An Algorithm for Time Dependent PDE-constrained Optimization Problems

Youngsoo Choi and Walter Murray

Institute for Computational and Mathematical Engineering,
Stanford University

May 16, 2011
Outline

1. Brief Overview of PDE-constrained Optimization
2. PITA (Parallel Implicit Time-integration Algorithm)
3. “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)
4. Flapping Beam
5. Computational Results
Outline

1. Brief Overview of PDE-constrained Optimization
2. PITA (Parallel Implicit Time-integration Algorithm)
3. “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)
4. Flapping Beam
5. Computational Results
Three applications of PDE constrained optimization

http://venturebeat.com

http://www.nhlbi.nih.gov
NAND vs. SAND

- Two ways of attacking PDE-constrained problem: NAND (Nested Analysis and Design, in reduced space) and SAND (Simultaneous Analysis and Design, in full space)
- $y$: state variables (i.e. displacement, velocity)
- $u$: control variables (i.e. external force, user defined displacement)

**NAND**

$$\text{minimize} \quad F(u)$$

- $u$ is a function of $y$ via PDE constraints $C(y, u) = 0$
- only $u$ is optimization variables

**SAND**

$$\text{minimize} \quad F(y, u)$$

subject to $C(y, u) = 0$

- both $u$ and $y$ are optimization variables
Outline

1. Brief Overview of PDE-constrained Optimization
2. PITA (Parallel Implicit Time-integration Algorithm)
3. “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)
4. Flapping Beam
5. Computational Results
PITA (Parallel Implicit Time-integration Algorithm)

- First developed by Charbel Farhat (Aero-S, 2003) and further developed by Julien Cortial (2011)
- Time domain is decomposed and parallelized
- Scalable and iterative procedure
- When combined with an optimization the splitting enables derivatives to be computed
Outline

1. Brief Overview of PDE-constrained Optimization
2. PITA (Parallel Implicit Time-integration Algorithm)
3. “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)
4. Flapping Beam
5. Computational Results
Overview of PITOP

- Programmed in C++
- PITA code and an SQP are integrated. Currently use an unsophisticated SQP algorithm based on SNOPT
Given we have $N$ time slices for PITA

Optimization Formulation

$$\text{minimize} \quad F(y, u) = \sum_{i=0}^{N} \int_{\Omega} (y_i(x) - y_{i,target})^2 \, dx + \frac{\eta}{2} |u|^2$$

subject to \quad C(y, u) = 0

Constraints

$$C(y, u) = \begin{bmatrix}
    y_0 - v_0 \\
y_1 - g(y_0, \Delta T; u) \\
    \vdots \\
y_N - g(y_{N-1}, \Delta T; u)
\end{bmatrix}$$
Lagrangian and KKT optimality condition

The Lagrangian, \( L(y, u, \lambda) \), of the optimization formulation

\[
L(y, u, \lambda) = F(y, u) + \langle \lambda, C(y, u) \rangle
\]

KKT optimality condition

\[
L_y(y, u, \lambda) = F_y(y, u) + C^*_y(y, u)\lambda = 0 \quad \text{(adjoint equation)}
\]

\[
L_u(y, u, \lambda) = F_u(y, u) + C^*_u(y, u)\lambda = 0 \quad \text{(stationarity equation)}
\]

\[
C(y, u) = 0 \quad \text{(state equation)}
\]
Hereafter, we will denote $x_k = \begin{bmatrix} y_k \\ u_k \end{bmatrix}$

Search direction $s_k$ is given by solving

\[
\begin{align*}
\text{minimize} & \quad q_k(s) = L(x_k, \lambda_k) + \langle L_x(x_k, \lambda_k), s \rangle + \frac{1}{2} \langle s, B_k s \rangle \\
\text{subject to} & \quad C(x_k) + C_x(x_k)s = 0
\end{align*}
\]

$B_k$, Limited memory BFGS approximation
Reduced QP

- The QP can be reduced
- $\lambda_k$ is updated by satisfying adjoint equation (first-order accuracy)
- the nullspace matrix, $Z$, is orthogonal to the rows of $C_x$

$$Z = \begin{bmatrix} -(C_y)^{-1}C_u \\ l_u \end{bmatrix}$$

Reduced QP

$$\min_{s_u} \hat{q}_k(s_u) = L(x_k, \lambda_k) + \langle L_u(x_k, \lambda_k), s_u \rangle + \frac{1}{2} \langle s_u, Z^T B_k Z s_u \rangle$$

- $B_k$, Limited memory BFGS approximation
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
Algorithm - “PITOP”

1. Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
2. Choose initial seeds ($y_0$) and $u_0$
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$

Choose initial seeds ($y_0$) and $u_0$

Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$
Choose some appropriate \( \eta > 0, \epsilon > 0, \) and \( \nu > 0 \)

Choose initial seeds \((y_0)\) and \(u_0\)

Set \( \hat{B}_0 \) to be \( Z^T Z \) and set \( k = 0 \)

Run one iteration of PITA if \( \|C(x_k)\| > \nu \)
Algorithm - “PITOP”

- Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
- Choose initial seeds($y_0$) and $u_0$
- Set $\hat{B}_0$ to be $Z^TZ$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda)\| \approx 0$
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$

Choose initial seeds ($y_0$) and $u_0$

Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \approx 0$
3. Test for convergence
Algorithm - “PITOP”

- Choose some appropriate \( \eta > 0, \epsilon > 0, \text{ and } \nu > 0 \)
- Choose initial seeds(\( y_0 \)) and \( u_0 \)
- Set \( \hat{B}_0 \) to be \( Z^T Z \) and set \( k = 0 \)

1. Run one iteration of PITA if \( \|C(x_k)\| > \nu \)
2. Compute \( \lambda_k \) s.t. \( \|Ly(x_k, \lambda_k)\| \simeq 0 \)
3. Test for convergence
4. Compute \( s_{u,k+1} \) by solving the reduced QP and set \( \Delta x_k = Zs_{u,k+1} \)
Algorithm - “PITOP”

- Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
- Choose initial seeds($y_0$) and $u_0$
- Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \simeq 0$
3. Test for convergence
4. Compute $s_{u,k+1}$ by solving the reduced QP and set $\Delta x_k = Zs_{u,k+1}$
5. Compute step length, $\alpha_k$ with respect to $L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k)$ and $x_{k+1} = x_k + \alpha_k \Delta x_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k$
Algorithm - “PITOP”

- Choose some appropriate \( \eta > 0 \), \( \epsilon > 0 \), and \( \nu > 0 \)
- Choose initial seeds \((y_0)\) and \(u_0\)
- Set \( \hat{B}_0 \) to be \( Z^T Z \) and set \( k = 0 \)

1. Run one iteration of PITA if \( \|C(x_k)\| > \nu \)
2. Compute \( \lambda_k \) s.t. \( \|L_y(x_k, \lambda_k)\| \sim 0 \)
3. Test for convergence
4. Compute \( s_{u,k+1} \) by solving the reduced QP and set \( \Delta x_k = Zs_{u,k+1} \)
5. Compute step length, \( \alpha_k \) with respect to \( L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k) \)
   and \( x_{k+1} = x_k + \alpha_k \Delta x_k \), \( \lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k \)
6. Compute new Hessian update \( \hat{B}_k \), using limited memory BFGS
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$

Choose initial seeds ($y_0$) and $u_0$

Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \simeq 0$
3. Test for convergence
4. Compute $s_{u,k+1}$ by solving the reduced QP and set $\Delta x_k = Zs_{u,k+1}$
5. Compute step length, $\alpha_k$ with respect to $L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k)$
   and $x_{k+1} = x_k + \alpha_k \Delta x_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k$
6. Compute new Hessian update $\hat{B}_k$, using limited memory BFGS
   Set $k := k + 1$ and goto 1
**Algorithm - “PITOP”**

- Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
- Choose initial seeds($y_0$) and $u_0$
- Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \simeq 0$
3. Test for convergence
4. Compute $s_{u,k+1}$ by solving the reduced QP and set $\Delta x_k = Zs_{u,k+1}$
5. Compute step length, $\alpha_k$ with respect to $L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k)$ and $x_{k+1} = x_k + \alpha_k \Delta x_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k$
6. Compute new Hessian update $\hat{B}_k$, using limited memory BFGS
   Set $k := k + 1$ and goto 1
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$

Choose initial seeds($y_0$) and $u_0$

Set $\hat{B}_0$ to be $Z^T Z$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu$ → not a complete solve!

2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \simeq 0$

3. Test for convergence

4. Compute $s_{u,k+1}$ by solving the reduced QP and set $\Delta x_k = Zs_{u,k+1}$

5. Compute step length, $\alpha_k$ with respect to $L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k)$
   and $x_{k+1} = x_k + \alpha_k \Delta x_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k$

6. Compute new Hessian update $\hat{B}_k$, using limited memory BFGS
   Set $k := k + 1$ and goto 1
Choose some appropriate $\eta > 0$, $\epsilon > 0$, and $\nu > 0$
Choose initial seeds($y_0$) and $u_0$
Set $\hat{B}_0$ to be $Z^TZ$ and set $k = 0$

1. Run one iteration of PITA if $\|C(x_k)\| > \nu \rightarrow$ not a complete solve!
2. Compute $\lambda_k$ s.t. $\|L_y(x_k, \lambda_k)\| \simeq 0 \rightarrow$ PITA can be used again!!
3. Test for convergence
4. Compute $s_{u,k+1}$ by solving the reduced QP and set $\Delta x_k = Zs_{u,k+1}$
5. Compute step length, $\alpha_k$ with respect to $L_A(x_k + \alpha \Delta x_k, \lambda_k + \alpha \Delta \lambda_k)$ and $x_{k+1} = x_k + \alpha_k \Delta x_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \Delta \lambda_k$
6. Compute new Hessian update $\hat{B}_k$, using limited memory BFGS
Set $k := k + 1$ and goto 1
Adjoint Equation

\[ L_y(y, u, \lambda) = F_y(y, u) + C_y^*(y, u)\lambda = 0 \quad \text{(adjoint equation)} \]

\[ C_y = \begin{bmatrix} I & & & \vdots & & & I \\ -G^m & I & & & & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & -G^m & I \end{bmatrix} \]

\[ G = \begin{bmatrix} \frac{1-\alpha_m}{(1-\alpha_f)^2} \tilde{K}^{-1}M - \frac{\alpha_f}{1-\alpha_f}I & \Delta t \frac{1-\alpha_m}{1-\alpha_f} \tilde{K}^{-1}M \\ \frac{\gamma}{\Delta t \beta (1-\alpha_f)} \left[ \frac{1-\alpha_m}{1-\alpha_f} \tilde{K}^{-1}M - I \right] & \frac{\gamma}{\beta (1-\alpha_f)} \frac{1-\alpha_m}{1-\alpha_f} \tilde{K}^{-1}M - \frac{\alpha_f}{1-\alpha_f}I \end{bmatrix} \]
Adjoint Equation

\[ L_y(y, u, \lambda) = F_y(y, u) + C_y^*(y, u)\lambda = 0 \] (adjoint equation)

\[ C_y^* = \begin{bmatrix}
I & -(G^*)^m \\
I & \ddots \\
 & \ddots & -(G^*)^m \\
 & I & -(G^*)^m \\
I & & I
\end{bmatrix} \]
1 Brief Overview of PDE-constrained Optimization

2 PITA (Parallel Implicit Time-integration Algorithm)

3 “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)

4 Flapping Beam

5 Computational Results
\[ \theta = \sum u_i \sin(\omega_i \cdot t) \]

- \( y \): displacements and velocities at each node and each time slice
- \( u \): amplitudes which correspond to different frequencies
Outline

1. Brief Overview of PDE-constrained Optimization
2. PITA (Parallel Implicit Time-integration Algorithm)
3. “PITOP” (Parallel Implicit Time-integration Optimization Algorithm)
4. Flapping Beam
5. Computational Results
Convergence of state variable, $y$

History of state variables, $y$, at the tip of the beam
Convergence of control variable, $u$

History of control variables, $u$, at the hinge of the beam
Thank you