In[ ]:=
```
(* Initialisation *)
(* Evaluate before start writing "real code" *)
(* Usage e.g.: "ld [Spacekey]" becomes "⊨",
so writing "a ld 5" turns into "a ⊨ 5" *)
SetOptions[EvaluationNotebook[],
            InputAutoReplacements → {(* special AceGen assignment operators: *)
   "ld" → "⊨", "ls" → "⊢", "rd" → "⫤", "rs" → "⊣",
                                    (* brackets and symbols: *) "dbl" → "⟦",
   "dbr" → "⟧", "lcb" → "{", "rcb" → "}", "lsb" → "[", "rsb" → "]", "->" → "→",
                                    (* shortcuts for
    starting/ending a comment block: *) "co" → "(*", "cc" → "*)"
                                   }
          ]
(* Output the current time,
so we know when AceGen has been executed the last time *)
Now
```

Out[ ]= Fri 19 Apr 2024 17:10:07 GMT+2

In[ ]:=
```
(* Clear all old variables initially to have a fresh start *)
ClearAll["Global`*"]
(* Start AceGen *)
<< AceGen`;
(* Name of the to be created subroutine/function
 in the below specified programming language *)
NAME = "InCalcOut";
(* Name of the AceGen session "NAME",
specify the programming language "Language"={C++,Matlab,Fortran,...},
and the execution mode "Mode"={Optimal,Prototype,Debug,Plain} *)
(* @note Changing the output programming language can be very simple here,
so feel free to take advantage of all the available languages. For instance,
first export to a Matlab-code,
because you can quickly and easily debug and check the code and its output. *)
SMSInitialize[NAME, "Language" → "Matlab", "Mode" → "Optimal"];
(* Start a module, which represents the to be created function,
with name "NAME" and the specified input and output arguments *)
SMSModule[NAME, Real[xInput$$, yOutput$$, DyDxOutput$$],
          "Input" → {xInput$$},
          "Output" → {yOutput$$, DyDxOutput$$}
         ];
(* Input declaration by copying AceGen
 input variables to Mathematica variables *)
x ⊨ SMSReal[xInput$$];
```

```
In[ ]:= (* Compute the output variable y based on the input x,
here using some arbitrary expression *)
y ⊨ 25 + x^2 + 1/x + Sin[x];

In[ ]:= (* Export the output variables by copying the
 Mathematica variables to AceGen output variables *)
SMSExport[y , yOutput$$];

In[ ]:= (* Compute the derivative of y with respect to x *)
Dy_Dx ⊨ SMSD[y, x];
(* Output/Export the derivative *)
SMSExport[Dy_Dx, DyDxOutput$$ ];

(* Debugging *)
(* Output y to the screen *)
SMSPrintMessage[NAME <> "<< y=", y];
(* Compute the analytical derivative and compare it to the AceGen-output *)
(* @note Note that for this simple model, Mathematica even optimised
 the expression for the error shown below in the generated code *)
derivative_analytical ⊨ 2 x + -1/x^2 + Cos[x];
SMSPrintMessage[NAME <> "<< error in derivative=", Dy_Dx - derivative_analytical ];

In[ ]:= (* Output the time at the end of the execution *)
Now
(* Write output file containing all the
 above defined functions introduced by SMSModule *)
(* Create output file named "NAME", '"LocalAuxiliaryVariables " →
 True' is a command to exclude the AceGen internal array "v" from
  the list of input and output arguments of the created subroutine *)
SMSWrite[NAME, "LocalAuxiliaryVariables " → True];
(* Print the content of the just created
 file on screen (sensible only for small file sizes) *)
FilePrint[ StringJoin[NAME, Which[SMSLanguage == "Fortran", ".f",
                                  SMSLanguage == "Matlab", ".m",
                                  SMSLanguage == "C++", ".cpp",
                                  SMSLanguage == "C", ".c"
                                 ]
                     ]
          ]

Out[ ]= Fri 19 Apr 2024 17:10:08 GMT+2
```

File: InCalcOut.m **Size:** 1300 **Time:** 1

Method	InCalcOut
No.Formulae	6
No.Leafs	59

```
%**************************************************************
%* AceGen    7.505 Linux (16 Aug 22)                          *
%*           Co. J. Korelc  2020           19 Apr 24 17:10:08 *
%**************************************************************
% User     : Full professional version
% Notebook : AceGen-InCalcOut
% Evaluation time                 : 1 s     Mode  : Optimal
% Number of formulae              : 6       Method: Automatic
% Subroutine                      : InCalcOut size: 59
% Total size of Mathematica  code : 59 subexpressions
% Total size of Matlab code       : 406 bytes

%********************** F U N C T I O N ************************
function[yOutput,DyDxOutput]=InCalcOut(xInput);
persistent v;
if size(v)<117
  v=zeros(117,'double');
end;
v(1)=xInput;
v(2)=25e0+1/v(1)+(v(1)*v(1))+sin(v(1));
yOutput=v(2);
DyDxOutput=-(1/Power(v(1),2))+2e0*v(1)+cos(v(1));
disp(sprintf("\n%s  %f ","InCalcOut<<  y=",v(2)));
disp(sprintf("\n%s  %f ","InCalcOut<<  error in derivative=",0));


function [x]=SMSKDelta(i,j)

if (i==j) , x=1; else x=0; end;

end

function [x]=SMSDeltaPart(a,i,j,k)

l=round(i/j);

if (mod(i,j) ~= 0 | l>k) , x=0; else x=a(l); end;

end

function [x]=Power(a,b)

x=a^b;

end

function [x]=SMSTernaryOperator(a,b,c)

if (c) , x=a; else x=b; end;

end

end
```