1. Expression Node

Pentru calcule de expresii cu **o singură variabilă**, putem apela structura **Expression Node** din paleta Functions (subpaleta Numeric)

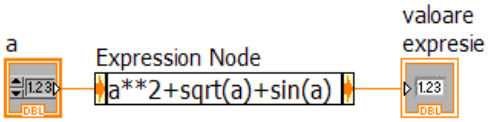


Fig. 1

2. **Formula Node** = structură care permite:

1. evaluarea de ***expresii mai complexe*** cu sintaxa C
2. execuţia de programe în Limbajul C

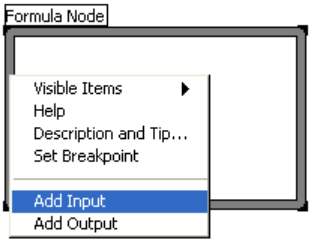


Fig. 1. Adăugare variabile I/ O

Variabile de intrare (Add Input)
- primesc valoare dinafara structurii Formula Node
- folosite în expresii scrise în format/sintaxa C

Variabile de ieşire (Add Output)
- primesc valoare în urma evaluării expresiilor:

Var_out=expresie;

Expresie= succesiune de **operatori** şi **operanzi**

Operatorul de atribuire: **=** (şi atribuiri multiple)

Operatori logici:

"şi" **&&** , "sau" || , "negativare" **!**

Valoarea numerică **0** are valoare logică FALSE iar valorile **nenule** TRUE

Operatori de inegalitate şi egalitate:

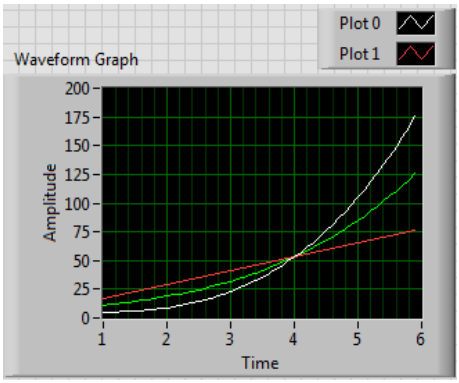
!= respectiv **= =**

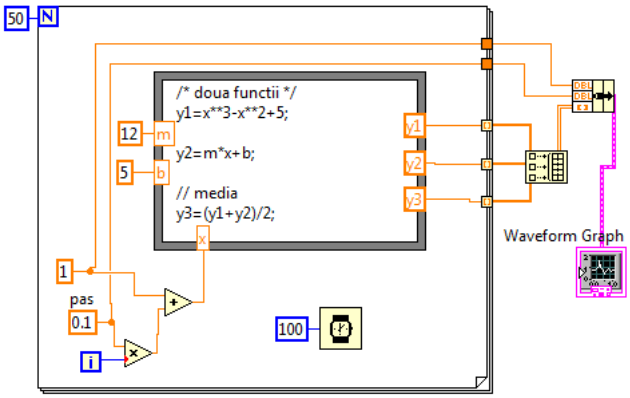
Operatori relaţionali: **< , >, <=, >=**

Operatori aritmetici: **+, - , *, / , ****

Operatori unari: **+, -**

Aplicaţie. Calcul valori 3 funcţii + grafice





In cadrul Formulei Node sunt calculate valorile a două funcţii y1(x) şi y2(x) şi media lor pentru fiecare pereche de valori. Argumentul este pregătit în afara Formulei Node:

x= valoare iniţială + i * pas

La fiecare iteraţie sunt calculate trei valori (y1, y2, y3); la ieşirea din ciclul For sunt disponibile 3 tablouri fiecare de câte 50 valori. Acestea se asamblează (Build Array) într-o matrice 3x50 şi prin Bundle se adaugă valoarea iniţială (1) şi pasul (0.1) se obţine o structură de date cu trei câmpuri în vederea reprezentării grafice cu Waveform Graph.

Operatorul ***ternar***:

variabila = **exp_cond ? *expT* : *expF***

1)dacă exp_cond este T => variabila=*expT*

2)dacă exp_cond este F => variabila=*expF*

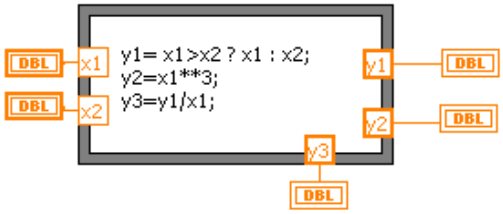


Fig. 3. Formula Node, Calcule 2

Op. pe bit:

| (bit or), **&** (bit and), ^ (bit exclusiv or),

<< (shift right), >> (shift left),

Ridicarea lui **x la puterea y**, folosim o expresie de forma **x**y**.
Comentarii în text: **/*** ...comentarii ... ***/** sau //

Funcţii C predefinite

- în cadrul expresiilor, poate fi apelată o gamă variată de **funcţii** a căror nume se scriu cu litere mici:

abs(), **int**(x) (rotunjire la intregul apropiat),
max(x,y), **min**(x,y), **mod**(x,y) (restul împărţirii x/y),
rand() (val. intre 0 şi 1), **sign**(x),

funcţii **trigo**nometrice (argument radiani): sin(x), cos(x), tan(x), cot(x),
secanta: sec(x), cosecanta: csc(x),
funcţii **trigo**nometrice inverse: asin(), acos(), atan(x/y), **atan2**(x,y)
funcţii hiperbolice: sinh(x), cosh(), tanh(),
funcţiile inverse hiperbolice: asinh(x), acosh(), atanh(),
funcţia exponenţială: exp(x),
logaritm natural: **ln**(x), logaritm zecimal: **log**(x), logaritm baza 2: **log2**(x)

Instrucţiuni C permise:

I. Condiţionale: **if, if-else**,
I. Ciclare: **for, do-while, while**
I. Selecţie multiplă: **switch** (case),
Instrucţiunea **bloc { }**, cu variabile declarate local,
I. **Break** pentru ieşire din instr. Ciclare (for, do-while, while)
I. **Continue** pentru salt la iteraţia următoare în ciclări.
Exemplu de calcule efectuate prin folosirea structurii *Formula node* (m=30, n=5).

3. Ciclul FOR, suma elementelor dintr-un şir real prin **cod C**

Se declară variabilele **i** şi **sum**.
In ciclul **for** se însumează elementele din şir astfel:

1) i=0, (expr1)
2) dacă **i < ne** (expr2) este T,
=> se execută corpul ciclului for,
iar dacă este F se iese din ciclul for...
3) i++, (expr3) creşte i cu 1
4) se execută pasul 2) din nou.

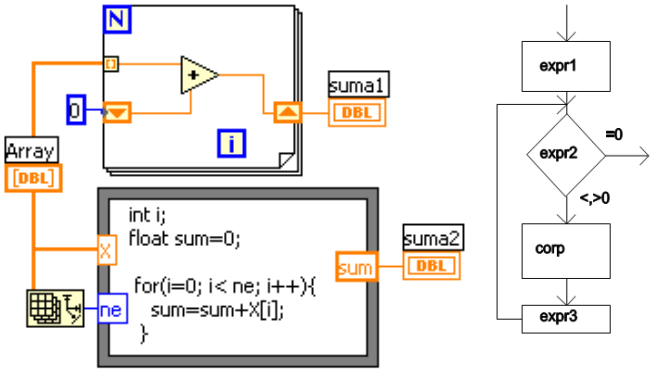
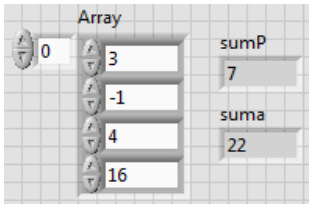
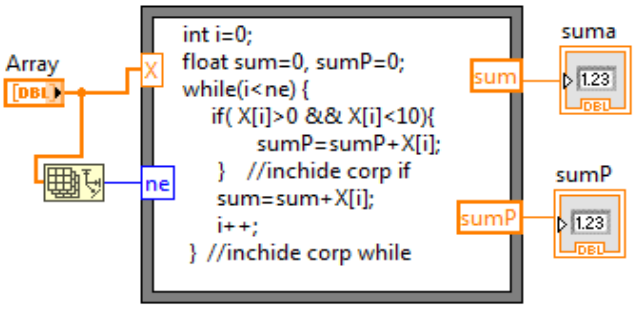


Fig. 5 Suma reale Labview şi cod C (For Loop)

4.Suma tablou val.reale, ciclul **While**



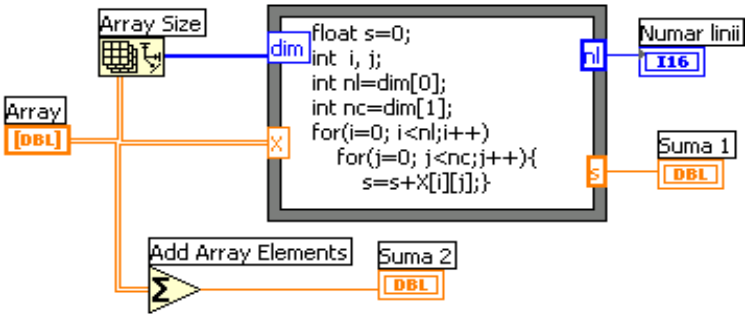


4. FOR imbricat: **suma elem. dintr-o matrice, cod C în "Formula Node"**

suma elementelor din matrice **nl** x **nc**
*parametrii de intrare **dim, X** sunt nume de tablouri 1D respectiv 2D,
*parametrii de ieşire **nl, s** sunt valori scalare

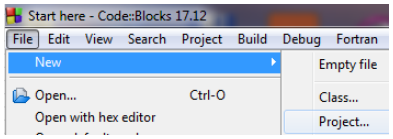
dim[0] -> numar de linii matrice
dim[1] -> numar coloane
X[i][j] -> element linia i, coloana j din X

```
   float s=0;  //declaratii şi initializare variabile
   int i, j;
for (i=0; i<nl; i++)  // ciclare linii
   for(j=0; j<nc; j++) { //ciclare coloane
     s=s+X[i][j]; }    // corp ciclu interior
```

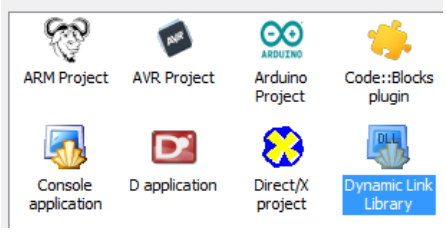


5. Apel C/C++ Dynamic Link Library (DLL) din Labview

5.1. Creează un proiect DLL din **Code::Blocks IDE** în C/C++



File/ New/ Project...; Din fereastră selectăm pictograma Dynamic Link Library;



Scriem titlul proiectului: ***SumaTablouDLL***
Selectăm Compiler: ***GNU GCC Compiler***
Bifăm numai Create "Release" configuration:
Se creează **fişierul sursă C++:** *main.cpp*
şi un **fişier header:** *main.h.*

Fişierul **main.cpp** conţine:

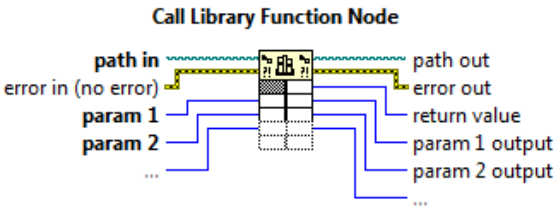
```
#include "main.h"

// a sample exported function
float DLL_EXPORT SumaTablou(const LPCSTR sometext,  float X[ ] ,  int ne )
{
   MessageBoxA(0, sometext, "DLL Message", MB_OK | MB_ICONINFORMATION);
   int i;
   float sum=0;
   for (i=0; i<ne; i++) {
        sum=sum+X[i];
    }
    return sum;
}
.....
```

Fişierul **main.h** conţine declaraţia funcţiei SumaTablou( ):

```
......
extern "C"
{  #endif
float DLL_EXPORT SumaTablou(const LPCSTR sometext, float * X, int ne );
#ifdef __cplusplus}
......
```

5.2. **Generare DLL** cu Build Ctrl+F9 =>Output file is ...bin\Release\SumaTablouDLL.dll
* se corecteza erori dacă sunt.

5.3. **Call Library Function Nod** din Labview



Funcţia Call Library Function Nod din *Connectivity/ Libraries& Executables\...* apelează cod extern din biblioteca DLL (Dynamic Link Library). Se poate expanda (asemănător cu Bundle) şi configura nodul 'Call Library Function Node' pentru a preciza biblioteca, funcţia, parametrii funcţiei, valoarea returnată. Nodul conţine câte un terminal pereche intrare-ieşire pentru fiecare parametru al funcţiei **SumaTablou**. Prin terminalul din stânga se trimite o valoare în funcţie iar prin cel din dreapta se citeşte val. param. după execuţia funcţiei.

Return value este valoarea returnată a funcţiei.

Dacă tipul datei returnate de funcţie este **Void**, terminalul cel mai de sus-dreapta nu este folosit. Implicit 'calling convention' este C.

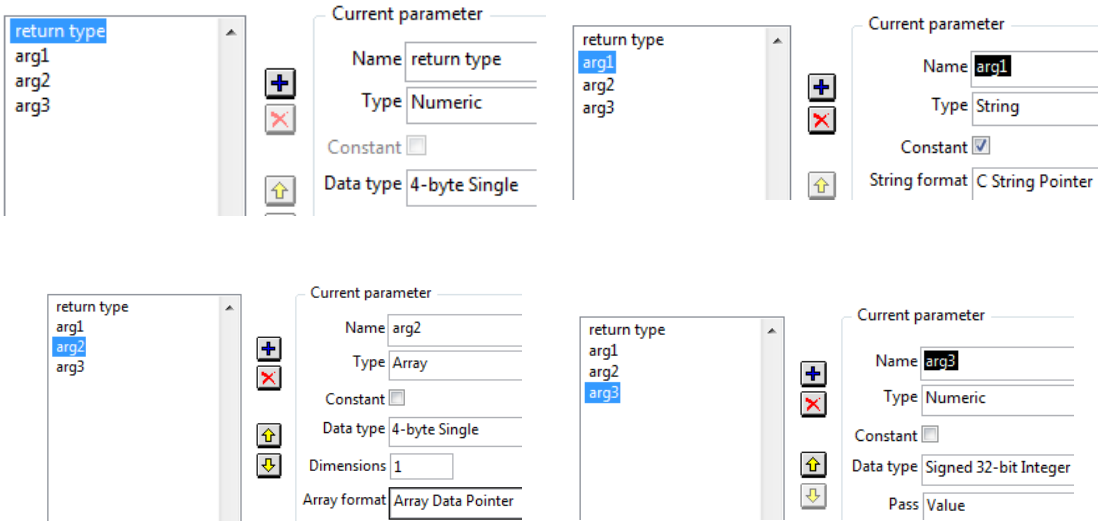
5.4. In funcţia **SumaTablou** din SumaTablouDLL.dll parametrii sunt şir de caractere/string, tablou/array numeric real (float) şi numeric întreg pe 32 biţi. Funcţia returnează suma valorilor din

tablou (float 4 bytes). In general pentru tipul numeric se pot folosi sub-tipurile: întreg pe 8, 16, 32, 64-bit, cu semn sau fără semn şi tipurile real pe 4-byte (single-precision numbers) şi 8-byte (double-precision numbers). In figuri este prezentat modul de setare a parametrilor aplicaţiei.

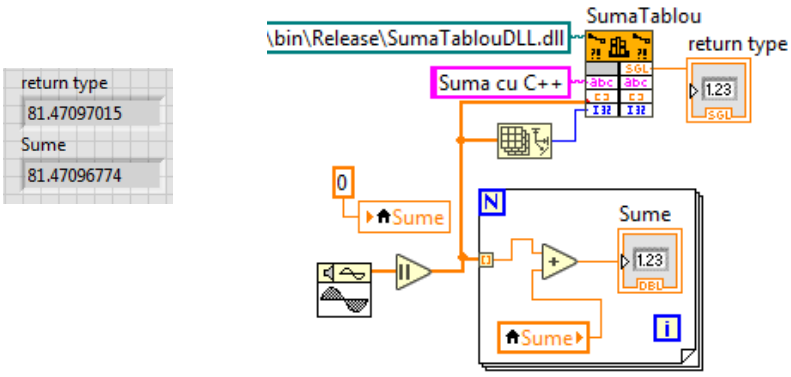
Funcţia **SumaTablou** primeşte 3 parametri actuali (cunoscuţi):

arg1: şirul 'Suma cu C++',
arg2: valorile absolute din tabloul de reale generat de Sine Pattern şi
arg3: numar elemente din tablou.

In continuare este prezentat modul de setare a parametrilor aplicaţiei.



In diagrama aplicaţiei observăm în paralel apelul Call Library Function Nod şi calculul sumei elementelor generate de Sine Patern.vi cu folosirea variabilelor locale.



DLL este o bibliotecă **dinamică** de programe compilate în format executabil (.exe). Programele pot fi folosite prin apelul lor în timpul execuţiei altor aplicaţii.

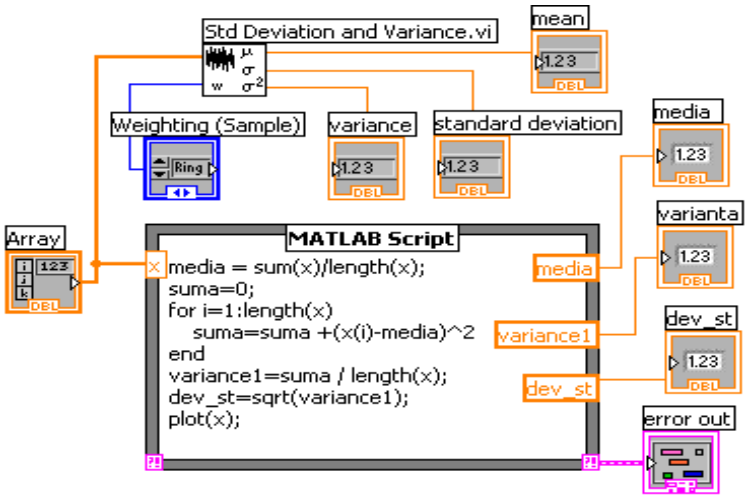
LIB este bibliotecă **statică**, conţine programe compilate care se leagă la început în cadrul unui program care le apelează şi vor fi incluse în executabilul acelui program. Dacă două programe (A, B) apelează acelaşi program executabil (E) din LIB, ambele A şi B vor fi mai mari prin includere la început prin linkeditare.

6. Apel script MATLAB (MATLAB script node) din Labview (v7.1)

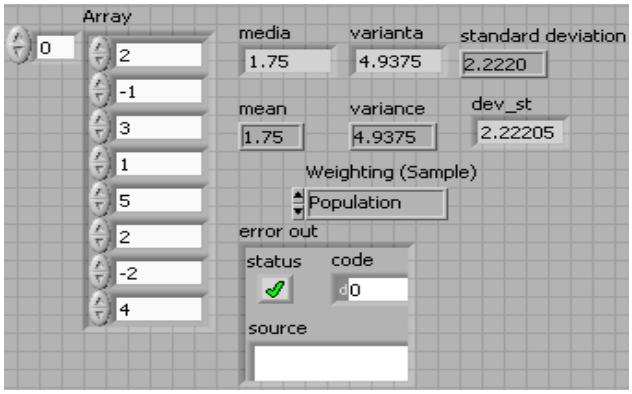
Se poate executa cod Matlab prin apelul produsului software MATLAB® pentru execuţia codului script Matlab.

Produsul Matlab trebuie să fie instalat în calculator (versiunea 6.5 sau o versiune mai recentă).

Exemplu calcul: media, deviaţia standard şi varianţa pentru un şir de valori numerice X(i), i=1…,n.



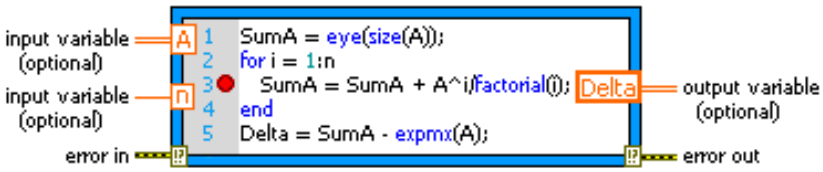
Variabile de intrare şi ieşire:
add input: tabloul **x**,
add output: **media, variance1, dev_st**



7. MathScript node in LabVIEW

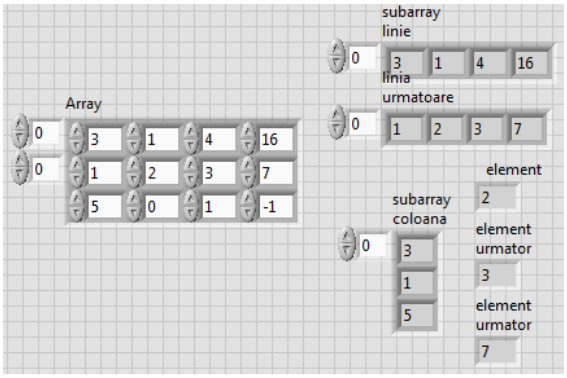
Este un modul Labview special dezvoltat pentru calcule numerice.

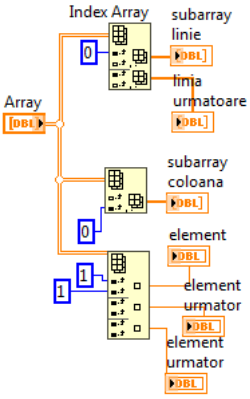
Trebuie instalat modulul: LabVIEW **MathScript RT**



8. Index Array - extragere linie, coloană sau element din tablou 2D

8.1. Indicele de sus selectează linia, cel de jos coloana





8.2. Variante pentru calcul sume elemente de pe DP şi DS, în matrice pătratică

* accesarea directă a elementelor diagonalei (fără parcurgerea întregii matrice).

Figura 16: suma elementelor de pe diagonala principală prin două variante. În prima variantă observăm indexarea liniilor matricei la intrare în ciclu, pentru controlul numărului de iteraţii.

** i, j - indice linie, coloana; **DP: i=j**; **DS: i+j=n-1** => ***j=n-1-i***

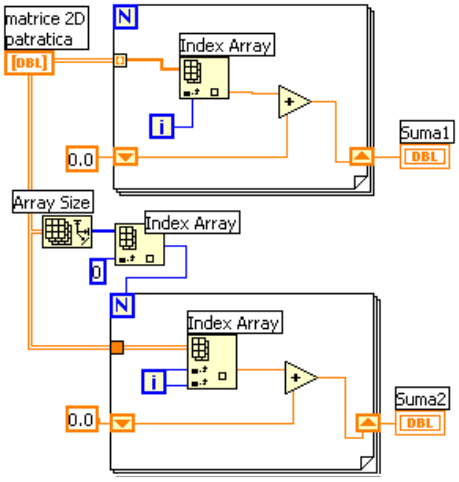


Fig. 16 Suma elem. de pe DP

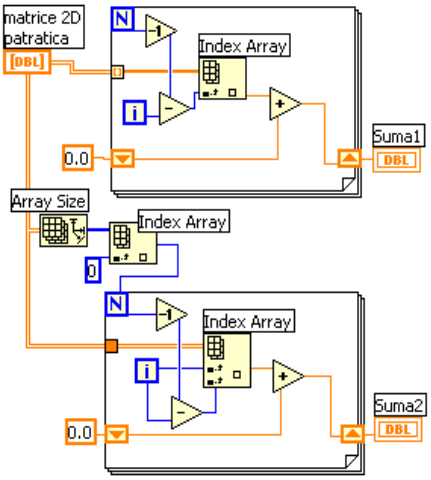


Fig. 17 Suma elem. de pe DS

Index array = extrage elementul indice i din tabloul 1D de intrare
extrage elementul indice i, j din tabloul 2D de intrare

9. Build Array

9.1. Separare valori din şir în 2 subşiruri:

***subşir cu valori pozitive şi zero**

***subşir cu valori negative.**

Initialize Array: este iniţializat, cu valori *0.0*, un tablou (1D) având zero elemente.

Build Array: adaugă la urmă (în varianta ***Concatenate Inputs***) elementul curent la tabloul de elemente pozitive sau negative.

- există (la alegere) şi varianta de alipire a două linii generând o matrice cu două linii

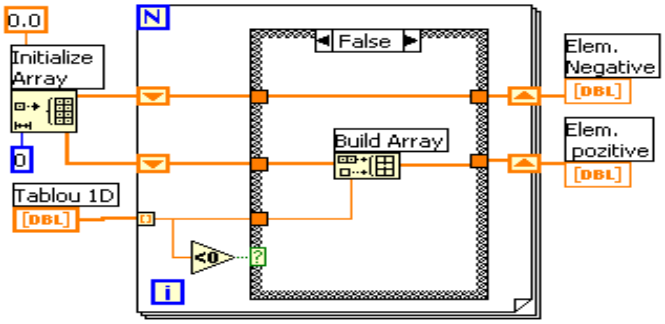
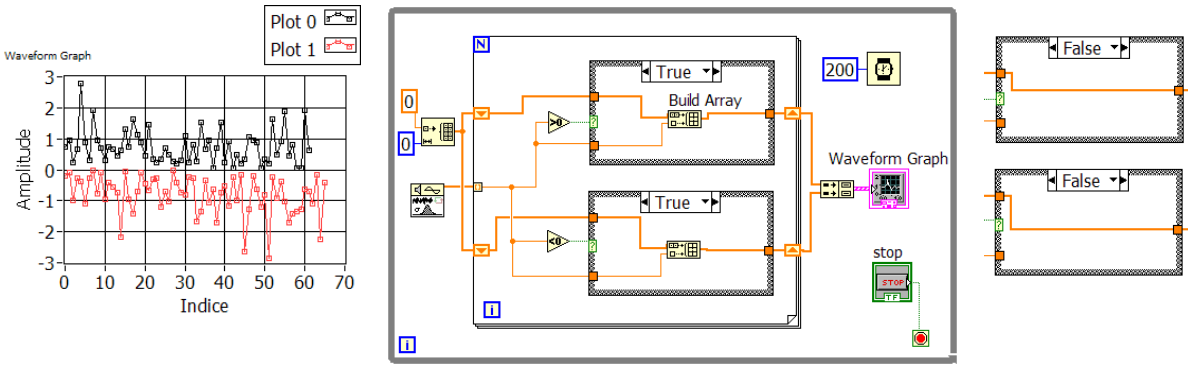


Fig. 18

9.2. Separare 2 subşiruri: unul strict pozitiv (>0); unul strict <0 + GRAFIC



Pentru <u>valoarea 0</u> sunt selectate cazurile False din ambele instr. Case iar tablourile trec neafectate.

Pentru **valoare >0** inst.**Case de sus execută cazul True** şi celălalt Case execută caz False.

Pentru **valoare <0** inst.**Case de jos execută cazul True** şi celălalt Case execută caz False

Sirurile sunt de lungimi diferite → Build Cluster Array → Waveform Graph

9.3. Aplicaţie. Comparare element curent cu elementele alăturate

Se parcurge un şir de numere reale.

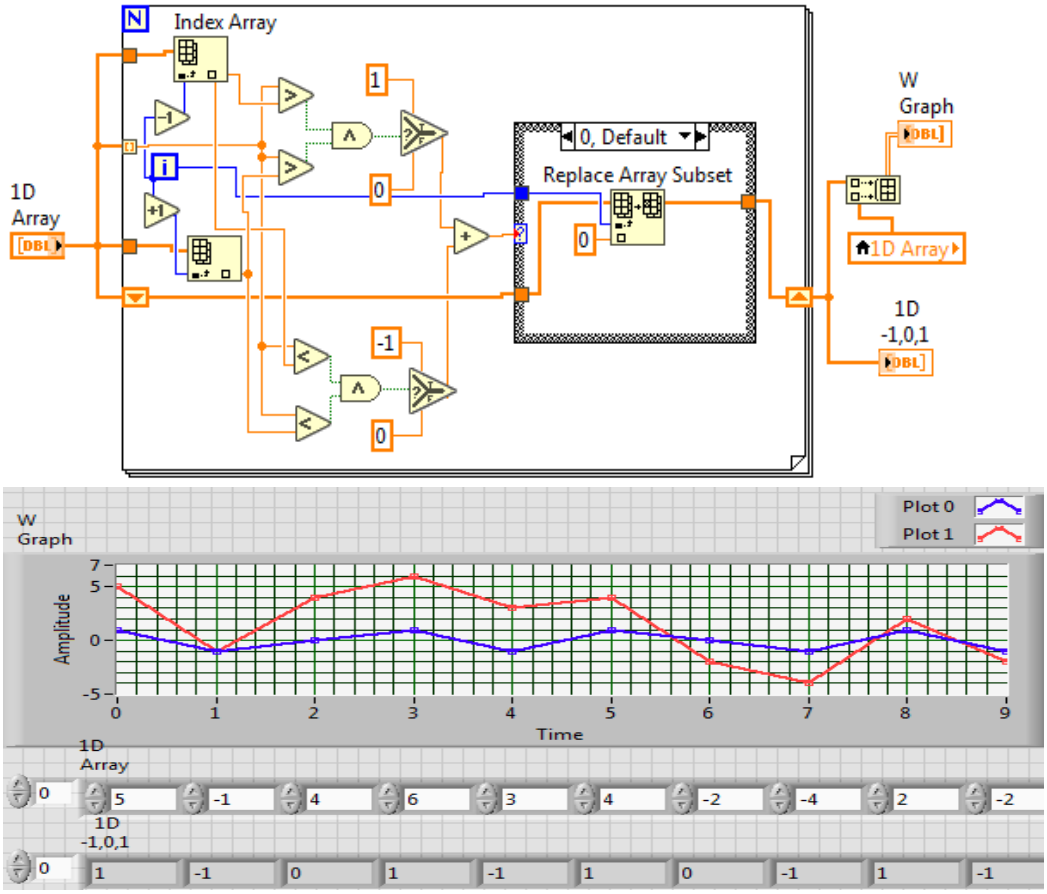
Dacă valoarea curentă este mai mare decât precedenta (indice i-1) şi decât următoarea (indice i+1) se pune 1 într-un nou tabel 1D.

Dacă valoarea curentă este mai mică decât precedenta şi următoarea se pune -1 în noul şir, altfel se pune valoarea zero.

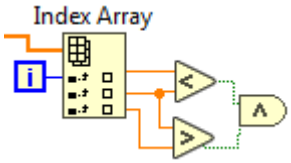
Valoarea sumei celor două valori care ies din ambii operatori Select:

1+ 0 sau 0 + -1 sau 0 + 0

selectează un caz din cele trei cazuri şi **pune 1, -1 sau 0 la poziţia i** în copia tabloului din reg. shift. Prima şi ultima valoare din tabloul 1D -1,0,1 se pot elimina.



9.4. Varianta de referire elemente:



10. Replace Array Subset

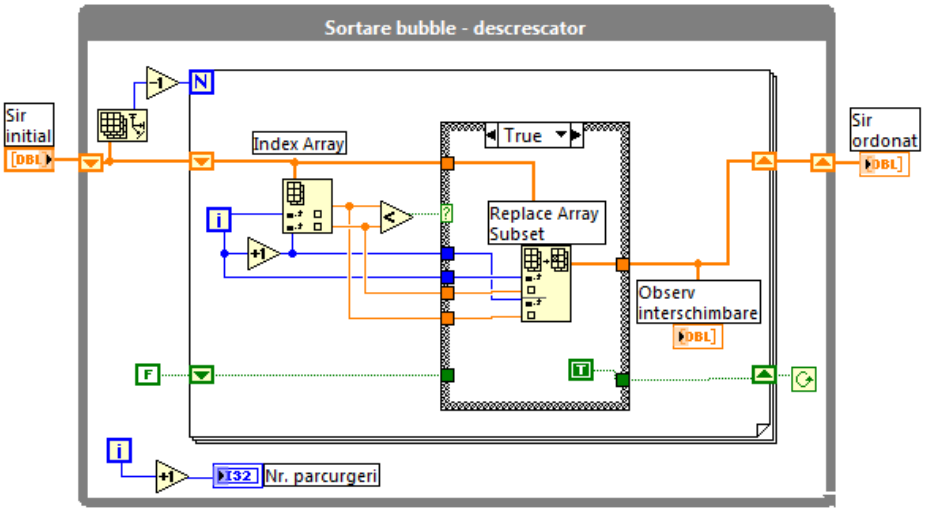
Ordonare prin metoda 'bulelor' (bubble sort)

Se va ordona **descrescător** şirul de valori furnizat în panoul frontal.

Se realizează următoarele comparaţii a câte două valori alăturate din şir:

a[j-1]<a[j]

Dacă relaţia logică este adevărat se interschimbă elementele altfel elementele rămân pe loc. După fiecare comparaţie elementul mai mic va fi pe poziţia cu indice mai mare adică a[j]. Se începe cu j=1, caz în care are loc prima comparaţie a[0]<a[1] şi se termină cu j=i. Ciclul exterior stabileşte valoarea lui i adică până la ce valoare indicele j poate creşte.



Replace Array Subset realizează interschimbarea prin două înlocuiri: valoarea din tablou de la indicele i este pusă la indicele i+1 iar valoarea din tablou de la indicele i+1 este pusă la poziţia i.

La început i=n-1 iar ciclul for interior în urma repetatelor comparări şi eventual interschimbări va deplasa cel mai mic element din tablou pe ultima poziţie a[n-1].

La următoarea execuţie a corpului ciclului for exterior, i devine n-2 iar în ciclul interior j va lua valori de la 1 la n-2. Efectul comparărilor şi eventual a interschimbărilor va aduce cel mai mic element al subşirului primelor n-1 elemente pe poziţia a[n-2]. Acest element este evident mai mic decât a[n-1] selectat în prima etapă de comparări. Se observă procesul de aducere a celui mai mic element din subşiruri to mai scurte (ce încep cu poziţia j=1 şi se termină pe poziţia j=i), pe ultima poziţie din subşir. Acest proces în final realizează aranjarea descrescătoare a elemenelor din tablou.

Pentru a se întrerupe procesul imediat când sirul devine ordonat se poate introduce un indicator logic care se poziţionează pe True la orice interschimbare. Când la parcurgerea unui subşir variabila logică râmâne False (False este setat automat în expresia de iniţializare a ciclului for interior) se va întrerupe ciclul for exterior.

11. Rezolvarea unui sistem de ecuaţii

11.1. Sistem de ecuaţii liniare

Un sistem de ecuaţii liniare scris matriceal:

$$\boxed{Ax=B}$$

este rezolvat prin apelul funcţiei ***Solve Linear Equations.vi*** conform figurii 20. Pentru a eficientiza calculul soluţiei se specifică tipul matricei coeficienţilor sistemului prin controlul ***matrix type*** prin selecţia unei valori din lista:

general (0),
matrice pozitiv definită (1),
matrice triunghiulară jos (2) sau
matrice triunghiulară sus (3).

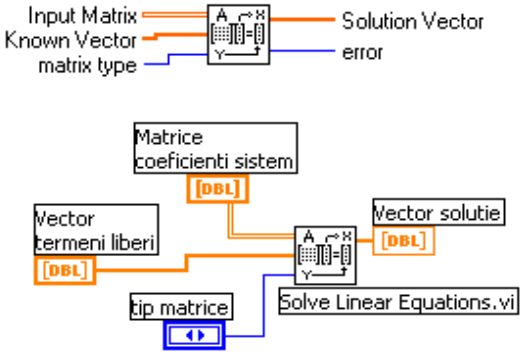
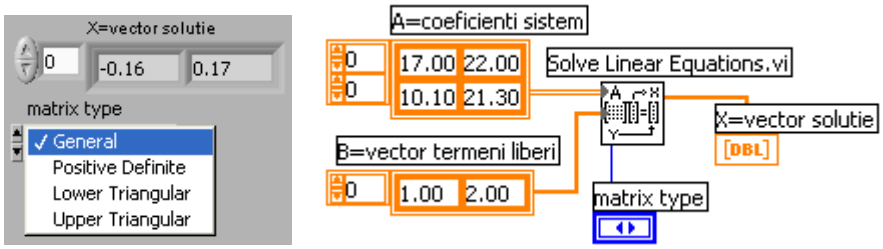


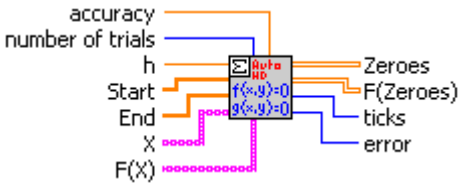
Fig. 20

Controalele pot fi în PF (Fig. 20) sau **constante tablou** (1D sau 2D) în diagrama:



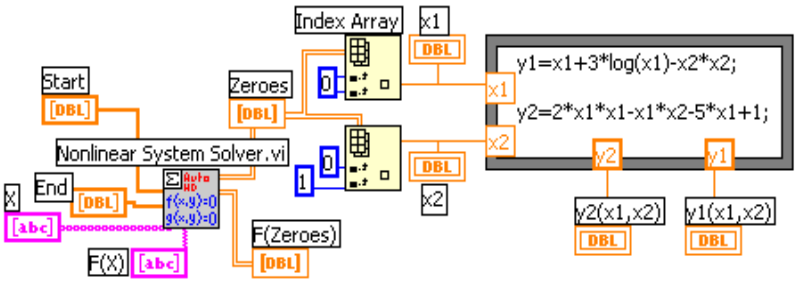
11.2. Soluţiile unui sistem de ecuaţii neliniare

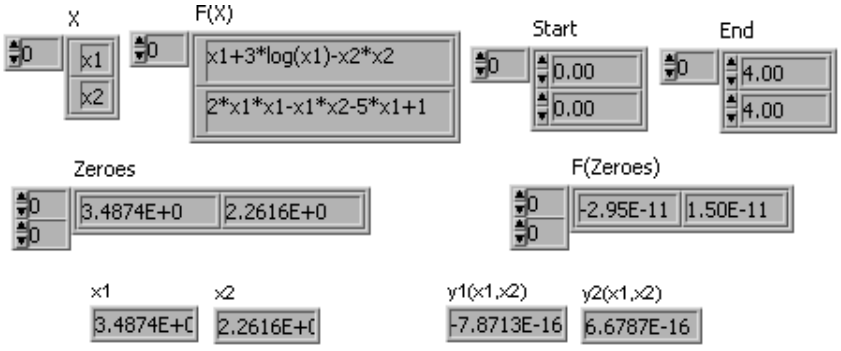
Folosim funcţia **Nonlinear System Solver.vi**, preluată din paleta Mathematics/.../Zeroes.



$$\begin{cases} x1 + 3\log(x1) - x2^2 = 0 \\ 2x1^2 - x1x2 - 5x1 + 1 = 0 \end{cases}$$

Soluţia (x1=3.487, x2=2.261) este verificată în Formula Node.





Observăm:
Intrările **F(X)** şi **X** sunt tablouri de şiruri de caractere.
In fiecare linie este introdusă expresia unei ecuaţii din sistem respectiv câte o variabilă (x1,x2)
Start, **End** specifică intervalul de căutare a soluţiei pe fiecare variabilă.
Ieşirile Zeroes, F(Zeroes) (tablouri reale 2D) returnează soluţiile sistemului şi valoarea expresiilor la înlocuirea soluţiilor.

II. Numere complexe în Labview

Exprimare z în **coordonate carteziene**	
Forma #1: $\boxed{z = x + j\,y}$	x, y sunt numere reale, $j = \sqrt{-1}$
x = partea reală a numărului complex z $j \cdot y$ = partea imaginară a numărului complex z	

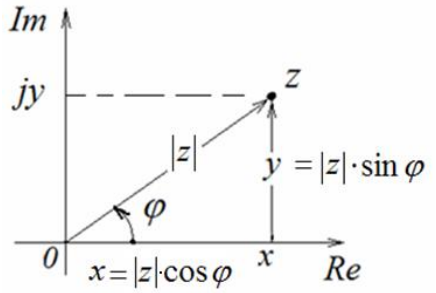


Fig. 1. Numărul complex z
în planul complex

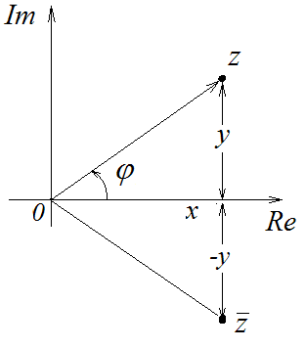
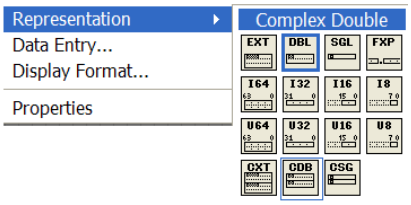


Fig.2 două numere
complex conjugate

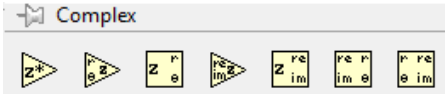
Exprimare z în **coordonate polare**	Exprimare z în coordonate carteziene Forma #2
$\boxed{z = \lvert z\rvert \cdot e^{j\varphi}}$ sau: $\lvert z\rvert\angle\varphi$	$z = \lvert z\rvert \cdot (\cos\varphi + j\sin\varphi)$ **Conversie** din polar în cartezian: $x = \lvert z\rvert \cdot \cos\varphi \quad y = \lvert z\rvert \cdot \sin\varphi$

1. Reprezentarea numerelor complexe, produsul şi câtul

1. Z1 şi Z2
   sunt controale numerice,
   tipul de dată complex
   (**CDB**= parte reală + parte imaginară):
   Control + ***Representation***

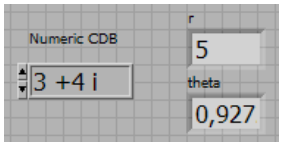


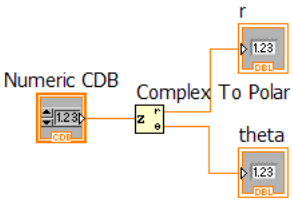
2. Operatorul
   Complex To Polar
   returnează două numere reale: ***modulul*** şi ***faza***;



r = |z| = sqrt(a² + b²)

theta = arg(z) = arctan2(b,a)





3. Operatorul

Complex To Re/Im

returnează două numere reale:

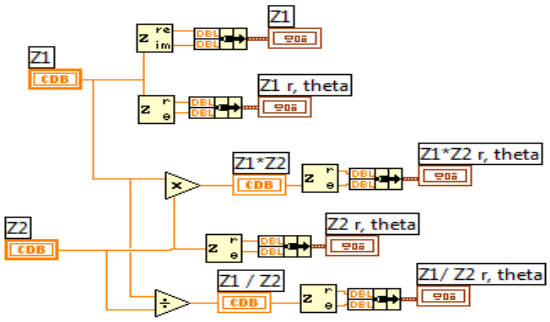
partea reală şi *partea* ***imaginară***;

3. **<u>Inmulţirea</u>** a două numere complexe =

=***produsul modulelor*** şi ***suma fazelor***.

4. **<u>Impărţirea</u>** a două numere complexe =

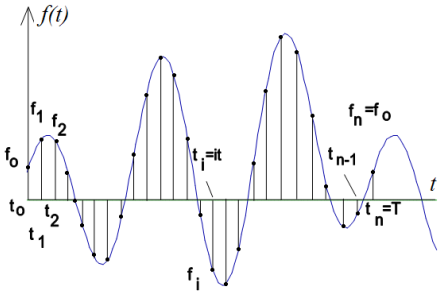
=***câtul modulelor*** şi ***diferenţa fazelor***.



2. Generare şir numere complexe cu Funcţia *Real FFT.vi*

Discretizarea unui semnal analogic (considerat periodic de perioadă T)**:** daca semnalul analogic x(t) este măsurat experimental iar semnalul este discretizat (t_0, **Δt**) rezultă şirul 1D de valori măsurate X:

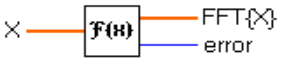
Momente echidistante: t_0, **Δt**	t_0, t_1, ... , t_{n-1}.
Sir de valori reale asociat **X**:	x_0, x_1, ... , x_{n-1}



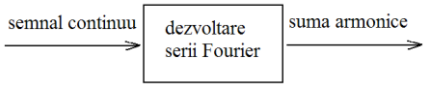
Tipul de dată **waveform**

=structură cu 3 câmpuri: t0, Δt, X

Instrumentul *Real FFT.vi* (Analyze/ Signal Processing/ Frequency Domain)



Real FFT.vi



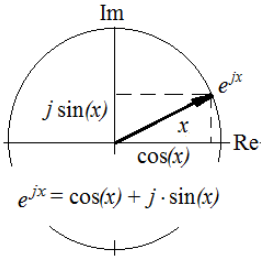
Intrarea în funcţie:

Tabloul de **<u>n</u>** valori **<u>reale X</u>**

Calcule efectuate: se aplică transformata Fourier rapidă (FFT) sau transformata Fourier discretă reală (DFT).

Coeficienţii spectrali <u>complecşi</u> y_k se calculează cu relaţia (1):

$$y_k = \sum_{i=0}^{n-1} \left(x_i \, e^{-jk2\pi\frac{i}{n}}\right) \qquad (1)$$



unde k, i sunt indici iar $j = \sqrt{-1}$

k=0...n-1 , i=0..n-1

Folosind relaţia Euler: $e^{jx} = \cos(x) + j \cdot \sin(x)$, *rezultă:*

$$\boxed{y_k = \sum_{i=0}^{n-1} x_i [\cos(k 2\pi \frac{i}{n}) - j \sin(k 2\pi \frac{i}{n})]} \qquad (1')$$

Ieşirea din funcţie:

Tabloul **Y=FFT{X}** de **<u>n</u>** valori **<u>complexe</u>**:

y_0 → componenta continuă (reală) $y_0 = \sum_{i=0}^{n-1} x_i \, e^0$ a semnalului de intrare X

y_1 → prima armonică (parte reală şi imaginară),

y_2 → a doua armonică,

…,

$y_{\frac{n}{2}-1}$ → a n/2-1 armonică,

$y_{n/2}$ → ***armonica Nyquist*** (reală)

$$y_{n/2} = \sum_{i=0}^{n-1} x_i [\cos(\pi i) - j \sin(\pi i)] = \sum_{i=0}^{n-1} x_i (-1)^i$$

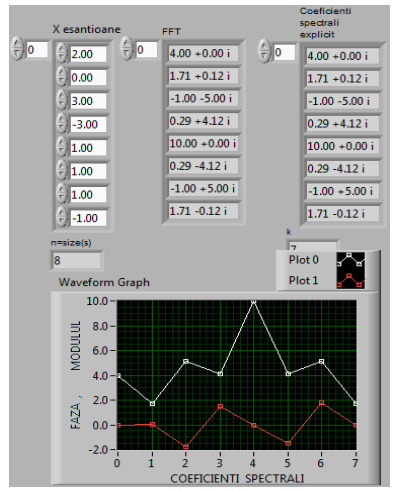
Urmează simetric faţă de armonica Nyquist, componentele complex conjugate de frecvenţe negative:

$y_{\frac{n}{2}+1}$ → a n/2-1 armonică ($= y^*_{\frac{n}{2}-1}$)

…,

y_{n-2} → a doua armonică ($=y_2^*$),

y_{n-1}, → prima armonică ($=y_1^*$).

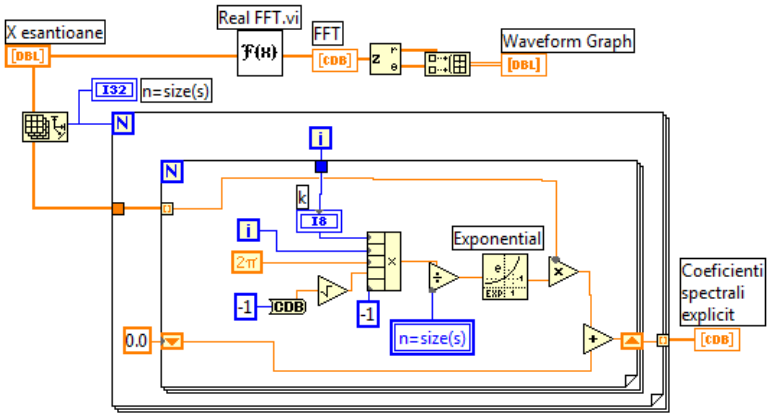


y_0 valoare *reală*

y_1, y_2, y_3 valori complexe

y_4 valoare *reală*

y_5, y_6 y_7 valori complex conj.



$$y_k = \sum_{i=0}^{n-1} (x_i \, e^{-jk2\pi \frac{i}{n}})$$

Calcul coeficienţi spectrali, relaţia (1)

Diagrama: *suma după i este efectuată de ciclul interior;*

la fiecare ciclu exterior se calculează câte un y_k *(k=contor ciclul exterior).*

Coeficienţii complecşi y_k (Re + Im) pot fi exprimaţi în varianta:

<u>Modul/magnitude</u>= $\sqrt{\mathrm{Re}^2 + \mathrm{Im}^2}$, <u>fază/phase</u>= $a\tan2(\frac{\mathrm{Im}}{\mathrm{Re}}) \in$ [-180, 180]

n = numărul de eşantioane/valori pe **perioada T** (=n*Δt) sau într-un bloc de date supus analizei,

Δt = timpul între două valori măsurate sau perioada de eşantionare,

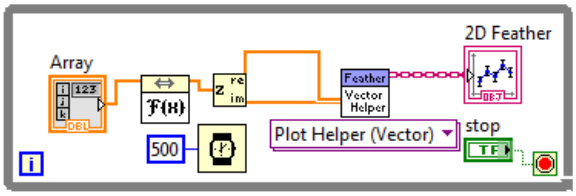
$\Delta f = \frac{f_s}{n}$ este spaţierea în frecvenţă între coef. spectrali y_k

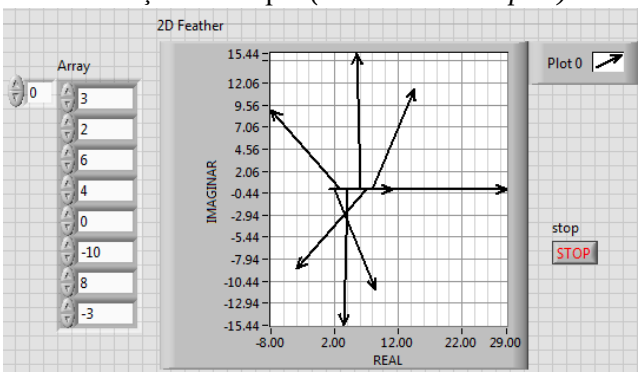
3. **Reprezentare grafică a coeficienţilor spectrali complecşi generaţi de funcţia FFT.vi**

1. **Feather Plot**

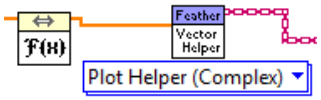
Se vizualizează perechi de numere complexe conjugate.

Pentru reprezentarea grafică se selectează Graph/ **Feather Plot** şi Plot Helper *(Vector sau Complex)*

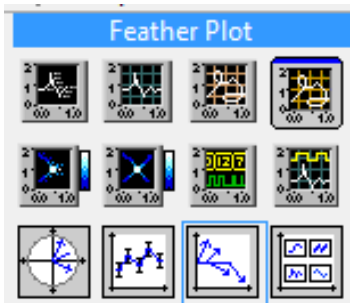




Varianta în care intrarea este tablou Complex:

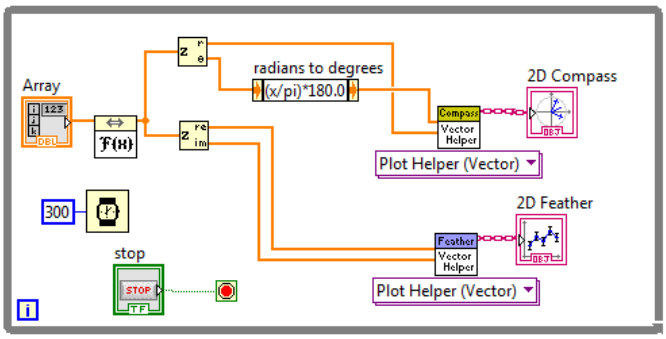


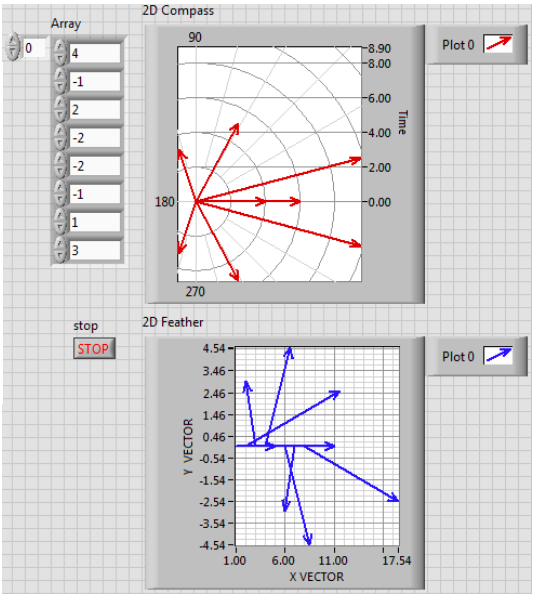
Modificaţi interactiv valorile din şirul real de intrare şi numărul de valori din vector. Observaţi perechile de vectori simetrici şi cei doi coeficienţi reali.



2. **Compass Plot**

Indicatorul grafic **Compass Plot** versiunea Plot Helper (***Vector*** *sau* ***Complex***) (etichetă 2D Compass) permite vizualizarea tabloului generat de funcţia Real FFT.vi reprezentând coeficienţii spectrali complex conjugaţi (plus cei doi coeficienţi reali), ca vectori cu aceeaşi origine.





Intrări:

a) varianta Vector:

tablou **module** şi tablou **unghiuri** grade

b)varianta Complex:

tablou de **numere complexe**