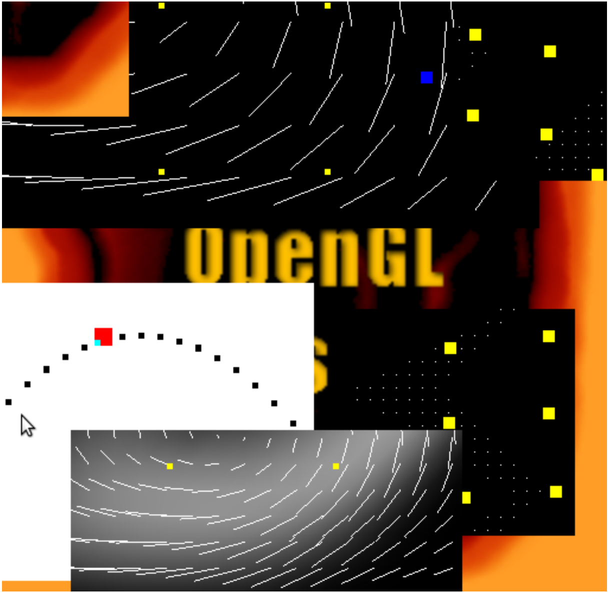
O Método Adjunto

Dalia Melissa Bonilla Correa
IMPA

11 de Novembro de 2009

Simulações

Simulações

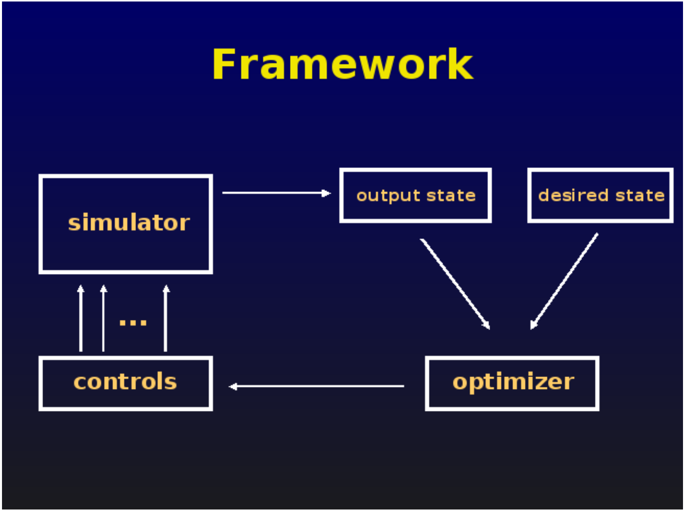


Simulações e Controle

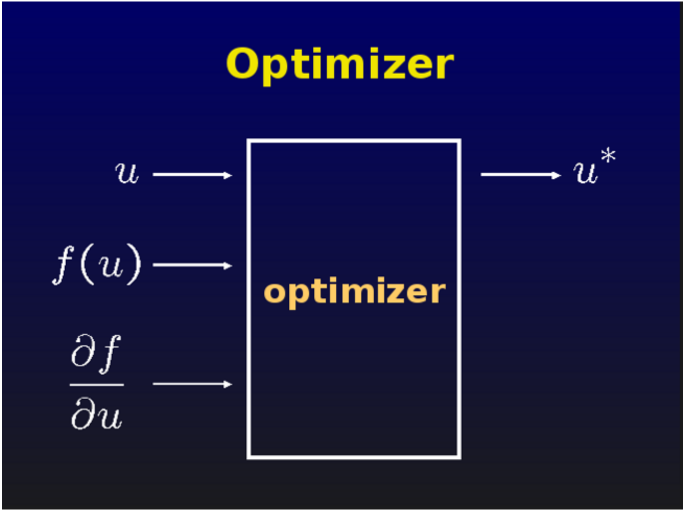
Uma simulação deve

- ser rápida (`Real time`)
- ter um código simples
- ser fácil de controlar

Simulação


Framework
simulator
output state
desired state
...
controls
optimizer

Otimização



Otimização

Gradient

$$\frac{\partial}{\partial u}\left(\boxed{\text{simulator}}\right) = ??$$

$$\frac{\partial}{\partial u}\left(a\,u^{3}\right) = 3\,a\,u^{2}$$

Modelos Adjuntos

Modelos adjuntos são ferramentas desenvolvidas para *modelagem reversa* de um sistema físico.

Modelos Adjuntos

- Sistema físico – modelo F
- Conjunto de observações $\mathfrak{D} \subset \mathbb{R}^m$, tal que $D \in \mathfrak{D}$
- O modelo calcula Y

$$J := \frac{1}{2}(Y - D, Y - D) \text{ producto interno } (\ .\ ,\ .\ )$$

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

$$J : \mathbb{R}^n \rightarrow \mathbb{R}$$
$$X \rightarrow \frac{1}{2}(F(X) - D, F(X) - D)$$

Modelos Adjuntos

- Por serie de Taylor temos

$$J(X) = J(X_0) + (\nabla_x J(X_0)\,,\, X - X_0) + o(|X - X_0|)$$

- Escrevendo

$$\delta J = (\nabla_x J(X_0)\,,\, \delta X)$$

- A é o Jacobiano de F em X_0, então

$$\delta Y = A(X_0)\delta X \qquad\qquad \begin{aligned} \delta Y &= F(X) - F(X_0) \\ \delta X &= X - X_0 \end{aligned}$$

Métodos Adjuntos

- Derivando a funação objetivo temos

$$
\begin{aligned}
\delta J =& \frac{1}{2}(A(X_0)\delta X\,,\,F(X_0)-D) + \frac{1}{2}(F(X_0)-D\,,\,A(X_0)\delta X) \\
=& (F(X_0)-D\,,\,A(X_0)\delta X)
\end{aligned}
$$

- Usando a definição de operador adjunto

$$(v\,,\,Aw) = (A^*v\,,\,w)$$

Gradiente

$$\delta J = (A^*(X_0)\,(F(X_0)-D)\,,\,\delta X)$$

$$\nabla_x J(X_0) = A^*(X_0)\,(F(X_0)-D)$$

Modelo Adjunto

- $A(X_0)$ representa o modelo *tangente linear*
- O operador $A^*(X_0)$ representa o *modelo adjunto*
- Calculo de $\nabla_x J(X_0)$ por diferencias finitas—$n+1$ calculos da função objetivo.
- Usando o modelo adjunto leva de $2-5$ calculos da função objetivo

Simulção Modelo e Código

- Modelo
  Equações diferencias
- Algoritmo
  Discretição
- Código
  Implementação

A construção do código do modelo adjunto depois dos 3 passos anteriores.

- Modelo
  Equações diferencias adjuntas
- Algoritmo
  Modelo Adjunto – eda
- Código
  Código adjunto

Código numérico como uma função

Um modelo numérico é um algoritmo que pode ser visto com a composição de funções diferenciais.
Cada função representa uma instrução no código numérico.

```
float x, y, z;
float u, v;
float f;

x = y + z + u*u;
y = z*y + x*v;
f = x*x + y*y + u*u + v*v;
```

Função H definida como um algoritmo numérico

$$H : \mathbb{R}^n \to \mathbb{R}^m$$
$$X \to Y$$

Cada passo do algoritmo é representado por

$$H^l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l} \qquad (l = 1, ..., K)$$
$$Z^{l-1} \to Z^l$$

$$H = H^K \circ \cdots \circ H^1 =: \bigodot_{l=1}^{K} H^l$$

$$Z^l = (H^l \circ \cdots \circ H^1)(X) = \bigodot_{i=1}^{l} H^i(X) \qquad (1 \leq l \leq K)$$

O jacobiano da função H é definido por

$$A_{ij}(X_0) := \left.\frac{\partial H_i(X)}{\partial X_j}\right|_{x=x_0} \qquad (i = 1, \ldots m\,;\, j = 1, \ldots, n)$$

$$A(X_0) = \left.\frac{\partial H^K}{\partial Z^{K-1}}\right|_{z^{k-1}=z_0^{k-1}} \cdots\cdots \left.\frac{\partial H^1}{\partial Z^0}\right|_{z^0=x_0}$$

$$Z_0^{k-1} = \bigodot_{i=1}^{l} H^i(X_0)$$

Modo para frente

Avaliamos

$$A(X_0) = \frac{\partial H^K}{\partial Z^{K-1}} \cdot \dots \cdot \frac{\partial H^2}{\partial Z^1} \cdot \frac{\partial H^1}{\partial Z^0}$$

no mesmo ordem em que a composição é avalida

$$H^l(\dots(H^2(H^1(X)))$$

Modo para frente

Avaliamos

$$A(X_0) = \frac{\partial H^K}{\partial Z^{K-1}} \cdot \cdots \cdot \frac{\partial H^2}{\partial Z^1} \cdot \frac{\partial H^1}{\partial Z^0}$$

no mesmo ordem em que a composição é avalida

$$H^l(\ldots(H^2(H^1(X)))$$

Multiplicamos primeiro

$$\frac{\partial H^2}{\partial Z^1} \cdot \frac{\partial H^1}{\partial Z^0}$$

e então

$$\frac{\partial H^3}{\partial Z^2} \quad \text{é multiplicado ao resultado}$$

Modo Reverso

Avaliamos

$$A(X_0) = \frac{\partial H^K}{\partial Z^{K-1}} \cdot \frac{\partial H^{K-1}}{\partial Z^{K-2}} \cdot \dots \cdot \frac{\partial H^2}{\partial Z^1} \cdot \frac{\partial H^1}{\partial Z^0}$$

Ao contrario, este modo começa de

$$\frac{\partial H^K}{\partial Z^{K-1}} \cdot \frac{\partial H^{K-1}}{\partial Z^{K-2}}$$

Função Escalar

$$H : \mathbb{R}^n \longrightarrow \mathbb{R}^m \qquad \text{onde} \quad m = 1$$
$$X \rightarrow Y$$

- O modo reverso é preferível!!
- Para $m = 1$ operar em modo reverso é chamado de **Método Adjunto**.
- O algoritmo para calcular o gradiente é chamado **Modelo Adjunto**.

Definições

$$H = \bigodot_{l=1}^{K} H^l$$

$$H^K : \mathbb{R}^{n_{K-1}} \to \mathbb{R} \qquad\qquad n_k = m = 1$$

$$Z_0^l = (H^l \circ \cdots \circ H^1)(X_0) = \bigodot_{i=1}^{l} H^i(X_0) \qquad (1 \leq l \leq K)$$

$$\delta Z^l = \left.\frac{\partial\left(\bigodot_{i=1}^{l} H^i(X_0)\right)}{\partial X}\right|_{x=x_0} \delta X \qquad \text{onde } \delta Z^0 := \delta X$$

$$\delta Z^l = \left.\frac{\partial H^l(Z^{l-1})}{\partial Z^{l-1}}\right|_{Z^{l-1}=Z_0^{l-1}} \delta Z^{l-1}$$

$$\delta^* Z^l := \nabla_{z^l} \left.\bigodot_{i=l+1}^{K} H^i(Z^l)\right|_{z^l=z_0^l}$$

$$\delta H = (\delta^* Z^l \, , \, \delta Z^l)$$

$$\begin{aligned}
(\delta^* Z^{l-1} \, , \, \delta Z^{l-1}) &= (\delta^* Z^l \, , \, \delta Z^l) \\
&= \left(\delta^* Z^l \, , \, \left(\frac{\partial H^l(Z^{l-1})}{\partial Z^{l-1}} \right) \Bigg|_{Z^{l-1} = Z_0^{l-1}} \delta Z^{l-1} \right) \\
&= \left(\left(\frac{\partial H^l(Z^{l-1})}{\partial Z^{l-1}} \right)^* \Bigg|_{Z^{l-1} = Z_0^{l-1}} \delta^* Z^l \, , \, \delta Z^{l-1} \right)
\end{aligned}$$

Resultados Intermediarios Adjuntos

$$\delta^* Z^{l-1} = \left(\frac{\partial H^l(Z^{l-1})}{\partial Z^{l-1}} \right)^* \Bigg|_{Z^{l-1}=Z_0^{l-1}} \delta^* Z^l$$

$$\delta^* Z_i^{l-1} = \sum_{j=1}^{n_l} \frac{\partial H_j^l(Z^{l-1})}{\partial Z_i^{l-1}} \Bigg|_{Z^{l-1}=Z_0^{l-1}} \delta^* Z_j^l$$

Problema

$$x = y + z + u^2;$$
$$y = zy + xv;$$
$$f = x^2 + y^2 + u^2 + v^2;$$

$$dx = dy + dz + 2udu;$$
$$dy = ydz + zdy + vdx + xdv;$$
$$f = 2xdx + 2ydy + 2udu + 2vdv;$$

```
float x, y, z;                    // variables
float u, v;                       // controls
float f;                          // cost variable
```

```
x = y + z + u*u;    // statement #1
```

```
y = z*y + x*v;      // statement #2
```

```
f = x*x + y*y + u*u + v*v; // cost function
```

```
float x, y, z, dx[2], dy[2], dz[2]; // variables
float u, v, du[2], dv[2];           // controls
float f, df[2];                     // cost variable
```

```
x = y + z + u*u;    // statement #1
```

```
y = z*y + x*v;      // statement #2
```

```
f = x*x + y*y + u*u + v*v; // cost function
// df[0] = ???;
// df[1] = ???;
```

```
float x, y, z, dx[2], dy[2], dz[2]; // variables
float u, v, du[2], dv[2];           // controls
float f, df[2];                     // cost variable

dx[0]=dx[1]=dy[0]=dy[1]=dz[0]=dz[1]=df[0]=df[1]=0;
du[0]=1; du[1]=0; dv[0]=0; dv[1]=1;

x = y + z + u*u;    // statement #1



y = z*y + x*v;      // statement #2



f = x*x + y*y + u*u + v*v; // cost function
```

```
float x, y, z, dx[2], dy[2], dz[2]; // variables
float u, v, du[2], dv[2];           // controls
float f, df[2];                     // cost variable

dx[0]=dx[1]=dy[0]=dy[1]=dz[0]=dz[1]=df[0]=df[1]=0;
du[0]=1; du[1]=0; dv[0]=0; dv[1]=1;

x = y + z + u*u;     // statement #1
   dx[0] = dy[0] + dz[0] + 2*u*du[0];
   dx[1] = dy[1] + dz[1] + 2*u*du[1];

y = z*y + x*v;       // statement #2
   dy[0] = z*dy[0] + y*dz[0] + v*dx[0] + x*dv[0];
   dy[1] = z*dy[1] + y*dz[1] + v*dx[1] + x*dv[1];

f = x*x + y*y + u*u + v*v; // cost function
```

```
float x, y, z, dx[2], dy[2], dz[2]; // variables
float u, v, du[2], dv[2];           // controls
float f, df[2];                     // cost variable

dx[0]=dx[1]=dy[0]=dy[1]=dz[0]=dz[1]=df[0]=df[1]=0;
du[0]=1; du[1]=0; dv[0]=0; dv[1]=1;

x = y + z + u*u;    // statement #1
   dx[0] = dy[0] + dz[0] + 2*u*du[0];
   dx[1] = dy[1] + dz[1] + 2*u*du[1];

y = z*y + x*v;      // statement #2
   dy[0] = z*dy[0] + y*dz[0] + v*dx[0] + x*dv[0];
   dy[1] = z*dy[1] + y*dz[1] + v*dx[1] + x*dv[1];

f = x*x + y*y + u*u + v*v; // cost function
   df[0] = 2*x*dx[0] + 2*y*dy[0] + 2*u*du[0] + 2*v*dv[0];
   df[1] = 2*x*dx[1] + 2*y*dy[1] + 2*u*du[1] + 2*v*dv[1]
```

Desvantagem

A tecnica anterior é muito costosa quando tem muitos controles

```
 #define NC 1,000,000

float x, y, z, dx[NC], dy[NC], dz[NC]; // variables
float u[NC], du[NC];                   // controls
float f, df[NC];                       // cost variable

x = y + z + u[0]*u[0];   // statement #1
for ( int i=0 ; i<NC ; i++ ) {
   dx[i] = dy[i] + dz[i] + ( i==0 ? 2*u[0]*du[0] : 0 );
}
```

- Um programa esta composto por um conjunto de instruções. $I_1, I_2, \ldots, I_n$
- Adjunto de uma instrução I é $A(I)$
- O programa adjunto inverte a ordem das instruções

$$A(I_n), A(I_{n-1}), \ldots, A(I_1)$$

$$x = x^3 + sen(y^2)z$$
$$dx = 3x^2dx \; + \; 2ycos(y^2)zdy \; + \; sen(y^2)dz$$

$$\begin{pmatrix} dz \\ dy \\ dx \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ sen(y^2) & 2ycos(y^2)z & 3x^2 \end{pmatrix} \begin{pmatrix} dz \\ dy \\ dx \end{pmatrix}$$

$$\begin{pmatrix} z^* \\ y^* \\ x^* \end{pmatrix} = \begin{pmatrix} 1 & 0 & sen(y^2) \\ 0 & 1 & 2ycos(y^2)z \\ 0 & 0 & 3x^2 \end{pmatrix} \begin{pmatrix} z^* \\ y^* \\ x^* \end{pmatrix}$$

$$\begin{aligned} z^* &= z^* + sen(y^2)x^* \\ y^* &= y^* + 2ycos(y^2)zx^* \\ x^* &= x^* + 3x^2x^* \end{aligned}$$

$$I :=$$

$$x = f(x, y, \ldots, z)$$

$$\begin{pmatrix} dz \\ \vdots \\ dy \\ dx \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 \\ f_z(x, y, \cdots, z) & \cdots & f_y(x, y, \cdots, z) & f_x(x, y, \cdots, z) \end{pmatrix} \begin{pmatrix} dz \\ \vdots \\ dy \\ dx \end{pmatrix}$$

$$
\begin{pmatrix} z^* \\ \vdots \\ y^* \\ x^* \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 & f_z(x, y, \cdots, z) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & f_y(x, y, \cdots, z) \\ 0 & \cdots & 0 & f_x(x, y, \cdots, z) \end{pmatrix} \begin{pmatrix} z^* \\ \vdots \\ y^* \\ x^* \end{pmatrix}
$$

$$
\begin{aligned}
z^* &= z^* + f_z(x, y, \cdots, z)x^* \\
&\vdots \\
y^* &= y^* + f_y(x, y, \cdots, z)x^* \\
x^* &= f_x(x, y, \cdots, z)x^*
\end{aligned}
$$

$$A\begin{pmatrix}\texttt{for i= } 0,\cdots,n \texttt{ do}\\ I_i\\ \texttt{end for}\end{pmatrix} = \begin{matrix}\texttt{for i= } n,\cdots,0 \texttt{ do}\\ A(I_i)\\ \texttt{end for}\end{matrix}$$

$$A\begin{pmatrix}\texttt{if B then}\\ I_1\\ \texttt{else}\\ I_2\\ \texttt{end if}\end{pmatrix} = \begin{matrix}\texttt{if B then}\\ A(I_1)\\ \texttt{else}\\ A(I_2)\\ \texttt{end if}\end{matrix}$$

```
x = y + z + u*u;    // statement #1
   dx[0] = dy[0] + dz[0] + 2*u*du[0];
   dx[1] = dy[1] + dz[1] + 2*u*du[1];
```

$$\begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 2*u & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix}$$

$$X_1 = A_1 X_0$$

```
y = z*y + x*v;       // statement #2
   dy[0] = z*dy[0] + y*dz[0] + v*dx[0] + x*dv[0];
   dy[1] = z*dy[1] + y*dz[1] + v*dx[1] + x*dv[1];
```

$$\begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ v & z & y & 0 & x \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix}$$

$$X_2 = A_2 X_1$$

```
f = x*x + y*y + u*u + v*v; // cost function
  df[0] = 2*x*dx[0] + 2*y*dy[0] + 2*u*du[0] + 2*v*dv[0];
  df[1] = 2*x*dx[1] + 2*y*dy[1] + 2*u*du[1] + 2*v*dv[1];
```

$$\begin{pmatrix} df[0] & df[1] \end{pmatrix} = \begin{pmatrix} 2*x & 2*y & 2*z & 2*u & 2*v \end{pmatrix} \begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix}$$

$$f = p_2^\top X_2$$

$$X_1 = A_1 X_0 \qquad p_1 = A_2^\top p_2$$

$$X_2 = A_2 X_1 \qquad p_0 = A_1^\top p_1$$

$$f = p_2^\top X_2 \qquad f = p_0^\top X_0$$

$$p_0^\top X_0 = \left(A_1^\top A_2^\top p_2\right)^\top X_0 = p_2^\top A_2 A_1 X_0 = p_2^\top X_2$$

$$A = A_2 A_1$$

$$X_2 = AX_0 \quad \longmapsto \quad f = p_0^\top X_2$$

$$f^* = p_1^\top X_0 \quad \longleftarrow \quad p_1 = A^\top p_0$$

$$f = f^*$$

$$f = p_0^\top X_2 = p_0^\top A X_0 = (A^\top p_0)^\top X_0 = p_1^\top X_0 = f^*$$

```
y = z*y + x*v;       // statement #2
   dy[0] = z*dy[0] + y*dz[0] + v*dx[0] + x*dv[0];
   dy[1] = z*dy[1] + y*dz[1] + v*dx[1] + x*dv[1];
```

$$\begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ v & z & y & 0 & x \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} dx[0] & dx[1] \\ dy[0] & dy[1] \\ dz[0] & dz[1] \\ du[0] & du[1] \\ dv[0] & dv[1] \end{pmatrix}$$

$$X_1 = AX_0$$

```
y = z*y + x*v;        // statement #2
```

$$\begin{pmatrix} ax \\ ay \\ az \\ au \\ av \end{pmatrix} = \begin{pmatrix} 1 & v & 0 & 0 & 0 \\ 0 & z & 0 & 0 & x \\ 0 & y & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & x & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} ax \\ ay \\ az \\ au \\ av \end{pmatrix}$$

$$X_1 = AX_0$$

```
y = z*y + x*v;       // statement #2
   ax = ax + v*ay;
   az = az + y*ay;
   au = au;
   av = av + x*ay;
   ay = z*ay;
```

$$\begin{pmatrix} ax \\ ay \\ az \\ au \\ av \end{pmatrix} = \begin{pmatrix} 1 & v & 0 & 0 & 0 \\ 0 & z & 0 & 0 & x \\ 0 & y & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & x & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} ax \\ ay \\ az \\ au \\ av \end{pmatrix}$$

$$X_1 = AX_0$$

```
float x, y, z;                          // variables
float u, v;                             // controls
float f;                                // cost variable

x = y + z + u*u;    // statement #1



y = z*y + x*v;      // statement #2



f = x*x + y*y + u*u + v*v; // cost function
```

```
float x, y, z, ax, ay, az;          // variables
float u, v, au, av;                 // controls
float f, af;                        // cost variable

x = y + z + u*u;    // statement #1



y = z*y + x*v;      // statement #2



f = x*x + y*y + u*u + v*v; // cost function
```

```
float x, y, z, ax, ay, az;          // variables
float u, v, au, av;                 // controls
float f, af;                        // cost variable

x = y + z + u*u;    // statement #1
   ay += ax; az += ax;
   au += 2*u*ax; ax = 0;

y = z*y + x*v;      // statement #2
   az += y*ay; ax += u*ay;
   av += x*ay; ay = z*ay;

f = x*x + y*y + u*u + v*v; // cost function
   ax += 2*x*af; ay += 2*y*af;
   au += 2*u*af; av += 2*v*af;

ax = ay = az = au = av = 0;
af = 1;
```

Codigo Adjunto

```
float x, y, z, ax, ay, az;          // variables
float u, v, au, av;                 // controls
float f, af, df[2];                 // cost variable

af = 1;
ax = ay = az = au = av = 0;
// cost function
ax += 2*x*af; ay += 2*y*af;
au += 2*u*af; av += 2*v*af;
// statement #2
az += y*ay; ax += u*ay;
av += x*ay; ay = z*ay;
// statement #1
ay += ax; az += ax;
au += 2*u*ax; ax = 0;
// gradient
df[0] = au;
df[1] = av;
```

```
float x, y, z, ax, ay, az;
float u, v, au, av;
float f, af, df[2];

af = 1;
ax = ay = az = au = av = 0;
// cost function
ax += 2*x*af; ay += 2*y*af;
au += 2*u*af; av += 2*v*af;
// statement #2
az += y*ay; ax += u*ay;
av += x*ay; ay = z*ay;
// statement #1
ay += ax; az += ax;
au += 2*u*ax; ax = 0;
// gradient
df[0] = au;
df[1] = av;
```

```
float x, y, z;
float u, v;
float f;

// statement #1
x = y + z + u*u;

// statement #2
y = z*y + x*v;

// cost function
f = x*x + y*y + u*u + v*v;
```

Referências

- Recipes for Adjoint Code Construction
  RALF GIERING
- An Introduction to the Adjoint Approach to Design
  MICHAEL B. GILES
- Simulation and Control of Physical Phenomena in Computer Graphics
  JOS STAM

Obrigada!!!

dalia@impa.br