I. EXPERIMENTAL RESULTS

III. COMPOSITION OF THE ALGORITHM

II. INTRODUCTION TO INTERVAL NEWTON METHODS

I. INTRODUCTION TO INTERVAL ARITHMETIC

In as little as 1/3 to 1/4 the time required by Krawczyk's method, we explain why each sub-algorithm is desirable and how they fit together to provide solutions.

- Solution of the Linearized Equations by Elimination
- A Real (Non-Interval) Newton Iteration
- A Gauss-Seidel Type Step

II. INTERVALS THREE SUB-ALGORITHMS:

We introduce an interval Newton method for bounding solutions of systems of nonlinear equations.
E. COMPUTER IMPLEMENTATIONS

D. GENERAL FUNCTIONS AND DIRECTED ROUNDDING

C. EXTENDED INTERVAL ARITHMETIC

B. THE FOUR ARITHMETIC OPERATIONS

A. MOTIVATION FOR INTERVAL ARITHMETIC

1. INTERVAL ARITHMETIC
LIMITED ACCURACY OF ITS FINE DIGIT ANSWER.

SLIGHTLY DIFFERENT DUE TO BASE CONVERSIONS.' THE COMPUTER GIVES NO INDICATION OF THE
BUT THE COMPUTER WILL GIVE A FINE DIGIT ANSWER, E.G., Z=10000E-4.' THE ANSWER MAY BE
SINCERE WE KNOW NOTHING ABOUT X AND Y AFTER THE FIFTH DIGIT, WE CAN SAY ONLY THAT 0.2 X 10^-4

0.00010 4 \times 10^{-4}
- 0.99886
  0.00114

COMPUTE Z=X-Y. BY HAND WE GET
FOR EXAMPLE, ON A FINE DECIMAL DIGIT MACHINE WITH X=99999 AND Y=99998, SUPPOSE WE
- LOSS OF LEADING SIGNIFICANT DIGITS IN SUBTRACTION OF NEARLY EQUAL VALUES (CANCELLATION)
- LIMITED ACCURACY OF INPUT DATA
- ROUNDING

PRIMARY SOURCES OF ERROR ARE:

**AUTOMATIC ERROR ANALYSIS**

A. MOTIVATION FOR INTERVAL ARITHMETIC
CALCULATED CAN BE REDUCED BY TESTING THE SIGNS OF THE INTERVAL ENDPOINTS.

IN MULTIPLICATION AND DIVISION THE NUMBER OF PRODUCTS OR QUOTIENTS

\[ \frac{A}{B} \text{ SIMILAR} \]

\[ \left[ p_{f} \right] \leq \left[ p_{f} \right] \leq \left[ p_{f} \right] \leq \left[ p_{f} \right] \]

\[ a \cdot b = \min \{ a, b \} \cdot \max \{ a, b \} \]

\[ a \cdot b = \left\{ \begin{array}{ll}
A - B & \text{if } a, b \in \mathbb{R}, \text{ and both positive or both negative} \\
A - B & \text{if } a, b \in \mathbb{R}, \text{ and one is positive and one is negative} \\
A + B & \text{if } a, b \in \mathbb{R}, \text{ and both positive or both negative} \\
A + B & \text{if } a, b \in \mathbb{R}, \text{ and one is positive and one is negative}
\end{array} \right. \]

\[ b \left[ \begin{array}{c}
p_{f} \\
p_{f} \\
p_{f} \\
p_{f}
\end{array} \right] = \left[ \begin{array}{c}
p_{f} \\
p_{f} \\
p_{f} \\
p_{f}
\end{array} \right] \]

B. THE FOUR ARITHMETIC OPERATIONS
\[ A / B = \begin{cases} 
[-\infty, B^r] & \text{if } A^r < 0 \text{ and } B^r = 0 \\
[A^l/B^l, +\infty] & \text{if } A^l > 0 \text{ and } B_l = 0 \\
[A^l/B^l] & \text{if } A^l > 0, \ B_l < 0, \text{ and } B^r > 0 \\
[-\infty, A^l/B^l] & \text{if } A^r < 0 \text{ and } B^r = 0 \\
[-\infty, A^l/B^l, +\infty] & \text{if } A^l > 0, \ B_l < 0, \text{ and } B^r > 0 \\
\end{cases} \]

A = \{[A^l, A^r]\}

B = \{[B^l, B^r]\}

C. EXTENDED INTERVAL ARITHMETIC
\[ \left[ 0,0 \right] + \left[ 0,1 \right] = \left[ 0,1 \right] \]

AND
\[ \left[ 0,0 \right] + \left[ 1/3,4 \right] = \left[ 1/3,4 \right] \]

LETTING \( x = 1,3 \) WE CAN CALCULATE

FOR EXAMPLE, \((x = 0,1) / x \) SHOULD BE WRITTEN AS \( 1 + [0,1] / x \).

EXPRESSION CAN MAKE A SUBSTANTIAL IMPROVEMENT.

LEAD TO OVERLY PESSIMISTIC NUMERICAL RESULTS. OFTEN SIMPLE REARRANGEMENT OF AN ARITHMETIC

PESSIMISTIC. IT IS A CHALLENGE IN INTERVAL ANALYSIS TO FIND ALGORITHMS WHICH DO NOT

THOUGH WE ARE SURE TO GET RESULTS BACKETING THE CORRECT ANSWER, OUR RESULT MAY BE

RIGHT ENDPOINT UP.

ARITHMETIC OPERATION IS PERFORMED, WE ROUND THE LEFT ENDPOINT OF THE RESULT DOWN AND THE

RESULT OF OUR CALCULATION IS AN INTERVAL CONTAINING THE DESIRED RESULT. AS EACH

COMPUTATIONS WITH A FINITE ACCURACY MACHINE, WE MUST PERFORM DIRECTED ROUNDING TO INSURE

OPERATIONS, WE CAN BE SURE THAT \( f(x) \text{ for } x \in [x_1, x] \) WHERE \( x = \left[ x_1, x \right] \). IN ACTUAL

USING EXACT INTERVAL ARITHMETIC TO EVALUATE ANY FUNCTION \( f \) INVOLVING THE FOUR BASIC

D. GENERAL FUNCTIONS AND DIRECTED ROUNDING
Compilers developed at the University of Minnesota Computing Center.

4) Additional variable type "XOTHER" added to the MNP and MM FORTRAN

3) Intel 8087 or 432 microprocessors

2) Precompiler

1) Kluge

E. Computer Implementations
\[ \text{where } m = b \cdot j(x) \text{ and } p = -bf(x). \]

Letting \( b \) denote an approximate inverse, the multiplication yields

\[ x_{(k+1)} = x_{(k)} \cdot n(x_{(k)}). \]

Given a box \( x(0) \), new iterates are obtained as

\[ f(x) + j(x)(z - x) = 0. \]

If \( z \) is contained in the set of points satisfying notation \( \mathcal{E} \) and contained in \( x \), thus \( \mathcal{E} \) is contained in the set of points satisfying 

\[ F(x) + j(e)(x - y) = 0. \]

By Taylor's theorem:

\[ \text{Let } y \text{ be a zero of } f. \]

Consider a continuously differentiable function \( f: \mathbb{R}^n \times \mathbb{R}^n \) with Jacobian \( J \) and

II. Introduction to Interval Newton Methods
INNER ITERATION REDUCES THE NUMBER OF TIMES \( f(x), b \), AND \( b(x) \) MUST BE COMPUTED.

REAL ITERATION IS EMPLOYED TO IMPROVE THE VALUE OF \( \alpha \) BEFORE ELIMINATION IS ATTEMPTED.

SO THAT POOR RESULTS MAY BE OBTAINED.

MATRICES AND MORE IMPORTANTLY, INTERVALS TEND TO GROW DURING THE ELIMINATION PROCESS.

BOUND FOR \( Z \), BUT THIS METHOD CANNOT ALWAYS BE EMPLOYED SINCE \( M \) MAY CONTAIN A SINGULAR

ELIMINATION METHOD: SOLVE THE EQUATION BY GAUSSIAN ELIMINATION, SEeks \( Z \) RATHER THAN \( A \).

- HANSEN-SENEFFA METHOD: SAME APPLIES, BUT CONVERGENCE SUPERIOR TO KRAMZVAY METHOD.

- KRAMZVAY METHOD: DOES NOT ATTEMPT TO OBTAIN \( Z \), BUT ONLY A BOUND FOR \( \alpha \).

INTERVAL NEWTON METHODS DIFFER IN HOW THIS EQUATION IS SOLVED.

\[ M = B \rangle (X) \text{ AND } B = BF(x) \]
\[ M(2-x) = q \]
E. Splitting the Box

D. Additioanal Hansen-Sen Gupta Iteration

C. Elimination Iteration

B. Real Iteration

A. Initial Hansen-Sen Gupta Step

III. Composition of the Algorithm
Here we may obtain a "gap" which may be used later to split the box. We then proceed to deal with those components for which $O^m_i$ in a similar manner.

- $X_i = \frac{X_i^f}{y_i}$

Replace $X_i$ by $X_i^f$ in $Z_i$. We perform the calculations first for those $i$ such that $O_m^m_i$, and in each case, we have replaced $Z_i$ by $Y_i$ for all $i \neq i$. The interval $Y_i$ contains every solution.

\[
    R_i = \frac{b_i - (\sqrt[3]{m_i} - x_i) - \frac{m_i}{3}}{\xi}
\]

Where

\[
    Y = x_i + R_i
\]

1-th variable. That is one Hansen-Sengupta step. We wish to solve the 1-th equation of $M(x) = b$ for the after calculating $j$ and $b$, and performing the multiplication by $b$, we first perform

A. Initial Hansen-Sengupta Step
\[ x_{k+1} = x_k - B f(x_k) \]

By starting at \( x(0) = m(x) \) and computing

- We attempt to find an improved approximation \( x \) for \( a \) zero of \( f \) if \( x \) exists.

By real iteration
CALCULATION OF $d = -bf(x)$ WITH THE NEW $x$, TO 9 OF THE PREVIOUS WIDTH, SAY. WE REPEAT THE ELIMINATION STEP, STARTING AT THE
- EACH TIME THE WIDTH OF $x$ (THE WIDTH OF ITS WIDEST COMPONENT) DECREASES SIGNIFICANTLY
- OF THE NEW $x$.

When we intersect our solution with $x$ and calculate a NEW $x$ AT THE CENTER
$M(x) = b$. THEN WE INTERSECT OUR SOLUTION WITH $x$ AND CALCULATE A NEW $x$ AT THE CENTER.

IF SUCCESSFUL, WE CALCULATE $d = -bf(x)$ AND PERFORM FORWARD AND BACK SUBSTITUTION TO SOLVE
LU DECOMPOSITION OF $M$.

IF WE HAVE FOUND AN $x$ SUCH THAT $||f(x)||$ IS SUFFICIENTLY SMALL, WE ATTEMPT TO PERFORM AN

C. ELIMINATION ITERATION
THE HANSEN-SEN Gupta Step
- Each time x improves significantly, we calculate a new x at its center and repeat

only for those i such that $\Phi_i$.

$x_i^{t+1} = x_i^{t}$

$y_i^{t+1} = y_i^{t} + R_i^{t+1}$

WHERE $R_i^{t+1} = R_i^{t} - \sum_{j \neq i} f(x_i^{t} - x_j^{t})$

THE CALCULATIONS

- Since we have already done our best to find a wide gap in the box $x$, we perform

INTERACTION instead.

- If we are unable to complete an elimination step, we perform Hansen-Sen Gupta Iteration

D. ADDITIONAL HANSEN-SEN Gupta Iteration
The calculation of the Jacobian matrix.

Whether or not we split the box, we now return to the beginning of the outer step.

Stack for later processing.

When the box is split, one part becomes the current box, and the other is placed on a stack for later processing.

At any stage, we split the box at the center of its widest component.

If we have found no gap and we have not managed to improve the box width significantly, we make use of that gap to split the box (if it still does so).

If we have found a gap using the initial Hansen-Sengupta step described earlier, E. splitting the box.
1. When we encounter an empty stack, we are done.
2. If in steps 5 and 6, in step 11, or in step 14, the box is narrowed
   to within the acceptable tolerance or is found not to contain a solution,
   improve significantly in steps 5 and 6, in step 11, or in step 14, the box did
   not improve significantly in steps 5 and 6, in step 11, one part on the stack,
   and keep the other part as the new X.
3. If we have a gap saved from step 6, use it. If the box splits, put
   one part on the stack, and keep the other part as the new X.
4. If X improved significantly, set \( x = \) equal to the center of \( x \), and
   perform a Hansen-Sengupta step for those components such that \( \Omega \|_{x} \).
5. Skip to step 16.
6. Save the largest gap.
7. If \( \| f(x) \|_{\text{small}} \) is not sufficiently small, skip to step 14.
8. If \( \| f(x) \|_{x} \) is not sufficiently small, skip to step 14.
9. Perform the LU decomposition of \( M \), if possible. Otherwise, skip to
   step 14.
10. Solve for \( z \) by forward and back substitution.
11. Replace \( x \) by \( x + z \).
12. If the width of \( x \) improved significantly, set \( x = \) equal to the
    center of \( x \), and
13. Skip to step 16.
14. Perform a Hansen-Sengupta step for those components such that \( \Omega \|_{x} \).
15. If \( x \) improved significantly, set \( x = \) equal to the center of \( x \), and
17. If the box did not improve significantly in steps 5 and 6, in step 11,
18. Return to step 1.

19. When the most recently stacked box becomes the new X, and we return to step 1.

F. SUMMARY OF ALGORITHM
That the solution box have a width less than 10.

The initial box was \([-1, 1]\) in each component and contains one solution. It was required

\[
\{ (1, i) = \begin{cases} 1, & \max (1, i - 5) \leq \min (u, i + 1) \\ \text{where } u \end{cases} 
\]

\[
I(x) = \sum_{i=1}^{n} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{2\pi} e^{-\frac{x^2 + 2\pi^2}{2}} dx
\]

We show representative results for the Brodsky banded function.

Interval arithmetic matrix product.

It is to calculate the Jacobian, the inverse of its center, and the necessary

difficulty is to apply some control of how much inner iteration is performed. According to how

factor representing significant improvement of the box width, this allows the user

in the final version of our algorithm as described above, it is possible to vary the

and, for comparison, various versions of the Krawczyk method.

Several problems were solved using various versions of the algorithm described above

IV. EXPERIMENTAL RESULTS
The time is minutes:seconds. The experiments were run on the slow HP 9845.

\[ N_{HS2} = \text{The no. of Hansen-Sengupta steps for those i with } \Omega \text{.} \]

\[ N_{HS1} = \text{The no. of Hansen-Sengupta steps for those i with } \Omega \text{.} \]

\[ N_{E} = \text{The no. of elimination steps} \]

\[ N_{U} = \text{The no. of interval arithmetic lu decomposition attempts} \]

\[ N_{M} = \text{The no. of interval arithmetic lu decomposition attempts} \]

\[ N_{R} = \text{The no. of real iterations} = \text{the no. of real function evaluations} = \text{the no. of real} \]

\[ N_{K} = \text{The no. of krawczyk steps} \]

\[ N_{M} = \text{The no. of matrix-vector products} = N_{K} + N_{E} + N_{HS1} \]

\[ N_{F} = \text{The no. of interval arithmetic function evaluations} = \text{the no. of interval arithmetic} \]

\[ N_{O} = \text{The no. of outer steps} = \text{the no. of jacobian evaluations} = \text{the no. of matrix} \]

\[ S = \text{The significant box width improvement factor} \]

The following values of interest are tabulated for the sample runs:
\[
\begin{array}{ccc}
23:17 & 06:46 & 09:00 \\
\text{time} & \text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=5 \), New Method

\[
\begin{array}{ccc}
30:03 & 08:00 & 09:00 \\
\text{time} & \text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=5 \), Hansen Sen Gupta Method Without Inner Iteration

\[
\begin{array}{ccc}
80:14 & 19:41 & 09:00 \\
\text{time} & \text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=5 \), Krawczyk Method With Inner Iteration

\[
\begin{array}{ccc}
2:28 & 12:12 & 09:49 \\
2:28 & 12:12 & 09:49 \\
2:29 & 13:08 & 09:48 \\
2:31 & 13:06 & 09:46 \\
\text{time} & \text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=3 \), New Method

\[
\begin{array}{ccc}
2:37 & 21:08 \\
\text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=3 \), Hansen Sen Gupta Method Without Inner Iteration

\[
\begin{array}{ccc}
11:48 & 36:36 & 09:51 \\
9:52 & 44:44 & 08:58 \\
7:30 & 46:46 & 07:57 \\
8:34 & 57:57 & 06:56 \\
\text{time} & \text{time} & \text{time} \\
\hline
\end{array}
\]

\( n=3 \), Krawczyk Method With Inner Iteration

\text{Time is minutes:seconds.} \\
\text{No represents the no. of outer steps.} \\
\text{S represents the significant box improvement factor.}
Programs with many function calls may be 12 times slower.

Slower than ordinary single precision implementations dominated by arithmetic operations.

Another approach taken involves using routine procedures to perform the desired operations. Easy to use, 4 to 6 times faster. Additionally, variables of type 'XOTHER' added to the MN and M7 FORTRAN compilers developed at the University of Minnesota Computing Center. Whenever the compiler encounters operations involving variables of type 'XOTHER,' a variable of type 'XOTHER' is added to the MN and M7 FORTRAN compilers.

Additional variable type 'XOTHER' added to the MN and M7 FORTRAN compilers.

Hardware.

Introduction of implementing internal arithmetic, but a compiler does not yet exist to exploit this new version of implementing internal arithmetic. But a compiler has already been developed.

Intel 8087 or 422 microprocessors - Intel has developed chips to perform direct rounding with the single precision version of the program.

(1) An external library of internal subroutine. Execution speed is 19 to 92 times slower than the precompiler - there does exist a precompiler which translates FORTRAN to include the requisite calls.

(2) Kluge - a subroutine call for every arithmetic operation. Laborous programming and slow execution.

(3) Intel 6807 or 422 microprocessors - Intel has developed chips to perform direct rounding with the single precision version of the program.