# NAG Library Function Document

# nag_ode_bvp_ps_lin_grid_vals (d02uwc)

## 1 Purpose

nag_ode_bvp_ps_lin_grid_vals (d02uwc) interpolates from a set of function values on a supplied grid onto a set of values for a uniform grid on the same range. The interpolation is performed using barycentric Lagrange interpolation. nag_ode_bvp_ps_lin_grid_vals (d02uwc) is primarily a utility function to map a set of function values specified on a Chebyshev Gauss–Lobatto grid onto a uniform grid.

## 2 Specification

```
#include <nag.h>
#include <nagd02.h>
```

```
void nag_ode_bvp_ps_lin_grid_vals (Integer n, Integer nip, const double x[],
     const double f[], double xip[], double fip[], NagError *fail)
```

## 3 Description

nag_ode_bvp_ps_lin_grid_vals (d02uwc) interpolates from a set of $n+1$ function values, $f(x_i)$, on a supplied grid, $x_i$, for $i=0,1,\ldots,n$, onto a set of $m$ values, $\hat{f}(\hat{x}_j)$, on a uniform grid, $\hat{x}_j$, for $j=1,2,\ldots,m$. The image $\hat{x}$ has the same range as $x$, so that $\hat{x}_j = x_{\text{min}} + ((j-1)/(m-1)) \times (x_{\text{max}} - x_{\text{min}})$, for $j=1,2,\ldots,m$. The interpolation is performed using barycentric Lagrange interpolation as described in Berrut and Trefethen (2004).

nag_ode_bvp_ps_lin_grid_vals (d02uwc) is primarily a utility function to map a set of function values specified on a Chebyshev Gauss–Lobatto grid computed by nag_ode_bvp_ps_lin_cgl_grid (d02ucc) onto an evenly-spaced grid with the same range as the original grid.

## 4 References

Berrut J P and Trefethen L N (2004) Barycentric lagrange interpolation *SIAM Rev.* **46(3)** 501–517

## 5 Arguments

1: **n** – Integer *Input*

*On entry*: $n$, where the number of grid points for the input data is $n+1$.

*Constraint*: **n** $> 0$ and **n** is even.

2: **nip** – Integer *Input*

*On entry*: the number, $m$, of grid points in the uniform mesh $\hat{x}$ onto which function values are interpolated. If **nip** $= 1$ then on successful exit from nag_ode_bvp_ps_lin_grid_vals (d02uwc), **fip**[0] will contain the value $f(x_n)$.

*Constraint*: **nip** $> 0$.

3: **x**[**n** + **1**] – const double *Input*

*On entry*: the grid points, $x_i$, for $i=0,1,\ldots,n$, at which the function is specified.

Usually this should be the array of Chebyshev Gauss–Lobatto points returned in nag_ode_bvp_ps_lin_cgl_grid (d02ucc).

4: **f**[**n** + **1**] – const double *Input*

*On entry*: the function values, $f(x_i)$, for $i = 0, 1, \ldots, n$.

5: **xip**[**nip**] – double *Output*

*On exit*: the evenly-spaced grid points, $\hat{x}_j$, for $j = 1, 2, \ldots, m$.

6: **fip**[**nip**] – double *Output*

*On exit*: the set of interpolated values $\hat{f}(\hat{x}_j)$, for $j = 1, 2, \ldots, m$. Here $\hat{f}(\hat{x}_j) \approx f(x = \hat{x}_j)$.

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

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

## 6 Error Indicators and Warnings

**NE_BAD_PARAM**

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

**NE_INT**

On entry, **n** = ⟨*value*⟩.
Constraint: **n** > 0.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** is even.

On entry, **nip** = ⟨*value*⟩.
Constraint: **nip** > 0.

**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.

## 7 Accuracy

nag_ode_bvp_ps_lin_grid_vals (d02uwc) is intended, primarily, for use with Chebyshev Gauss–Lobatto input grids. For such input grids and for well-behaved functions (no discontinuities, peaks or cusps), the accuracy should be a small multiple of ***machine precision***.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

None.

## 10 Example

This example interpolates the function $x + \cos(5x)$, as specified on a 65-point Gauss–Lobatto grid on $[-1, 1]$, onto a coarse uniform grid.

## 10.1 Program Text

```
/* nag_ode_bvp_ps_lin_grid_vals (d02uwc) Example Program.
 *
 * Copyright 2011, Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagd02.h>
#include <nagx02.h>

#ifdef __cplusplus
extern "C" {
#endif
  static double NAG_CALL exact(double x);
#ifdef __cplusplus
}
#endif

int main(void)
{
  /*  Scalars */
  Integer     exit_status = 0;
  Integer     i, n, nip;
  double      a = -1.0, b = 1.0;
  double      uerr = 0.0;
  double      teneps = 10.0 * nag_machine_precision;
  /*  Arrays */
  double      *f = 0, *fip = 0, *x = 0, *xip = 0;
  /* NAG types */
  Nag_Boolean reqerr = Nag_FALSE;
  NagError    fail;

  INIT_FAIL(fail);

  printf("nag_ode_bvp_ps_lin_grid_vals (d02uwc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%"NAG_IFMT "%*[^\n] ", &n);
  scanf("%"NAG_IFMT "%*[^\n] ", &nip);
  if (
    !(f = NAG_ALLOC((n + 1), double)) ||
    !(fip = NAG_ALLOC((nip), double)) ||
    !(xip = NAG_ALLOC((nip), double)) ||
    !(x = NAG_ALLOC((n + 1), double))
    )
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Set up solution grid:
   * nag_ode_bvp_ps_lin_cgl_grid (d02ucc).
   * Generate Chebyshev Gauss-Lobatto grid.
   */
  nag_ode_bvp_ps_lin_cgl_grid(n, a, b, x, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ode_bvp_ps_lin_cgl_grid (d02ucc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Set up problem right hand sides for grid. */
  for (i = 0; i < n + 1; i++) f[i] = exact(x[i]);
```

```
  /* Map to an equally spaced grid:
   * nag_ode_bvp_ps_lin_grid_vals (d02uwc).
   * Interpolate a function from Chebyshev grid to uniform grid
   * using barycentric Lagrange interpolation.
   */
  nag_ode_bvp_ps_lin_grid_vals(n, nip, x, f, xip, fip, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ode_bvp_ps_lin_grid_vals (d02uwc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Print solution. */
  printf("Numerical solution f\n\n");
  printf("      x          f\n");
  for (i = 0; i < nip; i++) printf("%10.4f %10.4f\n", xip[i], fip[i]);

  if (reqerr) {
    for (i = 0; i < nip; i++) uerr = MAX(uerr, fabs(fip[i] - exact(xip[i])));
    printf("f is within a multiple %"NAG_IFMT " of machine precision.\n",
           10 * ((Integer) (uerr/teneps) + 1));
  }
 END:
  NAG_FREE(f);
  NAG_FREE(fip);
  NAG_FREE(x);
  NAG_FREE(xip);
  return exit_status;
}

static double NAG_CALL exact(double x)
{
  return x + cos(5.0 * x);
}
```

## 10.2 Program Data

```
nag_ode_bvp_ps_lin_grid_vals (d02uwc) Example Program Data
  64              : n
  17              : nip
```

## 10.3 Program Results

```
nag_ode_bvp_ps_lin_grid_vals (d02uwc) Example Program Results

Numerical solution f

      x          f
   -1.0000    -0.7163
   -0.8750    -1.2060
   -0.7500    -1.5706
   -0.6250    -1.6249
   -0.5000    -1.3011
   -0.3750    -0.6745
   -0.2500     0.0653
   -0.1250     0.6860
    0.0000     1.0000
    0.1250     0.9360
    0.2500     0.5653
    0.3750     0.0755
    0.5000    -0.3011
    0.6250    -0.3749
    0.7500    -0.0706
    0.8750     0.5440
    1.0000     1.2837
```