Environment for Solving Inverse and Optimization Problems

- **Algorithm development**
- **Outline of Optimization Environment**
- **Inclusion of Artificial Neural Networks-based Approximations**

Igor Grešovnik
Feb 22, 2011

This is presentation of plans for optimization environment developed in COBIK, which will also be used to support the UNG-StoreSteel project.

Optimization Problems – Formulation

minimise $f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n$

subject to $c_i(\mathbf{x}) \leq 0, \; i \in I$

and $c_j(\mathbf{x}) = 0, \; j \in E$,

where $l_k \leq x_k \leq u_k, \; k = 1, 2, ..., n$.

Use of Optimization

Industrial use of simulations:

- **Improvement of Current Processes & Designs**
    - Virtual Prototyping

Development of numerical models:

- **Experimental Validation**
    - Inverse identification of model parameters

Inverse Identification of Model Parameters

Concept:

- **Perform laboratory & industrial measurements**
- **Prepare numerical models of these tests, some parameters unspecified**
- **Minimization of discrepancies between measurements and model results -> parameter estimates**

Numerical difficulties:

- **Long computational times**
- **Noise in model results**
- **Non-availability of derivatives w.r. parameters**

Engineering problems:

- **Complexity of industrial systems** (variability of process conditions, complex interactions)
- **Limited set of affordable measurements**
  - Insufficiency of data for solution of identification problem

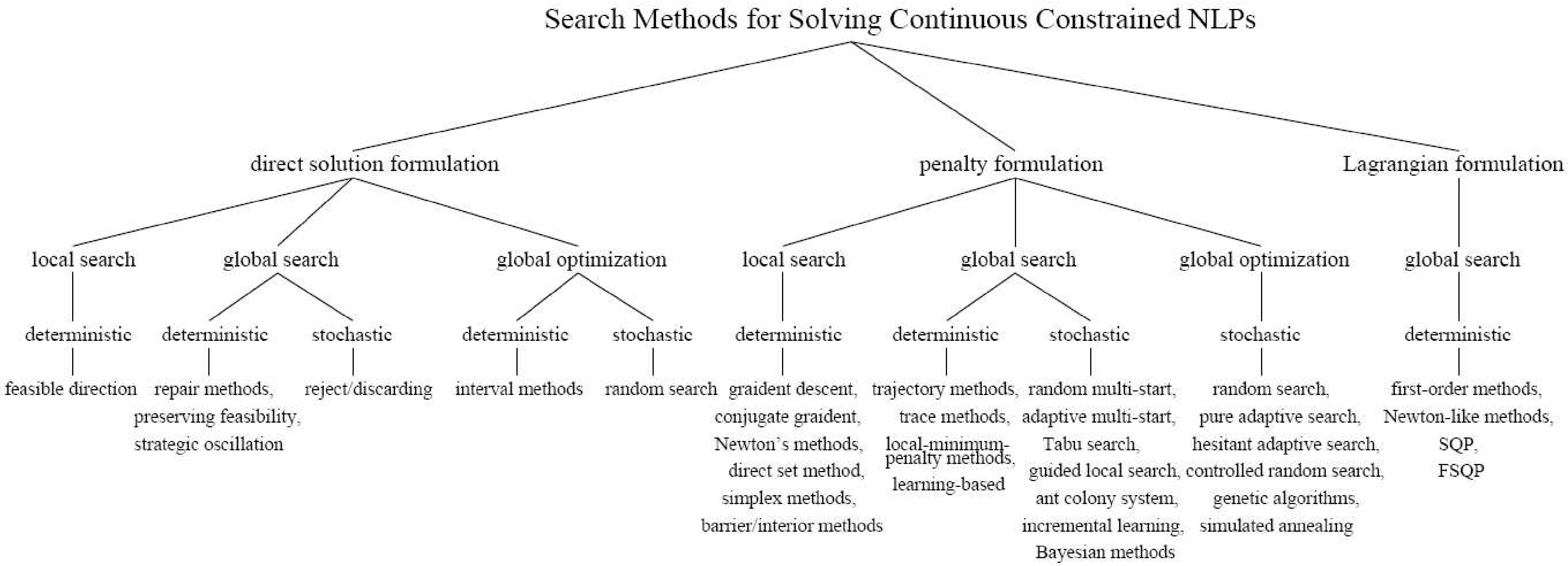
Efficient Optimization Algorithms for Constrained Problems

- **Transformation to unconstrained problems**
  - Penalty methods
  - Lagrangean methods (eliminate constraints by Lagrange multipliers)
  - Projection methods (feasible set)
- **SQP**
  - Newton method for 1st order conditions for a local minimum
  - QP subproblem
  - Feasible set method
  - Sequentially solve unconstrained problems by BFGS
  - Superilinear convergence

Popular Optimization Algorithms: Global Optimization

- **Predominantly stochastic approaches**
    - Simulated annealing, genetic algorithms, particle swarm method, differential evolution
- **Robust in terms of response requirements**
    - Perform on discontinuous, multimodal functions
- **Inefficient in terms of required number of function evaluations**
- **Notion of "global algorithm" is asymptotic**
    - When number of iterations goes to **infinity**, the **probability** of "missing" the neighbourhood of a global optimum tends to 0
- **Even if not "global" in practical sense, these methods less easily get stuck in a local minimum**
- **Notion of local convergence is difficult to define**
- **Incorporate heuristics often taken from nature**
    - Biological evolution, swarm intelligence, statistical mechanics, self-organization of dynamical systems
- **Performance typically increased by tuning a number of control parameters (case dependent)**

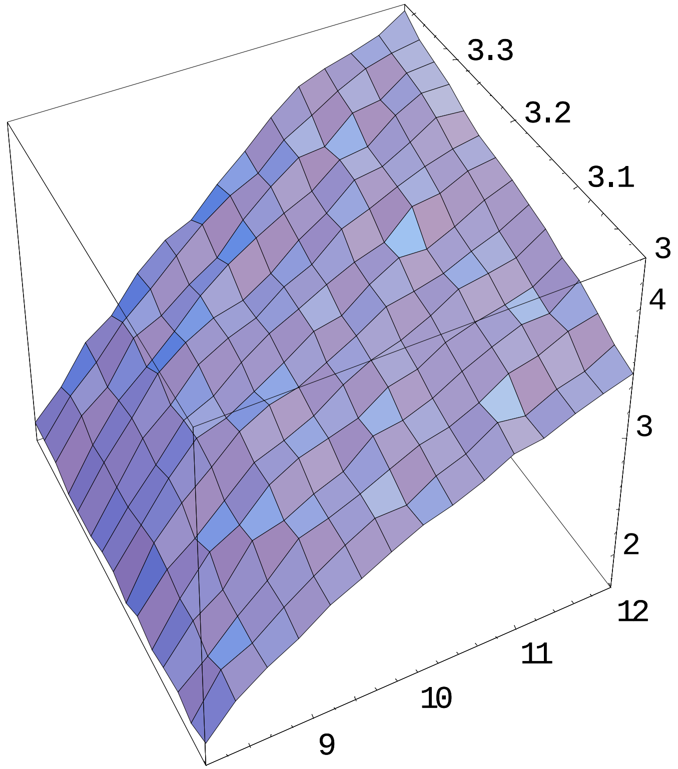
Classification of Optimization Algorithms



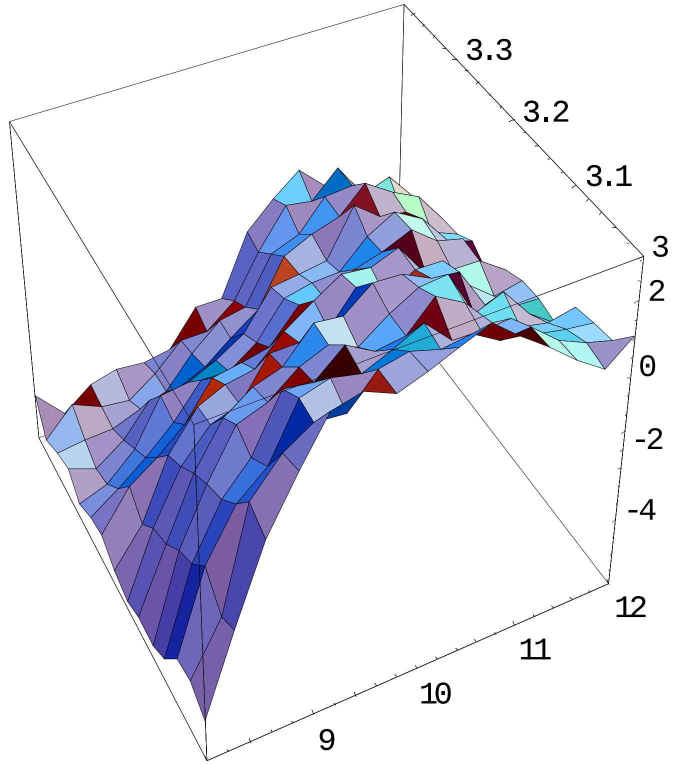
Response functions with noise

a) objective function

b) constraint function

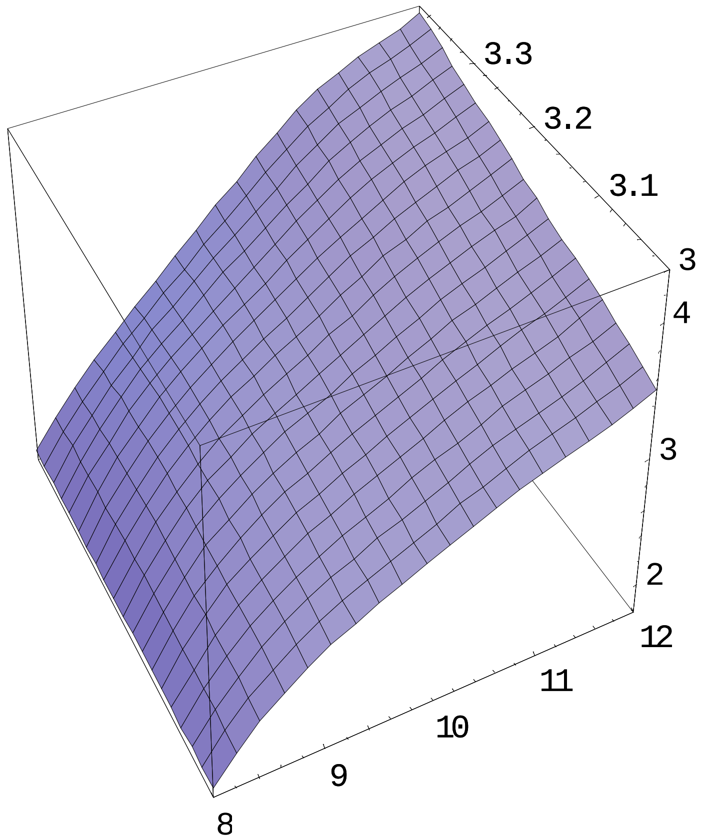


a)

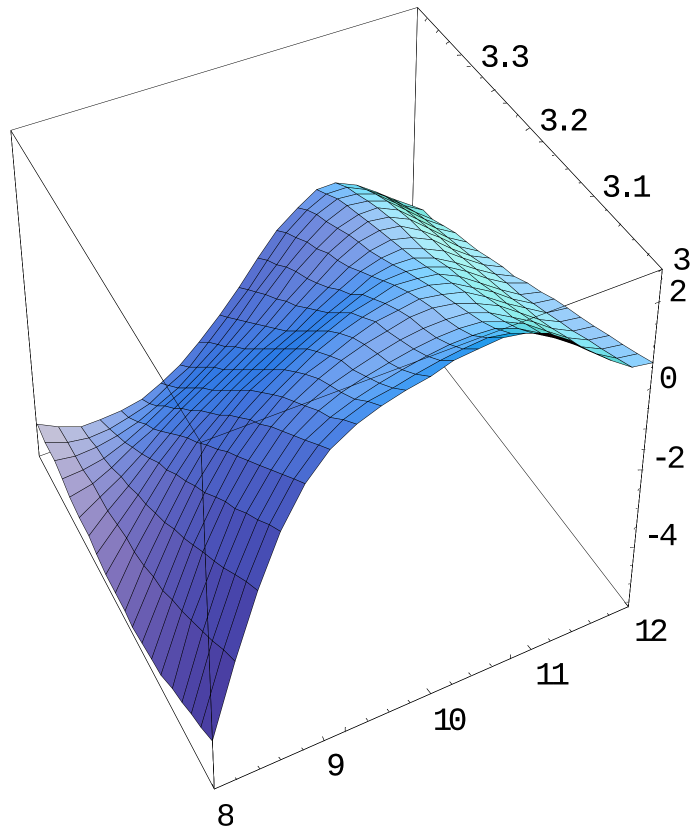


b)

Response Smoothing (Global, MLS Approximation)

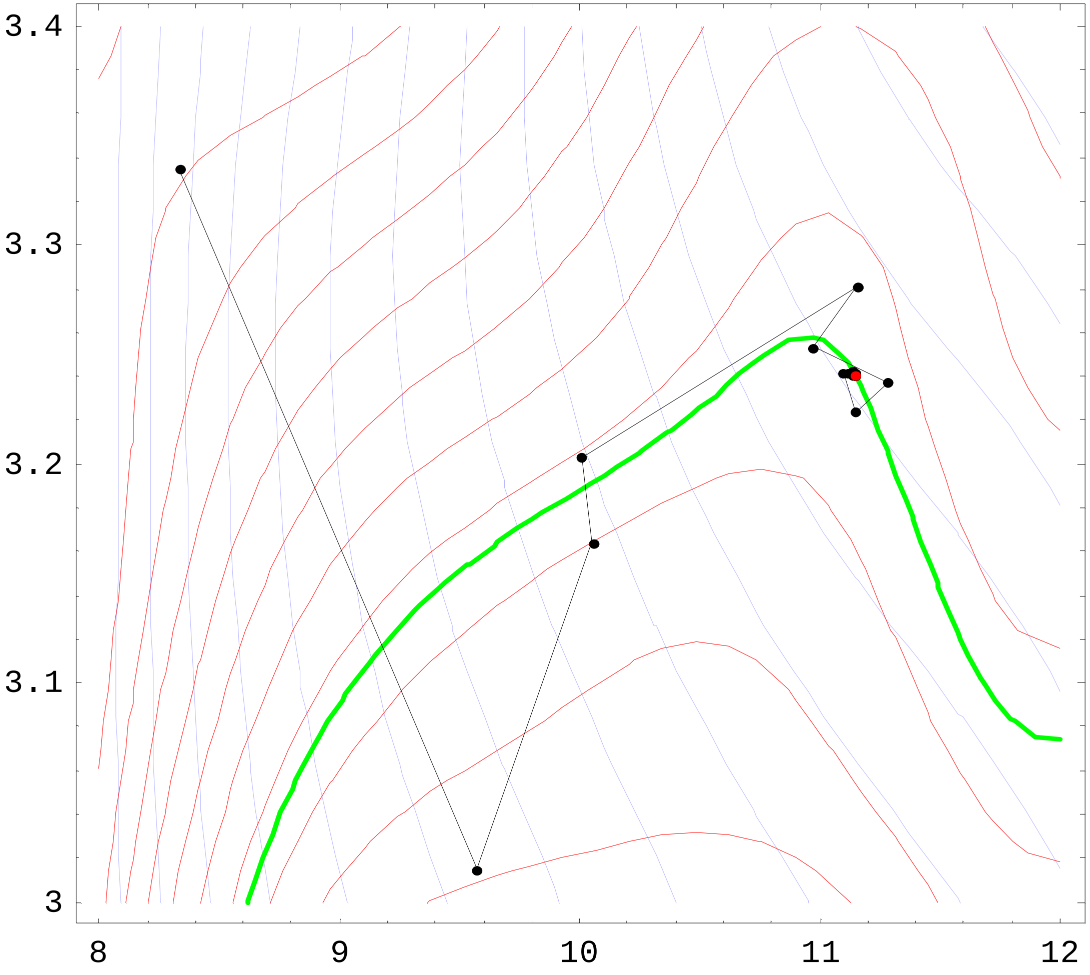


a)

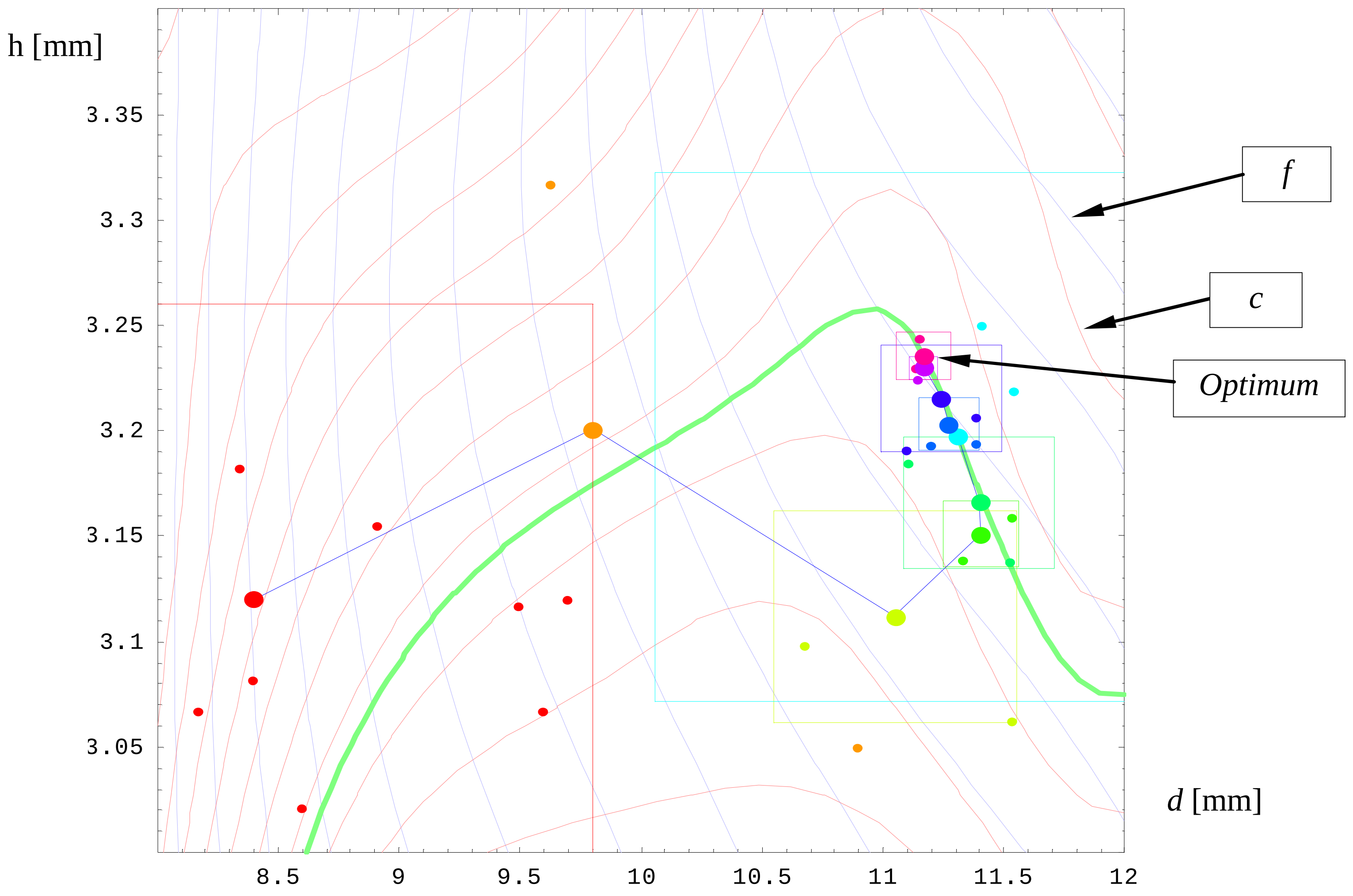


b)

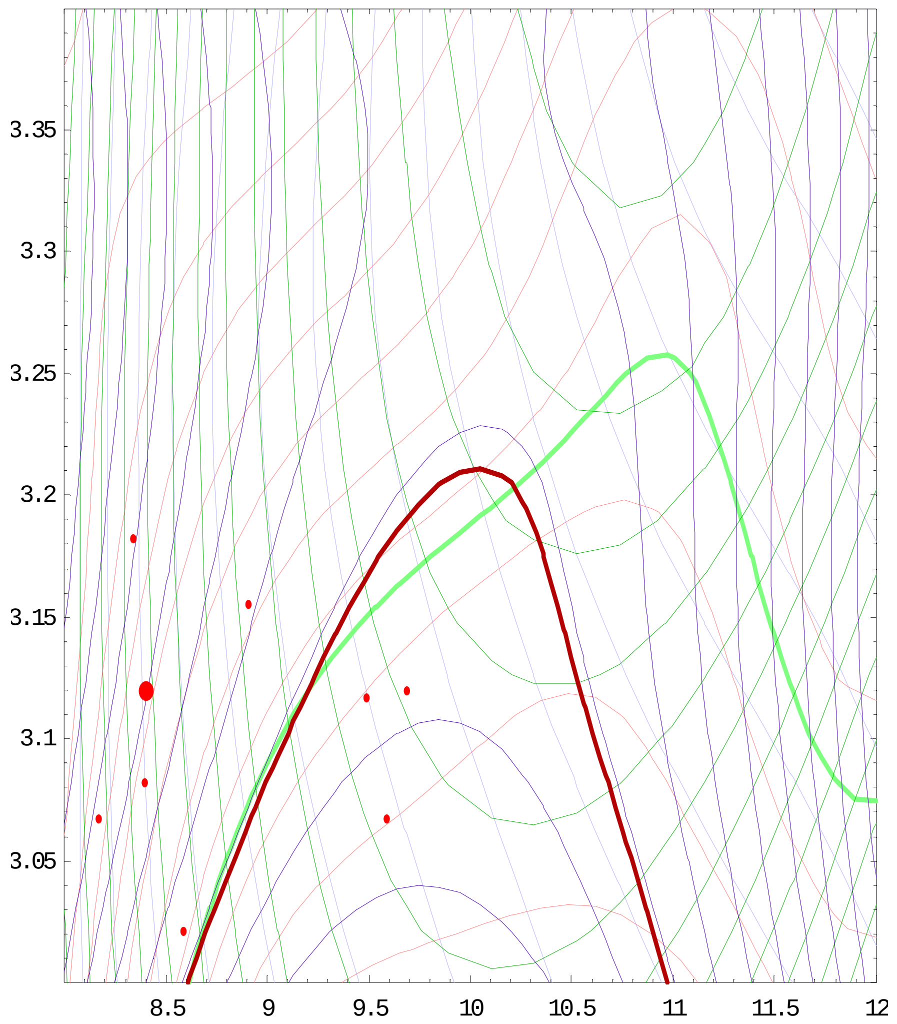
Optimization of Approximated Response (SQP)



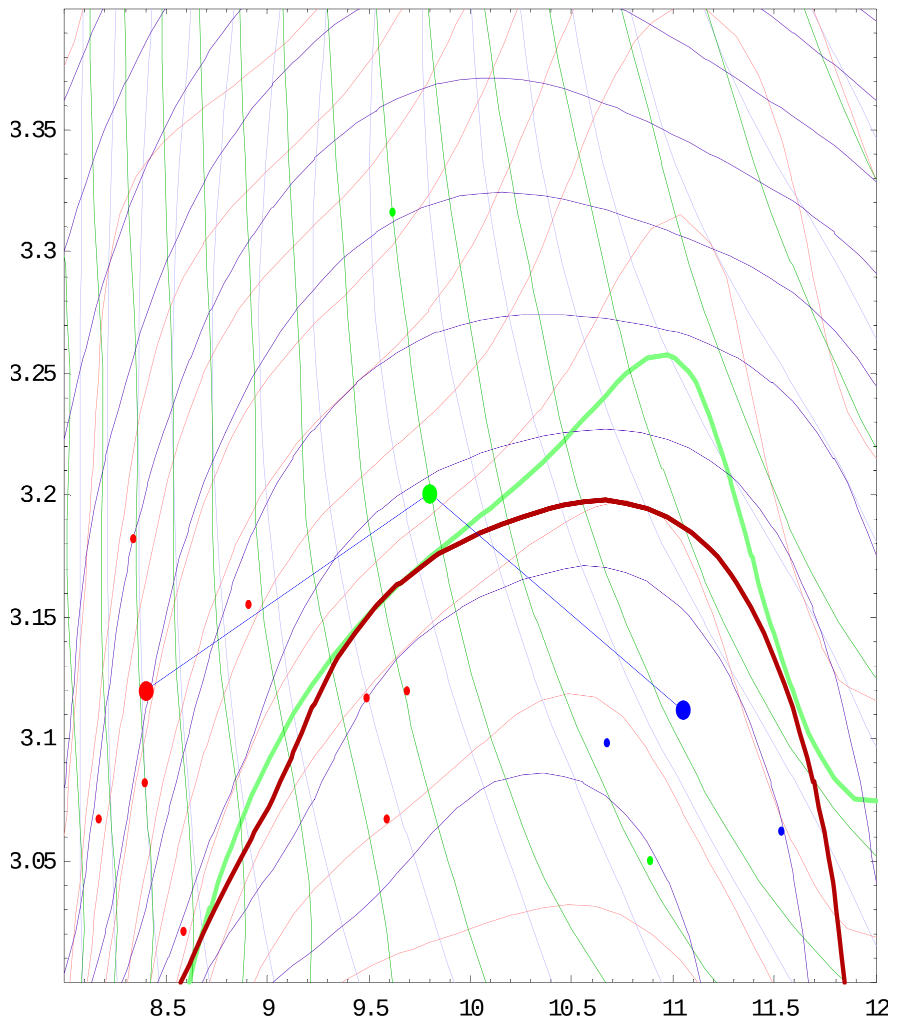
Feasible Alternative: Optimization Based on Successive MLS Approximations of Response Functions with Restricted Step Approach



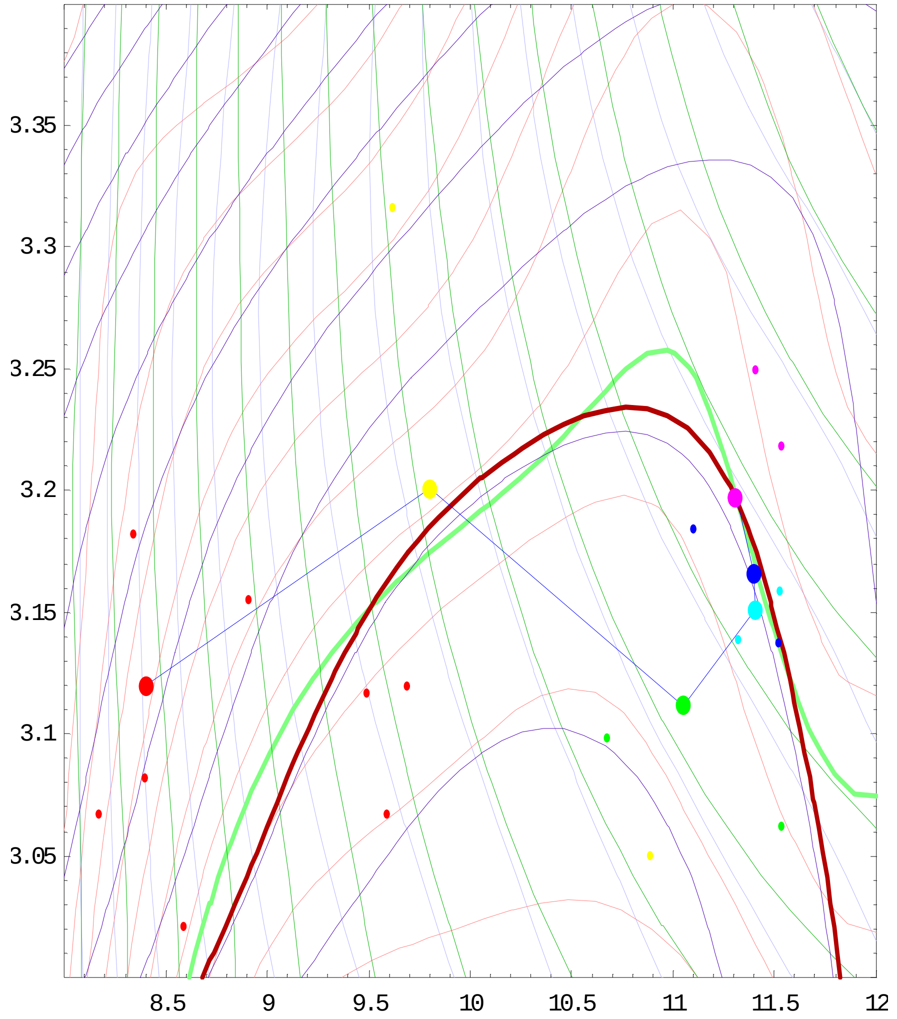
Approximated response for iteration 1



Approximated response for iteration 3

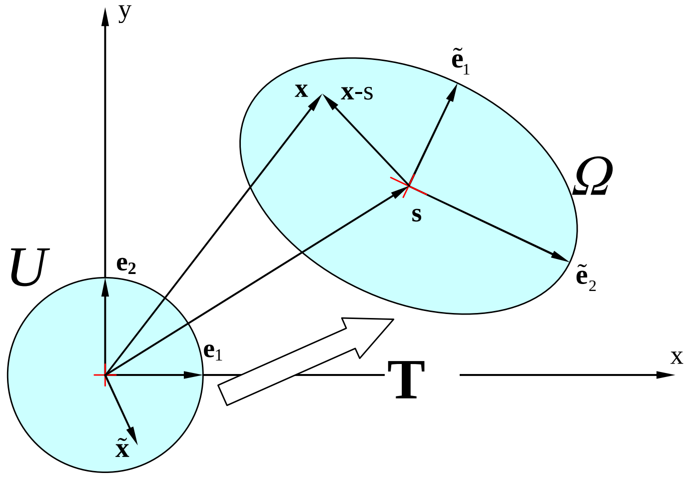


Approximated response for iteration 6

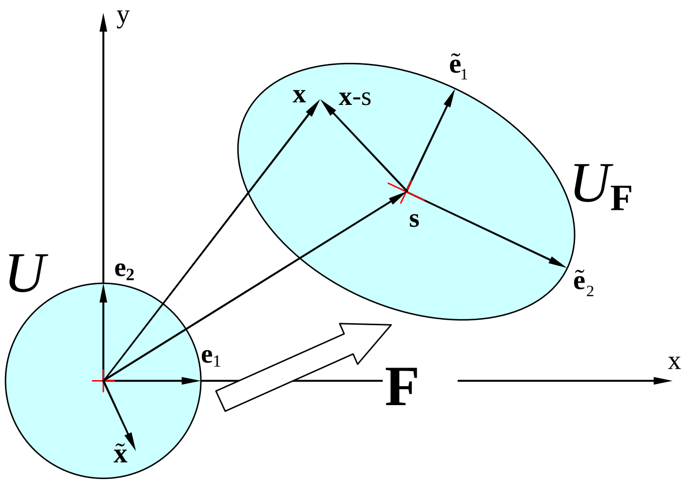


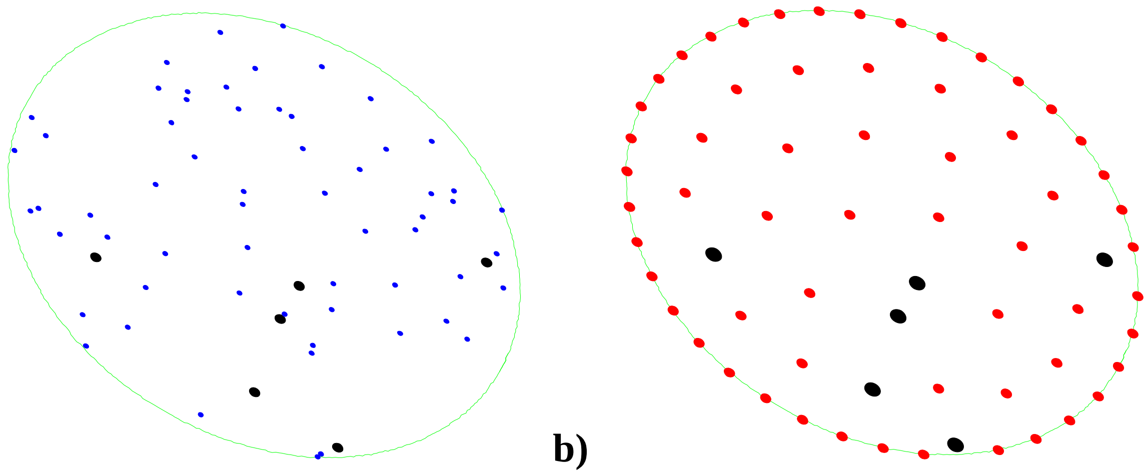
Building Blocks: Restricted Step Constraint

- **Restricts the feasible region to a neighborhood of the current guess**
- **Unit ball constraint:**
  - $c_U(\tilde{\mathbf{x}}) = \|\mathbf{x}\|_2 - 1 \leq 0$
- **General form:**
  - Obtained from unit ball constraint by affine transformation of co-ordinates:
  - $c_{rs}(\mathbf{x}) = \left\| \tilde{\mathbf{A}}^{-1}(\mathbf{x} - \mathbf{x}_0) \right\|_2 - 1 \leq 0$



Design of Experiments

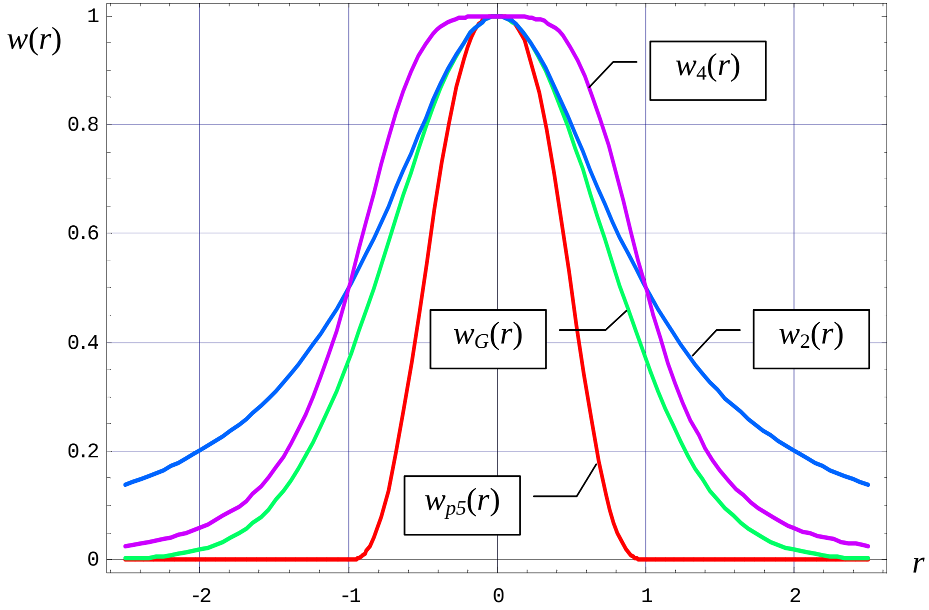




a) b)

Building blocks: Weighting Functions for Approximations

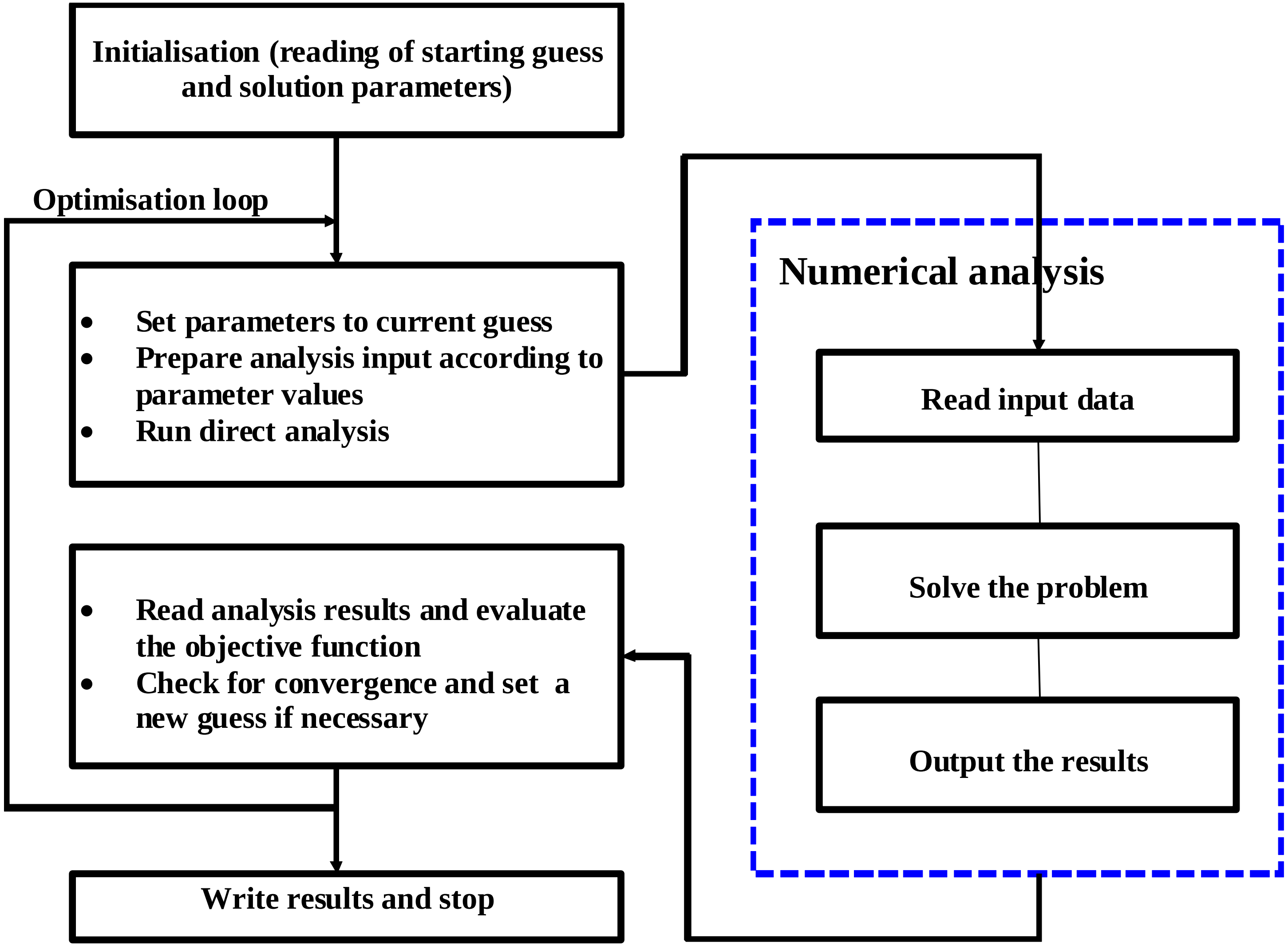
- **Different forms of 1D weighting functions** $w(r)$
- **Multivariate weighting functions obtained by affine maps:**



Integration in algorithm scheme

- **To build adaptive approximations of response functions**
- **To solve approximated sub-problem with restricted step constraint**

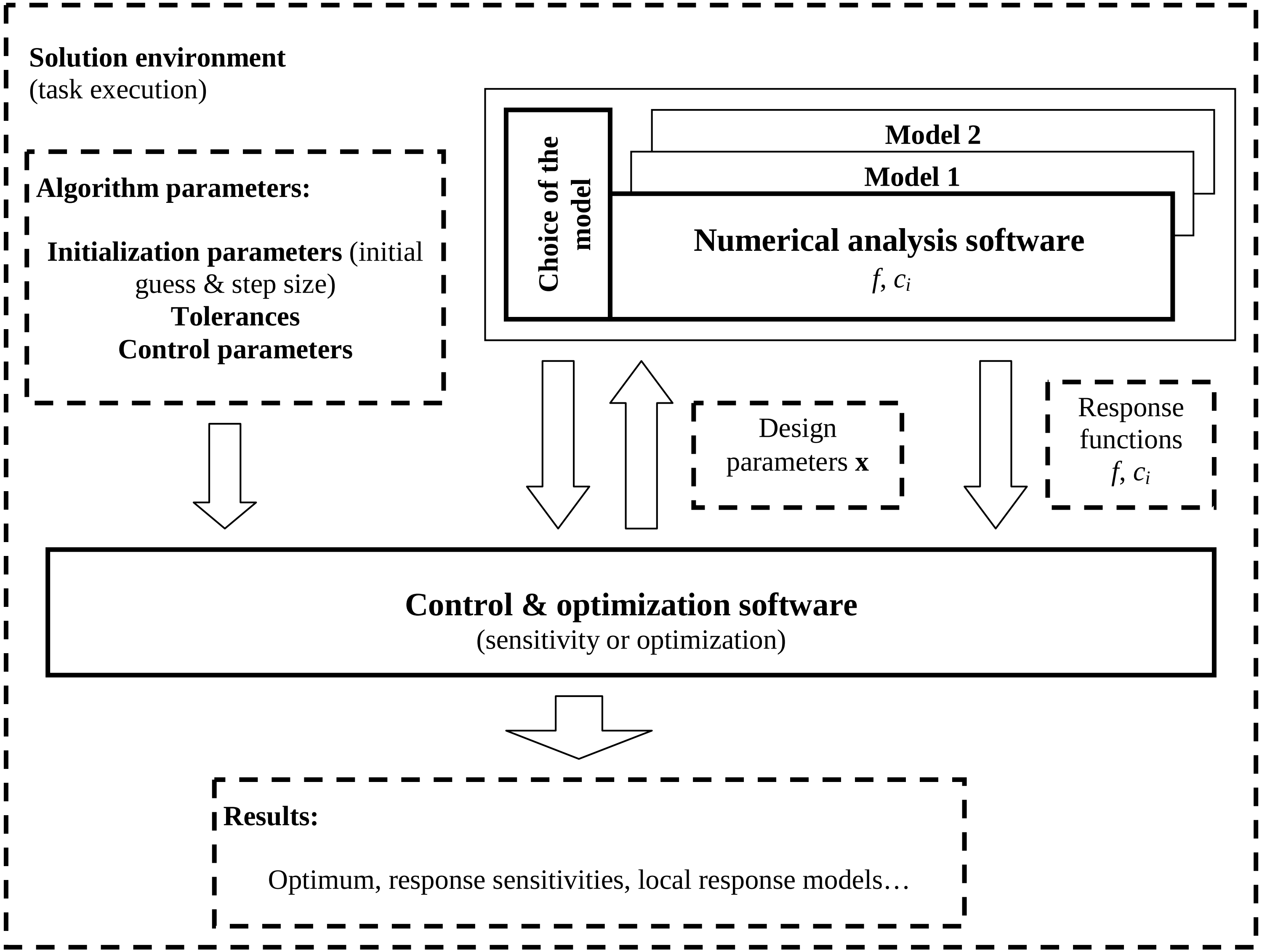
Optimization Problems – Solution Scheme



Optimization Problems – Solution Scheme

1. Take current optimization parameters
2. Prepare numerical model according to parameters
3. Run numerical simulation of the process
4. Extract the relevant quantities from simulation results
5. From measured data
   - Read result file
   - Extract relevant data
6. Calculate the response functions and eventually their gradients (in our case the discrepancy function *f*)
7. Store the response functions in output arguments and return

Integrated Optimization Platform



File Format for Data Exchange

Analysis input file:

```
{ { p1, p2, … }, { reqcalcobj, reqcalcconstr, reqcalcgradobj,
      reqcalcgradconstr }, cd }
```

Analysis output file:

```
{
  { p1, p2 ... },
  {
    calcobj, obj,
    calcconstr, { constr1, constr2, ... },
    calcgradobj, { dobjdp1, dobjdp2, ... },
    calcgradconstr,
    {
      { dconstr1dp1, dconstr1dp2, ... },
      { dconstr2dp1, dconstr2dp2, ... },
      ...
    },
    errorcode
  },
  { reqcalcobj, reqcalcconstr, reqcalcgradobj, reqcalcgradconstr }
  < , { ind1, ind2, ... }, { coef1, coef2, ... }, defdata >
}
```

Neural Networks: Response Approximation

- **Provides approximate relation between process parameters and outcomes**

Π*:*

π_1
π_2
π_3
π_4
…
…
…
…
…
$\pi_{N\pi}$

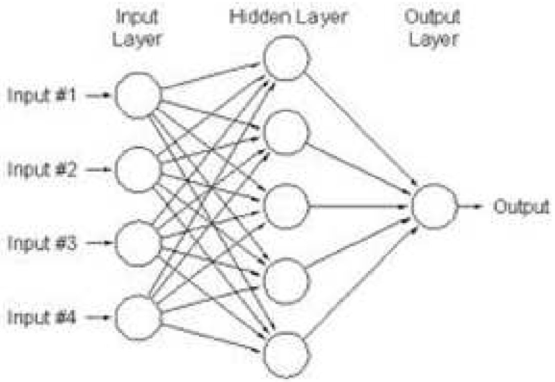
Trained neural network

(known layout and known weights w_i)

Ω*:*

ω_1
ω_2
…
…
…
$\omega_{N\omega}$

Neural Networks: Training



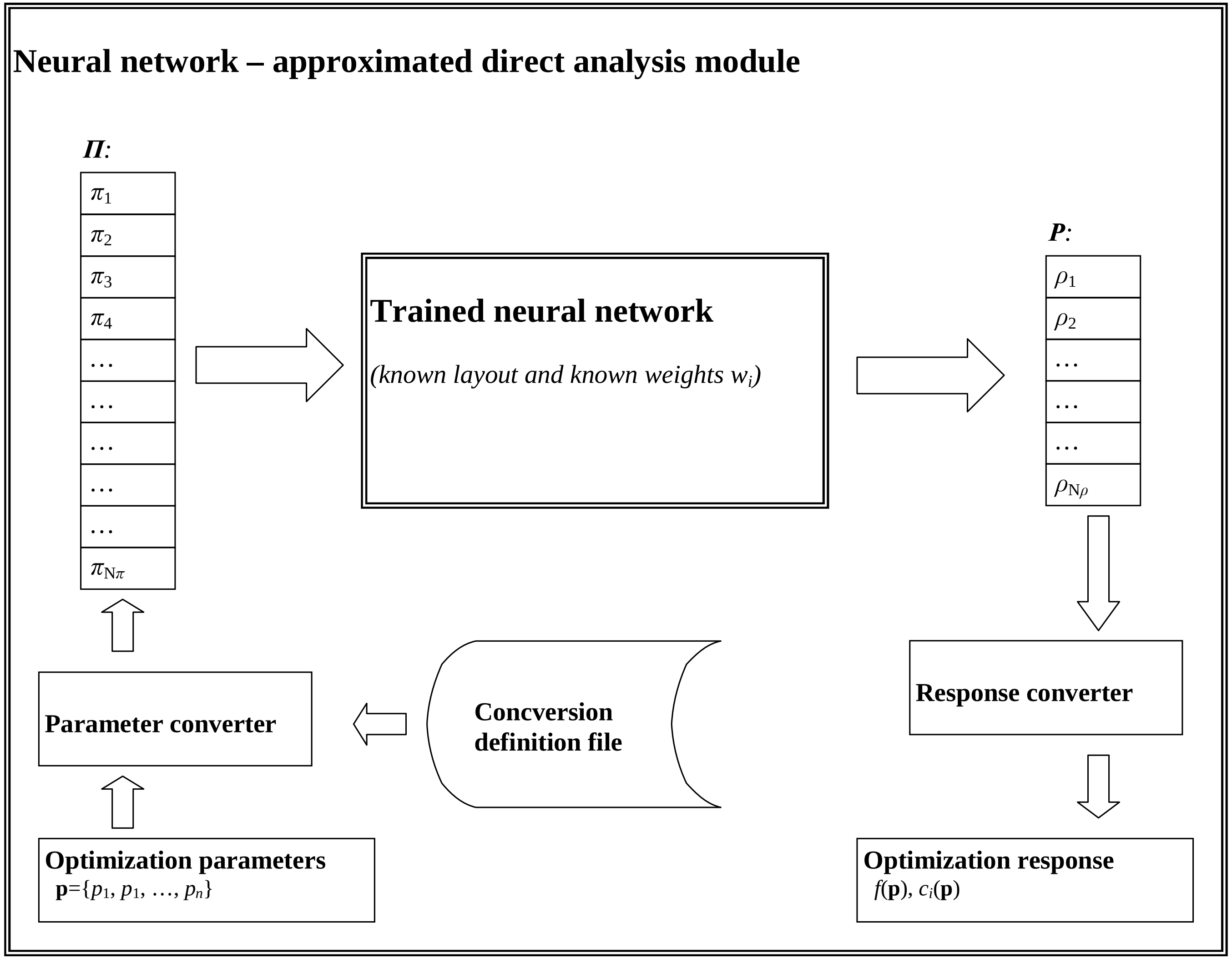
Training data

($\mathbf{\Pi}_1$, $\mathbf{\Omega}_1$)
($\mathbf{\Pi}_2$, $\mathbf{\Omega}_2$)
($\mathbf{\Pi}_3$, $\mathbf{\Omega}_3$)
($\mathbf{\Pi}_4$, $\mathbf{\Omega}_4$)
…
…
…
…
…
($\mathbf{\Pi}_5$, $\mathbf{\Omega}_5$)

Trained neural network

(known layout and known weights w_i)

Neural Networks: Direct Analysis Surrogate



Neural Networks: Direct Analysis Surrogate

Conversion definition file

***Π**:*	***active flag**:*	***Corresponding opt. parameter**:*	***Default value:**:*
π_1	*yes*	*1*	*3.56*
π_2	*no*	*0*	*1.2e7*
π_3	*no*	*0*	*109.3*
π_4	*yes*	*2*	*24.5*
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
…	…	…	…
$\pi_{N\pi}$	*yes*	*10*	*1.53e-3*

To Discuss:

- **Which data to take from process' databases**
  - Quality of data to be considered:
    - There must be enough measurements
    - Distribution of samples must cover parameter space well
    - There must be no hidden parameters (parameters that vary over provided data, but are not included in data sets)

- **Technicalities**
  - Procedures for gathering data
  - Formats of data

Optimization Environment – State & Plans:

IGLib – Investigative Generic Library

- **Origins: *IOptLib* (Investigative Optimization Library), *Inverse* (optimization framework)**
- **Purpose: base library for technical applications**
  - Emphasis on numerical modeling and optimization
- **Motivation: use in personal projects to speed up development process**
- **What is it:**
  - A set of software concepts for solution of various technical problems
  - A set of tools
  - Modular & extensible
  - Carefully structured, based on experience
  - Based on a number of external libraries

IGLib: Some Existent Tools

- **Application framework**
  - Initialization, directory structures, etc.
  - Error reporting, Notifications, Event logging
  - Export of application state & settings
- **Data exchange:**
  - Parsers (general text files, XML files)
  - Generic data storage/retrieval techniques (persistent objects)
  - Inter-process communication
- **Numerics:**
  - Linear algebra, FFT, interpolations, integration, differentiation,
  - Error reporting, Notifications, Event logging
- **Optimization:**
  - Various standards (e.g. standard form of direct analysis, response differentiation, modified response such as penalty formulations, parallel jobs)
  - Basic utilities for response approximation techniques (need to be extended)
  - Dragonfly Optimization Server

IGLib: To implement

- **Algorithmic support (optimization)**
  - Line searches
  - Convergence tests in difficult conditions (noise, etc.)
  - Improved & generalized dispatchers (distributed optimization)
  - Extend test environment
  - Extend algorithmic base (SQP, evolutionary, etc.)
  - Additional standardization of analysis & optimization servers
  - Error estimations – approximation + optimization
  - Implementation of iteration schemes in approximation-based optimization
  - Reliability based optimization
- **Environment**
  - Supplement standard interaction methods (analysis ⇔ optimization)
  - Improve application framework, including reporting & logging
  - Additional standards for interaction with simulation & control
  - Application server (multitier architecture, remote control, etc.)

http://dl.dropbox.com/u/12702901/code_documentation/generated/develop/html/index.html