# NAG Library Function Document

# nag_2d_spline_ts_eval (e02jec)

## 1 Purpose

nag_2d_spline_ts_eval (e02jec) calculates a vector of values of a spline computed by nag_2d_spline_fit_ts_scat (e02jdc).

## 2 Specification

```
#include <nag.h>
#include <nage02.h>
void nag_2d_spline_ts_eval (Integer nevalv, const double xevalv[],
     const double yevalv[], const double coefs[], double fevalv[],
     const Integer iopts[], const double opts[], NagError *fail)
```

## 3 Description

nag_2d_spline_ts_eval (e02jec) calculates values at prescribed points $(x_i,y_i)$, for $i = 1,2,\ldots,n$, of a bivariate spline computed by nag_2d_spline_fit_ts_scat (e02jdc). It is derived from the TSFIT package of O. Davydov and F. Zeilfelder.

## 4 References

Davydov O, Morandi R and Sestini A (2006) Local hybrid approximation for scattered data fitting with bivariate splines *Comput. Aided Geom. Design* **23** 703–721

Davydov O, Sestini A and Morandi R (2005) Local RBF approximation for scattered data fitting with bivariate splines *Trends and Applications in Constructive Approximation* M. G. de Bruin, D. H. Mache, and J. Szabados, Eds **ISNM Vol. 151** Birkhauser 91–102

Davydov O and Zeilfelder F (2004) Scattered data fitting by direct extension of local polynomials to bivariate splines *Advances in Comp. Math.* **21** 223–271

Farin G and Hansford D (2000) *The Essentials of CAGD* Natic, MA: A K Peters, Ltd.

## 5 Arguments

1: **nevalv** – Integer *Input*

*On entry*: $n$, the number of values at which the spline is to be evaluated.

*Constraint*: **nevalv** $\geq 1$.

2: **xevalv**[**nevalv**] – const double *Input*

*On entry*: the $(x_i)$ values at which the spline is to be evaluated.

*Constraint*: for all $i$, **xevalv**$[i-1]$ must lie inside, or on the boundary of, the spline's bounding box as determined by nag_2d_spline_fit_ts_scat (e02jdc).

3: **yevalv**[**nevalv**] – const double *Input*

*On entry*: the $(y_i)$ values at which the spline is to be evaluated.

*Constraint*: for all $i$, **yevalv**$[i-1]$ must lie inside, or on the boundary of, the spline's bounding box as determined by nag_2d_spline_fit_ts_scat (e02jdc).

4: **coefs**[$dim$] – const double *Communication Array*

**Note**: the dimension, $dim$, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument **coefs** in the previous call to nag_2d_spline_fit_ts_scat (e02jdc).

*On entry*: the computed spline coefficients as output from nag_2d_spline_fit_ts_scat (e02jdc).

5: **fevalv**[**nevalv**] – double *Output*

*On exit*: if **fail**.**code** = NE_NOERROR on exit **fevalv**[$i-1$] contains the computed spline value at $(x_i, y_i)$.

6: **iopts**[$dim$] – const Integer *Communication Array*

**Note**: the dimension, $dim$, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument **iopts** in the previous call to nag_fit_opt_set (e02zkc).

*On entry*: the contents of the array MUST NOT have been modified either directly or indirectly, by a call to nag_fit_opt_set (e02zkc), between calls to nag_2d_spline_fit_ts_scat (e02jdc) and nag_2d_spline_ts_eval (e02jec).

7: **opts**[$dim$] – const double *Communication Array*

**Note**: the dimension, $dim$, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument **opts** in the previous call to nag_fit_opt_set (e02zkc).

*On entry*: the contents of the array MUST NOT have been modified either directly or indirectly, by a call to nag_fit_opt_set (e02zkc), between calls to nag_2d_spline_fit_ts_scat (e02jdc) and nag_2d_spline_ts_eval (e02jec).

8: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_INITIALIZATION**

Option arrays are not initialized or are corrupted.

**NE_INT**

On entry, **nevalv** = ⟨*value*⟩.
Constraint: **nevalv** $\geq 1$.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

**NE_INVALID_SPLINE**

The fitting routine has not been called, or the array of coefficients has been corrupted.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

**NE_POINT_OUTSIDE_RECT**

On entry, **xevalv**$[\langle value \rangle] = \langle value \rangle$ was outside the bounding box.
Constraint: $\langle value \rangle \leq$ **xevalv**$[i-1] \leq \langle value \rangle$ for all $i$.

On entry, **yevalv**$[\langle value \rangle] = \langle value \rangle$ was outside the bounding box.
Constraint: $\langle value \rangle \leq$ **yevalv**$[i-1] \leq \langle value \rangle$ for all $i$.

## 7 Accuracy

nag_2d_spline_ts_eval (e02jec) uses the de Casteljau algorithm and thus is numerically stable. See Farin and Hansford (2000) for details.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

To evaluate a $C^1$ approximation (i.e., when **Global Smoothing Level** $= 1$), a real array of length $O(1)$ is dynamically allocated by each invocation of nag_2d_spline_ts_eval (e02jec). No memory is allocated internally when evaluating a $C^2$ approximation.

## 10 Example

See Section 10 in nag_2d_spline_fit_ts_scat (e02jdc).