PSR

Stochastic Optimization Models on Power Systems

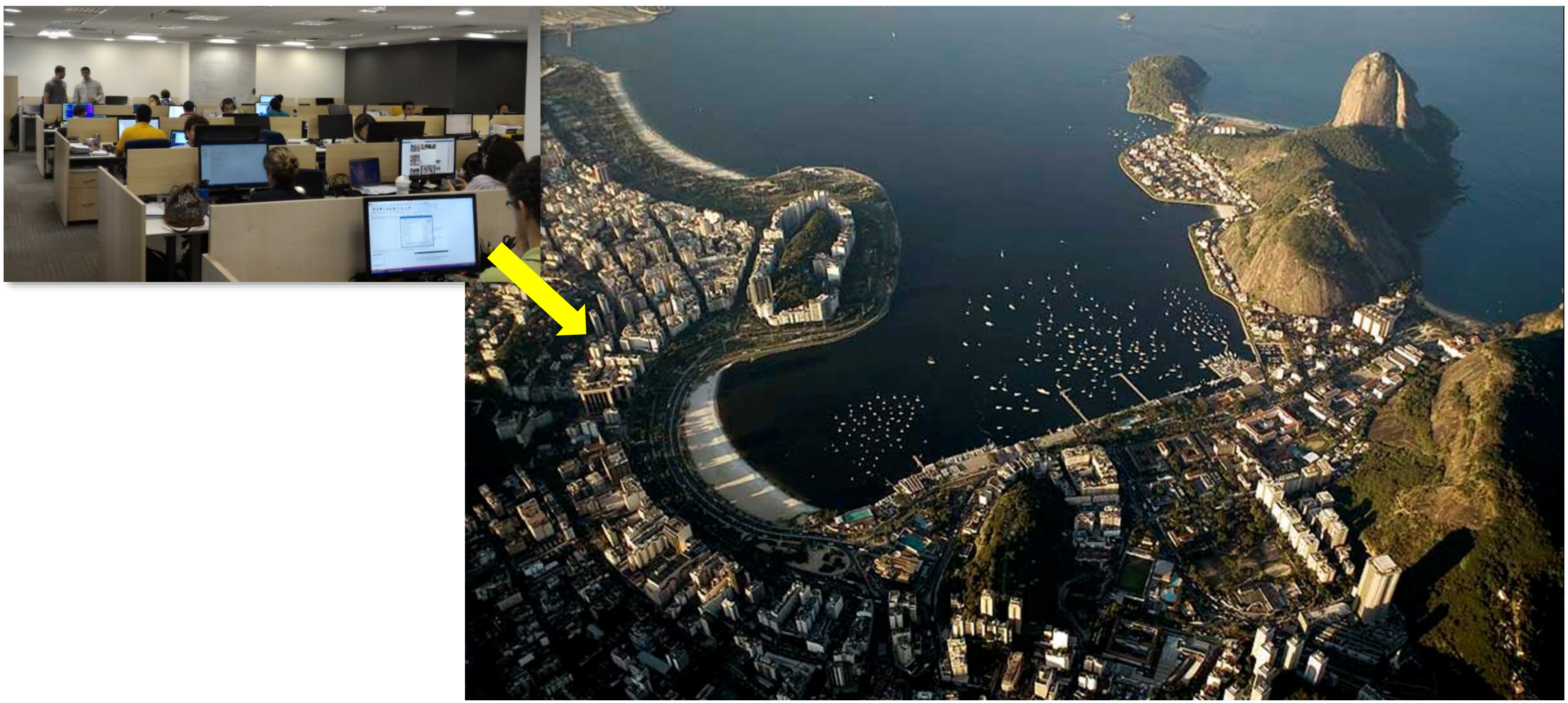
JuliaCon

Berkeley, June 22, 2017

PSR

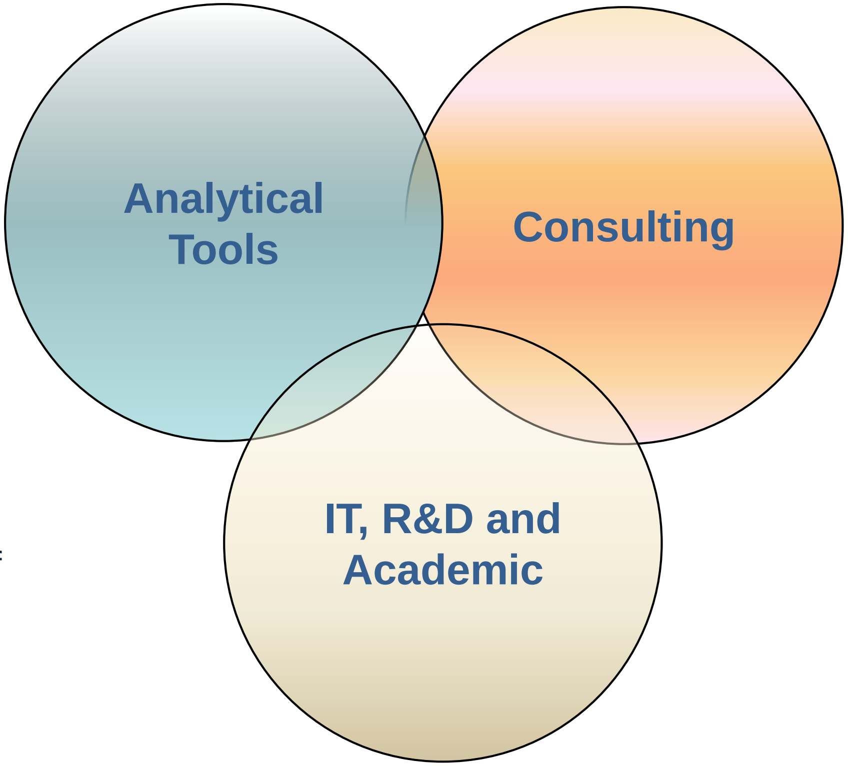
Provider of analytical solutions and consulting services in electricity and natural gas since 1987

Our team has more than 60 experts (17 PhDs, 31 MSc) in engineering, optimization, energy systems, statistics, finance, regulation, IT and environment analysis

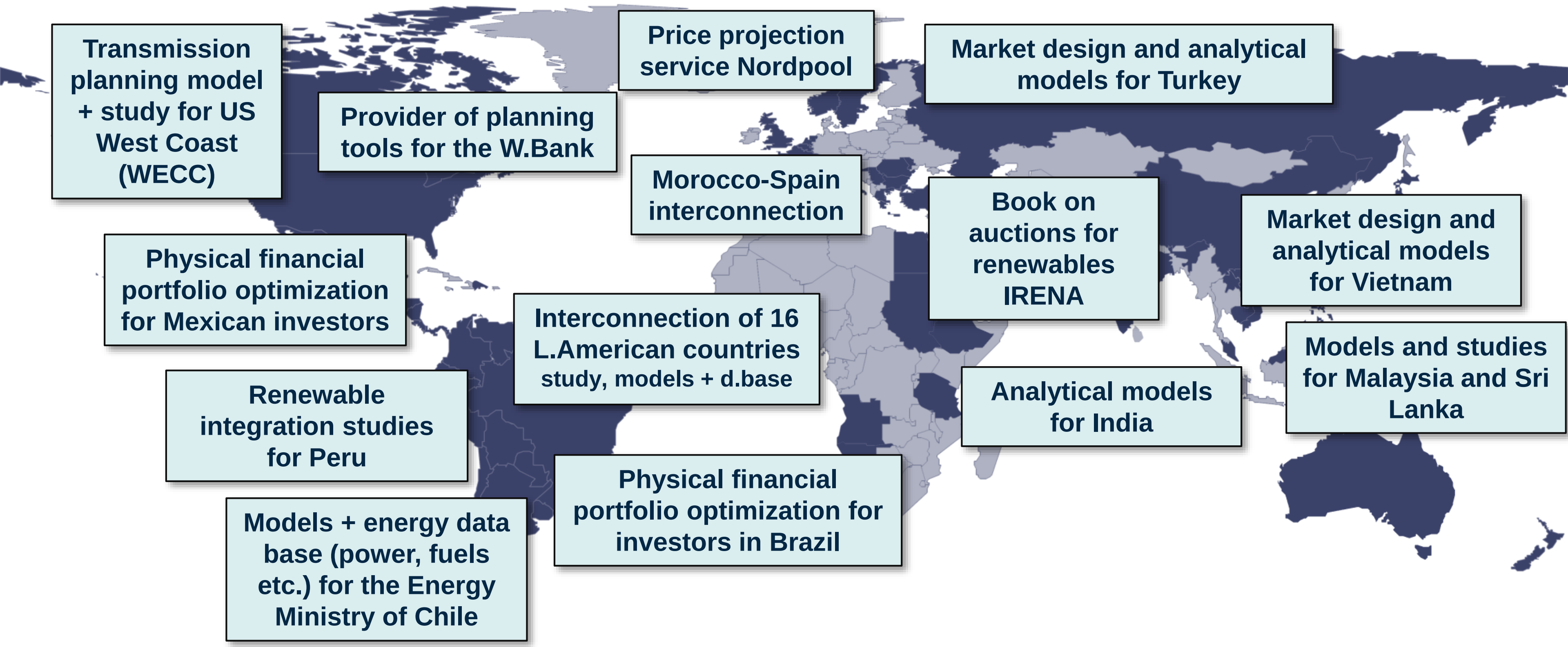


Core activities

- Development of analytical tools
  - integrated economic + technical + environmental evaluation of energy investment opportunities
- Strategic advice
  - to investors, lenders and governments (economic, financial, regulatory, risk assessment)
- Market & regulatory design
  - (e.g. organization of auctions, energy integration of Latin American countries, etc)
- Environmental analysis
  - (e.g. integrated river basin development, CO_2 emissions, due diligence, etc)
- Engineering studies
  - (e.g. generation & transmission planning, automated river-basin inventory, etc)



Some recent projects



- **Americas**: all countries in South and Central America, United States, Canada and Dominican Republic
- **Europe**: Austria, Spain, France, Norway (Nordic region), Belgium, Turkey and the Balkan region
- **Asia**: provinces in China (including Shanghai, Sichuan, Guangdong and Shandong), Philippines, Singapore, Malaysia, Kirgizstan, Sri Lanka, Tajikistan and India
- **Oceania**: New Zealand
- **Africa**: Tanzania, Namibia, Egypt, Angola, Sudan, Ethiopia and Ghana
- **More than 300 active licenses worldwide**

Quick Introduction

- Researcher and developer at PSR since 2013
- Industrial Engineer (PUC-Rio)
- MSc in Electrical Engineering -Decision support tools (PUC-Rio)
- Works in the development of the models of optimization of hydrothermal dispatch under uncertainty with network constraints (SDDP model) and electric systems expansion planning (OPTGEN model).
- Programs in: Julia, Python, R, MATLAB, Mosel, Fortran, C/C++

Research

- Optgen (SDDiP)
- Reliability analysis
- Optflow
  - **Progressive Hedging Applied to the Problem of Expansion Planning of Reactive Power Support Equipment** (online but in Portuguese)
  - Conic Models -> Decomposition
- Immediate cost function
  - **Analytical representation of immediate cost functions in SDDP** (online)
  - Pre computation of hyperplane representation (https://github.com/JuliaPolyhedra/Polyhedra.jl)
- ICSP:
  - **Representation of uncertainties in fuel cost and load growth in SDDP-based hydrothermal scheduling** (online)
  - **Modelling power markets with multi-stage stochastic Nash equilibria** (online)

Main planning and scheduling tools

- **SDDP**: optimal mid and long-term stochastic production scheduling
- **OPTGEN**: optimal mid and long term capacity expansion planning
- **OPTFLOW:** optimal power flow for reactive expansion
- All models optimize complex hydrothermal systems with renewable generation, transmission networks, price-responsive loads, gas pipelines and fuel storage

The beginning: Testing new ideas

- Optfolio: Portfolio optimization
  - Initially JuMP+CBC, first motivation for Xpress.jl
- Power systems reliability analysis
  - Monte Carlo methods + network flows + Fourier analysis
- Decomposition of integer problems for network expansion models
  - SDDiP

The beginning: Testing new ideas

- Op
  -
- Po
  -
- De … dels
  -

```
m = Model()

@defVar(m,            Alpha[l in 1:n.Openings] >=0)                                   # $
@defVar(m, Vmin[h] <= Vf[h in 1:n.Hydros] <= Vmax[h] )                                 # hm3
@defVar(m, 0       <= turbining[b in 1:n.Blocks, h in 1:n.Hydros] <= Umax[b,h])        # hm3
@defVar(m,            spillage[b in 1:n.Blocks, h in 1:n.Hydros] >=0 )                  # hm3
@defVar(m, 0       <= gT[b in 1:n.Blocks, j in 1:n.Thermals] <= potThermal[j]*duraci[t,b] ) # MWh = (MW*h)
@defVar(m, a_tmp[p in 1:max(n.ARparam,1), h in 1:n.Hydros] == inflow[t+n.ARparam+1-p,s,h] ) # m3/s

#depender do numero de lags do periodo?
@defVar(m, a_next[h in 1:n.Hydros, l in 1:n.Openings])                                 # m3/s

@addConstraint(m, HidroBalance[h in 1:n.Hydros],
Vf[h] == V0[t,s,h]  + sum{ (3600*duraci[t,b]*m3tohm3)*a_tmp[1,h] - turbining[b,h] - spillage[b,h], b in 1:n.Blocks}
+ sum{ turbining[b,j], j in MT[h], b in 1:n.Blocks }
+ sum{ spillage[b,j], j in MS[h], b in 1:n.Blocks }
)

@addConstraint(m, LoadSupply[b in 1:n.Blocks],
sum{(rho[h]/(m3tohm3*3600))*turbining[b,h], h in 1:n.Hydros}+ sum{gT[b,j], j in 1:n.Thermals} == dem[t,s,b]*1000
)#(rho[h]/(m3tohm3*3600)) = MWh/hm3 = (MW/m3/s*hm3/m3*h/s)

if t < n.Stages && Cuts[t,s] > 0
    #if nAR[per(t+1)] > 0 #remover esse condicional???
        @addConstraint(m, FCFcstr[l in 1:n.Openings, c in 1:Cuts[t,s]   , s in 1:n.Scenarios ],
        Alpha[l] >= delta[t,s,c] + sum{ piH[t,s,c,h]*Vf[h] , h in 1:n.Hydros} +
        sum{ piA[t,s,c,h,1]*a_next[h,l] , h in 1:n.Hydros; nAR[h,per(t+1)]>0  } +
        sum{ piA[t,s,c,h,p]*a_tmp[p-1,h], h in 1:n.Hydros, p in 2:nAR[h,per(t+1)]}
        )
        @addConstraint(m, InflowModel[ h in 1:n.Hydros, l in 1:n.Openings],
        a_next[h,l] == sum{phi[h,per(t+1),p]*a_tmp[p,h], p in 1:nAR[h,per(t+1)]}  + err[t+1,s,h,l]
        )
end


@setObjective(m, :Min,
sum{ cT[t,s,j]*gT[b,j], j in 1:n.Thermals, b in 1:n.Blocks}
    + (1/(1+tx_per))*(1/n.Openings)*sum{Alpha[l], l in 1:n.Openings}
```

Julia from research to market

- SDDP + Optgen
  - Peru energy ministry
  - Chile
- Portfolio Optimization (The Nature Conservancy - TNC)
  - Hydro expansion in Magdalena Basin in Colombia
  - https://global.nature.org/content/power-of-rivers?src=r.v_powerofrivers

Julia from research to market

- SDDP + Optgen
  - Peru energy ministry
  - Chile
- Portfolio Optimization (The Nature Conservancy - TNC)
  - Hydro expansion in Magdalena Basin in Colombia
  - https://global.nature.org/content/power-of-rivers?src=r.v_powerofrivers

Julia from research to market

- Evaluating Latin American Interconnections
  - For BID (Inter-American Development Bank)
- Refactoring Genesys from NWPCC
  - Large scale reliability evaluation model

Quick Introduction

- Graduated in Telecommunications and Mathematics
- Currently PhD candidate at PUC-Rio
- Researcher and developer at PSR since 2015
- Julia developer
  - Just arrived from US started internship in PSR (2015)
  - Started with C but had many hours of MATLAB and R
  - First day: A maintenance scheduling model
    - Mario: “Do you know julia?”
    - julia+JuMP X Mosel

SDDP – Power system operation modeling

- Physical parameters
  - Hydro (detailed topology (cascades), hydro production, reservoirs modeling, operative constraints etc.)
  - Thermal (efficiency curves, combined cycle plants, multiple fuel plants, fuel availability constraints, GHG emission factors, unit commitment decisions etc.)
  - Renewables (Wind, biomass, solar etc. represented scenarios)
  - Transmission Network (Linearized power flow model with quadratic losses, security constraints etc.)
- Stochastic parameters
  - Hydro inflows and renewable generation - Multivariate stochastic model
  - Uncertainty on fuel costs - Markov chains (hybrid SDDP/SDP model)
  - Wholesale energy market prices - Markov chains
  - Generation & transmission equipment outages - Monte Carlo

SDDP – Power system operation modeling

- Multi-stage: time coupling due to energy reservoirs
- Stochastic: multiple parameters are uncertain
- Solving the deterministic equivalent LP is not feasible
  - Too many scenarios and stages: the scenario tree grow too fast
- SDDP stands for Stochastic Dual Dynamic Programming, an algorithm developed by Mario Pereira (PSR founder and president)
  - ICSP: 5 sessions and 22 talks
  - julia
    - https://github.com/odow/SDDP.jl
    - https://github.com/blegat/StructDualDynProg.jl
    - https://github.com/JuliaOpt/StochDynamicProgramming.jl
    - Many others...

SDDP characteristics

- **Fortran code**
  - Hundreds of thousands of lines
  - Writing optimization models in non algebraic notation
- **Distributed processing**
  - Brazil's case: 100,000,000 optimization problems
  - The one-stage subproblems in both forward and backward steps can be solved simultaneously, which allows the application of distributed processing
  - SDDP has been running on computer networks since 2001; from 2006, in a cloud system with AWS

SDDP – Power system operation modeling

Iterative procedure

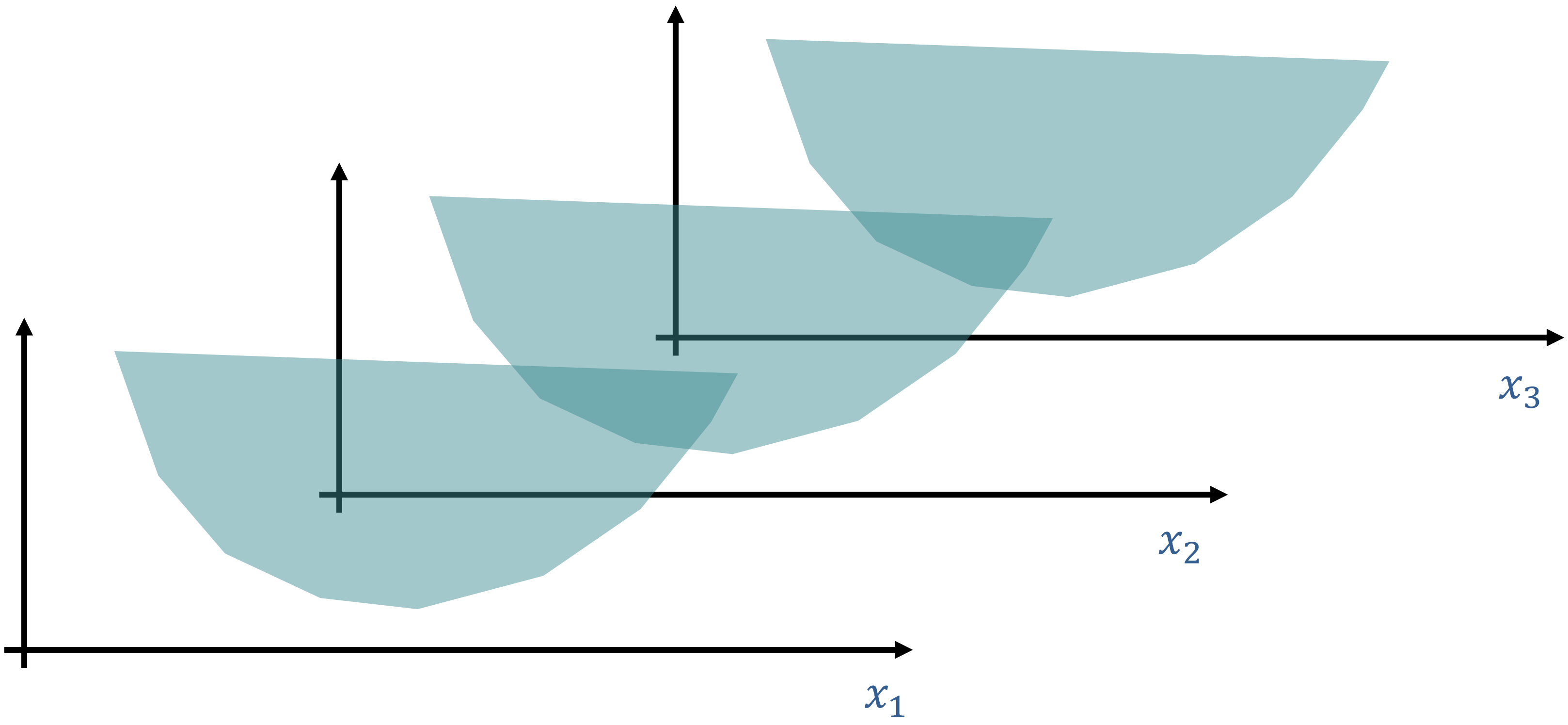
1. forward simulation: propagates volumes of reservoirs in time (states)
2. backward recursion: lower approximation in a Future Cost Function
3. convergence check (LB in UB confidence interval)

Multi-stage economic dispatch and SDDP

- Classical dispatch problem:
  - Objective $min \sum c_j g_j + \beta(v_{t+1}, a_{t+1}^s)$
  - Water Balance $v_{t+1} = v_t + a_t - u - s$
  - Load Balance $\sum g_j + \sum \rho_i u_i = d - \sum r_k$
  - AR model $a_{t+1}^s = \phi_1 a_t + \phi_2 a_{t-1} + \xi_{t+1}^s$
- And much more...

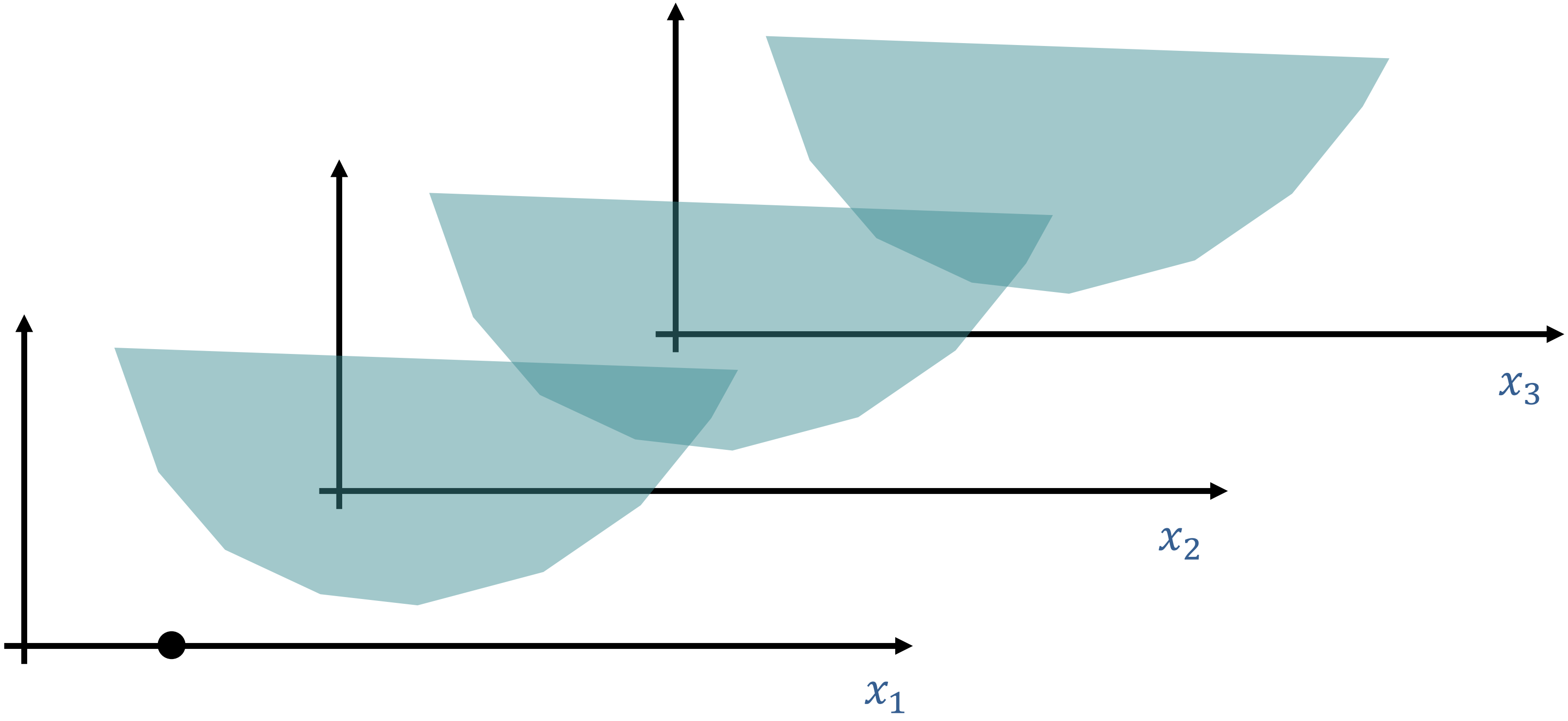
SDDP

Future cost function (cost-to-go)



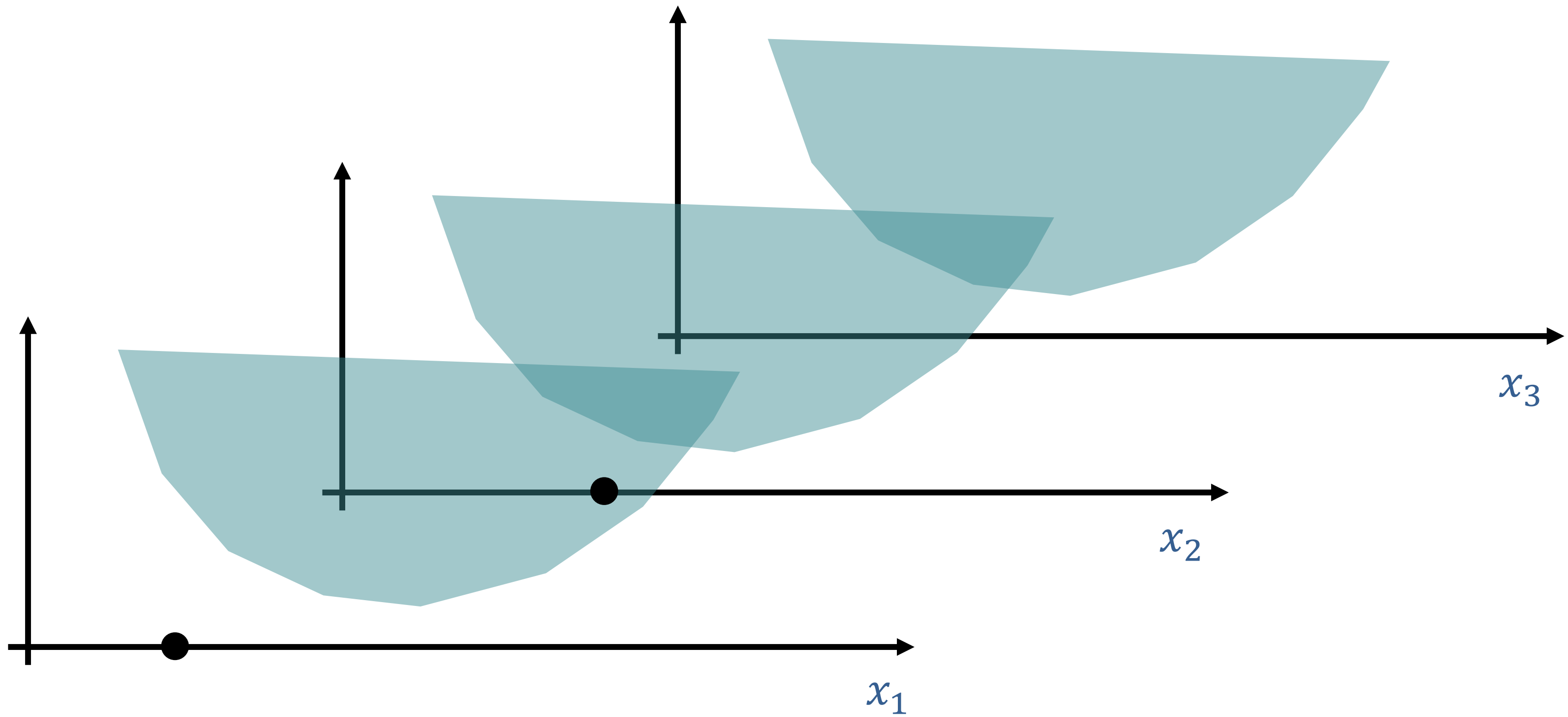
SDDP

Forward iteration 1



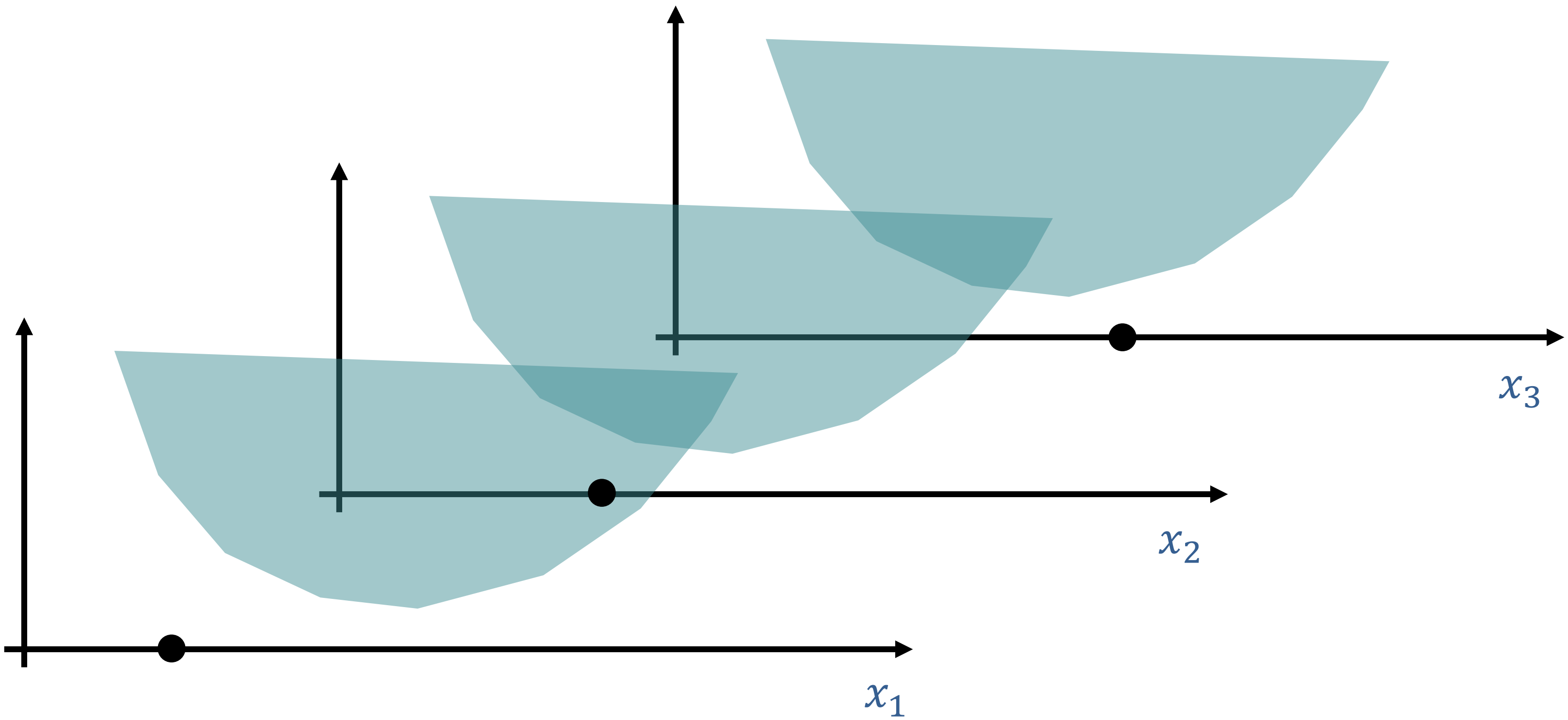
SDDP

Forward iteration 1



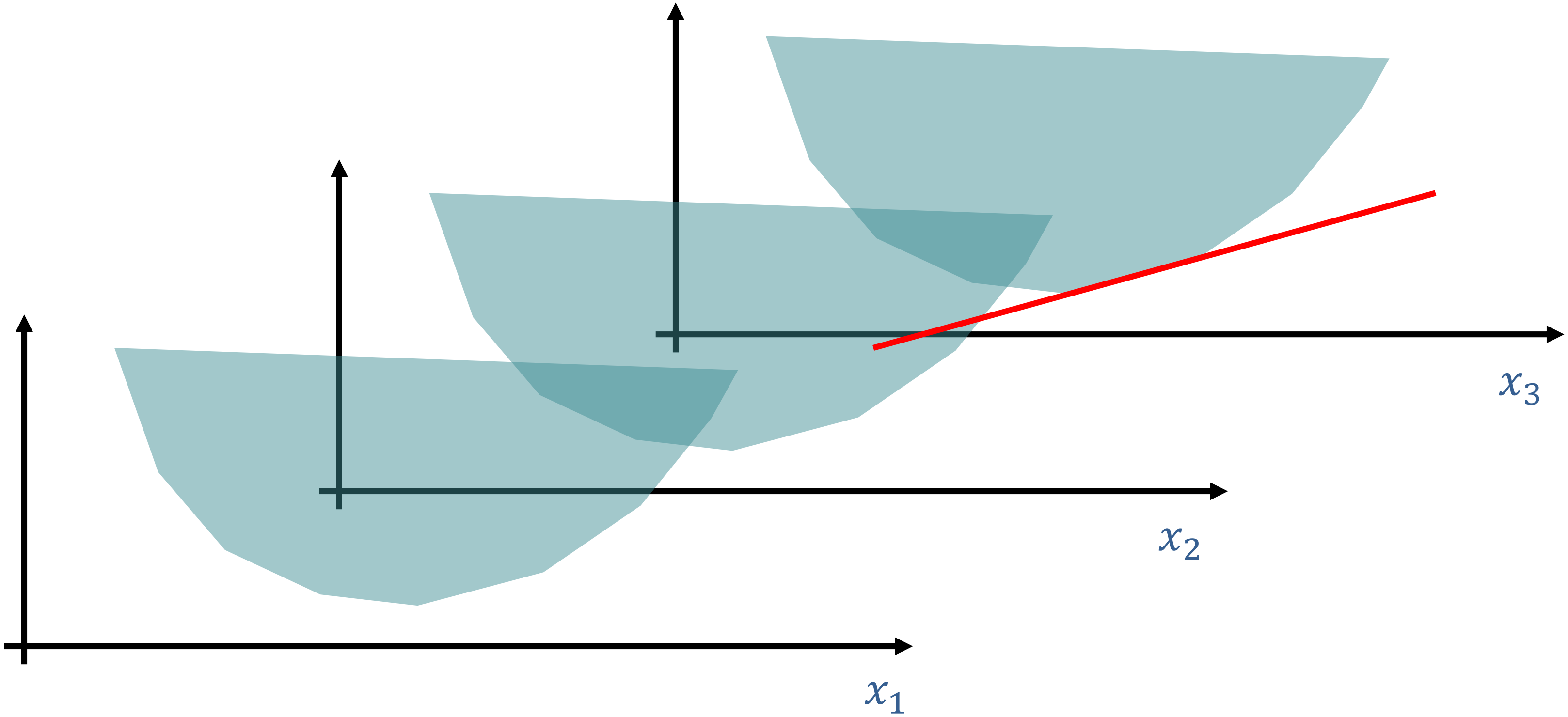
SDDP

Forward iteration 1



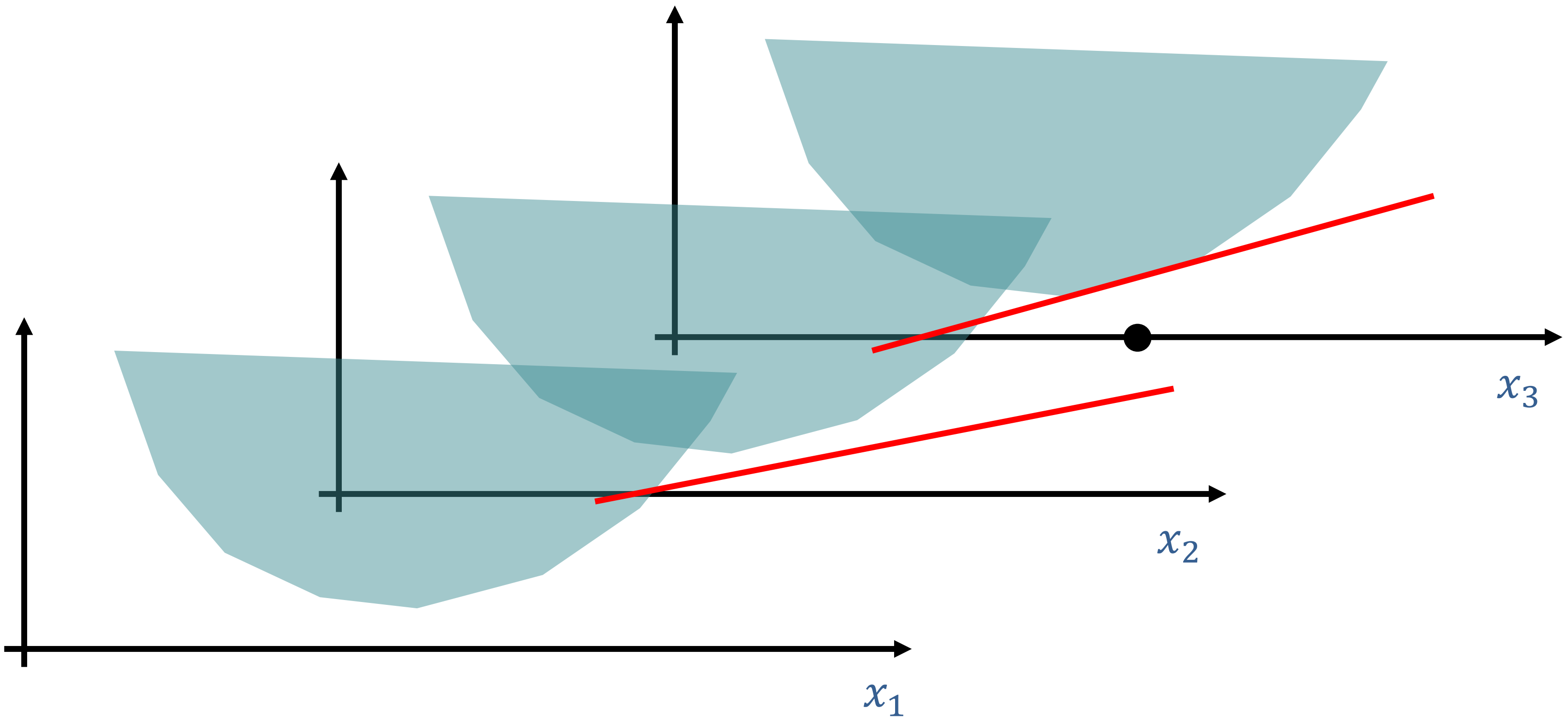
SDDP

Backward iteration 1



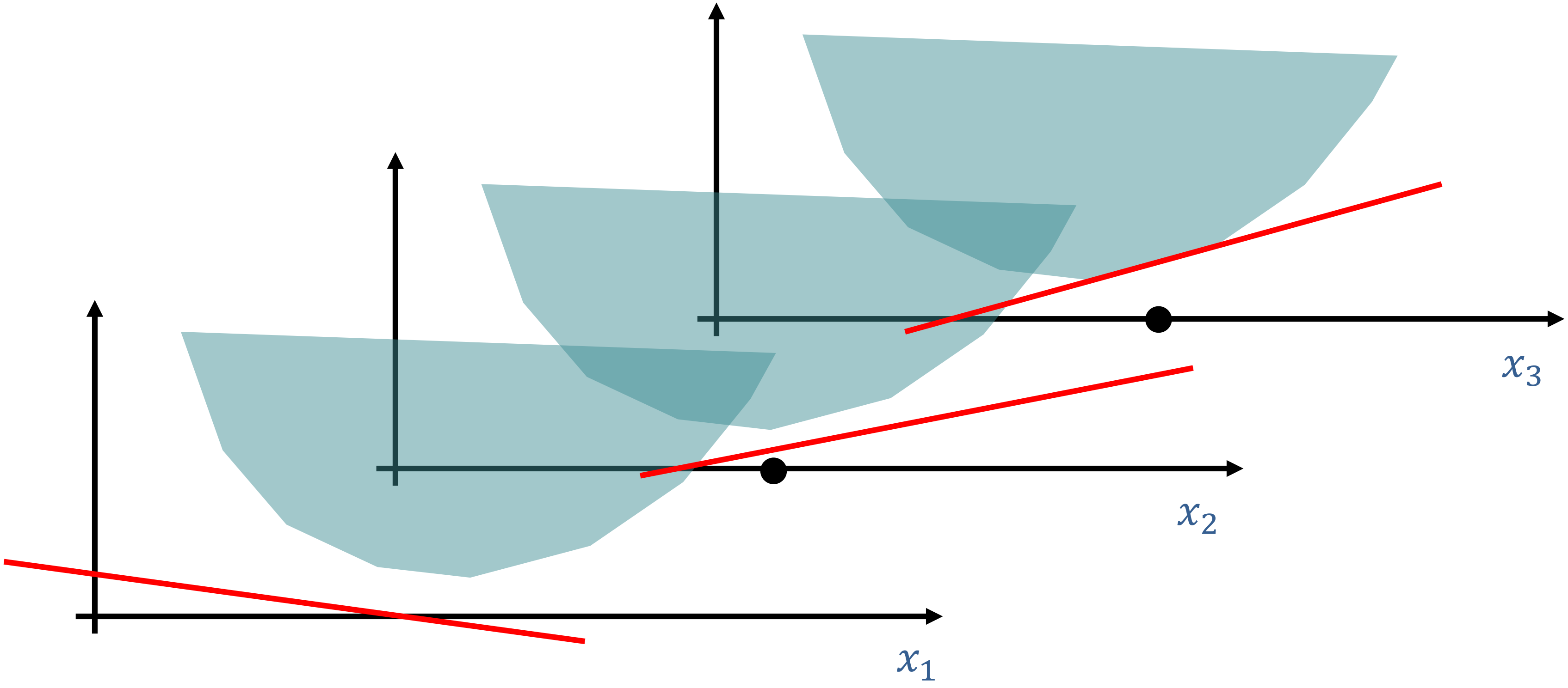
SDDP

Backward iteration 1



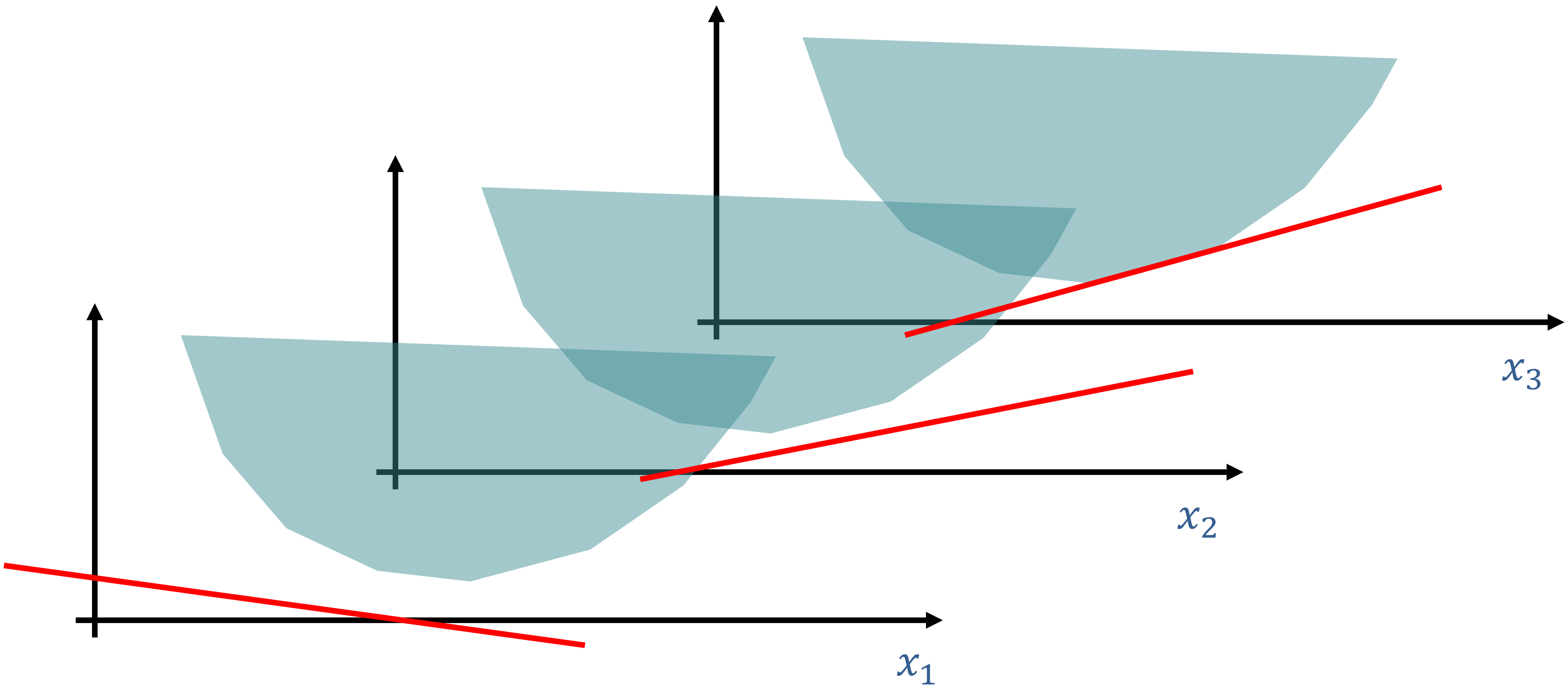
SDDP

Backward iteration 1



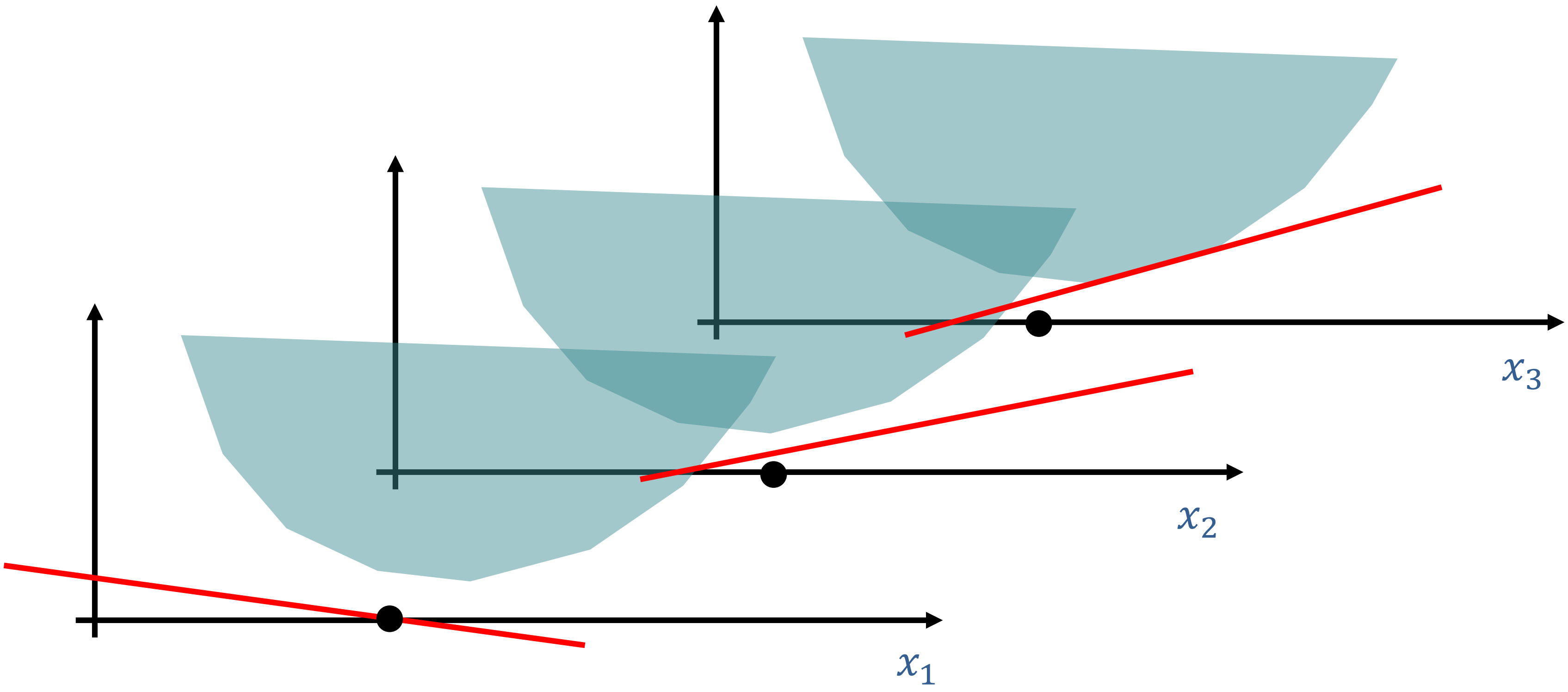
SDDP

Backward iteration 1



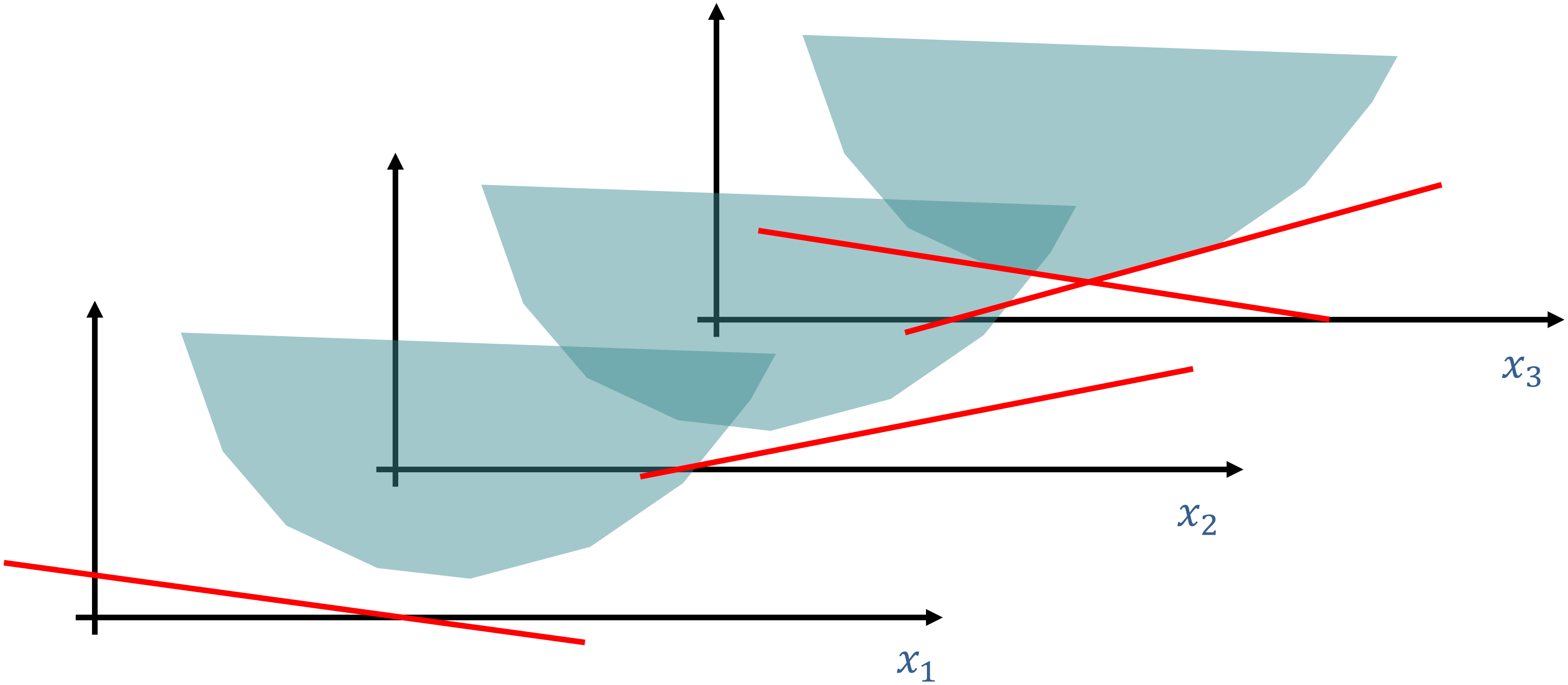
SDDP

Forward iteration 2



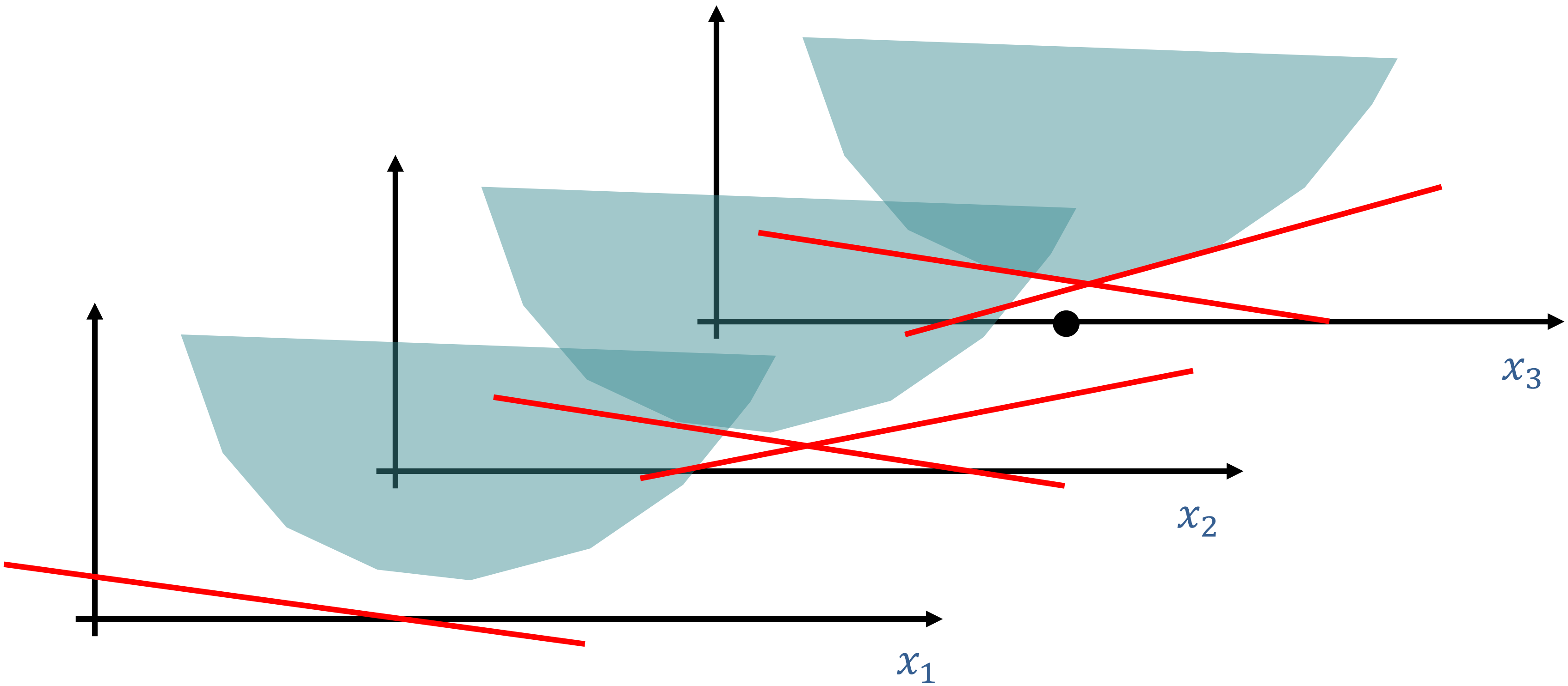
SDDP

Backward iteration 2

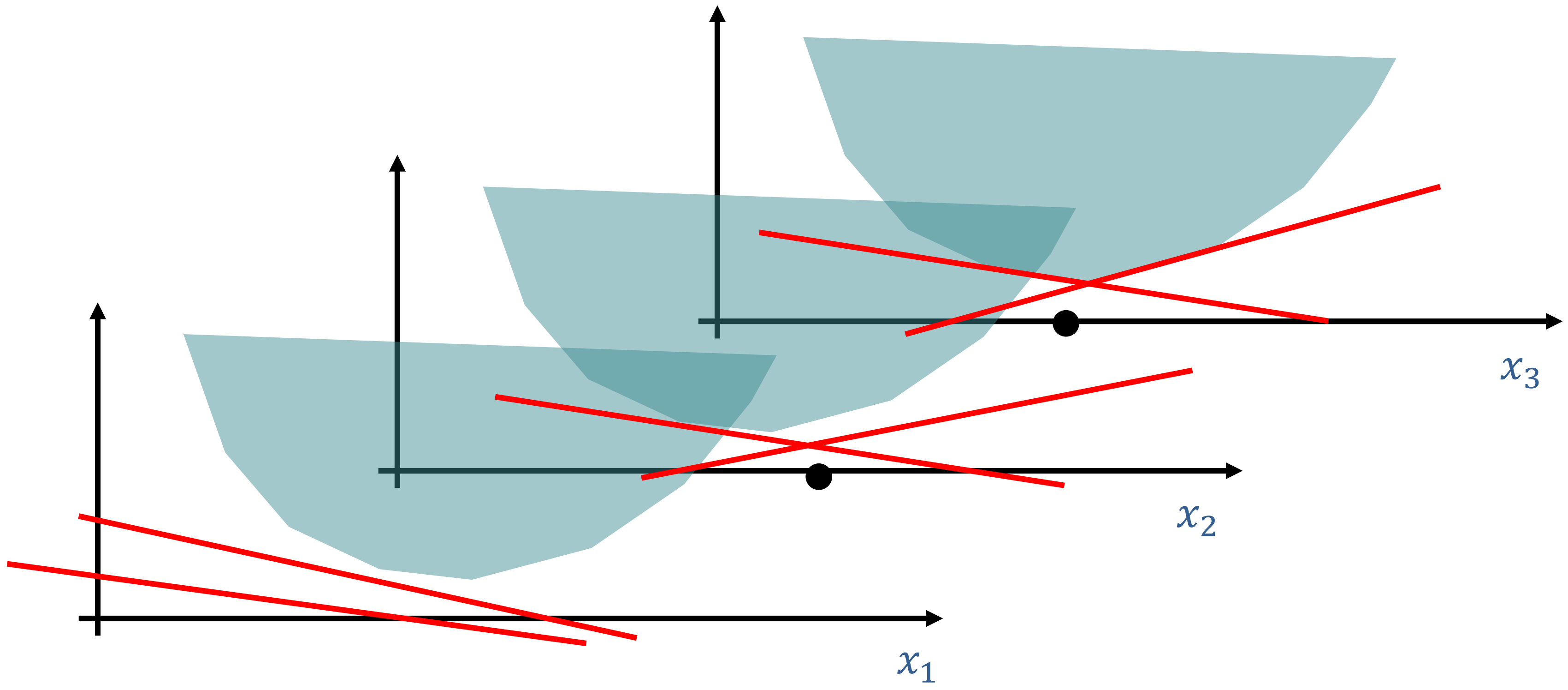


SDDP

Backward iteration 2



Backward iteration 2



SDDP – Applications

- Valuation studies for new projects (hydro, thermal and renewables) and transmission expansion
- Assessment of regional markets and international interconnections
- System price forecast
- Mid and long term production planning
- Regulatory studies and market design simulations
- Water resources analysis (eg. hydropower vs. irrigation) and evaluation of different hydro operative rules
- Natural gas x electricity coordination

SDDP – Implementation

- Why Julia & JuMP
  - Easy to prototype, test new features
  - Good performance
  - Miles: “I want to model and solve LP/MIP within a programming language, but python is to slow and C++ is too low level”
  - Miles:”I want to implement optimization algorithms in a fast, high-level language designed for numerical computing”
- EX: Central America (2500 vars, 25000 cons, 60 stages, 100 scenarios, 5 iterations. More than 200 build problems, 1.000.000 solved in 20min)
- Very good results: more than 70% of time in (Xpress) solver!
  - Thanks to TimerOutputs
- Trivially working in clusters on amazon with MPI

SDDP – Implementation

- Moreover
  - Miles: “Make it easy to access low-level features. Don’t get in the user’s way”
- A big problem: cut relaxation (selection)
  - Adding cuts as extra julia constraints: too slow
    - Use MPB lower level
- Rewriting many problems:
  - small problem: build in 4.5ms, solve in 0.4ms
  - Too expensive to write problems for each scenario
    - Build once per stage!
  - Remove constraints (also variables, changecoeffs, fixvalues)
- Vary parameters: tolerances etc
- More: fixglobals, inplace getters

SDDP – Implementation

- Moreover
  - Miles: “Make it easy to access low-level fea
- A big problem: cut relaxation (selectior
  - Adding cuts as extra julia constraints: too sl
    - Use MPB lower level
- Rewriting many problems:
  - small problem: build in 4.5ms, solve in 0.4m
  - Too expensive to write problems for each s
    - Build once per stage!
  - Remove constraints (also variables, changecoeffs
- Vary parameters: tolerances etc
- More: fixglobals, inplace getters

OptGen – Overview

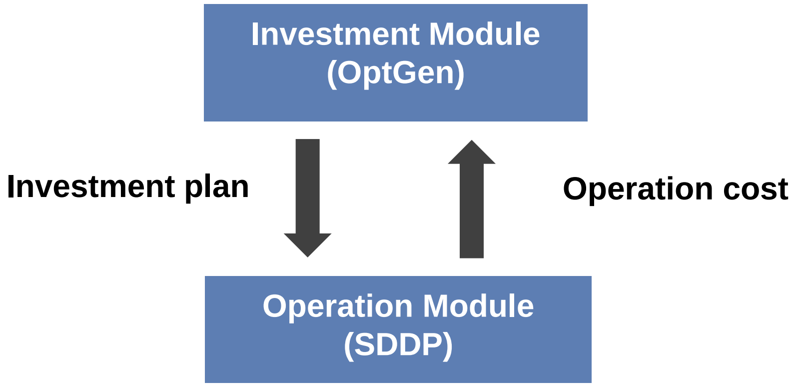
- Optimal mid and long term capacity expansion planning
- Horizon: monthly/weekly stages; up to several decades
- Detailed representation of hydro, thermal and renewable plants' production

OptGen – Plants and system modeling

- Candidate projects
  - Production components: Hydros, Thermals, Renewables
  - Interconnection links and transmission circuits (lines, transformers, etc.)
  - Gas pipelines and production nodes
- SDDP is used as the operation module
- Additional input data (for investment decision)
  - Investment costs
  - Investment timeframe windows
  - Relational constraints (association, precedence, exclusivity)
  - Budgetary constraints
  - Firm energy/ capacity constraints

OptGen – Solution method

- Two-stage optimization problem solved by a customized Benders decomposition
  - **First-stage:** investment problem, formulated as a large scale Mixed Integer Linear Programming (MILP) problem
  - **Second-stage:** operation problem, solved by SDDP
  - Solved by an industry leader optimization solver



OptGen – Applications

- Used by the World Bank globally and by utilities, market operators, regulators and investors in the several studies:
- Generation expansion planning studies including regional interconnections links with Central America systems (Panamá, Costa Rica, Nicaragua, Honduras, El Salvador e Guatemala) for the horizon 2009-2023
- Re-evaluation of the generation-transmission integrated studies with Bolivian system (10 years horizon period, 2009-2018)
- Expansion studies of Egypt-Sudan-Ethiopian interconnection system to provide an economic evaluation to justify the expansion of the interconnection and the construction of large reservoirs
- Studies for the economic evaluation for the construction of the second transmission line connecting all six Central American countries (Panamá, Costa Rica, Nicaragua, Honduras, El Salvador and Guatemala)
- Evaluation of the generation expansion planning for the Dominican Republic, horizon of 10 years (2007-2016)

OptGen – Applications

- Expansion planning studies for the evaluation of new methodologies for the Colombian system (horizon 2006-2017)
- Generation-transmission expansion planning of the Bolivian system with a detailed representation of the transmission system (horizon 2005-2015)
- Expansion studies for the Brazilian system, with 100 GW of installed capacity (85% hydro), considering 117 hydros, 108 thermals and 9 interconnected regions).
- Venezuela’s 2020 generation and transmission expansion plan with 13 interconnected regions
- SEETEC project to study the development and benefits of the Balkan regional energy market, with 8 interconnected countries, 30 GW of demand

OptGen – Model challenges

- Steep learning curve
  - 36,000 lines of code
  - Code is in Fortran
- Difficult to test new ideas
  - Code is rigid

OptGen – Model challenges

- Steep learning curve
  - 36,000 lines of code
  - Code is in Fortran
- Difficult to test new ideas
  - Code is rigid

```fortran
      subroutine ordena( ient, isai, n )
      implicit none

c     Parameter
c     ---------
      integer*4      n, ient(n), isai(n)

c     Local
c     -----
      integer*4      i, j, l, m, ir, kent

      do 10 i = 1, n
       isai(i) = i
 10   continue

      m = n
      l = n/2 + 1
 20   if( l .eq. 1 ) then
        ir = isai(m)
        kent = ient(ir)
        isai(m) = isai(1)
        m = m - 1
        if( m .le. 1 ) then
          isai(1) = ir
          go to 40
        end if
      else
        l = l - 1
        ir = isai(l)
        kent = ient(ir)
      end if

      j = l
 30   i = j
      j = 2*j
      if( j .le. m ) then
        if( j .lt. m ) then
          if( ient(isai(j)) .lt. ient(isai(j+1)) ) j = j + 1
        end if
        if( kent .lt. ient(isai(j)) ) then
          isai(i) = isai(j)
          go to 30
        end if
      end if
      isai(i) = ir
      go to 20

 40   return
      end
```

```
#ifndef LINUX
          cmd = 'copy /y '
          nul = ' > NUL'
          plc = '"'
#else
          cmd = 'cp      '
          nul = '        '
          plc = ' '
#endif
```

```fortran
      subroutine ordena( ient, isai, n )
      implicit none

c     Parameter
c     ---------
      integer*4      n, ient(n), isai(n)

c     Local
c     -----
      integer*4      i, j, l, m, ir, kent

      do 10 i = 1, n
       isai(i) = i
 10   continue

      m = n
      l = n/2 + 1
 20   if( l .eq. 1 ) then
        ir = isai(m)
        kent = ient(ir)
        isai(m) = isai(1)
        m = m - 1
        if( m .le. 1 ) then
          isai(1) = ir
          go to 40
        end if
      else
        l = l - 1
        ir = isai(l)
        kent = ient(ir)
      end if

      j = l
 30   i = j
      j = 2*j
      if( j .le. m ) then
        if( j .lt. m ) then
          if( ient(isai(j)) .lt. ient(isai(j+1)) ) j = j + 1
        end if
        if( kent .lt. ient(isai(j)) ) then
          isai(i) = isai(j)
          go to 30
        end if
      end if
      isai(i) = ir
      go to 20

 40   return
      end
```

```
#ifndef LINUX
          cmd = 'copy /y '
          nul = ' > NUL'
          plc = '"'
```

```
      subroutine ordena( ient, isai, n )
      implicit none

c     Parameter
c     ---------
      integer*4      n, ient(n), isai(n)

c     Local
c     -----
      integer*4      i, j, l, m, ir, kent

      do 10 i = 1, n
        isai(i) = i
 10   continue
```

```
.PHONY: fmts
fmts: $(INDEXDEF) $(DIR_PATH)$(SDDPCONF)
ifeq ($(OUTPUT),subdir)
	@echo ------------------------------------------------------------------------------------------------------
	@echo Copying files index\*.fmt sddp\*.fmt sddpuser.lic psr.lic and xpauth.xpr from root folder to $(DIR_PATH)
	@echo ------------------------------------------------------------------------------------------------------
	@$(CP) index*.fmt $(DIR_PATH)
  ifeq ($(DIM),csvcnv)
	@$(CP) sddpeng.fmt  $(DIR_PATH)csveng.fmt
	@$(CP) sddpesp.fmt  $(DIR_PATH)csvesp.fmt
	@$(CP) sddppor.fmt  $(DIR_PATH)csvpor.fmt
	@$(CP) sddpeng.fmt  csveng.fmt
	@$(CP) sddpesp.fmt  csvesp.fmt
	@$(CP) sddppor.fmt  csvpor.fmt
  else
	@$(CP) sddpeng.fmt  $(DIR_PATH)
	@$(CP) sddpesp.fmt  $(DIR_PATH)
	@$(CP) sddppor.fmt  $(DIR_PATH)
  endif
	@$(CP) $(DIR_PATH)$(SDDPCONF) .
	@$(CP) sddpuser.lic $(DIR_PATH)
	@$(CP) psr.lic      $(DIR_PATH)
	@$(CP) xpauth.xpr   $(DIR_PATH)
endif

$(INDEXDEF): index.def
	@$(PPROC) -P index.def $(INDEXDEF)

$(DIR_PATH)$(SDDPCONF): Makefile $(MAKEINC)
	@echo ------------------------------------------------------------------------------------------------------
	@echo Creating file $(DIR_PATH)$(SDDPCONF)
	@echo ------------------------------------------------------------------------------------------------------
ifneq ($(OS),Windows_NT)
	@echo "export MPI_PATH=$(MPI_PATH)"                 >  $(DIR_PATH)$(SDDPCONF)
	@echo "export LIBRARY_PATH=$(LIBRARY_PATH)"         >> $(DIR_PATH)$(SDDPCONF)
endif
	@echo "export HABILITAR_HIDRA=$(HABILITAR_HIDRA)"   >> $(DIR_PATH)$(SDDPCONF)
	@echo "export SDDP_CHECK=1"                         >> $(DIR_PATH)$(SDDPCONF)

$(cmm_files) $(f90_objects) $(f_objects) $(c_objects) $(fp_objects) $(DIR_PATH)$(PROJ).link: | $(DIR_PATH)

$(DIR_PATH)$(PROJ)$(EXE): cmm_local $(cmm_files) $(f90_objects) check_dep $(f_objects) $(c_objects) $(fp_objects) $(DIR_PATH)$
	@echo ------------------------------------------------------------------------------------------------------
	@echo "Updating version ID: $(GIT_VERSION)"
```

```
ı
)
ıen
ı
ıen
j)) .lt. ient(isai(j+1)) ) j = j + 1
ıt(isai(j)) ) then
(j)
```

```
      subroutine ordena( ient, isai, n )
```

```
#i

      do 10 i = 1, nc
 10   linptr(i) = 0

      i = ncir + 1

 20   i = i - 1
      if( i .eq. 0 ) go to 30
```

```
  n, ient(n), isai(n)
  i, j, l, m, ir, kent
```

```
.PHONY: fm
fmts: $(IN
ifeq ($(OU
	@echo
	@echo
	@echo
	@$(CP)
  ifeq ($(
	@$(CP)
	@$(CP)
	@$(CP)
	@$(CP)
	@$(CP)
	@$(CP)
  else
	@$(CP) sddpeng.fmt  $(DIR_PATH)
	@$(CP) sddpesp.fmt  $(DIR_PATH)
	@$(CP) sddppor.fmt  $(DIR_PATH)
  endif
	@$(CP) $(DIR_PATH)$(SDDPCONF) .
	@$(CP) sddpuser.lic $(DIR_PATH)
	@$(CP) psr.lic      $(DIR_PATH)
	@$(CP) xpauth.xpr   $(DIR_PATH)
endif

$(INDEXDEF): index.def
	@$(PPROC) -P index.def $(INDEXDEF)

$(DIR_PATH)$(SDDPCONF): Makefile $(MAKEINC)
	@echo ------------------------------------------------------------------------------------------
	@echo Creating file $(DIR_PATH)$(SDDPCONF)
	@echo ------------------------------------------------------------------------------------------
ifneq ($(OS),Windows_NT)
	@echo "export MPI_PATH=$(MPI_PATH)"                >  $(DIR_PATH)$(SDDPCONF)
	@echo "export LIBRARY_PATH=$(LIBRARY_PATH)"        >> $(DIR_PATH)$(SDDPCONF)
endif
	@echo "export HABILITAR_HIDRA=$(HABILITAR_HIDRA)"  >> $(DIR_PATH)$(SDDPCONF)
	@echo "export SDDP_CHECK=1"                        >> $(DIR_PATH)$(SDDPCONF)

$(cmm_files) $(f90_objects) $(f_objects) $(c_objects) $(fp_objects) $(DIR_PATH)$(PROJ).link: | $(DIR_PATH)

$(DIR_PATH)$(PROJ)$(EXE): cmm_local $(cmm_files) $(f90_objects) check_dep $(f_objects) $(c_objects) $(fp_objects) $(DIR_PATH)$(
	@echo ------------------------------------------------------------------------------------------
	@echo "Updating version ID: $(GIT_VERSION)"
```

```
hen
hen
j)) .lt. ient(isai(j+1)) ) j = j + 1
nt(isai(j)) ) then
(j)
```

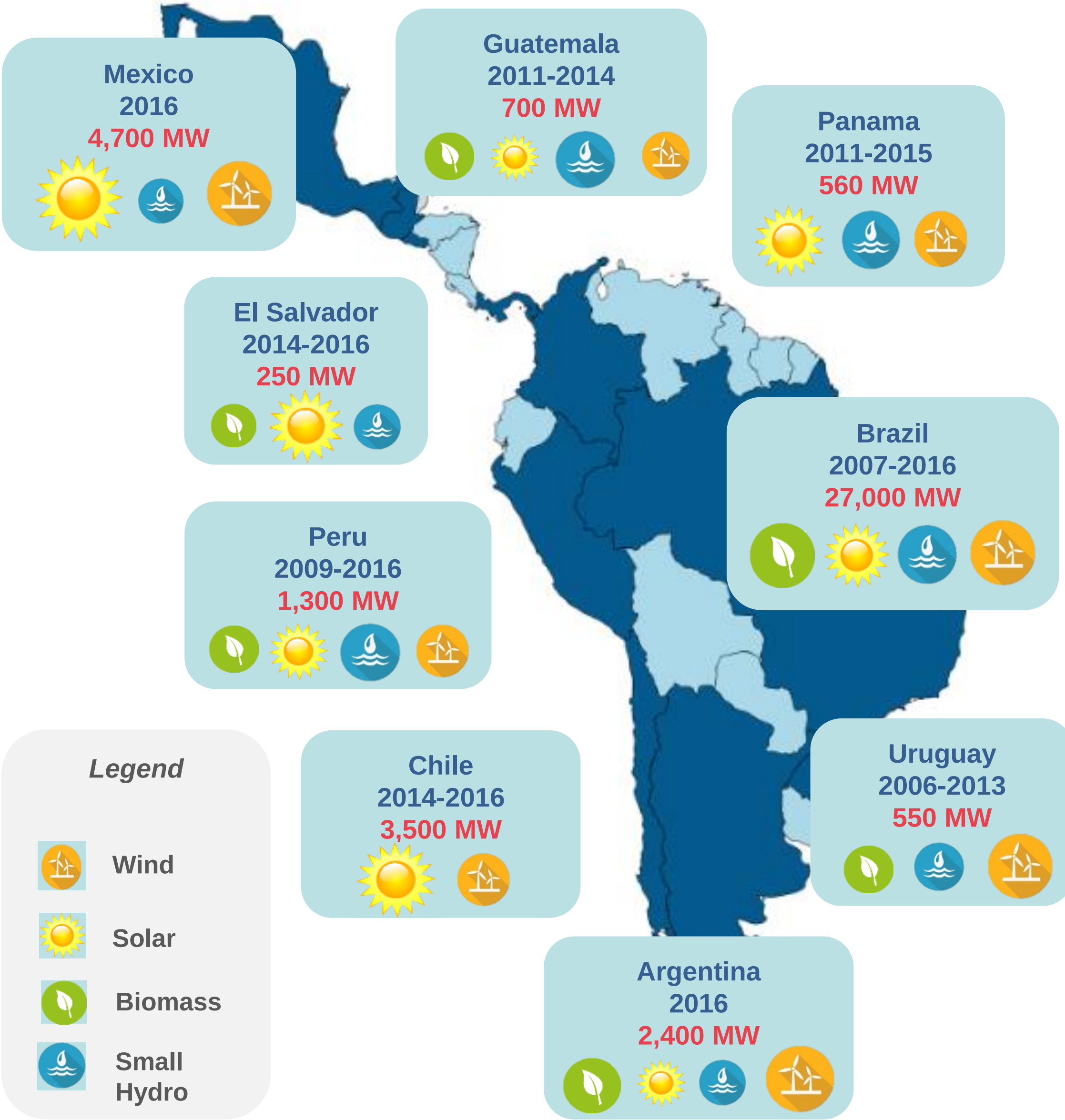
OptGen – Model challenges

- Steep learning curve
  - 36,000 lines of code
  - Code is in Fortran
- Difficult to test new ideas
  - Code is rigid
  - And things are changing fast...

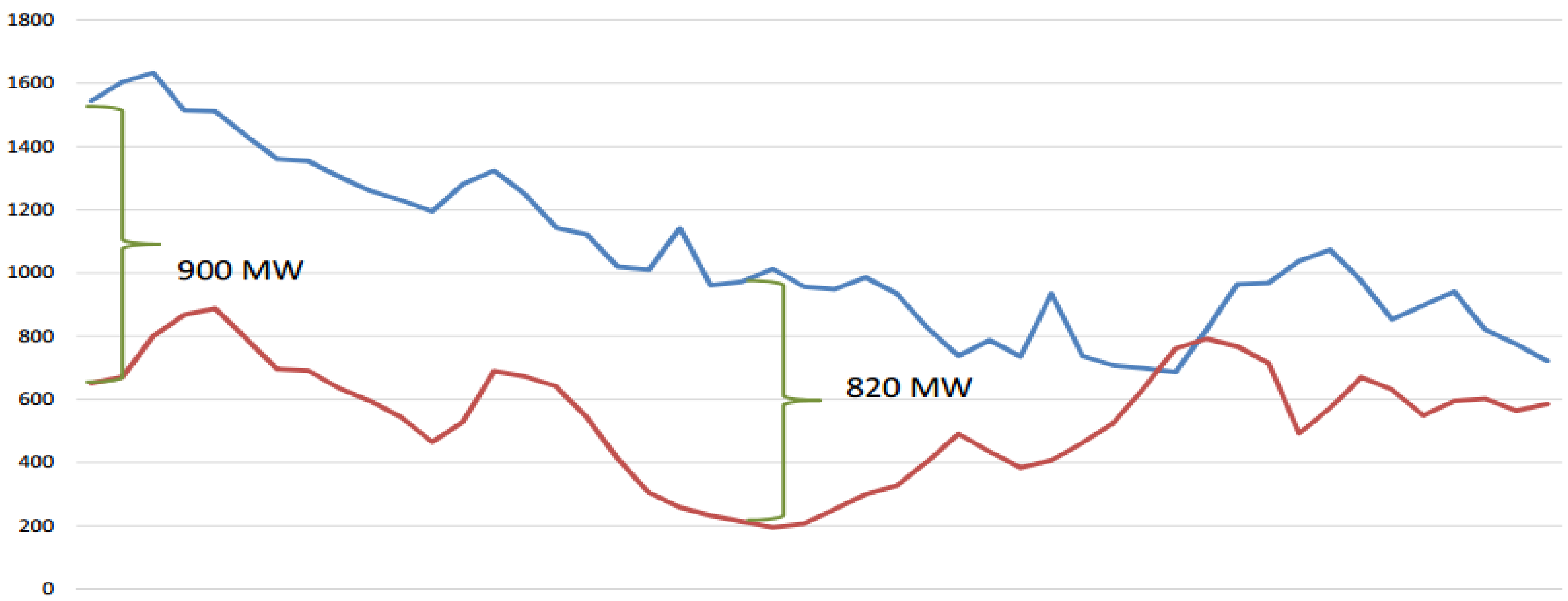
Renewable generation

All over the world:

In 2015, over 60 GW of wind generation sources were installed in the world, half of this capacity in China
In Germany, the goal is to have renewable generation responsible for 80% of total yearly energy generation in 2050



Renewable generation



OptGen – Julia

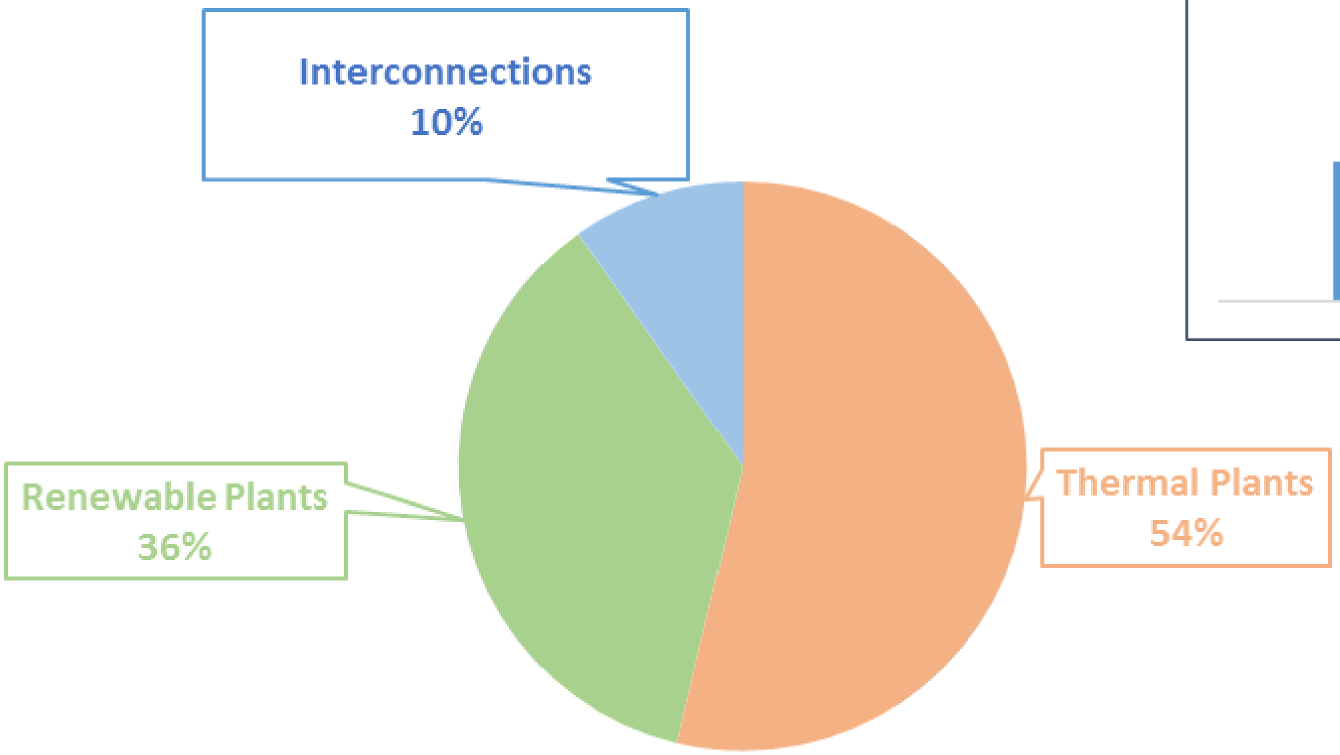
- Completely new formulation focusing on energy reserve:
  - Yearly subproblems
  - MILP: JuMP HEAVILY used
  - Investments are now compared to their required reserve
  - Approach completely adapted to renewable generation and battery projects
  - Test different decomposition approaches
  - 4,200 line model
- Motivated work on Xpress.jl (IIS, Callbacks)
- Easily converted in decomposition technique SDDiP
  - Generating cuts and solving problem in parallel (MPI) in large scale (AWS)
- Orders of magnitude faster for some classes of problems

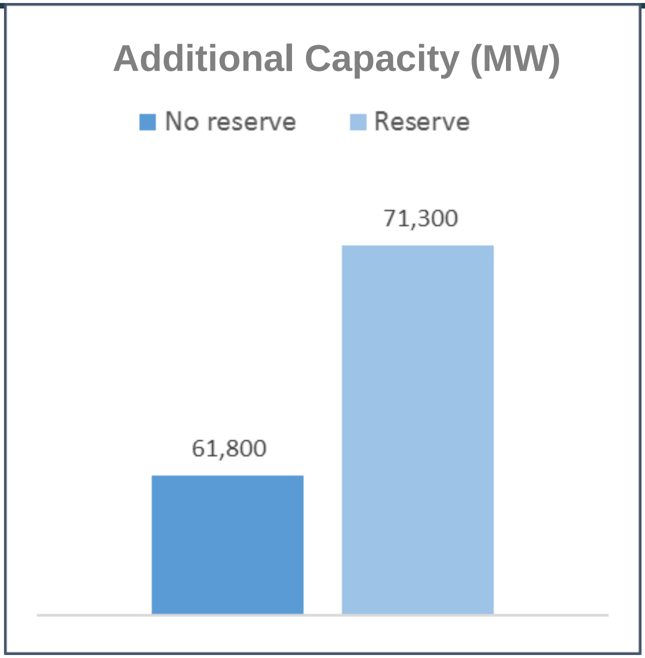
OptGen – Julia

- Already used in real studies:
  - Saudi Arabia
  - Colombia
  - Chile
  - Inter-American development bank
- Saudi Arabia expansion planning:
  - 100% thermal system wanting to invest in renewable sources
  - Multiarea
  - Considerable amount of integer variables (mainly because of thermal minimum generation representation)

OptGen – Julia

- Saudi Arabia expansion planning:





OptFlow – Julia

- Originally focused on non-linear optimal power flow for reactive expansion & investment
- Taylor made IPM
  - Extremely fast
  - Hard to add new constraints and prototype new ideas
- New version: based in JuMP desigend to accomodate non-linear power flow and its convex relaxations
  - Ready to generate valid cuts for decompostion algorithms
  - Easy implementation of progressive hedging

OptFlow – Julia

- Similar to PowerModels.jl…
- Abstracts on constraints classes:
  - Non-linear rectangular
  - Non-linear Polar
  - Convex SOCP
  - Convex SDP (complex and real)

OptFlow – Julia

- Some testing results (12600 problems solved sequentially)

TCPU(s)	NL Polar (Ipopt)	NL Retangular (Ipopt)	SOCP (Xpress)
Input	30.13	37.17	93.95
Write Model	19.80	22.66	136.44
Solver	1568.99	4340.25	1378.83
Output	16.38	18.39	46.22
Total	1635.31	4418.47	1655.44

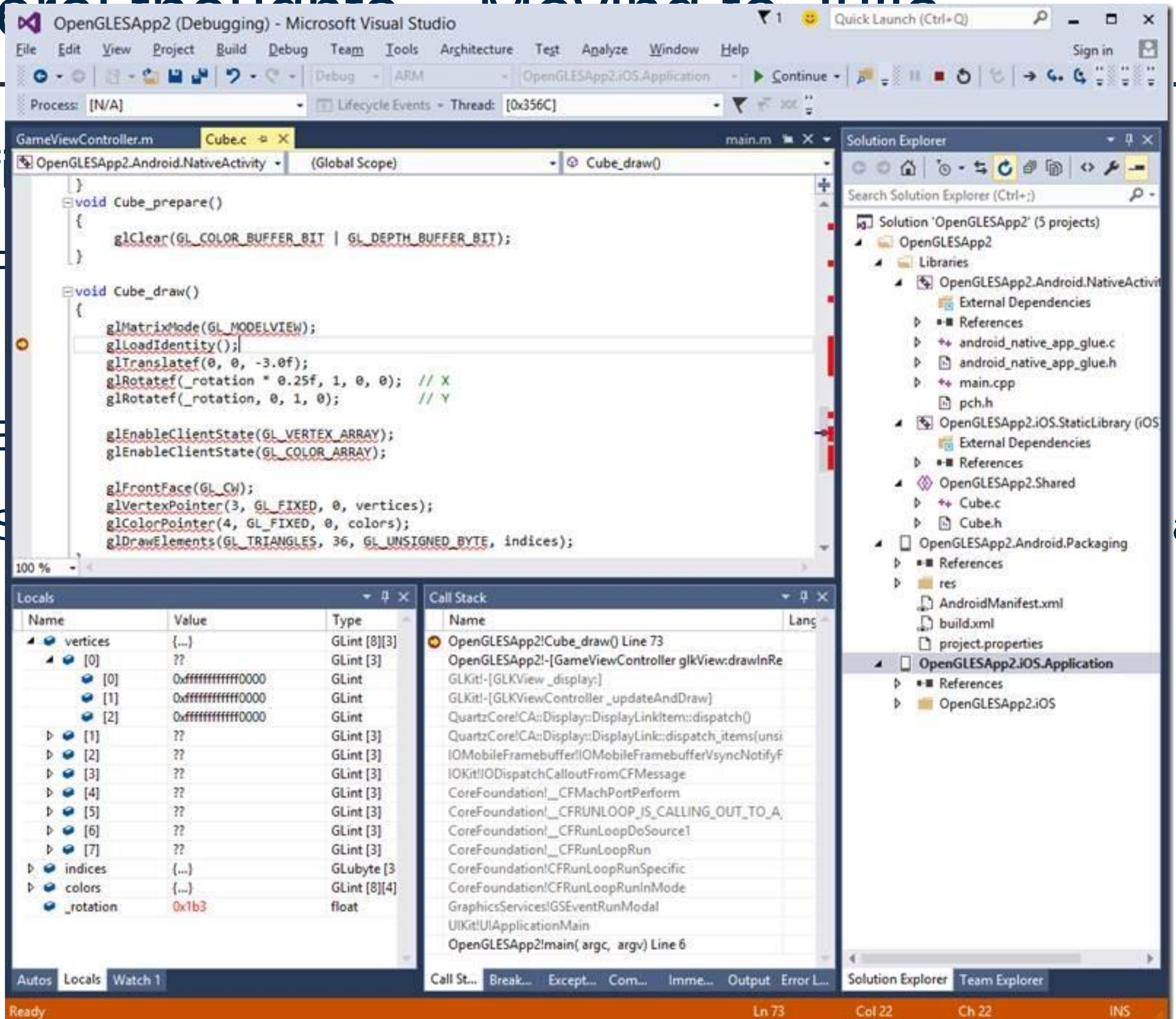
- FORTRAN: 162s

General thoughts – Moving to Julia

- Difficulties
  - PSR is fluent in Fortran
    - Lot's of auxiliary references
  - Self contained small executables (market version of SDDP has 5 MB)
  - Easy DEBUG using VS

Gene… – Moving to Julia

- Diff…
  - F…
  - E…
  - S… as 5 MB)



General thoughts – Moving to Julia

- Difficulties
  - PSR is fluent in Fortran
    - Lot’s of auxiliary references
  - Easy DEBUG using VS
  - 5 MB)

```
      20          for i=4:8:length(record)
  --> 21            @bp
      22            t_begin=2*([3600,60,1]'*map(x->parse(Int64,x)
(3600*9+60*15))+1

debug:21>
```

General thoughts – Moving to Julia

- Difficulties
  - PSR is fluent in Fortran
    - Lot’s of auxiliary references
  - Easy DEBUG using VS
  - Self contained small executables (market version of SDDP has 5 MB)
- Benefits
  - High level efficient language
  - Good optimization environment
  - Easy C interface
  - Good for parallel computing

PSR

Questions?

Thank you!

www.psr-inc.com

psr@psr-inc.com

+55 21 3906-2100

+55 21 3906-2121