10.0 Arrays in User Programming

- Both adjustable and static arrays (variables) are now supported via user programs.
- Up to 200 (total) unique adjustable and static arrays allowed for all user programs.
- Array data can be stored-to and read-from ASCII files. Maximum of Save and Load commands for all user programs is 200.

Array Format and Size

- All arrays are one dimensional. Index mapping is the responsibility of the user.

$$\begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ . \\ . \\ e_n \end{pmatrix}$$

- Each element (e_n) is a double precision floating point number (8 bytes).
- Index starts at 1 (not 0). Size is heap limited.

Defining Arrays

- Adjustable arrays:

Command: ADEFA or Array_Define_Adjustable

```
ADEFA Name Size ; "filename"
```

Name	= name of array
Size	= number of array elements
"filename"	= file containing initialization values (optional)

```
ADEFA my_array 100 ; "energy.dat"
```

Defining Arrays

- Static arrays:

Command: ADEFS or Array_Define_Static

```
ADEFS Name Size ; "filename"
```

Name	= name of array
Size	= number of array elements
"filename"	= file containing initialization values (optional)

```
ADEFS my_array 100 ; "energy.dat"
```

Addressing Arrays

- Array elements are called with ARCL and stored with ASTO commands.

When using ASTO, the *value* is held in the y register, the index in the x register:

```
7                   ; value in y register
50 ASTO my_array    ; placed at index 50 in my_array
                    ; stack rolled up, value (7) now
                      in x register
```

When using ARCL, the index is placed in the x register and ARCL replaces this index with the *value*.

Lifetime of Array Variables

- Adjustable array variables:
  - Are initialized to zero (or from a file) before any Initialize program segments are called.
  - Values are retained throughout the period of ion flying. Globally visible.
- Static array variables:
  - Are initialized to zero (or from a file) immediately before each ion (or group) begins to fly.
  - Values are retained only until the end of the flight. Globally visible.

Array Initialization Options

- All array elements (adjustable and static) are first initialized to zero as described previously.
- A file containing array initialization data can be loaded after the array has been pre-zeroed.

  ALOAD *array_name* ; "filename"

  (the name of the file comes after the semicolon and is inside quotation marks)

Array Initialization Options

- SIMION will read only the number of values needed to fill the array (defined by *size* in ADEF).
- If there are fewer values in the file than the defined array size, the remaining values in the array will be zero.
- Files are free format ASCII, numbers separated by spaces or commas, blank lines allowed, the ";" recognized as start of an inline comment. Maximum line length 200 characters.
- Errors in file format are trapped and flagged.

Array File Saving Options

- ASAVE or Array_Save command.

  ```
  ASAVE my_array ; "filename"
  ```

  Saves contents of array my_array to file named "filename"

- If "filename" already exists, contents will be destroyed and new data inserted (not appended).
- Format of file is ASCII with 5 coma-separated numbers per line.

  ```
  1,2,3,4,5
  3,5,7,6,8
  ```

Array Summary

Array Types:

• ADEFA	Defines Adjustable Array
• ADEFS	Defines Static Array
• ALOAD	Load array with values from an ASCII file
• ASAVE	Save array contents to an ASCII file
• ASTO	Store a number into an array element
• ARCL	Recall a number from an array element

The Init_P_Values Segment

New User Program Segment

- Used to initialize an entire fast-adjust array (e.g. .PA0) before flying any ions.
- Used in place of fast_adjust to increase speed (when appropriate).
  - Best for flying large number of ions in small arrays.
  - <u>Only applicable when fields *do not change* during ion flying</u>.

The Init_P_Values Segment

- Unlike all other program segments, ions do not have to be in the instance for Init_P_Values to be called.
- This means ion and instance context have no meaning within an Init_P_Values segment.
- Can access only these reserved variables:
  - Adj_Elect00 - Adj_Electro30
  - Adj_Pole00 - Adj_Pole30

The Init_P_Values Segment

- No instance related coordinate transformations allowed (>WBC etc).
- Can retain changed potentials at end of fly'm using the reserved variable:

  Retain_Changed_Potentials

  by saving a non-zero value to this variable within any Terminate segment.

Init_P_Values Example

```
Defa Tune_Voltage  100          ; tuning voltage default
Seg_Init_P_Values               ; beginning of segment
        RCL Tune_Voltage        ; get tuning voltage
        STO Adj_Elect01         ; store in fast adjust electrode 1
        Exit
Seg_Terminate                               ; beginning of segment
        1 STO Retain_Changed_Potentials     ; set variable to non-zero
                                            ; to save ending potential.
        Exit
```