PHASING

Content:

- Basic ideas
- Practical aspects in Lautrec
- Examples
- LAM impacts on the global model code, « Aladin Tech'book »
- Future evolution of the code: some outlook

Basic ideas

Aladin is (only) the bi-Fourier version of Arpège, with the
Noticeable exception of the NHS code and its diversions around
The Laprise solution.
Consequences:
• to understand Aladin, one absolutely needs to learn Arpège
• phasing 1: keep the LAM elements consistent with the core soft.
• to what extent do these stringent technical constraints bend back
On science ?
• phasing 2: new developments need usually to be phased before
Entering the Arp/Ald official cycles
• phasing 3: a (commonly) maintained common trunk should
Benefit to all the parties of the project

Basic ideas (2)

There is NO identifiable interface beteen Arpège and Aladin:
The LAM code is spread throughout the global code.

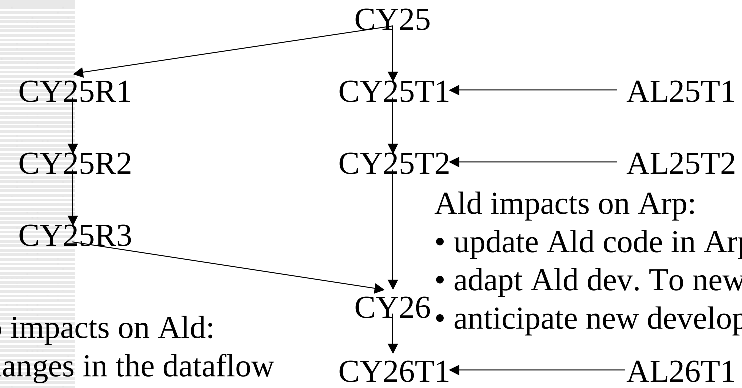
Modularization per function, instead per model, in the Arp/Ald System does not prevent from phasing (eg.: the spectral transforms)

Code maintenance is a must:
- optimization
- lifetime of the software (… and the project ?)
- evolution of coding rules, coding standards
- portability

Basic ideas (3)

Code phasing is a must:
- follow the continuing evolution of the Arpège/IFS code
- sources of modifications:
    - ECMWF -> IFS -> Reading cycles
    - CNRM(/GMAP) + COMPAS + Aladin partners -> Tlse cycles
- « old » rules for coding and phasing: « IFMG »
    - Integration
    - Flexibility
    - Modularity
    - Generality
- code management: clearcase & see Ryad's « from source code Management to the production of binaries »

Basic ideas (4)



Ald impacts on Arp:
- update Ald code in Arp
- adapt Ald dev. To new cycle
- anticipate new developments

Arp impacts on Ald:
- changes in the dataflow
- interfaces/modules
- architecture

Practical aspects: phasing in Lautrec

First, a common Arpège/IFS cycle is created: every 6 months
- Reading cycle sent to Toulouse
- evaluation of modifications and source code merging with Tlse: GMAP & COMPAS/GCO
- compilation and first validation
- pre-cycle sent back to Reading for evaluation and final validation
- Reading declares officially the common cycle

This is followed by the phasing of Aladin to the common global Model cycle, usually 1 or 2 months later.

Practical aspects: phasing in Lautrec (2)

Time for source code phasing:
• around 3 weeks (+/- 1 week)
• first binaries for validation ready 1 or 2 weeks later (taking into Account merger problems and other hesitations)
• team of foreigner phasers: 4-6 persons, among 1-4 newcomers In the last teams (AL13, AL15, AL25T1)
• phasers stay 6 weeks, and leave in general soon after the first Binaries for validation are made available (…)

Principle of a « lagged validation »: model confs are validated first, Then observation confs (usually two months later, sticking to the Arpège 4D-VAR validation)

Practical aspects: phasing in Lautrec (3)

• one phasing every 6 months (… 1 year …)
• regular phaser meetings, use of a common source code manager For a controlled source code evolution (ClearCase)
-> see ClearCase documentation on the web
• after validation, production of « export » cycles for remote Installation
• backphasing either for Lautrec or remote ALADIN cycles: Quite difficult -> should be (supposed to be) avoided (forbidden)

Example

How to phase the following codes ?:
CY25T1/AL25T1:

```
SUBROUTINE SUTOTO
REAL_B :: RCOEFF
IF (LELAM) THEN
  RCOEFF=_ZERO_
ELSE
  RCOEFF=_ONE_
ENDIF
```

Becomes in Arpège/IFS cycle CY26:

```
SUBROUTINE SUTOTO
REAL_B :: RCOEFF(2)
IF (LELAM) THEN
  RCOEFF=_ZERO_
THEN
  RCOEFF(1)=_ONE_
  RCOEFF(2)=_HALF_
ENDIF
```

Later on, one computes:
F = « map factor to the power R » and R=RCOEFF
Changed in Arpège into:
F = « map factor to R1 / max map factor to R2 »
While the aladin case was simply changed from:
F = 1 into F = « map factor to the power R »

Another change comes from Aladin AL25T1:

```
SUBROUTINE SUTOTO
REAL_B :: RCOEFF
IF (LELAM) THEN
  RCOEFF=0.25_JPRB
ELSE
  RCOEFF=_ONE_
ENDIF
```


Other documentations

- « Coding rules for Arpège/Aladin » by Ryad El Khatib
- tutorials on ClearCase by GCO team
- on-line documentation on « gmkpack » by Ryad El Khatib
- « Phaser's guide » by Jaouad Boutahar, Radi Ajjaji and Jean-François Geleyn -> a bit obsolete
- documentation from the Budapest 2002 workshop on code Maintenance, available on-line

LAM impacts on the global model code, « Aladin Tech'book »

By Mark Zagar and Claude Fischer
• tries to be an exhaustive but undetailed document about Aladin Plugs inside Arpège
• useful for phasers, but also sent regularly to ECMWF
• list of duplicated routines
• list of Aladin counterparts or additional routines
• « Golden rules » of LAM specifities:
  • coupling, lateral boundaries (SL, FP interpolations)
  • plane projection: SL, Obs operators, FP
  • spectral space: spectral coefficients, mean wind, elliptic Truncation and total wavenumber definition

LAM impacts on the global model code, « Aladin Tech'book » (2)

The LAM impacts dictate a methodology for phasing:
• think per application and about dataflow
• think also per routine
This is somehow contradictory for the brain, but complementary !
Rules:
• pure Arpège code -> keep it for granted, but check for possible Remote impacts on Aladin code
• Aladin developments inside Arpège -> keys LELAM and LRPLANE -> check for differences
• call to a specific Aladin routine -> check for differences in the Arpège counterpart
• call to an extra Aladin routine -> less danger for phasing (?)
• duplicated routines -> report Arpège modifications
Whatever modification is seen, a scientific and technical Understanding of it would be « better than expected »
-> **do not consider phasing as a blind-going operation**

Future evolution of the code: some outlook

• resist/adapt to the creeping modularization imposed by ECMWF
• manage the internal code divergences: NHS solutions, (possibly) new proposals for high-resolution LAM 3D-VAR, New microphysical variables, Meso-NH physics, other AROME Impacts (?)
• some parts of the code should remain « close » to Arpège: Fullpos, observation operators, general setup and control etc … but with strong local implications generally due to geometry: weights, data distribution
• some impacts and strategical decisions depend on the final Choices for AROME

Future evolution of the code: some outlook (2)

Presently, we move towards a mixed up situation, where the common Core of Arpège/Aladin would be steered by some strong LAM Specific switches.
One could keep in mind the stability problem of a 3-body Gravitational system: only two stable solutions exist: small planet Close to one star and the other star far away, or a tight binary System with a far away satellite.
The present dilemma is that none of the 2 solutions probably Perfectly suits to the Aladin system, neither today nor in future.

Future evolution of the code: some outlook (3)

Other challenges:
• data assimilation requires more external functions and more Scripts: data pre-processing, assimilation suites, verifications
• more global entries and more binaries: blending executables, Lam_flag, data pre-processed files
• up-to-date documentation

Future evolution of the code: some outlook (4)

Work at a distance: how to integrate local developments in the Common cycle ?
• architectural choices to provide a clearer guidance for developpers
• for small contributions: e-mail exchanges
• for huge contributions: visits to Toulouse of a required length, Workshops, … human tolerance !

Workstation version: (…)

VALIDATION

Content:

- Basic validation
- « mitraillette »
- Performances: memory and CPU
- Verification and scores

Basic validation

Check for:

- complete execution of the application (END CNT0)
- evolution of spectral norms: if no scientific change, nor any deep Recoding, then the spectral norms should not change from one Cycle to another (or with only a minor or « small exponential » Divergence)
- other diagnostics: RMS (GOM-arrays in Variational), norm of the Gradient, cost function, gridpoint norms (fullpos)
- plot from time to time the historical files, including the Mathematical extension zone
- be careful for: CFL criterion, read/write operations, coupling files
- some Arpège confs should run with Aladin exe.: forecast, screening

« mitraillette »

Automated validation tool for Aladin (supervised by Patrick Saez):

- e001 hydrostatic, adiabatic, DFI, eulerian/SL
- e001 hydrostatic plus physics
- e001 NHS, Bubnova and P/C solutions (later, +d4)
- 2D plane vertical model
- e001 without coupling
- incremental DFI
- e501, e401, e801, e601 hydrostatic, adiabatic, eulerian
- e501, e401 NHS adiabatic, eulerian
- e927, ee927, several fullpos domain extractions

Documentation on the web

« multi-user » and friendly interface

Performances: memory and CPU

Check for:

- memory consumption
- CPU time for execution
- I/O operations
- percentage of vectorization (if relevant)

At constant NPROMA, NPROC

Verification and scores

Check for:

• scores with respect to surface and 3D observations (Synop, Temp)
• scores with respect to previous cycle
-> scores could be computed on a personal basis, or in the frame Of a double suite (for operations)
• systematic re-launch of pre-defined remarkable cases
• subjective evaluation from a pre-operational team (the « Marc Tardy » guy) or from forecasters
• evaluation with respect to a verifying analysis