W = emgr(f,g,s,t,w,pr,nf,ut,us,xs,um,xm);

emgr – Empirical Gramian Framework (Version 3.9)

Mandatory Arguments

`f`	System Vector Field	(Function Handle)	`x = f(x,u,p)`	i.e.: **f = @(x,u,p) A*x+B*u+F*p;**
`g`	Output Functional	(Function Handle)	`y = g(x,u,p)`	i.e.: **g = @(x,u,p) C*x+D*u;**
		`1`	`y = x`	
`s`	System Dimensions	(Vector)	`s = [J,N,O]`	(Inputs, States, Outputs)
`t`	Time	(Vector)	`t = [h,T]`	(Step, Stop)
`w`	Gramian Type	(Character)		
		`'c'`	Empirical Controllability Gramian (returns **WC**)	
		`'o'`	Empirical Observability Gramian (returns **WO**)	
		`'x'`	Empirical Cross Gramian (returns **WX**)	
		`'y'`	Empirical Linear Cross Gramian (returns **WY**)	
		`'s'`	Empirical Sensitivity Gramian (returns **WS = {WC,WS}**)	
		`'i'`	Empirical Identifiability Gramian (returns **WI = {WO,WI}**)	
		`'j'`	Empirical Joint Gramian (returns **WJ = {WX,WI}**)	

Optional Arguments

`pr`	Parameters	(Vector)	Column Vector of System Parameters (Default: **pr = 0**)
		(Matrix)	Set of Parameter Column Vectors ('**s**','**i**','**j**' requires two)
`nf`	Options	(Vector)	Ten Components (Default: **nf = 0**), see Option Flags
`ut`	Input Function	(Scalar)	Uniformly Scaled Impulse Input (Default: **ut = 1**)
		(Vector)	Individual Scaled Impulse Input (**J x 1**)
		(Matrix)	Discrete Input Function (**J x (T-S)/h**)
		(Function Handle)	Function Handle (**u = ut(t)**)
		`∞`	Chirp Function
`us`	Steady-State Input	(Scalar)	Uniform Steady-State Input (Default: **us = 0**)
		(Vector)	Individual Steady-State Input (**J x 1**)
`xs`	Steady State	(Scalar)	Uniform Steady State (**Default: xs = 0**)
		(Vector)	Individual Steady States (**N x 1**)
`um`	Input Scales	(Scalar)	Uniform Maximum Input Scales (Default: **um = 1**)
		(Vector)	Individual Maximum Input Scales (**J x 1**)
		(Matrix)	Custom Input Scales (**J x ***)
`xm`	Steady-State Scales	(Scalar)	Uniform Maximum Steady State Scales (Default: **xm = 1**)
		(Vector)	Individual Maximum Steady-State Scales (**N x 1**)
		(Matrix)	Custom Steady-State Scales (**N x ***)

Option Flags

`nf(1)`	Trajectory Centering	
	`0`	None (Default)
	`1`	Initial State
	`2`	Final Steady State
	`3`	Arithmetic Average
	`4`	Median
	`5`	Midrange
	`6`	Root-Mean-Squared
`nf(2)`	Input Scale Sequence	
	`0`	Linear (Default)
	`1`	Logarithmic
	`2`	Geometric
	`3`	Single
	`4`	Sparse
`nf(3)`	State Scale Sequence	
	`0`	Linear (Default)
	`1`	Logarithmic
	`2`	Geometric
	`3`	Single
	`4`	Sparse
`nf(4)`	Input Transformation	
	`0`	Unit (Default)
	`1`	Inverse
	`2`	Dyadic
	`3`	Single
`nf(5)`	State Transformation	
	`0`	Unit (Default)
	`1`	Inverse
	`2`	Dyadic
	`3`	Single
`nf(6)`	Preconditioning	
	`0`	None (Default)
	`1`	Jacobi (Double Run)
	`2`	Steady-State Scaled
`nf(7)`	Non-Symmetric Cross Gramian	
	`0`	Off (Default)
	`1`	Non-Sym. Cross-Gramian (**WX,WJ** only)
`nf(8)`	Robust Parameters	
	`0`	Off (Default)
	`1`	Treat Parameters as Inputs
`nf(9)`	Parameter Action	
	`0`	Active Parameters
	`1`	Passive Parameters
`nf(10)`	Center Parameter Scales	
	`0`	No Centering
	`1`	Mean Centered Parameters
	`2`	Logarithmic Centered Parameters
`nf(11)`	Exclusive Options	
	`0`	None (Default)
	`1`	Root-Mean-Square-Centering (**WS** only)
	`1`	Schur Complement (**WI** only)
	`1`	Detailed Schur Complement (**WJ** only)
`nf(12)`	Gramian Symmetry	
	`0`	Assume Symmetry
	`1`	Enforce Symmetry

Custom Solver

Set global variable **ODE** to solver function handle
with signature: **y = solver(f,g,t,x,u,p);**
default solver: 2nd Order Ralston's Runge-Kutta

Minimal Usage: W = emgr(f,g,s,t,w);

About Info: V = emgr('version');

More info at: http://gramian.de