A STANDARD INTERFACE FOR ELIMINATION SEQUENCES IN AUTOMATIC DIFFERENTIATION

UWE NAUMANN *AND SHAUN FORTH, MOHAMED TADJOUDDINE AND JOHN PRYCE †

Abstract. We propose a standard format for exchanging linearized computational graphs and vertex, edge, and face elimination sequences between various AD software tools.

1. CGS Files. Let

$$F : \mathbb{R}^3 \to \mathbb{R}^3 : \{\mathtt{v1}, \mathtt{v2}, \mathtt{v3}\} \mapsto \{\mathtt{v4}, \mathtt{v6}, \mathtt{v7}\}$$

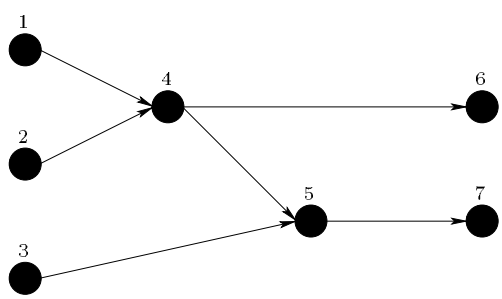
be given by the following code list:

```
v4=v1-v2
v5=v4*v3
v6=v4+1
v7=sqrt(v5)
```

Local Partial Derivatives.

```
c[4,1]=1
c[4,2]=-1
c[5,3]=v4
c[5,4]=v3
c[6,4]=1
c[7,5]=1/(2*v7)
```

LCG.



CGS File.

```
title='Example 1'
% A simple example
cg_size=[3 4 3]
dep_vertices=[4 6 7]
% 1, 2, and 3 are minimal vertices
% 4, 6, and 7 are maximal vertices
% they are connected by the following 6 edges
cg_edges=[
4 1 1
```

*ARGONNE NATIONAL LABORATORY, USA, ANL/MCS-P934-0202
†ALL: CRANFIELD UNIVERSITY (RMCS SHRIVENHAM), UK - AMOR REPORT 2001/3

```
4 2 1
5 4 2
5 3 2
6 4 1
7 5 2
]
% three of which are trivial
```

2. ESS Files. Elimination sequences are given as *elimination sequence specification (ESS) files*. A general elimination sequence is based on faces. Both edge and vertex eliminations represent special cases. However it is deemed desirable that the elimination sequence may be given as either a sequence of vertices or edges or faces, or a combination. To specify a vertex for elimination requires specification of that vertex, i.e. one non-negative integer. To specify an edge requires the two vertices at it's ends (two non-negative integers). To specify a face requires three non-negative integers.

`size_elim_seq` is used to define the number of eliminations and their maximum rank (vertex= 1, edge= 2, face=3).

Any edge $(\mathtt{j}, \mathtt{i}) \in E$ can be either *front*- or *back*-eliminated. In the ESS file the front-elimination of `(j,i)` is specified by `j i` Analogous. back-elimination takes place for `i j` Neither for vertex nor for face elimination further semantical clarifications are neccessary.

ESS File.

```
title='Example 1'
% ESS file example
size_elim_seq=[3 2]
% back-elimination of edge (6,4) followed by
% front-elimination of edge (5,4) and
% vertex elimination of 5 and
% back-elimination of edge (7,4)
elim_seq=[
4 6
5 4
5
4 7
]
% total cost would be 6 mults
```

Jacobian Code.

```
v4 = v1 - v2
c_4_1 = 1
c_4_2 = -1

v5 = v4 * v3
c_5_3 = v_4
c_5_4 = v_3

v6 = v4 + 1
c_6_4 = 1
```

```
v7 = sqrt(v5)
c_7_5 = 1 / (2 * v_7)

! back-eliminate (6,4)
c_6_1 = c_6_4 * c_4_1
c_6_2 = c_6_4 * c_4_2

! front-eliminate edge (5,4)
c_7_4 = c_7_5 * c_5_4

! eliminate vertex 5
c_7_3 = c_7_5 * c_5_3

! back-eliminate (7,4)
c_7_1 = c_7_4 * c_4_1
c_7_2 = c_7_4 * c_4_2

! form Jacobian
J(1,1) = c_4_1; J(2,1) = c_6_1; J(3,1) = c_7_1
J(1,2) = c_4_2; J(2,2) = c_6_2; J(3,2) = c_7_2
J(1,3) = 0;     J(2,3) = 0;     J(3,3) = c_6_3
```

Optimized Jacobian Code.

```
v4 = v1 - v2
c_4_1 = 1
c_4_2 = -1

v5 = v4 * v3
c_5_3 = v_4
c_5_4 = v_3

v6 = v4 + 1
c_6_4 = 1

v7 = sqrt(v5)
c_7_5 = 1 / (2 * v_7)

! back-elimination of (6,4) is free

! front-eliminate edge (5,4)
c_7_4 = c_7_5 * c_5_4

! eliminate vertex 5
c_7_3 = c_7_5 * c_5_3

! back-eliminate (7,4) is free

! form Jacobian
J(1,1) = c_4_1; J(2,1) = c_6_1; J(3,1) = c_7_1
J(1,2) = c_4_2; J(2,2) = c_6_2; J(3,2) = c_7_2
```

```
J(1,3) = 0;     J(2,3) = 0;     J(3,3) = c_6_3
```

3. Vertex Elimination.

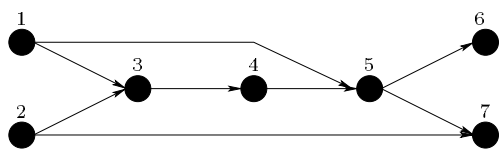
Source Code.

```
x1=x1/exp(x1*x2)
y1=sin(x1)
y2=x1*x2
```

Code List.

```
v1=x1
v2=x2
v3=v1*v2
v4=exp(v3)
v5=v1/v4
v6=sin(v5)
v7=v5*v2
y1=v6
y2=v7
```

LCG.



CGS File.

```
title='Example 2'
cg_size=[2 5 2]
% v1, v2 independent
% v6, v7 dependent
cg_edges=[
3 1
5 1
3 2
7 2
4 3
5 4
6 5
7 5
]
% all edges non-trivial
```

ESS File.

```
title='Example 2'
size_elim_seq=[3 1]
% optimal vertex elimination sequence
elim_seq=[
4
3
5
]
```

4. Edge Elimination.

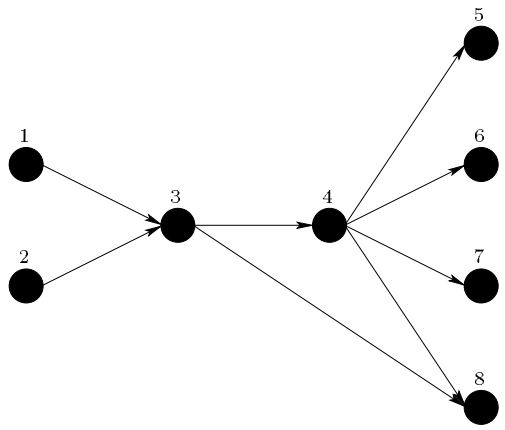
Source Code.

```
h1=x1*x2
h2=sin(h1)
y1=h2+3.4
y2=5.3*h2+1.0
y3=exp(h2)
y4=h1/h2
```

Code List.

```
v1=x1
v2=x2
v3=v1*v2
v4=sin(v3)
v5=v4+3.4
v6=5.3*v4+1.0
v7=exp(v4)
v8=v3/v4
y1=v5
y2=v6
y3=v7
y4=v8
```

LCG.



CGS File.

```
title='Lion Graph'
cg_size=[2 2 4]
cg_edges=[
3 1
3 2
4 3
8 3
5 4
6 4
7 4
8 4
```

```
]
```

ESS File.

```
title='Lion Graph'
size_elim_seq=[3 2]
% back-elimination of edge (8,4) followed by
% vertex elimination of 3 and 4
elim_seq=[
4 8
3
4
]
```

5. Face Elimination.

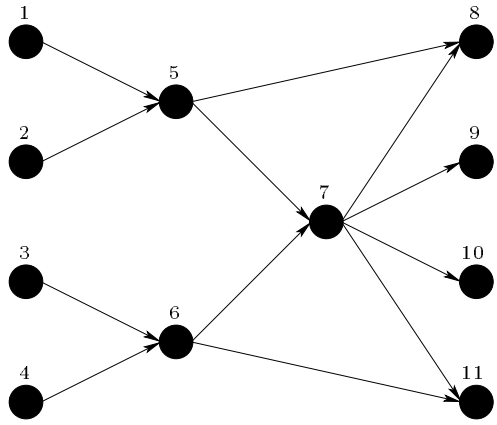
Source Code.

```
v5=x1*x2
v6=x3*x4
v7=v5*v6
y1=v5*v7
y2=exp(v7)
y3=cos(v7)
y4=v6*v7
```

Source Code.

```
v1=x1
v2=x2
v3=x3
v4=x4
v5=v1*v2
v6=v3*v4
v7=v5*v6
v8=v5*v7
v9=exp(v7)
v10=cos(v7)
v11=v6*v7
y1=v8
y2=v9
y3=v10
y4=v11
```

LCG.



CGS File.

```
title='Bat Graph'
cg_size=[4 3 4]
cg_edges=[
5 1
5 2
6 3
6 4
7 5
8 5
7 6
11 6
8 7
9 7
10 7
11 7
]
```

ESS File.

```
title='Bat Graph'
size_elim_seq=[10 3]
elim_seq=[
8 7 5
11 7 6
5
6
11 7 1
11 7 2
8 7 3
8 7 4
7 9
7 10
]
```

Jacobian Code.

```
! computation of c_5_1 ... c_11_7
! during one forward sweep
```

```
...

! eliminate face (8,7,5)
c_8_5 = c_8_5 + c_8_7 * c_7_5

! eliminate face (11,7,6)
c_11_6 = c_11_6 + c_11_7 * c_7_6

! eliminate vertex 5
c_7_1 = c_7_5 * c_5_1
c_7_2 = c_7_5 * c_5_2
c_8_1 = c_8_5 * c_5_1
c_8_2 = c_8_5 * c_5_2

! eliminate vertex 6
c_7_3 = c_7_6 * c_6_3
c_7_4 = c_7_6 * c_6_4
c_11_3 = c_11_6 * c_6_3
c_11_4 = c_11_6 * c_6_4

! eliminate face (11,7,1)
c_11_1 = c_11_7 * c_7_1

! eliminate face (11,7,2)
c_11_2 = c_11_7 * c_7_2

! eliminate face (8,7,3)
c_8_3 = c_8_7 * c_7_3

! eliminate face (8,7,4)
c_8_4 = c_8_7 * c_7_4

! back-eliminate (9,7)
c_9_1 = c_9_7 * c_7_1
c_9_2 = c_9_7 * c_7_2
c_9_3 = c_9_7 * c_7_3
c_9_4 = c_9_7 * c_7_4

! back-eliminate (10,7)
c_10_1 = c_10_7 * c_7_1
c_10_2 = c_10_7 * c_7_2
c_10_3 = c_10_7 * c_7_3
c_10_4 = c_10_7 * c_7_4

! J(i,j)=c_{i+7}_j for i,j=1,...,4
...
```

6. Vector Operations.

Source Code.

```
real, dimension(4) :: y, h1, h3
```

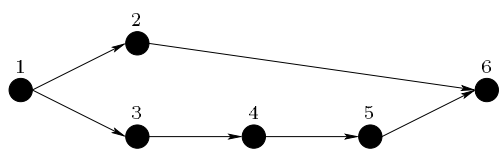
```
real, dimension(2) :: x, h2

h1=lion(x)
h2=iron(x)
h3=bat(lion(h2))
y=h1+h3
```

Code List.

```
v1=x
v2=lion(v1)
v3=iron(v1)
v4=lion(v3)
v5=bat(v4)
v6=v2+v5
```

LCG.



CGS File.

```
title='Vector-valued Elemental Functions'
cg_size=[1 4 1]
cg_vertices=[
1 2
2 4
3 2
4 4
5 4
6 4
]
cg_edges=[
2 1 2
3 1 2
4 3 2
5 4 2
6 2 1
6 5 1
]
```

ESS File.

```
title='Vector-valued Elemental Functions'
size_elim_seq=[2 1]
elim_seq=[
2
3
4
]
```

Jacobian Code.

```
real, dimension(4) :: y, v3, v4
real, dimension(2) :: x, v2
real, dimension(4,2) :: C_2_1, C_4_3
real, dimension(2,2) :: C_3_1
real, dimension(4,4) :: C_5_4

C_2_1=jac_lion(x)

v3=iron(x)
C_3_1=jac_iron(x)

v4=lion(v2)
C_4_3=jac_lion(v2)

v5=bat(v3)
C_5_4=jac_bat(v3)

J=C_2_1+C_5_4*C_4_3*C_3_1
```

7. Formal Specification of CGS and ESS File Formats.

```
digit            [0-9]
number1          [1-9]{digit}*
number0          0|{number1}
char             [A-Z,a-z_]
string           {char}({char}|{digit})*
qstring          "'"({string}|{number0})*"'"
delim            [ \t\n]
{delim}          {}
cm               "%"(.* \n)*(\n)?
{cm}             {}

vertex           {number1}
result_vertex    {vertex}
argument_vertex  {vertex}
label            {trivial}|{non_trivial}
trivial          1
non_trivial      2
edge             {result_vertex}{argument_vertex}{label}?
cg_edges         "cg_edges=[\n"({edge}"\n")*"]"
dim              {number0}
cg_vertices      "cg_vertices=[\n"({vertex}{dim}"\n")*"]"
title            "title="{qstring}
no_indep         {number1}
no_inter_dep     {number1}
no_dep           {number1}
cg_size          "cg_size=["{no_indep}{no_inter_dep}{no_dep}"]"
dep_vertices     "dep_vertices=["{vertex}+"]"
cgs_file         {cm}*{title}{cg_size}{dep_vertices}?{cg_vertices}?{cg_edges}

size_elim_seq    "size_elim_seq=["{no_elims}{elim_rank}"]"
no_elims         {number1}
elim_rank        {number1}
elim_seq         "elim_seq=["({elimination}"\n")+"]"
elimination      {vertex_elim}|{edge_elim}|{face_elim}
vertex_elim      {vertex}
edge_elim        {vertex}{vertex}
face_elim        {vertex}{vertex}{vertex}
ess_file         {title}{size_elim_seq}{elim_seq}
```