Accelerating interior point methods with GPUs for smart grid systems

Nicolai Fog Gade-Nielsen

Department of Informatics and Mathematical Modelling
Technical University of Denmark

GPU Computing Today and Tomorrow, 2011
Technical University of Denmark

Outline

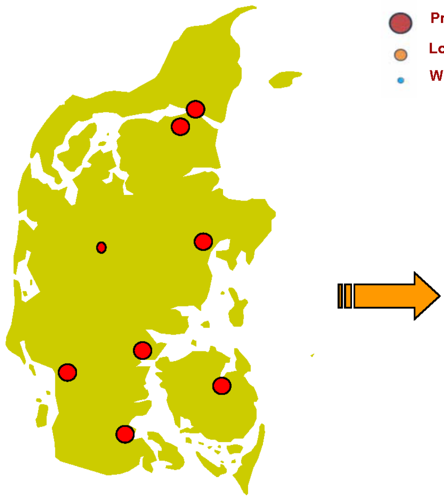
Introduction

- Ph.D. student at GPUlab, DTU IMM since November 2010.
  - Title: Scientific GPU Computing for Dynamical Optimization
  - Investigate GPUs for solving optimization problems, primarily model predictive control (MPC).
  - Initial test problem is distribution and control of electrical power via smart grid through the use of MPC.
- Completed M.Sc. at DTU IMM in October 2010:
  - Title: Implementation and evaluation of fast computational methods for high-resolution ODF problems on multi-core and many-core systems
  - Used GPUs for ray tracing and matrix-free SpMV.

What is a smart grid?

- Dynamically control energy production and consumption according to some objective, eg:
  - Increased use of uncontrollable renewable power such as wind and solar.
  - Lower production costs.
  - Lower CO2 production.
  - etc.
- Examples
  - Heat pump control in houses
  - **Power plant production control**

Power plants in Denmark



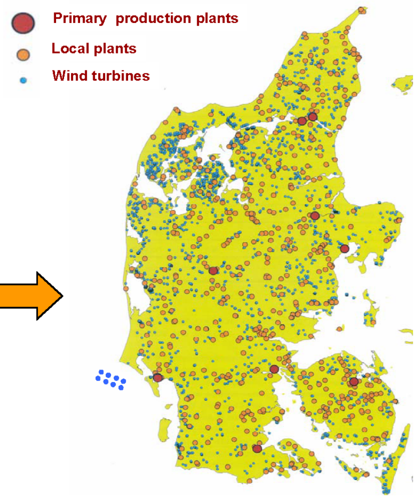
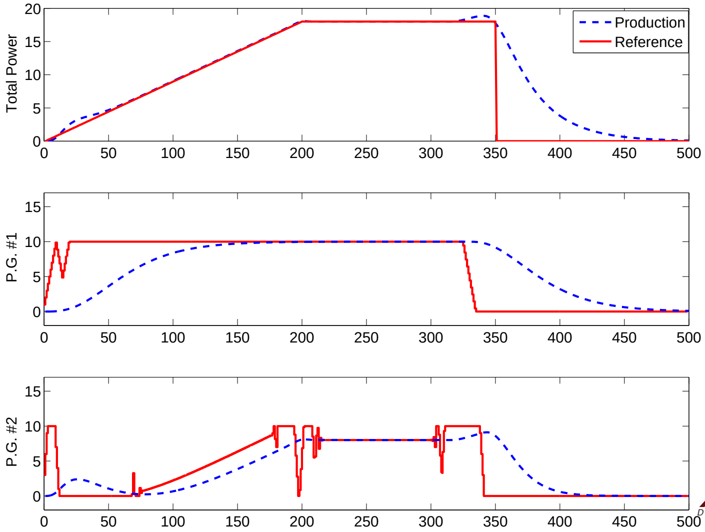


Image credit: Economic MPC for Power Management in the Smart Grid | Tobias Gybel Hovgaard

Simple Economic MPC 1

- Prediction horizon of N time steps.
- N_p number of power plants with different properties.
- Properties on each power plant:
  - Response time.
  - Minimum and maximum change in one time step.
  - Minimum and maximum power production.
  - Cost.
- Simple example: Two power plants.
  - A cheap but slow power plant.
  - An expensive but fast power plant.

Simple Economic MPC example



Simple Economic MPC 2

- Define power plant production as an economic MPC problem:

$$\begin{aligned} min \quad & \phi = g's + c'x \\ s.t. \quad & Ax - s = b \\ & s \geq 0 \end{aligned}$$

- c is the cost and x is the control variables of each power plant in each time step.
- s is a slack variable and g is the cost of using it.
- A is our constraint matrix which encodes the constraints of each power plant and dynamics of the system. Very sparse and highly structured.
- Number of control variables: $N_p * N + N$

IPM Algorithm (without Predictor-Corrector)

- While not converged do
    - Compute duality gap.
    - Set centering parameter.
    - Compute residuals.
    - Solve newton step.

$$\begin{bmatrix} 0 & -A' & 0 \\ -A & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_s \\ r_\lambda \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 0 & -A' \\ -A & -D^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} r_L \\ r_s - \Lambda^{-1} r_\lambda \end{bmatrix} \tag{2}$$

$$D = S^{-1}\Lambda$$

$$S = diag(s), \Lambda = diag(\lambda)$$

    - Compute step length.
    - Update step.

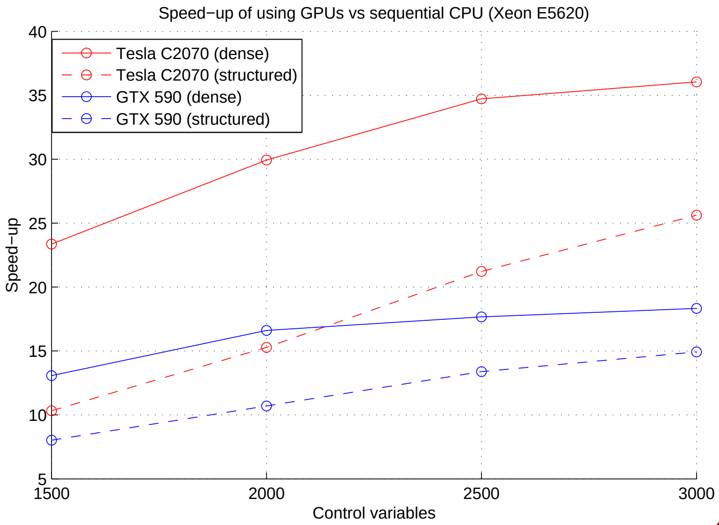
Solving the Newton step

- Bottleneck: Solving the Newton step
- Normally solved with direct method:
  - Compute the Hessian matrix and use Cholesky factorization.

$$H_A = A^T(D)A$$
$$Cholesky : H_A = LL^T$$

- Implemented using CUBLAS and MAGMA.
- What is MAGMA?
  - 'The MAGMA project aims to develop a dense linear algebra library similar to LAPACK but for heterogeneous/hybrid architectures, starting with current "Multicore+GPU" systems.'
  - Very fast Cholesky factorization.
  - Developed at University of Tennessee.

Results


Speed−up of using GPUs vs sequential CPU (Xeon E5620)
Tesla C2070 (dense)
Tesla C2070 (structured)
GTX 590 (dense)
GTX 590 (structured)
Speed−up
Control variables
40
35
30
25
20
15
10
5
1500
2000
2500
3000

Problems

- Ill-conditioning
  - Normal equations -> condition number squared

$$H_A = A^T(D)A$$
$$Cholesky : H_A = LL^T$$

  - In the later iterations of the IPM, either s_i or λ_i goes toward 0.

$$\begin{bmatrix} 0 & -A' & 0 \\ -A & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_L \\ r_s \\ r_\lambda \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 0 & -A' \\ -A & -D^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} r_L \\ r_s - \Lambda^{-1} r_\lambda \end{bmatrix} \tag{2}$$

$$D = S^{-1}\Lambda$$
$$S = diag(s), \Lambda = diag(\lambda)$$

- Loss of sparsity
  - $N_p = 1000, N = 48$: 48048 decision variables.
  - 48048^2 hessian matrix to factorize: about 17 GB.

Future plans

- Iterative methods and preconditioning
  - Identify iterative solvers for solving the newton step iteratively.
  - Identify good preconditioners.
  - Evaluate which of the solvers and preconditioners are suitable for GPU implementation.
- Create toolbox for solving optimization problems using GPUs
- Smart grid problems used as test case for toolbox.