`dpvar`: A new class for polynomial variables in MATLAB

Sachin Shivakumar * Declan Jagt † Matthew Peet ‡

September 15, 2021

Contents

*Arizona State University
†Arizona State University
‡Arizona State University

1 Introduction

SOSTOOLS is a toolbox designed for parsing and solving of polynomial sum-of-squares optimization problems in MATLAB. A polynomial sum-of-squares optimization problem is typically solved by following the steps:

- Define symbolic independent variables (typically of type `polynomial` or `sym` in MATLAB).
- Define polynomials with unknown coefficients as decision variables (decision variables are of **same type** as above)
- Define equality or inequality constraints on the polynomials involving decision variables.
- Declare an objective function.
- Convert the constraints and objective function (from SOSTOOLS format) to parameters compatible with an SDP solver.
- Solve the optimization problem by calling the specified SDP solver.
- Convert solver output back to SOSTOOLS format (or other user-friendly formats).

While computationally solving an SOS problem, in most cases, the first 5 steps take a significant amount of computation time in comparison to the solution step where the actual SDP is solved. The reason behind this is the highlighted in the 2^{nd} bullet above.

Specifying decision variables and storing them along with independent symbolic variables in a single structure complicates many operations such as indexed access, multiplication, addition, etc., and artificially inflates time taken during search or sorting operations. Moreover, while decision variables will always appear linearly in the polynomial, never to be multiplied, differentiated, etc., the implementation of polynomial objects in SOSTOOLS up to version 3.04 fails to fully exploit the benefits of this linearity.

With SOSTOOLS 4.0, we reduce computational complexity and memory requirements by introducing a new MATLAB class of objects: the `dpvar`. Objects of this class are different from `polynomial` and `syms` objects, in the sense that decision variables and independent variables are now stored under separate fields (see 2.2). As a result, operations on `dpvar` objects are significantly less computationally demanding than the same operations on `polynomial` and `syms` objects, allowing for much more efficient parsing of polynomial optimization problems then in previous versions of SOSTOOLS.

This manual presents an overview of how the `dpvar` object is implemented in MATLAB, along with some basic operations that may be performed on `dpvar` objects.

2 Class definitions

In this section, we introduce the existing class `polynomial` along with the new class `dpvar` to highlight differences between the two classes.

2.1 `polynomial` class object

Syntax:

```
pvar s t; %creates independent polynomial variables
C = polynomial(coeff, degmat, [s;t], matdim); %construct using polynomial structure
```

Currently, a polynomial class object has the four properties. For any `[m,n]` dimensional polynomial object with `p` independent polynomial variables and a monomial set of length `z` (with various degrees of `p`) has the properties

- `coeff`: sparse matrix of size `[z,(m*n)]`
- `matdim`: array `[m,n]`

- `varname`: a cell-string of size `[p 1]`
- `degmat`: a matrix of size `[z p]`

Given these properties, the polynomial can be reconstructed in two ways:

1. First, using elementwise exponential and elementwise multiplication, get the monomial vector

$$Z(p) = \texttt{varname}(1)^{\texttt{degmat}(:,1)}. * \cdots * \texttt{varname}(p)^{\texttt{degmat}(:,p)}.$$

Note that $\texttt{varname}(i)^{\texttt{degmat}(:,i)}$ is a `[z,1]`-dimensional matrix (`z`-dimensional vector). Then $Z(p)$ is a vector of dimension `z`, with each element a monomial in the variables `varname`.

2. Then, get the polynomial in linear-indexed vector (`m*n` dimensional ROW vector) using the formula

$$P(p) = Z(p)^T * \texttt{coeff}.$$

Use `reshape(P,m,n)` to reshape $P(p)$ to get the matrix of size `[m,n]`.

Alternatively, we can first reshape the coefficient matrix, and then multiply with the monomials. In particular, for a matrix pvar of size $m \times n$, the coefficient matrix is structured as

$$\begin{bmatrix} b_1 & \cdots & b_m & \cdots & b_{m(n-1)+1} & \cdots & b_{mn} \end{bmatrix} \\ = \begin{bmatrix} b_{1,1} & \cdots & b_{m,1} & b_{1,2} & \cdots & b_{m,2} & \cdots & b_{1,n} & \cdots & b_{m,n} \end{bmatrix},$$

where each b_i is a `z`-dimensional column vector. Constructing the monomial vector $Z(p)$ as in the previous approach, we may than reconstruct the polynomial itself as

$$P(p) = \overbrace{\begin{bmatrix} b_1^T & \cdots & b_{m(n-1)+1}^T \\ b_2^T & \cdots & \\ \vdots & & \vdots \\ b_m^T & \cdots & b_{mn}^T \end{bmatrix}}^{\hat{B}} (I_n \otimes Z(p)),$$

or, alternatively,

$$P(p) = (I_m \otimes Z(p))^T \overbrace{\begin{bmatrix} b_1 & \cdots & b_{m(n-1)+1} \\ b_2 & \cdots & \\ \vdots & & \vdots \\ b_m & \cdots & b_{mn} \end{bmatrix}}^{\bar{B}}$$

As an example, the polynomial

$$P(d,x) = \begin{bmatrix} 1 & 3d_1x_1 & 5d_1x_1^2 \\ 2d_2x_2 & 4d_2x_1 & 6d_2x_1x_2 \end{bmatrix}$$

may be stored as a `polynomial` object `P`, by assigning it the field values

```
P.varname = {'d_1';'d_2';'x_1';'x_2'};
P.dim = [2,3];
P.degmat = [0,0,0,0; 0,1,0,1; 1,0,1,0; 0,1,1,0; 1,0,2,0; 0,1,1,1];
P.coeff = [1,0,0,0,0,0; 0,2,0,0,0,0; 0,0,3,0,0,0,0;
           0,0,0,4,0,0; 0,0,0,0,5,0; 0,0,0,0,0,0,6];
```

Note that, since `coeff` and `degmat` will be stored as sparse objects, all the zeros in their expressions will not be stored.

2.2 `dpvar` class object: modification of `polynomial` class object

Syntax:

```
pvar p_1 p_2 ...  p_m; %creates independent polynomial variables
dvarnames = [‘coeff_0’,...,‘coeff_n’]; %creates array of strings with decision variables
C = dpvar(coeff, degmat, [p_1, ..., p_m], dvarnames, matdim); %construct using dpvar structure
```

Let us now introduce a new class `dpvar`, with an additional property that stores the list of decision variables. Let us define a polynomial with decision variables d and polynomial variables p as

$$\begin{aligned} D(p;d) &= (I_m \otimes Z(d))^T C (I_n \otimes Z(p)) \\ &= \begin{bmatrix} Z(d)^T C_{11} Z(p) & \cdots & Z(d)^T C_{1n} Z(p) \\ \vdots & \ddots & \vdots \\ Z(d)^T C_{m1} Z(p) & \cdots & Z(d)^T C_{mn} Z(p) \end{bmatrix} \end{aligned}$$

where I is an identity matrix and $A(p)$ is a `polynomial` class object of dimensions `[(d+1)*m,n]`. Unique property of decision variables is that their highest degree is always 1, integration and differentiation is never performed with respect to decision variables, and decision variables *should* not be multiplied with each other. In MATLAB, the `dpvar` class has the properties

- `C`: Sparse matrix of size `[((d+1)*m),z*n]`, describing the coefficient matrix C.
- `matdim`: array `[m,n]`, describing the matrix dimensions of the polynomial
- `varname`: a cell-string of size `[p 1]`, describing the independent variable names
- `degmat`: a matrix of size `[z p]`, describing the degrees of variables p in monomial vector $Z(p)$
- `dvarname`: a cell-string of size `[d 1]`, describing the decision variable names

As an example, the polynomial

$$P(d,x) = \begin{bmatrix} 1 & 3d_1x_1 & 5d_1x_1^2 \\ 2d_2x_2 & 4d_2x_1 & 6d_2x_1x_2 \end{bmatrix}$$

with decision variables d, may be stored as a `dpvar` object `P`, by assigning it the field values

```
P.varname = {‘x_1’;‘x_2’};
P.dvarname = {‘d_1’;‘d_2’};
P.dim = [2,3];
P.degmat = [0,0; 1,0; 0,1; 2,0; 1,1];
P.C = [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
       0,0,0,0,0,0,3,0,0,0,0,0,0,5,0;
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0;
       0,0,2,0,0,0,4,0,0,0,0,0,0,0,6];
```

Note that, since `C` and `degmat` will be stored as sparse objects, all the zeros in their expressions will not be stored.

From now on, ‘sparse matrix operations’ is used as an umbrella term to represent operations such as addition, multiplication, factorization, etc. Sparse matrix operations depend linearly on number of non-zero elements, number of rows, and number of columns but not on product of number of rows and columns.

Remark 1. *Some benefits, that are outright evident by adopting* ***`dpvar`*** *class representation instead of* ***`polynomial`*** *class representation are listed below.*

1. *Size of* `degmat` *reduces significantly.*
2. *Given a polynomial with decision variables, both number of non-zero elements and total size of* `coeff` *and* `C` *remain the same. However, the number of rows and columns are different.*
3. *Since sparse matrix operations are linearly dependent on number of rows and columns, complexity of sparse matrix operations on the two representations will be different.*
4. *In general,* `[m,n]` *is fixed and* `m<<z<<d`*. Hence, sparse matrix operations complexity may not change significantly. However, since* `m<z`*,* `dpvar` *representation is slightly better.*
5. *Concatenation and addition operations are likely of same complexity as* `polynomial` *representation.*

3 Operations involving `dpvar` objects in MATLAB

We now go over some operations on `dpvar` class objects, giving some mathematical background for how each operation has been implemented in MATLAB.

3.1 Auxiliary operations

3.1.1 `degmat_kron` product

Given two monomials $Z_1(p) \in \mathbb{R}^{l_1}$ and $Z_2(p) \in \mathbb{R}^{l_2}$, a commonly required operation is to find $(I_{l_2 p} \otimes Z_1(p))(I_p \otimes Z_2(p))$ which is equivalently written as

$$\begin{aligned}(I_{l_2 p} \otimes Z_1(p))(I_p \otimes Z_2(p)) &= \begin{bmatrix} Z_1(p) & & \\ & \ddots & \\ & & Z_1(p) \end{bmatrix}_{l_1 l_2 p \times l_2 p} \begin{bmatrix} Z_2(p) & & \\ & \ddots & \\ & & Z_2(p) \end{bmatrix}_{l_2 p \times p} \\ &= \begin{bmatrix} Z_1(p) \otimes Z_2(p) & & \\ & \ddots & \\ & & Z_1(p) \otimes Z_2(p) \end{bmatrix}_{l_1 l_2 p \times p} \\ &= (I_p \otimes (Z_1(p) \otimes Z_2(p)))\end{aligned}$$

In MATLAB, each row of 'degmat' stores degrees of one monomial. Then

$$Z_i = \prod_{j=1}^{n_i} varname\{j\}^{degmat(:,j)}$$

where the product '$\prod$' and exponent are performed elementwise. When 'varname' is same for Z_1 and Z_2, the kronecker product can be performed by adding rows of degmat.

To perform kronecker product between 'dmat1' and 'dmat2',

1. Add every row of 'dmat2' to i^{th} row of 'dmat1' and stack them vertically as shown below.
2. $dmat_{new} = \begin{bmatrix} dmat_1(1,:) + dmat_2(:,:) \\ dmat_1(2,:) + dmat_2(:,:) \\ \vdots \\ dmat_1(end,:) + dmat_2(:,:) \end{bmatrix}$.
3. Then, find unique rows in dmatnew. `[dmat_uni, , IA] = unique(dmat_new,'rows','stable');`
4. Find a matrix $A_{combine}$ such that, `dmat_new = Acombine*dmat_uni;`.
5. Using MATLAB 'unique' function, the index of rows of `dmat_new` found in `dmat_uni` is stored in `IA`. Then `Acombine(i, IA(i)) = 1; % i goes from 1 to length(IA).`

In `mtimes`, step 2 of the above list is performed in `rowwise_sum` and step 5 in `combine_degmat`.

3.1.2 `dpvar2poly` Conversion of `dpvar` to `polynomial`

Syntax:

```
polyVar = dpvar2poly(dpVar); %returns a polnomial object
```

Inputs:

1. `dpVar` : A dpvar class object with fields shown in Section 2.2

Outputs:

1. `polyVar` : A polynomial class object with fields shown in Section 2.1

Let D be a `dpvar` class object of the form shown below. We split the coefficient matrix of the `dpvar` into block matrices, where each $C_{ij} \in \mathbb{R}^{(d+1)\times 1}$, as

$$
\begin{aligned}
D(p;d) = (I_m \otimes Z(d))^T C (I_n \otimes Z(p)) &= \begin{bmatrix} Z(d)^T & & \\ & \ddots & \\ & & Z(d)^T \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1,zn} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,zn} \end{bmatrix} (I_n \otimes Z(p)) \\
&= \begin{bmatrix} Z(d)^T C_{11} & Z(d)^T C_{12} & \cdots & Z(d)^T C_{1,zn} \\ \vdots & \vdots & \ddots & \vdots \\ Z(d)^T C_{m,1} & Z(d)^T C_{m,2} & \cdots & Z(d)^T C_{m,zn} \end{bmatrix} (I_n \otimes Z(p)) \\
&= \begin{bmatrix} C_{11}^T Z(d) & C_{12}^T Z(d) & \cdots & C_{1,zn}^T Z(d) \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1}^T Z(d) & C_{m,2}^T Z(d) & \cdots & C_{m,zn}^T Z(d) \end{bmatrix} (I_n \otimes Z(p)) \\
&= \begin{bmatrix} C_{11}^T & C_{12}^T & \cdots & C_{1,zn}^T \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1}^T & C_{m,2}^T & \cdots & C_{m,zn}^T \end{bmatrix} (I_{zn} \otimes Z(d))(I_n \otimes Z(p)) = C_{new}(I_n \otimes Z_{new}(p,d)).
\end{aligned}
$$

To obtain the `polynomial` object, construct the degmat (combine not needed since the intersection of the sets p and d is a null set) and then rearrange C_{new} into the shape $z(d+1) \times mn$.

To rearrange C_{new}, construct a block matrix structure

$$
C_{new} = \begin{bmatrix} C_{1,1} & \cdots & C_{1,n} \\ \vdots & \ddots & \vdots \\ C_{m,1} & \cdots & C_{m,n} \end{bmatrix}
$$

where $C_{ij} \in \mathbb{R}^{1 \times z(d+1)}$. Then, set

$$
C_{new} = \begin{bmatrix} C_{1,1} \\ \vdots \\ C_{m,1} \\ C_{1,2} \\ \vdots \\ C_{m,2} \\ \vdots \\ C_{m,n} \end{bmatrix}.
$$

The required coefficient for `polynomial` is C_{new}^T.

3.1.3 Conversion of `polynomial` object to `dpvar`

Syntax:

```
dpVar = poly2dpvar(polyVar); %returns a dpvar object
```

Inputs:

1. `polyVar` : A polynomial class object with fields shown in Section 2.1

Outputs:

1. `dpVar` : A dpvar class object with fields shown in Section 2.2

Suppose $P(p;d)$ is a polynomial of dimension $m \times n$ with decision variables d and polynomial variables p. Assuming d have degrees either 0 or 1 exclusively **and** there are no products of decision variables in d, we can convert P into a `dpvar` as shown below.

1. Let $P(p;d).coeff = B$, $P(p;d).matdim = [m,n]$ and $P(p;d).degmat = D = Z(p;d)$.
2. First, reshape B using `Bshape` to get C. $C = Bshape(B, P(p;d).matdim, length(D))$.
3. Then $P(p;d) = C(I_n \otimes Z(p;d))$.
4. Split $(I_n \otimes Z(p;d)) = E(I_{zn} \otimes Z(d))(I_n \otimes Z(p))$ where E is some row permutation matrix. E can be found using intersect or ismember function in Matlab.
5. Then

$$
\begin{aligned}
P(p;d) = C(I_n \otimes Z(p;d)) &= (C \otimes E)(I_{zn} \otimes Z(d))(I_n \otimes Z(p)) \\
&= \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1,zn} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1} & C_{m,2} & \cdots & C_{m,zn} \end{bmatrix} (I_{zn} \otimes Z(d))(I_n \otimes Z(p)) \\
&= \begin{bmatrix} C_{11}Z(d) & C_{12}Z(d) & \cdots & C_{1,zn}Z(d) \\ \vdots & \vdots & \ddots & \vdots \\ C_{m,1}Z(d) & C_{m,2}Z(d) & \cdots & C_{m,zn}Z(d) \end{bmatrix} (I_n \otimes Z(p)) \\
&= \begin{bmatrix} Z(d)^T C_{11}^T & Z(d)^T C_{12}^T & \cdots & Z(d)^T C_{1,zn}^T \\ \vdots & \vdots & \ddots & \vdots \\ Z(d)^T C_{m,1}^T & Z(d)^T C_{m,2}^T & \cdots & Z(d)^T C_{m,zn}^T \end{bmatrix} (I_n \otimes Z(p)) \quad = (I_m \otimes Z(d)^T) C_{new} (I_n \otimes Z(p)).
\end{aligned}
$$

3.1.4 `Bshape` Reshape a pvar to compatible format

Syntax:

```
B = Bshape(polyVar); %returns a sparse matrix
```

Inputs:

1. `polyVar` : A polynomial class object with fields shown in Section 2.1

Outputs:

1. `B` : A sparse matrix of coefficients built reshaped as described below

Recall that, for a matrix pvar of size $n \times p$, the coefficient matrix is structured as

$$
\begin{aligned}
&\begin{bmatrix} b_1 & \cdots b_n & \cdots & b_{n(p-1)+1} & \cdots & b_{np} \end{bmatrix} \\
&= \begin{bmatrix} b_{1,1} & \cdots b_{n,1} & b_{1,2} & \cdots & b_{n,2} & \cdots & b_{1,p} & \cdots & b_{n,p} \end{bmatrix}
\end{aligned}
$$

If $Z(p)$ is the monomials corresponding to `degmat`, then the polynomial itself is

$$P(p) = \overbrace{\begin{bmatrix} b_1^T & \cdots & b_{n(p-1)+1}^T \\ b_2^T & \cdots & \\ \vdots & & \vdots \\ b_n^T & \cdots & b_{np}^T \end{bmatrix}}^{\hat{B}} (I_p \otimes Z(p))$$

Alternatively,

$$P(p) = (I_n \otimes Z(p))^T \overbrace{\begin{bmatrix} b_1 & \cdots & b_{n(p-1)+1} \\ b_2 & \cdots & \\ \vdots & & \vdots \\ b_n & \cdots & b_{np} \end{bmatrix}}^{\hat{B}}$$

3.1.5 `common_bases`

Syntax:

```
[E,F] = common_bases(E,F); %dpvars E and F with same monomials and varnames
```

Inputs:

1. `E, F` : Two dpvar class objects with (possibly) different monomials, dvarnames, and varnames

Outputs:

1. `E, F` : Same two dpvar class objects with (now unified) same monomials, dvarnames, and varnames

Given two dpvar objects A and B of same matrix dimensions, where

$$A = \left((I_m \otimes Z_{dA}(d_A)^T) C_A (I_n \otimes Z_{pA}(p_A))\right),$$
$$B = \left((I_m \otimes Z_{dB}(d_B)^T) C_B (I_B \otimes Z_{pB}(p_B))\right),$$

'DPcommon_bases' is frequently used in many binary operations (such as addition, concatenation, etc.) on A and B.

This function is used to find a common monomial sets Z_d and Z_p in common variables d_p and p_p such that

$$A = \left((I_m \otimes Z_d(d_p)^T) C_{pA} (I_n \otimes Z_p(p_p))\right),$$
$$B = \left((I_m \otimes Z_d(d_p)^T) C_{pB} (I_n \otimes Z_p(p_p))\right).$$

The common basis sets are obtained by following the steps shown below.

1. Find the union $d_p = d_A \cup d_B$ and $p_p = p_A \cup p_B$ using matlab union function.

   Note: the $Z_d(d_p) = \begin{bmatrix} 0_{1\times nt} \\ I_{nt\times nt} \end{bmatrix}$ where nt is size of the set d_p.

2. Find $Z_p(p_p)$: first add zero columns to degmat Z_{pA} (and Z_{pB}) corresponding to variables in set $p_B - p_A$ (set $p_A - p_B$ for Z_{pB}).

3. Rearrange columns of extended degmats of Z_{pA} and Z_{pB} to have variable names in the same order.

4. Find row-wise union on extended degmats using matlab function to get degmat corresponding to $Z_p(p_p)$.

5. Extend C_A by introducing zero rows/columns corresponding to missing variables in set $d_p - d_A / p_p - p_A$ to obtain C_{pA}.

6. Repeat previous step for C_B.

3.1.6 `compress`

Syntax:

```
E = compress(E); %returns a dpvar object with minimal monomial and varname set
```

Inputs:

1. `E` : A dpvar class object

Outputs:

1. `E` : Same dpvar class object with reduced varnames, dvarnames, and monomials

This function is used to reduce the size of a dpvar degmat by eliminating rows and columns with all zero coefficients. This is performed in three stages listed below.

1. First identify all zero columns in `degmat`. Remove the any such column and the variable corresponding to that column number in `varname`.
2. Remove all zero rows in `degmat`, then remove all the columns in `C` matrix corresponding to the discarded row numbers in `degmat`. Note each i, j-block matrix in `C` has a column corresponding to discarded rows.
3. Find `dvarnames` with all zero coefficients in `C` matrix and remove rows corresponding to those `dvarnames`. Reduce the `dvarname` set.

3.2 `mtimes` Multiplication by pvar matrix

Syntax:

```
A*B %returns product of a dpvar and polynomial object
```

Let

$$D(p;d) * P(p) = (I_m \otimes Z(d))^T C(I_n \otimes Z_1(p)) * B(I_p \otimes Z_2(p))$$

where $Z(d) \in \mathbb{R}^{l_d}$, $Z_i(p) \in \mathbb{R}^{l_i}$, $P(p) \in \mathbb{R}^{n \times p}$ and $D(p;d) \in \mathbb{R}^{m \times n}$. Then $C \in \mathbb{R}^{l_d m \times l_1 n}$ and $B \in \mathbb{R}^{n \times l_2 p}$. First, split C into $l_d m \times n$ block matrix and B into $n \times l_2 p$ as

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{l_d m,1} & C_{l_d m,2} & \cdots & C_{l_d m,n} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & \cdots & B_{1,l_2 p} \\ \vdots & \ddots & \vdots \\ B_{n,1} & \cdots & B_{n,l_2 p} \end{bmatrix}.$$

Then,

$$C(Z_1(p) \otimes I_n) = \begin{bmatrix} C_{11}Z_1(p) & C_{12}Z_1(p) & \cdots & C_{1n}Z_1(p) \\ \vdots & \vdots & \ddots & \vdots \\ C_{l_d m,1}Z_1(p) & C_{l_d m,2}Z_1(p) & \cdots & C_{l_d m,n}Z_1(p) \end{bmatrix}.$$

Therefore,

$$\begin{aligned} D(p;d) * P(p) &= (I_m \otimes Z(d))^T C(I_n \otimes Z_1(p)) * B(I_p \otimes Z_2(p)) \\ &= (I_m \otimes Z(d))^T \begin{bmatrix} C_{11}Z_1(p) & C_{12}Z_1(p) & \cdots & C_{1n}Z_1(p) \\ \vdots & \vdots & \ddots & \vdots \\ C_{l_d m,1}Z_1(p) & C_{l_d m,2}Z_1(p) & \cdots & C_{l_d m,n}Z_1(p) \end{bmatrix} \begin{bmatrix} B_{11} & \cdots & B_{1,l_2 p} \\ \vdots & \ddots & \vdots \\ B_{n,1} & \cdots & B_{n,l_2 p} \end{bmatrix} (I_p \otimes Z_2(p)) \end{aligned}$$

$$= (I_m \otimes Z(d))^T \begin{bmatrix} (\sum_{i=1}^n C_{1i}B_{i,1})Z_1(p) & \cdots & (\sum_{i=1}^n C_{1,i}B_{i,l_2p})Z_1(p) \\ \vdots & \ddots & \vdots \\ (\sum_{i=1}^n C_{l_dm,i}B_{i,1})Z_1(p) & \cdots & (\sum_{i=1}^n C_{l_dm,i}B_{i,l_2p})Z_1(p) \end{bmatrix} (I_p \otimes Z_2(p))$$
$$= (I_m \otimes Z(d))^T C_{new}(I_{l_2p} \otimes Z_1(p))(I_p \otimes Z_2(p))$$

where

$$C_{new} = \begin{bmatrix} (\sum_{i=1}^n C_{1i}B_{i,1}) & \cdots & (\sum_{i=1}^n C_{1,i}B_{i,l_2p}) \\ \vdots & \ddots & \vdots \\ (\sum_{i=1}^n C_{l_dm,i}B_{i,1}) & \cdots & (\sum_{i=1}^n C_{l_dm,i}B_{i,l_2p}) \end{bmatrix}.$$

3.3 `subs` Substitution

Syntax:

```
A_sub = subs(A, vars, vars_b); %returns dpvar A with vars substituted by vars_b
```

Inputs:

1. `A` : A dpvar class object
2. `vars` : An array of polynomial variables to be substituted
3. `vars_b` : An array of polynomial variables (or constants) to be substituted (must be of same length as `vars`)

Outputs:

1. `A_sub` : A dpvar class object (same as `A` but `vars` replaced by `vars_b`)

Substitution is valid only for independent variables. Define $subs(Z(p), p = a) = AZ_2(p)$. Then

$$\begin{aligned} subs(D(d;p), p = a) &= (I_m \otimes Z(d))^T C(I_n \otimes subs(Z(p), p = a)) \\ &= (I_m \otimes Z(d))^T C(I_n \otimes AZ_2(p)) \\ &= (I_m \otimes Z(d))^T C(I_n \otimes A)(I_n \otimes Z_2(p)). \end{aligned}$$

If `S=subs(Z,p=a)`, then `A=S.coeff'`.

3.4 `transpose` Transpose of `dpvar`

Syntax:

```
A' %returns transpose of A, also use A.'
```

Transpose of a matrix valued `dpvar`, with monomials $Z_1(d) \in \mathbb{R}^{l_d}$ and $Z_2(p) \in \mathbb{R}^{l_p}$ is computed using the following formulae.

$$\begin{aligned} D(d;p)^T &= \left((I_m \otimes Z_1(d))^T C(I_n \otimes Z_2(p))\right)^T = (I_n \otimes Z_2(p))^T C^T (I_m \otimes Z_1(d))^T \\ &= (I_n \otimes Z_2(p))^T D(I_m \otimes Z_1(d)) \\ &= \begin{bmatrix} Z_2(p)^T D_{11} Z_1(d) & Z_2(p)^T D_{12} Z_1(d) & \cdots & Z_2(p)^T D_{1,m} Z_1(d) \\ \vdots & \vdots & \ddots & \vdots \\ Z_2(p)^T D_{n,1} Z_1(d) & Z_2(p)^T D_{n,2} Z_1(d) & \cdots & Z_2(p)^T D_{n,m} Z_1(d) \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} Z_1(d)^T D_{11}^T Z_2(p) & Z_1(d)^T D_{12}^T Z_2(p) & \cdots & Z_1(d)^T D_{1,m}^T Z_2(p) \\ \vdots & \vdots & \ddots & \vdots \\ Z_1(d)^T D_{n,1}^T Z_2(p) & Z_1(d)^T D_{n,2}^T Z_2(p) & \cdots & Z_1(d)^T D_{n,m}^T Z_2(p) \end{bmatrix}$$
$$= (I_n \otimes Z_1(d))^T \begin{bmatrix} D_{11}^T & D_{12}^T & \cdots & D_{1,m}^T \\ \vdots & \vdots & \ddots & \vdots \\ D_{n,1}^T & D_{n,2}^T & \cdots & D_{n,m}^T \end{bmatrix} (I_m \otimes Z_2(p))$$

where $D_{ij} \in \mathbb{R}^{l_p, l_d}$ is a block matrix and $Z_2(p)^T D_{ij} Z_1(d)$ is has a scalar dimension for all i, j.

3.5 `int` Integral of `dpvar`

Syntax:

```
Aint = int(A,var,LL,UL); %returns definite integral of A w.r.t.  var from LL to UL
```

Inputs:

1. `A` : A dpvar class object
2. `var` : A polynomial variable, the variable of integration
3. `LL, UL` : Real numbers (or scalar polynomial variables) corresponding to lower and upper limits of the integral

Outputs:

1. `Aint` : A dpvar class object

   Integration is for only independent variables.

$$\begin{aligned} int(Z(p), p, a, b) &= (I_m \otimes Z(d))^T C(I_n \otimes int(Z(p), p, a, b)) \\ &= (I_m \otimes Z(d))^T C(I_n \otimes AZ_2(p)) \\ &= (I_m \otimes Z(d))^T C(I_n \otimes A)(I_n \otimes Z_2(p)). \end{aligned}$$

   If `S=int(Z,p,a,b)`, then `A=S.coeff'`.

3.6 `plus` for dpvar objects

Syntax:

```
A+B %returns sum of dpvar objects
```

Given two dpvar objects A and B of same matrix dimensions, where

$$A = \left((I_m \otimes Z_{dA}(d_A)^T) C_A (I_n \otimes Z_{pA}(p_A))\right),$$
$$B = \left((I_m \otimes Z_{dB}(d_B)^T) C_B (I_B \otimes Z_{pB}(p_B))\right),$$

addition is performed as described by the following steps.

1. Find the unions $d_p = d_A \cup d_B$ and $p_p = p_A \cup p_B$.
2. Find the monomial set $Z_d(d_p)$ such that $Z_d = Z_{dA}(d_A) \cup Z_{pB}(d_B)$.
3. Similarly, find $Z_p(p_p) = Z_{pA}(p_A) \cup Z_{pB}(p_B)$.
4. Introduce zero rows and columns in C_A and C_B, respectively, such that

$$A = \left((I_m \otimes Z_d(d_p)^T) C_{pA} (I_n \otimes Z_p(p_p))\right),$$
$$B = \left((I_m \otimes Z_d(d_p)^T) C_{pB} (I_n \otimes Z_p(p_p))\right).$$

5. Then required addition of dpvars A and B is simply given by

$$A + B = \left((I_m \otimes Z_d(d_p)^T)(C_{pA} + C_{pB})(I_n \otimes Z_p(p_p))\right).$$

3.7 `horzcat` and `vertcat`

Syntax:

```
[A,B]%returns horizontal concatenation of dpvar objects
[A;B]%returns vertical concatenation of dpvar objects
```

Given two dpvar objects A and B with compatible dimensions, where

$$A = \left((I_m \otimes Z_{dA}(d_A))^T C_A (I_n \otimes Z_{pA}(p_A))\right),$$
$$B = \left((I_m \otimes Z_{dB}(d_B))^T C_B (I_B \otimes Z_{pB}(p_B))\right),$$

concatenation is performed as described by the following steps.

1. Repeat steps 1-4 from previous section
2. Then required horizontal concatenation of dpvars A and B is simply given by

$$\begin{bmatrix} A & B \end{bmatrix} = \left((I_m \otimes Z_d(d_p)^T) \begin{bmatrix} C_{pA} & C_{pB} \end{bmatrix} (I_{2n} \otimes Z_p(p_p))\right).$$

3. Similarly, vertical concatenation is given by

$$\begin{bmatrix} A \\ B \end{bmatrix} = \left((I_{2m} \otimes Z_d(d_p)^T) \begin{bmatrix} C_{pA} \\ C_{pB} \end{bmatrix} (I_n \otimes Z_p(p_p))\right).$$

3.8 `subsref/subsasgn`

`subsref` is used to access rows and columns of a dpvar A using subscripts. For example, i to k rows and j to l columns of A are accessed by using the command $A(i:k, j:l)$.
Syntax:

```
A(rows,cols) %returns rows and cols of A specified by the indices
A.prop %returns the specified property 'prop' of A
```

Since `subsref` indexing only deals with coefficients (degmats, dvarnames and varnames potentially remain the same), first generate indices of C, that correspond to rows and columns specified by the subscripts. The formula is given by

$$idxrows = (i-1)*(nd+1)+1 : k*(nd+1)$$
$$idxcols = (j-1)*(ndeg)+1 : l*(ndeg)$$

where nd is number of dvarnames and $ndeg$ is number of rows of degmat. Then required rows and columns are extracted from C matrix using the command $C(idxrows, idxcols)$.
Syntax:

```
A(rows,cols) =B; %reassigns rows and cols of A specified by the indices by B
A.prop = B; %reassigns the specified property 'prop' of A by B
```

`subsasgn` works in the same way as `subsref` except at the last stage where instead of extracting the values from $C(idxrows, idxcols)$, new values are assigned at specified index locations.

3.9 `jacobian`

SYNTAX:

```
J = jacobian(A);
```

Inputs:

1. `A` : A dpvar class object

Outputs:

1. `J` : A dpvar class object that represents the jacobian of `A`

This function will be used to find Jacobian matrix from a `dpvar` object.
For $p_d \subseteq p$, jacobian is found using

$$\begin{aligned} J(,A(p;d),p_d) &= \big((I_m \otimes Z(d)^T)C_A(I_n \otimes J(Z_p(p),p_d))\big) \\ &= \big((I_m \otimes Z(d)^T)C_A(I_d \otimes DZ_p(p))\big) = \big((I_m \otimes Z(d)^T)(C_A(I_d \otimes D))(I_d \otimes Z_p(p))\big) \end{aligned}$$

for some differentiation matrix D.

NOTE: A minor variation of the jacobian is the `diff()` function which takes `vars` as an input whereas Jacobian is calculated by using all varnames.

SYNTAX:

```
J = diff(A, vars);
```

Inputs:

1. `A` : A dpvar class object

2. `vars` : polynomial variables with respect to which the differentiation is performed

Outputs:

1. `J` : A dpvar class object that represents the partial differentiation of `A` with respect to `vars`