client-inquire-periodic)
(domain-name->id)
(client-inquire-default-domain-name)
(client-inquire-domain-ids-and-names)
(client-has-multiple-domains?)
(client-support-relative-vectors?)
(client-draw-grid-partitions?)
(client-support-grid-partitions?)
(client-support-grid-levels?)
(client-add-monitor-command)
(client-set-current-dataset)
(client-inquire-current-dataset)
(client-inquire-datasets)
(client-support-multiple-data?)
(client-copy-node-values-to-temp)
(client-node-suff-temp?)
(client-fill-face-zones)
(client-cell-only-field?)
client-cell-only-fields         list
(client-fill-face-zone-values)
(client-fill-cell-values)
(client-support-cell-values?)
(client-fill-node-values)
(client-set-node-values)
(client-inquire-node-values)
(inquire-section-domain-list-for-cell-
functions)
(inquire-domain-for-cell-vector-functions)
(inquire-domain-for-cell-functions-
sectioned)
(inquire-domain-for-cell-functions)
(client-inquire-cell-vector-functions)
(client-inquire-cell-functions-sectioned)
(client-inquire-cell-functions)
(client-zone-name->id)
(client-zone-id->name)
(client-inquire-zones-of-type)
(client-inquire-zone-types)
(client-inquire-zone-name)
```

```
(client-inquire-zone-names)
(client-inquire-boundary-zones)
(client-inquire-interior-zones)
(client-inquire-zones)
(client-inquire-bc-name)
(client-inquire-axis)
(client-inquire-domain-extents)
(client-unsteady?)
(client-solver-sm?)
(client-inquire-reference-depth)
(client-solver-axi?)
(client-check-data)
(client-set-time-step)
(client-inquire-time-step)
(client-inquire-iteration)
(client-inquire-title)
(client-inquire-release)
(client-inquire-version)
(client-inquire-name)
*date/time-format*     %b %d, %Y
(cx-initialize)
(cx-dot-pathname)
*cx-multithread*       #f
(cx-data-3d?)
(cx-mesh-3d?)
(cx-3d?)
(cortex?)
(write-journal-file-part)
(read-journal-file-part)
(read-journal-file-commands)
(fnmatch)
(ti-read-scheme)
(ti-stop-transcript)
(ti-start-transcript)
(ti-macro-load)
(ti-macro-save)
(ti-execute-macro)
(ti-stop-macro)
(ti-start-macro)
(ti-read-journal)
(ti-stop-journal)
(ti-start-journal)
(cx-file-type)
*cx-filetypes*  list
(remove)
(suffix)
(basename)
(strip-directory)
(directory)
(temp-file)
(cx-pause)
(cx-set-pause-time)
(cx-set-delay-time)
*cx-pause-time*        -1
(gui-read-scheme)
(stop-transcript)
(transcript-open?)
(stop-journal)
(journal-open?)
(gui-start-transcript)
(cx-stop-transcript)
(cx-start-transcript)
(cx-transcript-open?)
(cx-macro-load)
(cx-macro-save)
(cx-macro-define)
(gui-execute-macro)
(gui-start-macro)
(cx-list-macros)
*cx-macros*     list
(cx-executing-macro?)
(cx-execute-macro)
(cx-stop-macro)
(cx-start-macro)
(cx-macro-open?)
(cx-gui-batch-options)
(gui-read-journal)
(gui-start-journal)
(client-exit-on-error)
(cx-reading-journal?)
(cx-read-journal)
(cx-stop-journal)
(cx-start-journal)
(cx-journal-open?)
(cx-save-recent-files)
(cx-update-recent-files)
(cx-add-recent-file)
(cx-enable-recent-files)
*cx-recent-files-limit*        4
(cx-write-file)
(cx-read-file-with-suffix)
(cx-read-file)
(compress-filename)
(uncompress-filename)
*uncompress*    #f
*compress*      #f
(append-file)
(write-file)
(read-file-with-suffix)
(read-file)
(read-file-with-suffix-and-leave-port-open)
(read-file-and-leave-port-open)
(%append-file)
(%write-file)
(%read-file-and-leave-port-open)
(%read-file)
(remote-file-pattern-exists?)
(find-remote-file-with-suffix)
(find-remote-file-pattern-with-suffix)
(remote-file-exists?)
(client-default-basename)
(client-inquire-binary-files)
(client-set-binary-files)
(client-support-binary-files?)
(quote-if-needed)
(ok-to-overwrite-remote?)
(ok-to-overwrite?)
(ti-set-batch-options)
(ti-exit-on-error)
(ti-set-answer-prompt)
*cx-answer-prompt?*    #f
(ti-set-overwrite-prompt)
*cx-overwrite-prompt?* #t
(syncdir)
(dump-scheme)
*cx-execute-macros-quietly?*   #t
*cx-exit-on-error*     #f
journal-file-commands  list
cxrelease.provided     #t
*cortex-run-time-release*      3
cxfiles.provided       #t
display/set/rendering-options  list
(display/set/rendering-options/hsm-menu)
display/set/pathlines-menu     list
display/set/contours   list
display/set/vectors    list
(display/set/color-ramp-menu)
display/set/colors-menu        list
(ti-set-edge-visibility)
(pick-hsm-method)
(pick-pathline-style)
(pick-color-ramp-from-list)
(cx-gui-annotate)
(scene-max-index)
(scene-list-text-objs)
(text-name->segment-key)
(text-name->scene-index)
(parse-string)
*cx-pfa-fonts?*        #t
*cx-font-sizes*        list
*cx-font-names*        list
(cx-gui-display-options)
cx-hsm-methods  list
(cx-set-graphics-driver)
(cx-show-graphics-drivers)
(describe-graphics)
```

```
light-interp-menu      list
(ti-toggle-headlight)
(ti-set-ambient-color)
(ti-set-light)
cx-lights-menu  list
(cx-gui-lights)
(id->symbol)
(light->xyz)
(light->rgb)
(light->on?)
(cx-set-light)
*cx-light-symbol-inactive*     @
*cx-light-symbol-active*       (x)
*cx-light-segment*     #f
*cx-max-light*  9
cxlights.provided      #t
cxdisplay.provided     #t
cx-view-menu    list
cx-camera-menu  list
(cx-gui-views)
(cx-gui-camera)
(cx-gui-define-mirror)
(cx-draw-mirrors)
(cx-gui-define-periodic)
(cx-gui-write-views)
(cx-write-views)
(non-standard-views)
(cx-read-views)
(cx-set-camera-relative)
(cx-compute-default-views)
*max-stack-size*       20
(cx-pop-view)
(cx-push-view)
(new-view-name)
(cx-save-view)
(cx-restore-view)
(cx-get-views)
(cx-delete-view)
(cx-add-view)
(cx-default-view)
(extent->zmax)
(extent->zmin)
(extent->ymax)
(extent->ymin)
(extent->xmax)
(extent->xmin)
(zoom-camera)
(roll-camera)
(pan-camera)
(orbit-camera)
(dolly-camera)
(camera-up-vector)
(camera-target)
(camera-projection)
(camera-position)
(camera-field)
(cx-show-camera-projection)
(cx-show-camera-field)
(cx-show-camera-up-vector)
(cx-show-camera-target)
(cx-show-camera-position)
(camera->projection)
(camera->height)
(camera->width)
(camera->up-vector)
(camera->target)
(camera->position)
(view->transform)
(view->camera)
(view->name)
cxview.provided        #t
(cx-cmap-editor)
(incr-cmap-name-count)
(cx-set-color-ramp-range)
(new-cmap-name)
(cx-show-cmap-names)
(cx-set-color-map)
(cx-get-cmap)
(cx-add-cmap)
(cx-gui-cmap-editor)
(ti-vector)
(ti-display-custom-vector)
(ti-add-custom-vector)
(default-vector-name)
(ti-velocity-vector)
(ti-profile)
(ti-contour)
(pick-cell-vector-function)
(pick-cell-function-domain)
(pick-cell-function)
(ti-re-scale-generic)
(ti-render-generic)
(ti-zone-grid)
ti-current-vector-domain       #f
ti-current-domain      #f
(profile-options)
(render/grid-cb)
(dpm-graphics-setup)
(gui-set-grid-rendering-options!)
*sweep/domain*  #f
*sweep/sub-function*   #f
*sweep/function*       #f
*sweep/vector-domain*  #f
*sweep/vector-name*    velocity
(gui-display-sweep-surface)
(gui-display-vectors)
(gui-display-contours)
(gui-display-grid-colors)
(gui-display-grid)
(cx-gui-vector-options)
(gui-custom-vectors)
(cx-inquire-user-def-vector-names)
(customvec->z-comp)
(customvec->y-comp)
(customvec->x-comp)
(customvec<-name)
(customvec->name)
(custom-vector-function/define)
(cx-rename-vector)
(cx-delete-vector)
(cx-get-vector)
(cx-add-vector)
(gui-manage-plot)
(gui-update-unsectioned-cell-function-
lists)
(gui-update-domains-for-vector-functions)
(gui-update-domain-from-section)
(gui-update-domain-lists)
(gui-update-cell-function-lists)
(cx-check-toggle-buttons)
(cx-cell-only-function?)
(cx-cell-only-field?)
(custom-gui-vector-function-label->name)
(custom-gui-vector-function-label)
(custom-vector-function-label->name)
(custom-velocity-vector)
(set-display-custom-vv)
(display-vector-function)
(styled-format)
(velocity-vector)
(set-display-vector-function)
(set-display-vv)
(profile)
(rgcb-profile)
(contour)
(rgcb-contour)
(set-profile-attr)
(set-contour-attr)
(set-cont-prof-attr)
(cx-update-range-vars)
(restore-segment-state)
(cx-add-to-vv-list)
(cx-restore-render-surfaces)
(cx-save-render-surfaces)
(render-grid)
(cx-draw-grid-zone)
```

```
(cx-show-open-segments)
(render-surface)
(cx-reset-grid-colors)
(cx-zone-color)
(show-thunk-list)
(show-title-list)
(get-rg-title)
(get-rg-thunk)
(add-thunk-to-list)
(add-title-to-list)
(render-generic)
(re-render-generic)
custom-vector-list     list
(start-title-frame)
(start-standard-title-frame)
(cx-init-right-frame-titles)
(cx-list-delete-entry)
(type-name->id)
(display-name)
(cx-gui-preselect-contour-functions)
(cx-set-viz-lists)
(cx-get-viz-lists)
(cx-reset-viz-lists)
(cx-write-viz-lists)
(cx-read-viz-lists)
(cx-gen-viz-name)
(cx-get-viz-all-names)
(cx-get-viz-attr)
(cx-add-viz-attr)
(cx-add-to-viz-list)
cxvlist.provided       #t
(ti-path-lines)
(render-path-lines)
(dpm-path-lines)
(path-lines)
(cx-reset-path-vars)
(get-path-min-max-units)
(pick-path-cell-function)
(inquire-path-cell-functions)
(num-surf)
(max-surf-id)
(total-facets)
(gui-display-particle-tracks)
(gui-display-path-lines)
*twist/max*      n/a
*twist/min*      n/a
*cx-viz/name*
cxpath.provided        #t
cxrender.provided      #t
cxcmap_ed.provided     #t
cxalias.provided       #t
(cx-set-plot-window-id)
*max-plot-window-id*   11
(cx-activate-tab)
(cx-create-tab)
(cx-create-frame-tabbed)
(gui-update-cell-function-widget)
(gui-update-cell-sub-function-widget)
(gui-update-cell-domain-widget)
(gui-update-cell-vector-function-widget)
(gui-fill-cell-values-sectioned)
(gui-fill-node-values-sectioned)
(gui-fill-cell-values)
(gui-fill-node-values)
(name->gui-function-labels)
(gui-vector-function-label->name)
(gui-function-labels->names)
(gui-function-domain-list)
(gui-function-label->name)
(gui-domain-label)
(gui-vector-function-label)
(gui-function-label)
(function-name->labels)
(vector-function-label->name)
(function-label->name)
(string-downcase)
(gui-get-selected-surface-ids)
(gui-get-selected-zone-ids)
(gui-pick-single-list-item)
(gui-unpick-list-items)
(gui-pick-list-items)
(gui-update-changed-list)
*gui-name-list-width*  #f
(cx-add-separator)
(cx-add-form)
(cx-show-symbol-list-selections)
(cx-rename-symbol-list-items)
(gui-toggle-symbol-list-selections)
(gui-delete-symbol-list-selections)
(gui-add-symbol-list-selections)
(cx-set-symbol-list-selections)
(cx-set-symbol-list-items)
(cx-delete-symbol-list-items)
(gui-add-selected-symbol-list-items)
(cx-add-symbol-list-items)
(cx-add-drop-down-symbol-list)
(cx-add-symbol-list)
(cx-create-drop-down-symbol-list)
(cx-create-symbol-list)
(cx-create-pattern-selector)
(gui-add-group-zone-list)
(gui-get-group-list-widget)
(gui-get-zone-list-widget)
(gui-add-group-zone-widgets)
(gui-update-zone-list)
(gui-init-zone-list)
(gui-add-zone-list)
(client-inquire-group-names)
client-groups   list
(cx-rename-list-items)
(cx-panel-designer)
(cx-add-real-entry)
(cx-create-profile)
(cx-create-draw-area)
(cx-create-list-tree)
(cx-create-dial)
(cx-create-scale)
(cx-create-drop-down-list)
(cx-create-list)
(cx-create-real-entry)
(cx-create-integer-entry)
(cx-create-text-entry)
(cx-create-toggle-button)
(cx-create-button)
(cx-create-text)
(cx-create-button-box)
(cx-create-table)
(cx-create-frame)
(cx-add-check-buttons)
(cx-add-radio-buttons)
(cx-show-check-button)
(cx-set-check-button)
(cx-add-toggle-button)
(cx-add-radio-button-box)
(cx-add-check-button-box)
(cx-hide-panel)
(cx-create-hoops-panel)
(cx-create-panel)
(cx-hide-item)
(cx-get-item-id)
(cx-get-menu-id)
(cx-update-menubar)
(cx-delete-item)
(cx-clear-menubar)
(cx-add-menu)
(cx-add-hitem)
(cx-add-item)
*cx-sort-surface-lists*        #t
*cx-panel-apply-close* #f
cxgui.provided  #t
render-specific-vars   list
(cxisetvar)
(cx-set-state)
(cx-show-state)
color-list      list
(cxgetvar)
```

```
(cxsetvar)
(cx-set-var-val-list)
(cx-get-var/value-list)
(cx-get-name-matching-varlist)
(cx-download-vars)
(cx-upload-vars)
(cx-show-var-stack)
(cx-pop-vars)
(cx-push-vars)
(cx-get-active-varlist)
(cx-get-env-varlist)
(cx-show-envstack)
(cx-show-varenv-unfiltered)
(cx-varenv-status)
(cx-show-varenv)
(cx-close-varenv)
(cx-open-varenv)
(cx-var-default-value)
(cx-var-value)
(cx-var-value-set!)
(cx-var-define)
(cx-var-object)
(cx-var-initialize)
cx-variables    list
(set-unit)
(cx-show-units)
(to-si-units)
(to-user-units)
(read-list-with-units-prompt)
(read-with-units-prompt)
(read-list-with-units)
(read-with-units)
(write-with-units)
(cx-lookup-units)
(ti-set-unit)
(cx-set-unit)
*cx-unit-table*        list
(cx-inquire-unit-systems)
(cx-set-unit-system)
*cx-conversion-table*  list
units.provided  #t
cxvar.provided  #t
(cx-update-pending?)
(cx-changed)
(cx-show-dependents)
(cx-delete-dependents)
(cx-add-dependent)
(rainbow-ramp)
(fea-ramp)
(red-ramp)
(green-ramp)
(blue-ramp)
(gray-ramp)
(cyan-yellow-ramp)
(blue-purple-ramp)
(purple-magenta-ramp)
(bgrb-ramp)
(rgb-ramp)
(bgr-ramp)
(interpolate-color-ramp)
cmap.provided   #t
(summary-table-end)
(summary-table-hline)
(summary-table-row)
(summary-table-begin)
(summary-line)
(summary-current-level)
(summary-section)
(summary-init)
(summary-title)
(cx-tui-complex-profile-string)
(cx-tui-complex-profile)
(cx-gui-complex-profile)
(cx-register-profile-method)
(cx-tui-profile-string)
(cx-tui-profile)
(ti-menu-load-string)
(ti-menu-load)
(ti-menu-load-port)
(ti-menu-error)
(alias)
alias-table     list
(read-generic-in-range-prompt)
(read-real-in-range)
(read-integer-in-range)
(read-object-generic-list)
(read-object-generic)
(read-object-id/name-list)
(read-object-id/name)
(read-string-from-list)
(read-symbol-from-list)
(read-symbol-list)
(read-symbol)
(yes-or-no?)
(y-or-n?)
(read-string/symbol-list)
(read-string-list)
(read-boolean-list)
(read-real-list)
(read-integer-list)
(read-filename)
(read-symbol/string)
(read-string/symbol)
(read-string)
(read-real)
(read-integer)
(ti-read-unquoted-string)
(read-generic-list-prompt)
(read-generic-prompt)
(read-real-pair-list)
(real-pair?)
(read-generic-list-pair)
(read-generic-list)
(read-generic)
(readq-generic)
(insert-menu-item!)
(ti-menu-insert-item!)
(ti-menu-item->help)
(ti-menu-item->thunk)
(ti-menu-item->name)
(ti-menu-item->test)
*ti-menu-load-delay*   1
*menu-prompt*
*menu-print-always*    #f
(cx-check-journal)
(cx-gui-do)
(ti-text-processing?)
(menu-get)
(menu-do-1)
(menu-do)
(ti-info)
(ti-menu-print)
(ti-read-default?)
(ti-input-pending?)
(ti-strip-blanks)
(ti-flush-input)
_       0
(string->valid-symbol)
(flush-white-space)
(flush-char-set)
(read-delimited-string)
char-set:newline       list
char-set:whitespace    list
(char-set)
journal-file-count     0
iface.provided  #t
cx.provided     #t
*cx-disclaimer*        WARNING        This
is a prototype version that has not yet
been tested and validated. Fluent Inc.
makes no commitment to resolve defects
reported against this prototype version.
However, your feedback will help us improve
the overall quality of the product.
client.provided        #t
```