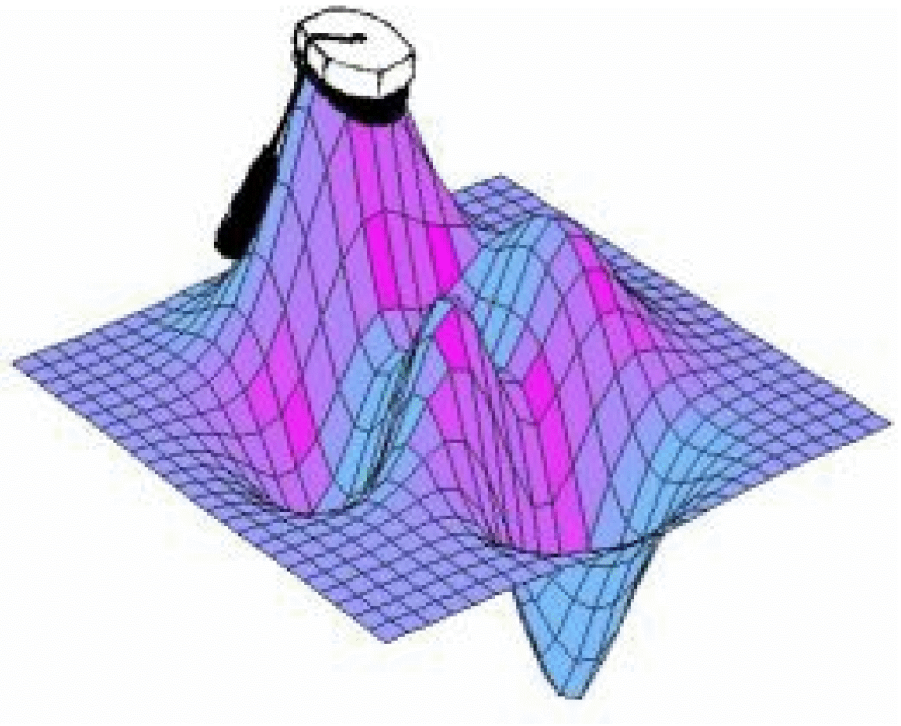
# TUTORIAL METODE OPTIMASI dalam LabVIEW



Exhausted Search ● Random Search Methods ● Steepest Descent ● Pengenalan LabVIEW ● Variabel dan Operator pada LabVIEW ● Struktur Keputusan ● File Input dan Output ● Tugas Mata Kuliah Metode Optimasi

**Jurusan Teknik Elektro**
**Universitas Komputer Indonesia**
**Copyright @2005**

# 1

# Pengenalan Lab-View

**LabVIEW** adalah sebuah software pemrograman buatan **National Instruments** dengan menggunakan bahasa pemrograman berbasis grafik atau blok diagram. **LabVIEW** memiliki fungsi dan peranan yang sama dengan bahasa pemrograman lainnya seperti *C++*, *Matlab* atau *Visual Basic*.

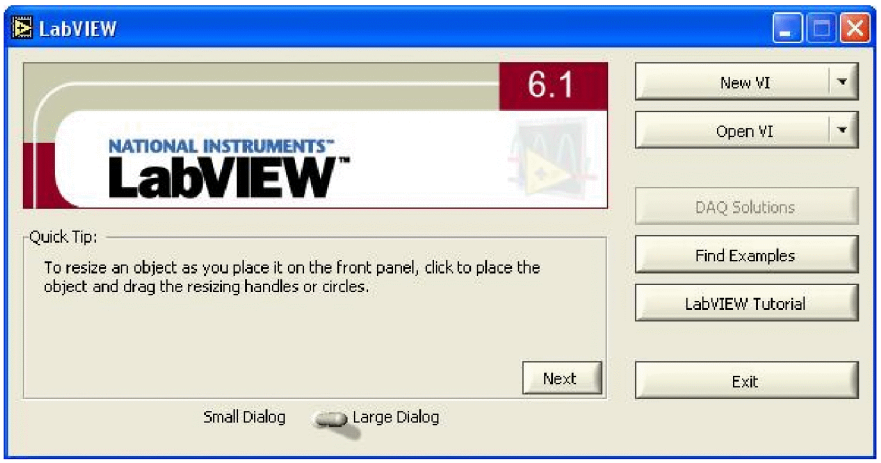
Program yang dibuat dengan LabVIEW disebut juga sebagai *Virtual Instruments* atau VIs karena cara kerjanya penampilan dan operasinya yang mirip dengan suatu instrument fisik.

Keuntungan bahasa pemrograman menggunakan LabVIEW :
- Mudah mendebug atau mendeteksi kesalahan
- Mudah mengikuti jalannya aliran VI
- Program dibuat secara hirarki dan modular, artinya setiap VI dapat dipakai sebagai subVI dari VI lainnya.
  Bila kita hendak membuat suatu aplikasi yang cukup rumit, maka kita dapat membaginya menjadi beberapa modul dan kemudian anda dapat membaginya menjadi beberapa submodul yang lebih sederhana.
- Mudah membuat simulasi yang menampilkan *Graphical User Interface*

## Memulai *LabVIEW*

LabVIEW menyediakan *template* yang berisi informasi yang dapat membantu memulai LabVIEW. Berikut ini langkah lengkap untuk memulai LabVIEW.

1. Jalankan program LabVIEW
2. Pada LabVIEW *dialog box*, yang ditunjukkan pada gambar 1 - 2, click tombol *New VI* untuk menampilkan *New VI*.

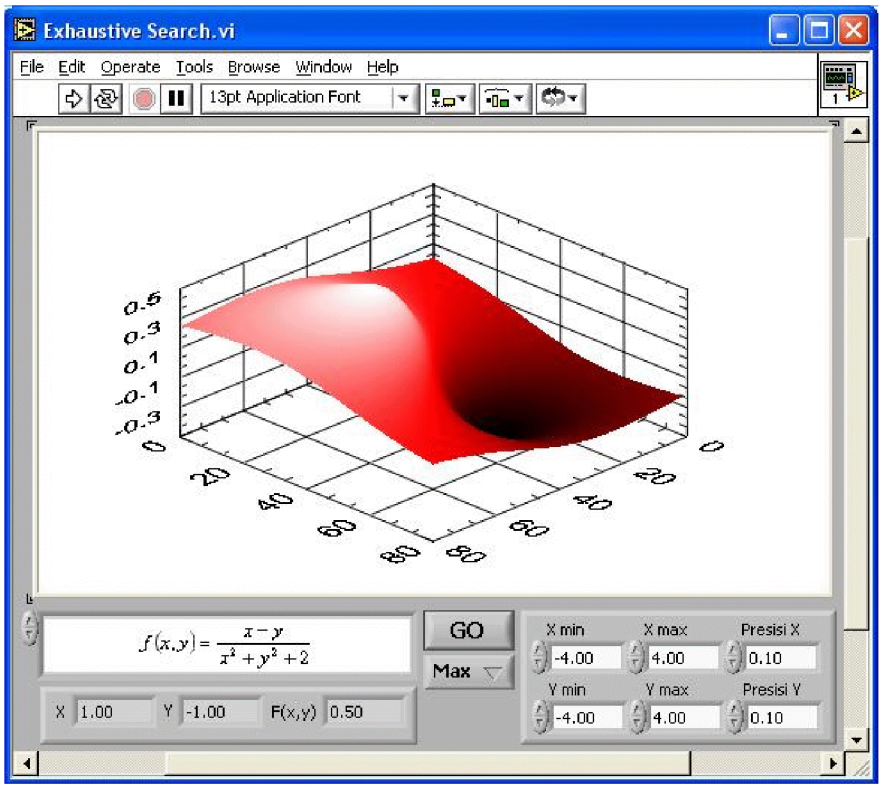



**Gambar 1 – 1.** LabVIEW *Dialox Box*

Untuk dapat bekerja dengan LabVIEW, anda perlu mengenal komponen-komponen berikut.

## Front Panel

**Front Panel** berfungsi sebagai interface untuk *user* yang akan mensimulasikan panel untuk instrumen. Gambar 1-1 menunjukkan contoh dari suatu **Front Panel**.



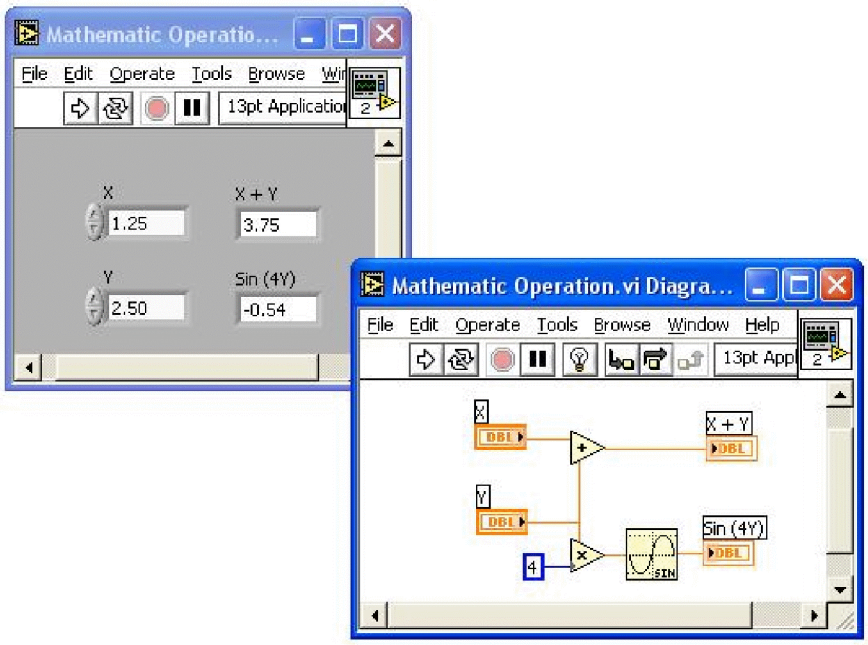

**Gambar 1 – 2.** Contoh dari **Front Panel**

**Front Panel** terdiri dari **controls** dan **indicators**, yang berfungsi sebagai terminal input dan output yang interaktif pada VI. Contoh **controls** adalah knobs, push buttons, dials dan device input lainnya. Contoh **indicators** adalah graphs, LEDs dan display lainnya. **Controls** mensimulasikan alat instrument input dan supply data pada diagram blok VI. **Indicators** mensimulasikan alat instrument output dan penampil data yang dihasilkan blok diagram.

## Block Diagram

Setelah **Front Panel** dibuat, Anda perlu membuat instruksi-instruksi yang mengatur kerja instrument-instrument pada **Front Panel**. **Block diagram** mengandung kode-kode yang berfungsi sebagai instruksi untuk **Front Panel**. Objek pada **Front Panel** akan menjadi terminal pada **Block Diagram**.

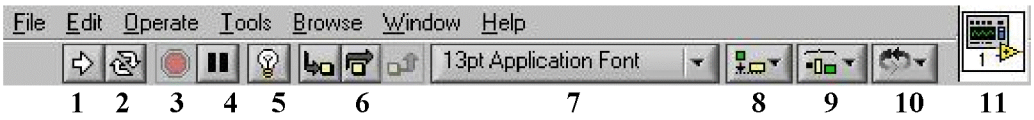
VI pada gambar 1-2 menampilkan beberapa objek **Block Diagram** utama, seperti terminal, functions dan wires.




**Gambar 1 – 3.** Contoh dari **Block Diagram** beserta **Front Panel**

## Menu Bar

Dalam membuat suatu VI, perlu dipahami **Menu Bar** dari **Block Diagram** dan **Front Panel** dimana terdapat beberapa bagian yang akan sering dipakai dalam membuat suatu VI. Beberapa *button* pada **Block Diagram** tidak terdapat pada **Front Panel**.




**Gambar 1 – 4.** Contoh dari **Menu Bar** pada **Front Panel**

Fungsi dari masing-masing bagian adalah :

1. **run** : mengeksekusi VI sampai proses selesai
2. **run continuously** : mengeksekusi VI secara kontinu, setelah program selesai, maka program akan dieksekusi dari awal kembali. Proses ini berulang kontinu sampai tombol **abort** ditekan
3. **abort** : menghentikan proses eksekusi
4. **pause** : menghentikan eksekusi sementara
5. **highlight** : melihat jalan aliran program pada diagram secara perlahan.
6. **start single stepping** : mengeksekusi VI per step
7. **text setting** : mengatur setting text
8. **align objek** : mengatur tampilan objek
9. **distribute objek** : mengatur tampilan beberapa objek
10. **reorder** : mengatur tampilan beberapa objek yang saling bertumpukan
11. **icon** : gambar yang ditampilkan VI tersebut jika dijadikan subVI

## Tools Pallete

Dalam membuat suatu VI ada beberapa *tools* yang harus dipakai dan masing-masing mempunyai kegunaan tersendiri.
Tools tersebut dapat diakses melalui menu bar : **Windows » Show Tools Palette**

**Gambar 1 – 5.** Contoh dari **Tools Pallete**

Kegunaan dari masing-masing *tools* tersebut adalah :

1. **Operate Value** :
   mengubah nilai parameter dari suatu objek

2. **Connect Wire** :
   Menghubungkan beberapa objek dengan kabel

3. **Set/Clear Breakpoint** :
   Membuat atau menghilangkan sebuah *breakpoint*

4. **Probe Data** :
   Membuat sebuah *probe* yang berfungsi memonitoring data

5. **Object Pop Up** :
   Memunculkan menu yang berhubungan dengan objek tersebut atau memunculkan daftar objek

6. **Position / Size / Select**
   Mengedit atau membuat tulisan

7. **Edit Text**
   Mengedit atau membuat tulisan

8. **Scrool Window**
   Memindahkan sudut pandang pada layar

9. **Get Color**
   Mengambil sample warna

10. **Set Color**
    Mengubah warna dari suatu objek

Untuk mempermudah pemakaian *tools* yang berbeda, Anda dapat menggunakan tombol Tab untuk mengubah jenis *tools* yang sedang aktif.

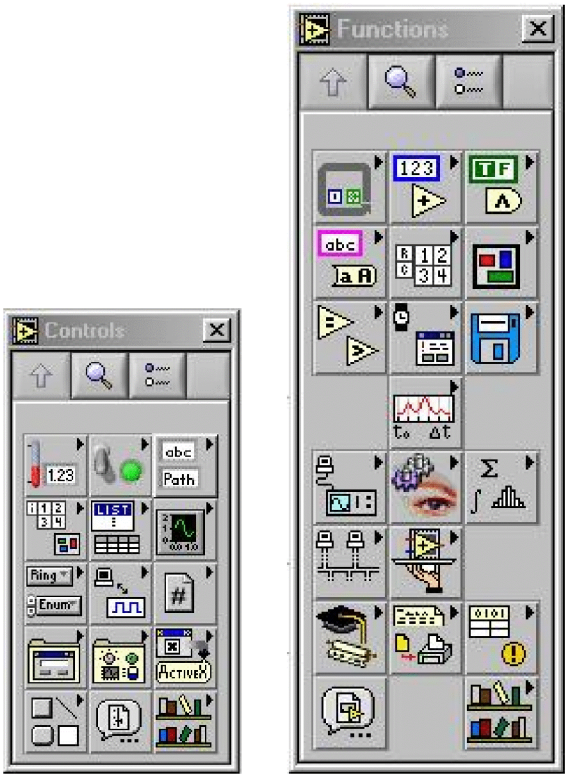
## Control Palette

Dalam pemrograman berbasis grafis, hal yang perlu dilakukan untuk membuat program adalah menaruh beberapa fungsi dan kemudian menghubungkannya dengan kabel pada bagian diagram.

Fungsi-fungsi tersebut terletak pada **Control Palette**. Banyaknya fungsi yang terletak pada **Control Palette** bervariasi tergantung seberapa lengkap LabVIEW yang diinstall. Begitu pula banyak *add-ons* (tambahan) dari LabVIEW yang akan menambah jumlah fungsi pada **Control Palette** misalnya *PID Toolkit*, *Internet Toolkit*, *Fuzzy Toolkit* dan lain-lain. Dalam membuat suatu VI kita akan banyak berurusan dengan **Control Palette**.

Cara mengakses **Control Palette** ada dua :

- Klik pada menu bar **Windows « Show Control Palette**
- klik kanan pada bagian *background* baik **block diagram** maupun **front panel**.




**Gambar 1 – 6.** Contoh dari **Control Palette** dan **Function Palette**
**Function Palette** terletak pada **Block Diagram**.
**Control Palette** terletak pada **Front Panel**.

## Terminal

**Terminal** adalah bagian dari sebuah VI atau fungsi yang menjadi tempat aliran data (sebagai input atau output data). **Terminal** identik dengan parameter pada pemrograman berbasis text. Terminal diakses dengan klik kanan pada fungsi kemudian **Show » Terminal**.

## Wires (Kabel)

Teknik wiring (pengkabelan) yang baik dibutuhkan dalam membuat suatu VI. Untuk menghubungkan sebuah fungsi dengan **control**, **constant** dan **indicator** digunakan **wires** (kabel). Apabila program sudah sangat kompleks diperlukan pengkabelan yang baik. Harus diusahakan pengkabelan yang membuat Anda akan lebih mudah untuk melihat jalannya aliran dari program Anda.

Bila kabel sudah terhubung dengan baik maka kabel akan menampilkan garis yang tidak terputus-putus dan program dapat dirunning. Tetapi jika ada suatu kesalahan, maka kabel akan menjadi garis yang terputus-putus. Dan program tidak dapat dirunning.

## Running dan Debug VI

Untuk menjalankan program, aktifkan Front Panel. Lalu tekan tombol **Run** yang berada pada sebelah kiri Menu Bar.

Untuk menghentikan program, tekan tombol **Abort** pada Menu Bar.

**Debug** adalah cara untuk mengetahui kesalahan apa yang terjadi sehingga program tidak dapat berjalan.
Jika terjadi kesalahan pada program, maka panah pada tombol *Run* akan terputus. Tekan tombol tersebut dan LabVIEW akan menampilkan daftar kesalahan pada program yang dibuat.

# 2

# Variabel dan Operator pada LabVIEW

## Control, Constant dan Indikator

Dalam membuat suatu VI kita harus banyak berurusan dengan **Control**, **Constant** dan **Indicator**.

**Control** adalah sebuah objek yang ditempatkan pada **Front Panel** yang berguna untuk memasukkan input ke dalam suatu VI. **Control** dapat dibuat melalui diagram tetapi hanya dapat dimodifikasi melalui **Front Panel**.

**Constant** adalah sebuah objek yang ditempatkan pada diagram yang berguna untuk memasukkan input ke dalam suatu VI. **Constant** hanya dapat dilihat dan dimodifikasi melalui **Block Diagram**.

**Indicator** adalah sebuah objek yang ditempatkan pada **Front Panel** yang berguna untuk menampilkan output. **Indicator** dapat dibuat melalui diagram tetapi hanya dapat dimodifikasi melalui **Front Panel**.

## Bekerja dengan Tipe Data Tertentu

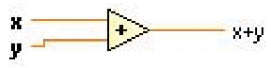
Beberapa tipe data yang sering digunakan :

- **Numerik**
  Terdiri atas beberapa sub tipe antara lain :
  **Double** : bilangan desimal dengan range –1,797E+308 sampai 1,797E+308 (simbolnya **"DBL"**)
  **Single** : bilangan desimal dengan range –3,402E+38 sampai 3,402E+38 (simbolnya **"SGL"** )
  **Long Integer** : bilangan integer / bulat dengan range –32768 sampai 32768 (simbolnya **"I32"**)
- **Boolean** : TRUE atau FALSE (simbolnya **"TF"**)
- **String** : huruf (simbolnya **"abc"**)

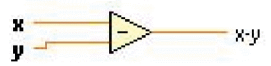
## Operator Numerik

Jenis-jenis operator **Numerik** dapat dilihat pada **Control Palette » Numeric** pada bagian **Block Diagram**. Beberapa operator yang sering dipakai adalah :

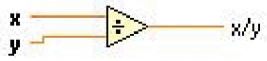
- **Add**




- **Subtract**

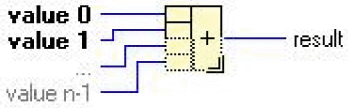


- **Divide**




- **Multiply**


- **Compound Arithmetic** : penjumlahan, pengurangan, perkalian atau pembagian dengan input lebih dari 2




- **Square Root**


- **Random Number** : output bilangan acak antara 0 dan 1


- **Integer Quotient & Remainder** : sisa pembagian


Contoh : x = 20 dan y = 6, maka R = 2 dan IQ = 3

## Operator Comparison

Jenis-jenis operator **Comparison** (perbandingan) dapat dilihat pada **Control Palette » Comparison** pada bagian **Block Diagram**. Beberapa operator yang sering dipakai adalah :

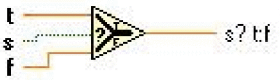
- **Equal**


- **Not Equal**


- **Greater**


- **Less**


- **Select**



## Operator Boolean

Jenis-jenis operator **Boolean** dapat dilihat pada **Control Palette » Boolean** pada bagian **Block Diagram**. Beberapa operator yang sering dipakai adalah :

- **And**

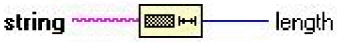

- **Or**


- **Not**

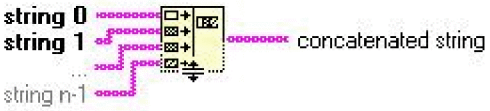

## Operator String

Jenis-jenis operator **Stirng** dapat dilihat pada **Control Palette » String** pada bagian **Block Diagram**. Beberapa operator yang sering dipakai adalah :

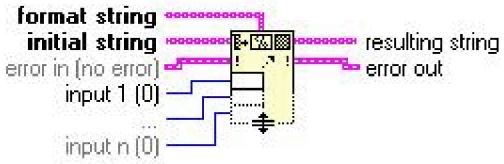
- **String Length**




- **Concenate String** : menyatukan beberapa string




- **Format Into String** : mengubah tipe data menjadi string




## Formula Node

**Formula Node** adalah suatu metode berupa kotak dimana Anda dapat memasukkan formula perhitungan atau persamaan matematik secara text.
**Formula Node** akan sangat berguna apabila persamaan tersebut mempunyai variable yang sangat banyak sehingga menjadi kompleks.

Untuk dapat menggunakan **Formula Node** ini, Anda perlu memahami fungsi-fungsi dan operasi aritmatik pada LabVIEW.

**Tabel 2.1. Beberapa Fungsi Penting Pada Formula Node**

| *Fungsi* | *Keterangan* |
|---|---|
| ***Abs(x)*** | Menghitung nilai mutlak dari x |
| ***acos(x)*** | Menghitung inverse cosinus dari x dalam radians |
| ***asin(x)*** | Menghitung inverse sinus dari x dalam radians |
| ***atan(x)*** | Menghitung inverse tangent dari x dalam radians |
| ***cos(x)*** | Menghitung cosinus dari x dalam radians |
| ***exp(x)*** | Menghtiung exponensial dari x |
| ***int(x)*** | Mengembalikan nilai x pada bilangan bulat terdekat |
| ***ln(x)*** | Menghitung logaritma natural dari x |
| ***log(x)*** | Menghitung logaritma basis 10 dari x |
| ***log2(x)*** | Menghitung logaritma basis 2 dari x |
| ***max(x,y)*** | Membandingkan x dan y untuk mencari nilai terbesar |
| ***min(x,y)*** | Membandingkan x dan y untuk mencari nilai terkecil |
| ***mod(x,y)*** | Mencari sisa bagi x terhadap y |
| ***pi(x)*** | Merepresentasikan nilai 3,14159…<br>pi(x) = x * π |
| ***rand()*** | Menghasilkan bilangan random antara 0 dan 1 |
| ***sin(x)*** | Menghitung sinus dari x dalam radians |
| ***sqrt(x)*** | Menghitung akar kuadrat dari x |
| ***tan(x)*** | Menghitung tangent dari x dalam radians |

**Tabel 2.2. Beberapa Operator Penting Pada Formula Node**

| *Operator* | *Keterangan* |
|---|---|
| ***+ dan -*** | Jumlah dan kurang |
| **** dan /*** | Kali dan bagi |
| ** | Perpangkatan |

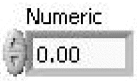
# Latihan 2.1 : Membuat Fungsi menggunakan Operator Numerik

Pada program berikut ini Anda akan membuat program VI yang mengkalkulasi operasi fungsi berikut :

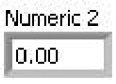
$$f(x,y)=\frac{x-y}{y^2+x^2+2} \qquad (2.1)$$

## *Membangun Display pada Front Panel*

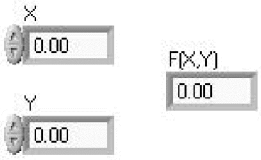
1. Buat VI baru.
2. Aktifkan **Front Panel**.
3. Tampilkan **Controls palette**, tidak terlihat pada **front panel**, pilih **Window » Show Controls Palette** untuk menampilkannya
4. Gerakkan cursor melalui icon-icon yang ada pada **Controls Palette** untuk mencari **Numeric Controls Palette**.
   Perhatikan bahwa saat Anda menggerakkan cursor melalui icon-icon pada Controls palette, maka judul dari subpalette yang ada diatas susunan icon-icon akan berubah-ubah sesuai judul dari tiap icon masing-masing.




5. Klik **Numeric Controls** icon untuk mengakses **Numeric Controls** palette. Pilih **Digital Control** dari Numeric Controls palette lalu tempatkan pada **front panel**.
6. Buat sebuah lagi **Digital Control** dengan cara yang sama seperti diatas. (Pilih **Numeric » Digital Control** pada **Controls Pallete**)




7. Buat sebuah **Digital Indicator** dengan cara pilih **Numeric » Digital Indicator**.
8. Aktifkan **Edit Text Tools** pada **Tools Palette**. Edit label **Digital Control** menjadi "X" dan "Y". Edit label **Digital Indicator** menjadi "F(X,Y)"
9. Sesuaikan hasilnya dengan gambar berikut




**Gambar 2 – 1.** Tampilan Front Panel program latihan 2.1

## *Membangun VI Diagram*

Untuk membuat *front panel* berfungsi sesuai yang diharapkan, maka Anda harus mengatur *source code* dan menghubungkan setiap objek pada *Block Diagram*. Lakukan langkah-langkah berikut untuk memfungsikan display pada *front panel* yang telah dibuat menjadi pengkalkulasi fungsi yang diberikan

1. Aktifkan **Block Diagram**.
2. Jika **Functions Palette**, tidak terlihat pada **Block Diagrams**, pilih **Window » Show Functions Palette** untuk menampilkannya.

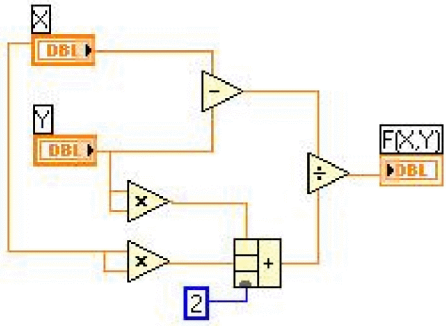
3. Dari **Functions Palette**, pilih **Numeric » Subtract**. Letakkan disebelah kanan terminal X dan Y.

4. Pilih **Numeric » Multiply**. Letakkan disebelah kanan terminal Y
5. Pilih **Numeric » Multiply**. Letakkan disebelah bawah operator multiply sebelumnya

6. Pilih **Numeric » Compound Aritmetic**. Letakkan disebelah kanan Square Root. Resize menjadi 3 input.

7. Pilih **Numeric » Numeric Constant**. Letakkan di kiri bawah **Compound Aritmetic**. Edit angkanya menjadi ”2”.

8. Pilih **Numeric » Divide**. Letakkan disebelah kanan atas **Compound Aritmetic**
9. Aktifkan **Wiring Tools**. Lakukan wiring sehingga menjadi seperti gambar 2 – 2.




**Gambar 2 – 2**. **Block Diagram** dari program latihan 2 – 1.

10. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini. Simpanlah dengan judul “*Fungsi.vi*”

## *Menjalankan Program*

Untuk menjalankan program, aktifkan Front Panel. Lalu tekan tombol **Run** yang berada pada sebelah kiri Menu Bar.

## *Pengujian Program*

Untuk menguji apakah program telah sesuai dengan operasi fungsi yang diharapkan, uji nilai F(X,Y) untuk beberapa nilai X dan Y berikut ini :

**Tabel 2.3** Nilai Uji Fungsi (2.1)

| X | Y | F(X,Y) |
|---|---|---|
| 1 | 0 | 0,33 |
| 0 | 1 | -0,33 |
| -1 | 1 | -0,5 |

**Keterangan** : Anda dapat mengubah-ubah nilai X dan Y dengan cara mengedit nilai **Digital Control** X dan Y pada **Front Panel**.

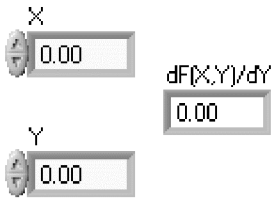
# Latihan 2.2 : Membuat Fungsi menggunakan Formula Node

Pada program berikut ini Anda akan membuat program VI yang mengkalkulasi turunan fungsi (2.1) terhadap variable y.

$$\frac{\partial f(x,y)}{\partial y} = \frac{-x^2 + y^2 - 2xy - 2}{\left(x^2 + y^2 + 2\right)^2} \qquad (2.2)$$

## *Membangun Display pada Front Panel*

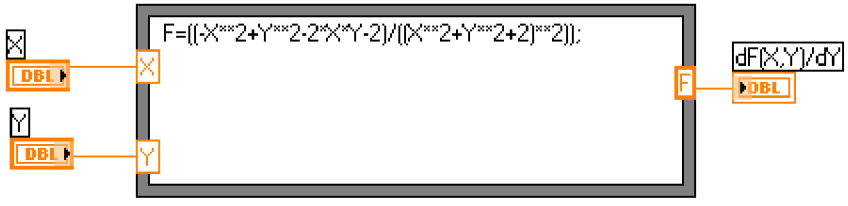
Lakukan langkah yang sama seperti pada Latihan 2.1. Tetapi edit label **Digital Indicators** menjadi dF(X,Y)/dY sehingga menjadi seperti gambar berikut.




**Gambar 2 – 3.** Tampilan Front Panel program latihan 2.2

## *Membangun VI Diagram*

1. Aktifkan **Block Diagram**
2. Dari **Functions Palette**, pilih **Structure » Formula Node**. Drag **Formula Node** agar berukuran cukup panjang.
3. Klik kanan pada sisi bagian kiri **Formula Node**. Pilih ”**Add Input**”. Edit namanya menjadi X.
4. Klik kanan pada sisi bagian kiri **Formula Node**. Pilih ”**Add Input**”. Edit namanya menjadi Y.
5. Klik kanan pada sisi bagian kanan **Formula Node**. Pilih ”**Add Output**”. Edit namanya menjadi F.
6. Aktifkan **Edit Text Tools**. Tulis persamaan berikut didalam **Formula Node** :
   F=((-X**2+Y**2-2*X*Y-2)/((X**2+Y**2+2)**2));
   **Perhatikan** :
   - Penamaan variabel pada persamaan harus sama dengan penamaan variabel pada input dan output
   - Berikan tanda ”;” setiap diakhir suatu persamaan
7. Aktifkan **Wiring Tools**, lakukan wiring menjadi seperti gambar 2 – 4.




**Gambar 2 – 4**. **Block Diagram** dari program latihan 2 – 2.

8. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini. Simpanlah dengan judul "*Diff fungsi terhadap Y.vi*"

## *Pengujian Program*

Untuk menguji apakah program telah sesuai dengan operasi fungsi yang diharapkan, uji nilai dF(X,Y)/dY untuk beberapa nilai X dan Y berikut ini :

**Tabel 2.4** Nilai Uji Fungsi (2.2)

| X | Y | dF(X,Y)/dY |
|---|---|---|
| 0 | 0 | -0,5 |
| 1 | 0 | -0,33 |
| 1 | 1 | -0,25 |
| 0 | 1 | -0,11 |

# 3

# SubVI dan Variabel

## SubVI

**SubVI** adalah VI yang dijadikan sebuah diagram blok oleh VI lainnya. **SubVI** berfungsi seperti sebuah fungsi.

Saat Anda akan menggunakan suatu VI sebagai **SubVI**, maka Anda perlu menyesuaikan *icon*, *connector* serta *input-output*nya.

## Variabel

Sebenarnya **Control** dan **Variabel**-lah yang biasanya dimaksud dengan **Variabel**.

Dalam LabVIEW ada 2 macam **Variabel** yaitu :

- **Local Variabel**
  Adalah variabel yang bisa diakses hanya dari lingkungan file VI itu sendiri
- **Global Variabel**
  Adalah variabel yang bisa diakses dari file VI yang berbeda-beda.

**Local** dan **Global Variabel** termasuk dalam kelompok **Structure** pada **Control Pallete**. Kedua **Variabel** tersebut menyediakan mode untuk menuliskan variable atau membaca variabel.
**Local Variable** berfungsi untuk mengakses **Control** dan **Indicator** pada VI tersebut tanpa menghubungkannya dengan kabel. Fitur ini akan sangat berguna bila VI sudah berkembang menjadi sangat kompleks dan sulit untuk menghubungkan beberapa **Control** dan **Indikator** dengan kabel.
Penting untuk diperhatikan adalah mode pada **Local Variabel**. **Local Variable** dengan mode **"Write Local**" harus dihubungkan dengan **Control** sementara **Local Variable** dengan mode **"Read Local"** harus dihubungkan dengan **Indikator**.

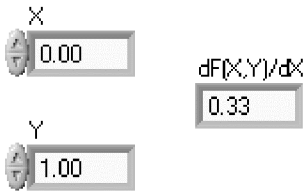
# Latihan 3.1 : Membuat Fungsi menggunakan Local Variable

Pada latihan ini Anda akan mempelajari cara pemanfaatan **Local Variable** untuk membuat suatu operasi fungsi yang tidak sederhana.
Fungsi yang akan dibuat adalah turunan fungsi (2.1) terhadap variabel X.

$$\frac{\partial f(x,y)}{\partial x} = \frac{-x^2 + y^2 + 2xy + 2}{\left(x^2 + y^2 + 2\right)^2} \qquad (3.1)$$

## *Membangun Display pada Front Panel*

Lakukan langkah yang sama seperti pada Latihan 2.1. Tetapi edit label **Digital Indicators** menjadi dF(X,Y)/dX sehingga menjadi seperti gambar berikut.

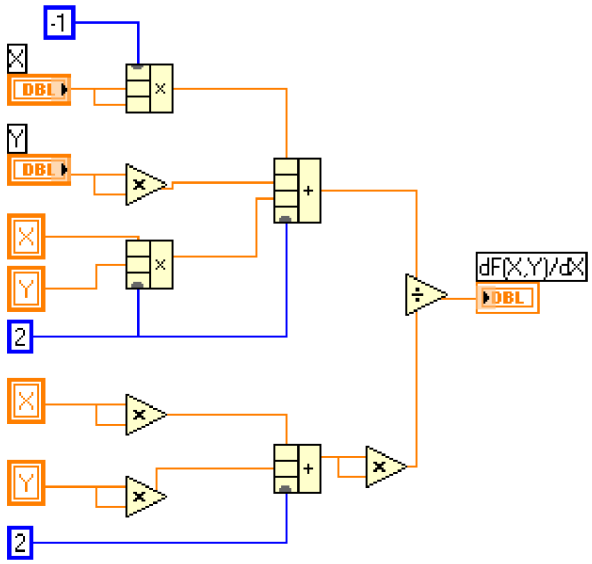



**Gambar 3 – 1.** Tampilan Front Panel program latihan 3.1

## *Membangun VI Diagram*

1. Aktifkan **Block Diagram**
2. Posisikan Terminal Y dibawah Terminal X. Dan posisikan Terminal dF(X,Y)/dX disebelah kanan bawah Terminal X
3. Pilih **Numeric » Compound Aritmetic**. Letakkan disebelah kanan Terminal X. Resize menjadi 3 input, klik kanan, pilih **Change Mode » Multiply**.
4. Pilih **Numeric » Numeric Constant**. Letakkan diatas **Compound Aritmetic**. Edit nilainya menjadi –1.
5. Pilih **Numeric » Multiply**. Letakkan disebelah kanan terminal Y
6. Pilih **Numeric » Compound Aritmetic**. Letakkan disebelah kanan **Multiply**. Resize menjadi 4 input.
7. Pilih **Structure » Local Variable**. Buat 2 buah **Local Variable**, letakkan disebelah berurutan disebelah bawah Terminal X. Ubah modenya menjadi **Read Local** (dengan cara klik kanan pada *Local Variable* lalu pilih **Change to Read**). Klik kanan, kemudian **Select Item** masing-masing menjadi *"X"* dan "*Y*".
8. Pilih **Numeric » Numeric Constant**. Letakkan dibawah **Local Variable Y**. Edit nilainya menjadi 2.
9. Pilih **Numeric » Compound Aritmetic**. Letakkan disebelah kanan **Local Variable**. Resize menjadi 3 input, klik kanan, pilih **Change Mode » Multiply**.
10. Buat kembali dua buah **Local Variable** dibawah **Numeric Constant** "2". **Selecet Item** masing-masing "X" dan "Y". Tambahkan dibagian bawahnya **Numeric Constant** dengan nilai "2".
11. Pilih **Numeric » Multiply**. Letakkan disebelah kanan **Local Variable X**
12. Pilih **Numeric » Multiply**. Letakkan disebelah kanan **Local Variable Y**

13. Pilih **Numeric » Compound Aritmetic**. Letakkan disebelah kanan **Multiply**. Resize menjadi 3 input.
14. Pilih **Numeric » Multiply**. Letakkan disebelah kanan **Compound Aritmetic**.
15. Pilih **Numeric » Divide**. Letakkan disebelah kanan atas **Compound Aritmetic**.
16. Aktifkan **Wiring Tools**, lakukan wiring menjadi seperti gambar 3 – 2.




**Gambar 3 – 2**. **Block Diagram** dari program latihan 3 – 1.

17. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini. Simpanlah dengan judul "*Diff fungsi terhadap X.vi*"

## *Pengujian Program*

Untuk menguji apakah program telah sesuai dengan operasi fungsi yang diharapkan, uji nilai dF(X,Y)/dX untuk beberapa nilai X dan Y berikut ini :

**Tabel 3.1** Nilai Uji Fungsi (3.1)

| X | Y | dF(X,Y)/dX |
|---|---|---|
| 0 | 0 | 0,5 |
| 1 | 0 | 0,11 |
| 1 | 1 | 0,25 |
| 0 | 1 | 0,33 |

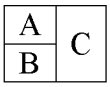
## Latihan 3.2 : Membuat Program VI Sebagai SubVI

**SubVI** adalah VI yang dijadikan sebuah diagram blok oleh VI lainnya.
Pada latihan ini, program latihan 3.1. akan dibuat menjadi SubVI. Lakukan langkah-langkah berikut ini :

1. Pilih **File « Open** pada **Menu Bar** untuk mengambil file "*Diff fungsi terhadap X.vi*"
2. Aktifkan Front Panel, lalu klik kanan pada icon di Menu Bar. Pilih **"Show Connector"**. Maka akan terlihat icon seperti disamping :

**Catatan :** Perhatikan bahwa saat icon ditampilkan, maka menu cursor akan berubah menjadi **Wiring Tool**.

Untuk memudahkan langkah berikutnya, setiap kotak-kotak tersebut diberi nama sebagai berikut :




3. Klik kiri Control *"X"* lalu klik kiri pada A
   Klik kiri Control *"Y"* lalu klik kiri pada B
   Klik kiri Control *"dF(X,Y)/dX"* lalu klik kiri pada C

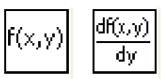
**Catatan :** Jika saat Anda melakukan wiring ini tidak ada reaksi, mungkin menu cursor belum diset menjadi **Wiring Tool**.


4. Klik kanan pada icon di **Menu Bar** lagi, pilih **"Show Icon"**. Kemudian Klik kanan lagi, pilih **"Edit Icon**". Maka akan tampil menu untuk mengedit gambar dari icon ini. Anda bisa membuat Icon sesuai kreasi Anda, atau buatlah seperti icon pada gambar disamping.
5. Pilih **File « Save** pada **Menu Bar** untuk menyimpan perubahan pada VI ini.

Anda telah selesai menjadikan Modul Pengontrol PID ini sebagai **subVI** pada **LabVIEW**.

## Latihan 3.3 : Latihan Mandiri

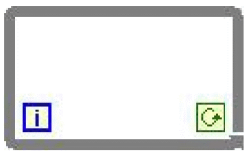
Buatlah file *"Fungsi.vi"* dan "*Diff fungsi terhadap Y.vi*" sebagai subVI dengan langkah-langkah yang sama. Anda dapat memodifikasi tampilan icon dari masing-masing file menjadi seperti berikut.



# 4

# Struktur Keputusan, Delay dan Chart

## While Loop



**While Loop** adalah metode perulangan dimana ada kondisi yang harus dipenuhi supaya *looping* bisa berjalan terus.

Pada bagian kanan bawah **While Loop** terdapat **Conditional Terminal**. Fungsi input pada **Conditional Terminal** adalah untuk mengontrol apakah **While Loop** akan terus dieksekusi ataukah akan berhenti. Jika pada **Conditional Terminal** diberi input *True*, maka **While Loop** akan terus dieksekusi. Jika pada **Conditional Terminal** diberi input *False*, maka **While Loop** akan berhenti dieksekusi.

## Delay

Fungsi **Wait** berupa gambar jam diatas adalah untuk memperlambat looping dengan satuan *milidetik*. Bila *looping* tidak diberikan fungsi tersebut, maka *looping* akan dieksekusi secepat mungkin sehingga akan sulit diikuti oleh mata.

## Chart

**Chart** adalah indicator untuk memplot data numeric yang akan selalu berubah seiring dengan masuknya data baru. **Chart** selalu dieksekusi didalam *looping*. Perlu diperhatikan disini adalah jika Anda hendak memplot beberapa kurva pada satu chart, maka Anda harus menggabungkan data-data kurva tersebut menggunakan **cluster** (tidak bisa menggunakan **array**)

## Shift Register

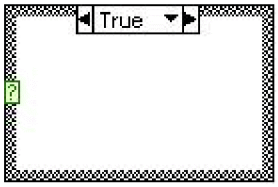
**Shift Register** (digunakan pada **While Loop** dan **For Loop**) mentransfer suatu data yang digunakan atau dihitung pada iterasi sebelumnya. Anda dapat membuat **shift register** dengan mengklik kanan pada suatu *looping* baik **While Loop** maupun **For Loop**.

**Shift Register** terdiri dari sepasang terminal yang berhadapan pada sisi vertical dari *looping*. Data yang berakhir pada setiap iterasi akan memasuki terminal di sebelah kanan sementara data dari iterasi sebelumnya akan muncul dari terminal di sebelah kiri.

**Shift Register** dapat menampung berbagai macam tipe data baik numeric, boolean ataupun string, tetapi hanya dapat dipakai 1 tipe data di setiap *looping*.

**Shift Register** juga dapat digunakan untuk mengingat nilai dari beberapa iterasi sebelumnya. Hal ini dapat dilakukan dengan menambah terminal di sebelah kiri. Klik kanan pada terminal kemudian pilih **Add Element**.

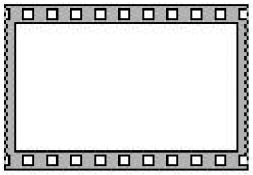
## Case




**Case Structure** adalah metode pencabangan aliran program. **Case Structure** mempunyai beberapa subdiagram dimana hanya salah satu yang akan dieksekusi selama program dijalankan. **Case Structure** bekerja seperti IF (kondisi) THEN (aksi1) ELSE (aksi2) dalam pemrograman berbasis text.

Kotak hijau pada bagian sebelah kiri **Case Structure** merupakan kondisi input. **Case** dapat menerima input berupa Boolean maupun Numeric.
Jika input **Case** berupa Boolean, maka jika input bernilai **True**, maka subdiagram dengan label **True** yang akan dieksekusi dan demikian jika sebaliknya.
Jika input **Case** berupa Numeric, maka jika nilai input bernilai 0, maka subdiagram dengan label 0 yang akan dieksekusi dan demikian seterusnya.
**Keterangan** : Label subdiagram akan segera menyesuaikan dengan jenis input yang diberikan pada input **Case**.

## Sequence



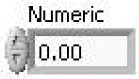
**Sequence** adalah kumpulan dari beberapa subdiagram yang dieksekusi secara sekuensial atau berurutan. Jadi hasil perhitungan dari proses sebelumnya dapat dipakai dalam perhitungan berikutnya.

# Latihan 4.1 : Membuat Grafik dari Fungsi Satu Variabel

Pada program berikut ini Anda akan membuat program VI yang menampilkan grafik 1D dari suatu fungsi dengan satu variabel. Fungsinya adalah :

$$Y(x) = x - \tfrac{1}{6}x^3 + \tfrac{1}{120}x^5 - \tfrac{1}{5040}x^7 \qquad (4.1.)$$

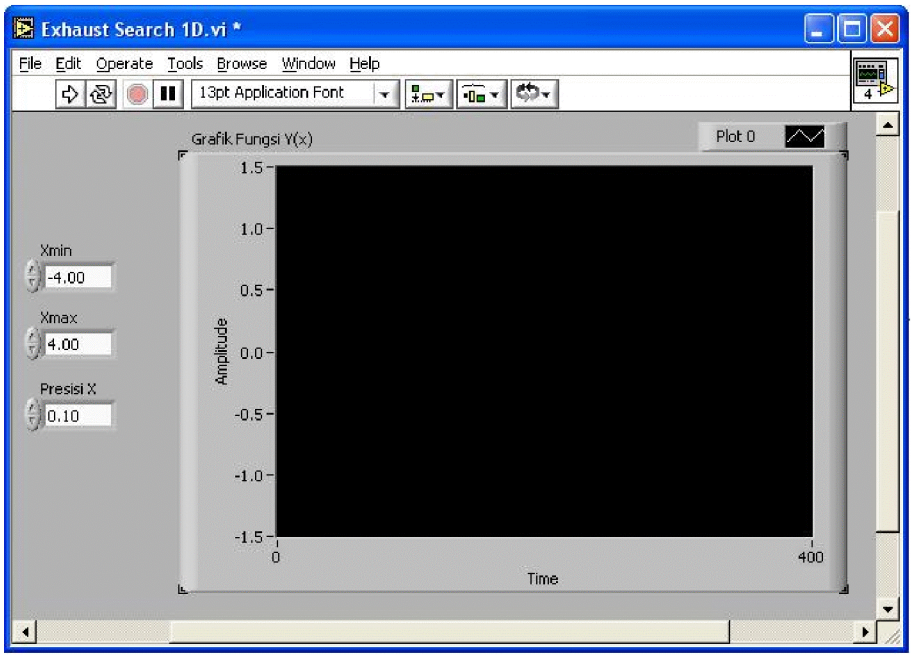
## *Membangun Display pada Front Panel*




1. Buat VI baru. Save dengan nama "*Exhaust Search 1D*"
2. Aktifkan **Front Panel**.
3. Buat 3 buah **Digital Control** dengan cara pilih **Numeric » Digital Control**.
4. Edit labelnya menjadi "*X min*" , "*X max*" dan "*Presisi X*"
5. Isikan nilai Xmin = -4, Xmax = 4 dan Presisi X= 0.1
6. Pilih **Graph » Waveform Chart**. . Edit labelnya menjadi "Grafik Fungsi Y(x)". Klik kanan pada bagian skala Y. Pilih **Y Scale » AutoScale Y**. Ukuran **Waveform Chart** dapat Anda ubah menggunakan fungsi drag.

**Catatan :** AutoScale berfungsi agar skala pada sumbu tersebut dapat berubah-ubah mengikuti besar nilai yang diberikan.

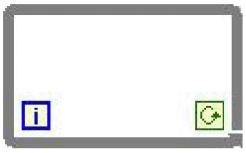
7. Atur posisi dan nilai dari **Digital Control** sesuai gambar 4 – 1 berikut.




**Gambar 4 – 1.** Tampilan Front Panel program latihan 4.1

## *Membangun VI Diagram*

1. Aktifkan **Block Diagram**
2. Dari **Functions Palette**, pilih **Structure » While Loop**. Drag **While** Loop, agar semua **Control** dan **Indicator** pada *Block Diagrams* berada dalam **While Loop**.

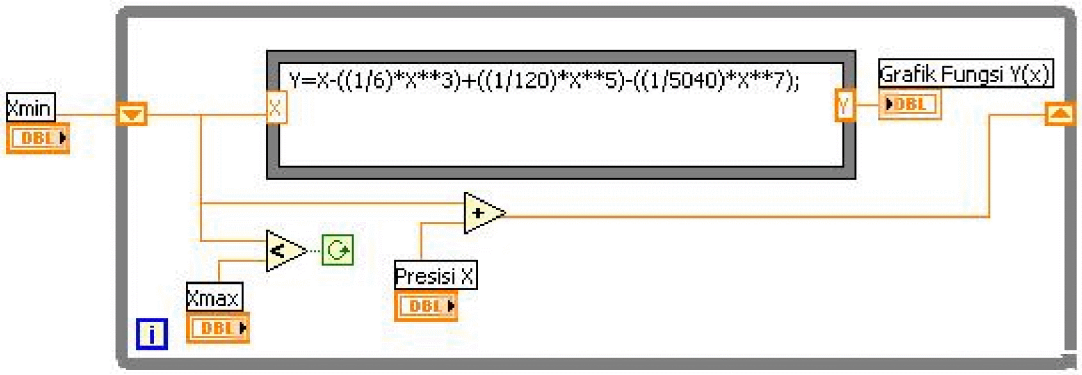


3. Buat **Shift Register**, klik kanan pada border **While Loop**, pilih **Add Shift Register**. Shift Register ini digunakan untuk mengeluarkan data X dari iterasi sebelumnya

4. Pilih **Comparison » Less?**. Tempatkan disebelah kanan bawah **Shift Register**.
5. Pindahkan posisi **Conditional Terminal** disebelah kanan **Less?**

6. Pilih **Numeric » Add**. Letakkan disebelah kanan atas **Less?**

7. Dari **Functions Palette**, pilih **Structure » Formula Node**. Drag **Formula Node** agar berukuran cukup panjang.
8. Klik kanan pada sisi bagian kiri **Formula Node**. Pilih "**Add Input**". Edit namanya menjadi X.
9. Klik kanan pada sisi bagian kanan **Formula Node**. Pilih "**Add Output**". Edit namanya menjadi Y.
10. Aktifkan **Edit Text Tools**. Tulis persamaan berikut didalam **Formula Node** :
    Y=X-((1/6)*X**3)+((1/120)*X**5)-((1/5040)*X**7);
11. Lakukan wiring sehingga diagram menjadi seperti gambar 4 – 2.




**Gambar 4 – 2. Block Diagram** dari program latihan 4 – 2.

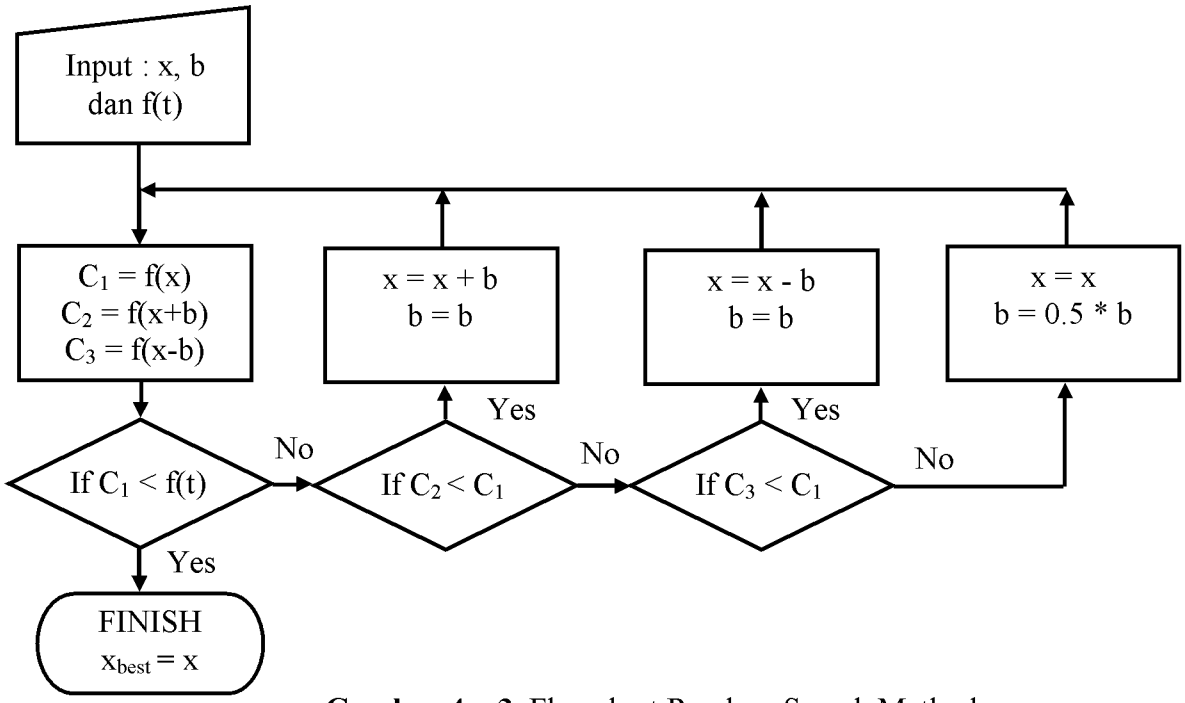
12. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini

## *Menjalankan Program*

Tekan tombol **Run** yang berada pada sebelah kiri Menu Bar.
Anda akan memperoleh grafik fungsi Y(x) dari nilai x = -4 hingga x = 4 dengan selang data 0,1.
Untuk membersihkan kembali tampilan **Weveform Chart**, klik kanan pada **Waveform Chart**, pilih **Data Operations » Clear Chart**.

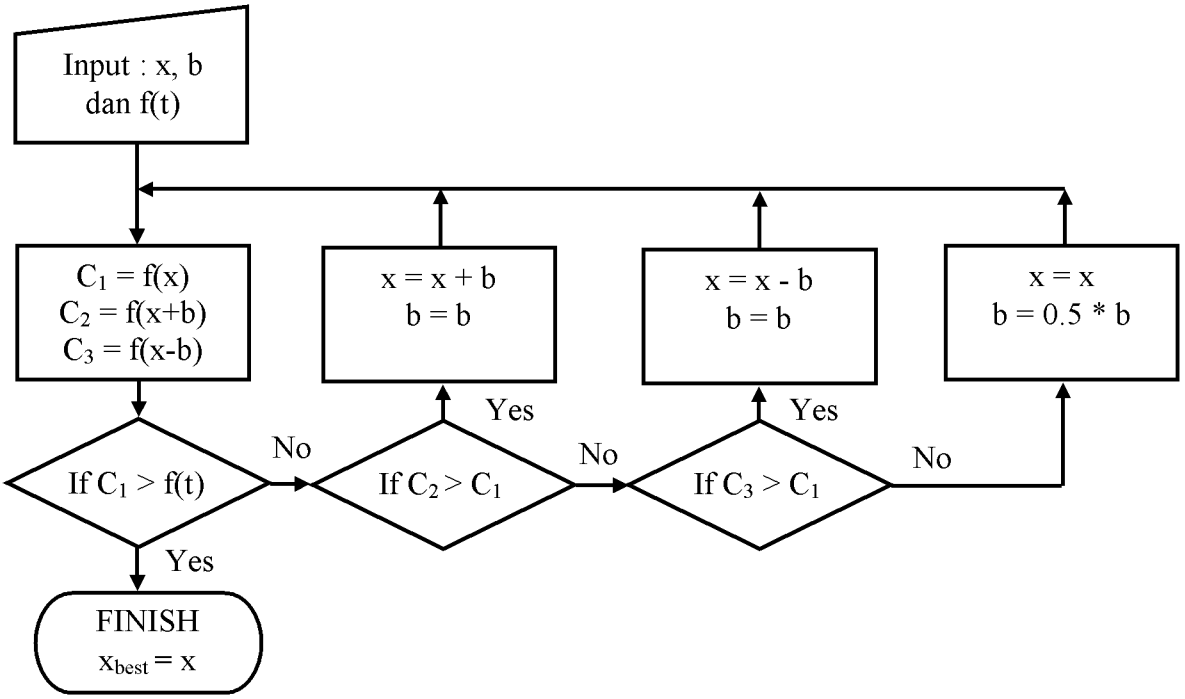
## Latihan 4.2 : Membuat Program Random Search Method

Pada program berikut ini Anda akan membuat program Random Search Method dengan algoritma untuk mencari nilai minimum dari suatu fungsi f(x) adalah sebagai berikut




**Gambar 4 – 3**. Flowchart Random Search Method untuk mencari nilai minimum

Dan algoritman untuk mencari nilai maksimum adalah sebagai berikut :

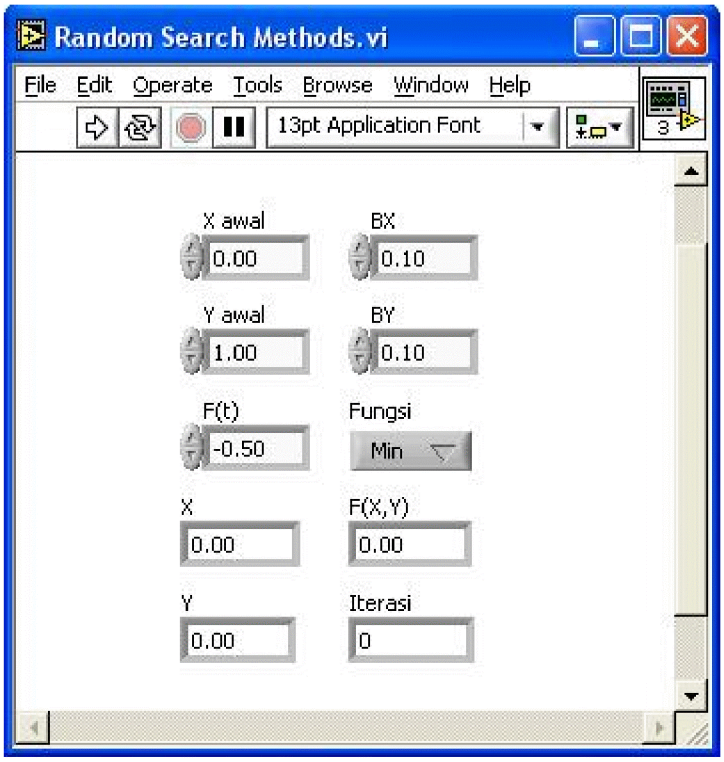



**Gambar 4 – 4**. Flowchart Random Search Method untuk mencari nilai maximum

Perhatikan bahwa perbedaan antara gambar 4 – 3 dan gambar 4 – 4 terletak pada fungsi pembanding pada bagian decisionnya.

## *Membangun Display pada Front Panel*

1. Buat VI baru. Save as dengan nama "*Random Search Methods*"
2. Aktifkan **Front Panel**. Dan buat menjadi seperti gambar 4 – 5. Edit dan atur nilai dari setiap **Digital Control** seperti pada gambar




**Gambar 4 – 5**. Tampilan Front Panel program latihan 4.2

Pada **Front Panel** diatas terdiri dari 5 buah **Digital Control**, 4 buah **Digital Indicator** dan 1 buah **Menu Ring**.
Untuk menampilkan **Menu Ring** lakukan langkah-langkah berikut.

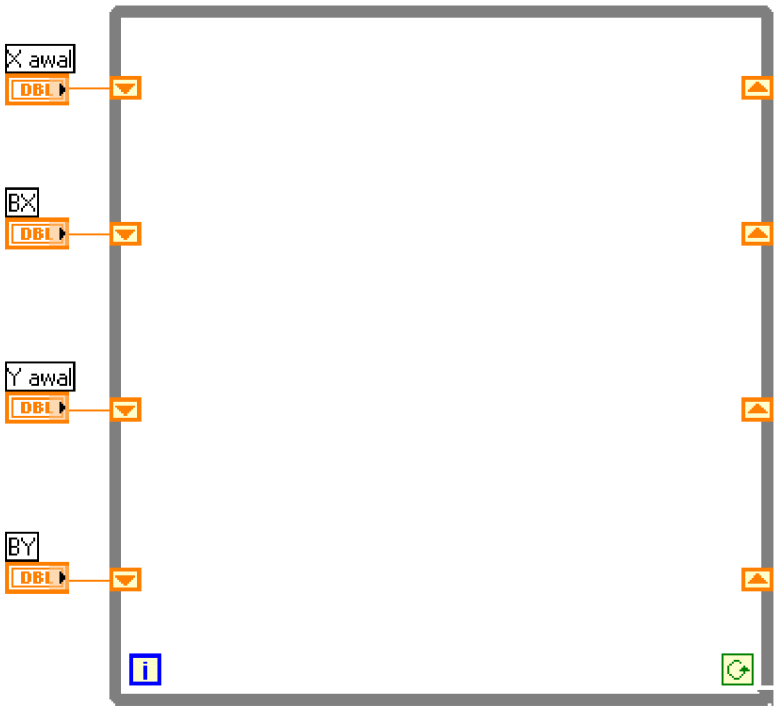

3. Dari Numeric Controls Palette, pilih **Ring & Enum » Menu Ring**.

4. Aktifkan **Edit Text Tools**. Ubah labelnya menjadi "Fungsi". Klik pada bagian tengan **Menu Ring**, isikan label "Min"
5. Klik kanan pada **Menu Ring**, pilih **Add Item After**. Klik pada bagian tengahnya, isikan label "Max"

**Menu Ring** digunakan untuk memilih pilihan dari listing String yang ada. Pada program Random Search Methods ini, **Menu Ring** digunakan untuk menyediakan pilihan apakah program Random Search Methods akan mencari nilai maksimum atau mencari nilai minimum.

## *Membangun VI Diagram*

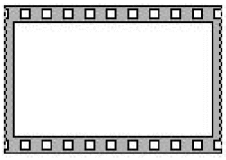
1. Aktifkan **Block Diagram**
2. Dari **Functions Palette**, pilih **Structure » While Loop**. Drag **While** Loop agar berukuran cukup besar.
3. Buat empat buah **Shift Register**. Dan posisikan masing-masing **Digital Control** disebelah kiri luar **While Loop** seperti pada gambar 4 – 6.



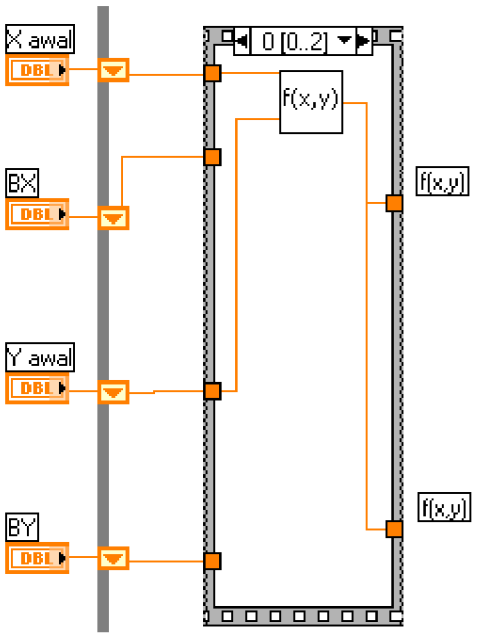

**Gambar 4 – 6**. **Block Diagram** Latihan 4.2.

Program ini akan terdiri **Sequence** yang terdiri dari tiga proses.
Proses pertama untuk menghitung f(x,y)
Proses kedua untuk menghitung f(x+b,y) dan f(x,y+b)
Proses ketiga untuk menghitung f(x-b,y) dan f(x,y-b)

4. Buat **Sequence** pertama. Pilih **Structure » Sequence**. Buat berukuran sedang. Letakkan disebelah kanan **Shift Register**.

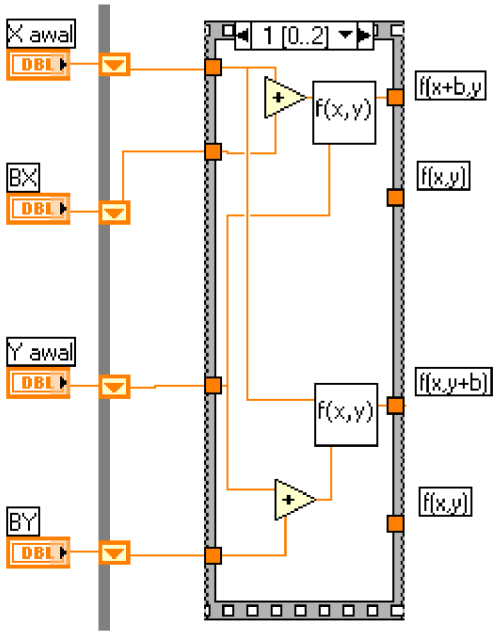


5. Buat Sequence menjadi 3 frame. Cara menambah frame dengan klik kanan di bagian atas **Sequence**, pilih "**Add Frame After**".
   **Keterangan** : 3 frame berarti terdiri dari frame 0, frame 1 dan frame 2.
6. **Frame 0 : Menghitung f(x,y)**
   Panggil subVI *"Fungsi.vi"* dengan cara pada Control Palette pilih **Select a VI…** lalu akan tampil Windows **Choose the VI to open**. Pilih file *"Fungsi.vi"*. Tempatkan dibagian pada frame 0 **Sequence**.
   Lakukan langkah-langkah yang diperlukan agar diperoleh diagram frame 0 seperti gambar 4 - 7.



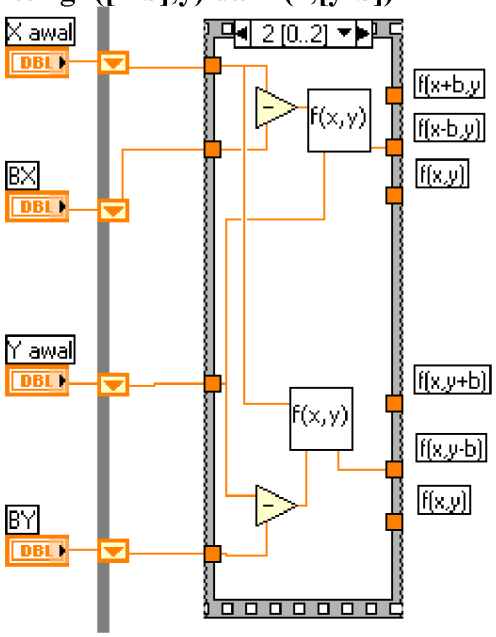

**Gambar 4 – 7.** Frame 0 Sequence

7. **Frame 1 : Menghitung f([x+b],y) dan f(x,[y+b])**




**Gambar 4 – 8.** Frame 1 Sequence

8. **Frame 2 : Menghitung f([x-b],y) dan f(x,[y-b])**

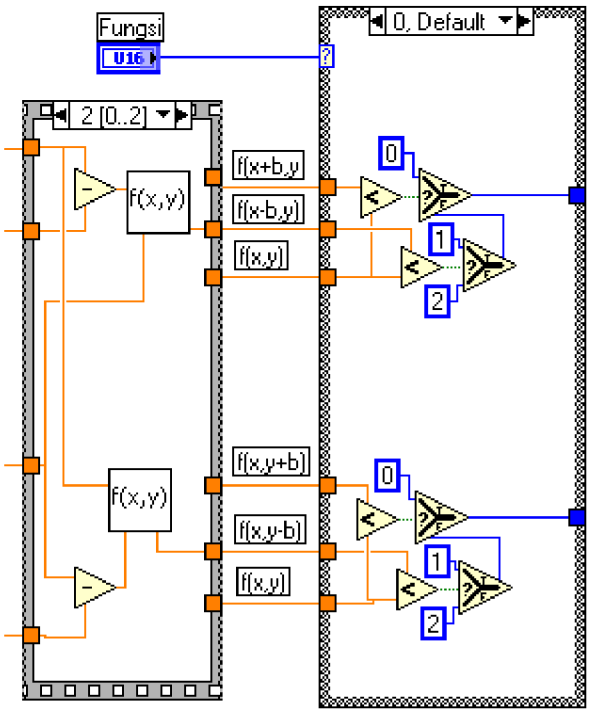



**Gambar 4 – 9.** Frame 2 Sequence

Langkah selanjutnya adalah untuk menentukan apakah program akan mencari nilai maksimum atau nilai minimum.

9. Pilih **Structure » Case Structure**. Drag dan letakkan disebelah kanan **Sequence**. Hubungkan input **Case Structre** (kotak berisi tanda ?) dengan **Menu Ring** (berlabel "Fungsi").
   Setelah terhubung, maka label **Case Structure** yang semula berlabel **True** atau **False** akan berubah menjadi **"0, default"** dan **"1"**.
   Case 0 akan berisi perintah agar program mencari nilai minimum.
   Case 1 akan berisi perintah agar program mencari nilai maksimum. [Perhatikan kembali algoritma pada gambar 4 – 3 dan 4 – 4]

10. **Case 0 : Pilihan agar program mencari nilai minimum**
    Lakukan langkah-langkah yang diperlukan sehingga **Case 0** memiliki diagram seperti gambar 4 – 10.




**Gambar 4 – 10.** Case 0

Operator dengan simbol seperti disamping dapat diakses dengan cara **Comparison » Less?**.

Operator dengan simbol seperti disamping dapat diakses dengan cara **Comparison » Select**.

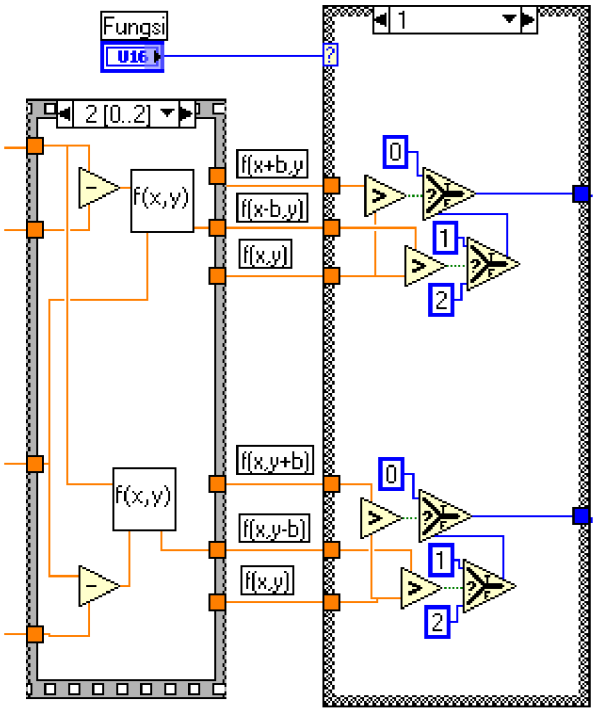
Diagram pada case 0 diatas berisikan perintah sebagai berikut :

- Jika f(x+b,y) < f(x,y) maka output atas akan bernilai 0
- Jika f(x-b,y) < f(x,y) maka output atas akan bernilai 1
- Jika f(x+b,y) > f(x,y) dan f(x-b,y) > f(x,y) maka output atas akan bernilai 2

- Jika f(x,y+b) < f(x,y) maka output bawah akan bernilai 0
- Jika f(x,y-b) < f(x,y) maka output bawah akan bernilai 1
- Jika f(x,y+b) > f(x,y) dan f(x,y-b) > f(x,y) maka output bawah akan bernilai 2

**Keterangan** : Yang dimaksud output atas adalah output dari **Case Structure** yang keluar dari kotak biru yang terletak dibagian kanan atas **Case Structure**. Demikian juga output bawah.

11. **Case 0 : Pilihan agar program mencari nilai maksimum**

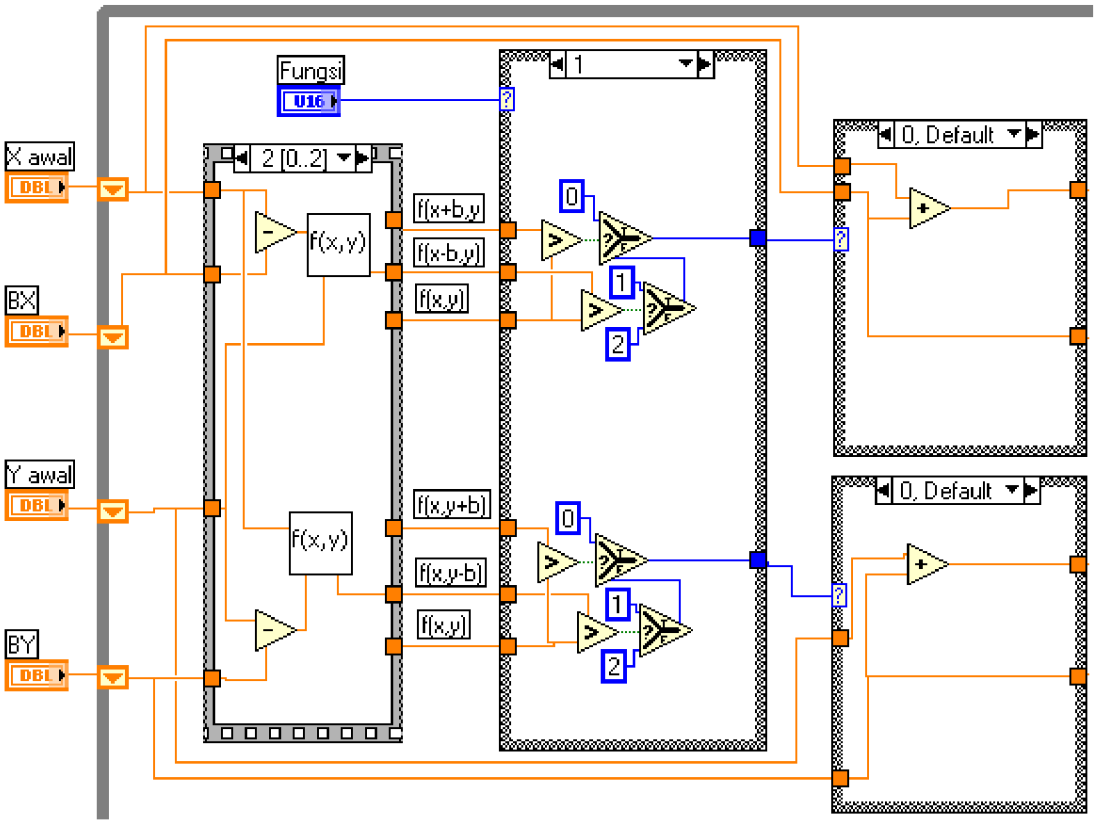



**Gambar 4 – 11.** Case 1

Langkah berikutnya adalah menyediakan kelompok perintah yang akan dieksekusi untuk setiap nilai output **Case Structure** diatas.

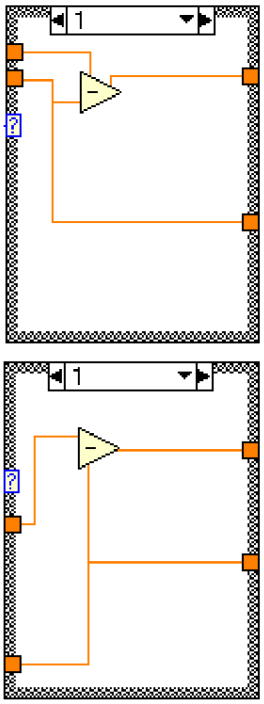
12. Buat kembali **Case Structure** kedua yang diletakkan disebelah kanan atas **Case Structure** pertama. Hubungkan inputnya dengan output atas **Case Structure** pertama.
13. Lalu buat **Case Structure** ketiga yang diletakkan disebelah kanan bawah **Case Structure** pertama. Hubungkan inputnya dengan output bawah **Case Structure** pertama.
    Karena output dari **Case Structure** pertama memiliki tiga alternatif nilai, maka buat agar setiap **Case Structure** memiliki tiga Case, yaitu Case 0, Case 1 dan Case 2. Gunakan perintah klik kanan pada Case » **Add Case** untuk menambah Case.

14. Posisikan **Case Structure** kedua dan ketiga serta buat isi dari Case 0 untuk masing-masing **Case Structure** seperti gambar 4 – 12.

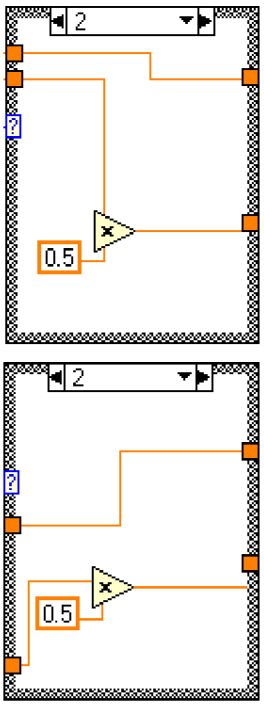



**Gambar 4 – 12**.

15. Buat isi **Case 1** seperti gambar 4 – 13.
16. Buat isi **Case 2** seperti gambar 4 – 14.



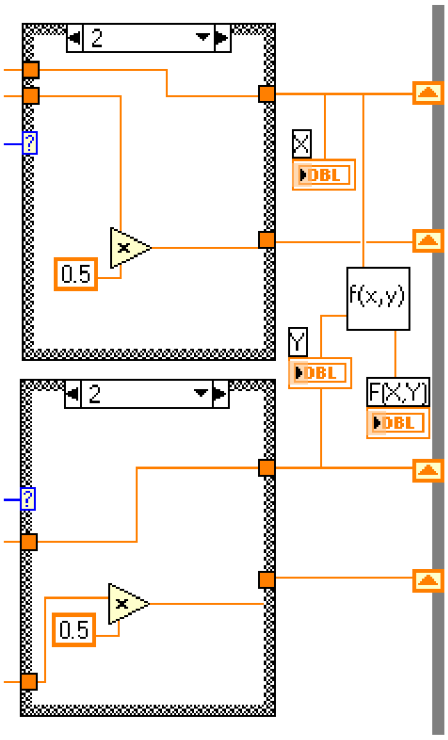

**Gambar 4 – 13.**




**Gambar 4 – 14.**

Langkah selanjutnya adalah menampilkan nilai X, Y dan F(X,Y) hasil dari iterasi.

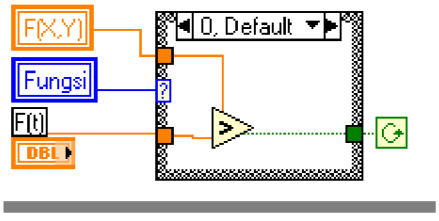
17. Posisikan **Digital Indicator** "X", "Y" dan "F(X,Y)" seperti gambar 4 – 15. Tambahkan SubVI "Fungsi.vi" dan *wiring* yang diperlukan.



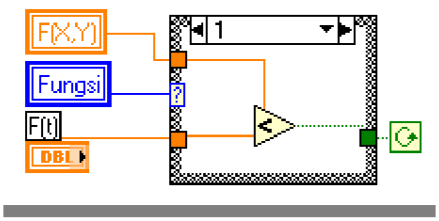

**Gambar 4.15.**

Langkah berikutnya adalah membuat kondisi dimana looping program boleh dihentikan yaitu saat f(x) < f(t).

18. Pilih **Structure » Local Variable**. Buat 2 buah **Local Variable**, letakkan dibagian bawah **While Loop**. Ubah modenya menjadi **Read Local** (dengan cara klik kanan pada *Local Variable* lalu pilih **Change to Read**). Klik kanan, kemudian **Select Item** masing-masing menjadi *"F(X,Y)"* dan *"Fungsi"*.
19. Posisikan **Digital Control** "*F(t)*" dibagian bawah **Local Variable** tersebut.
20. Buat **Case Structure** disebelah kanan **Local Variable**. Hubungkan inputnya pada **Local Variable** "*Fungsi*".
21. Buat isi Case 0 menjadi seperti gambar 4.16 dan case 1 menjadi seperti gambar 4.17. Hubungkan output dari **Case Sequence** pada **Conditional Terminal** dari **While Loop**.

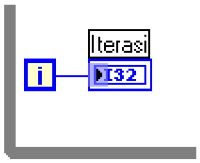



**Gambar 4.16**



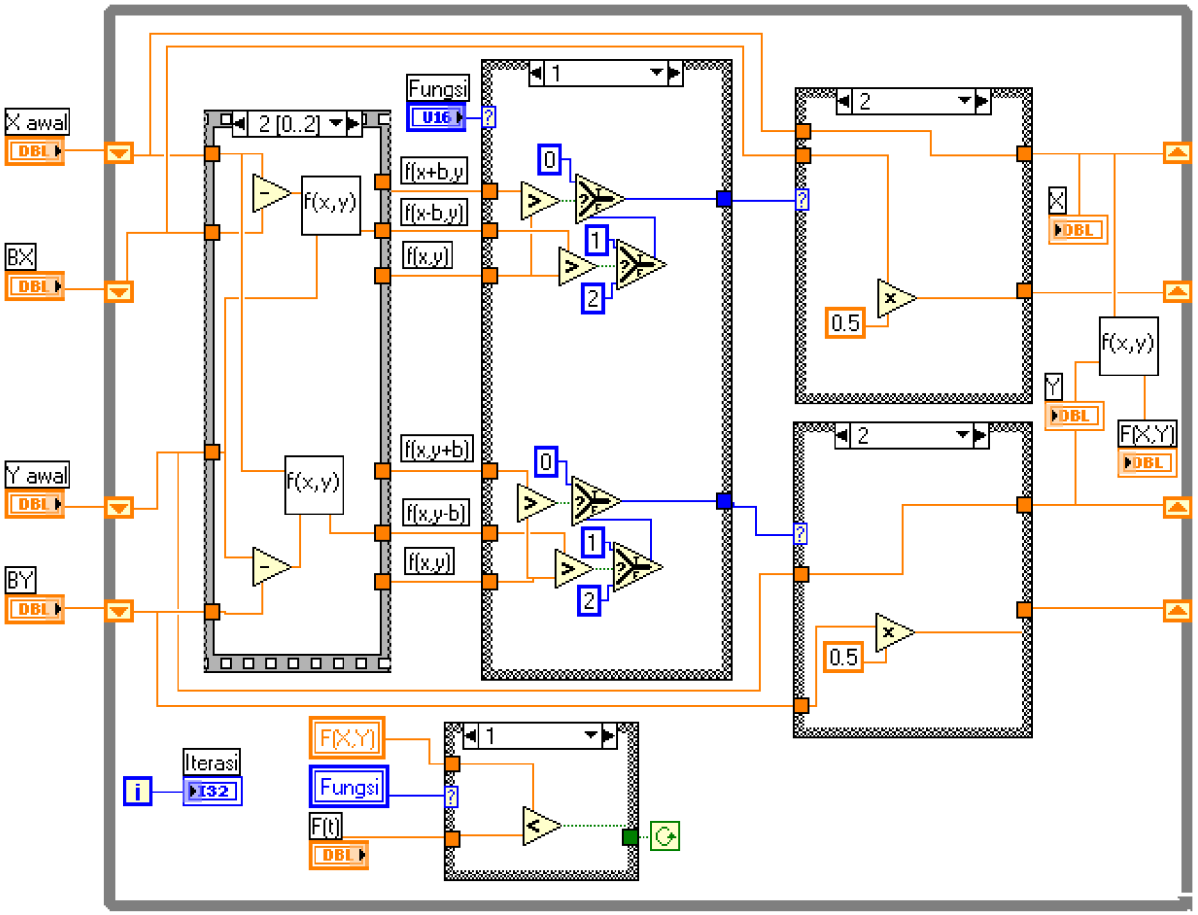

**Gambar 4.17**

22. Untuk mengetahui jumlah iterasi yang dilakukan maka hubungkan **Digital Indicator** "*Iterasi*" pada kotak iterasi seperti gambar 4.18.




**Gambar 4.18**

23. Sesuaikan **Block Diagram** akhirnya dengan gambar 4.19




**Gambar 4.19. Block Diagram** dari latihan 4.2.

24. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini

## *Pengujian Program*

Untuk menguji apakah program telah mampu melakukan optimasi mencari nilai minimum, lakukan langkah berikut :

1. Set **Menu Ring** "*Fungsi*" : Min dan set harga F(t) = -0.5. Isikan harga "X awal"; "Y awal", "BX" dan "BY" dengan harga tertentu (bebas)
2. Jalankan program.
   Jika program dapat berhenti dan menghasilkan nilai "F(X,Y)" kurang dari –0.5 berarti program telah mampu menjalankan optimasi.

Untuk mencari nilai maksimum dari fungsi, set **Menu Ring** "*Fungsi*" menjadi Max dan ubah harga F(t) menjadi 0.5

# 5

# Array dan Cluster

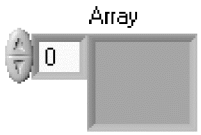
## Array

**Array** adalah kumpulan dari elemen data yang mempunyai tipe data yang sama. Sebuah **array** yang mempunyai satu atau lebih dimensi dapat mempunyai $2^{31} - 1$ elemen per dimensi. **Array** dapat diakses melalui indexnya. **Index Array** berada pada *range* 0 sampai n-1 dimana n adalah jumlah elemen data pada setiap dimensi.
Perlu dicatat bahwa **Index array** dimulai dari 0, jadi elemen pertama mempunyai index 0, elemen kedua index 1 dan seterusnya.

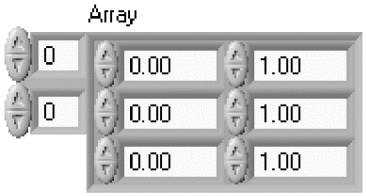
### *Cara Membuat Array*

Ada beberapa cara untuk membuat array, yaitu :
- Membuat Array melalui **Front Panel**
- Membuat Array dengan **Initialize Array**
- Membuat Array dengan **Build Array**

**Contoh Membuat Array Melalui Front Panel**




1. Buat sebuah VI baru
2. Aktifkan **Front Panel**. Pilih **Array & Cluster » Array**
3. Pilih **Numeric » Digital Control** tempatkan didalam array yang sudah dibuat
4. Aktifkan **Resize Tool**, tambah di mensi array dengan mendrag kotak kecil dikiri *array* ke arah bawah. (Langkah ini mengubah ukuran dimensi *array*, yang semula satu dimensi menjadi dua dimensi)
5. Ubah ukuran *array* dengan mendrag *array* ke arah bawah dan kanan sehingga *array* memiliki 2 kolom dan 3 baris.
6. Aktifkan **Edit Text Tool**, isikan angka 1 pada kolom ke-1 *array*.
7. Anda akan memperoleh *array* seperti gambar berikut.

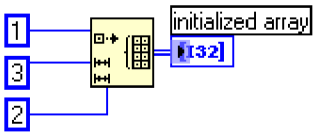



**Gambar 5.1.** Array dengan 2 kolom dan 3 baris

Anda telah dapat membuat *array* dua dimensi.

**Contoh Membuat Array Dengan Initialize Array**

1. Buat sebuah VI baru
2. Aktifkan **Block Diagram**. Pilih **Array » Initialize Array**
3. Aktifkan **Resize Tool**. Drag **Initialize Array** ke bawah
4. Aktifkan **Wiring Tool**. Arahkan cursor pada bagian terminal kanan **Initialize Array**, klik kanan kemudian **"Create Indicator"**.
5. Buat **Constant** 3 buah, hubungkan dengan 3 input **Initialize Array**.
6. Ubah besar **Constant** seperti gambar di bawah



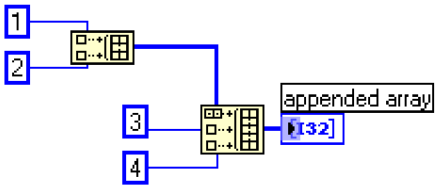

**Gambar 5.2.** Initialize Array

**Initialize Array** adalah salah satu cara untuk membuat *array* dengan besar masing-masing elemen adalah sama.
Jika program diatas dijalankan, Anda akan memperoleh pada **Front Panel** sebuah *array* yang terdiri dari 2 kolom dan 3 baris dengan nilai setiap *sub-array* = 1.

**Contoh Me mbuat Array Dengan Build Array**



1. Buat sebuah VI baru.
2. Aktifkan **Block Diagram**. Pilih **Array » Build Array**
3. Buat 2 buah **Build Array**. Resize **Build Array** seperti gambar di bawah.
4. Aktifkan **Wiring Tool**. Arahkan cursor pada bagian terminal kanan **Build Array**, klik kanan kemudian **"Create Indicator"**.
5. Buah 4 buah **Constant**, buat seperti gambar di bawah.




**Gambar 5.3.** Build Array

Jika program dijalankan, Anda akan memperoleh pada **Front Panel** sebuah *array* yang terdiri dari 1 kolom dan 4 baris dengan nilai datanya adalah 1, 2, 3, 4. Fungsi **Build Array** ini adalah untuk menggabungkan data-data menjadi sebuah *array*. Jika terdapat data baru yang ditambahkan, maka **Build Array** akan menambahkan data tersebut pada *array* yang telah ada sebelumnya.

## *Mengakses Elemen Array*

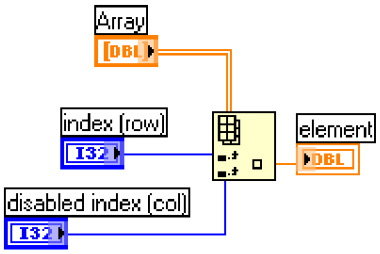
**Contoh Cara Mengakses Elemen Array**

1. Buat sebuah VI baru
2. Aktifkan **Front Panel**. Buat *array* seperti gambar berikut.


**Gambar 5.4.** Array

3. Buat dua buah **Control** dengan nama "*index (row)*" dan "*disabled index (col)*"
4. Aktifkan **Block Diagram**. Pilih **Array » Index Array**

5. Wiring output *Array* pada terminal kiri atas **Index Array**. Secara otomatis ukuran indeks ak an berubah seperti gambar disamping
6. Buat seperti gambar berikut.




**Gambar 5.5.** Cara mengakses element array

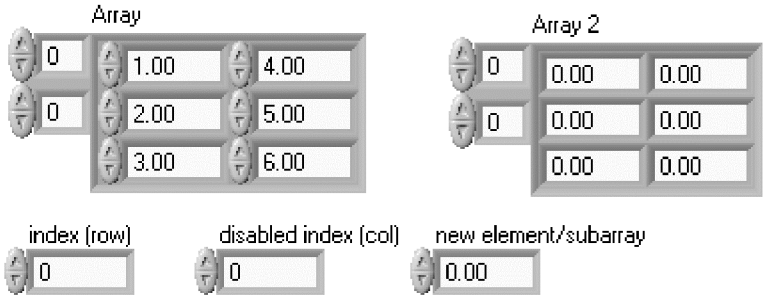
7. Klik *Run*. Ubah-ubah besar **Control** kemudian klik *Run* lagi.

Anda akan mendapatkan nilai element yang berbeda sesuai harga control yang diberikan. **Ingat bahwa baris ke-1 berarti indes ke-0**.

## *Mengganti Elemen Array*

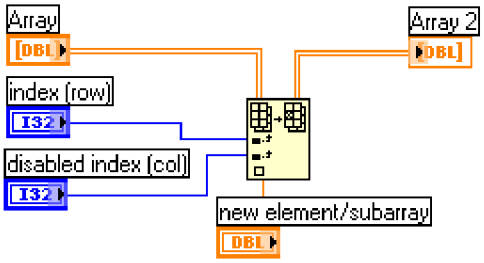
**Contoh Cara Mengganti Elemen Array**

1. Buat sebuah VI baru
2. Aktifkan **Front Panel**. Buat 2 buah *array* dan 3 **control** lalu edit label-nya dan posisikan seperti gambar berikut. Satu buah *array* berisi **Digital Controls** dan *array* lainnya berisi **Digital Indicators**.




**Gambar 5.6.**

3. Aktifkan **Block Diagram**. Pilih **Array » Replace Array Element**
4. Buat seperti diagram berikut ini :




**Gambar 5.7.** Replace Array Element

5. Klik *Run*, ubah-ubah besar element baru, baris dan kolom. Klik *Run* lagi.

Anda akan memperoleh nilai Array 2 akan sama dengan Array pertama tetapi berbeda pada satu sub-array tergantung nilai indeks dari **Digital Control** yang diberikan.

# Cluster

Seperti *array*, **Cluster** juga merupakan kumpulan dari elemen data. Perbedaannya adalah **cluster** bisa terdiri dari kumpulan elemen data dengan tipe data yang berbeda.

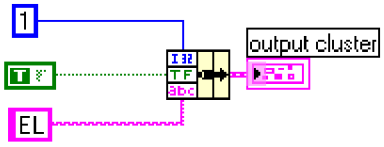
Bila **cluster** terdiri dari kumpulan elemen data yang mempunyai tipe yang sama, maka **cluster** dapat diubah ke *array* dan sebaliknya.

Keistimewaan dari **cluster** adalah kita dapat mengakses elemen data pada cluster dengan fungsi **Unbundle** sementara pada *array* harus diakses elemen data berdasarkan indexnya.

Fungsi yang dipakai untuk mengubah *cluster* ke *array* dan sebaliknya :
- **Cluster » Array to Cluster**
- **Cluster » Cluster to Array**

## *Cara Membuat Cluster*

**Membuat Cluster dengan Constant**
1. Buat sebuah VI baru
2. Aktifkan **Block Diagram**
3. Pilih **Numeric » Numeric Constant**
4. Pilih **Boolean » Boolean Constant**
5. Pilih **String » String Constant**
6. Pilih **Cluster » Bundle**, kemudian resize menjadi 3 input.
7. Buat seperti gambar berikut. Kemudian klik *Run*.



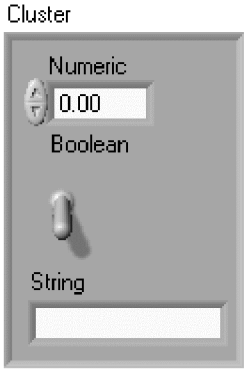

**Gambar 5.8.** Membuat Cluster

Anda telah membuat **cluster** dari kumpulan **constant** dengan tipe data yang berbeda. Anda pun bisa membuatnya dengan memakai input **Control**.

## *Memisahkan Elemen Data pada Cluster*

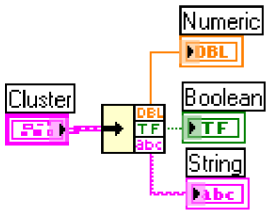

**Memisahkan Elemen Data pada Cluster**

1. Buat sebuah VI baru
2. Aktifkan **Front Panel**
3. Pilih **Array & Cluster » Cluster**
4. Pilih **Numeric » Digital Control**, tempatkan dalam *cluster*
5. Pilih **Boolean » Vertical Toggle Switch**, tempatkan dalam *cluster*
6. Pilih **String » String Control**, tempatkan dalam *cluster*
7. Anda akan memperoleh **Front Panel** seperti gambar berikut.




**Gambar 5.9.** Cluster

8. Aktifkan **Block Diagram**
9. Pilih **Cluster » Unbundle**
10. Buat seperti gambar berikut, kemudian klik *Run*.




**Gambar 5.10.** Unbundle Cluster

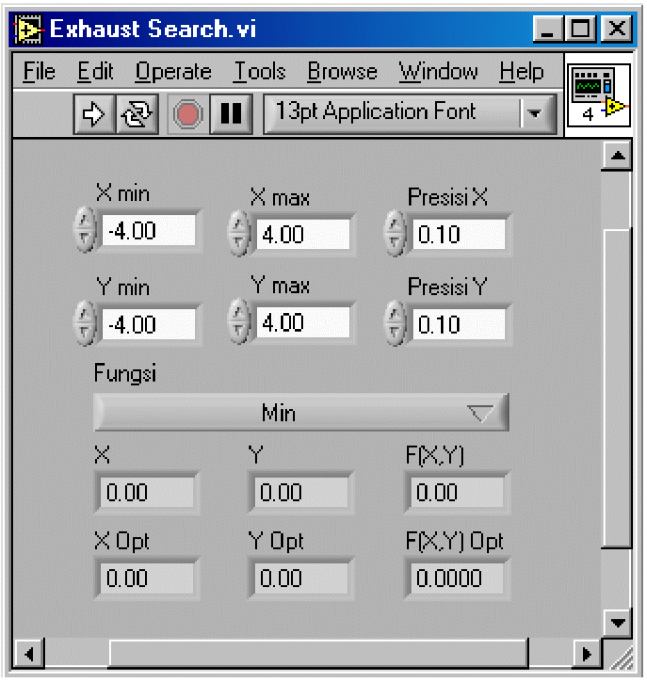
Anda juga bisa memisahkan element data pada **cluster** dengan fungsi **Unbundle by Name**. Fungsi ini akan memisahkan elemen data berdasarkan label **Control** yang telah Anda buat di **Front Panel**.

# Latihan 5.1 : Membuat Program Exhausted Search

Pada program berikut ini Anda akan membuat program VI untuk melakukan optimasi persamaan (2.1) dengan metode Exhausted Search. Metode ini akan memprogram LabVIEW untuk menghitung setiap harga persamaan dalam range yang diberikan. Pada saat yang bersamaan, programpun harus mencari nilai optimal yang telah dihasilkan dari hasil perhitungan.

## *Membangun Display pada Front Panel*

1. Buat VI baru. Save dengan nama "*Exhausted Search*"
2. Aktifkan **Front Panel**. Dan buat menjadi seperti gambar 5 – 11. Edit dan atur nilai dari setiap **Digital Control** menjadi seperti pada gambar.




**Gambar 5 – 11. Front Panel** dari program Latihan 5.1.

Keterangan **Digital Control :**

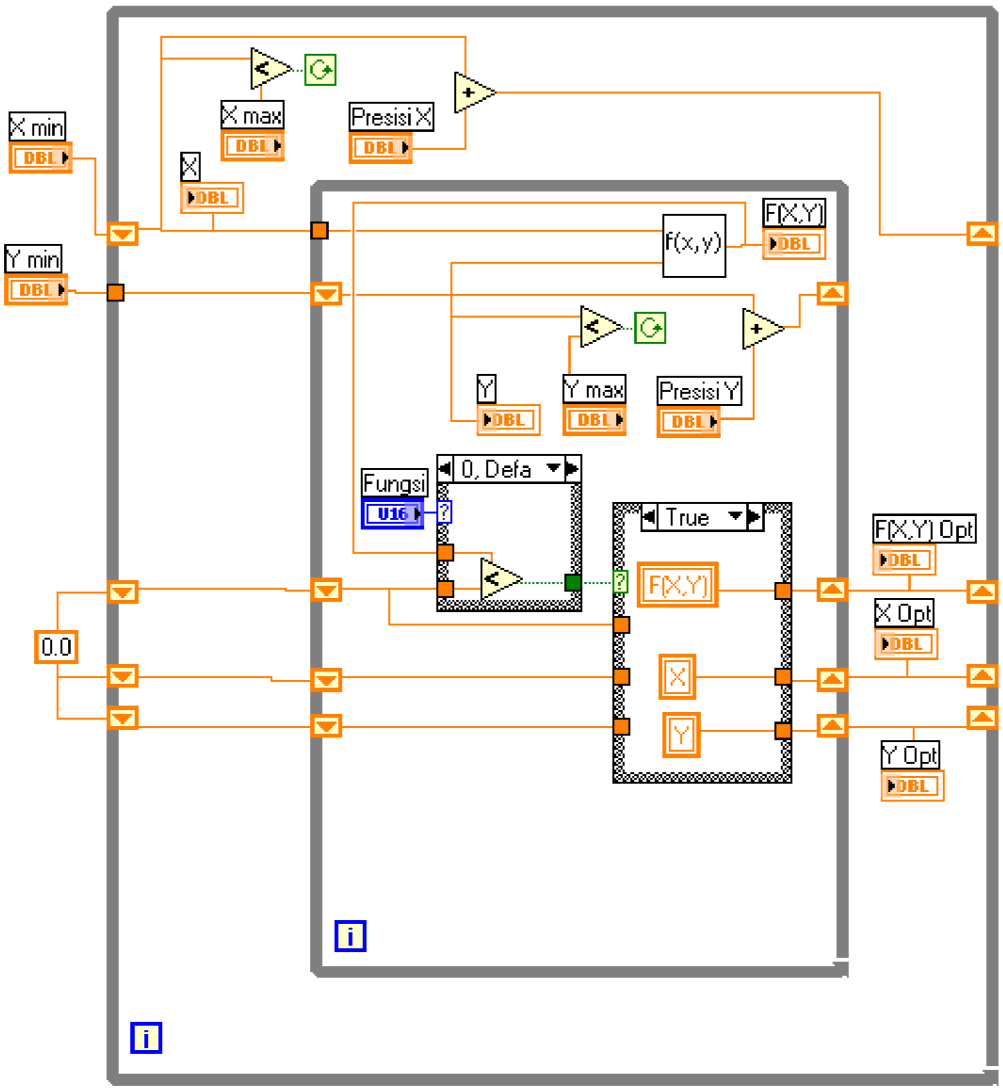
- "*X min*" dan "*X max*" adalah batas pencarian pada sumbu X dengan selang sesuai nilai "*Presisi X*"
- "*Y min*" dan "*Y max*" adalah batas pencarian pada sumbu Y dengan selang sesuai nilai "*Presisi Y*"
- "Fungsi" adalah jenis pencarian optimasi yang dilakukan, apakah Min ataukah Max.

Keterangan **Digital Indicator :**

- "*X*", "*Y*" dan "*F(X,Y)*" adalah harga X, Y dan F(X,Y) pada setiap kali iterasi
- "*F(X,Y)*" adalah harga F(X,Y) paling optimal yang telah diketemukan dari hasil perhitungan
- "*X Opt*", "*Y Opt*" adalah harga X dan Y yang membuat F(X,Y) optimal.

## *Membangun VI Diagram*

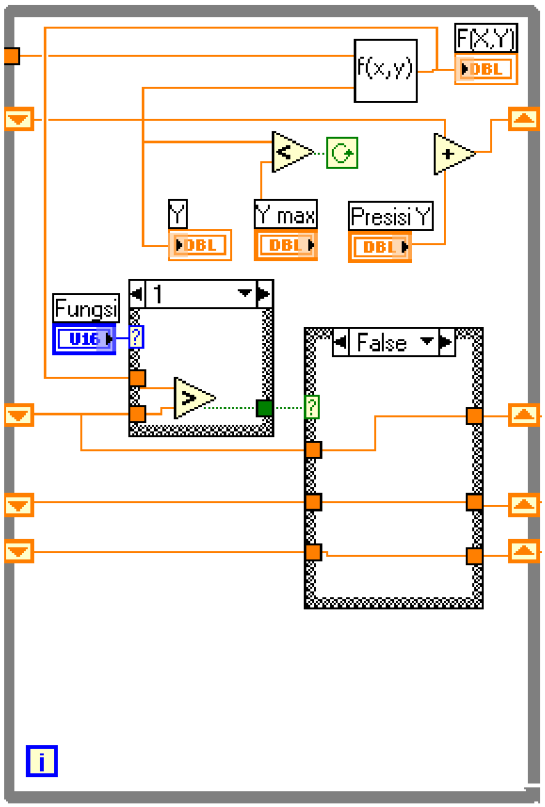
1. Aktifkan **Block Diagram**
2. Buat **Block Diagram** menjadi seperti gambar 5 – 12. Perhatikan bahwa terdapat satu buah subVI yang digunakan yaitu "*Fungsi.vi*
   Terdapat tiga buah **Local Variable** yang digunakan dalam mode **Read**. Masing-masing "*X*", "*Y*" dan "*F(X,Y)*".
   Terdapat dua buah **While Loop** yang digunakan. **While Loop** bagian dalam melakukan iterasi nilai Y dari "*Y min*" hingga "*Y max*" untuk harga X tetap. Sedangkan **While Loop** bagian luar melakukan iterasi nilai X dari "*X min*" hingga "*X max*"
   **Conditional Terminal** pada **While Loop** akan berharga *false* (program berhenti) jika nilai "*X max*" dan "*Y max*" telah dicapai




**Gambar 5 – 12. Block Diagram** Latihan 5.1.

Terdapat dua buah **Case Structure** yang digunakan. **Case Structure** dengan input numeric memiliki dua case yang berfungsi untuk men-set apakah program akan mencari nilai minimum atau maksimum.

**Case Structure** dengan input Boolean berfungsi untuk mencari dan mencatat nilai F(X,Y), X dan Y yang optimal yang telah ditemukan dari hasil perhitungan.
Isi dari Case pertama telah terlihat dari gambar 5 – 12. Isi Case kedua dapat dilihat dari gambar 5 – 13.




**Gambar 5 – 13**. Case Structure kedua

## *Pengujian Program*

Ketik tombol *Run*. Jika program mampu mencapai harga F(X,Y) = 0,5 untuk pencarian nilai maksimum dan harga F(X,Y) = -0,5 untuk pencarian nilai minimum berarti algoritma program telah benar.

## *Menambah Fasilitas Menampilkan Grafik 3 Dimensi dari Fungsi*

Langkah-langkah berikut adalah cara untuk menampilkan grafik dari persamaan (2.1) sehingga dapat diketahui bentuk dari persamaan tersebut.

1. Aktifkan **Front Panel**
2. Dari **Controls Palette**, pilih **Graph » 3D Surface Graph**. Letakkan disebelah kanan terminal "*Presisi X*". Ubah ukurannya agar tampilannya menjadi lebih jelas.
3. Aktifkan **Block Diagram**. Posisikan diagram **3D Surface Graph** disebelah kanan atas luar **While Loop**.

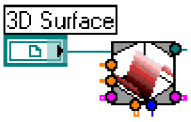
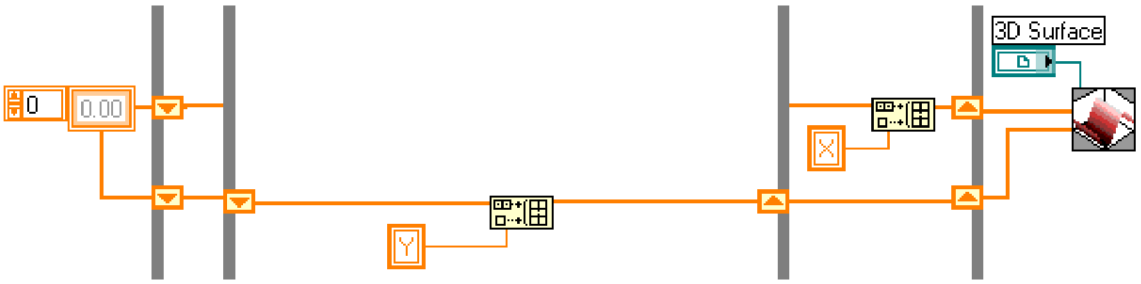



Diagram **3D Surface Graph** membutuhkan 3 input yang akan digunakan pada latihan ini yaitu "*x vector*", "*y vector*" dan "*z matrix*"
"*x vector*" yaitu *array* 1D berisi urutan nilai X min hingga X max pada selang tertentu.
"*y vector*" yaitu *array* 1D berisi urutan nilai Y min hingga Y max pada selang tertentu.
"*z matrix*" yaitu *array* 2D berisi nilai fungsi untuk setiap harga X dan Y yang ada.
Langkah-langkah berikut bertujuan untuk membuat *array-array* yang diperlukan tersebut.

**Membuat "*x vector*" dan "*y vector*"**

4. Pada sisi kiri bawah setiap **While Loop** buatlah **Shift Register**.
5. Pada **Functions Pallete** pilih **Array » Array Constant**. Letakkan disebelah kiri **Shift Register**.
6. Buat sebuah **Constant** bernilai 0.0. Letakkan didalam **Array Constant**.
7. Selanjutnya lakukan langkah-langkah yang diperlukan agar diperoleh diagram seperti gambar 5 – 14.
   Sambungkan output **Shift Register** atas dengan terminal "*x vector*" pada **3D Surface Graph**.
   Sambungkan output **Shift Register** bawah dengan terminal "y *vector*" pada **3D Surface Graph**.

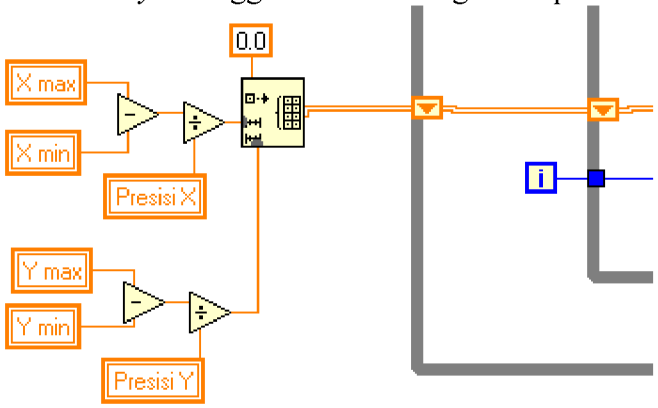



**Gambar 5 – 14**. Membuat "*x vector*" dan "*y vector*"

Perhatikan bahwa wiring *array* data X tidak melalui **While Loop** bagian dalam.

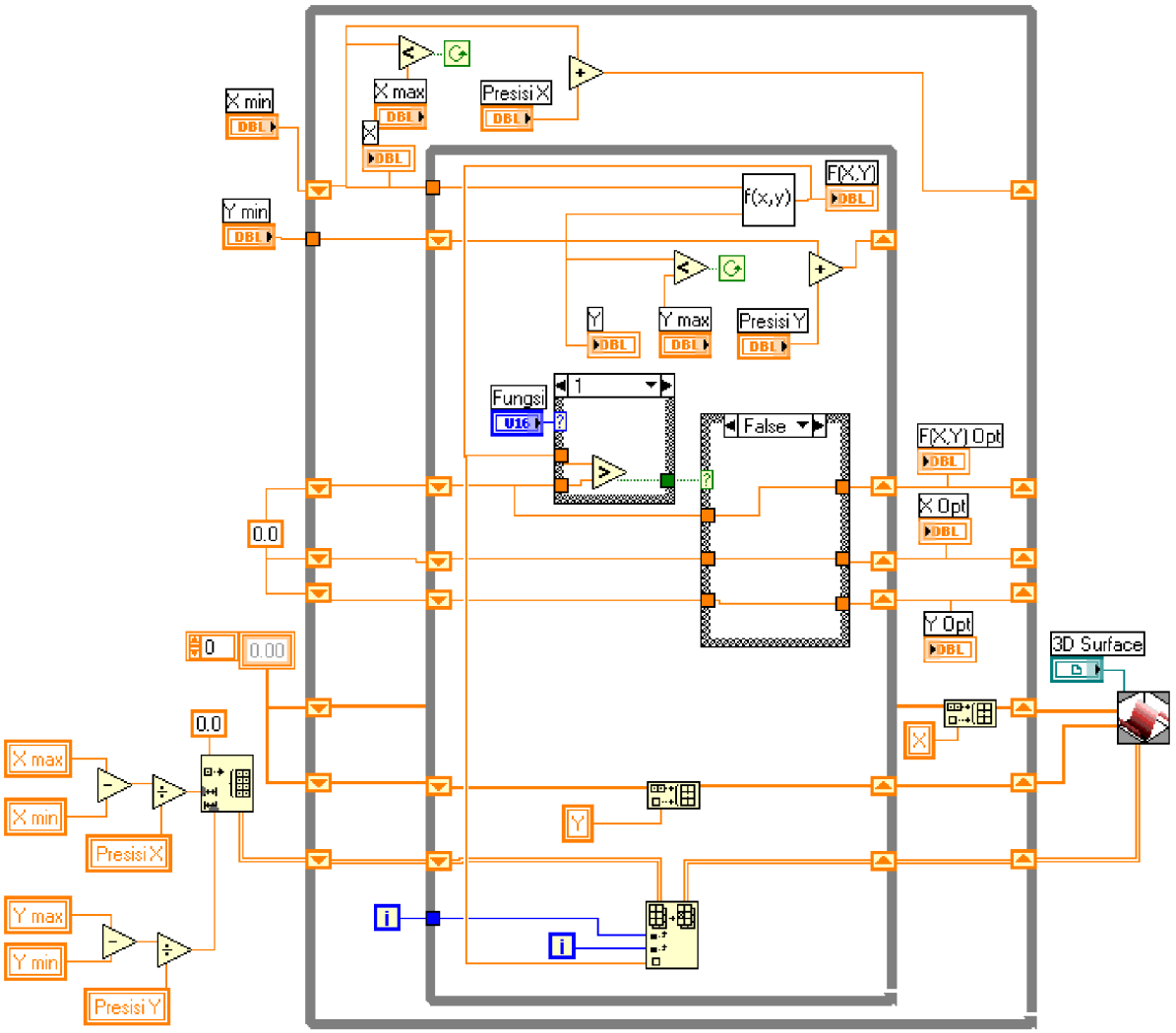
**Membuat "*z matrix*"**

8. Pada bagian kiri bawah **While Loop** buat **Shift Register**. Kemudian lakukan inisialisasi *array* sehingga terbentuk diagram seperti berikut.




**Gambar 5 – 15.** Inisialisasi Array "*z matrix*"

9. Kemudian *array* awal tersebut harus diganti dengan nilai F(X,Y) untuk setiap kali iterasinya. Gunakan **Replace Array Element** untuk membentuk diagram sebagai berikut.




**Gambar 5.16. Block Diagram** Exhausted Search

10. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini

## *Pengujian Program*

Jalankan program. Jika grafik 3D dapat muncul di **Front Panel** berarti algoritma program telah benar

# 6

# File Input - Output

Dalam membuat suatu VI yang melibatkan banyak data, diperlukan suatu metoda untuk menganalisa data yang telah dihasilkan. Biasanya dipakai suatu file *spreadsheet* misalnya *.xls yang dijalankan dengan **Microsoft Excel**.

**LabVIEW** telah menyediakan fitur untuk menuliskan data ataupun membaca file *spreadsheet*.

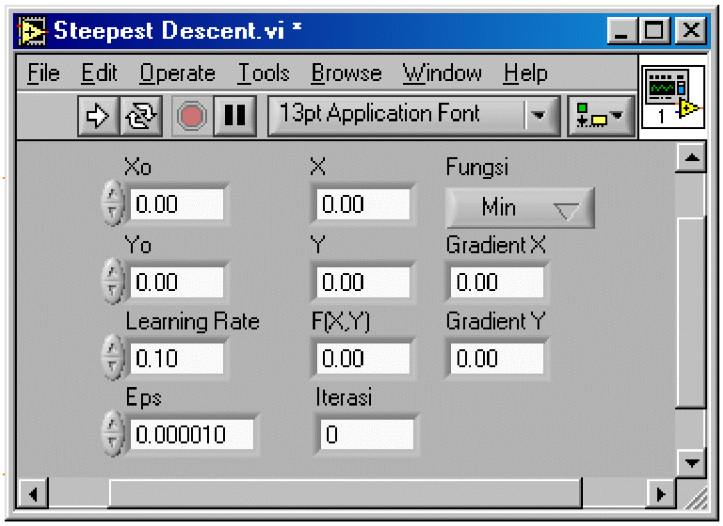
Jika ingin menyimpan lebih dari satu kelompok data sekaligus, maka kelompok-kelompok data tersebut perlu disatukan terlebuh dahulu menggunakan fungsi **Build Array**.

## Latihan 6.1 : Membuat Program Steppest Descent

Pada program berikut ini Anda akan membuat program VI optimasi dengan metode Steppest Descent untuk mengoptimasi persamaan (2.1)

### *Membangun Display pada Front Panel*

3. Buat VI baru. Save dengan nama "*Steepest Descent*"
4. Aktifkan **Front Panel**. Dan buat menjadi seperti gambar 6 – 1. Edit dan atur nilai dari setiap **Digital Control** menjadi seperti pada gambar.




**Gambar 6 – 1. Front Panel** dari Latihan 6.1.

Keterangan **Digital Control :**

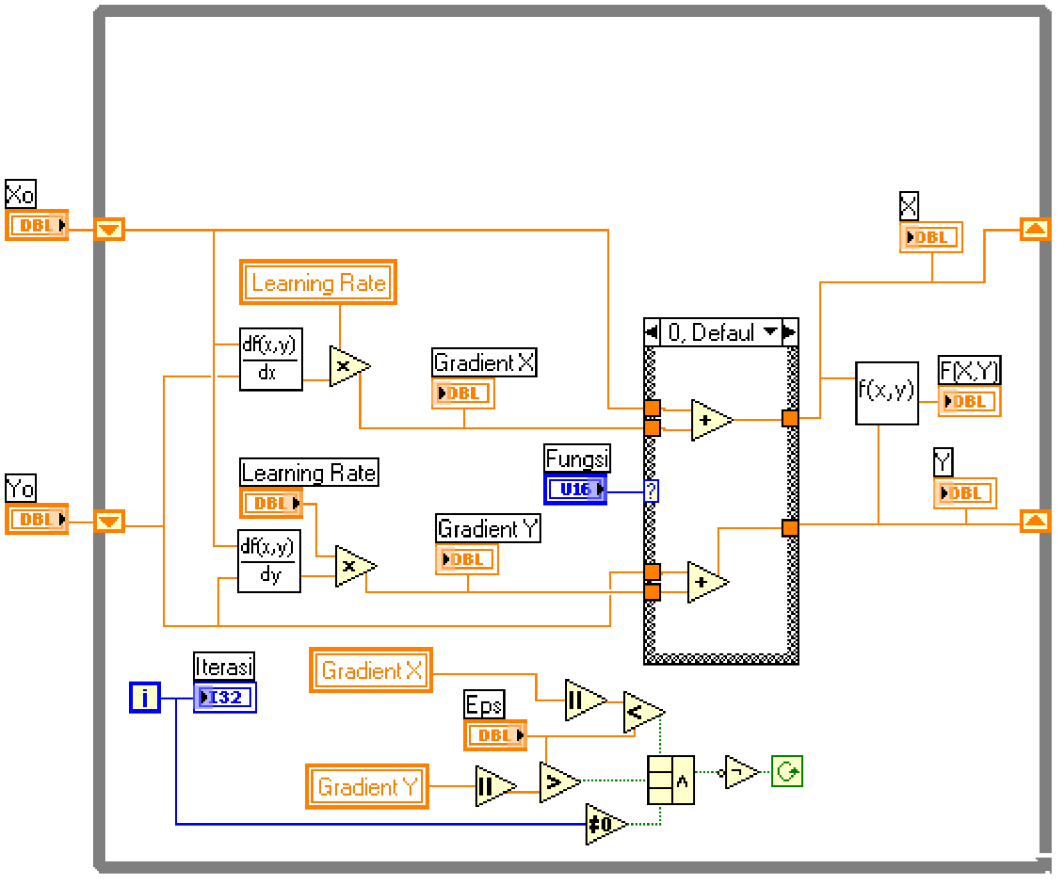
- "Xo" adalah harga X awal (nilainya bebas)
- "Yo" adalah harga Y awal (nilainya bebas)
- "Learning Rate" adalah harga η yang menentukan lebar langkah pencarian. Semakin kecil, maka semakin besar kemungkinan diketemukannya nilai optimal, tetapi semakin lambat proses pencariannya.
- "Eps" adalah nilai maksimum gradien X dan gradien Y yang diizinkan. Jika harga gradien minimal ini telah dicapai, maka program akan memberhentikan proses pencariannya.
- "Fungsi" adalah jenis pencarian optimasi yang dilakukan, apakah Min ataukah Max.

Keterangan **Digital Indicator :**

- "X" adalah harga X hasil pencarian yang membuat fungsi optimal
- "Y" adalah harga Y hasil pencarian yang membuat fungsi optimal
- "(X,Y)" adalah harga fungsi saat nilai optimal
- "Iterasi" adalah jumlah iterasi yang telah dilakukan
- "Gradien X" adalah kemiringan kurva oleh variabel X yang telah dicapai
- "Gradien Y" adalah kemiringan kurva oleh variabel Y yang telah dicapai

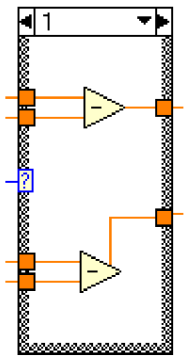
## *Membangun VI Diagram*

3. Aktifkan **Block Diagram**
4. Buat **Block Diagram** menjadi seperti gambar 6 – 2. Perhatikan bahwa terdapat tiga buah subVI yang digunakan yaitu "*Fungsi.vi*", "*Diff Fungsi terhadap X.vi*" dan "*Diff Fungsi terhadap Y.vi*"
   Terdapat tiga buah **Local Variable** yang digunakan dalam mode **Read**. Masing-masing "*Learning Rate*", "*Gradien X*" dan "*Gradien Y*".
   **Conditional Terminal** pada **While Loop** akan berharga *false* (program berhenti) jika nilai mutlak dari gradient X dan gradient Y telah melampaui batas Eps.




**Gambar 6 – 2**. **Block Diagram** dari Latihan 6 – 1.

5. **Case Structure** yang digunakan terdiri dari dua case. Case 0 terlihat seperti pada gambar 6 – 2. Case 1 dibuat seperti gambar 6 – 3.




**Gambar 6 – 3.**

**Case Structure** disini berisi perintah yang mengatur apakah program ini akan mencari nilai maksimum atau nilai minimum.

6. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini

## *Pengujian Program*

Ketik tombol *Run*. Jika program mampu mencapai harga F(X,Y) = 0,5 untuk pencarian nilai maksimum dan harga F(X,Y) = -0,5 untuk pencarian nilai minimum berarti algoritma program telah benar.

## *Menambah Fasilitas Menyimpan Data Iterasi*

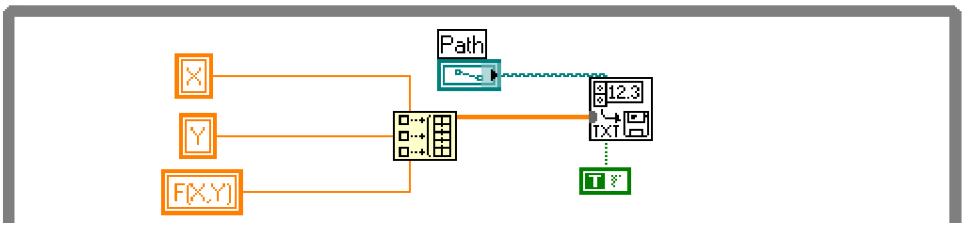
Langkah-langkah berikut adalah cara untuk menyimpan setiap data iterasi dari X, Y dan F(X,Y) untuk setiap langkah iterasi ke dalam file *spreadsheet*.


1. Aktifkan **Front Panel**. Pada Control Palette, pilih **String & Path » File Path Control**. Letakkan dibawah **Digital Indicator** "*Gradient Y*".

2. Aktifkan **Block Diagram.** Dari **Functions Palette**, pilih **File I/O » Write To Spreadsheet File.vi**. Tempatkan didalam **While Loop** disebelah atas.
3. Posisikan **Path** disebelah kiri **Write To Spreadsheet File.vi**. lalu hubungkan dengan bagian kiri atas dari **Write To Spreadsheet File.vi** (bagian "file path").

4. Dari **Functions Palette**, pilih **Boolean » True Constant** lalu posisikan dibagian bawah **Path**. Hubungkan **True Constant** dengan bagian kiri bawah dari **Write To Spreadsheet File.vi** (bagian "append to file?")

5. Dari **Functions Palette**, pilih **Array » Build Array**, resize menjadi 3 input elemen, tempatkan disebelah kiri **Write To Spreadsheet File.vi**.
6. Pilih **Structure » Local Variable**. Buat 3 buah **Local Variable**, letakkan disebelah kiri **Build Array**. Ubah modenya menjadi **Read Local**. Klik kanan, kemudian **Select Item** masing-masing menjadi *"X"*, *"Y"*, *"F(X,Y)"*.
7. Hubungkan output **Build Array** dengan terminal "1D data" disebelah kiri **Write To Spreadsheet File.vi**
8. Lakukan wiring sehingga **Block diagram** akhirnya akan menjadi seperti gambar 6 – 4.




**Gambar 6 – 4**. Diagram penyimpanan data

9. Pilih **File « Save** pada **Menu Bar** untuk menyimpan VI ini

## *Pengujian Program*

1. Masukkan nama File pada **File Path Control** dalam bentuk "Nama file".xls
2. Jalankan program. Data iterasi dari "X", "Y" dan "F(X,Y)" akan disimpan pada file dengan nama yang ada pada **File Path Control**.
3. Untuk melihat data yang disimpan, maka file *spreadsheet* tersebut dapat dibuka pada program Microsoft Excel.

# 7

# Tugas Metode Optimasi

Persyaratan pengerjakan tugas.

1. Tugas dikerjakan per-mahasiswa
2. Tugas dikumpulkan sebelum UTS mata kuliah Metode Optimasi dilaksanakan
3. Konsultasi dengan dosen dapat dilaksanakan antara tanggal 13 s.d. 19 November 2005

**Tugas** :

- Buat suatu program optimasi menggunakan LabVIEW untuk mencari nilai optimum dari salah satu fungsi berikut ini.
- Alternatif metode yang digunakan :
  i. Exhausted Search
  ii. Random Search Method
  iii. Steepest Descent
- Yang dikumpulkan adalah printing program LabVIEW dan hasilnya

Pilihan fungsi :

| | |
|---|---|
| 1. | $f(x,y) = x^2 - 4x + y^2 - y - xy$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 2. | $f(x,y) = 100\left(y - x^2\right)^2 + (1-x)^2$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 3. | $f(x,y) = 5x^2 + x^4 - 9x^2y + 3y^2 + 2y^4 + \frac{1}{4}x$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 4. | $f(x,y) = 0{,}55\left(2x^2 + y^2\right) + 0{,}09y + 0{,}0055 - 0{,}9xy - 0{,}11x$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 5. | $f(x,y) = e^{x+3y-0{,}1} + e^{x-3y-0{,}1} + e^{-x-0{,}1}$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 6. | $f(x,y) = \cos\left(\frac{1}{2}x\right)\cos\left(\frac{1}{2}y\right)x$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |
| 7. | $f(x,y) = 1000(x-2y)^2 + (1-x)^2$ , untuk $-4 < x < 4$ dan $-4 < y < 4$ |