Which Method for Solution of the System of Interval Equations Should we Choose?

A. Pownuk¹, J. Quezada¹, I. Skalna², M.V. Rama Rao³, A. Belina⁴

¹ The University of Texas at El Paso, El Paso, Texas, USA
² AGH University of Science and Technology, Krakow, Poland
³ Vasavi College of Engineering, Hyderabad, India
⁴ Silesian University of Technology, Gliwice, Poland

21th Joint UTEP/NMSU Workshop on Mathematics, Computer Science, and Computational Sciences
Solution of PDE

Parameter dependent Boundary Value Problem

\[ A(p)u = f(p), \quad u \in V(p), \quad p \in P \]

Exact solution

\[ \underline{u} = \inf_{p \in P} u(p), \quad \bar{u} = \sup_{p \in P} u(p) \]

\[ u(x, p) \in [\underline{u}(x), \bar{u}(x)] \]

Approximate solution

\[ \underline{u}_h = \inf_{p \in P} u_h(p), \quad \bar{u}_h = \sup_{p \in P} u_h(p) \]

\[ u_h(x, p) \in [\underline{u}_h(x), \bar{u}_h(x)] \]
High dimension $n > 10000$.
Linear and nonlinear equations.
Multiphysics (solid mechanics, fluid mechanics etc.)
Ordinary and partial differential equations, variational equations, variational inequalities, numerical methods, programming, visualizations, parallel computing etc.
Two point boundary value problem

Sample problem

\[
\begin{aligned}
\left\{ \\
-(a(x)u'(x)) &= f(x) \\
u(0) &= 0, \quad u(1) = 0
\end{aligned}
\]

and \( u_h(x) \) is finite element approximation given by a weak formulation

\[
\int_0^1 a(x)u_h'(x)v'(x)dx = \int_0^1 f(x)v(x)dx, \quad \forall v \in V_h^{(0)}
\]

or

\[
a(u_h, v) = l(v), \quad \forall v \in V_h^{(0)} \subset H^1_0
\]

where \( u_h(x) = \sum_{i=1}^n u_i \varphi_i(x) \) and \( \varphi_i(x_j) = \delta_{ij} \).
The Finite Element Method

Approximate solution
\[ \int_0^1 a(x)u'_h(x)v'(x)dx = \int_0^1 f(x)v(x)dx. \]

\[ \sum_{j=1}^{n} \left( \sum_{i=1}^{n} \int_0^1 a(x)\varphi_i(x)\varphi_j(x)dxu_i - \int_0^1 f(x)\varphi_j(x)dx \right) v_j = 0 \]

Final system of equations (for one element) \( Ku = q \) where

\[ K_{i,j} = \int_0^1 a(x)\varphi_i(x)\varphi_j(x)dx, \quad q_i = \int_0^1 f(x)\varphi_i(x)dx \]

Calculations of the local stiffness matrices can be done in parallel.
Global stiffness matrix

\[ \sum_{p=1}^{n} \left( \sum_{q=1}^{n} \sum_{e=1}^{n_e} \sum_{i=1}^{n_u^e} \sum_{j=1}^{n_u^e} U_{j,p}^e \int_{\Omega_e} a(x) \frac{\partial \varphi_i^e(x)}{\partial x} \frac{\partial \varphi_j^e(x)}{\partial x} \, dx U_{i,q}^e u_q - \right) \]

\[ \sum_{q=1}^{n} \sum_{e=1}^{n_e} \sum_{i=1}^{n_u^e} \sum_{j=1}^{n_u^e} U_{j,p}^e \int_{\Omega_e} f(x) \varphi_i^e(x) \varphi_j^e(x) \, dx \right) \nu_p = 0 \]

Final system of equations

\[ K(p)u = Q(p) \Rightarrow F(u, p) = 0 \]

Computations of the global stiffness matrix can be done in parallel.
Solution Set

Nonlinear equation $F(u, p) = 0$ for $p \in P$.

$$F : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$$

Implicit function $u = u(p) \iff F(u, p) = 0$

$$u(P) = \{u : F(u, p) = 0, p \in P\}$$

Interval solution

$$u_i = \min\{u : F(u, p) = 0, p \in P\}$$

$$\bar{u}_i = \max\{u : F(u, p) = 0, p \in P\}$$
Interval Methods


Optimization methods

Interval solution

\[
\bar{u}_i = \min\{u(p) : p \in P\} = \min\{u : F(u, p) = 0, p \in P\}
\]

\[
\underline{u}_i = \max\{u(p) : p \in P\} = \max\{u : F(u, p) = 0, p \in P\}
\]

\[
\underline{u}_i = \begin{cases} 
\min u_i \\
F(u, p) = 0, p \in P
\end{cases}, 
\bar{u}_i = \begin{cases} 
\max u_i \\
F(u, p) = 0, p \in P
\end{cases}
\]
KKT Conditions

Nonlinear optimization problem for $f(x) = x_i$

$$\min_x f(x) \quad \begin{cases} h(x) = 0 \\ g(x) \geq 0 \end{cases}$$

Lagrange function $L(x, \lambda, \mu) = f(x) + \lambda^T h(x) - \mu^T g(x)$

Optimality conditions can be solved by the Newton method.

$$\begin{cases} \nabla_x L = 0 \\ \nabla_{\lambda} L = 0 \\ \mu_i \geq 0 \\ \mu_i g_i(x) = 0 \\ h(x) = 0 \\ g(x) \geq 0 \end{cases}$$
KKT Conditions - Newton Step

\[ F'(X) \Delta X = -F(X) \]

\[ F'(X) = \begin{bmatrix} \left( \nabla_x^2 f(x) + \nabla_x^2 h(x)y \right)_{n \times n} & \nabla_x h(x)_{n \times m} & -I_{n \times n} \\ \left( \nabla_x h(x) \right)^T_{m \times n} & 0_{n \times m} & 0_{n \times m} \\ Z_{n \times n} & 0_{n \times m} & X_{m \times n} \end{bmatrix} \]

\[ \Delta X = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}, \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \]

\[ F(X) = -\begin{bmatrix} \nabla_x f(x) + \nabla_x h^T(x)y - z \\ h(x) \\ XYe - \mu_ke \end{bmatrix} \]
Steepest Descent Method

In order to find maximum/minimum of the function $u$ it is possible to apply the steepest descent algorithm.

1. Given $x_0$, set $k = 0$.
2. $d^k = -\nabla f(x_k)$. If $d^k = 0$ then stop.
3. Solve $\min_\alpha f(x_k + \alpha d^k)$ for the step size $\alpha_k$. If we know second derivative $H$ then $\alpha_k = \frac{d_k^T d_k}{d_k^T H(x_k) d_k}$.
4. Set $x_{k+1} = x_k + \alpha_k d_k$, update $k = k + 1$. Go to step 1.

After discretization

\[ Ku = q \]

Calculation of the gradient

\[ Kv = \frac{\partial}{\partial p_k} q - \frac{\partial}{\partial p_k} Ku \]

where \( v = \frac{\partial}{\partial p_k} u \).
Gradient Method and Sensitivity Analysis


Postprocessing of the interval solution.

\[ \varepsilon = Cu \]

\[ \sigma = D\varepsilon \]
Linearization

\[ \Delta f(x) = f(x + \Delta x) - f(x) \approx f'(x)\Delta x \]

Derivative can be calculated numerically.

\[ f'(x) \approx \frac{f(x + h) - f(x)}{h} \]

The method can be used together with incremental formulation of the Finite Element Method.

\[ K(p)\Delta u = \Delta Q(p) \]
Monte Carlo Method (inner approximation of the solution set)

\[ u(P) \approx \text{Hull}(\{ u : K(p)u = Q(p), p \in \{ \text{random values from } P \} \} ) \]

Search Method. \( P \approx \{ \text{special points} \} \)

\[ u(P) \approx \text{Hull}(\{ u : K(p)u = Q(p), p \in \{ \text{special points} \} \} ) \]

Vertex Method

\[ u(P) \approx \text{Hull}(\{ u : K(p)u = Q(p), p \in \{ \text{set of vertices} \} \} ) \]
Cauchy Based Monte Carlo Simulation

\[ \rho_{\Delta}(x) = \frac{\Delta}{\pi} \cdot \frac{1}{1 + x^2/\Delta^2}. \]

when \( \Delta x_i \sim \rho_{\Delta_i}(x) \) are indep., then

\[ \Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i \sim \rho_{\Delta}(x), \text{ with } \Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i. \]

Thus, we simulate \( \Delta x_i^{(k)} \sim \rho_{\Delta_i}(x) \); then,

\[ \Delta y^{(k)} \overset{\text{def}}{=} \tilde{y} - f(\tilde{x}_1 - \Delta x_1^{(k)}, \ldots) \sim \rho_{\Delta}(x). \]

Maximum Likelihood method can estimate \( \Delta \):

\[ \prod_{k=1}^{N} \rho_{\Delta}(\Delta y^{(k)}) \rightarrow \max, \text{ so } \sum_{k=1}^{N} \frac{1}{1 + (\Delta y^{(k)})^2/\Delta^2} = \frac{N}{2}. \]

To find \( \Delta \) from this equation, we can use, e.g., the bisection method for \( \underline{\Delta} = 0 \) and \( \overline{\Delta} = \max_{1 \leq k \leq N} |\Delta y^{(k)}| \).
Theory of perturbations


\[ A = A_0 + \varepsilon^1 A_1 + \varepsilon^2 A_2 + \ldots \]

Interval Boundary Element Method


\[ cu = \int_{\partial \Omega} \left( G \frac{\partial u}{\partial n} - \frac{\partial G}{\partial n} u \right) dS \]

\[ \Pi^* = \frac{1}{2} \{U\}^T \{K\} \{U\} - \{U\}^T \{P\} + \lambda_1^T \{C\} \{U\} - \{F\} + \lambda_2^T \{B_1\} \{U\} - \{k\} \]  \hspace{1cm} (40)

Invoking the stationarity of \( \Pi^* \), that is \( \delta \Pi^* = 0 \), and considering Eq. (40), we obtain

\[
\begin{pmatrix}
0 & C^T & B_1^T & 0 \\
C & 0 & 0 & 0 \\
B_1 & 0 & 0 & -I \\
0 & 0 & -I & 0 \\
\end{pmatrix}
\begin{bmatrix} A \\ 0 \\ 0 \\ 0 \end{bmatrix}
+ 
\begin{bmatrix} \lambda_2 \\ \lambda_2 \\ \kappa \end{bmatrix}
= 
\begin{bmatrix} P_C \\ 0 \\ 0 \\ 0 \end{bmatrix}
+ 
\begin{bmatrix} M \\ 0 \\ 0 \end{bmatrix}
\]  \hspace{1cm} (41)

\[
\begin{align*}
\text{Table I. Algorithm} \\
R &:= \text{mid}(A([p]))^{-1}; \\
\bar{x} &:= R \cdot \text{mid}(b([p])); \\
[Z]_i &= \sum_{j=1}^{n} R_{ij} \left( \omega(0, j) - \sum_{k=1}^{n} \bar{x}_k \omega(j, k) \right)^{T}[p] \\
[D]_{ij} &:= \left( \sum_{\nu=1}^{n} R_{i\nu} \omega(\nu, j) \right)^{T}[p]; \\
\text{outer} &:= \bar{x} + [-1, 1] \langle[D]\rangle^{-1} ||Z|| 
\end{align*}
\]
The use of diagonal matrix


\[ K = A^T \ast D \ast A \]
M. V. Rama Rao, R. L. Muhanna, and R. L. Mullen. Interval Finite Element Analysis of Thin Plates 7th International Workshop on Reliable Engineering Computing, At Ruhr University Bochum, Germany, 2016

\[
[K] = \begin{bmatrix} K_1^{(e)} & K_2^{(e)} & \cdots \\ K_2^{(e)} & K_2^{(e)} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} A_1^{(e)} & A_2^{(e)} & A_3^{(e)} & \cdots \end{bmatrix} \begin{bmatrix} \text{diag}(\Lambda_1\alpha_1) \\ \text{diag}(\Lambda_2\alpha_2) \\ \text{diag}(\Lambda_3\alpha_3) \\ \cdots \end{bmatrix} \begin{bmatrix} A_1^T^{(e)} \\ A_2^T^{(e)} \\ A_3^T^{(e)} \\ \cdots \end{bmatrix}
\]

This can be denoted as

\[
[K] = [A][D][A]^T
\]
Comparison between the different methods

\[
\text{Comp. Complexity}(\text{Method 1}) < \text{Comp. Complexity}(\text{Method 2})
\]

\[
\text{Accuracy}(\text{Method 1}) < \text{Accuracy}(\text{Method 2})
\]

Accuracy includes also information about guaranteed accuracy.

\[
\text{PossibleApplications}(\text{Method 1}) < \text{PossibleApplications}(\text{Method 2})
\]

\[
\text{Scalability}(\text{Method 1}) < \text{Scalability}(\text{Method 2})
\]

Scalability includes information about parallelization.
How to find the best method?

Example:
method 1: linearization
method 2: Monte Carlo

The problem is small

\[
\text{EasyToImplement}(\text{Method1}) < \text{EasyToImplement}(\text{Method2})
\]
\[
\text{Accuracy}(\text{Method1}) < \text{Accuracy}(\text{Method2})
\]

Better method is the method 2, i.e. the Monte Carlo method.
What to do in the conflict situations?

Example:
method 1: linearization
method 2: interval methods

\[
\text{Comp. Complexity}(\text{Method1}) < \text{Comp. Complexity}(\text{Method2})
\]

\[
\text{Accuracy}(\text{Method1}) > \text{Accuracy}(\text{Method2})
\]

If the main requirement is guaranteed solution, then we can use the interval methods.
What to do in the conflict situations?

Example:
method 1: linearization
method 2: interval methods

\[ \text{Comp. Complexity}(\text{Method } 1) < \text{Comp. Complexity}(\text{Method } 2) \]

\[ \text{Accuracy}(\text{Method } 1) > \text{Accuracy}(\text{Method } 2) \]

If the problem is very large or nonlinear, then it is not possible to apply the interval methods and it is necessary to use linearization.
What to do in the conflict situations?
Linear model

Example:
method 1: $m_1$
method 2: $m_2$
Total score

$$\mu_1 = \sum_i w_i f_i(m_1)$$
$$\mu_2 = \sum_i w_i f_i(m_2)$$

If $\mu_1 > \mu_2$ then we need to pick the method 1.
If $\mu_1 < \mu_2$ then we need to pick the method 2.
What to do in the conflict situations? Nonear model

Example:
method 1: \( m_1 \)
method 2: \( m_2 \)
Total score

\[
\begin{align*}
\mu_1 &= \Phi(f_1(m_1), f_2(m_1), \ldots, f_k(m_1)) \\
\mu_2 &= \Phi(f_1(m_1), f_2(m_1), \ldots, f_k(m_1))
\end{align*}
\]

If \( \mu_1 > \mu_2 \) then we need to pick the method 1.
If \( \mu_1 < \mu_2 \) then we need to pick the method 2.

or more generally

\[
\Omega(f_1(m_1), \ldots, f_k(m_1), f_1(m_2), \ldots, f_k(m_2)) > 0
\]
Conclusions

- Interval equations can be solved by using many different methods.
- Every method has some advantages and disadvantages.
- In order to choose the optimal method it is necessary to consider many different features of every computational method.