ME964 - Midterm Project Topic 4
Solving a Sparse KKT Linear System

Dan Negrut

March 15, 2011

Quick Overview

- Motivation for making this problem a Midterm Project topic: solving sparse linear systems is very common in optimization problems. As an example, consider solving the following quadratic optimization problem with equality constraints:

$$\begin{cases} \text{minimize} & q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{k} \\ \text{subject to} & \mathbf{J}\mathbf{x} = \mathbf{c} \end{cases} \tag{1}$$

- $\mathbf{x} \in \mathbb{R}^n$ – the set of variables over which the cost function needs to be minimized
- $\mathbf{M} \in \mathbb{R}^{n\times n}$ – given to you, it is a symmetric and positive matrix.
- $\mathbf{k} \in \mathbb{R}^n$ – given to you
- $\mathbf{J} \in \mathbb{R}^{m\times n}$ – given to you, it is a sparse rectangular matrix
- $\mathbf{c} \in \mathbb{R}^m$ – given to you
- The above quadratic optimization problem with equality constraints can be solved by finding the solution of the following linear system:

$$\begin{bmatrix} \mathbf{M}_{n\times n} & \mathbf{J}^T_{m\times n} \\ \mathbf{J}_{m\times n} & \mathbf{0}_{m\times m} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix}_{n+m} = \begin{bmatrix} \mathbf{k} \\ \mathbf{c} \end{bmatrix}_{n+m} \tag{2}$$

- NOTE:
  - λ is the Lagrange Multiplier associated with the optimization problem at hand.
  - Each of the m constraints present in the problem has a corresponding Lagrange Multiplier λ. In other words, constraint j has an associated Lagrange Multiplier λ_j.
  - The solution $\mathbf{x}$ of the above system provides the optimal point for the quadratic optimization problem in Eq. (1)
- The coefficient matrix $\begin{bmatrix} \mathbf{M} & \mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{bmatrix}$ is called the KKT matrix

- The matrix $\mathbf{E} \equiv \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ is called the Schur complement of the KKT matrix
- Simplifying assumptions for ME964 project
  - The matrix $\mathbf{M}$ is diagonal. Since it was assumed to be positive definite, $m_{ii} > 0, i = 1, \ldots, n$
  - The matrix $\mathbf{J}$ does not have in any of its rows more than two entries and a row could sometime have only one entry
  - The matrix $\mathbf{J}$ is full row rank (the equality constraints are independent)

Midterm Project Topic 4 requires you to find the solution of the linear system in Eq. (2). To this end, since $\mathbf{M}$ is nonsingular, the suggested solution sequence calls first for solving for λ

$$\mathbf{E}\lambda = \mathbf{J}\mathbf{M}^{-1}\mathbf{k} - \mathbf{c} \ , \tag{3}$$

and then, once λ is available, solving for $\mathbf{x}$:

$$\mathbf{x} = \mathbf{M}^{-1}\left(\mathbf{k} - \mathbf{J}^T\lambda\right) \ . \tag{4}$$

A slightly different strategy is suggested due to improved handling of vastly different entries in the matrix $\mathbf{M}$ (for instance, $\mathbf{M}[0][0] = 10^{-6}$ and $\mathbf{M}[1][1] = 10^{6}$). Several new variables are defined as follows: $\bar{\mathbf{x}} = \mathbf{M}^{\frac{1}{2}}\mathbf{x}$, $\bar{\mathbf{J}} = \mathbf{J}\mathbf{M}^{-\frac{1}{2}}$, and $\bar{\mathbf{k}} = \mathbf{M}^{-\frac{1}{2}}\mathbf{k}$. Note that using this notation, the optimization problem can be expressed as

$$\begin{cases} \text{minimize} & q\left(\bar{\mathbf{x}}\right) = \frac{1}{2}\bar{\mathbf{x}}^T\bar{\mathbf{x}} - \bar{\mathbf{x}}^T\bar{\mathbf{k}} \\ \text{subject to} & \bar{\mathbf{J}}\bar{\mathbf{x}} = \mathbf{c} \end{cases} \tag{5}$$

Using the notation $\bar{\mathbf{d}} \equiv \bar{\mathbf{J}}\bar{\mathbf{k}} - \mathbf{c}$, the suggested solution sequence calls first for solving for λ

$$\mathbf{E}\lambda = \bar{\mathbf{d}} \ , \tag{6}$$

and then, once λ is available, solving for $\bar{\mathbf{x}}$:

$$\bar{\mathbf{x}} = \bar{\mathbf{k}} - \bar{\mathbf{J}}^T\lambda \ . \tag{7}$$

If you end up adopting this strategy for solving the linear system in Eq. (2), a possible sequence of steps that implements this strategy is as follows (all these steps are meant to be implemented on the GPU):

1. Given $\mathbf{J}$, compute $\bar{\mathbf{J}}$, which amounts to scaling the non-zero entries in column i of the original $\mathbf{J}$ matrix by the quantity $1/\sqrt{\mathbf{M}[i][i]}$
2. Given $\mathbf{k}$, compute $\bar{\mathbf{k}}$, which amounts to scaling the entry in row i of the original $\mathbf{k}$ vector by the quantity $1/\sqrt{\mathbf{M}[i][i]}$
3. Compute $\bar{\mathbf{d}} = \bar{\mathbf{J}}\bar{\mathbf{k}} - \mathbf{c}$
4. Do parallel forward elimination on the reduced system $\mathbf{E}\lambda = \bar{\mathbf{d}}$ (see lecture notes, 03/01)
5. Do parallel backward substitution to get λ (see lecture notes, 03/01)
6. Compute $\bar{\mathbf{J}}^T\lambda$, which should be a sparse matrix-vector operation
7. Compute $\bar{\mathbf{x}} = \bar{\mathbf{k}} - \bar{\mathbf{J}}^T\lambda$
8. In interested in this value, compute $\mathbf{x} = \mathbf{M}^{-\frac{1}{2}}\bar{\mathbf{x}}$, which amounts to scaling the entry in row i of $\bar{\mathbf{x}}$ vector by the quantity $1/\sqrt{\mathbf{M}[i][i]}$

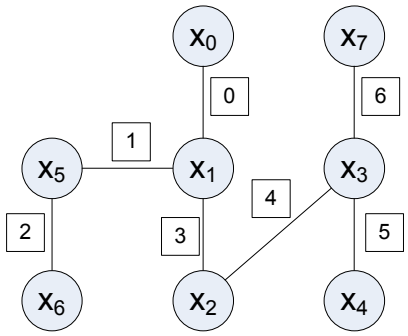

Figure 1: Example connectivity graph for Schur complement matrix.

Defining the Problem Input

Although when your project will be evaluated the dimension of the problem will be $n \approx 10^7$, for the sake of this discussion, assume that the number of variables is $n = 8$, and the number of constraints is $m = 7$, as shown in Fig. (1). An input file defining the KKT linear system will be provided as a command line argument to your executable. This file will have the following format: the first line contains the value of n (8 in this case). The second line contains the value m (7 in this case). The subsequent n lines will contain a set of two numbers: the index of a variable, i, followed by the diagonal entry $\mathbf{M}[i][i]$, which is a double precision number. The subsequent m lines will define the set of constraints. Since we made the simplifying assumption that each row of $\mathbf{J}$ contains two entries (each constraint equation constrains at most two variables), each of these last rows will contain a set of five numbers (three integers, followed by two double precision numbers):

$$j \quad l(j) \quad u(j) \quad \mathbf{J}[j][l(j)] \quad \mathbf{J}[j][u(j)]$$

Here j represents the id of the constraint and $l(j)$ and $u(j)$ represent the indexes of the two variables, $x_{l(j)}$ and $x_{u(j)}$, that show up in the constraint equation j, where the former represents the variable of lower index and the latter represents the variable of higher index (the lower and upper variables). A negative value for $l(j)$ indicates that the constraint j actually involves only one variable; i.e., $x_{u(j)}$, in which case the values $l(j)$ and $\mathbf{J}[j][l(j)]$ should be ignored. The input file for a problem whose connectivity graphs is illustrated in Fig. (1) might look as follows (note that anything that comes after a % sign should be interpreted as a comment and as such ignored upon parsing):

```
8 % the number of variables: n=8 (x_0 through x_7)
7 % the number of constraints: m=7 (lambda_0 through lambda_6)
3 2.3              % says that M[3][3] = 2.3
2 1.1              % says that M[2][2] = 1.1
0 1.3              % etc.
1 0.5
6 0.4
5 1.9
4 0.88
7 1.00
0 0 1 -0.75 1.22             % line 0 of J: J[0][0] = -0.75 and J[0][1]=1.22
2 5 6 -0.22 0.38             % line 2 of J: J[2][5] = -0.22 and J[2][6]=0.38
3 1 2 1.1 1.1                % etc.
```

```
1 1 5 -0.23 2.1
5 3 4 0.44 3.2
6 3 7 -1.5 -1.8
4 2 3 -0.92 0.88
```

The data provided in the input file will only define $\mathbf{M}$ and $\mathbf{J}$. You will have to choose $\mathbf{k}$ and $\mathbf{c}$ so that you can easily verify whether your GPU implementation is correct. To this end, you will choose $\mathbf{k}$ and $\mathbf{c}$ so that $\mathbf{x} = \mathbf{1}_n$ and $\lambda = \mathbf{2}_m$, where $\mathbf{1}_n \equiv [1, \ldots, 1]^T$ has n values of 1. In other words, $\mathbf{k} = \mathbf{M} \cdot \mathbf{1}_n + \mathbf{J}^T \cdot \mathbf{2}_m$, and $\mathbf{c} = \mathbf{J} \cdot \mathbf{1}_n$.

Remarks:

- In relation to matrix $\mathbf{J}$, note that for a row j at most two of its entries are nonzero, specifically the entries in columns $l(j)$ and $u(j)$. All the other entries of row j are zero and therefore there is no point in storing them.
- In terms of storing only non-zero entires, please note that there is a very precise pattern of non-zeros in the reduced matrix $\mathbf{E}$. To specify this pattern, the following definition is first introduced: the connectivity set of a variable $x(i)$, $0 \leq i < n$, denoted in what follows by $\zeta(i)$, is the set of all constraints in which the variable $x(i)$ shows up. Formally, $\zeta(i) = \{j : l(i) = j \text{ or } u(i) = j\}$. In this context, the entry $\mathbf{E}[j][k]$ is non-zero provided there is $0 \leq i < n$ such that $j, k \in \zeta(i)$; i.e., the variable $x(i)$ appears both in constraint j and k. With this in mind, you will also want to only store the non-zero entries of the $\mathbf{E}$ matrix. Note that if $j, k \in \zeta(i)$
  - If $j \neq k$,
  $$\mathbf{E}[j][k] = \mathbf{E}[k][j] = \bar{\mathbf{J}}[j][i] \cdot \bar{\mathbf{J}}[k][i] \tag{8}$$
  - Else,
  $$\mathbf{E}[j][j] = \bar{\mathbf{J}}^2[j][u(j)] + \bar{\mathbf{J}}^2[j][u(j)] \tag{9}$$
- Given how the problem was posed, we must have $m \leq n$. If $m > n$, the KKT system has a solution only under certain conditions. Very succinctly, some rows of the matrix $\mathbf{J}$ must be linearly dependent, and this linear dependency should also carry over to the entries of the vector $\mathbf{c}$. The details are skipped here, we will assume that always $m < n$, as is the case for the example considered in Fig. (1). This assumption in turn leads to the conclusion that the reduced matrix $\mathbf{E}$ is positive definite. As such, a Cholesky algorithm can be used to find the solution of the system $\mathbf{E}\lambda = \mathbf{d}$.
- In your code you should always work with floating point data in *double precision*.

Deliverables

For the intermediate report you should provide an overview of the algorithm that you plan to implement. It should include a flow diagram, data structures that you plan to use, discuss how your algorithm maps upon the underlying SIMD architecture, a discussino of possible limiting factors that work against your solution implementation (for instance, if all threads executing a kernel need to synchronize, or to perform atomic operations, etc.). You will have to indicate the use of any third party CUDA libraries such as *thrust*, for instance. The use of existing libraries is encouraged as long as they don't completely solve your problem.

For the Midterm Project deadline, you will have to implement a GPU solution to the KKT problem stated.

For the Final Project deadline you could adopt one of the following two directions: optimize your implementation and compare it with a state of the art sparse solver available in the open-source community (I suggest Pardiso, http://www.pardiso-project.org/). Alternatively, you could generalized the implementation and start working with a block diagonal $\mathbf{M}$ matrix and with a block Jacobian $\mathbf{J}$ matrix. An ambitious project would work with 3×3 blocks, which is something that is badly needed in the solution of 3D frictional-contact multi-body dynamics problems.