Including Improvement of the Execution Time in a Software Architecture of Libraries with Self-Optimisation

Luis-Pedro García

Javier Cuenca

Domingo Giménez

University of Murcia

SPAIN

Outline

- Introduction
- Self-Optimised Linear Algebra Routine Samples
- Experimental Results
- Conclusions

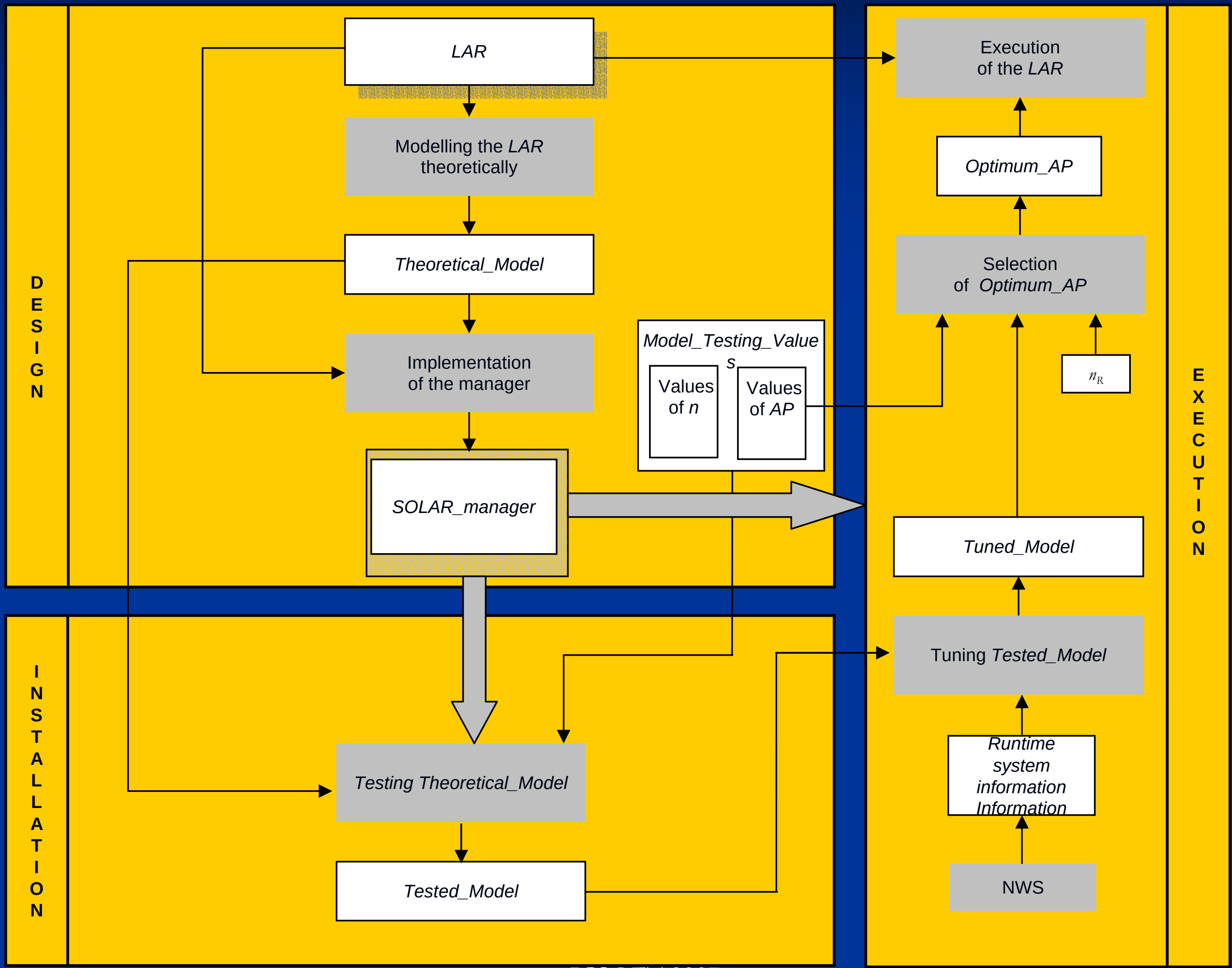
Outline

- **Introduction**
- Self-Optimised Linear Algebra Routine Samples
- Experimental Results
- Conclusions

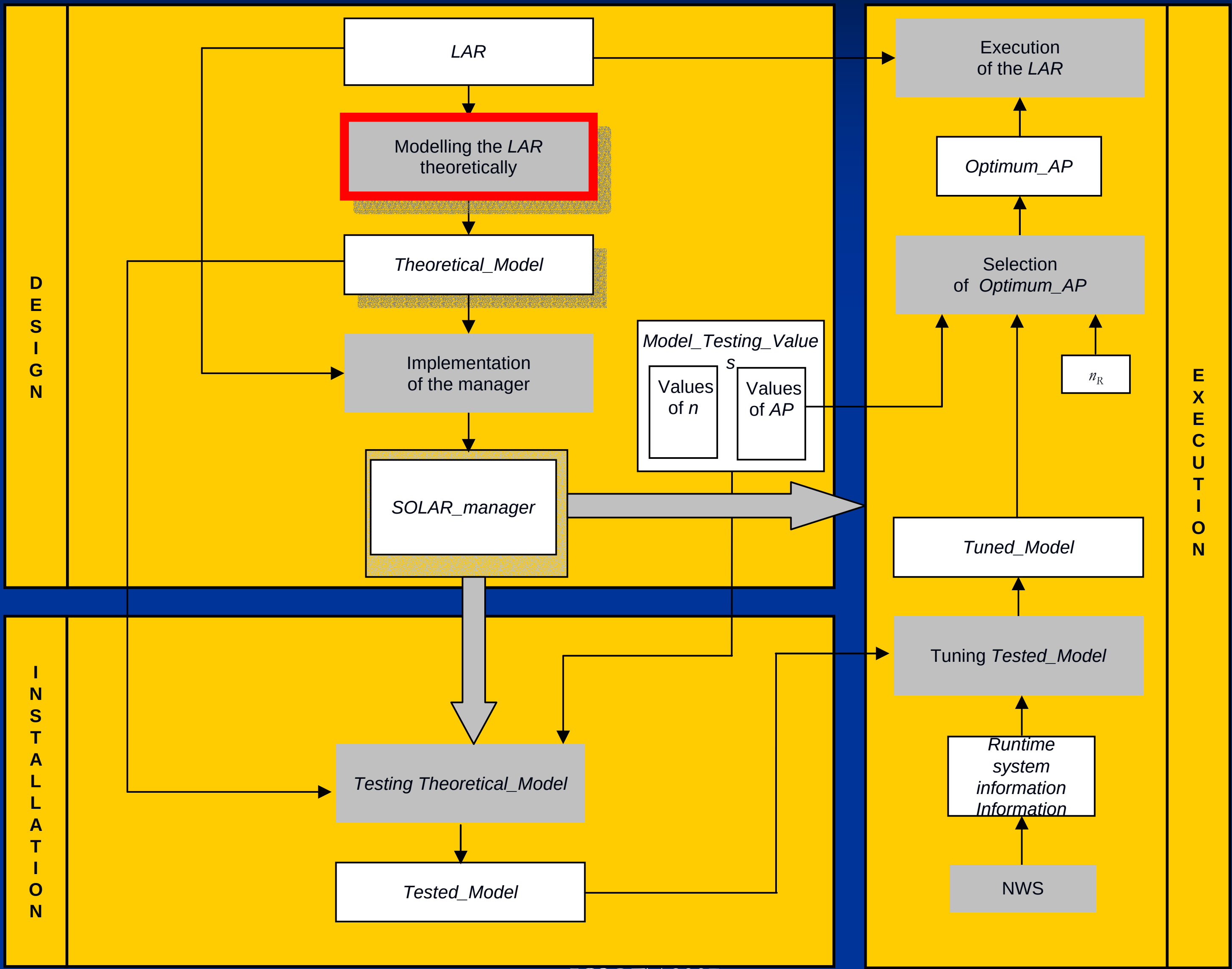
Introduction

- **Our goal**: to obtain linear algebra parallel routines with auto-optimization capacity.
- **The approach**: model the execution time of the routine to tune, taking advantage of the natural hierarchy existing in linear algebra programs.
- **The basic idea** is to start from lower level routines (multiplication, addition, etc.) To model the higher level ones (Strassen multiplication, parallel multiplication, LU, QR, Cholesky, etc).
- **In this talk**:
  - A **remodelling stage** is proposed if the information at one level is not accurate enough.
  - This new model will be built using **polynomial regression**.

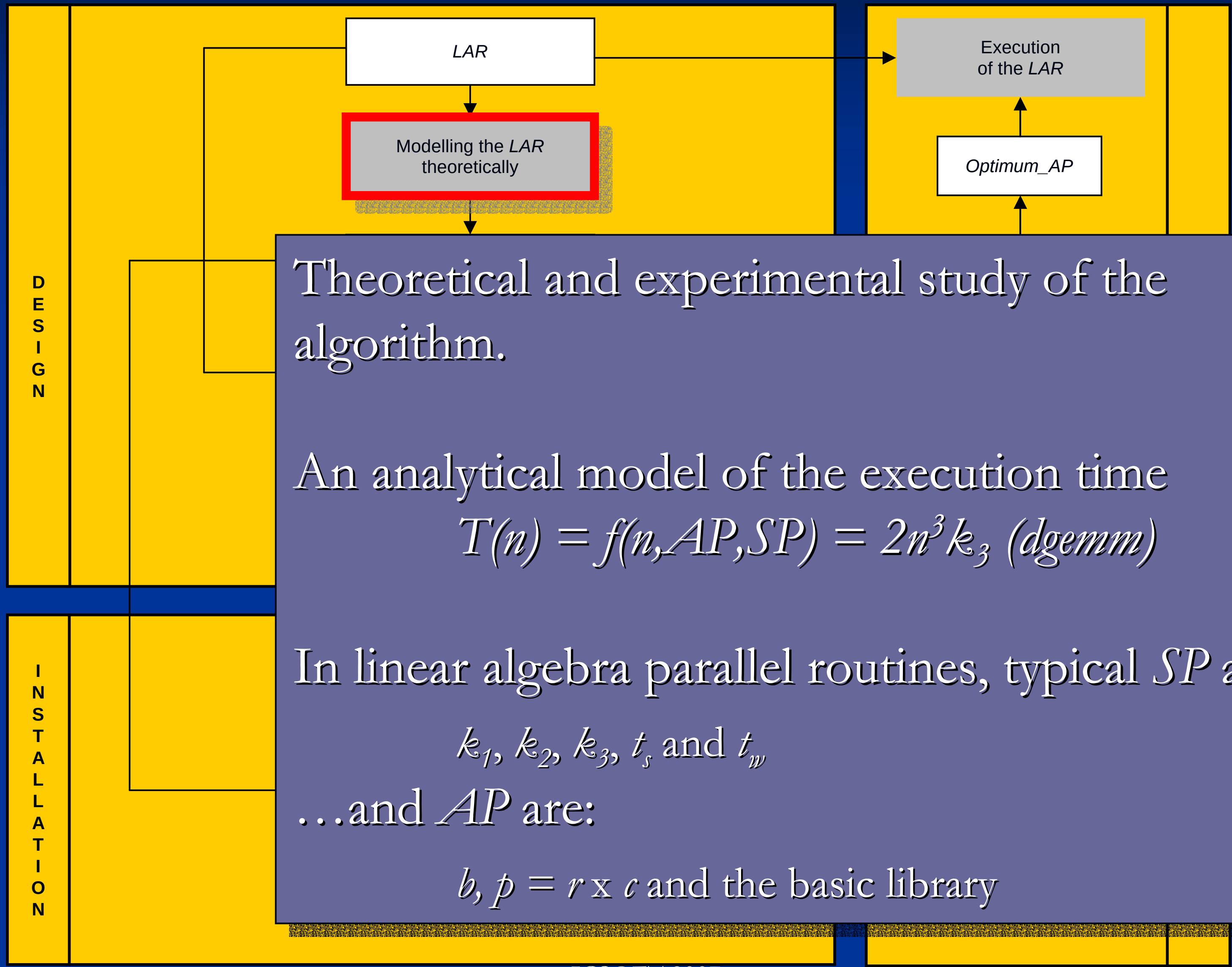
Introduction



Introduction



Introduction



Theoretical and experimental study of the algorithm.

An analytical model of the execution time

$$T(n) = f(n, AP, SP) = 2n^3 k_3 \ (dgemm)$$

In linear algebra parallel routines, typical *SP* are:

k_1, k_2, k_3, t_s and t_w

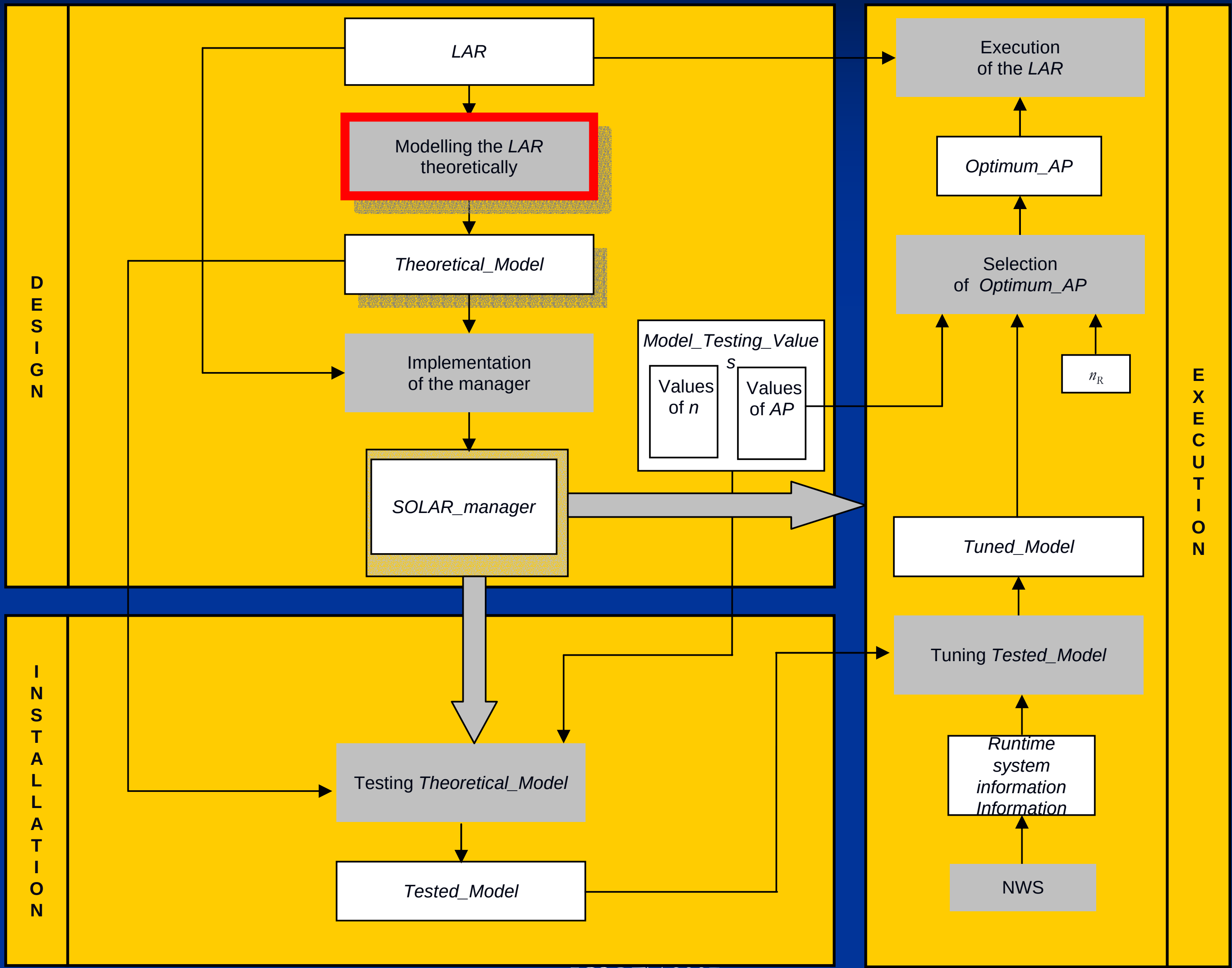
…and *AP* are:

b, $p = r$ x c and the basic library

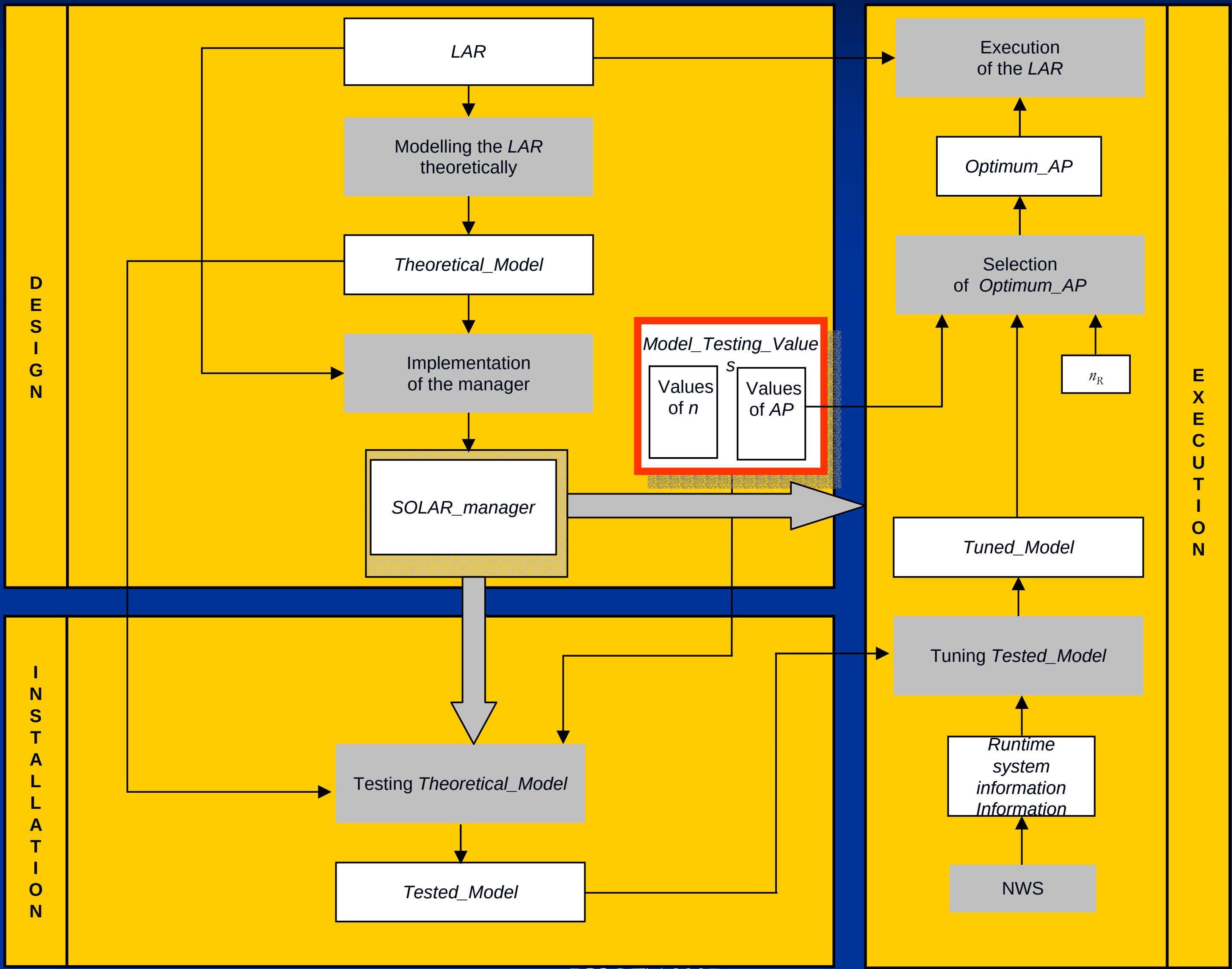
Introduction

- Theoretical and experimental study of the algorithm
- An analy OCULTA me
  - $T(n) = f(n,AP,SP) = 2n^3k3$ *(dgemm)*
- In linear algebra parallel routines, typical *AP* are:
  - b, $p = r \times c$ and the basic library
- …and *SP* are:
  - k_1, k_2, k_3, t_s and t_w

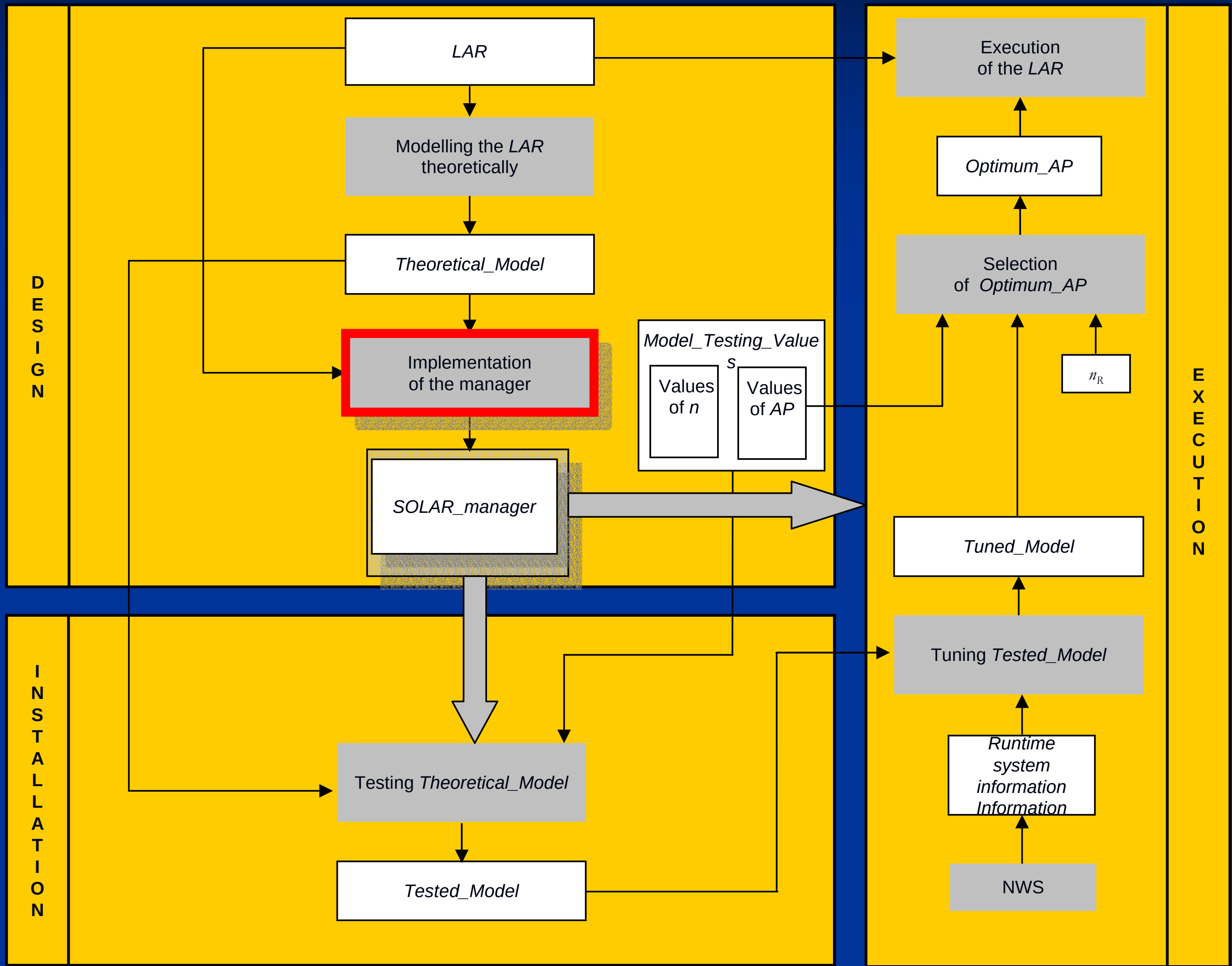
Introduction



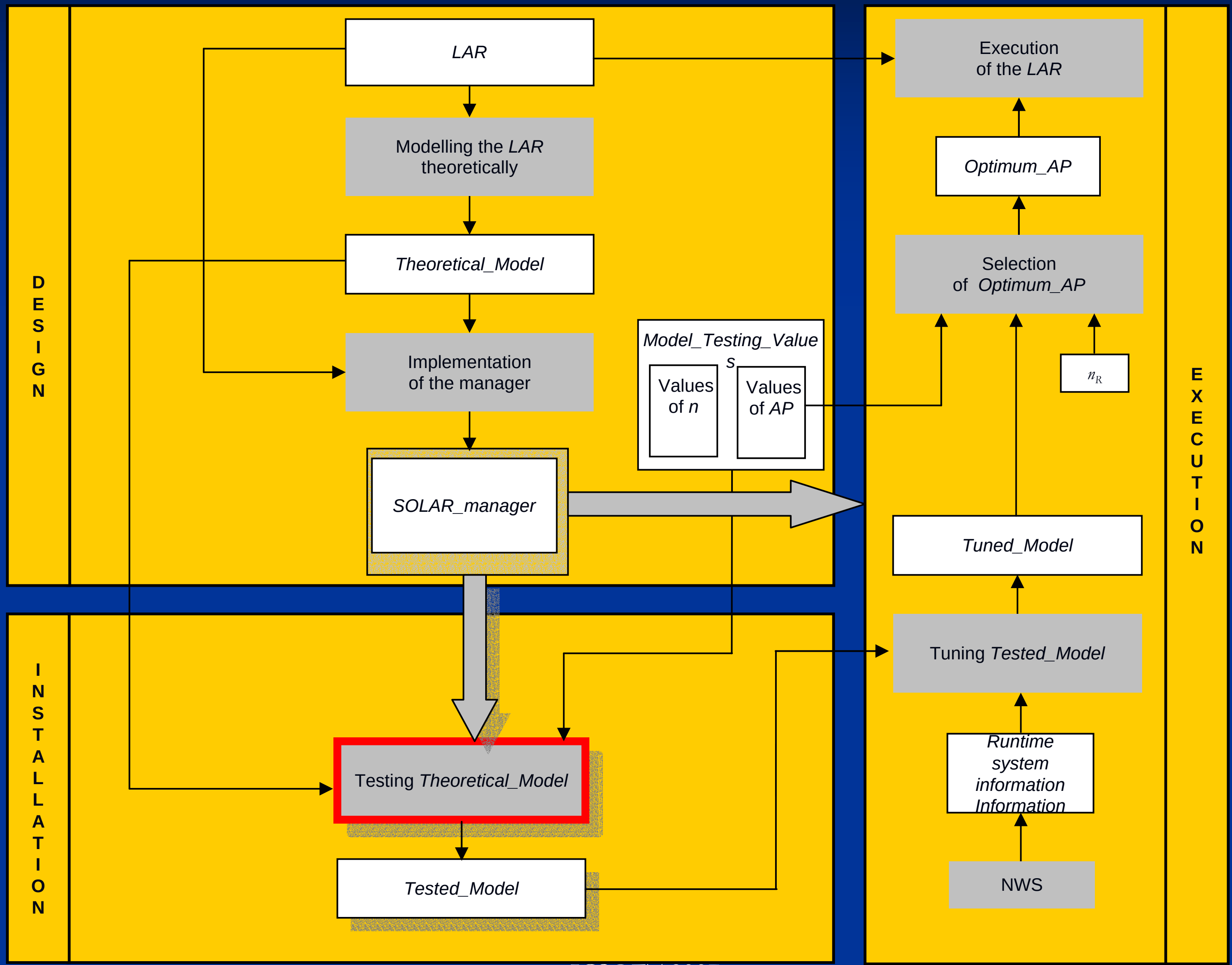
Introduction



Introduction



Introduction



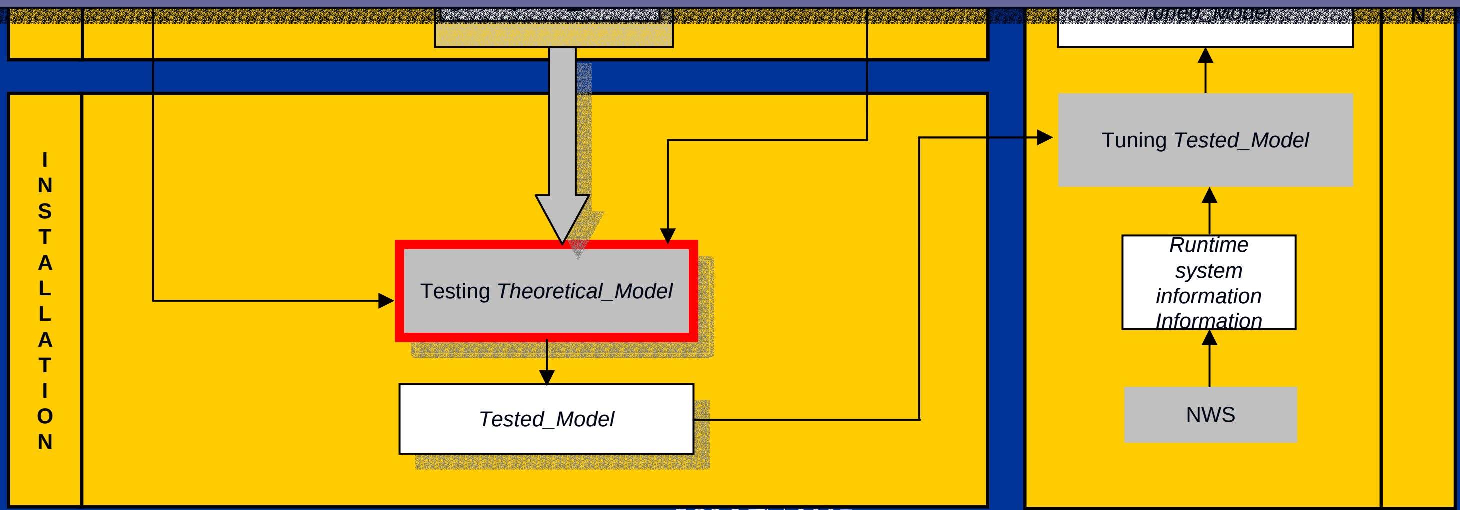
Testing the model:

Remodelling de Linear Algebra Routine (*LAR*)

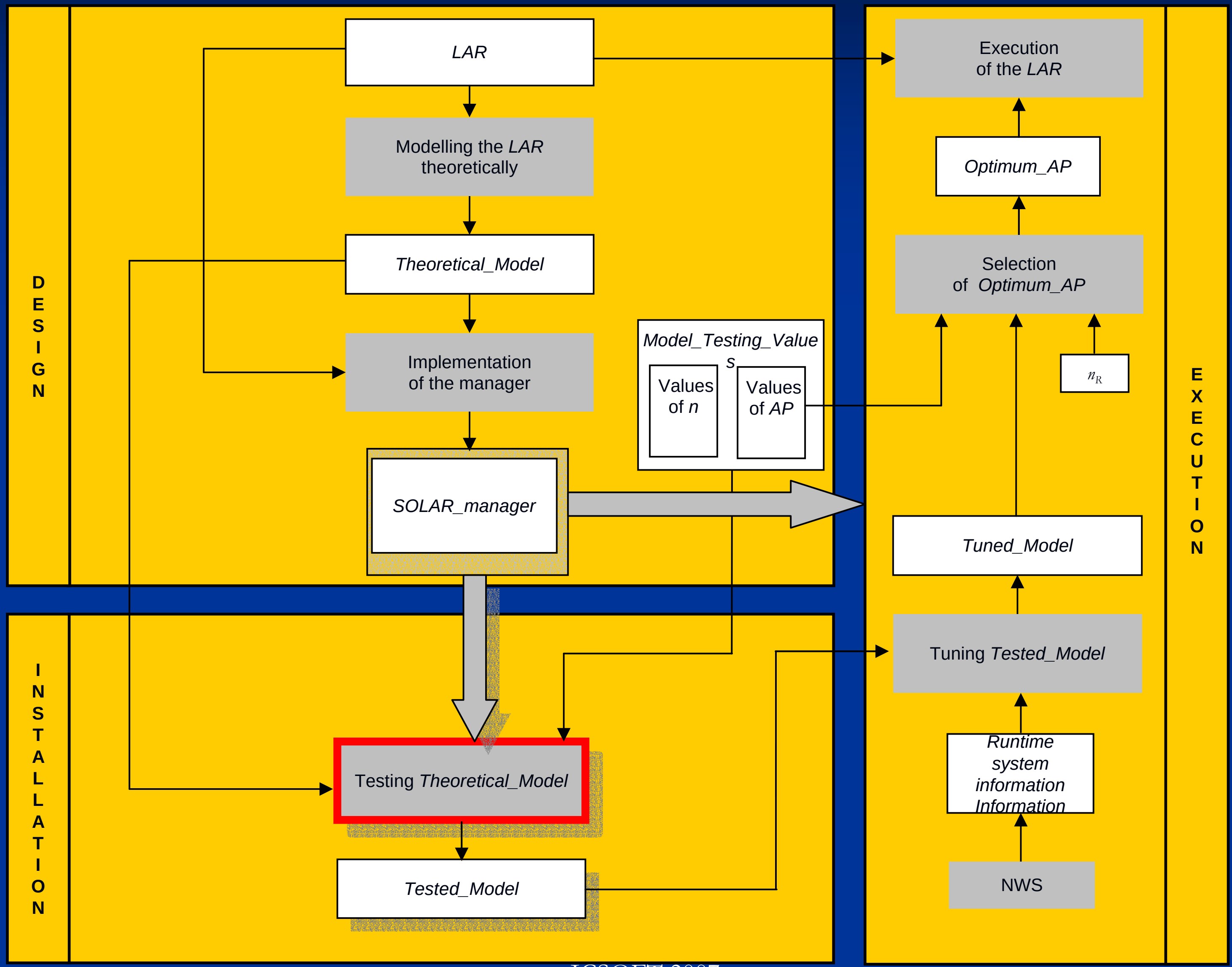
Designing a polynomial scheme from the original model for different combinations of *n* and *AP*:

$$T(n,AP) = a_0 n^3/p + a_1 n^3 p + a_2 n^3 + a_3 n^2/p + a_4 n^2 p + a_5 n^2 + \ldots$$

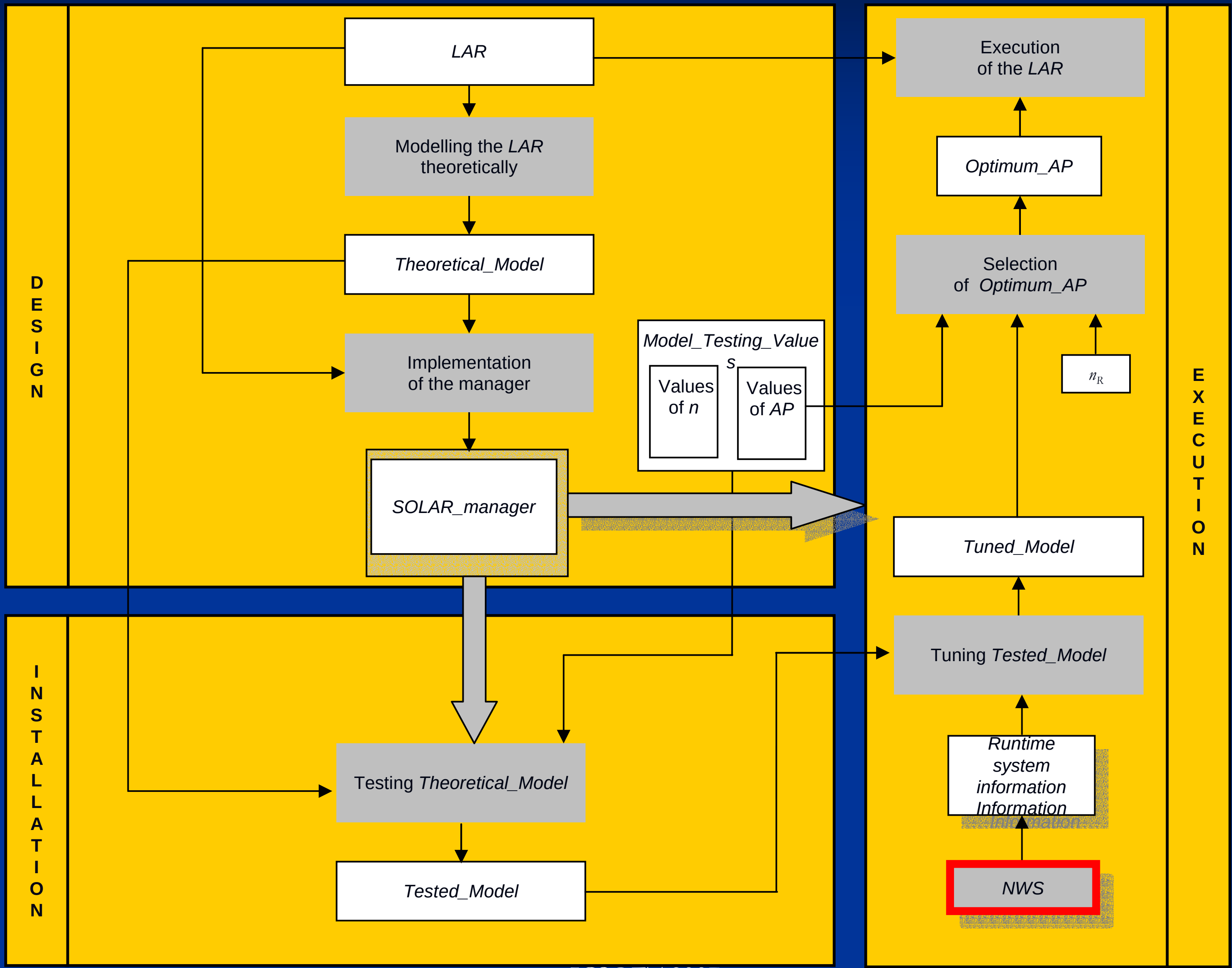
The coefficients $a_0, a_1, a_2, \ldots$ must be calculated



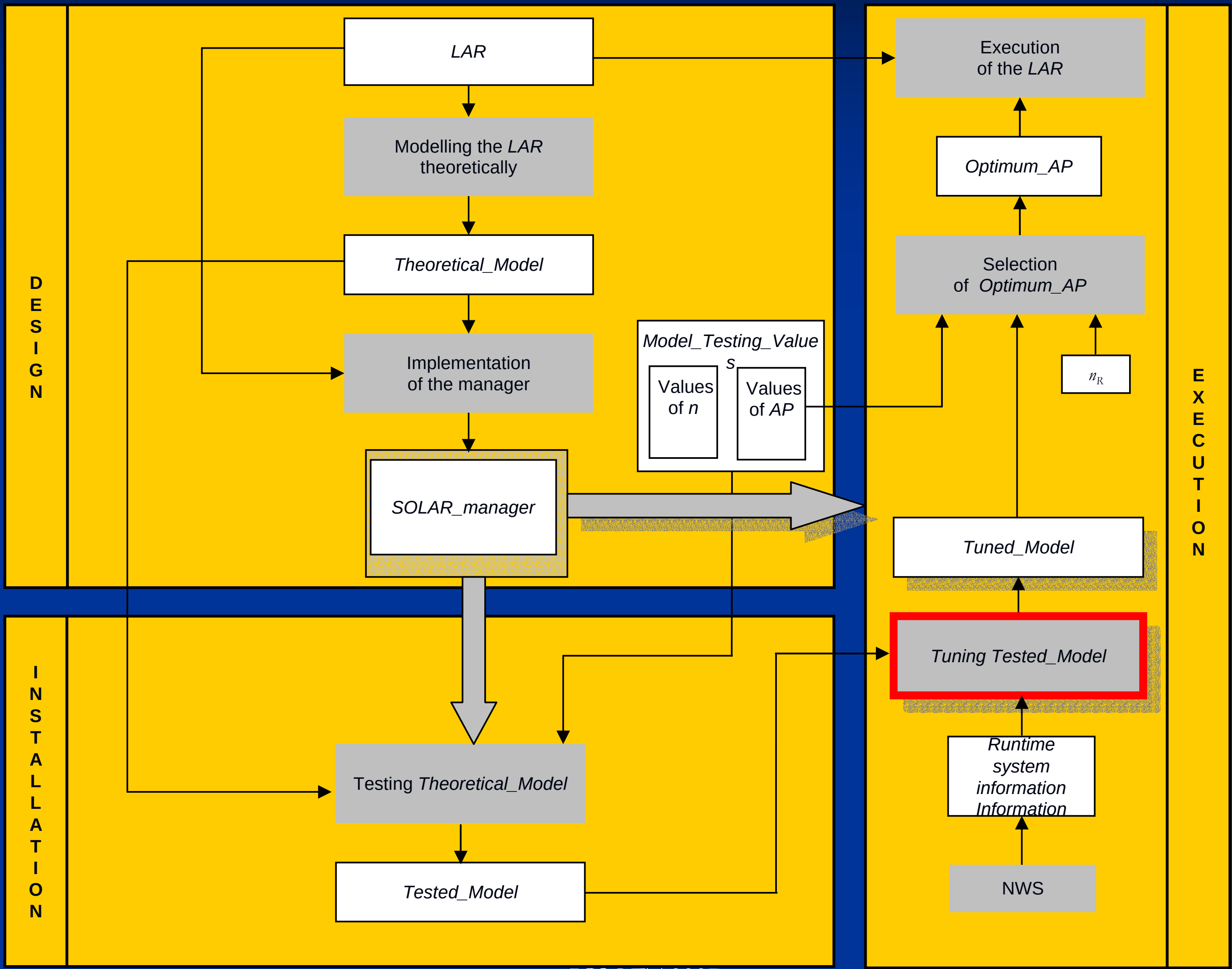
Introduction



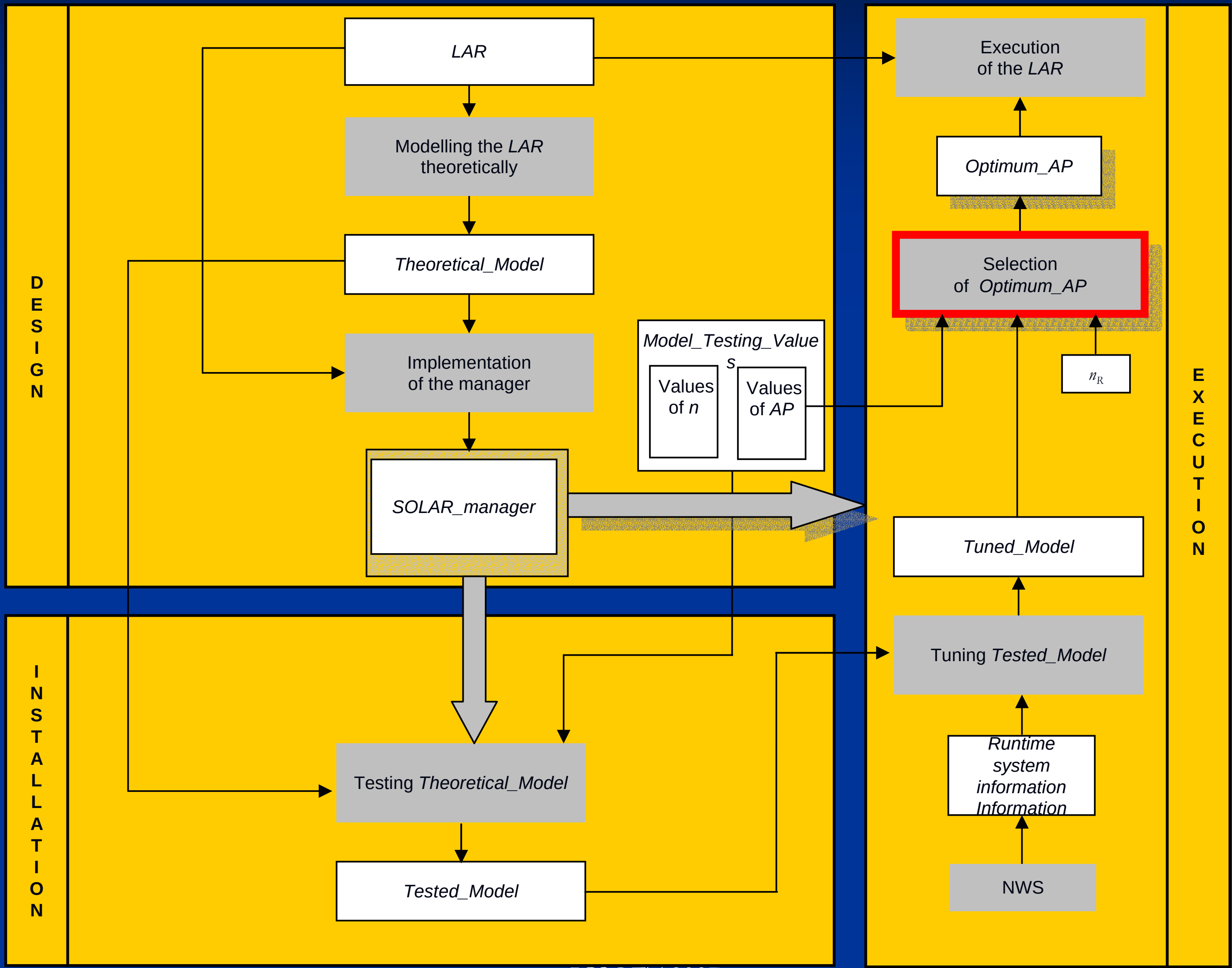
Introduction



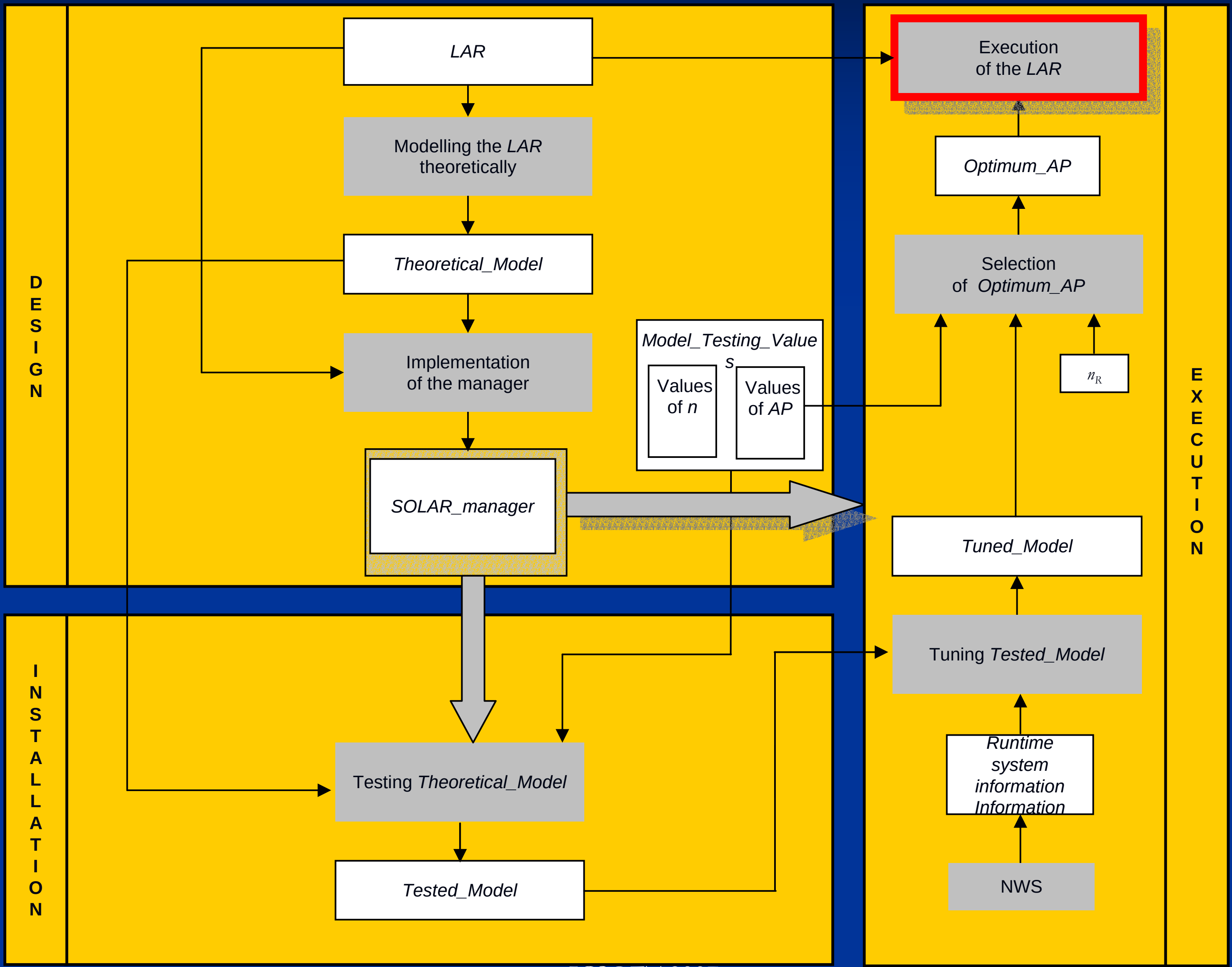
Introduction



Introduction



Introduction



Introduction

- Remodelling de Linear Algebra Routine (*LAR*)
  - Designing a polynomial scheme from the original model OCULTA d AP:

$$T(n,AP) = a_0 n^3/p + a_1 n^3 * p + a_2 n^3 + a_3 n^2/p + a_4 n^2 * p + a_5 n^2 + \ldots$$

  - The coefficients $a_0, a_1, a_2, \ldots$ must be calculated

Introduction

- In order to determine these coefficients, four different methods are proposed:

OCULTA

  - FI-ME: FIxed Minimal Executions
  - VA-ME: VAriable Minimal Executions
  - FI-LS: FIxed Least Square
  - VA-LS: VAriable Least Square

Outline

- Introduction
- Self-Optimised Linear Algebra Routine Samples
- Experimental Results
- Conclusions

Self-Optimised LAR

- Strassen Matrix-Matrix multiplication

$$T = 7^l t_{mult}\left(\frac{n}{2^l}\right) + 18\sum_{i=1}^{l} 7^{i-1} t_{add}\left(\frac{n}{2^i}\right)$$

- $t_{mult}(n/2^l)$: Theoretical execution time for matrix multiplication. BLAS3 function DGEMM

- $t_{add}(n/2^i)$: Theoretical execution time for matrix addition. BLAS1 function DAXPY

Outline

- Introduction
- Self-Optimised Linear Algebra Routine Samples
- Experimental Results
- Conclusions

Experimental Results: Strassen

- **Systems:**

  Xeon: Linux Intel Xeon 3.0 GHz workstation

  Alpha: Unix HP-Alpha 1.0 GHz workstation

- **Models for DGEMM and DAXPY**
  - DGEMM: Third order polynomial (20 samples)
    - *n_min* = 500, *n_max* = 10000, *n_inc* = 500
  - DAXPY: Sixth order polynomial (31 samples)
    - *n_min* = 64, *n_max* = 2000, *n_inc* = 64

Experimental Results: Strassen

- Testing de Model in Xeon.

(Time in seconds)

n	*l*	Mod.	Exp.	Dev.(%)
3072	1	**11.75**	**12.86**	8.58
3072	2	13.90	13.63	1.99
3072	3	37.04	15.76	135.06
4096	1	**27.21**	**29.71**	8.41
4096	2	28.59	30.10	5.02
4096	3	48.76	33.34	46.26
5120	1	**53.14**	56.83	6.51
5120	2	53.53	**56.43**	5.13
5120	3	71.08	60.19	18.09
6144	1	96.48	96.32	0.17
6144	2	**95.39**	**93.69**	1.82
6144	3	110.40	98.39	12.21

- Testing de Model in Alpha.

(Time in seconds)

n	*l*	Mod.	Exp.	Dev. (%)
3072	1	29.96	29.70	0.89
3072	2	28.54	27.82	2.57
3072	3	**17.55**	**27.61**	36.46
4096	1	69.85	70.85	1.43
4096	2	66.04	64.55	2.30
4096	3	**57.82**	**62.56**	7.58
5120	1	135.03	134.67	0.26
5120	2	125.76	123.38	1.92
5120	3	**118.12**	**118.45**	0.28
6144	1	229.79	232.27	1.07
6144	2	211.10	210.88	0.11
6144	3	**201.15**	**199.33**	0.92

Experimental Results: Strassen

- The optimal value of *AP* vary for different systems and problem sizes.
- In Xeon and for *n* = 5120 the model make a wrong prediction, but the execution time is only 0.71 % higher.
- However, in Xeon, the deviation ranged from 0.17 % to 135.06 %:

IT IS NECESSARY TO BUILD AN IMPROVED MODEL

6144	3	110.40	98.39	12.21

6144	3	**201.15**	**199.33**	0.92

Remodelling Strassen

- The scheme consists of defining a set of third grade polynomial functions from the theoretical model:

$$T(n,l) = 2 \times 7^l \left(\frac{n}{2^l}\right)^3 M(l) + \frac{18}{4} n^2 A(l) \sum_{i=1}^{l} \left(\frac{7}{4}\right)^{i-1}$$

- *M(l)* and *A(l)* must be calculated.
- For each *l*, *n* varies and the values of *M(l)* and *A(l)* are obtained by least squares.

Remodelling Strassen

- The sc
  definin
  grade
  functi
  theore

l	$M(l)$	$A(l)$
1	2.22×10^{-10}	3.89×10^{-08}
2	2.24×10^{-10}	3.03×10^{-08}
3	1.99×10^{-10}	3.03×10^{-08}
4	3.48×10^{-10}	1.53×10^{-08}

- $M(l)$ and $A(l)$ must be calculated.
- For each l, n varies and the values of $M(l)$ and $A(l)$ are obtained by least squares.

Remodelling Strassen

- Now the set of values for $M(l)$ and $A(l)$ can be approximated by a polynomial in l and thus we have a single model for any combination of n and l.
- $M(l)$ is approximated by a second grade polynomial

$$M(l) = m_0 + m_1 l + m_2 l^2$$

- $A(l)$ is approximated by a first grade polynomial

$$A(l) = a_0 + a_1 l$$

Remodelling Strassen

$$M(l) = 1.9 \times 10^{-10} + 4.58 \times 10^{-11} \times l - 1.45 \times 10^{-11} \times l^2$$

$$A(l) = 4.38 \times 10^{-08} - 5.131 \times 10^{-09} \times l$$

single model for any combination of n and l.

- $M(l)$ is approximated by a second grade polynomial

$$M(l) = m_0 + m_1 l + m_2 l^2$$

- $A(l)$ is approximated by a first grade polynomial

$$A(l) = a_0 + a_1 l$$

Remodelling Strassen

n	l	Mod.	Exp.	Dev. (%)
2688	1	**7.87**	**8.80**	11.92
2688	2	8.40	9.67	15.23
2688	3	10.28	10.52	2.38
3200	1	**13.02**	**14.51**	11.92
3200	2	13.56	15.51	14.38
3200	3	16.00	16.30	1.87
5120	1	56.80	56.71	0.17
5120	2	**56.44**	57.01	1.00
5120	3	60.04	**55.09**	8.25
5632	1	75.78	74.92	1.12
5632	2	73.50	74.56	1.45
5632	3	**71.70**	**70.97**	1.03

Remodelling Strassen

n	l	Mod.	Exp.	Dev. (%)
2688	1	**7.87**	**8.80**	11.92
2688	2	8.40	9.67	15.23
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

- In Xeon and for n = 5120 the model make a wrong prediction, but the execution time is only 3.49 % higher.
- Now, with remodelling, the deviation is smaller and ranged from 0.17 % to 15.23 %

Outline

- Introduction
- Self-Optimised Linear Algebra Routine Samples
- Experimental Results
- Conclusions

Conclusions

- The use of modelling techniques can contribute to reduce the execution time of the routines.
- The modelling time must be small:
  - Reduce the number of samples.
  - Use small problem sizes for modelling.
- The method has been applied successfully to the Strassen Matrix-Matrix multiplication and can be applied to other linear algebra routines.