web page: http://w3.pppl.gov/~zakharov

At the ground level of integrated modeling[1]

Leonid E. Zakharov

Princeton Plasma Physics Laboratory, MS-27 P.O. Box 451, Princeton NJ 08543-0451

Peter Yushmanov

Abeam technology, CA

Grigori Pereverzev

IPP Garching, Germany

Symposium

for the Future of Integrated Modeling

July 19, 2005, PPPL, Princeton, NJ

[1] This work was partially supported by US DoE contract No. DE–AC020–76–CHO–3073.

Abstract

For any numerical code the number of possible runs can be comprehended only if expressed at the logarithmic scale in the form of entropy. What really limits the entropy is the intellect of the author and his understanding of the problem rather than "`if()`" statements inside the code. The computer languages, perfect in handling the "`if()`"-like statements, do not provide the adequate mechanism for expressing the author's understanding, thus, leaving the entropy of the code unlimited for the unexperienced user.

This problem of interfacing, superficial from the point of view of single code developer, becomes a real issue at the level of integrated modeling dealing with the totally dissimilar things like communications, interfaces for the code control, distribution of the results and input data, on-line help, documentations, visualization, reports, etc. Even the single code gradually grows up to the limit of comprehension by its author, and then they both stay forever as the "master-slave" pair.

Thanks to the involvement and contributions of Walter Sadowski and Arnold Kritz, the problem became visible even to the fusion community. It could be addressed only if the very basic level of integration is formalized in an "entropy-free" manner using the computer ("entropy-free") power for its implementation.

The talk discusses the lowest level of integrated modeling and our experience in developing tools for ITER and other tokamak simulations.

Contents

1 Integrations in programming and in use of numerical codes.

Computers, perfectly predictable, are among few amazing things with zero entropy

People, who control computers, unavoidably generate a “mess” in two areas:

- *in programming and code development, and*
- *in using numerical codes.*

In both areas, formalized structuring and integration are the key for controlling the mess.

1. Computer languages provide powerful means for code development:
   (a) *Clustering the lines of the code into reusable routines.*
   (b) *Extracting routines from the codes into library files for sharing*
   (c) *Clustering data into structures and* $\{\ldots\}$ *blocks* (`C`)*.*
   (d) *Combining data structures and pointers to routines into classes (e.g.,* `C++`*).*
   (e) *Extracting classes from the codes to the level of OS-independent environment and integrating them with its capabilities (e.g.,* `java`*).*
   (f) *. . . of other integrations (WEB of D.Knuts, FWEB of J.Krommes)*

... still leaving room for undetectable (by the computer) bugs

The use of codes is prone to much easier mess creation than the code development

2. *It deals with "unexperienced" users, communications, code control, and is not yet addressed with the same level of success.*

   Following examples allow to introduce formalized intellectual information into the data source.

   (a) *XML became recently widely acceptable*

   ```
   XML was designed to carry data.
   .....
   XML was designed to describe data and to focus on what data is.
   .......
   XML does not DO anything
   ........
   XML was not designed to DO anything.
   ........
   ```

   (b) *Triggered by Walter Sadowski, CodeBuilder (Cb) became functional in 1997.*

   CodeBuilder $\equiv$ *(structure recognition)* $+$ *(drivers for servicing virtual structures)*

   In structuring, it uses 2-D names, i.e. `<type><Name>`, *with irreducible set of 3 types, while names are non-functional.*

   Cb deals with even more basic level of integration than XML

It is the task of people to introduce the structural information they have in mind

Then, the computer task is subdivided into two: (a) structure recognition, and (b) servicing it.

At the level of structuring Cb provides:

- *generality of the approach (three types of Cb structures are always present),*
- *use of both real (hardwired) and virtual structures (comment lines),*
- *freedom in using drivers for structural elements,*
- *non-intrusiveness*
- *multiple use of the same structure and automatic consistency, e.g, of*
    1. *communication system,*
    2. *system of On-Line Help,*
    3. *I/O file system or visualization system,*
    4. *compact and unambiguous GUI, etc,*

    — the tasks impossible in practice without computer assistance.

The following simple demonstration illustrates the control of `emacs`-editor by Cb

Virtual structures inside the code can be used for computer assisted integration

At the functional level of drivers the Cb simplifies

- use of existing tools (rather than “reinventing the wheel”), and
- creation of new drivers for specific purposes

Three steps generate a Cb-code

```
wrk> zcb Src/esc.c           makes the Code Control file

wrk> zcb esc.Cb              displays the Control Panel
                             generates templates for
                             Communication Control files
                             and OnLine Help

wrk> zcb esc.Cb -m Src/esc.Src     makes final command Mkesc
```

After this, the only necessary command for the code development is

```
wrk> Mkesc;Cbesc
```

The On-Line Help files are created and updated by

```
wrk> Hbesc
```

All code related structures can be updated while preserving consistency

2 Toward Real Time Forecasting (RTF) the ITER discharges

RTF$\equiv$ a transport simulation code running faster than the discharge

RTF targets the near term predictions (in fraction of τ_E) during the tokamak discharge.

- *RTF does not exist yet, but seems to be possible*
- *It would extend EFIT/rtEFIT experience to more "intelligent" prediction and control of a tokamak regime*
- *It would include a situation-sensitive adaptation of transport models with real time adjustment by the discharge data flow*
- *In its turn, RTF can serve as a generator of new signals for feed-back to the regime.*

RTF is the integration of transport and MHD with speed considered as the first priority

As integration problem RTF is more challenging than the integration of physics models into transport simulations. It should make them useful for the discharge control.

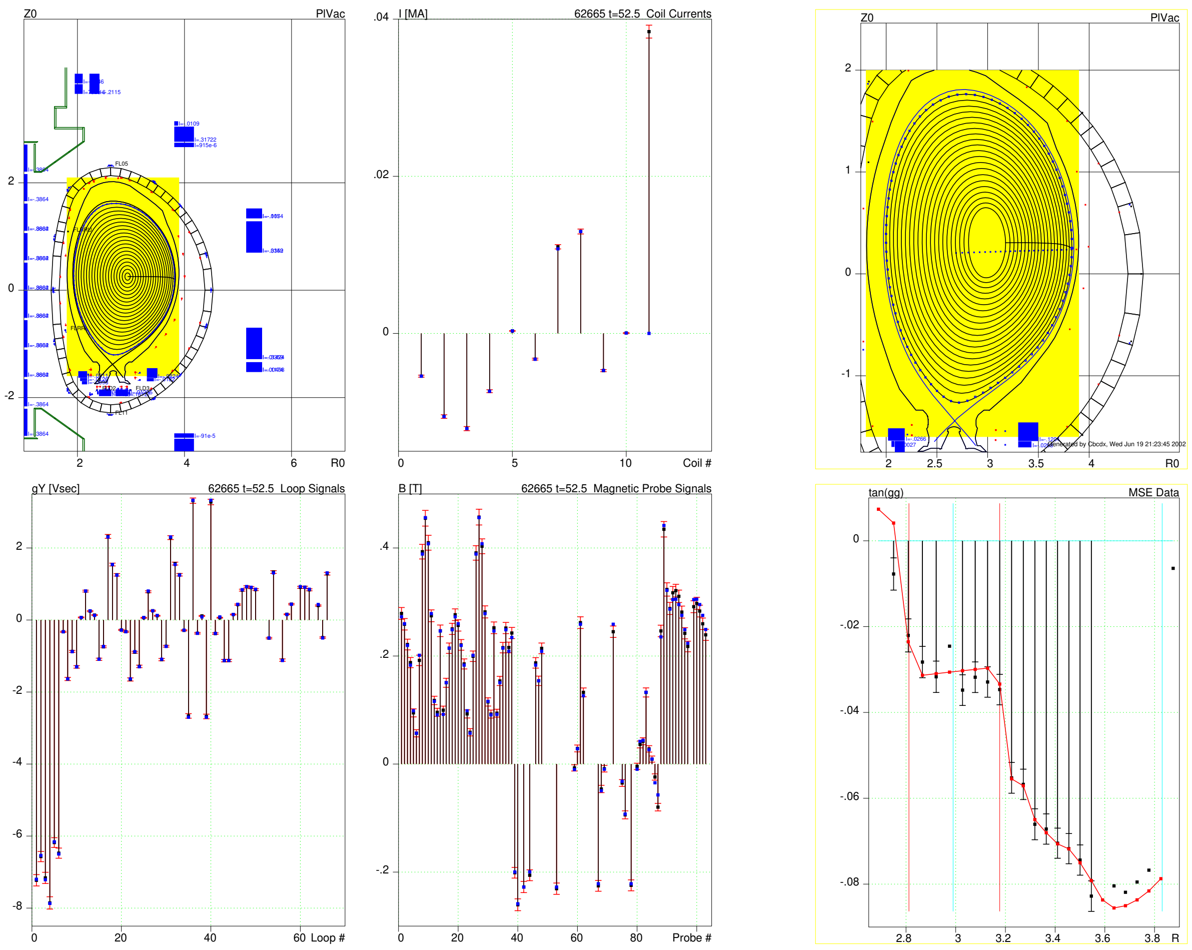
ESC-ASTRA-DCON-BALLON code system is an attempt to approach RTF

Referred later on as a RTF prototype, this system

1. *uses fastest (so far) equilibrium algorithms, but*
2. *still standard (and slow for stiff models) transport calculation algorithms,*
3. *uses Cb-drivers (`zcb, zhb, zdb`) for integrating its components*
   (a) *in conventional (subroutine, libraries) manner as well*
   (b) *as parallel processes on OS, and*
   (c) *as "code talking"*

The system is open for insertion/deletion and insertable to others structures of codes

All capabilities of EFIT are present in ESC in addition to extensive fixed boundary ones



ESC can reconstruct both conventional and hollow current equilibria

ESC-ASTRA became the first pair of integrated reconstruction + transport codes

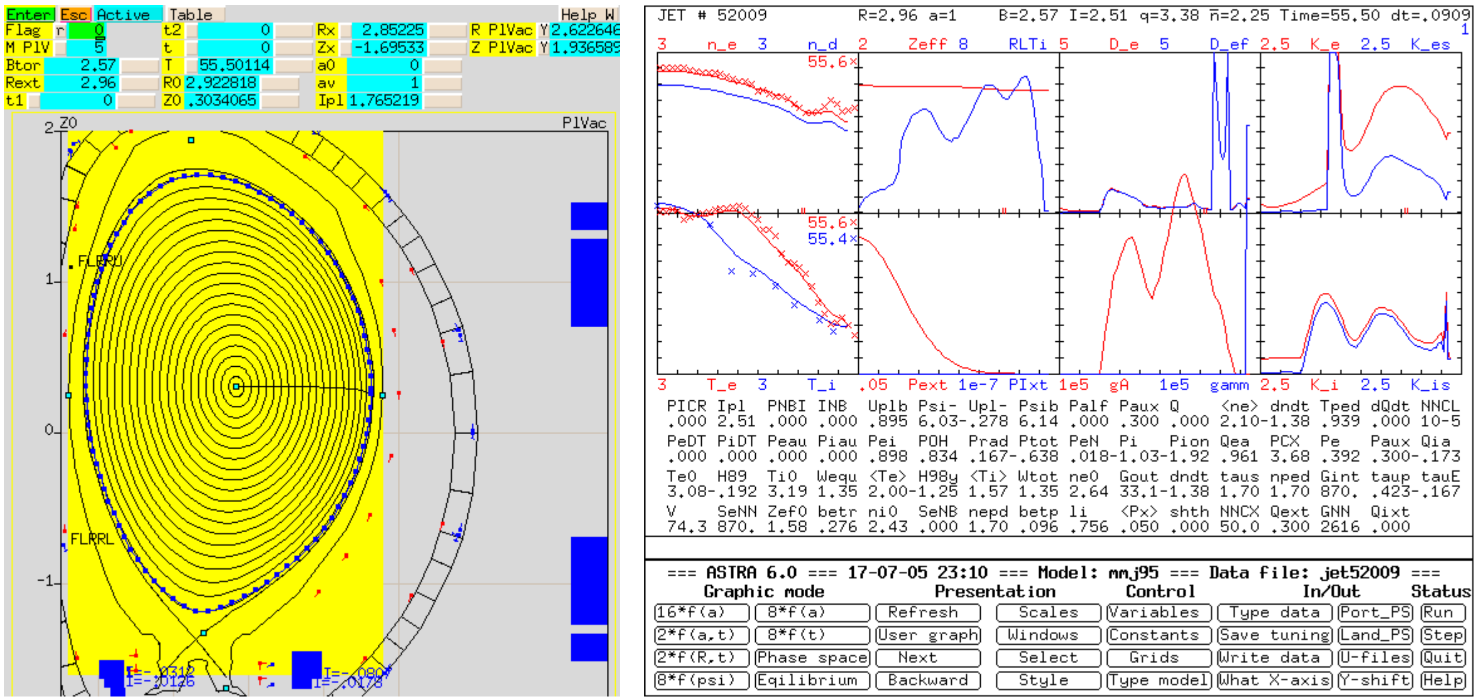
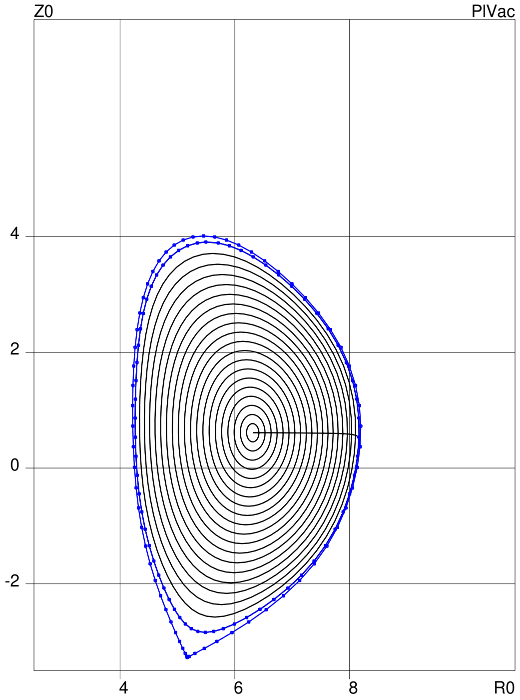


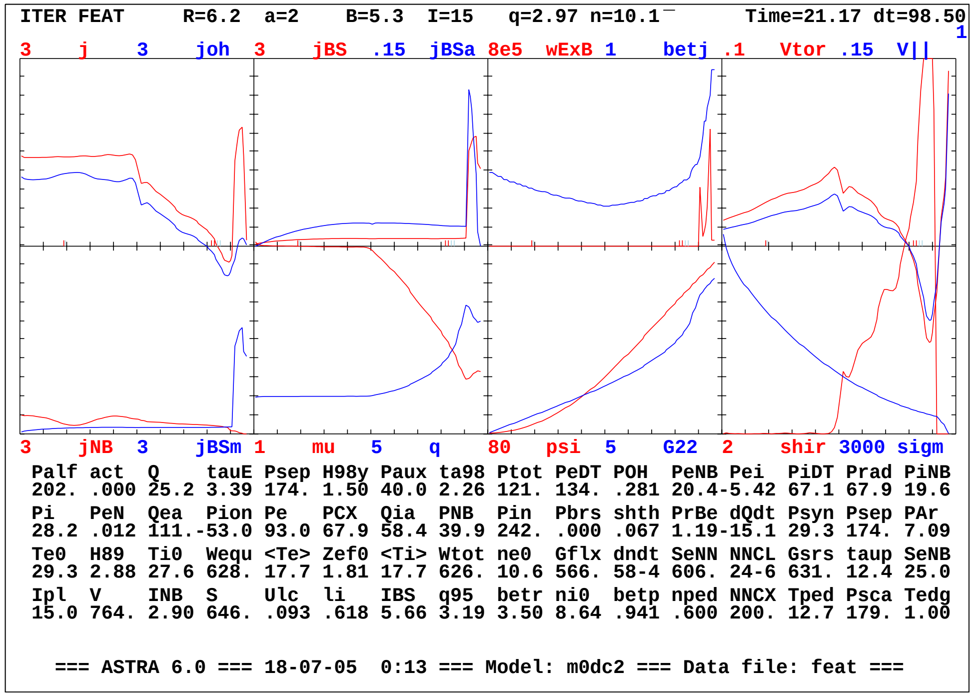
Illustration of JET simulation with multi mode model

Transport simulations may resolve uncertainties in equilibrium reconstruction what is important for stability tests.

On-the-fly ballooning and low-n stability tests are available

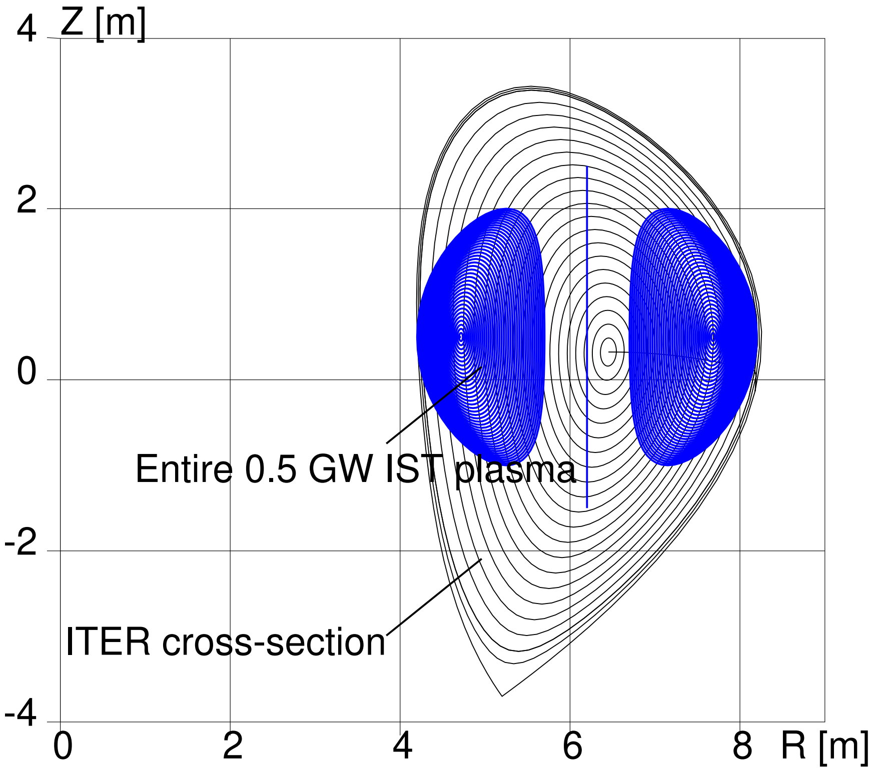
Special Fourier representation is used in ESC for separatrix limited configurations





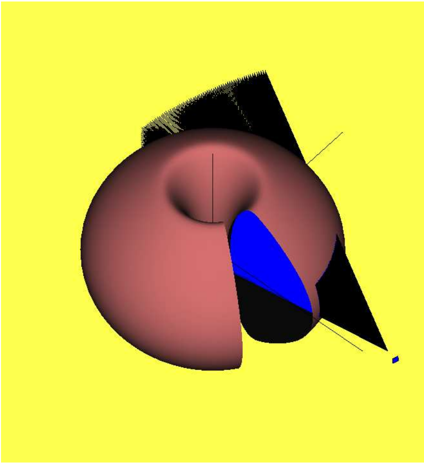
ASTRA made first steps toward compatibility with separatrix configurations

IST are compact ($\simeq 30$ m^3) high $\beta \simeq 40$ % devices implementing the LiWall concept

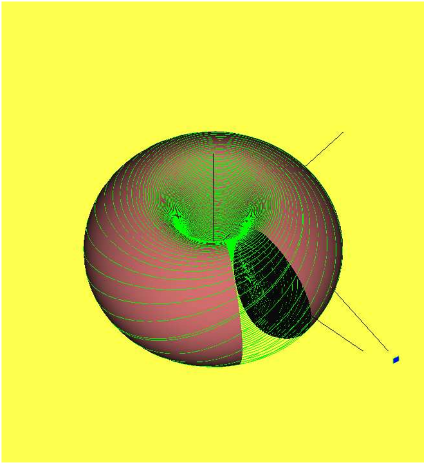


IST was developed as a concept for the reactor development complimentory to ITER

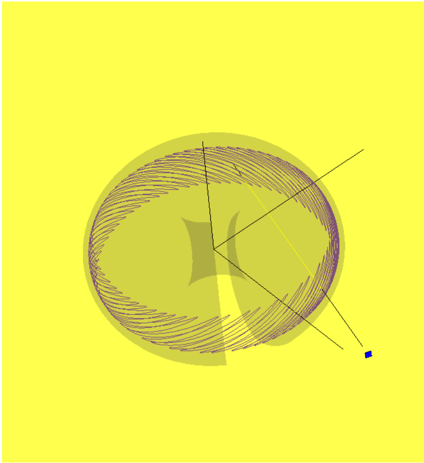
The real time 3-D OpenGL graphics is implemented into Cb for fast code development



Handling sightlines



Field line integration



Particle gyro-orbits

All components are available for particle analysis, e.g., for making a NBI module

3 Theory exercise on managing multi-parameter systems

At basic level people think in terms of nested structures with three types present.

Examples like

- *arithmetic expressions*

$$a + bc - d/(e + fg) -> [a + \{b * c\} - \{d/[e + \{f * g\}]\}] \tag{3.1}$$

- *logical expressions*

$$a|b\&c+!(d\hat{}(e|f\&g)) -> [a + \{b * c\} - \{d/[e + \{f * g\}]\}] \tag{3.2}$$

- *TeX/LaTeX control sequences*
- *section, subsections of books, instructions, documents,*
- *. . . etc,*

are comprehended in a form of nested structures (typically also nested into existing life experience of interpreter).

At the level of communications the problems are typically because

- *structural information is hidden, unspoken, or ambiguous,*
- *people are uncertain what structures they may need or have in mind,*
- *people are very limited in following evolving structures.*

Formalization and special software is necessary for handling the issue

Unstructured lists, typical for programming, is a source of entropy

E.g., the understanding of the code control can be illustrated by problem of matching elements in two columns in the Table

#	Control parameters user has in mind	FORTRAN namelist
0	promotion to AL	`igrid`
1	promotion to group leader	`rleft`
2	major monetary award	`rright`
3	promotion within the rank	`zbotto`
	...	`ifcoil`
	...	`iecoil`
	...	...
	...	`af2`
	...	`fcturn`
95	minor disciplinary actions	`he`
96	suspension for a week	`ecid`
97	layoff	`vsid`
98	torture	`rvs`
99	electric chair	`zvs`

The total number N_0 of possible sporadic matches of n items is $N_0 = n!$ with entropy

$$\boxed{S_0 \equiv \ln N_0 \simeq n(\ln n - 1) + \frac{1}{2}\ln(2\pi n)}$$

The common (overlapping) knowledge may significantly reduce the amount of job

Suppose both sides subdivide each set on n/k mutually recognized sections with k elements in each.

$$\begin{aligned}
\underbrace{\ldots\ldots\ldots\ldots\ldots}_{user's\ n} &= \underbrace{\underbrace{\ldots}_{k}\ \underbrace{\ldots}_{k}\ \underbrace{\ldots}_{k}\ \cdots\ \underbrace{\ldots}_{k}}_{n/k} \\
\underbrace{\ldots\ldots\ldots\ldots\ldots}_{author's\ n} &= \underbrace{\underbrace{\ldots}_{k}\ \underbrace{\ldots}_{k}\ \underbrace{\ldots}_{k}\ \cdots\ \underbrace{\ldots}_{k}}_{n/k}
\end{aligned} \tag{3.3}$$

The job is reduced to matching k parameters inside each group.

Then,

organizing the job can be made in two ways (corresponding to `'|'` and `'&'` signs)

`'|'` stands for logical "`or`", while `'&'` stands for logical "`and`".

Uncorrelated permutations (' | ' choice) inside each section is the easiest way

The total number of actions N_1 in this case

$$N_1 = \underbrace{k!\,k!\,k!\ \ldots\ k!}_{n/k\ times} = (k!)^{\frac{n}{k}},$$
$$S_1 = \ln N_1 \simeq \frac{n}{k}(k\ln k - k) = n\,(\ln k - 1) \quad \simeq \quad n(\ln n - 1) \tag{3.4}$$

Simple grouping of variables (with no management control of the job) has a little effect on entropy in the system,

while being deceptively "efficient" for small n, k

As a rule, k and n rise in time (with n/k fixed) and initial "effect" disappears

$$S_1 = n\left(\ln n - \ln\frac{n}{k} - 1\right) \quad \rightarrow \quad n(\ln n - 1) \tag{3.5}$$

"Organizing" job as uncorrelated parallel processes is a typical mistake in management

Imposing correlations is crucial for reducing the entropy level and amount of job

E.g., matching one section after another in sequence limits the number N_2 of possibilities by

$$N_2 = \underbrace{k! + k! \ldots + k!}_{n/k \text{ times}} = (k!)\frac{n}{k}, \qquad S_2 = \ln N_2 \simeq (k-1)\ln(k-1) + \ln n \quad \ll \quad n(\ln n - 1) \tag{3.6}$$

Any coherency in action results in dramatic reduction in entropy for any n.

```
{ Step0;
  { Step1;
    { Step10;
    }
    { Step11;
    }
  }
  { Step2;
  }
}
```

Coherency requires a rigorous control system. Mistake at top may cost a lot.

Nested structures automatically provide a means for controlling entropy rise

```
Box0{
  BIGBOSS b0,b1,b2,b3; /* list of control parameters*/
  JobBox0();
   Box1{
    ADMIRAL a0,a1,a2;
    JobBox1();
    ---------------------------------------------------------
     Lab0[               |     Lab1[             |     Lab2[
      HEAD h0,h1,h2;     |    HEAD h0,h1,h2;     |    HEAD h0,h1,h2;
      JobLab0();         |    JobLab1();         |    JobLab2();
    ]                    |  ]                    |  ]
   ----------------------------------------------------------
    Box2{
      SELLP h0,h1,h2;
      JobBox2();
    }
  }
}
```

Two types of structures, File-Directory type and Flow Charts are widely used.

Cb-structure includes these both types and also recognizes parallel processes

Functional position of control parameters (or managers) inside the structure is crucial

The typical (Cb-)stuctural element is shown as

$$
\begin{array}{l}
\{ \\
\quad \mathtt{LEADER}\quad \mathtt{L0,L2,L3;}\quad \text{/* providing correlations across the parallel sections */} \\
\quad [\,\mathtt{CX}_0] \;\;|\;\; [\,\mathtt{CX}_1] \;\;|\;\; [\,\mathtt{CX}_2] \;\;|\;\; [\,\mathtt{CX}_3] \;\;|\;\; [\,\mathtt{CX}_4] \;\;|\;\; [\,\mathtt{CX}_5] \\
\quad \mathtt{MONITOR}\quad \mathtt{M0,M2,M3;}\quad \text{/* monitoring and selling the output junk */} \\
\}
\end{array}
\qquad (3.7)
$$

and includes both "leading" and "monitoring" control parameters. In management such a sutuation corresponds to meta-stable "inverse population" with a tendency of conversion the leaders into monitors, and then, loss of correlations.

Business relying on uncorrelated "CX"s is inefficient and falls into "activity trap"

Integrated modeling is a new level of code development and simulating plasma physics

Preserving consistency in dissimilar things: physics, programming, and use of the codes, should be recognized as a fundamental problem faced by integration.

It cannot be solved in a fusion loved way by hiring a person per subroutine or module

The key to its solution is in formalization starting from the ground level and developing computer means for assisting the integration and control.

The example of CodeBuilder, presented in the talk, demonstrates how significant can be progress at different levels of integration

- *implementation of physics*
- *code development for reactor relevant devices (ITER, IST)*
- *uniform control of numerical codes and design of GUI*
- *communications of code components (modules) and separate codes ("code taliking")*
- *organizing I/O*
- *On-Line Help for unexperience user*
- *code development documentation*

Managing integration is itself the integration problem of efficient use of "zero entropy"
things, (i.e., money & computers & intellects of people), and for success
it can be addressed only in a scientifically rigorous manner

Money comes from the top and their flow control is indirect (by results you produce)

Compared to managers, who have at their disposal only the intellect,

integrated modelling community has a big advantage of being fluent with
using computer power (not only for calculations)