



# Plans d'expériences pour équations différentielles ordinaires

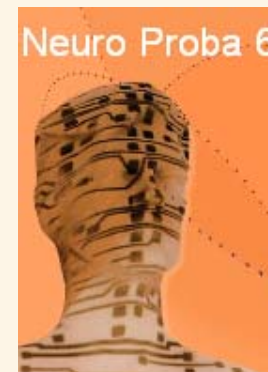
[patrice.kiener@netral.com](mailto:patrice.kiener@netral.com)

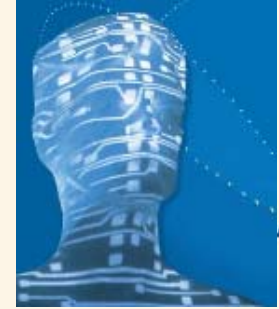
# Editeur de logiciels en statistique non-linéaire



## + Conseil, Etudes et Formation en :

1. **Modélisation, simulation et optimisation** de phénomènes statiques et dynamiques décrits par :
  - des modèles physiques de connaissance
  - des modèles boîte noire (réseaux de neurones)
  - des équations différentielles ordinaires
2. **Réduction du nombre d'essais** :  
Plans d'expériences pour modèles non-linéaires
3. Estimation précise des quantiles et indices de capacité des **distributions non-gaussiennes**
4. **Génération automatique de code C** pour tout modèle non-linéaire algébrique et EDO.





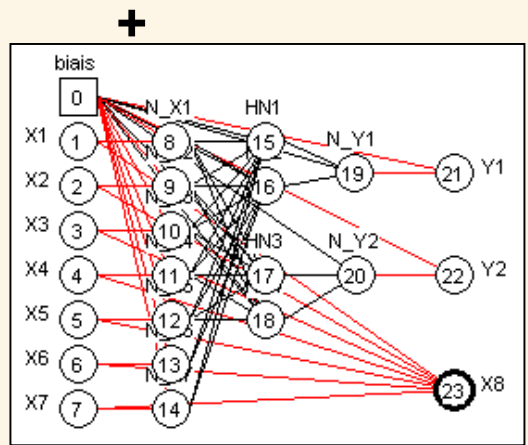
# Exemple NEURO ONE

## Modélisation, analyse de sensibilité et calcul inverse

X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0	0	0	0,13	0,09	0,05	0,73	550	31
0	0	0,03	0,13	0	0,12	0,03	0,69	551	22
0	0,01	0,01	0,05	0,09	0,13	0	0,72	551	28
0	0	0,04	0,12	0,02	0,13	0,01	0,69	552	26
0	0,04	0	0,04	0,1	0,13	0	0,7	552	21

$$\sum_{i=1}^8 X_i = 1$$

$$X_8 = 1 - \sum_{i=1}^7 X_i$$



**ENTREES**

Valeur  $\nabla d(Y1)/dx$

X1	0.4286	148.462
X2	0.5	-210.133
X3	0.32	14.508
X4	0.5	103.001
X5	0.607	119.222
X6	0.5	-269.515
X7	0.4	-138.266

**SORTIES**

Désirabilité globale 0.492

Y1	Levier	Désirabilité	
480.11	0.073	0.984	
Y2	Levier	Désirabilité	
54.994	0.395	0.5	
X8		Désirabilité	
0.29		1	

**Algorithme d'apprentissage**  
(Levenberg-Marquardt précis à 10 décimales)



# Exemple NEURO PROBA

## Lois de probabilité multimodales

L'estimation des quantiles et indices de capabilité est cruciale en fiabilité

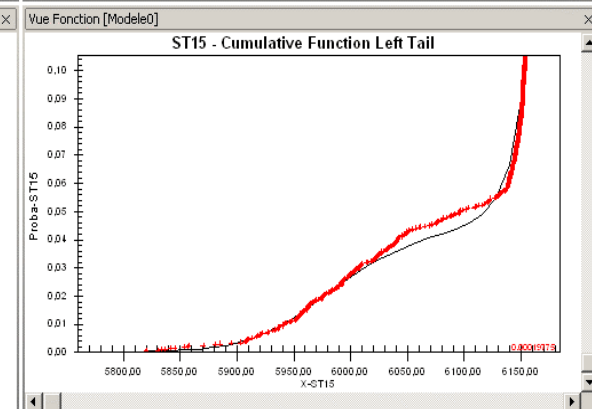
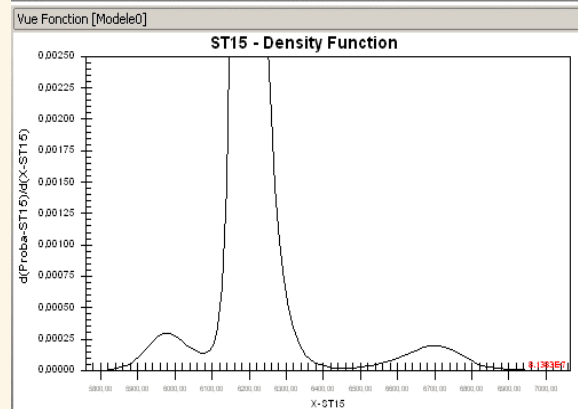
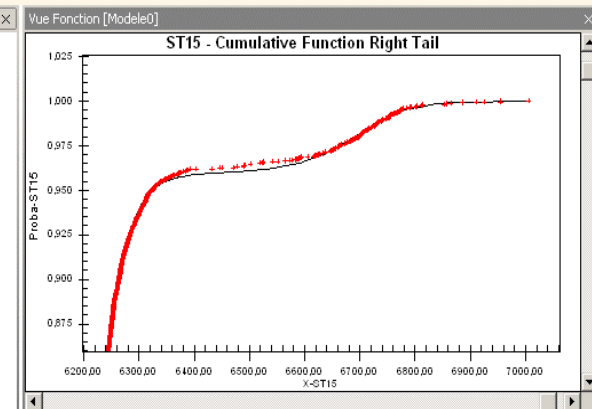
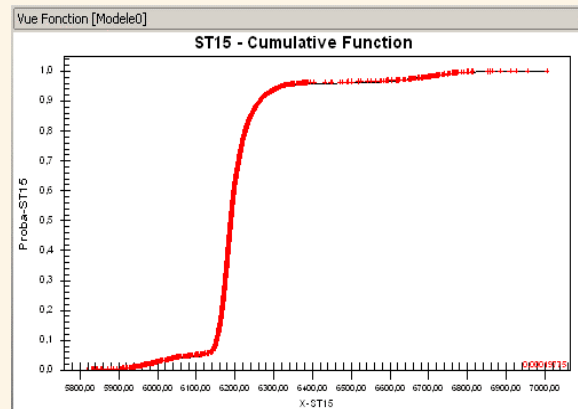
Dans cet exemple :

Cpk-Gauss = 2,19

Cpk-RN = 1,30

(seuil 0/1 = 1,65)

NEURO PROBA dispose d'un algorithme de calcul inverse pour simuler des phénomènes aléatoires





# Exemple NEURO CODE

## Générateur automatique de code C

- Tout modèle de connaissance non-linéaire
- Tout réseau de neurones



- Le logiciel Neuro Code génère en moins de 10 s 3000 lignes de code C :
  - Fonction de transfert
  - Intervalles de confiance
  - Algorithme d'apprentissage (Levenberg-Marquardt)



```
Neuro Code [NSDE54.NML]
Fichier  Edit  Affichage  Atelier  Aide
Général  X  Transfert  En-tête
Code C (*.c;*.h)

NetworkName : nsde54
FileName    : C:\MES DOCUMENTS\N1-EXTENSION\GAZONOR\NSDE54.NML

*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "nsde54.h"
/* Please check name of included header file */

void nsde54transfer(double *inputs, double *outputs, double *states)
/*Module : nsde54
Method : nsde54transfer
Visibility : Public
Arguments : inputs: double* -> variables d'entrée
           outputs: double* -> variables de sortie
           states: double* -> variables d'état

Description: Effectue le transfert par le réseau.
Note : les poids utilisés sont les poids par défaut nsde54weights */
{
    nsde54transferw(nsde54weights, inputs, outputs, states);
}

void nsde54transferw(double *weights, double *inputs, double *outputs, double *states)
/*Module : nsde54
Method : nsde54transferw
Visibility : Public
Arguments : weights: double* -> les poids courants
           inputs: double* -> variables d'entrée
           outputs: double* -> variables de sortie
           states: double* -> variables d'état
```



# Exemple NEURO PEX

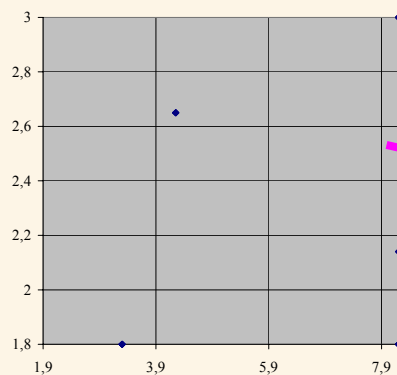
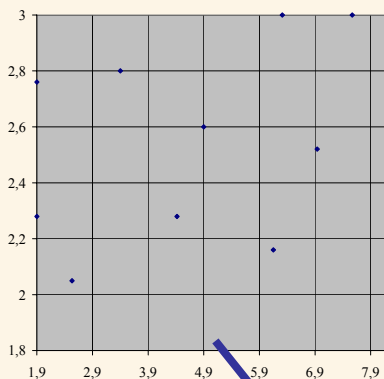
## Plans d'expériences pour modèles explicites

→ Minimisation du nombre et du coût global des essais  
(i.e. Meilleure qualité du modèle pour un plus faible coût)

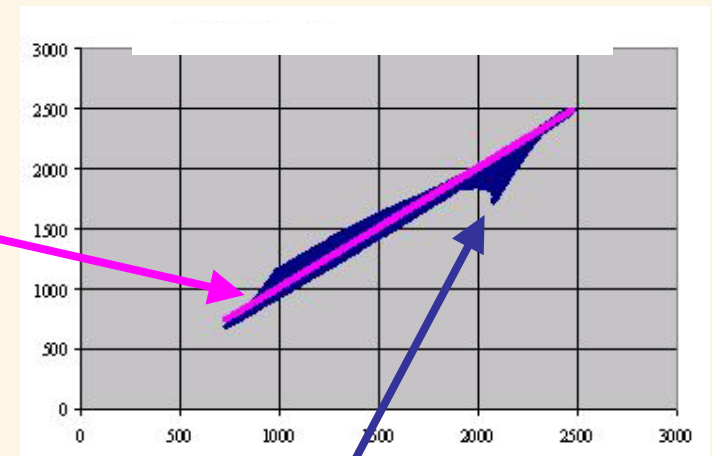
$$Z = (aX + b) \sin \left[ 2 \arctan \left( \frac{Y}{cX + d} \right) \right] + e \quad \text{avec } e \rightarrow N(0, \sigma^2) \quad X, Y, a, b, c, d$$

10 points répartis sur

un plan aléatoire ou un plan optimal



1.500 points de validation



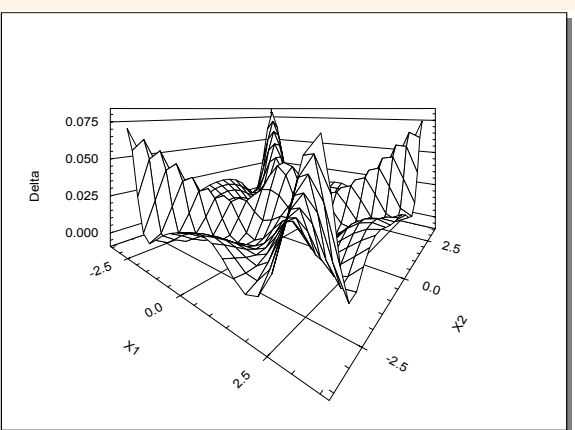
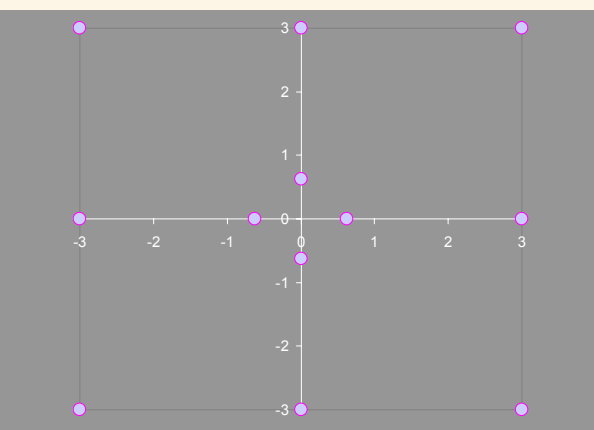
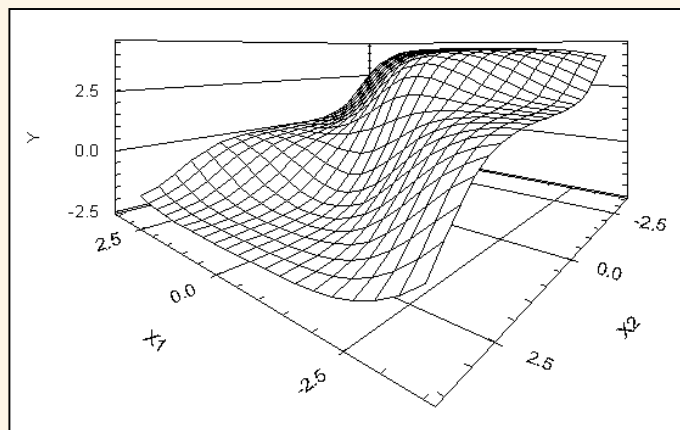
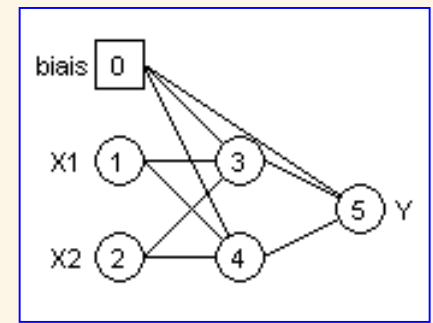


# Exemple NEURO PEX

## Plans d'expériences pour réseaux de neurones

→ les points optimaux sont situés là où la variance est la plus élevée

$$Y = \theta_1 + \theta_2 \tanh(\theta_3 + \theta_4 X_1 + \theta_5 X_2) + \theta_6 \tanh(\theta_7 + \theta_8 X_1 + \theta_9 X_2) + \varepsilon \quad \text{avec } \varepsilon \rightarrow N(0, \sigma^2)$$





# Exemple NEURO PEX

## Plans d'expériences pour équations différentielles

Cinétique chimique  $A \rightarrow B \rightarrow C$

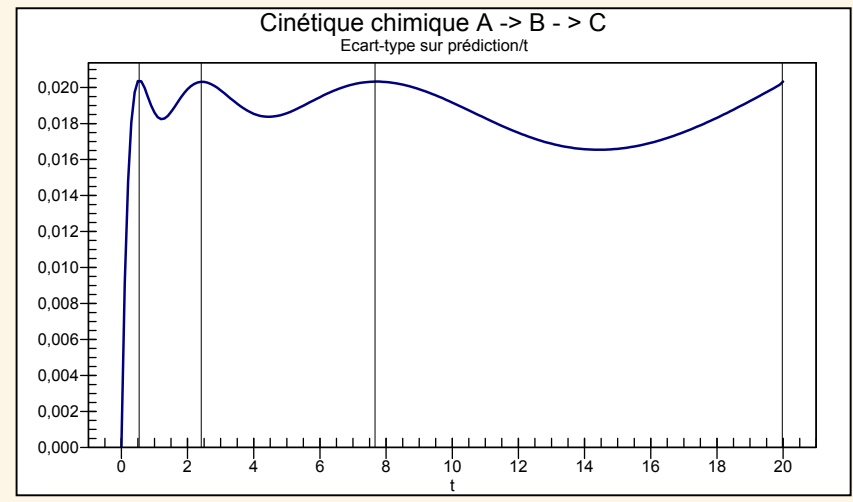
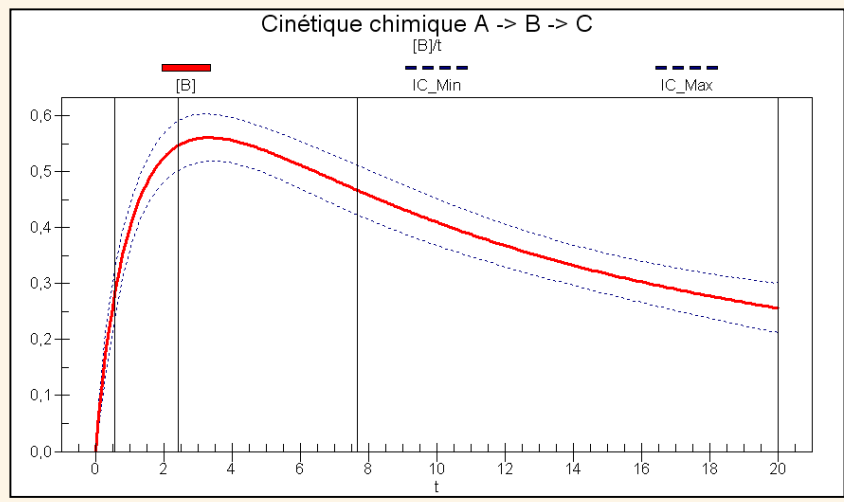
$$\frac{\partial[A]}{\partial t} = -\theta_1[A]^{\lambda_1}$$

$$\frac{\partial[B]}{\partial t} = \theta_1[A]^{\lambda_1} - \theta_2[B]^{\lambda_2}$$

$$\frac{\partial[C]}{\partial t} = \theta_2[B]^{\lambda_2}$$

$[A], [B], [C]$   
 $\theta_1, \lambda_1, \theta_2, \lambda_2$

**Solution via DLL dédiée      4 points optimaux     $t = 0,5$      $t = 2,4$      $t = 7,7$      $t = 20$**

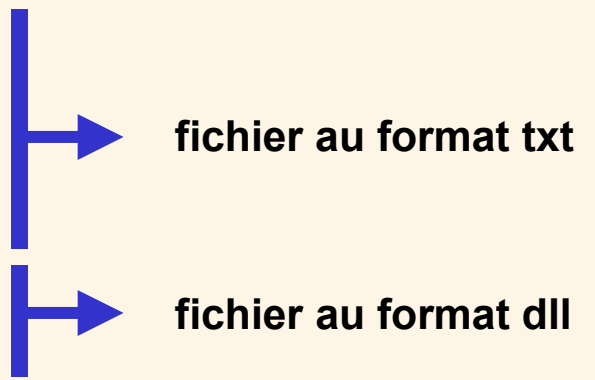




# Formalisme NETRAL

## Différents modèles...

	Fonction	Fonction + Gradient	Fonction + Gradient + Hessien
Modèle explicite	équation à fournir	auto	auto
Réseau de neurones	auto	auto	auto
EDO	équation à coder	à coder	à coder facultatif





# Formalisme NETRAL

Format txt converti en format XML pour les modèles de connaissance

## Format XML (réseaux de neurones)

```
sin-arctg.mgl - Bloc-notes
Fichier Edition Format Affichage ?
Model
Const
  ModelName = Modele X-Y-Z;
  Comment = ;
Input
  X in [1.9..8.2];
  Y in [1.8..3];
Output
  Z;
Param
  a=737 in [0.41..0.5];
  b=757 in [1..2.5];
  c=6.4 in [0..1.5];
  d=-3.95 in [1..2.5];
Function
  Z = (a*Y+b)*sin(2*arctan(X/(c*Y+d)));
end.
```

RN2E\_2NC\_RIVAL5.NML - Bloc-notes

Fichier Edition Format Affichage ?

```
<?xml version="1.0" encoding="ISO-8859-1"
<!DOCTYPE MODEL SYSTEM "http://www.netral
<MODEL version="27">
<NETWORK class="NeuronNetwork" neuronna
<LAYER rank="1">
<NODE class="Neuron" nodeId="0" neu
</NODE>
</LAYER>
<LAYER rank="0">
<NODE class="NeuronSigma" nodeId="1"
<INTERFACE position="0" maximum=
</NODE>
<NODE class="NeuronSigma" nodeId="2"
<INTERFACE position="1" maximum=
</NODE>
</LAYER>
<LAYER rank="1">
<NODE class="NeuronSigma" nodeId="3"
<LINK class="Synapse" synapsename
<VALUE>0</VALUE>
</LINK>
<LINK class="Synapse" synapsename
<VALUE>1</VALUE>
</LINK>
<LINK class="synapse" synapsename="w(2)" style="0" linkid="2" parentid="2">
<VALUE>-1</VALUE>
</LINK>
</NODE>
<NODE class="NeuronSigma" nodeId="4" neuronname="H0(1)" fonction="Hyperbolic tangent">
<LINK class="Synapse" synapsename="w(3)" style="0" linkid="3" parentid="0">
<VALUE>0</VALUE>
</LINK>
<LINK class="Synapse" synapsename="w(4)" style="0" linkid="4" parentid="1">
<VALUE>1</VALUE>
</LINK>
<LINK class="Synapse" synapsename="w(5)" style="0" linkid="5" parentid="2">
<VALUE>1</VALUE>
</LINK>
</NODE>
</LAYER>
<LAYER rank="2">
```

Nom Réseau de neurones

Architecture

Pré traitement

Entrées  Activation

Cachés

Sorties

Post traitement



# Formalisme NETRAL

## Format dll pour les équations différentielles ordinaires (1)

```
Kinetic_Box_Lucas_Ex | UBox_Lucas_Ex  
library Kinetic_Box_Lucas_Ex;  
  
uses  
  SysUtils,  
  Classes,  
  UBox_Lucas_Ex in 'UBox_Lucas_Ex.pas';  
  
{$R *.res}  
{$E dlm}  
  
exports  
{$WARNINGS OFF}  
  getauthorcode index 1,  
  getinfo index 2,  
  getparam index 3,  
  setparam index 4,  
  transfer index 5,  
  getgradient index 6,  
  getparamspace index 7,  
  getexperimentalspace index 8,  
  getiniparams index 9,  
  getinputgradient index 10,  
  gethessian index 11,  
  getfindiff index 12,  
  getname index 13,  
  getmsg index 14,  
  getgradient2 index 15,  
  multitransfer index 16,  
  setexperimentalspace index 17;  
{$WARNINGS ON}
```

```
Kinetic_Box_Lucas_Ex | UBox_Lucas_Ex  
function DeriveMainx(X, Y: NReal; var LTX,  
  stdcall;  
begin  
  if X = 0 then  
    Result := Teta1  
  else  
    try  
      T1X := Teta1 * X;  
      LTX := 1 - (1 - Lambda1) * T1X;  
      ELO := Power(LTX, L1L);  
      PYO := Power(Y, Lambda2);  
      Result := Teta1 * ELO - Teta2 * PYO;  
    except  
      Result := 0;  
      LTX := 1;  
      ELO := 0;  
      PYO := 0;  
    end;  
end;
```

```
Kinetic_Box_Lucas_Ex | UBox_Lucas_Ex  
procedure DeriveOrdre1(X: NReal; Y, DY: PArrayOfReal; var LTX,  
  EL1, LLN, PY1, TLB, LYO: NReal); stdcall;  
begin  
  DY[0] := DeriveMainx(X, Y[0], LTX, ELO, PYO, T1X);  
  if Y[0] <= 0 then  
    DY[1] := 1 // dy/dTeta1  
  else  
    try  
      EL1 := Power(LTX, L1L - 1);  
      LLN := Ln(LTX);  
      PY1 := Power(Y[0], Lambda2 - 1);  
      TLB := -Teta2 * Lambda2 * PY1;  
      LYO := ln(Y[0]);  
  
      DY[1] := TLB * Y[1] + (1 - T1X) * EL1; // dy/dTeta1  
      DY[2] := TLB * Y[2] - PYO; // dy/dTeta2  
      DY[3] := TLB * Y[3] + Teta1L12 * ELO * LLN + L1L * Teta1  
        // dy/dLambda1  
      DY[4] := TLB * Y[4] - Teta2 * PYO * LYO; // dy/dLambda2  
    except  
      FillChar(DY[NEntree], NParam * SizeOf(NReal), #0);  
    end;  
end;
```



# Formalisme NETRAL

## Format dll pour les équations différentielles ordinaires (2)

```
Kinetic_Box_Lucas_Ex  UBox_Lucas_Ex
procedure DeriveOrdre2(X: NReal; Y, DY: PArrayOfReal); stdcall
var
  LTX, ELO, EL1, EL2, LLN, PYO, T1X, PY1, TLB, PY2, LYO, T2
  T2L21PY2Y2L1, LLYO1: NReal;
  nd, np: Integer;
begin
  np := NParam * (NParam + 1) div 2;
  nd := NEntree + NParam + np;
  FillChar(DY^, nd * SizeOf(NReal), #0);
  DeriveOrdre1(X, Y, DY, LTX, ELO, PYO, T1X, EL1, LLN, PY1,
if Y[0] > 0 then
try
  PY2 := Power(Y[0], Lambda2 - 2);
  EL2 := Power(LTX, L1L - 2);
  T2L21PY2 := Teta2 * Lambda2 * (Lambda2 - 1) * PY2;
  T2L21PY2Y2L1 := T2L21PY2 * Y[2] + Lambda2 * PY1;
  LLYO1 := Lambda2 * LYO + 1;

  DY[5] := TLB * Y[5] - T2L21PY2 * Sqr(Y[1]) - X * EL2 *
  // d2y/dTeta1 dTeta1
  DY[6] := TLB * Y[6] - Lambda2 * Y[1] * (PY1 + Teta2 * P
  // d2y/dTeta1 dTeta2
  DY[7] := TLB * Y[7] - T2L21PY2 * Y[1] * Y[3] + (1 - T1X
  LLN + (L1L - 1) * T1X / LTX); // d2y/dTeta1 dLambda1
  DY[8] := TLB * Y[8] - Teta2 * PY1 * Y[1] * (1 + Lambda2
  1) * Y[4] / Y[0]); // d2y/dTeta1 dLambda2
  DY[9] := TLB * Y[9] - Lambda2 * Y[2] * (2 * PY1 + Teta2
  // d2y/dTeta2 dTeta2
  DY[10] := TLB * Y[10] - T2L21PY2Y2L1 * Y[3]; // d2y/dTe
  DY[11] := TLB * Y[11] - T2L21PY2Y2L1 * Y[4] - PYO * LYO
  * Y[2]; // d2y/dTeta2 dLambda2
```

```
Kinetic_Box_Lucas_Ex  UBox_Lucas_Ex
function DoCompute: Boolean;
var
  i: Integer;
  YY: array[0..14] of Real;
begin
if not Computed then
begin
  for i := 1 to nPasMax do
    TableX[i] := XSpace[0] + (XSpace[1] - XSpace[0])
    ZeroMemory(@TableRes, 15 * nPasMax * SizeOf(NReal)
    ZeroMemory(@YY, 15 * SizeOf(NReal));
  // Intégration par Runge-Kutta d'ordre 4.
    IntegrerRk4_n(DeriveOrdre2, XSpace[0], XSpace[1],
    @TableRes);
    Computed := true;
end;
  Result := Computed;
end;
```



# Exemple NEURO PEX

## Chargement du modèle

Modèle

Nombre de facteurs

Modèle graphique

Domaine expérimental

Navigation: [Left] [Right] [Up] [Down] [Check] [List]

Facteur	Minimum	Maximum
t	0	20

Domaine paramétrique

Navigation: [Left] [Right] [Up] [Down] [Check] [List]

Paramètre	Valeur	Minimum	Maximum
Teta1	0.7	0	6
Teta2	0.2	0	4
Lambda1	2	1.01	5
Lambda2	2	1.01	5

Homogène estimé

Ecart-Type du bruit  Estimation bruit à x%

Nombre d'expériences à générer

t	[B]
0	NAN
0.1	NAN
0.2	NAN
0.3	NAN
0.4	NAN
0.5	NAN
0.6	NAN
0.7	NAN
0.8	NAN
0.9	NAN
1	NAN
1.1	NAN
1.2	NAN
1.3	NAN
1.4	NAN
1.5	NAN
1.6	NAN
1.7	NAN
1.8	NAN
1.9	NAN
2	NAN



# Exemple NEURO PEX

## Calcul des plans D- ou X-optimaux

- Liste des plans disponibles
- Plan factoriel complet
  - Plan aléatoire
  - Plan D-Optimal
  - Plan X-Optimal

Paramètres:

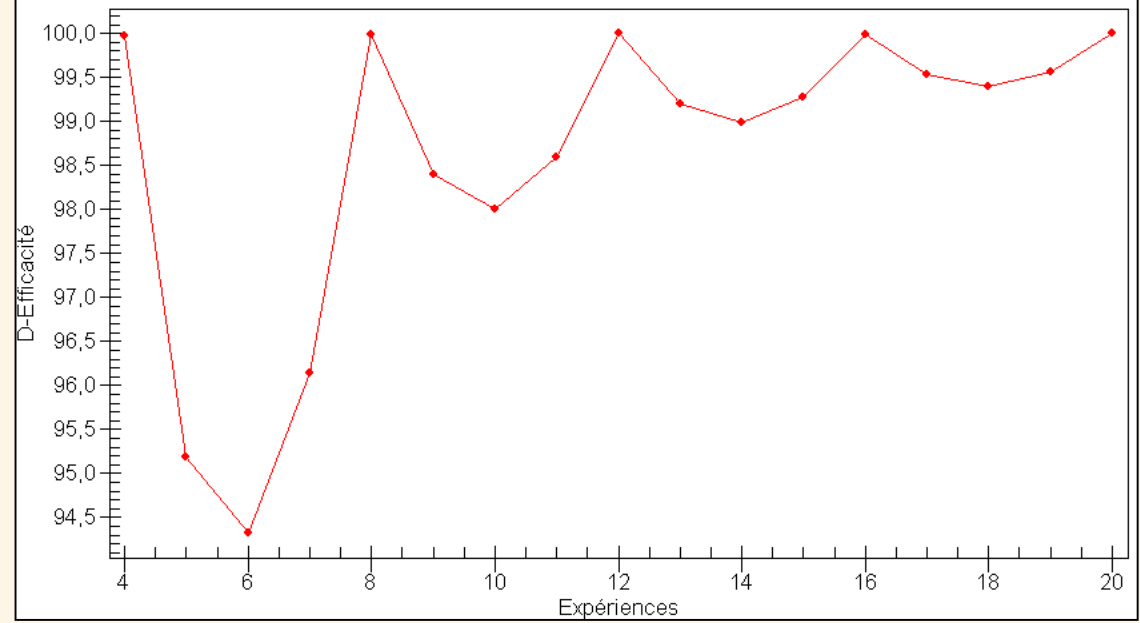
Nombre minimum d'expériences:

Nombre maximum d'expériences:

**Plan**

t  
7.7  
2.4  
0.5  
20

Résultats  
D-Efficacité/Expériences





# Exemple NEURO PEX

## Calcul des plans D- ou X-optimaux (2)

- Liste des plans disponibles
- Plan factoriel complet
  - Plan aléatoire
  - Plan D-Optimal
  - Plan X-Optimal

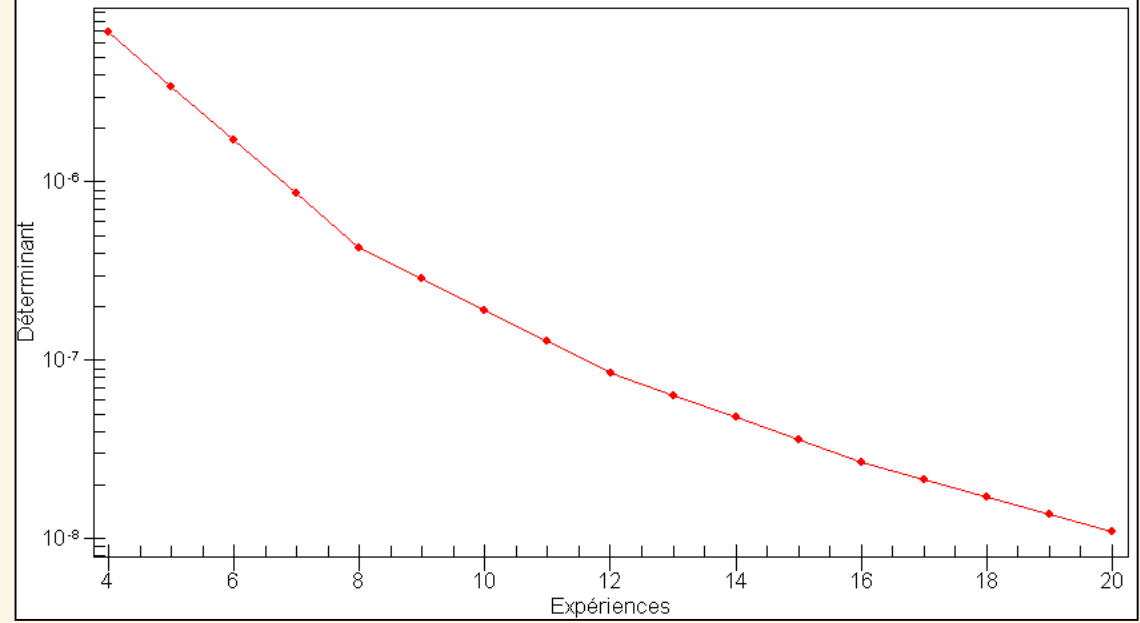
Paramètres:

Nombre minimum d'expériences:

Nombre maximum d'expériences:

Plan	
t	
7.7	
2.4	
0.5	
20	

Résultats  
Déterminant/Expériences





# Exemple NEURO PEX

## Rapport de diagnostic

### RAPPORT DE DIAGNOSTIC

Critère : D-Optimalité  
 Nombre d'expériences : 8  
 Modèle : Atkinson-Bogacka.dlm / Box\_Lucas\_Etendu

#### Plan

t  
 7.7  
 2.4  
 0.5  
 20  
 0.6  
 20  
 7.7  
 2.4

#### Diagnostics obtenus par développement de Taylor du modèle

#### I/ Résumé des estimations paramétriques

Paramètre	Valeur	Biais	Ecart type	IC_Min
Teta1	0.7	0.0276306635293975	0.161603225881505	0.25131751451712
Teta2	0.2	0.0618254260860576	0.142348126361084	0
Lambda1	2	0.0296752009713439	0.593251283382924	1.01
Lambda2	2	0.0920845109990299	0.794714772015645	1.01

#### III/ Matrice de variance-covariance de l'estimateur des moindres carrés des param

Paramètre	Matrice de variance-covariance			
Teta1	0.0261156026153088			
Teta2	-0.0123023275886269	0.0202629890785113		
Lambda1	0.0833284156530533	-0.0554976557420797	0.351947085235487	
Lambda2	-0.0602761319033328	0.111149208210308	-0.272719818572836	0.631

Déterminant = 4.30159973564912E-007  
 Trace = 1.02989724578919  
 Valeur propre max = 0.828860241369913  
 Conditionnement = 0.000538052039138927

#### III/ Matrice de corrélation de l'estimateur des moindres carrés des paramètres

Paramètre	Matrice de corrélation			
Teta1	1			
Teta2	-0.534792744200987	1		
Lambda1	0.869169368387548	-0.657179786625459	1	
Lambda2	-0.469336211212113	0.982524372799554	-0.578451201156315	1

Corrélation maximum = 0.982524372799554  
 Corrélation moyenne = 0.706349813779924

#### IV/ Mesures de la D-efficacité du plan

D-Efficacité = 99.9949628109253 %  
 Déterminant de la matrice d'information normée = 567.557745962987

#### V/ Mesures de non-linéarité

Courbure intrinsèque maximale = 0.047672904937013  
 Courbure intrinsèque moyenne = 0.0160239416728005  
 Courbure paramétrique maximale = 9.18838451688847  
 Courbure paramétrique moyenne = 3.24502439283992  
 Courbure de référence au seuil 0.05 = 0.395648596200289

#### VI/ Performances prédictives

<REQM> Prédiction = 0.0263164315040826  
 Max REQM Prédiction = 0.0289739125709929

#### Volume de la région de confiance

Volume région de confiance asymptotique = 2.11332462793907

# NEURO PEX

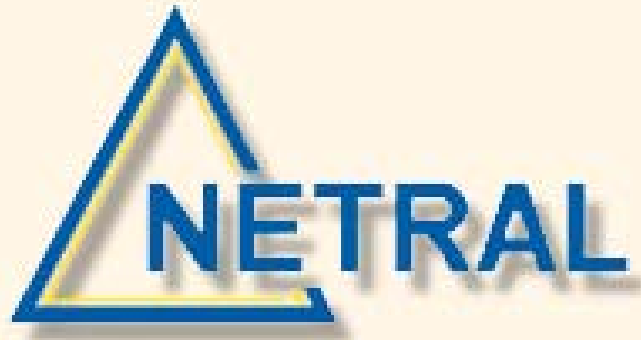


## Conclusion

**Neuro Pex est le premier logiciel commercial au monde pour le calcul de plans d'expériences pour modèles non-linéaires.**

**Il est complet et très simple d'emploi.**

**Il autorise le calcul de plans pour modèles de connaissance, réseaux de neurones et EDO !!**



**Merci de votre attention**

**[patrice.kiener@netral.com](mailto:patrice.kiener@netral.com)**

**Tel : 01.46.38.75.12**